RED DE ALERTAS DE EVENTOS DE SEGURIDAD BASADA EN UNA ARQUITECTURA DE HONEYNET EN IOT, CENTRALIZADA Y MONITOREADA POR UN SENSOR AUTO-CONFIGURABLE

Eric Emiro Escudero Almeida

Instituto Tecnológico Metropolitano
Facultad de Ingenierías
Medellín, Colombia
2021

RED DE ALERTAS DE EVENTOS DE SEGURIDAD BASADA EN UNA ARQUITECTURA DE HONEYNET EN IOT, CENTRALIZADA Y MONITOREADA POR UN SENSOR AUTO-CONFIGURABLE.

Eric Emiro Escudero Almeida

Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título de:
Magister en Seguridad Informática

Director (a):
Msc Héctor Fernando Vargas Montoya

Grupo de Investigación:
Ciencias de la Computación

Instituto Tecnológico Metropolitano
Facultad de Ingenierías
Medellín, Colombia
2021

Lema

“Tu día será bueno si te despiertas sabiendo que construirás un futuro mejor”.

Autor: Elon Musk

Agradecimientos

Inicialmente quiero agradecer a todas las personas que se encuentra detrás del telón de este proyecto.

Sin lugar a dudas al Msc. Héctor Fernando Vargas Montoya, mi director, tutor y amigo quien con su abundante paciencia se esforzó en brindar su gran conocimiento para culminar este proyecto.

Gracias a tres mujeres que son un motor en mi vida, Noris Almeida de Escudero mi madre que le debo la vida, quien es un gran ejemplo de lucha, porque aprendió a ser padre y madre esforzándose a buscar lo mejor para mí, Yury Alexandra Serna Urrego y Samantha Escudero Serna mi esposa e hija, gracias por su comprensión entendiendo tantas horas dedicadas a este sueño, donde no era posible dedicarles horas, sin embargo, siempre su felicidad y amor estuvieron allí para darme fortaleza en mis largas horas de desvelo.

Y por último y no menos importante, al más grande, a Jesucristo Dios manifestado en carne quien es el dueño de nuestras vidas, quien permite que nuestros sueños se cumplan según su voluntad, de manera especial gratitud infinita por permitir que pueda conservar el aliento de vida para culminar esta meta.

Resumen

El presente proyecto de grado se trabajó en la construcción de una arquitectura Honeynet, integrando dos componentes principales, servidor y sensor auto-configurable, ambos se diseñan y crean como prototipos para utilizarlos en las redes IoT, usando tecnologías disponibles en el mercado como lo son los dispositivos embebidos en IoT (Raspberry Pi), que conlleve a que cuando un atacante intente ingresar a la Honeynet, esta le responda de manera automática activando o no servicios que un atacante pueda ver atractivos (auto- configurando nuevos servicios), permitiendo a los oficiales de seguridad obtener mayor información y eficiencia en los resultados del aprendizaje.

Palabras clave: Auto-Configuración; Detección De Intrusos (IDS); Honeynet; Honeypot; Internet De Las Cosas (IOT); Sistema De Ataque Informático; Sistemas Embebidos.

Abstract

This degree project worked on the construction of a Honeynet architecture, integrating two main components, server and self-configurable sensor, both are designed and created as prototypes for IoT networks, using technologies available in the market such as devices embedded in IoT (raspberry PI). This allows that when an attacker tries to enter the Honeynet, it responds automatically by activating or inactivating services that an attacker may see attractive (auto-configuring new services), allowing security officials to obtain greater information and efficiency in learning outcomes.

Keywords: Self-configuration; Intrusion Detection System(IDS); Honeynet; Honeypot; Internet of things(IOT); Security Information and Event Management(SIEM); Embedded Systems.

Contenido

Pág.

Lista de figuras

Pág.

Lista de tablas

Pág.

Introducción

En el campo de las Tecnologías de la información y las comunicaciones (TIC) la digitalización de las señales, el poder creativo de la transmisión de datos, la telefonía móvil, entre otros ejemplos de avance tecnológico son síntomas positivos de la evolución y transformación del mundo, teniendo muy en claro el papel protagónico de los dispositivos embebidos y aparatos electrónicos que se enlazan con internet como base o plataforma de la mayoría de las tecnologías desarrolladas, muchos conceptos y paradigmas planteados en el pasado, literalmente se podría decir que están anidadas en el pasado, hoy la humanidad se enfrenta a nuevos retos y desafíos donde la innovación es uno de los fuertes pilares en el avance tecnológico. Internet de las cosas – IoT (Internet of thing, por sus siglas en inglés) es un juego de palabras que se podría decir que está de moda, ya no se espera el futuro si no que con esto se construye el futuro [1]. La innovación y la tecnología provocan impactos serios en el entorno, pero aun en el paisaje tecnológico existen raíces en el tema de seguridad que no se pueden olvidar, dichas raíces son objetivos inmutables en el ecosistema de internet de las cosas.

Los dispositivos embebidos (aquellos que cumplen una o pocas funciones de forma dedicada) y aparatos electrónicos han evolucionado a velocidades extremas en las últimas décadas, y hoy los fabricantes de circuitos integrados y componentes electrónicos cuentan con una amplia gama de prototipos que satisfacen necesidades en distintos ambientes, no se puede negar que IoT mantiene una estrecha relación con los dispositivos embebidos, ya que son programados para cumplir una función específica, donde el microcontrolador cuenta con un firmware o programa grabado que se ejecuta infinitamente, recibe un mensaje alimentado por un sensor, lo interpreta y se conecta directamente a un router o dispositivo central de procesamiento, este a su vez hace conexión a un servidor en el cloud computing (internet). Un script en el servidor ejecuta un Query a una base de datos proporcionando una respuesta específica en una rutina [2].

Ahora bien, actualmente existen varios protocolos de comunicación orientados a la comunicación entre máquinas (M2M) y los dispositivos con recursos limitados, entre ellos se destacan por MQTT (Message Queue Telemetry Transport), CoAP (Constrained Application Protocol) y AMQP (Advanced Messaging Queuing Protocol)" [3].

La seguridad en muchos dispositivos IoT puede ser un gran vacío para hacer frente a las diferentes amenazas en los sistemas, se debe enfocar el esfuerzo en optimizar un alto nivel de seguridad y privacidad acorde al servicio que presta cada dispositivo, varios pasos a tener en cuenta son:

- Autenticación.
- Control de acceso.

- Confidencialidad de los datos.
- Integridad de los datos.
- No repudio.
- Disponibilidad.
- Aumento de seguridad con protocolos de bajo consumo y procesamiento ligero.

Gran parte de los servicios del IOT se produce mediante redes inalámbricas de sensores, los dispositivos se interconectan entre sí, cuya misión es recopilar información en su entorno y transmitirla hacia un punto específico donde se almacena, analiza, procesa y en algunos casos su captura proporciona la toma de decisiones. Pero estos ataques son identificados en redes reales y posiblemente cuando ya ha pasado el ataque o éste está en curso, no son registros de posibles amenazas capturadas en ambientes de pruebas, ambientes señuelos.

Los ataques a las redes de computadores, redes de telecomunicaciones y demás redes informáticas, suponen un reto cada vez más grande, desde el punto de vista de la detección, identificación de ataques, y reducción de los riesgos asociados a éstos, es así como en las redes IoT se vienen presentando [4] múltiples amenazas en la capa de red, lo que hacen a los protocolos de enrutamientos más vulnerables que en una red de datos. En la medida que se tenga información de los diferentes ataques que se van ejecutando, se tendría más poder de decisión con respecto a las medidas preventivas y de control hacia posibles riesgos.

Parte de la seguridad en las redes IOT es buscar con prelación posibles riesgos; por lo cual, es importante lograr detectar el mayor número posible de ataques, pero que esta identificación se haga de forma controlada sería lo ideal, de allí que surge la idea de considerar las redes señuelos (Honeypot y Honeynet). La empresa de servicios y productos de seguridad Kasperky, identifica un número promedio de intentos de ataques a los sistemas señuelos (Honeypot) que ellos poseen para recolección de información, lo que supone un elemento importante en la detección de nuevas amenazas, de enero de 2017 a abril del mismo año, un promedio de 40.000 ataques se han identificado, sin embargo en el primer semestre del 2019 después de implementar más de 50 Honeypots en el mundo, [5] se detectó 105 millones de ataques en dispositivos IoT. Esto tendría un gran beneficio en la recolección, análisis y conclusión frente a las amenazas (nuevas y antiguas), lo que tiene éste proveedor es una gran herramienta para tomas de decisiones, afinar sus productos y desarrollar nuevos.

Un Honeypot traducido al español como “tarro de miel”, es un elemento tecnológico (hardware o software) capaz de detectar, recopilar y analizar información de ataques informáticos en forma de señuelo [6], por lo cual, los posibles atacantes ingresarían a un sistema que está en un ambiente productivo con sus servicios, pero éste no es real (se le hace creer al atacante que está en un ambiente real, pero no lo es). La Honeynet recopila información de seguridad sobre una red entera o interconectando varios Honeypots de manera simultánea, con ello lograr obtener más información de lo normal [7]. La información que es objeto de los Honeypots o Honeynets es analizada y sintetizada, obteniendo información relevante de ataques de seguridad en las redes y servicios, con el fin de potencializar los diferentes controles en las redes y servicios reales de las organizaciones.

El objetivo final de una Honeynet es recolectar todo tipo de información que los posibles atacantes van dejando a medida que ejecutan sus ataques sobre las redes, esto para adquirir conocimiento sobre las técnicas de ataque, dado que el sistema es un señuelo, no habrá daños ni eventos de seguridad sobre información real [8] que pueda afectar a la organización.

Gracias a este grupo de elementos tecnológicos llamados Honeynet y al análisis de la información que estos detectan, se puede dimensionar la magnitud del problema al que se enfrenta un sistema de información, dado que se puede analizar con más detalles todos los registros e información adquirida, proporcionando datos valiosos, que entre otros son:

- Tendencias de ataques.
- Servicios que se pretenden vulnerar o detección de vulnerabilidades tempranas.
- Países más activos en ciberataques
- Muestras de malware no identificadas por motores antivirus.
- Técnicas usadas por los atacantes.
- Distribuidores de malware.
- Equipos pertenecientes a Botnets.
- Centros de comando y control (C&C) que son usados en ataques dirigidos o malware.

El conjunto anterior reúne la mayoría de elementos propios para la obtención de información y aprendizaje desde el atacante [9], así, conocer sus movimientos y aprender de éstos en la fijación de controles, basándonos en dispositivos embebidos para poder construir mecanismos de respuesta ante incidentes de seguridad, esto es, estar preparados en las redes reales por si ocurre un evento de seguridad.

Los sistemas embebidos son dispositivos integrados en productos electrónicos compuestos por hardware y software que controlan una o varias funciones, capaces de interactuar con varios protocolos de comunicación, a diferencia de los computadores que están diseñados a cumplir una infinidad de tareas y por su alta capacidad de funciones representa mayor costo para el usuario. Los sistemas embebidos generalmente están orientados a minimizar los costos sin quitar su flexibilidad, y que a la hora de necesitar alguna modificación solo es cambiar algunas líneas de código al software, en vez de reemplazar todo el circuito integrado. Estos sistemas se encuentran frecuentemente bajo el concepto de IoT y son usados tanto en hogares, ciudades como en la industria, para fortalecer los procedimientos técnicos administrativos o recolectar información del ambiente a través de los sensores configurados, por ejemplo; el horno microondas, el ascensor, sistemas de riego en la agricultura, el equipo de audio, el despertador electrónico, paraderos de buses y otros dispositivos.
Así mismo, los dispositivos embebidos serán constructores fundamentales en el escenario de implementación de nuevas tecnologías, y un ejemplo importante en ese avance dentro de IoT, teniendo un enfoque principal de proporcionar una puerta de entrada y salida a gran cantidad de datos relevantes [10], que en la medida que se logre visualizar a tiempo éstos datos, la prevención en

términos de seguridad sería más efectiva, pero para ello los Honeynet podrían cumplir un papel importante en IoT.

Tener una Honeynet auto-configurable permitirá solucionar la problemática de no contar con información especializada de ataques que es recolectada desde los diferentes atacantes en un dispositivo IoT, así como establecer las estrategias y conocimientos de los atacantes, a medida que van explorando y van ejecutando ataques avanzados, permitirá activar o desactivar servicios dentro del IoT, con ello, obtener información cada vez más relevante y de posibles atacantes expertos. El dispositivo IoT no activara servicios en otros componentes tecnológicos (por lo que no es necesario que los servicios y protocolos estén activos todo el tiempo), todo se activaría dentro del mismo IoT, esto es, no se usará el IoT como activador central en otros elementos computacionales.

Dado lo anterior, el presente proyecto se ha enfocado en el cumplimiento de los siguientes objetivos que fueron aprobados:

Objetivo general:

“Proponer un método de alerta de eventos de seguridad basada en una arquitectura Honeynet monitoreada por un sensor IoT auto-configurable que permita identificar diferentes tipos de ataques”.

Objetivos específicos:

- Diseñar la red Honeynet en IoT funcional en la que se logre la captura y almacenamiento de logs o registros de auditoría, para el análisis e identificación de posibles ataques por un dispositivo IoT, logrando la activación de servicios informáticos en la medida que se ejecuten ataques.

- Diseñar un método de auto-configuración en un sensor IoT que permita, a través de script o un programa software, activar diferentes servicios de acuerdo con los ataques detectados en sus logs o registros de auditorías, con el fin de recolectar datos de posibles ataques.

- Establecer la capacidad de recolectar la información de ataques en la Honeynet con el sensor IoT auto-configurable, para demostrar la captura del evento de seguridad y generar las recomendaciones de seguridad hacia las redes reales.

1. Marco Teórico y Estado del Arte

1.1 Marco Teórico

1.1.1 Honeypot y Honeynet

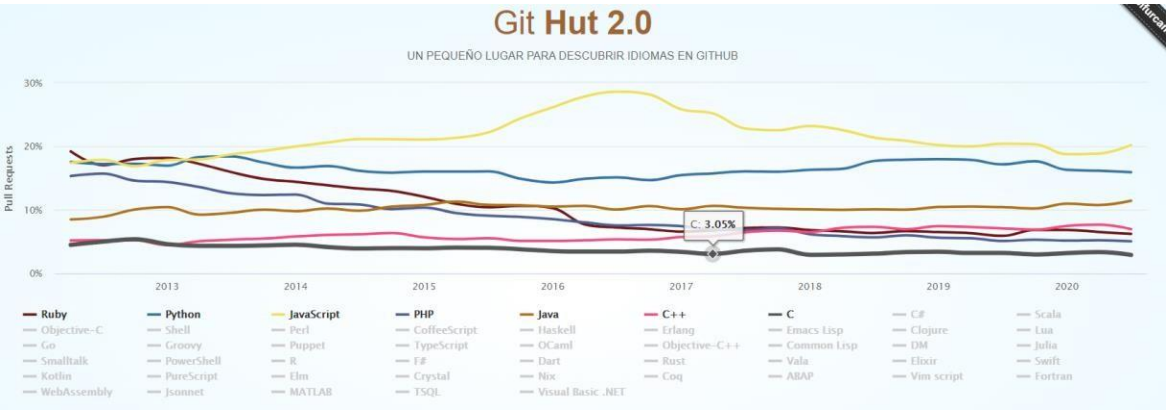
En el año 400 A.C. aparece este escenario donde el general chino Sun Tzu en su libro "El arte de la guerra" [21], utiliza una frase que 2.400 años más tarde fuera utilizada como un icono en la tecnología, "conoce a tu enemigo", bajo este lema existe una variedad incontable de escritos que direccionan a buscar herramientas, tácticas y razones de los atacantes.

Un Honeypot, es un elemento tecnológico (hardware y/o software) capaz de detectar, recopilar y analizar información de ataques informáticos en forma de señuelo, por lo cual, los posibles atacantes ingresarían a un sistema (cliente, servidor, sensor IoT o cualquier elemento computacional) que está en un ambiente productivo con sus servicios, pero éste no es real (se le hace creer al atacante que está en un ambiente real, pero no lo es). La Honeynet, similar a los Honeypot, tienen su objetivo en la recopilación de información de seguridad, pero sobre una red entera o interconectando varios Honeypot de manera simultánea, con ello lograr obtener más información de lo normal. La información que es objeto de los Honeypot o Honeynet es analizada y sintetizada, obteniendo información relevante de ataques de seguridad en las redes y servicios, con el fin de potencializar los diferentes controles en las redes y servicios reales de las organizaciones [22], los Honeypot que presentan adaptabilidad se denominan Honeypots dinámicos [11].

La terminología se empieza a utilizar formalmente en el año 2000, cuando un Honeypot instalado en un sistema operativo Solaris captura la primera huella de ataque en internet, conocida como la vulnerabilidad DTSPCD. Ese hecho fue contundente para demostrar que una herramienta como esta, se estaba convirtiendo en un potencial enorme, ya que se sabía de la vulnerabilidad mas no se conocía la forma del ataque [23].

Según Gartner [48] Python es uno de los lenguajes de programación más utilizados porque puede ser cinco veces más productivo que los lenguajes de propósito general como C++, Java y C#. Python lo tiene todo, es de código abierto, maneja librerías muy completas que permiten la invocación de variables en la gran mayoría de escenarios, están versátil que tiene integración con la inteligencia artificial gracias a librerías como Keras o TensorFlow, esto sin descuidar el hecho que genera utilidad para aplicaciones de Big Data, dando como resultado, este lenguaje de programación multiplataforma, multiparadigma orientado a objetos es imperativo, funcional y reflexivo también tiene su mirada hacia el desarrollo web gracias a sus frameworks Django o Flask. En la *Figura 1-1* se puede observar la tendencia de python al transcurrir de los años.

Figura 1-1: Participación de Python en los últimos años. [50]



Algo que si es claro es que las Honeypot y Honeynet no garantizan seguridad en la red, ni tampoco son mecanismos de defensa, más bien su participación está dirigida a ser parte integral de una gran herramienta tecnológica diseñada para obtener información que permita reforzar la seguridad o de forma paralela distraer como red señuelo a los atacantes.

Para las redes reales y en producción, un paradigma opuesto seria la implementación de señuelos o sistemas trampa, para que pudiesen ser atacados deliberadamente. Así se engaña a los atacantes y a su vez permite el aprendizaje del mismo.

La interacción comprende el grado de relación que puede tener el atacante VS el sistema, entre mayor sea la integración más será el nivel de riesgo de la red [7].

- Nivel de interacción bajo: Este tipo de Honeypot solo simula servicios como http, FTP, TELNET entre otros; se limita solo a escuchar peticiones, sin embargo, no deja de grabar en los logs las acciones del atacante.

- Nivel de interacción media: Este Honeypot compromete un poco más sus servicios buscando captar una mayor atención de su atacante, respondiendo a peticiones como un script que simula el comportamiento del puerto objetivo. Realmente en esta posición el Honeypot ya no es pasivo sino activo buscando una relación de proximidad con su atacante.

- Nivel de interacción alta: Este tipo de Honeypot realmente no simula servicio alguno, sino que contiene servicios totalmente reales, convirtiéndose en Honeynet que brindan una red totalmente convincente para el atacante.

No obstante, este nivel es el más peligroso ya que el atacante podría penetrar el sistema capturando la red trampa, destruyéndola y dirigiéndose a la red real, en el aspecto positivo en este nivel se adquiere mayor aprendizaje del atacante, ya que permite de cerca tener paso a paso y a profundidad, todos los métodos y estrategias utilizados por el visitante. [12].

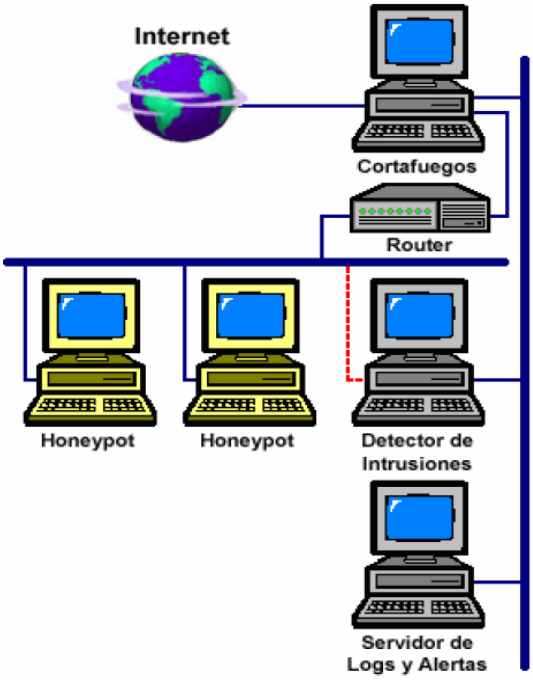
Arquitecturas Honeynets

Teniendo en cuenta lo dicho por FH Abbasi y RJ Harris en el proyecto Honeynet [47], donde esencialmente se presenta tres arquitecturas Honeynets de primera, segunda y tercera generación.

1.1.2 Honeynet de primera generación

Este modelo fue desarrollado en 1999 por Honeynet Project. Su construcción se basó en la implementación de un firewall de frente respaldado por un IDS, ubicando los Honeypots en la parte de atrás. Esta arquitectura esta segmentada por dos interfaces en la puerta de enlace, una frente a la red externa y la otra frente a la red interna de los Honeypots, este escenario se puede apreciar en la ***¡Error! No se encuentra el origen de la referencia.*** [47].

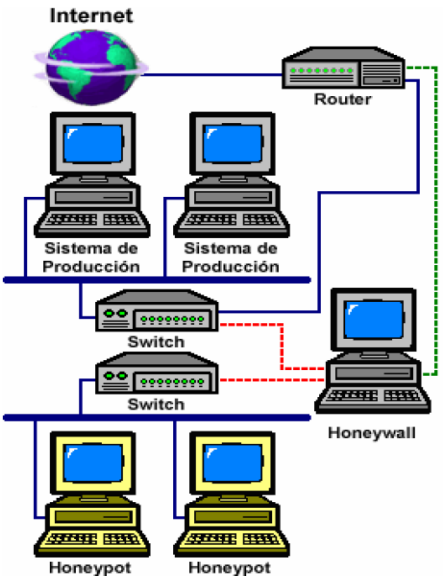
Figura 1-2: Honeynet de Primera Generación. [47]



1.1.3 Honeynet de segunda generación

En la *Figura 1-3*, se muestra la segunda generación que se destaca desde principios del año 2002, su esfuerzo se basa en que su entorno sea más difícil de identificar por el atacante, aunque relaciona su entorno con un ambiente productivo, mientras que la primera generación se relaciona con un ambiente de investigación [47]. La arquitectura es más sencilla debido a que la captura y recolección de datos es centralizada por un único sistema que el Honeynet Project denomina honeywall.

Figura 1-3: Honeynet de segunda generación. [47]

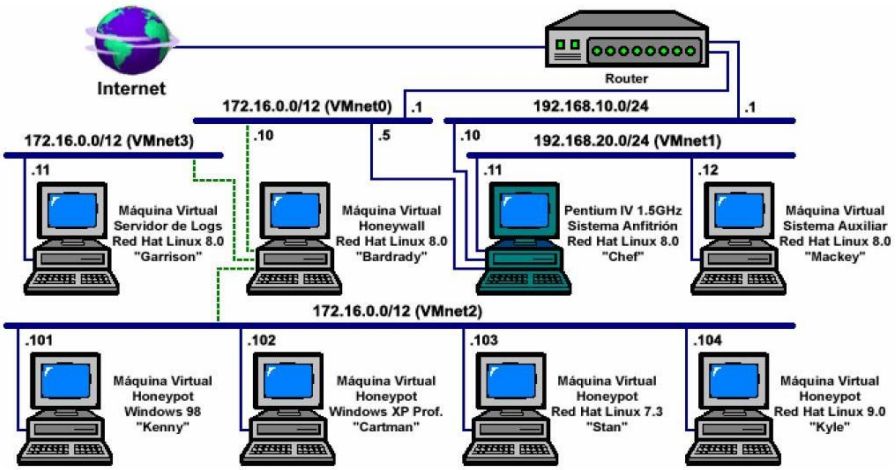


1.1.4 Honeynet de tercera generación

Su lanzamiento fue a finales del 2004. Aunque esta arquitectura es muy parecida a la generación II, se diferencia con la adición de un servidor Sebek integrado a la puerta de enlace, integrando 3 interfaces en honeywall [47]. Dos de ellas juegan un papel de puente entre la red externa y la red interna de Honeypot; y la tercera interface es utilizada en tareas de gestión y configuración. Esta tercera generación combina herramientas de virtualización como se puede validar en la

Figura 1-4.

Figura 1-4: Honeynet de tercera generación. [47]



En consecuencia y con base en lo anterior, se validaron las necesidades respectivas para la selección

de la Honeynet.

1.1.5 Honeynet de virtuales e hibridas

La existencia el mundo virtual ha permitido potencializar diferentes herramientas Honeypot y Honeynet para la exploración de diferentes mecanismos de obtener datos relevantes de ataques, en ese sentido, implementar soluciones desde la virtualidad da una ventaja en la implementación y monitoreo de los eventos de ciberseguridad y con ello, el potencial de tener diferentes sistemas de procesamiento de manera controlada. Éstas Honeynets hacen una combinación de las diferentes generaciones, adicionando elementos virtuales y elementos físicos, lo que permite una mejor opción a la hora de reducir recursos y potencializar más las opciones de seguridad, en consecuencia, las Honeynet híbridas dan mejor movilidad para su implementación, pues permite interconectar elementos físicos (como IoT) con elementos que deben ser virtualizados por su costo (como los sistemas SCADA), adicional a esto, las Honeynet hibridas pueden ser extendidas a otros sistemas y configuraciones [51][52][53].

- **Honeypot y Componentes en la arquitectura**

En este apartado se muestran componentes inmersos en la topología, ya que son parte fundamental en el funcionamiento de la arquitectura Honeynet.

- **HoneyPI**

Es un Honeypot de alta interacción de código abierto que se puede instalar en sistemas operativos Linux, teniendo un potente componente que funciona como IDS “sistema de detección de intrusos” llamado PSAD, que permite de forma temprana la detección de un comportamiento hostil por parte de uno o varios atacantes en el host, es posible personalizar las reglas de detección y como resultado ofrecer un protagonismo importante relacionado con el ambiente de seguridad en la red [32].

- **Pentbox**

Es una suite de seguridad orientada test de penetración [33] dirigido a sistemas operativos GNU/Linux y Windows, dicha suite incluye las siguientes funciones:

- Crackeador de Hash como MD5, SHA1, SHA256 y SHA512 mediante fuerza bruta numérica.
- Creador rápido de Honeypot.
- Generador de contraseñas seguras ante Ataque de fuerza bruta y Ataque de diccionario.
- Generadores de tráfico masivo en la red para probar posibles denegaciones de servicio.
- Escáner de puertos.

Pentbox como suite de seguridad presenta un amplio portafolio de servicios de test de penetración, esta herramienta desarrollada en lenguaje de programación Ruby, permite emular cualquier servicio

informático que pueda ser atractivo frente al atacante, para el proyecto la emulación se hizo pensando en el tráfico web.

- **Cowrie**

Es un Honeypot SSH y Telnet de media interacción escrito en python y twisted direccionado para registrar ataques de fuerza bruta y la interacción de Shell realizada por el atacante, también tiene la capacidad de trasladarse al grupo de alta interacción configurando su proxy como un proxi SSH y TELNET [34].

- **Sistemas de monitoreo y recolección de eventos de seguridad**

"El término log o registro a la grabación secuencial en un archivo o en una base de datos de todos los acontecimientos (eventos o acciones) que afectan a un proceso particular (aplicación, actividad de una red informática, etc.). De esta forma constituye una evidencia del comportamiento del sistema [24].

Generalmente los acontecimientos vienen anotados con:

- El momento exacto o data (fecha, hora, minuto, segundo) en el que ocurrió, lo que permite analizar paso a paso la actividad.

- Una o más categorizaciones del acontecimiento registrado. Es frecuente usar distintas categorías para clasificar la importancia del acontecimiento estableciendo distintos niveles de registro los cuales suelen ser: depuración, información, advertencia y error.

Un sistema SIEM recoge registros y otra documentación relacionada con la seguridad, para ser analizados. La mayoría de los sistemas SIEM funcionan desplegando múltiples agentes de recopilación de forma jerárquica para recopilar eventos relacionados con la seguridad de dispositivos de usuario final, servidores, equipos de red e incluso equipos de seguridad especializados como firewalls, antivirus o sistemas de prevención de intrusiones. Los recolectores envían eventos a una consola de administración centralizada, que realiza inspecciones y señala anomalías. Para permitir que el sistema identifique eventos anómalos, es importante que el administrador de SIEM cree primero un perfil del sistema en condiciones normales de evento, este tipo de sistemas vienen con reglas básicas de detección de ataques y generación de alertas" [24].

- **Sistemas de detección de intrusos**

Son parte esencial dentro del ecosistema de la ciberseguridad, ya que su objetivo es mejorar el nivel de seguridad detectando comportamientos maliciosos o sospechosos que podrían involucrar el dispositivo o el sistema de red [35]. Un sistema de detección de intrusos (IDS) se puede clasificar en pasivos o activos, es decir, podría ser pasivo dado a que detecta actividad maliciosa y solo genera alertas o guarda registros en los logs, mientras que si fuera activo su actuación frente a algún comportamiento hostil generaría alertas, registros y tendría la capacidad de auto-configurarse para tomar medidas como cerrar el acceso, cerrar puertos o bloquear direcciones IP entre otras acciones. Actualmente dentro de la configuración del IDS se puede establecer como sistema de detección de intrusos en el host (HIDS) estos están direccionados al monitoreo de forma local es decir su objetivo principal está orientado hacia el dispositivo o host. En este largo camino de la seguridad también

se debe tener en cuenta los sistemas de prevención de intrusos (IPS), estos dispositivos proactivos de seguridad de red monitorizan el tráfico, resolviendo ambigüedades en las diversas situaciones que se presentan, supervisando los paquetes de entrada buscando detectar algún comportamiento sospechoso emitir una alerta anticipada.

- **Wazuh**

Es una solución de monitoreo de seguridad gratuita, de código abierto, su funcionalidad es ser HIDS, que mediante la instalación de un agente permite el monitoreo permanente para la detección de amenazas. Wazuh reemplazo a OSSEC brindando una nueva plataforma de altos niveles de seguridad, ya que ofrece un dashboard que permite tener una interface y sincronización con Kibanna, logstash, Elastisearch entre otros. Brindando mayor análisis y detección de amenazas [40].

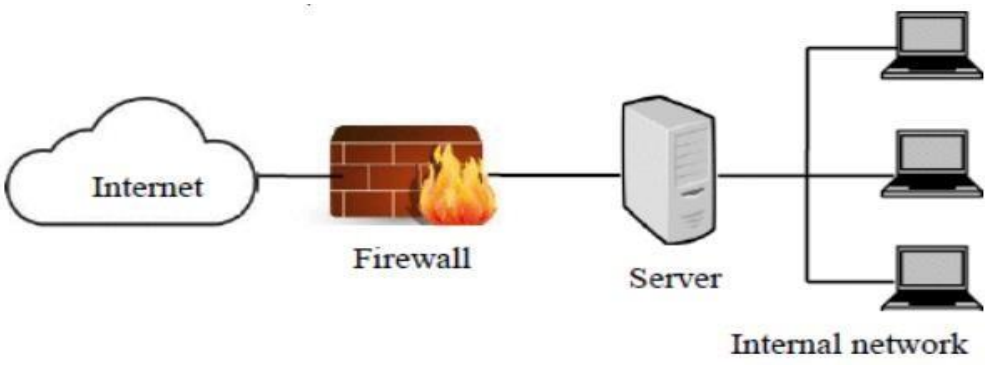
- **Casos de uso con Wazuh**

- Análisis de seguridad
- Detección de intrusiones
- Análisis de datos de registro
- Supervisión de la integridad de los archivos
- Detección de vulnerabilidades
- Evaluación de la configuración
- Respuesta al incidente
- Cumplimiento normativo
- Monitoreo de seguridad en la nube
- Seguridad de contenedores

- **Firewall**

Un firewall o cortafuegos puede ser un hardware o software o una combinación de ambos [37], que ejerce un papel protagónico en la seguridad de la red debido a que su misión es examinar todo el tráfico de adentro hacia fuera y viceversa, de acuerdo con las políticas o reglas definidas solo transita el tráfico autorizado a pasar a través de él, como se muestra en la ***¡Error! No se encuentra el origen de la referencia.***.

Figura 1-5: Esquema del servidor protegido por firewall. [37]



Conciencia del riesgo: tener un inventario claro donde se identifiquen los escenarios de riesgo donde de manera implícita se puede estar como objetivo final.

Sistema de monitorización: Tener componentes de monitoreo en una red Honeynet son de gran utilidad, el sistema se complementa con el análisis de sus fuentes de detección (registros de log) el cual es centralizada para su análisis y correlación de eventos de seguridad, es claro que, para tener objetividad en el análisis, el sistema esta direccionado a la información relevante en la detección de ataques.

Capacidad de respuestas: Es de gran importancia que las organizaciones comprendan el riesgo que corren, ya que la cultura de la seguridad nos invita a la atención de eventos adversos como una oportunidad para aprender y mejorar, en un escenario de crisis el oficial de seguridad es responsable de diagnosticar teniendo herramientas de análisis, planteando como sería la mejor acción en un caso de uno o varios eventos de seguridad.

- **Dispositivos IoT red de sensores**

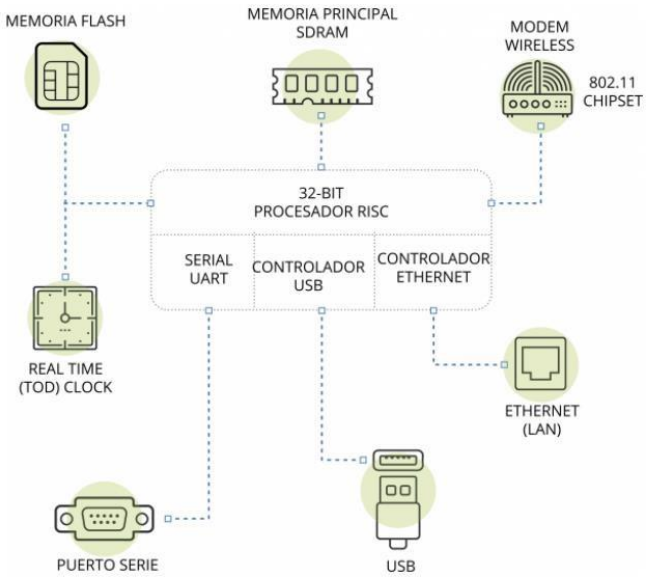
Respaldado en la apreciación de Yi, S., Li, C., & Li, Q [36], una red de sensores IoT tienen el potencial y capacidad de comunicarse entre sí guiados bajo diferentes tecnologías, Las redes IoT hablan de la revolución entre los objetos convencionales y las personas, aun entre los mismos dispositivos, esta idea pretende que dispositivos que años atrás se conectaban en circuito cerrado hoy día lo hagan globalmente, utilizando la nube como plataforma de comunicación basada en sistemas embebidos que mediante la programación en tareas especializadas permite la ejecución de actividades de forma remota. Las plataformas IoT permiten que las cosas puedan procesar o actuar, brindando la posibilidad de comunicarse y coordinarse con otras cosas para la toma de decisiones. “Los sensores integrados y los protocolos de Internet permiten que IoT muestre su importancia.

- **Sistemas embebidos**

Un sistema embebido objetivamente es un sistema informático que está basado en un

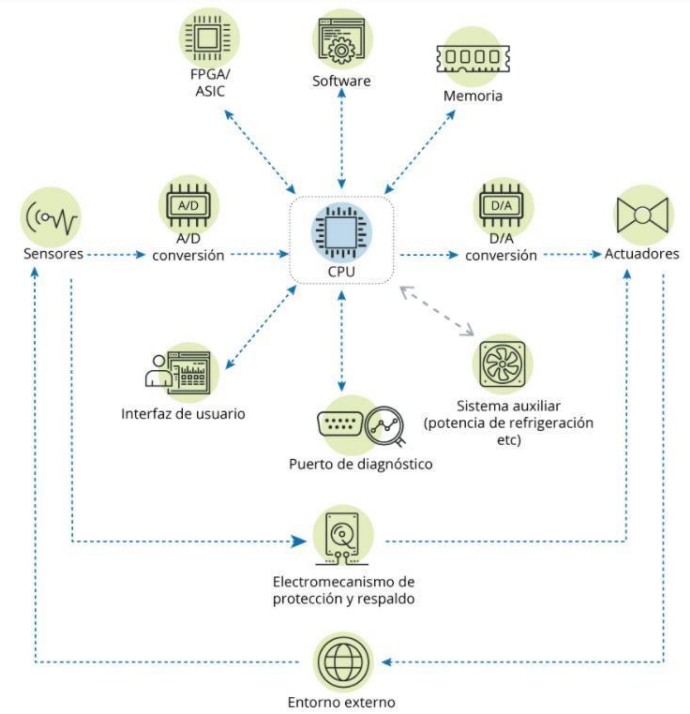
microprocesador, sensores y otros elementos funcionales para las tareas asignadas, y como todo sistema informático este también cuenta nivel físico y lógico como se puede apreciar en la siguiente *Figura 1-6* y Figura 1-*7* [38].

Figura 1-6: Descripción de un sistema embebido (nivel físico). [38]



A nivel físico diferentes componentes pueden ser conectados y desplegados para una interacción concreta dentro de la misma board, con ello, los sistemas embebidos tienen diferentes funcionalidades que perite la conexión de elementos externos como sensores o tener una conexión a elementos de red como switch.

Figura 1-7: Descripción de un sistema embebido (nivel lógico). [38]



Los dispositivos embebidos (aquellos que cumplen una o pocas funciones de forma dedicada) y aparatos electrónicos han evolucionado a velocidades extremas en las últimas décadas, y hoy los fabricantes de circuitos integrados y componentes electrónicos cuentan con una amplia gama de prototipos que satisfacen necesidades en distintos ambientes, no se puede negar que IoT mantiene una estrecha relación con los dispositivos embebidos, ya que son programados para cumplir una función específica, donde el microcontrolador cuenta con un firmware o programa grabado que se ejecuta infinitamente, recibe un mensaje alimentado por un sensor, lo interpreta y se conecta directamente a un router o dispositivo central de procesamiento, este a su vez conexión a un servidor en el cloud computing[11].

Dentro de la gran variedad de dispositivos embebidos se encuentra la Raspberry Pi, el cual es un sistema de cómputo de bajo costo de tamaño reducido, consta de una placa base conformada por un procesador, un chip gráfico, puertos usb, hdmi, ethernet, bluetooth, wifi y memoria RAM entre otras bondades, nativamente con su kernel de Linux permite la instalación de varios sistemas operativos, sin embargo, por el fabricante se realiza la recomendación de instalar el sistema operativo Raspbian, esto debido a que fue creado específicamente para optimizar el hardware brindando mayor rendimiento en la ejecución de procesos [27].

- **Servicios informáticos**

Se cataloga como un conjunto de actividades que buscan la integración de un ecosistema informático que permita interactuar con usuarios, pretendiendo dar soluciones. Los escenarios donde se presentan estas necesidades VS las soluciones pueden ser Computación en la nube, gestión de base de datos, inteligencia empresarial, seguridad, gestión de activos y desarrollo de aplicaciones móviles, análisis estadísticos, sistemas de detección de eventos, entre otras. Para profundizar en el alcance que se tiene se destacan servicios informáticos como Secure Shell (SSH), este protocolo que integra las comunicaciones seguras entre dos máquinas usando una arquitectura cliente/servidor, permitiendo

que el usuario de forma remota tenga la facilidad de conexión a un host. El servicio Teletype Network (telnet) es un protocolo de red a otra máquina también buscando la conexión de forma remota. El servicio protocolo de transferencia de archivos (FTP) permite en medio de la red la transferencia de archivos basados en la arquitectura cliente/servidor. Apache es un servicio que fácilmente se puede emplear como servidor web HTTP de código abierto, teniendo en cuenta características importantes como soporte de seguridad SSL y TLS, autentificación de datos utilizando SGDB y brinda soporte a diferentes lenguajes, como Perl, PHP, Python y tcl [41].

- **Python**

Es un lenguaje de programación dinámico multiplataforma, que soporta orientación a objetos, programación imperativa y programación funcional, este lenguaje de scripting no necesita compilar código fuente para poder ser ejecutado, lo que permite resaltar la ventaja de una ejecución con mayor rapidez, viene acompañado de funciones y librerías que se pueden importar en la construcción del código fuente [39].

- **Auto-configuración o Configuración Automática**

La auto-configuración es la capacidad que tiene el dispositivo IoT en su sistema operativo de un funcionamiento autónomo, donde tiene la oportunidad de subir y bajar servicios informáticos, teniendo en cuenta lo indicado por TW, Kim [44] es posible llegar a la auto-configuración con herramientas de desarrollo de lenguaje, python, flex y bison.

- **Script**

Es un documento que contiene instrucciones construidas en códigos de programación, dicho documento contiene una lógica implícita y una serie de etiquetas o atributos que permite ejecutar de una forma aleatoria o consecutiva acciones, actualmente el mundo informático se encuentra rodeado de scripts que ejecutan comandos con el fin de iniciar, pausar, concluir y auto-configurar el funcionamiento de tareas. Los dispositivos IoT en su código fuente se enfrentan a lenguajes de programación como tipos de script: Actionscript, JavaScript, Lua, PHP, Python, ShellScript, Ruby, VBScript [39].

- **Ataque**

Se puede identificar como toda incursión informática sin previa autorización, donde una o varias personas e inclusive dispositivos previamente configurados intentan violar parámetros de seguridad. Entre muchos ataques que hoy día existen se podría resaltar ataques por contraseña o diccionario, ataques por fuerza bruta, ataque SQL Injection, ataque DoS – DDOS, entre otros. Un atacante direcciona todo su potencial cuando en el dispositivo informático identifica un patrón de vulnerabilidad, es decir, identifica la existencia de una debilidad o defecto de programación en el software o en el sistema operativo, este escenario se convierte en una potencial amenaza ya que es un riesgo en la red y la infraestructura [42].

- **AHP –Proceso de análisis Jerárquico**

Es una metodología para estructurar, medir y sintetizar en escenarios de toma de decisión multi-criterio, es decir que este método matemático optimizado para evaluar alternativas cuando se tienen varios criterios [43]. El AHP (por sus siglas en inglés-*Analytic Hierarchy Process*) utiliza comparaciones entre patrones o datos construyendo matrices, la toma de decisiones juega un papel importante en cualquier proyecto de corto, mediano o largo plazo y a esto se le suma la complejidad en los diversos escenarios que se convierte en variable imperativa que puede lograr la desviación del objetivo. Este método cuantitativo permite generar escalas de prioridades basándose en análisis expertos reflejados a través de comparaciones por pares, esencialmente se puede definir su ejecución en 4 pasos básicos [43]:

- Definición del problema.
- Estructuración del problema.
- Construcción de matrices de comparación.
- Síntesis de cada una de las matrices

1.2 Estado del arte

Para lograr identificar si cada uno de los componentes IoT son vulnerables o tienen ataques informáticos de manera continua, es necesario la detección e identificación de posibles eventos de seguridad, y aunque se cuenta con soluciones manuales o adaptativas de tipo Honeypot o Honeynet que ayudan en la identificación y rastreo de ataques, aún se tiene el problema de tener Honeypots de configuración manual para la activación e inactivación de servicios informáticos, estas pre configuraciones siempre necesitan asistencia de forma manual, por lo cual es necesaria la utilización de personal calificado en todas sus fases, negando la oportunidad de tener una Honeynet en IoT que de forma auto-configurable pueda activar o desactivar servicios informáticos adicionales dentro del mismo dispositivo IoT, para que, a través de los atacantes, se pueda generar nuevo conocimiento que será capitalizado por medio de análisis de datos posteriormente.

En el trabajo de [11] exploraron aspectos de la construcción de un sistema trampa (Honeypot), que se despliega en toda la red de forma dinámica en su configuración, implementación y mantenimiento ejecutado bajo técnicas de aprendizaje automático. Este tipo de sistema utiliza como implementación el lenguaje Python y como enfoque de aprendizaje automático utiliza una versión de k-means-clustering, para posteriormente estudiar patrones de comportamiento de los atacantes. Sin embargo, dentro de los puntos exitosos del proyecto no se contempla la posibilidad que el dispositivo interactúe con la arquitectura Honeynet, o que tenga la posibilidad de ser auto-configurable, es decir que no solamente desempeñe la tarea de realizar funciones que le fueron asignadas, sino que también posea la inteligencia que le permita de forma autónoma reconfigurar sus servicios informáticos, sobre todo buscando mayor eficiencia y eficacia en el monitoreo y captura de información.

En segundo lugar [12], presentan un diseño Honeypot para un sistema Smartphone (Teléfono inteligente), dada la amplitud de servicios que hoy en día puede ofrecer la portabilidad de un Smartphone como: Conectarse a la nube, acceder a redes sociales, continuamente visualizar el correo electrónico, grabar videos, fotos, aplicaciones de salud y otro amplio panorama de actividades. Toda esta evolución lo convierte en una herramienta de trabajo o diversión, contagiando a muchos a poseerlo. Pero paralelamente a su evolución el paisaje de amenazas se ve acelerado por el ataque y complejidad de las aplicaciones maliciosas que se dirigen a ellos. Aunque algunos poseen herramientas básicas de protección como firewalls, sistemas de detección de intrusos y antivirus, la seguridad de la información se ve vulnerada, por lo tanto, se necesitan nuevas técnicas para identificar y reconocer diferentes tipos de ataques. Sin embargo, el diseño propuesto no contempla la construcción de una arquitectura Honeynet y ejecución de un sensor auto-configurable.

Así mismo, se tiene un sistema multi-agente para el análisis de datos de una Honeynet [13], en la cual a través de diferentes agentes de software recolecta información y toma decisiones sobre ésta, con estos resultados es posible observar las posibles vulnerabilidades de los sistemas y tomar acciones futuras para su protección, sin embargo, se centra en redes de computadores y no comprende el concepto de auto-configurable.

Similarmente se han trabajado algunos Honeypot que buscan capturar tráfico malicioso para protocolos como SSH, Telnet o HTTP [14] desplegando una serie de Honeypot de propósito

particular, pero no concretan servicios que puedan ser o bien más extensos (dns, ftp) o bien auto-configurables, esto es, que a medida que un posible atacante descubra servicios, estos se puedan reconfigurar con otros valores atractivos para los siguientes atacantes.

[15] un Honeypot que de manera interactiva realice cierto escáner en las redes IoT en busca de posibles vulnerabilidades, se plantea un sistema de simulación inteligente que logre recolectar diferentes muestras de sensores como cámaras, firewall, router entre otros, dentro de los resultados simulados obtuvieron información de servicios y protocolos como HTTP, tcp, udp, ssh entre otros; para dichos resultados proponen el uso de cadenas de Markov con el fin de tomar decisiones con respecto a los ataques, pero en éste modelo no plantean una red real que pueda realizar configuraciones a medida que se desarrollan los ataques en IoT, además, sigue siendo un prototipo en construcción.

Por otro lado, [16] proponen un Honeypot adaptativo y auto-configurable capaz de cambiar dinámicamente su comportamiento usando algoritmos de aprendizaje minimax-learning, enfocado en juegos estocásticos enfrentando al atacante, en donde el enfoque es brindado en el modo consola de tal forma que solo relaciona un patrón de comandos que interactúan con el atacante en la shell, brindando la oportunidad de utilizar varios comandos que permitan el ingreso del atacante al Honeypot. Sin embargo, la literatura no contempla la posibilidad de un término auto-configurable enfocado en los servicios de la red señuelo y tampoco menciona alguna relación con Honeynet en IoT.

Así mismo, en [17] se propone una arquitectura de despliegue de Honeypot adaptativos, configurados dinámicamente, tomando como punto primario los requisitos del malware, es decir que pre configuran en un dispositivo con diferentes tipos de Honeypot y de acuerdo con el malware lanzado por el atacante, el sistema señuelo tiene la capacidad de clasificar que Honeypot es el que interactúa con el atacante. Sin embargo, no existe evidencia de un Honeypot utilizando el papel de sensor auto-configurable en IoT, que permita subir servicios según la petición del atacante.

De igual forma [18], especifica que las Honeynet tradicionales poseen una configuración estática en cuanto a servicios expuestos se refiere, en el caso de requerir un nuevo servicio, éste debe ser aprovisionado en cualquiera de los componentes de la Honeynet, y las Honeynet dinámicas permiten una auto-configuración básica en la capa de transporte, sin embargo, no relaciona un funcionamiento en IoT auto-configurable, que permita en el mismo dispositivo sensor activar o desactivar servicios y con ello, obtener la información de los posibles ataques.

Consecuentemente, [19] en su trabajo identifican las vulnerabilidades en dispositivos IoT debido a sus limitados recursos de red y sistemas operativos complejos, no obstante, implementan 3 tipos de Honeypots como una herramienta confiable a la hora de capturas de solicitudes maliciosas, identificando la vulnerabilidad CVE-2017-17215 intenta simular servicios UPnP de enrutado, que permite la simulación de servicio, grabación de registros, descarga de muestras maliciosas y autocomprobación del servicio. En segundo lugar, utilizan el Firmware de un dispositivo IoT, enlazando la vulnerabilidad para la construcción de un Honeypot de alta interacción.

Por su parte, las redes IoT se identifican con igualdad de vulnerabilidades que las redes

informáticas, la empresa CISCO, por ejemplo, las clasifica en redes de consumo o industriales, y cada una con directrices diferentes. [20], así mismo centran su trabajo en redes IoT buscando la forma de anticipar las amenazas cibernéticas y sugieren puntos de enfoque para los investigadores digitales.

Se resaltan algunas características de los proyectos de investigación y artículos que se hablaron en el estado del arte, todos obtienen resultados exitosos en la evidencia de detección de intrusos, aunque utilizan metodologías totalmente diferentes, sin embargo, todos coinciden en la utilización de Honeypot de manera programada es decir que los dispositivos utilizados operan de manera manual con una programación inicial, mas no contempla la posibilidad de un comportamiento auto-configurable.

Hasta el momento se tiene la necesidad de configurar una arquitectura de Honeynet que permitan auto-configurar dicha arquitectura en un mismo IoT, que pueda facilitar una forma de capturar información de posibles atacantes para diferentes servicios, pero que dichos servicios se puedan auto-configurar dentro del mismo dispositivo IoT de acuerdo a los intentos de explotación que se vayan ejecutando, con ello, logran obtener mucha más información de cómo es el actuar de los atacantes en las redes IoT.

2. Metodología

Este proyecto de investigación es aplicado, con base en la revisión de literatura, aplicación de métodos y evaluación, mediante el análisis de datos recolectados acorde a los ataques que se van realizando, que, para el caso, con ataques controlados simulados. La implementación de la Honeynet se realizó en un ambiente de pruebas controlada que permitió la simulación de un ambiente real. Las redes señuelos si bien son ambientes no productivos (dado que no poseen la información real), si fueron operativos para los atacantes. La información resultante es Cuasi- Cuantitativa; es decir, el enfoque metodológico fue tipo mixto, y al mismo tiempo, descriptiva, explicativa y prospectiva, dado que se describe una serie de hechos, para lograr alcanzar cada uno de los objetivos específicos y así mismo, los resultados serán analizados para encontrar las razones o causas que fundamenten la argumentación de cada uno de ellos.

A continuación, en la *Figura 2-8*, se hace referencia a la metodología utilizada para la consecución de los resultados:

Figura 2-8: Fases de la metodología. Fuente propia.

2.1 Fase 1: Diseñar la Red Honeynet

Esta fase consta de 3 actividades:

- Selección de la arquitectura Honeynet
- Selección del dispositivo IoT

- Definición de los servicios y los Honeypot que harán parte de la Honeynet
- Definición de los ataques informáticos a ser implementados.

A continuación, se explica cada una de las actividades:

2.1.1 Selección de la arquitectura Honeynet

En esta actividad se analizaron las diferentes arquitecturas de red que se pueden implementar, aunque no hay una exigencia en cuanto a la implementación, es decir, que se tiene plena libertad a la hora de escoger tanto su tipología como los componentes a utilizar como lo pueden ser dispositivos IoT y mecanismos de comunicación, ya que es un tema muy abierto de acuerdo a las necesidades de cada escenario, sin olvidar factores importantes como lo son control de datos, captura y recopilación de datos. Dependiendo de las tecnologías adoptadas y de la forma en que se han llevado a cabo las actividades de captura, control y recopilación de datos dentro de la red Honeynet a lo largo de los años, las Honeynet han evolucionado a través de varias arquitecturas o generaciones hasta las actuales que son combinaciones o hibridas (siendo esta última la que continúe en la actualidad), y de éstas, se ha seleccionado la más acorde a la necesidad del cumplimiento del objetivo, considerando:

- Que sea funcional: La aplicabilidad y configuración de los servicios y protocolos en IoT requeridos para el cumplimiento de los objetivos se pueden establecer desde esta arquitectura.
- Que se puedan implementar servicios bases como SSH o Telnet: Dado que es una Honeynet, se puede configurar los servicios requeridos para las pruebas y diferentes configuraciones.
- Que sirva como Honeynet en ambiente de pruebas, con funcionalidad simple: No es requerido configuraciones adicionales, pues se trata de probar la funcionalidad de auto-configuración ante diferentes ataques informáticos.

2.1.2 Selección de dispositivo IoT

En este segmento el enfoque fue analizar varios dispositivos IoT mediante la descripción de los fabricantes, con el fin de seleccionar un sensor que pueda ser configurado con diferentes servicios informáticos y a su vez que tenga la capacidad de adaptarse a aun método de auto-configuración. Debido a la necesidad del proyecto en cuanto a tener la posibilidad de contar con un sensor que tenga la capacidad de gran procesamiento de logs y potencial para responder a la interacción con el atacante, se evaluaron los diferentes dispositivos considerando, al menos, capacidad en procesamiento o CPU, cantidad de memoria, capacidad de instalación de sistema operativo y configuración de servicios informáticos, si puede manejar o ser configurados sistema de logs o registros de auditoria (de tipo syslog o similar) y que pueda soportar lenguajes de programación como Python, luego por medio del método de decisión multi-criterio (AHP) [25], se procede a determinar el sensor utilizado.

Teniendo en cuenta que un sensor IoT es una placa reducida y en consecuencia su desempeño es limitado, ya que no puede abarcar un ecosistema multiprocesos a diferencia de lo que puede ser una mainboard de un computador que está catalogado como un dispositivo que puede estar ejecutando

muchas tareas al mismo tiempo, se debe ser consciente de esta verdad para lograr un método que no exceda las capacidades del sensor IoT.
En ese sentido se realizaron los siguientes pasos:

- Definición de las características a evaluar o buscar en los dispositivos
- Definición de los IoT
- Aplicación del método multi-criterio (AHP) para la selección de un dispositivo de acuerdo a las necesidades y características requeridas.

A continuación, se da más detalle de cada uno de los pasos:

a) Definición de características

Como se indicó, las siguientes son las características esenciales para el proyecto:

- **Capacidad en procesamiento o CPU**: Al tener un buen procesador esto se será reflejado en una mayor rapidez a la hora de realizar las operaciones, ya que este componente será el encargado de leer, interpretar y procesar las instrucciones del sistema operativo y posteriormente de programas o aplicaciones.

- **Memoria**: Este jugador tendrá un papel fundamental que ira de la mano del procesador, ya que es la mesa de trabajo donde la CPU registra y almacena los datos de forma temporal, mientras realiza la variedad cálculos en las tareas que le son asignadas al dispositivo.

- **Capacidad de instalación de sistema operativo y Configuración de servicios informáticos**: Este software y sus servicios informáticos son vitales en el ecosistemas del dispositivo ya que enlazan una estrecha relación con el hardware, son los brazos que representan todo el esqueleto lógico que permite desarrollar los objetivos plasmados en el lenguaje máquina, teniendo un Sistema operativo liviano pero potente, se tendrá la oportunidad de ajustar el sensor en un mismo esquema que no lo limite sino que le genere un valor agregado, favorable sobre la carga operativa del dispositivo IoT.

- Conectividad: En una red de señuelos Honeypot es de suma importancia tener acceso a la red en la mayor actividad de protocolos ya sea vía ethernet o wifi, esto para tener mayores oportunidades a la hora de que el atacante busque filtrar sus conexiones hacia el dispositivo IoT objetivo.

- **Almacenamiento**: El dispositivo debe ser capaz de almacenar o administrar un sistema de almacenamiento (local o remoto), con lo cual, se valida si puede manejar o ser configurados sistema de logs o registros de auditoria (de tipo syslog o similar) y que pueda soportar lenguajes de programación como Python.

b) Selección de dispositivos IoT

En el mercado existe una gran variedad de dispositivos embebidos que pueden ofrecer un ecosistema IoT acorde a las necesidades de una organización o personas naturales, con el fin de llevar a cabo este proyecto se realizó la medición sobre al menos 5 dispositivos IoT que fueron buenos candidatos para la construcción de la arquitectura Honeynet y en ese sentido, se revisó la documentación de los diferentes proveedores y de dicha información se obtuvieron datos relevantes asociados a la capacidad de CPU, RAM, Almacenamiento, Conectividad y la posibilidad de soportar servicios informáticos. Luego de ello se procede con la selección del dispositivo a utilizar.

c) proceso de análisis jerárquico (AHP)

El proceso de análisis jerárquico (AHP), tiene su enfoque en la evaluación de diferentes criterios que permiten jerarquizar un proceso, fue propuesto por Thomas Saaty en 1980, dicho método resalta la comparación por pares para buscar alternativas basadas en ponderaciones y segmentación de criterios, este método se ha empleado en grandes proyectos como lo son evaluaciones de infraestructura para transporte y evaluación de criterios monetarios entre otros. Uno de los inconvenientes que presenta el método AHP es que no permite calificar un elemento de manera independiente sino en comparación, reforzando la teoría de su inclinación al diagnóstico por pares [26].

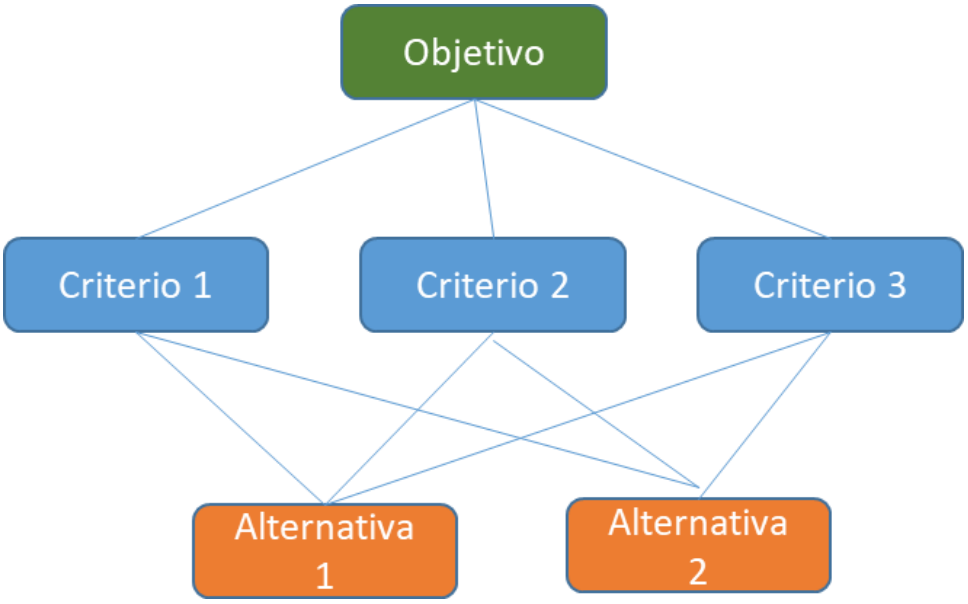
Dado a que el método AHP permite jerarquizar el proceso, se debe fragmentar el problema de decisión en los siguientes pasos.

- Definir el problema y determinar el tipo de conocimiento que se genera.
- Estructurar la jerarquía de decisión desde la parte superior con la meta que se busca alcanzar, construir un grupo de matrices con las comparaciones por pares.
- Usar las prioridades obtenidas para compararlas con las prioridades en el nivel inferior.

De esa manera, se respalda el método realizando en 1994, validando y dando una puntuación acorde a una tabla (tabla 1), la cual permite indicar el valor en escala que debe asumir cada comparación según sus diferencias y preferencias, dicha tabla es de gran ayuda sobre todo cuando se debe establecer una relación entre los criterios y las alternativas.

El proceso general para la selección de un dispositivo IoT consistió en definir primero el objetivo a buscar, luego los criterios de búsqueda y finalmente realizar las comparaciones con las alternativas a seleccionar como se puede observar en la *Figura 2-9*.

Figura 2-9: Proceso global de selección multi-criterio. Fuente propia.



Ahora bien, el objetivo fue seleccionar un dispositivo IoT que cumpla, en combinación, siguiendo el siguiente procedimiento:

- Identificación de alternativas a comparar, para el caso, sería los diferentes dispositivos IoT a seleccionar.
- Identificación de criterios para los elementos
- Realizar la matriz de comparación pareada entre criterios (haciendo uso de escala de comparación pareada), normalizar y obtener el vector promedio.
- Realizar la ponderación de criterios vs. elementos, normalizar y sacar vector promedio.
- Obtener la matriz jerárquica.

Para la comparación entre alternativas, se usó la matriz Saaty (Tabla 2-*1*), donde cada elemento se compara con otro y dando un valor de acuerdo con la característica a evaluar.

Tabla 2-1: Escala de comparación pareada. Fuente [25]

Escala de Comparación pareada en AHP por Saaty		
Intensidad	**Definición**	**Explicación**
1	De igual importancia	2 actividades contribuyen de igual forma al mismo objetivo
3	Moderada importancia	La experiencia y el juicio favorecen levemente a una actividad sobre la otra
5	Importancia fuerte	La experiencia y el juicio favorecen fuertemente a una actividad sobre la otra
7	Muy fuerte	Una actividad es mucha más favorecida que la otra
9	Extrema	La evidencia que favorece una actividad sobre la otra es absoluta y totalmente clara
2,4,6,8	Valores intermedios	Cuando se necesita un compromiso de las partes entre valores adyacentes

- **Método para jerarquizar**

- Se utilizó un método que combina modelos cualitativos y cuantitativos.

- Se midió con base en indicadores nacionales, de eficiencia y puntajes de cumplimiento con objetivos.

- La jerarquización se hizo con base en puntajes ponderados obtenidos de valores normalizados de los indicadores.

Para poder realizar la matriz de comparación pareada entre criterios, normalizar y obtener el vector promedio, se utilizó la tabla 2, en donde hace una comparación entre características con base en la escala de la tabla 1, luego, se genera la matriz normalizada (Ecuación 2-1):

$$\text{Característica (i,j) x total} \quad (2\text{-}1)$$

En donde I es la fila 1 y J es la columna 1. Como resultado final, se obtiene el vector promedio considerando la sumatoria de la matriz normalizada (Ecuación 2-2):

$$\sum_{k=0}^{n} Matriz \quad (2\text{-}2)$$

Tabla 2-2: Matriz normalizada vacía. Fuente propia.

Performance	Cpu	Ram	Almacenamiento Interno	Conectividad	Serv. Info	Matriz normalizada					Vector promedio
Cpu											
Ram											
Almacenamiento Interno											
Conectividad											
Serv. Info											
Totales											

Los componentes que se ilustran en la Tabla 2-*2* son características a resaltar de cada dispositivo IoT, los cuales representan gran relevancia, ya que son de gran impacto a la hora del desempeño cuando se instala la HoneyNet, dicho performance es el punto de apoyo para lograr completitud en la matriz de comparación.

- CPU: Al tener un buen procesador esto se será reflejado en una mayor rapidez a la hora de realizar las operaciones.
- Ram: Este jugador tendrá un papel fundamental que ira de la mano del procesador, ya que es la mesa de trabajo donde la CPU registra y almacena los datos de forma temporal.
- Almacenamiento Interno: El dispositivo debe ser capaz de administrar un sistema de almacenamiento (local o remoto), con lo cual, se valida si puede manejar o ser configurados sistema de logs o registros de auditoría.
- Conectividad: En una red de señuelos Honeypot es de suma importancia tener acceso a la red en la mayor actividad de protocolos ya sea vía ethernet o wifi, esto para tener mayores oportunidades a la hora de que el atacante busque filtrar sus conexiones.
- Servicios informáticos: Son vitales en el ecosistema del dispositivo ya que enlazan una estrecha relación con el hardware y además representan puertas de entrada a la hora de llamar la atención del atacante.

En la Tabla 2-3 se realiza la ponderación de criterios vs. elementos, normalizar y sacar vector promedio, se toma cada alternativa y se hace una comparación entre ellas, validando cada característica que tan fuerte es o no para cada alternativa, con lo cual, se tendría una matriz por cada criterio. Luego de la evaluación de cada alternativa con respecto a las demás, se obtiene la matriz normalizada y el vector promedio final resultante (acorde a las fórmulas ya indicadas):

Tabla 2-3: Ponderación de criterios con respecto a las alternativas a evaluar.

Ponderación de criterios vs. elementos							
	Criterio a evaluar			Matriz normalizada			Vector promedio
	Alternativa 1	Alternativa 2	Alternativa 3				
Alternativa 1							
Alternativa 2							
Alternativa 3							
Totales							

Finalmente, en la Tabla 2-*4* se evidencia la fase final para obtener la matriz jerárquica, se llevan todos los resultados del vector promedio y se obtiene la ponderación total acuerdo a la siguiente formula:

Tabla 2-4: Matriz jerárquica final. Fuente propia

$$[\Sigma]x[\Pi]$$

MATRIZ JERÁRQUICA					
	Criterio 1	Criterio 1	Criterio 1	Criterio 1	**Total**
Alternativa 1					0,00%
Alternativa 2					0,00%
Alternativa 3					0,00%
Vector Promedio					

El dispositivo IoT que se seleccionó es aquel que obtuvo el mayor puntaje.

2.1.3 Definición de los Honeypot que harán parte de la Honeynet y sus servicios

Para la definición de los Honeypot y los servicios informáticos y en consideración de la selección anterior, se consideraron aquellos servicios básicos que pueden estar de cara a Internet y que son atractivos para un atacante. Dichos servicios fueron seleccionados considerando lo que ya ofrecen los Honeypot disponibles, para lo cual, se diligenció la siguiente

Tabla 2-5 de características.

Tabla 2-5: Características de los diferentes Honeypot. Fuente propia

HONEYPOT	SERVICIOS	OPCIONES DE CONFIGURACIÓN	OPEN SOURCE

- Honeypot: Nombre del Honeypot investigado.
- Servicios: Cuales son los servicios informáticos que tiene o se puede usar.
- Opciones de configuración: Si es posible realizar otras configuraciones adicionales a los servicios o programas que ya se tiene.
- Open Source: Que pueda ser usado bajo la licencia GNU, esto, dada la importancia de contar con herramientas de libre uso.

Una vez realizada la caracterización, se seleccionó el software que cumpla todas las condiciones.

2.1.4 Definición de los ataques informáticos a ser implementados

Con respecto a la selección de los ataques informáticos, éstos fueron seleccionados con base en los servicios a implementar y en consideración que es un modelo para validar la auto-configuración, la selección de estos consistió en varios ataques comunes asociados a dichos servicios.

2.2 Fase 2: Diseñar el método de Auto-configuración en un sensor IoT

Para el diseño del método, en esta fase se realizaron 2 actividades:

- Implementación de la arquitectura.
- Definición y construcción del método de autoconfiguración.

2.2.1 Implementación de la arquitectura seleccionada

En esta actividad de tomaron todos los componentes definidos en la fase 1 y se realizó la implementación respectiva, adicionalmente, se hizo una validación básica de de la conectividad, servicios funcionales, honeypot y que los ataques informáticos seleccionados si fuesen capturados y registrados en los diferentes Logs (pruebas unitarias).

2.2.2 Definición y construcción del método de auto-configuración

En esta actividad se analizaron diferentes mecanismos para la construcción de la auto-configuración, así como la construcción de la solución, buscando la opción que represente la mayor eficiencia dentro de las capacidades que posee el dispositivo IoT, ya que dentro del objetivo general se menciona la necesidad de que la auto-configuración provenga del sensor IoT, produciendo el impacto que permita la detección de diferentes tipos de ataques.

Para la definición del mecanismo usado para la auto-configuración se analizaron las siguientes opciones: Machine learning, Deep learning, uso de Python

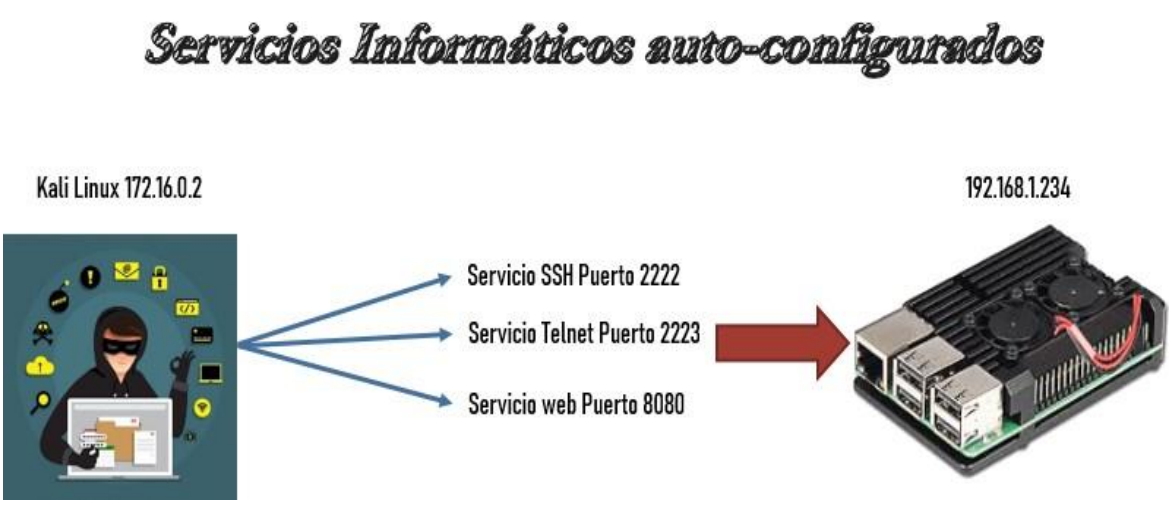
Dentro de las tareas que tiene que cubrir el método de auto-configuración se tiene en cuenta lo siguiente:

- Definición de constantes.

- Iniciar Honeypots en la red.
- Diferencia una detección para ssh, telnet y web.
- Definir las líneas con detección.
- Reinicia de servicios aleatoriamente.
- Apagado y encendido de Honeypots.

Con el método de auto-configuración se pretende que el sensor IoT pueda seguir actuando con el rol cliente-servidor, el objetivo de este método es lograr la percepción de los diferentes ataques como lo son ataques de fuerza bruta y/o diccionario y ataque vía web, que a su vez el sensor tenga la capacidad de responder subiendo y bajando servicios informáticos, estos servicios están respaldados por una red de señuelos (Honeypot), los cuales están a la escucha de las técnicas utilizadas por el atacante, estos señuelos tienen el alcance de ejecutar servicios informáticos como se indican en la *Figura 2-10*.

Figura 2-10: Servicios informáticos auto-configurados. Fuente propia.



Según el instituto de tecnología SANS especialistas en seguridad de la información y capacitación en ciberseguridad, indica que dentro del top20 se encuentran como principales objetivos de ataques los servicios SSH, telnet, y servicios web entre otros [49].

El diseño final será entonces la definición de la arquitectura y su implementación, la definición de la estrategia de auto-configuración y el desarrollo de la misma, finalmente una prueba unitaria para validar su funcionalidad.

2.3 Fase 3: Ejecutar y documentar las pruebas

En esta fase, una vez configurados todos los elementos dentro de la red de prueba, se ejecuta (con Kali Linux) los diferentes ataques informáticos seleccionados en la fase 1, se validó el funcionamiento del programa creado de acuerdo a la selección realizada en la fase 2 y se validó como se van reflejando en los Logs los datos y registros, y a partir de esto, el sistema activa o desactiva los servicios para que el atacante puede visualizar, así mismo la visualización en Wazuh como herramienta gráfica de monitoreo. Para cada prueba se obtiene imágenes como evidencia del funcionamiento.

Para la infraestructura tecnológica, se realizó la conexión física desde el dispositivo IoT a una red LAN de prueba, así mismo, los Honeypots se configuraron y conectaron a la red, realizando las diferentes pruebas de conectividad entre componentes y pruebas funcionales de los programas.

Mediante la implementación en tiempo real de la Honeynet, se validó el comportamiento del sensor IoT, con ello, verificar y analizar los log recolectados. Uno de los objetivos principales de una Honeynet es aprender del atacante, en esta fase se buscó recolectar y almacenar las huellas del atacante con el fin de generar conocimiento, y a su vez analizar el comportamiento del sensor IoT en el ambiente de producción, esto se logrará manteniendo un trabajo colaborativo entre dispositivos informáticos configurados y el sensor IoT, ejecutando diferentes ataques informático, revisando y documentando el comportamiento de éstos, así mismo y dependiendo de que lo que se logre identificar, se entregará recomendaciones de seguridad acorde a lo detectado (más no se implementará).

En esta última fase se pretende brindar respuesta al objetivo general donde se propone un método de alerta de eventos de seguridad, toda alerta está dirigida al aviso de una amenaza que se podría materializar en un ataque, ya que los sistemas informáticos son redes complejas, el tener un método de alerta nos suma un escenario de 3 tres componentes: conciencia del riesgo, sistema de monitorización y capacidad de respuesta.

3. Resultados

A continuación, se declaran los resultados obtenidos posterior a la ejecución de los 3 objetivos específicos planteados y de acuerdo con la metodología indicada (3 fases con sus respectivas actividades):

3.1 Fase1: Diseñar la red Honeynet

A continuación, se entregan los resultados de las diferentes actividades indicadas en la metodología.

3.1.1 Definición de la arquitectura Honeynet

Como se menciona anteriormente, existen varias arquitecturas que son sugeridas, primera, segunda, tercera generación, adicional, la opción de virtualidad y las Honeynets híbridas.

Para el caso la primera generación ofrece un escenario que se identificó con el proyecto, ya que se puede visualizar la funcionalidad requerida como lo es la utilización de un firewall que pueda ofrecer la manipulación de reglas iptables, mientras que la generación 2 y 3 establece la implementación de un honeywall y este componente tiene un alcance en redes funcionales en producción, más no las de ambiente controlado como lo es este proyecto.

La opción de hacer uso de máquinas virtuales en ambientes híbridos, permite la conexión de elementos físicos con los componentes software emulados a través de la virtualidad y en ese sentido, da más funcionalidad.

Con respecto a la selección de la arquitectura a implementar, la opción de una solución híbrida cumple con los criterios definidos:

- **Que sea funcional**: La interacción de u IoT físico con los Honeypot en máquinas virtuales da una interacción para el cumplimiento de los objetivos.

- **Que se puedan implementar servicios informáticos**: Todas las arquitecturas permiten esto,

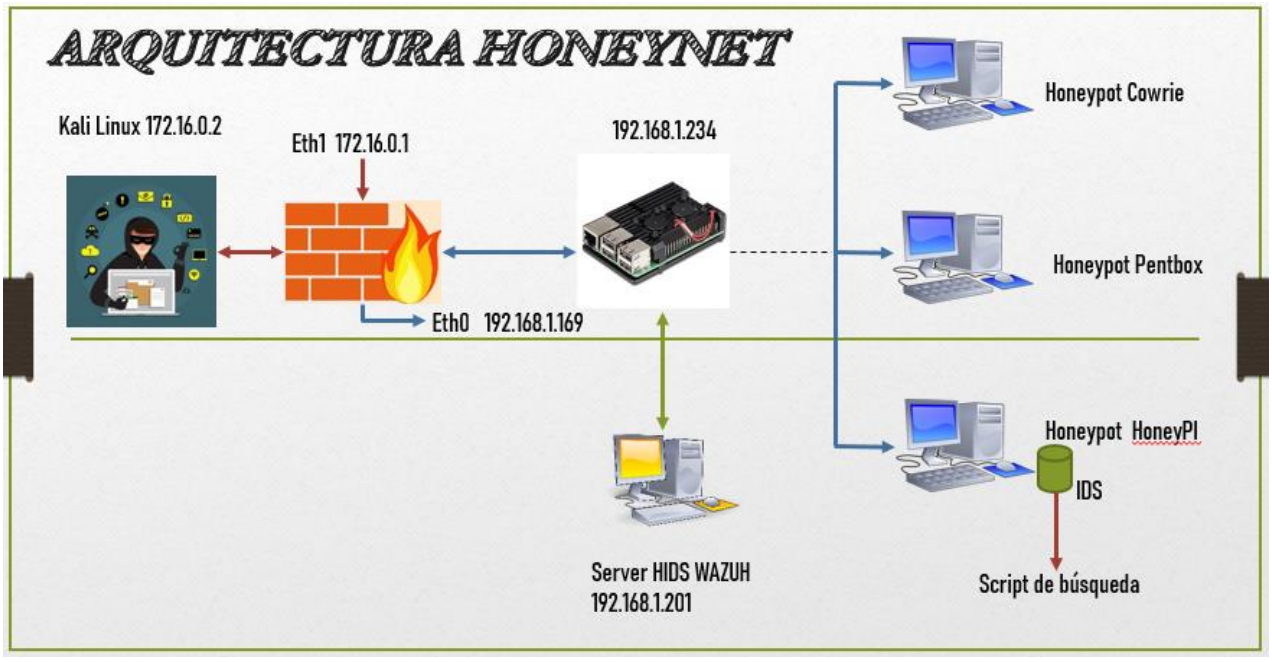
pero al contar con la virtualidad, se torna más simple la reutilización o uso de otras máquinas para tal fin.

- **Que sirva como Honeynet en ambiente de pruebas, con funcionalidad simple**: La opción híbrida puede brindar tanto opciones complejas como simples, y se puede configurar de diferentes formas.

En consecuencia, en este punto se validaron las diferentes arquitecturas Honeynet ya definidas, identificando la arquitectura híbrida como la más conveniente para desarrollar el proyecto, en su implementación entonces, se considera utilizar una serie de plataformas que permite el enlace entre dispositivos de cómputo y el dispositivo físico IoT, es un escenario cliente servidor que permite la auto-configuración de servicios informáticos.

Como resultado del diseño, en la *Figura 3-11*, se puede observar la arquitectura definida, la cual está compuesta por una red de sistemas señuelos, entre los que se encuentran un firewall (en máquina virtual) que funciona como filtro de paquetes a nivel de red con el único fin de tener relación entre el interior de la Honeynet e internet (atacante), con ellos, se divide la red en dos segmentos, esta configuración propicia cualquier conexión desde el exterior del sistema y permite controlar las conexiones que se intente establecer desde los Honeypots hacia el exterior. Otro componente importante es el dispositivo físico IoT encargado de centralizar y monitorear el comportamiento de los logs, y, por último, se cuenta con un componente servidor HIDS (virtual) que mediante la instalación de un agente en el sensor IoT permite el constante monitoreo de los servicios y logs del sistema que son de gran importancia para la toma de decisiones de cara al oficial de seguridad.

Figura 3-11: Arquitectura Honeynet. Fuente propia



Para el montaje del laboratorio realizado en un ambiente controlado, se simuló lo que se puede

presentar en una red real, dicha red en IoT tiene la potencialidad de generar impactos como lo es en un ambiente de producción, esta instancia refleja los impactos que resultan de la auto-configuración en un lenguaje o esquema computacional.

De acuerdo a la arquitectura definida, para la simulación de los ataques se tiene un Kali Linux y como sistema firewall, un servidor Ubuntu con *iptable* activo y funcionando, así mismo como mecanismos de monitoreo, se tiene el HIDS Wazuh.

3.1.2 Selección del Dispositivo IoT

El proceso desarrollado en esta fase (como se indicó en la metodología), fue revisar los diferentes dispositivos IoTque pueden cumplir con las condiciones antes indicadas (características), de ello, aplicar el método multi-criterio para seleccionar un dispositivo que más se ajusta a las necesidades del proyecto (indicado en la metodología, aplicando cada uno de los pasos). Para la revisión de los diferentes IoT, se tuvo en cuenta la literatura científica, así como los sitios Web de los fabricantes, de donde se obtuvo información como el set de configuración para cada elemento.

El proceso realizado fue, revisar para cada dispositivo con sus características y a través de la selección multi-criterio y el uso de las tablas comparativas descritas en la metodología, se evaluó lo siguiente y se tomó la decisión final:

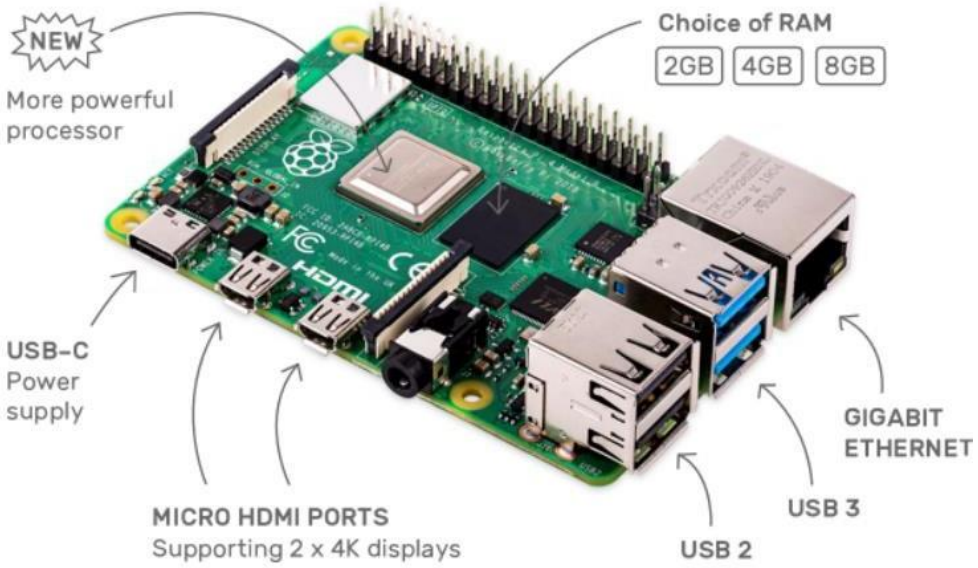
- Mejor rendimiento en CPU
- Mejor memoria RAM
- Capacidad de almacenamiento
- Permite conectividad
- Posibilidad de instalación de servicios informáticos

a) Selección del dispositivo

A continuación, se relacionan las características y funciones de los siguientes dispositivos: Raspberry PI, Arduino Yun, CubieBoard, BeagleBone Black y Pandaboard ES para realizar luego una selección de ellos:

b) Raspberry

Figura 3-12: Raspberry PI 4. Fuente [27]



La Raspberry Pi 4 Figura 3-*12*, es un prototipo reciente que toma pasos agigantados sobre sus antecesores y otros dispositivos IoT, este dispositivo IoT combina velocidad y rendimiento entre otras bondades, es un hardware de bajo costo capaz de ejecutar un sistema operativo maniobrando recursos propios como lo son memoria, CPU, Almacenamiento, red interactuando con diversos periféricos de entrada y salida. A continuación, se mencionan sus principales características:

- Cpu: Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz.
- Ram: 4GB LPDDR4
- Almacenamiento: SD/MMC/ Ranura para SDIO hasta 256 G
- Conectividad: Wifi Dual Band 2.4 GHz y 5.0 GHz IEEE 802.11b/g/n/ac, Bluetooth 5.0, BLE. Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports.
- Servicios Informáticos: si

c) Arduino Yun

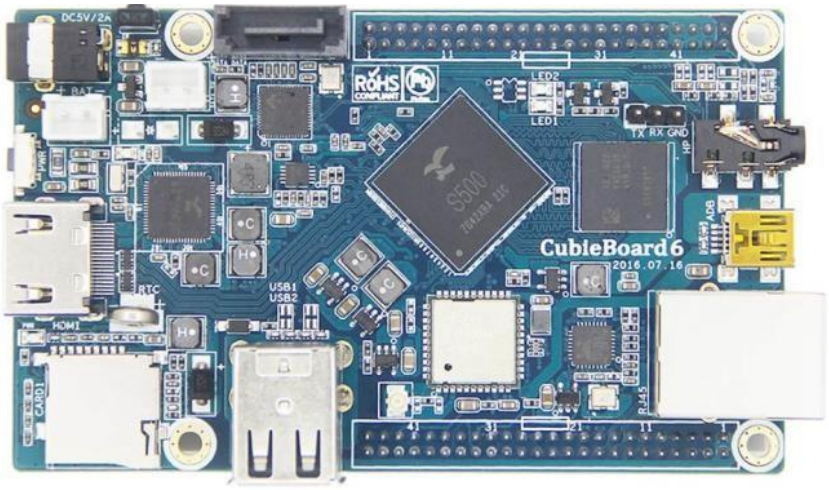
Figura 3-13: Arduino Yun. Fuente [28]

El Arduino Yun Figura 3-*13*, es una placa que ofrece grandes prestaciones relacionadas a plataformas IoT sin embargo debido a sus limitantes como su baja capacidad de procesamiento por tener un solo núcleo, esto representa poco rendimiento al momento de ejecutar tareas.

- Cpu: Atheros AR9331 - 2.4 GHz SoC, 1 Core
- Ram: 64 MB DDR2
- Almacenamiento: Micro-SD Hasta 32
- Conectividad: Ethernet: 802.3 10 / 100Mbit / s Wifi: 802.11b / g / n 2.4 GHz
- Servicios Informáticos: si

d) CubieBoard

Figura 3-14: CubieBoard6. Fuente [29]



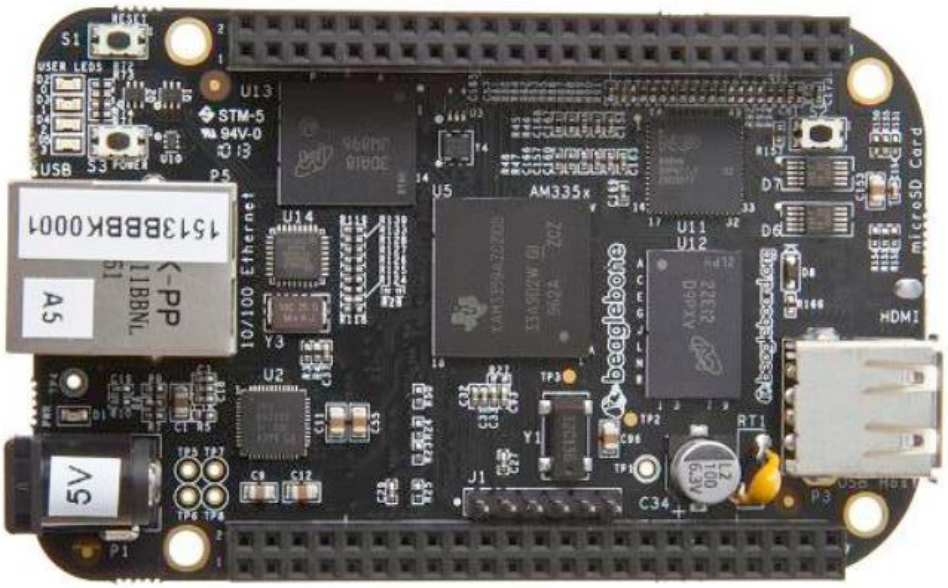
El CubieBoard6

Figura 3-14, es una placa reducida muy potente que proporciona agilidad al momento de procesar data, sin embargo, de fábrica está condicionado a soportar solo sistema operativo Linux y también viene con Android pre-instalado, generando inconvenientes en la convergencia de otros protocolos de comunicación entre dispositivos IoT.

- Cpu: ARM Cortex-A9 CPU de cuatro núcleos, 2,4 Ghz
- Ram: 2 GB LPDDR3
- Almacenamiento: Ranura para tarjeta Micro SD, hasta 32 GB
- Conectividad: Ethernet: 10M / 100M RJ45 - Inalámbrico: Wifi, Bluetooth - receptor remoto infrarrojo Philips estándar.
- Servicios Informáticos: si

e) BeagleBone

Figura 3-15: BeagleBone Black. Fuente [30]



Es fascinante encontrar placas como el BeagleBone Black Figura 3-*15*, que trabajan de forma segmentada el transporte de datos, pero teniendo en cuenta su baja capacidad de procesamiento dado a que tiene solo dos núcleos y poco ram, daría como resultados lentitud en los tiempos de respuesta.

- Cpu: AM335x 1GHz ARM® Cortex-A8, 2 Core
- Ram: 512 MB de memoria RAM DDR3
- Almacenamiento: Ranura para tarjeta 64 Micro SD
- Conectividad: TI ® WiLink ™ 8 WL1835MOD 802.11b / g / n WiFi, Bluetooth,RJ45
- Servicios Informáticos: si

f) Pandaboard

Figura 3-16: Pandaboard ES. Fuente [31]

Para este proyecto es clave tener un gran potencial de procesamiento en cuanto al sensor IoT, para el caso del Pandaboard ES (Figura 3-*16*), limitaría la exigencia de dicha característica ya que cuenta con un 50% que se define en dos núcleos, eso sumado a su poca ram serian una relación que deja a este dispositivo en desventaja.

- Cpu: Procesador OMAP4460: Dual-core 1.2 GHz ARM® Cortex ™ -A9 MPCore ™ con SMP
- Ram: 1 GB de memoria DDR2 de baja potencia
- Almacenamiento: SD/MMC/SDIO hasta 32 GB
- Conectividad: Módulo WiLink ™ 6.0 de Texas Instruments o WLAN 802.11 bgn o Bluetooth® con soporte BLE · Ethernet 10/100 802.3u a bordo
- Servicios Informáticos: si

- **Selección del dispositivo**

Los resultados de evaluar cada criterio como se indicó en la metodología (matriz pareada entre criterios), se puede apreciar en la **¡Error! No se encuentra el origen de la referencia.**6, esta matriz i ndica la puntuación pareada entre criterios donde los componentes de mayor relevancia reciben la

puntuación y como resultado observar que poder instalar servicios informáticos es el elemento más importante en la evaluación, seguido de la CPU y la memoria RAM que debe tener cada dispositivo.

Tabla 3-6: Matriz pareada entre criterios. Fuente propia.

Performance	Cpu	Ram	Almacenamiento Interno	Conectividad	Serv. Info	Matriz normalizada					Vector promedio
Cpu	1,00	5,00	9,00	5,00	0,20	0,66	0,77	0,50	0,45	0,02	0,48
Ram	0,20	1,00	3,00	5,00	0,50	0,13	0,15	0,17	0,45	0,04	0,19
Almacenamiento Interno	0,11	0,33	1,00	0,20	0,50	0,07	0,05	0,06	0,02	0,04	0,05
Conectividad	0,20	0,20	5,00	1,00	5,00	0,13	0,03	0,28	0,09	0,45	0,20
Serv. Info	5,00	2,00	2,00	0,20	1,00	3,31	0,31	0,11	0,02	0,09	0,77
Totales	1,51	6,53	18,00	11,20	6,20						

Teniendo en cuenta la escala de comparación pareada (tabla 2-1), se calificó cada criterio con respecto al otro, donde la matriz diagonal queda en uno (1), pues es el mismo criterio evaluado consigo mismo, para luego pasar a la normalización aplicando la fórmula respectiva y obteniendo el vector promedio, el cual irá a la matriz final resultante.

En la evaluación de cada criterio, se indaga cómo es el comportamiento en cada dispositivo (Tabla 3-6), así se puede observar para cada característica, qué dispositivo administra, tiene o maneja mejor ésta.

En este proceso, similar al anterior, se toma cada criterio a evaluar y se hace la comparación entre dispositivos IoT, validando cuál de estos cumple mejor la característica y en ese sentido, se evalúa la CPU, Conectividad, Almacenamiento, RAM y la posibilidad de instalar o no servicios informáticos (tabla 3-7). Se hace uso igualmente de la escala de comparación pareada (tabla 2-1), se obtiene la matriz normalizada y el valor promedio para cada dispositivo IoT evaluado en cada una de las características, obteniendo lo siguiente:

- Para el uso de CPU, se puede observar que la Raspberry PI 4 tienen mejores capacidades.
- En memoria RAM, el dispositivo Cubieboard6 tiene mejores características
- Con respecto al almacenamiento interno y poder instalar servicios informáticos, la Raspberry Pi se lleva los puntos, y finalmente.
- En términos de conectividad, la Pandaboard tiene mejores resultados.

Tabla 3-7: Matriz normalizada. Fuente propia

Sensor IoT	CPU					Matriz normalizada					Vector promedio
	Raspberry PI 4	Arduino YUN	Cubieboard6	BeagleBone Black	Pandaboard ES						
Raspberry PI 4	1,00	9,00	5,00	5,00	5,00	0,65	0,43	0,77	0,29	0,35	0,50
Arduino YUN	0,11	1,00	0,20	3,00	3,00	0,07	0,05	0,03	0,18	0,21	0,11
Cubieboard6	0,20	5,00	1,00	5,00	5,00	0,13	0,24	0,15	0,29	0,35	0,23
BeagleBone Black	0,11	3,00	0,20	1,00	0,33	0,07	0,14	0,03	0,06	0,02	0,07
Pandaboard ES	0,11	3,00	0,11	3,00	1,00	0,07	0,14	0,02	0,18	0,07	0,10
Totales	1,53	21,00	6,51	17,00	14,33						

Sensor IoT	RAM					Matriz normalizada					Vector promedio
	Raspberry PI 4	Arduino YUN	Cubieboard6	BeagleBone Black	Pandaboard ES						
Raspberry PI 4	1,00	9,00	3,00	5,00	3,00	0,2	0,2	0,7	0,2	0,3	0,34
Arduino YUN	0,11	1,00	0,11	0,11	0,11	0,0	0,0	0,0	0,0	0,0	0,02
Cubieboard6	3,00	9,00	1,00	9,00	5,00	0,6	0,2	0,2	0,4	0,5	0,40
BeagleBone Black	0,20	9,00	0,11	1,00	0,2	0,0	0,2	0,0	0,0	0,0	0,08
Pandaboard ES	1,00	9,00	0,11	5,00	1,00	0,2	0,2	0,0	0,2	0,1	0,16
Totales	5,31	37,00	4,33	20,11	9,31						

Sensor IoT	Almacenamiento Interno					Matriz normalizada					Vector promedio
	Raspberry PI 4	Arduino YUN	Cubieboard6	BeagleBone Black	Pandaboard ES						
Raspberry PI 4	1,00	5,00	0,20	9,00	9,00	0,2	0,2	0,0	0,8	0,5	0,36
Arduino YUN	0,20	1,00	0,11	0,2	1,00	0,0	0,0	0,0	0,0	0,1	0,03
Cubieboard6	5,00	9,00	1,00	0,2	1,00	0,8	0,4	0,1	0,0	0,1	0,28
BeagleBone Black	0,11	5,00	5,00	1,00	5,00	0,0	0,2	0,7	0,1	0,3	0,27
Pandaboard ES	0,11	1,00	1,00	0,2	1,00	0,0	0,0	0,1	0,0	0,1	0,06
Totales	6,42	21,00	7,31	10,60	17,00						

Sensor IoT	Conectividad					Matriz normalizada					Vector promedio
	Raspberry PI 4	Arduino YUN	Cubieboard6	BeagleBone Black	Pandaboard ES						
Raspberry PI 4	1,00	5,00	2,00	1,00	1,00	0,18	0,26	0,50	0,12	0,12	0,24
Arduino YUN	0,2	1,00	0,3	0,2	0,2	0,04	0,05	0,08	0,02	0,02	0,04
Cubieboard6	0,5	3,0	1,00	3,00	3,00	0,09	0,16	0,25	0,37	0,37	0,25
BeagleBone Black	1,00	5,00	0,3	1,00	3,00	0,18	0,26	0,08	0,12	0,37	0,20
Pandaboard ES	3,00	5,00	0,3	3,00	1,00	0,53	0,26	0,08	0,37	0,12	0,27
Totales	5,7	19,0	4,0	8,2	8,2						

Sensor IoT	Servicios informáticos					Matriz normalizada					Vector promedio
	Raspberry PI 4	Arduino YUN	Cubieboard6	BeagleBone Black	Pandaboard ES						
Raspberry PI 4	1,00	5,0	2,0	1,00	1,00	0,18	0,26	0,50	0,11	0,24	0,26
Arduino YUN	0,2	1,00	0,3	5,00	0,2	0,04	0,05	0,08	0,56	0,05	0,15
Cubieboard6	0,5	3,0	1,00	1,00	1,00	0,09	0,16	0,25	0,11	0,24	0,17
BeagleBone Black	1,00	1,00	1,00	1,00	1,00	0,18	0,05	0,25	0,11	0,24	0,17
Pandaboard ES	1,00	1,00	1,00	1,00	1,00	0,18	0,05	0,25	0,11	0,24	0,17
Totales	3,7	11,0	5,3	9,0	4,2						

Finalmente, para seleccionar el dispositivo IoT más adecuado, se realizó la matriz normalizada (Tabla 8). Como se ha indicado, para dicha matriz de resultados, la fuente de cada columna son los resultados individuales de cada matriz pareada (vector promedio) obtenida en los ítems anteriores, y se aplica la fórmula respectiva para hallar el valor total.

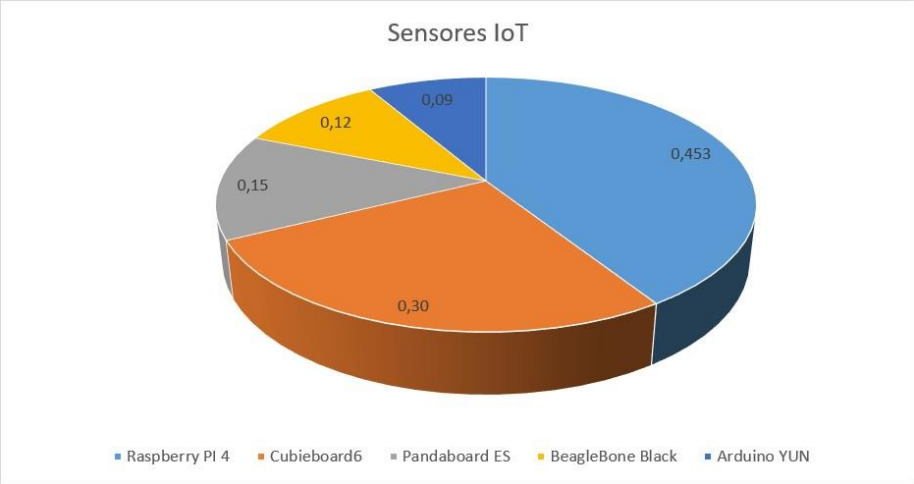
Tabla 3-8: Resultados Matriz normalizada. Fuente propia

Sensor IoT/Performance	Cpu	Ram	Almacenamiento Interno	Conectividad	Serv. Info	Total
Raspberry PI 4	0,50	0,34	0,36	0,24	0,26	0,453
Arduino YUN	0,11	0,02	0,03	0,04	0,15	0,093
Cubieboard6	0,23	0,40	0,28	0,25	0,17	0,296
BeagleBone Black	0,07	0,08	0,27	0,20	0,17	0,116
Pandaboard ES	0,10	0,16	0,06	0,27	0,17	0,148
Ponderación	0,59	0,22	0,06	0,12	0,12	

Como se puede confirmar en la Tabla 8 y Figura 3-*17*, posterior a la normalización según el método multi-criterio, se tiene un porcentaje bastante elevado en cuanto al sensor Raspberry pi 4 (0.453 como

valor total), brindando la total certeza que ese sensor sería el más apropiado para el desarrollo del proyecto, en consecuencia, el sistema operativo a ser instalado es el Raspbian (sistema operativo recomendado para las Raspberry Pi [54]).

Figura 3-17: Resultados matriz normalizada. Fuente propia.



3.1.3 Definición de los Honeypot y servicios informáticos

Para la selección de los servicios informáticos y, como se indicó en la metodología, se tomaron las diferentes soluciones Honeypot potenciales a ser instaladas en el IoT y de éstas, se realizó una caracterización considerando: servicios pre-configurados, posibilidad se ajustar la configuración, si se puede obtener OpenSource y si tiene compatibilidad con Raspberry, esto último es fundamental dada la selección de la Raspberry Pi. En la tabla 9 se pueden observar los resultados obtenidos:

Tabla 3-9: Tabla comparativa Honeypot. Fuente propia a partir de sus sitios Web.

HONEYPOT	SERVICIOS	OPCIONES DE CONFIGURACION	OPEN SOURCE	COMPATIBILIDAD CON RASPBIAN
Glashtopf	Web	No	Si	No
Kippo	SSH	No	Si	No
HoneyPI	FTP-TELNET-VNC-PSAD	Si	Si	Si
MailOney	SMTP	Si	Si	No
PentBox	Web-HTTP-Otros	Si	Si	Si
Honeyd	SSH	No	Si	No
Cowrie	SSH-TELNET	Si	Si	Si
Sippot	SIP	No	Si	No
MysqlPot	MySQL	No	Si	No
KfSensor	Varios Configurables	Si	No	No

Se puede observar que los Honeypots *"honeyPI", "PentBox"* y *"Cowrie"* son los sistemas indicados a ser instalados y configurados en la HoneyNet, adicional a esto, en la tabla 3-9 se adicionó una nueva columna de compatibilidad con el sistema operativo recomendado (Raspbian).

En consecuencia, dada la selección de los Honeypots, los servicios informáticos usados son los que ya vienen de forma predefinida: Telnet, FTP, SSH, HTTP

Ataques informáticos

Los ataques generados a la Honeynet no son exhaustivos, pues se requiere demostrar el funcionamiento de la autoconfiguración, y en ese sentido considerando los servicios informáticos que ya existen, los 4 ataques seleccionados fueron:

- **Escaneo de puertos**: Proceso ejecutado como uno de los primeros pasos por los atacantes.
- **Cracking de contraseñas en SSH o FTP**: El puerto 2222 es vulnerado por medio de un ataque de diccionario o fuerza bruta, el cual se utiliza para lograr penetrar o ingresar al sistema, el objetivo del atacante es realizar por medio de conexión SSH la utilización de un diccionario que contiene nombres de usuarios y contraseñas, para este caso se utiliza el comando: hydra –s 2222 –l root –P /root/Documentos/John.txt 192.168.1.234 –t 4 ssh.
- **Acceso no autorizado o intento de acceso en Telnet:** El puerto 2223 es vulnerado debido a que el atacante intenta realizar conexión remota con el protocolo telnet, en esta ocasión no utiliza un diccionario sino de forma individual hace la intrusión, utilizando login y password, utilizando en la consola el comando telnet 192.168.1.234 2223.
- **Posible ingreso malicioso a través HTTP:** En el puerto 8080 el atacante desde su máquina Kali Linux realiza conexión desde el navegador bajo el protocolo http://192.168.1.234:8080, utilizando la IP de la víctima relacionada al puerto. Se hace una conexión por el puerto desde la IP de Kali Linux, bajo el supuesto que dicha IP puede ser una fuente de ataque y que el sistema debe reaccionar.

3.2 Fase 2: Diseñar el método de Auto-configuración en un sensor IoT

Como se indicó, el diseño del método consta de 2 actividades:

- Implementación de la arquitectura definida: Montaje, validación de conectividad, validación de servicios y algunos ataques informáticos.
- Selección y configuración del sistema de autoconfiguración.

A continuación, se muestran los resultados de estas 2 actividades.

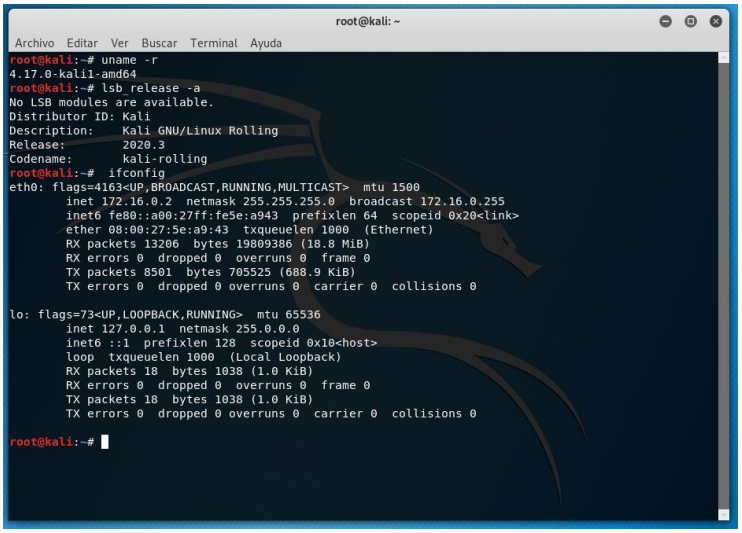
3.2.1 Implementación de la arquitectura definida

Se implementó cada uno de los componentes de la arquitectura definida en la fase 1, un sistema híbrido combinando máquinas virtuales con componentes físicos (IoT), utilizando virtualBox como

herramienta de virtualización, emulando el ecosistema informático con las siguientes configuraciones base:

- **Kali Linux:** Los ataques fueron realizados en un ambiente controlado desde una maquina con sistema operativo Kali Linux que cuenta los últimos upgrade y update en su SO, esta máquina es total mente funcional ya que responde a un ambiente productivo y real que utiliza un atacante para ejecutar intrusiones de manera premeditada, en la Figura 3-*18*, se puede visualizar la identidad de este host con su respectivo direccionamiento.

Figura 3-18: Configuración base del componente Kali Linux. Fuente propia.



Firewall: Las configuraciones realizadas en el corta fuego fueron habilitar dos tarjetas de red con la IP 192.168.1.269 que obedece a la tarjeta enp0s3 y la IP 172.16.0.1 con la tarjeta enp0s8, en la Figura 3-*19* yFigura 3-*20* se visualiza los dos segmentos que permiten la interpelación en la red, nateando la IP del atacante al momento de realizar la conexión o intervención hacia la Honeynet.

Figura 3-19: Direccionamiento del Firewall. Fuente propia

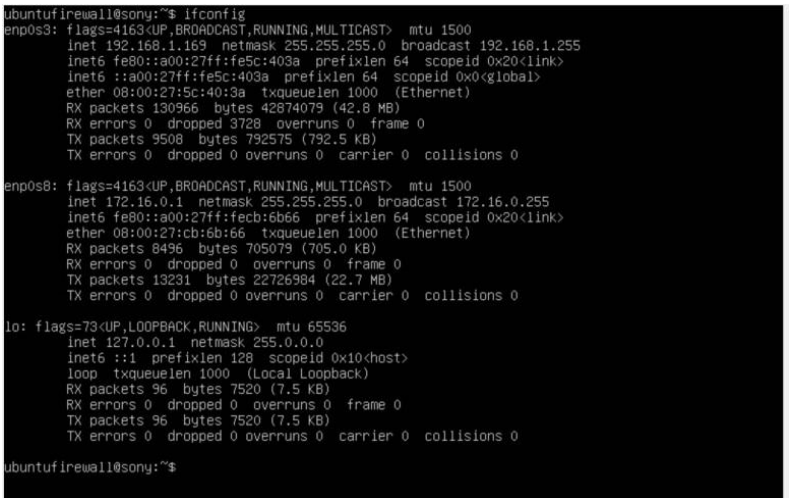
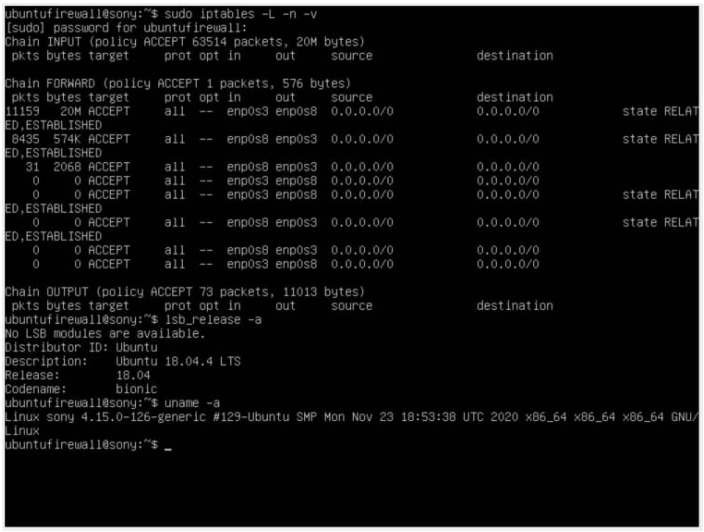


Figura 3-20: Configuración y Kernel del SO en el Firewall. Fuente propia.



Honeypot: Para la selección de los Honeypots se tuvo en cuenta cual fue el dispositivo IoT a utilizar (para el caso, acorde al resultado anterior, es Raspberry Pi) y en ese sentido, teniendo en cuenta lo indicado por el instituto SANS [49], donde declara los servicios informáticos más vulnerados y atraídos por el atacante, se considera relevante escoger varios Honeypots (configurando así una Honeynet) que cumplan con dichos servicios y además tenga la capacidad en su núcleo de adaptabilidad en cuanto a los recursos del dispositivo IoT, esto relacionado con su sistema operativo Raspbian y capacidades. En la tabla 9 se observa varios Honeypots de iguales o similares características que cumplen con estos servicios informáticos.

Se puede apreciar que existen diferentes Honeypots que cubren parte de los servicios y protocolos, en donde la mayoría son OpenSource, lo que da una buena ventaja para el desarrollo de proyecto. Así mismo, solo 3 son compatibles con Raspbian (Sistema Operativo recomendado para RaspBerry),

los demás son instalables en otros sistemas. En ese sentido, los sistemas señuelos configurados para formar la HoneyNet y que pueden ser instalados en una RaspBerry Pi (tabla 9) son:

- Honeypi
- PentBox
- Cowrie

En la Figura 3-*21*, se puede visualizar la configuración del Honeypot Cowrie, un Honeypot que permite emular los servicios Telnet y SSH, posibilitando al sensor la capacidad de registro de todos los movimientos realizados en cuanto a uno o más ataques realizados por fuerza bruta o diccionario y en la Figura 3-22, se observa Cowrie habilitado prestando los servicio SSH por el puerto 2222 y el servicio telnet por el puerto 2223 los cuales se encuentran 100% operativos y en modo escucha.

Figura 3-21: Configuración Honeypot Cowrie. Fuente propia.

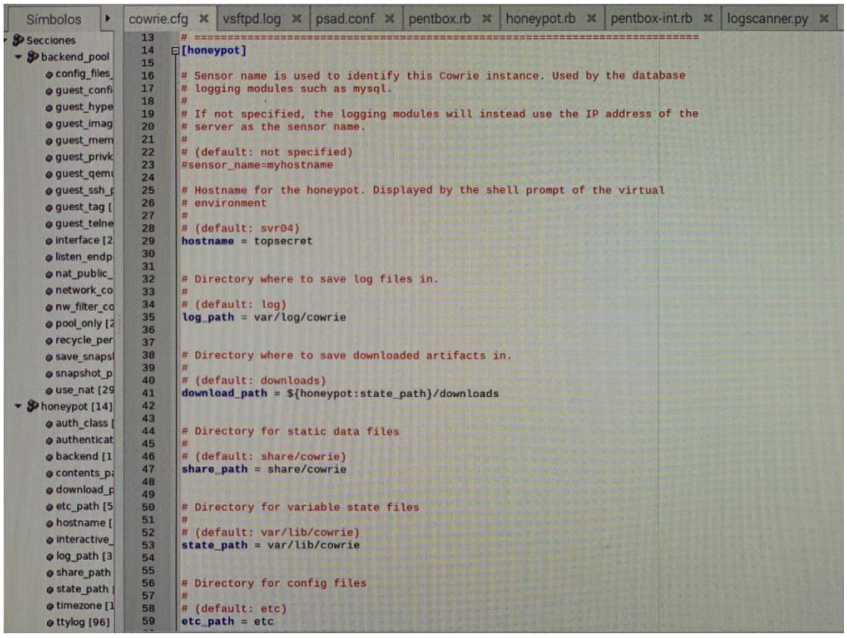
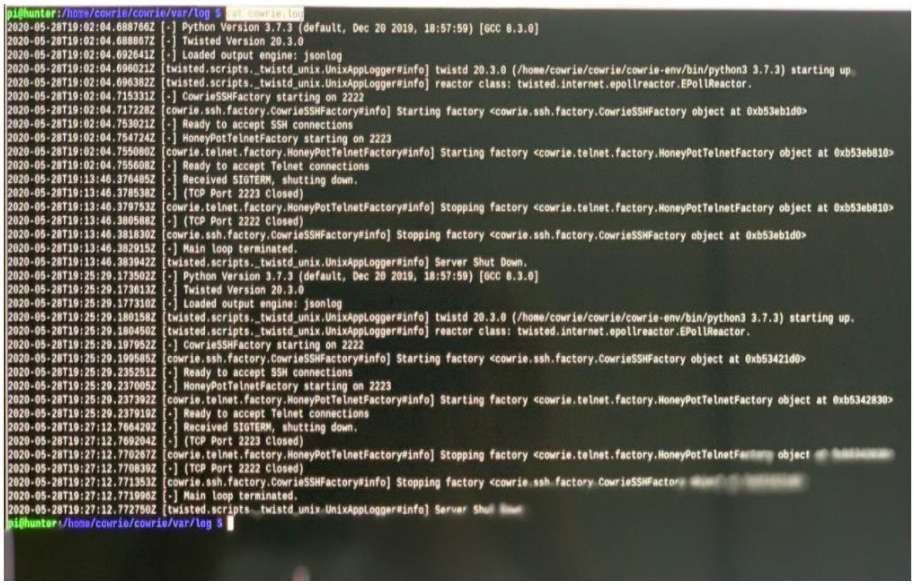


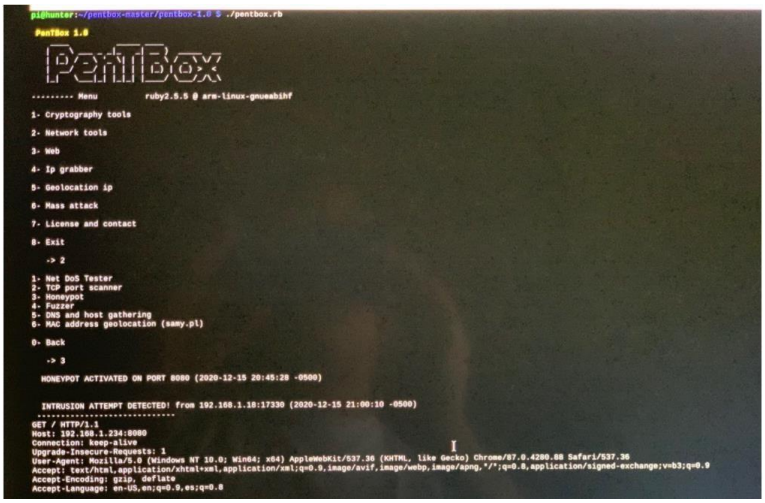
Figura 3-22: Honeypot Cowrie habilitado. Fuente propia.



Para el caso de Pentbox, como resultado de la configuración del servicio está utilizando el protocolo HTTP en el puerto 8080, en la

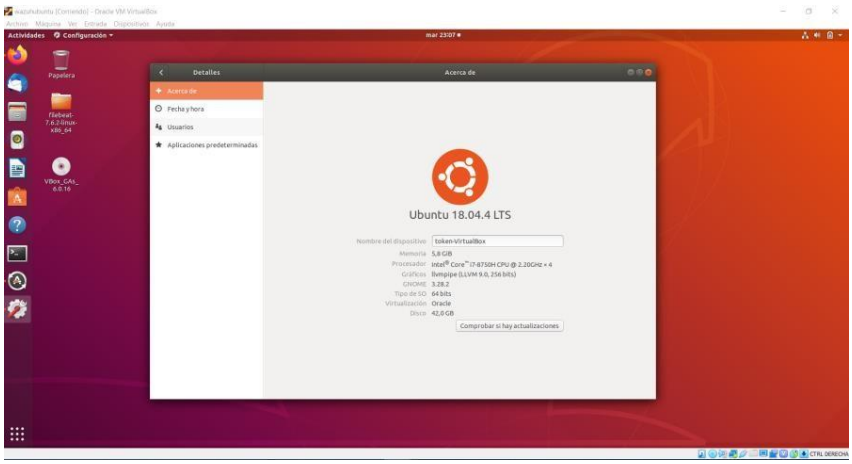
Figura 3-*23*, se observa su configuración, el menú de operaciones y confirma que el Honeypot se encuentra activo, frente a este escenario se lanza un ataque desde la máquina con IP 192.168.1.18 demostrando que el Honeypot es totalmente funcional permitiendo atrapar las huellas del atacante bajo el protocolo HTTP en el servicio Web.

Figura 3-23: Configuración de Honeypot Pentbox. Fuente propia.



Wazuh es un potente HIDS que permite la detección de amenazas, el monitoreo de la seguridad, las respuestas a incidentes y cumplimiento normativo, a medida que el sensor indique cambios en su patrón de comportamiento, Wazuh concede el monitoreo en tiempo real generando alertas que indiquen anomalías sospechosas. A continuación, se muestra los resultados obtenidos con este componente. En la Figura 3-*24*, se puede observar que la instalación de la solución fue en una máquina Ubuntu 18.04.4 LTS y diversas propiedades en virtualbox.

Figura 3-24: Propiedades maquina Wazuh Server. Fuente Propia.



Dentro de las configuraciones de Wazuh se le asignó IP fija para tener una comunicación estable con el Sensor IoT, brindando la IP 192.168.1.201 con el canal de comunicación del servicio por el puerto 5601, en la

Figura 3-*25*, Figura 3-*26* y Figura 3-27, se puede visualizar eventos de seguridad donde la solución brinda un informe detallado de las conexiones, alertas, autenticaciones fallidas, autenticaciones exitosas.

Figura 3-25:Monitoreo de Wazuh sobre el Dispositivo IoT. Fuente Propia.

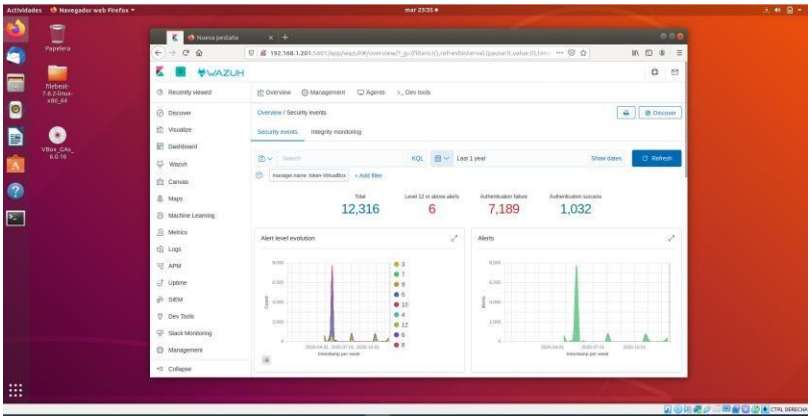


Figura 3-26: Monitoreo de Wazuh sobre el Agente. Fuente Propia

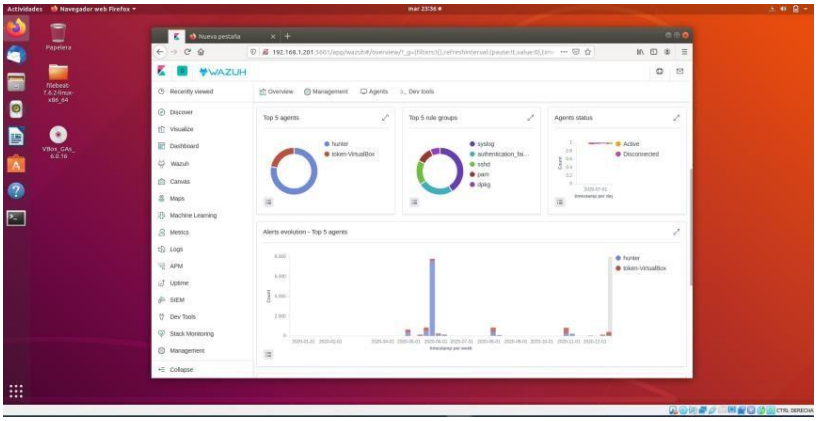
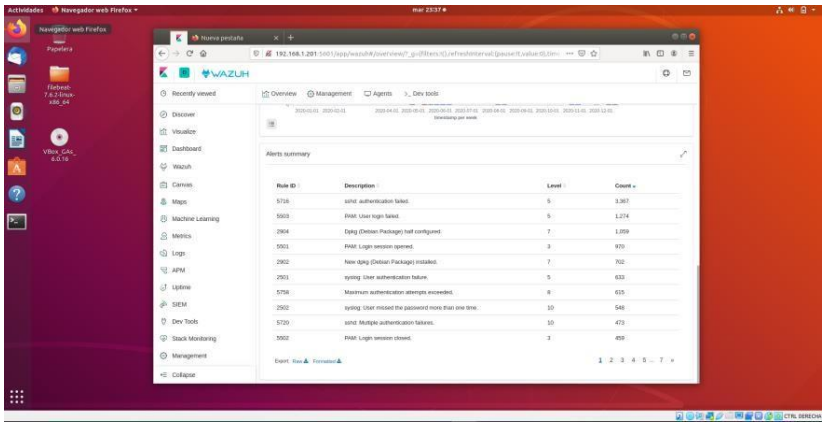


Figura 3-27: Resumen estadístico de eventos de seguridad. Fuente Propia.



Esta potente herramienta permite la segmentación de los tiempos para su análisis, es decir que cuenta con la facilidad de identificar líneas de tiempo y clasificar los tipos de alertas de manera detallada en la *Figura 3-28* y ***¡Error! No se encuentra el origen de la referencia.*** se puede identificar el nombre del sensor IoT que para el caso es Hunter con IP 192.168.1.234 y también se identifica las variaciones realizadas desde su kernel y su sistema operativo, también indica que el agente se encuentra funcionando de manera exitosa, esto entre otras bondades de información que se refleja de manera clara.

Figura 3-28: Log Wazuh 1. Fuente Propia

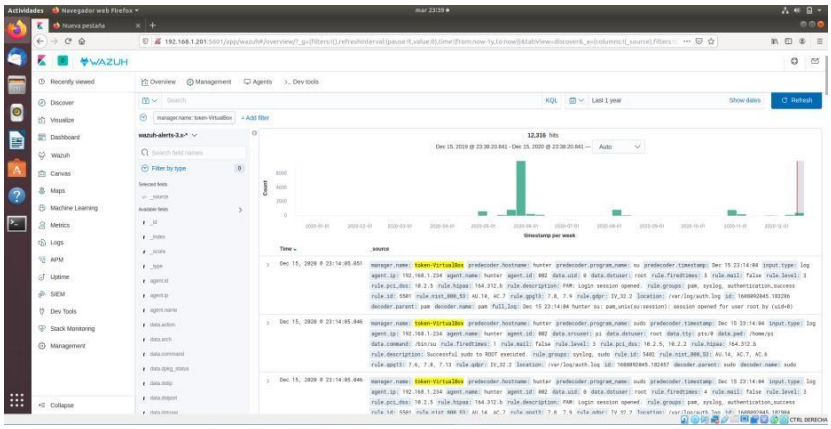
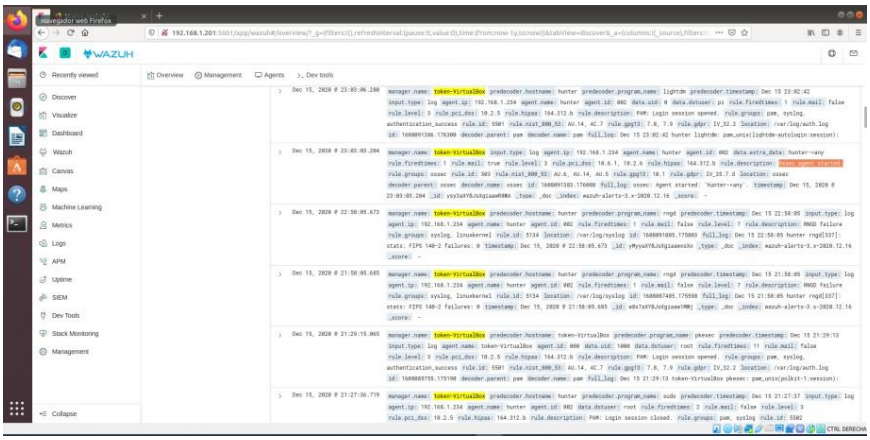


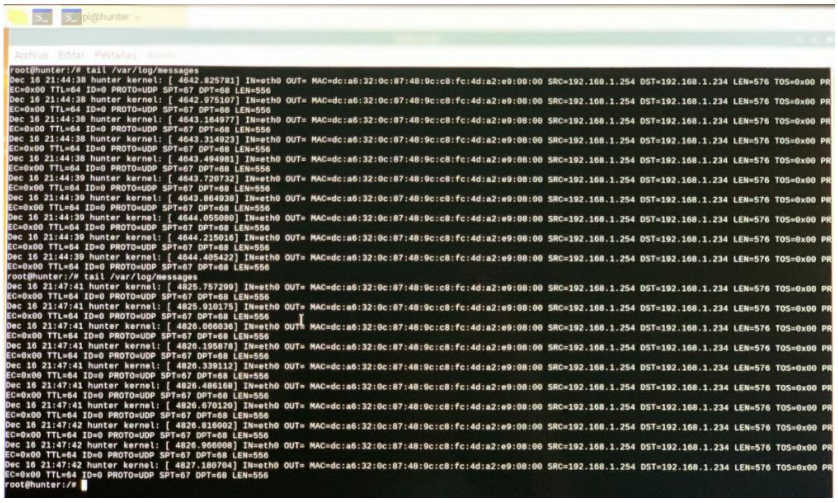
Figura 3-29: Log Wazuh 2. Fuente Propia



Lectura de Log y visualización: Todos los Honeypots fueron configurados para dirigir sus logs hacia la ruta /var/log/messages, es allí donde se guardan las alteraciones o comportamientos sospechosos hacia el dispositivo IoT, utilizando el comando TAIL como se puede visualizar en la Figura 3-*30*, se refleja los actores participantes en el ecosistema Honeynet como lo es el host declarado como hunter , también se identifica el Honeypot HoneyPI donde se destaca su componente Psad que es el IDS que permite alertar sobre comportamientos hostiles en la red, siendo víctima de algún escaneo no autorizado u otro tipo de acción hostil, teniendo en cuenta que la IP del sensor es la 192.168.1.234, puerta de enlace 192.168.1.254 y también la herramienta Wazuh con IP 192.168.1.201, estas son listas blancas declaradas que evitan generar falsos positivos ya que sus

configuraciones hacen parte de la red.

Figura 3-30: Visualización de logs en el IoT. Fuente Propia.

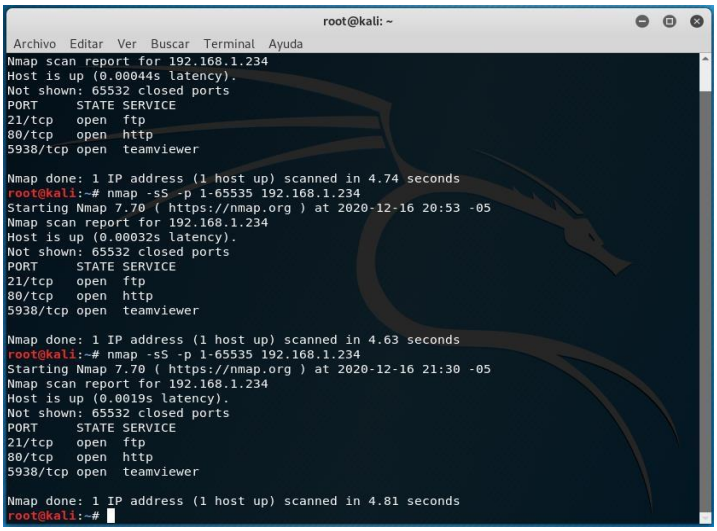


Comprobación de funcionamiento: pruebas unitarias

En la red Honeynet todo se convierte en algo tan sensible que es capaz de estar alerta para captar comportamientos fuera de lo común, esta red trampa es inducida a tener mapeada todo tipo de acción que quiera comprometer sus servicios o sistema operativo, en las *Figuras 3-31*, *3-32*, *3-33* y *3-34* se utiliza como ejemplo desde una máquina externa el envío de Nmap y ping para un sondeo del comportamiento del sensor IoT, reflejando un alto grado de sensibilidad al delatar a la otra máquina que está escaneando al dispositivo.

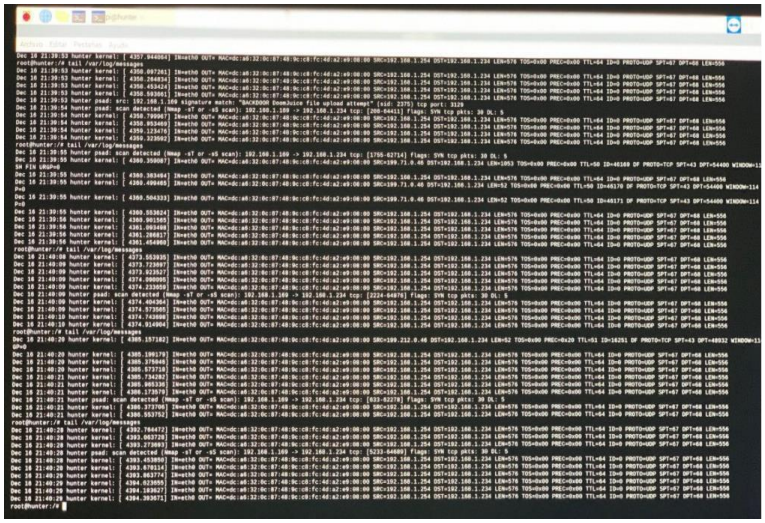
El objetivo de esta prueba unitaria es validar que los diferentes componentes tienen conectividad y se está haciendo un registro de los eventos, o sea que sea funcional la Honeynet y la simulación del atacante.

Figura 3-31: Nmap desde Kali Linux hacia el sensor IoT. Fuente Propia



En la figura 31 se puede observar la ejecución de nmap utilizando diferentes flags para validar la activación en la IP 192.168.1.234

Figura 3-32: Detección del Nmap en el sensor IoT. Fuente propia.



En la *Figura 3-32* se tiene el log de la detección de *nmap* desde una maquina atacante, con ello, se valida de manera unitaria que el dispositivo recibe y guarda los rastros de la ejecución.

Otra prueba realizada fue la ejecución de un comando ping simple (*Figura 3-33*), para validar conectividad y que el sistema capture el evento:

Figura 3-33: Ping desde Kali Linux hacia el sensor IoT. Fuente Propia

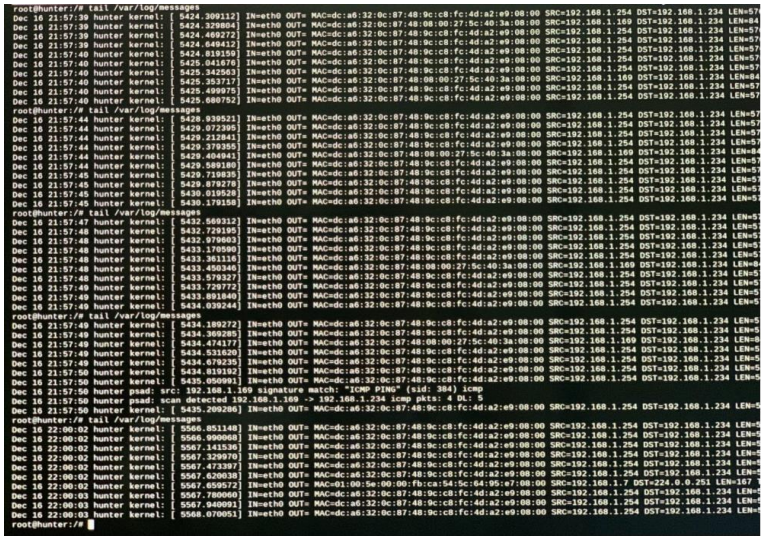
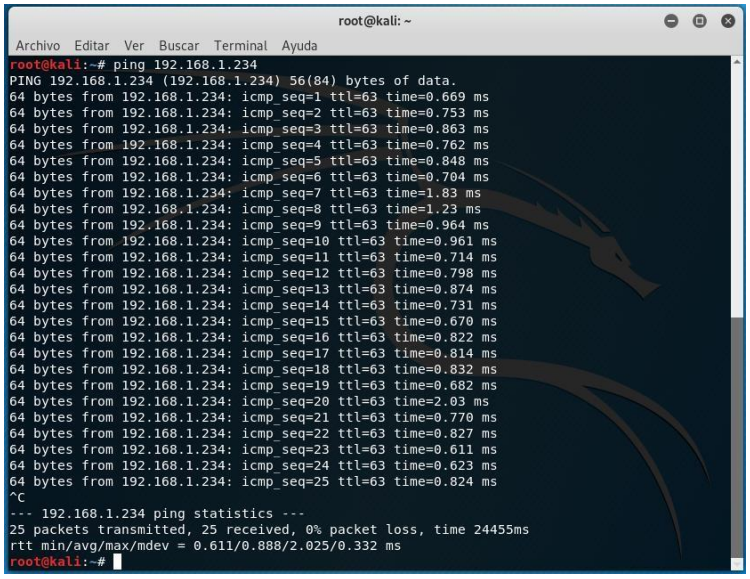


Figura 3-34: Detección desde en el sensor IoT. Fuente propia



En la Figura 3-34 se puede observar las diferentes detecciones a través de los logs del sistema.

3.2.2 Definición y construcción del método de autoconfiguración

Para el diseño del método se indagaron las diferentes tecnologías en la cual se podría hacer la programación de la autoconfiguración, para lo cual:

a) Machine learning:

Basa su conocimiento en la IA permitiendo a un sistema aprender de los datos en lugar del aprendizaje mediante la programación explicativa, teniendo en cuenta que no es un proceso sencillo ya que al interior su algoritmo ingiere datos de entrenamiento. Por ejemplo, un algoritmo predictivo

creará un modelo predictivo, cuando se proporcione el modelo predictivo con datos, recibirá un pronóstico basado en los datos que entrenaron al modelo.

Machine learning tiene un aprendizaje iterativo que permite modelos a entrenar con conjuntos de datos antes de ser implementados. Este proceso iterativo de modelos conduce a una mejora en los tipos de asociaciones hechas entre los elementos de datos. Debido a su complejidad y tamaño, estos patrones podrían fácilmente ser pasados por alto por la observación humana.

b) Deep learning:

Hace parte de la rama de machine learning, ya que es un método especifico que incorpora las redes neuronales en capas sucesivas para aprender de los datos de manera iterativa, muy útil cuando se trata de aprender patrones de datos no estructurados. Las redes neuronales complejas de Deep learning están diseñadas para emular cómo funciona el cerebro humano.

c) Python:

Este lenguaje de programación orientado a objetos de propósito general, funciona con fines multiparadigma ya que soporta varios modelos de desarrollo de la programación, es interpretado debido a que en su actuación permite al interior la traducción del código según la exigencia del leguaje maquina a medica que sea necesario, su ejecución dinámica autoriza la mutación, es decir de manera versátil permita la transformación de variables.

Este lenguaje de programación es ideal para scripting y desarrollo rápido de aplicaciones en muchas áreas. Los instaladores de Python para la plataforma Windows frecuentemente incluyen la biblioteca estándar completa y suelen también incluir muchos componentes adicionales. Para los sistemas operativos tipo Unix Python suele ser provisto como una colección de paquetes, así que puede requerirse usar las herramientas de empaquetado disponibles en los sistemas operativos para obtener algunos o todos los componentes opcionales.

Python es fácil de usar, siendo un lenguaje real de programación, el cual ofrece mejor estructura y soporte para programas extensos que scripts desarrollados en shell o batch. Permite modular uno o varios programas y por lo tanto el código puede ser reusado en futuros desarrollos. Tambien es una muy buena opción de lenguaje para la ciencia de datos, y no solo en el nivel de entrada. Gran parte del proceso de ciencia de datos gira en torno al proceso ETL (extracción-transformación-carga). Esto hace que la generalidad de Python encaje perfectamente [39].

En la Tabla 3-9, se ilustra un cuadro comparativo de las herramientas mencionadas, en donde se destaca alguna de sus ventajas y desventajas.

Tabla 3-9: Tabla comparativa ventajas y desventajas. Fuente propia.

HERRAMIENTAS	VENTAJAS Y DESVENTAJAS					
	Desarrollo más rápido.	Procesamiento mas rapido.	Auto-configuración de procesos.	Reduce el error humano.	Mayor cobertura por profesionales cualificados.	Menos tiempo de dedicacion.
Machine learning	✗	✗	→	→	✗	✗
Deep learning	✗	✗	→	→	✗	✗
Python	→	→	→	→	→	→

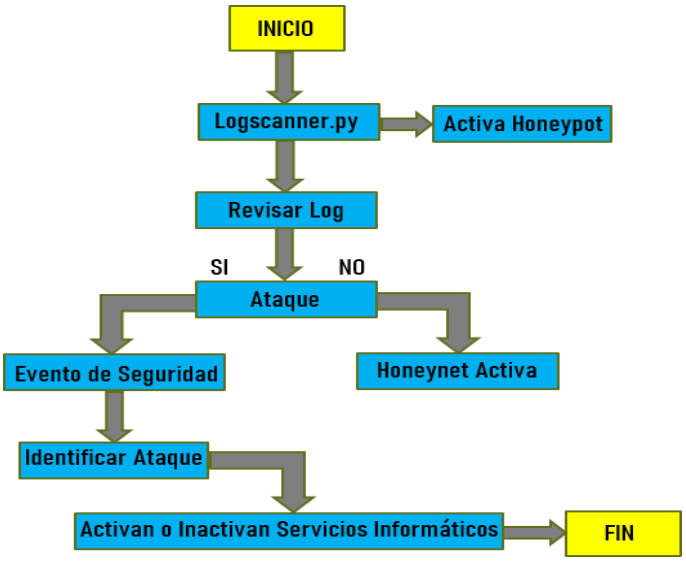
Los métodos para auto-configurar una o varias tareas en un sistema informático como: Machine Learning, Deep Learning que emplean aprendizaje automático y profundo, centran su disciplina en el ámbito de la inteligencia artificial, otros casos no menos importantes son los algoritmos que se direccionan en la construcción del desarrollo, mediante lenguajes de programación como lo es python.

De acuerdo a lo anterior, la mejor opción para la construcción del método de auto-configuración fue desarrollar un script con Python, dado que presenta mejores condiciones para el alcance planteado y el tiempo estipulado en el proyecto de grado, pues es una de las formas simples de leer logs y ejecutar cambios en los servicios (activación y desactivación).

Por otro lado, en consideración de la arquitectura definida (figura 11), se configuró cada elemento dentro de la red, para lo cual, se definió las mejores opciones de Honeypot a ser instaladas dentro de la honeynet.

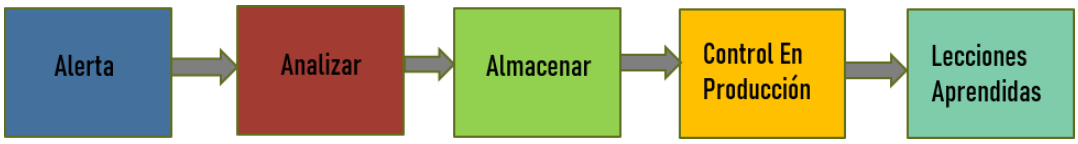
Como lo indica la figura 35, se muestra la línea de tiempo de forma general de cómo funciona el script, y cómo se ejecutaron las pruebas, en donde se contempla las fases del método de alerta de eventos de seguridad, de nada serviría la implementación de una Honeynet si no se recolecta la información ya que este posee gran capacidad de almacenamiento de log, y como consecuencia utilizarla con fines productivos, esta información será muy valiosa para el equipo de respuesta ante incidentes o al SOC.

Figura 3-35: Método de alerta de eventos de seguridad. Fuente propia.



En la implementación del script Logscanner.py se validaron parámetros a tener en cuenta que generan valor agregado en sus logs, dentro de su algoritmo se contempla la oportunidad de generar resultados que sean fáciles de analizar al momento de la toma de decisiones. Por lo anterior, se generó un flujo de proceso Figura 3-36 de como funcional en si el sistema.

Figura 3-36: Flujograma de proceso. Fuente propia.



A continuación, se explica cada uno de los pasos.

Alerta: La alerta es el mecanismo detonador de todo el engranaje, ya que tiene la responsabilidad de avisar que hay un evento de seguridad tratando de vulnerar el ecosistema informático.

Analizar: en esta fase el oficial de seguridad tiene la responsabilidad de ejecutar un análisis exhaustivo que lo conlleve a detectar una o varias actividades sospechosas que indiquen que la integridad del sistema ha podido ser vulnerada por algún tipo de intrusión, tendrá la oportunidad de capturar y visualizar los metadatos que son el resultado de la interacción entre el atacante y el dispositivo IoT dentro de la Honeynet. Es allí un escenario que permite identificar patrones como ip, fecha y hora, tipo de estrategia utilizada por el atacante entre otras particularidades que facilita la correlación de eventos.

Almacenar: La captura de todos los movimientos y estrategias empleadas por el atacante en nuestra

Honeynet nos revela sus técnicas y motivaciones, por eso es de vital importancia la custodia y almacenamiento de cada etapa ocurrida posterior al evento de seguridad.

Control En Producción: la Honeynet es una red pasiva que está a la espera de ser vulnerada o comprometida, esta red no tiene riesgos de sufrir daños ya que no contiene información sensible, su único fin es ser un blanco atractivo para todo tipo de atacante que desea concentrar sus esfuerzos en derivar los perímetros de seguridad en la arquitectura. Este ecosistema IoT pretende actuar como método de alerta temprana que permita diseñar controles basados en la experiencia, para en una etapa posterior aplicar medidas de seguridad en la red real es decir en la productiva. Conocer la taxonomía del ataque en un ambiente de prueba que simula ser real, brinda cierta ventaja frente al posible ataque que se presente en la arquitectura real.

Lecciones Aprendidas: El propósito de esta última fase, es tener un cuadro de mejora año tras año, es de vital importancia llevar a cabo acciones que nos ayuden a combatir los errores del pasado, con el fin que los mismo no se repitan. Una buena práctica es recopilar y aprender del atacante.

Un buen ejemplo para la ejecución de las lecciones aprendidas, es realizar un cuestionario del incidente que contenga, al menos, la siguiente información:

- ¿Cuándo fue detectado el problema por primera vez?
- ¿Cómo fue contenido y erradicado?
- ¿Qué tareas se realizaron durante todo el proceso?
- ¿Cuál fue el alcance del incidente?
- ¿Cuáles son las oportunidades de mejora?

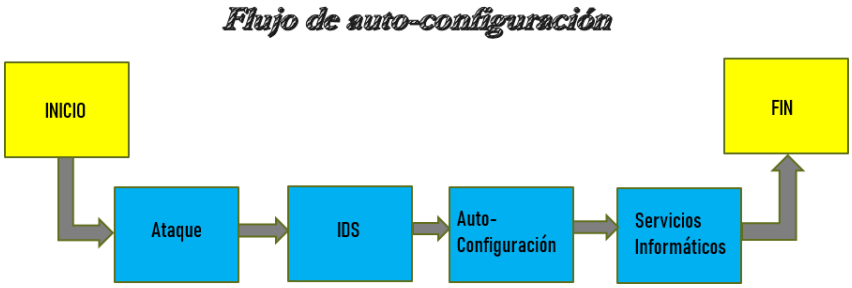
El método implementado, como se indicó, fue basado en un script desarrollado en el lenguaje de programación Python, este algoritmo tiene la responsabilidad de auto-configurar el sensor IoT convirtiéndolo en un arma letal para la atracción de atacantes que de forma nativa les revelen sus estrategias de ataques basados en los parámetros mencionados en este documento, y así tener la capacidad de identificar los diferentes tipos de ataques hacia el dispositivo. A medida que un atacante va descubriendo servicios, el sistema activa o desactiva éstos, de modo que, si el servicio no es revisado por un atacante, se desactiva y si está siendo validado, se le mostrará.

Implementación del método

El sistema propuesto consta de cuatro tareas consecutivas como lo indica la Figura 37, el flujo determinado para la Auto-configuración es relevante en su proceso inicial con la primera tarea activada por el comportamiento hostil del atacante, ya sea intentando establecer una conexión por vía SSH, telnet o servicio web mediante el protocolo HTTP, en la segunda tarea se tiene instalado en su arquitectura un Honeypot denominado HoneyPI, el cual cumple la función de centinela ya que en su configuración posee un potente IDS llamado Psad, el cual tiene como objetivo monitorear

permanentemente el comportamiento del sensor IoT llevando sus logs de eventos hacia la ruta /var/log/messages y cuando hay una situación anómala que incurre en un evento de seguridad, este lo percibe el script de auto-configuración denominado logscanner que se encuentra ejecutándose de forma permanente en segundo plano al interior del sistema operativo, la tercera tarea se ve impactada por el escenario anterior ya que constantemente se encuentra leyendo los logs ubicados en la carpeta messages, debido a esto, el script tiene la capacidad de identificar qué tipo de ataque el sensor está siendo sometido, es decir ataque de diccionario, fuerza bruta por ssh, telnet o vía http, En la cuarta tarea de forma visual se puede apreciar el panorama de activar e inactivar servicios informáticos, los cuales se manifiestan indicando los puerto 2222, 2223, 8080, servicios de apache y vsftpd activos e inactivos de forma aleatoria, el sistema activa otros servicios para que el atacante los puede ver y así, la Honeynet puede obtener mejores resultados teniendo siempre la atención y curiosidad del atacante mostrando mayor interés sobre las capacidades de la Honeynet.

Figura 3-37: Metodología de Auto-configuración. Fuente propia.



En términos generales, el flujo de auto-configuración inicia con un ataque informático sobre un servicio que este activo, el cual es detectado por la HoneyNet (a través del IDS) y resguardado en los sistemas Logs, una vez detectado el ataque, el software de auto-configuración valida el servicio atacado y activa otros servicios informáticos disponibles, con ello, termina el flujo respectivo.

En consecuencia, con el desarrollo del código en Python (Anexo 1), el programa valida constantemente los diferentes logs del sistema en busca de posibles ataques informáticos, para ello, se deja un demonio activo en la Raspberry que, a través de la lectura de los registros logs, busca palabras claves, las cuales son generadas por los Honeypot cuando se ejecuta ciertos ataques informáticos y guardadas de manera centralizada en el *messages*:

- "Login Attempt" and "Succeeded"
- "Untrusion Attempt Detected"
- “HoneypotSSHTransport”

Una vez el programa detecta algunas de las palabras, revisará los servicios y subirá el que se requiera, así le deja disponible una nueva oportunidad a los atacantes. La activación consiste en enviar una orden al Honeypot que tiene el servicio para que lo active o desactive. El proceso de desactivación se hace cuando, el atacante está en un servicio (por ejemplo, el SSH) y no en el HTTP (puerto 8080), entonces

el sistema baja el puerto Web respectivo, y así de forma aleatoria para todos los servicios. Finalmente, el diseño del método fue posible diagramarlo y realizar una prueba unitaria de su funcionamiento.

3.3 Fase 3: Ejecutar y documentar las pruebas

Para la ejecución de las pruebas como se indicó en la metodología, se realizaron diferentes pruebas de seguridad simulando los siguientes ataques informáticos definidos en los resultados de la fase 1:

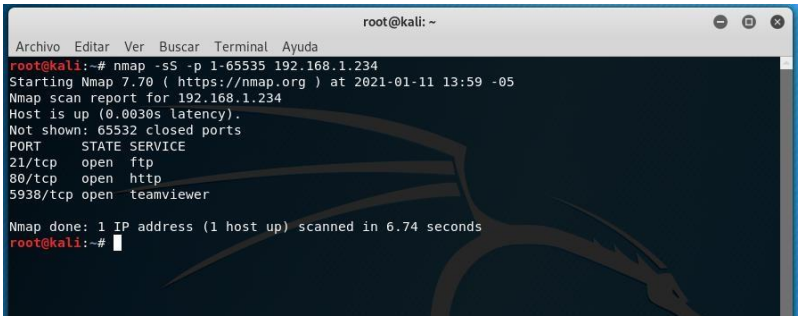
- Escaneo de puertos con *nmap*
- Cracking de contraseñas en SSH o FTP
- Acceso no autorizado en Telnet
- Posible ingreso malicioso a través HTTP

Para obtener los resultados finales acordes a la investigación, se tuvo como fin aprovisionar Honeypots idóneos para lograr los resultados planteados (acorde a la arquitectura definida en la fase 1), es bueno precisar que en la implementación del red Honeynet se obtuvo la instalación de los Honeypot Pentbox que como anteriormente se ha mencionado, es una herramienta con un portafolio de servicios que ofrece como herramienta un Honeypot direccionado a indicar alertas de intentos de conexión cuando existe una conexión en el protocolo http en el puerto 8080, y por otro lado el Honeypot Cowrie que presenta los servicios SSH y telnet en los puertos 2222 y 2223, en este orden de ideas es necesario tener en cuenta que si el atacante agrede los servicios antes mencionados, es allí donde se produce la activación o inactivación de los servicios.

Utilizando la herramienta NMAP desde la máquina del atacante, teniendo en cuenta que este Ciberdelincuente ya tiene la IP de su objetivo, se puede visualizar en la *Figura 3-38* que se hace un escaneo en general hacia la maquina víctima, es decir host 192.168.1.234 y para el escaneo se utiliza la siguiente sintaxis Nmap –sS que es enviar un SYN, es una técnica rápida, fiable y relativamente sigilosa, posteriormente -p 1-65535 192.168.1.234 esa es la indicación del rango de los puerto que será escaneados seguidos de la IP víctima o host destino. Es claro que actualmente solo se puede ver tres puertos TCP con sus respectivos servicios en estado abierto.

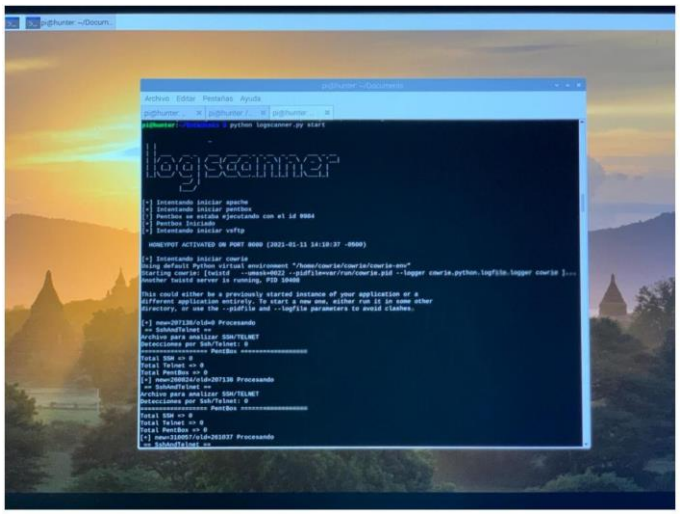
El corazón de la auto-configuración de servicios se encuentra alojado en el desarrollo del script en python, que se puede apreciar en el anexo 1, en la *Figura 3-38* evidencia la simulación del accionar de un atacante, que a su vez revela un informe que indica los servicios activos como primer paso de la auto-configuración, en segunda instancia se observa los tipos de ataque según el Honeypot el cual sería vulnerado.

Figura 3-38: Escaneo de puertos. Fuente propia.



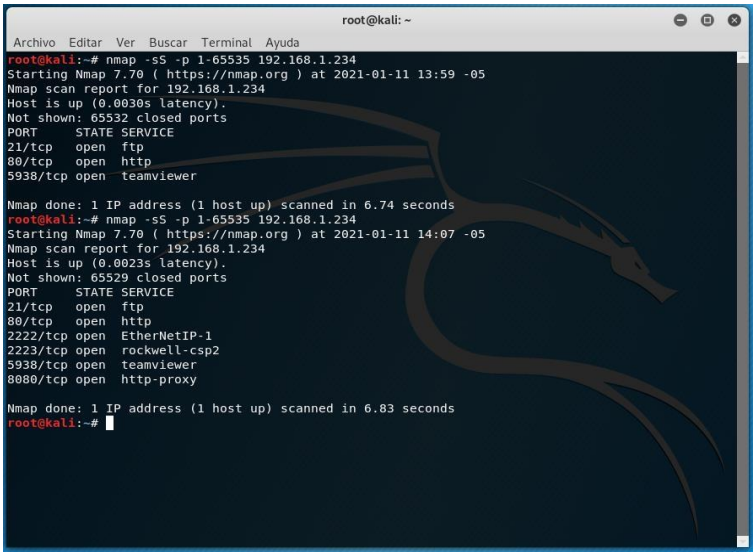
En el sistema de autoconfiguración se visualiza que un evento llegó y este se muestra en pantalla (*Figura 3-39*)

Figura 3-39: Ejecución de Script Logscanner.py



Posterior a la ejecución del script logscanner.py (*Figura 3-39*), inmediatamente se activan servicios que son atractivos en esta red señuelo, se ejecuta de nuevo una serie escaneos de puertos y en la figura 40 se realiza un comparativo de los servicios habilitados antes de la ejecución del script y posterior, demostrando la diferencia.

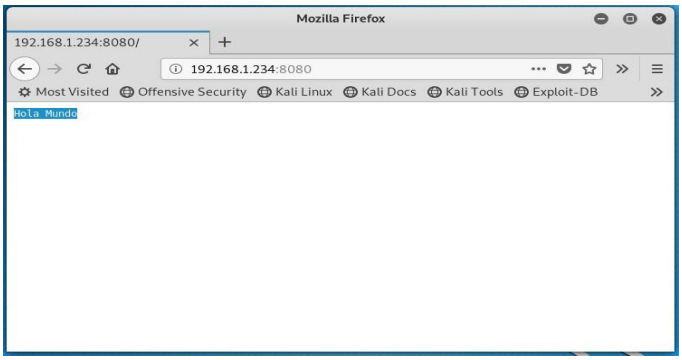
Figura 3-40: Servicios informáticos activos: Fuente propia.



Se puede observar que en el primer escaneo aparecen los puertos y servicio 21 (ftp), 80 (http) y 5938 (teamsviewer), en un segundo escaneo y luego de que el *script* de autoconfiguración hace su trabajo, se visualiza otros servicios que el atacante podría eventualmente animarse a atacar (habilita, entre ellos, el 2222-ssh y 2223-telnet).
Se puede ver entonces que, cuando el atacante genera un primer escaneo, el cual es detectado por la HoneyNet como un posible ataque, esta se reconfigura activando otros servicios, de modo que el atacante pueda observar otros grupos de protocolos que pueden ser atacados.

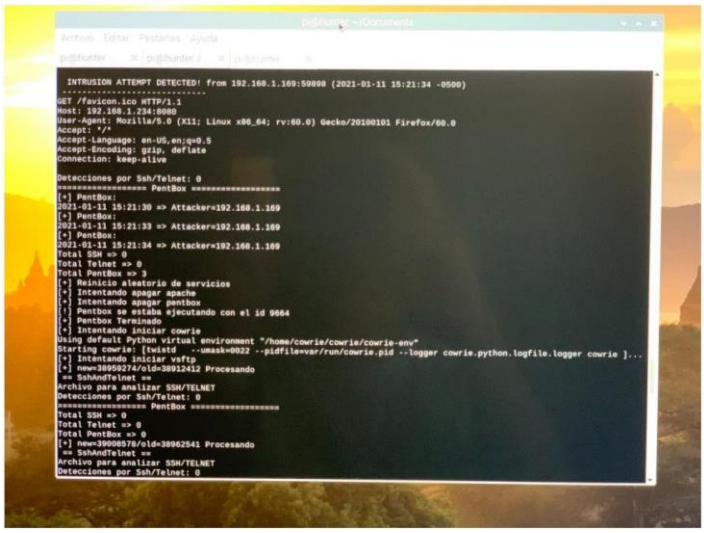
Con la validación inicial y teniendo los servicios informáticos activos se hace el despliegue de ataques, en primera instancia se dará lugar a la utilización del Honeypot Pentbox el cual tiene como utilidad levantar el puerto 8080, que obedece al servicio web, en este escenario el atacante desde su máquina Kali Linux pretende realizar conexión desde el navegador bajo el protocolo http://192.168.1.234:8080, utilizando la IP de la víctima relacionada al puerto como se observa en la *Figura 3-41*.

Figura 3-41: Intento de conexión web. Fuente propia.



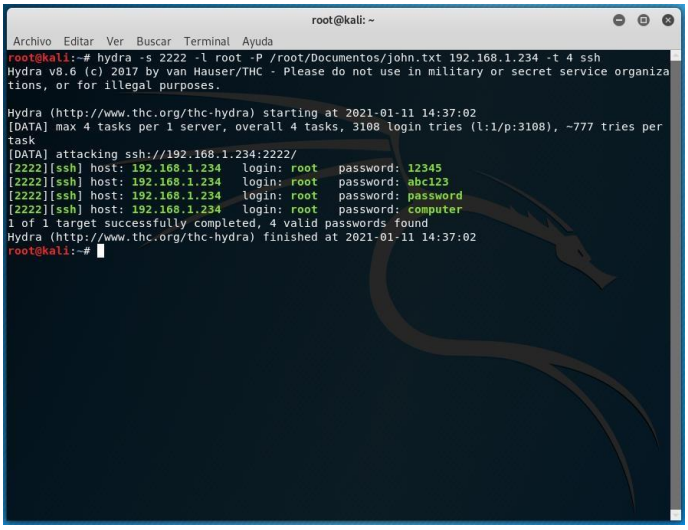
Tal como se aprecia en la *Figura 3-42*, el Honeypot Pentbox en medio del monitoreo que está realizando en el puerto 8080, actúa generando la alerta y atrapando la IP del atacante, hecha, hora y otras huellas que son valiosas para el aprendizaje, de forma simultanea se observa que al interactuar con el atacante activa e inactiva servicios informáticos de forma aleatoria.

Figura 3-42: Alerta de ataque por el puerto 8080. Fuente propia.



En otro de los escenarios se tiene el servicio SSH que es custodiado y ejecutado por el Honeypot Cowrie, este servicio por el puerto 2222 es vulnerado por medio de un ataque de diccionario o fuerza bruta, el cual se utiliza para lograr penetrar o ingresar al sistema, el objetivo del atacante es realizar por medio de conexión SSH la utilización de un diccionario que contiene nombres de usuarios y contraseñas, para este caso se utiliza la funcionalidad Hydra que hace parte de los alcances de Kali Linux que se conjuga con una sentencia que se ilustra la Figura 3-43 y evidencia los usuarios y contraseñas exitosas, con esta información el atacante ya tendría acceso a la maquina víctima de forma remota.

Figura 3-43: Ataque SSH. Fuente propia

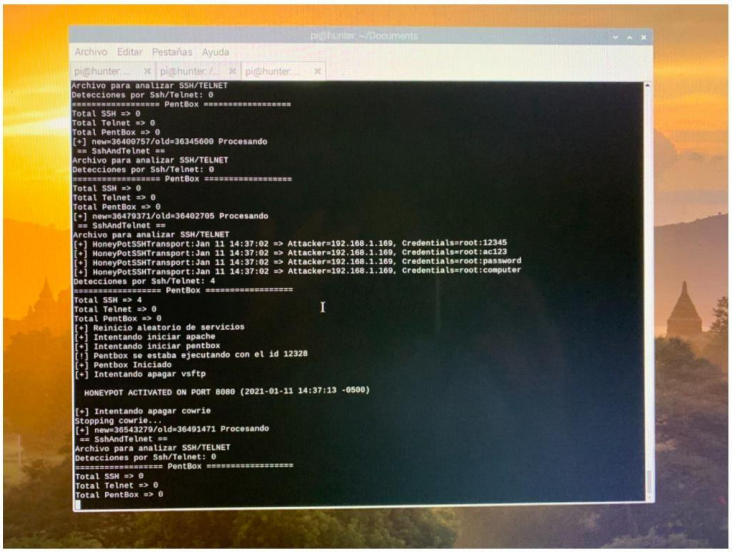


De forma paralela se puede observar en la consola del sistema operativo Raspbian la detección de forma automática de los usuarios y password utilizados en el diccionario del atacante, los cuales son

exitosos a la conexión remota SSH, también en la

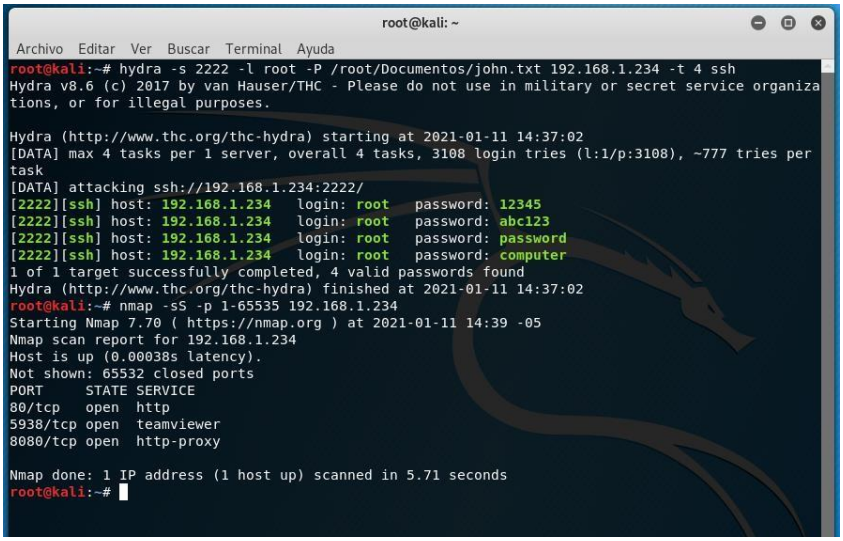
Figura 3-44, se observa el conteo de ataques exitosos y la activación e inactivación de servicios informáticos en el sensor IoT, generando con ello la auto-configuración.

Figura 3-44: Detección de ataque SSH. Fuente propia.



Al utilizar nuevamente el comando Nmap se visualiza en la Figura 3-45, la variación en los servicios informáticos.

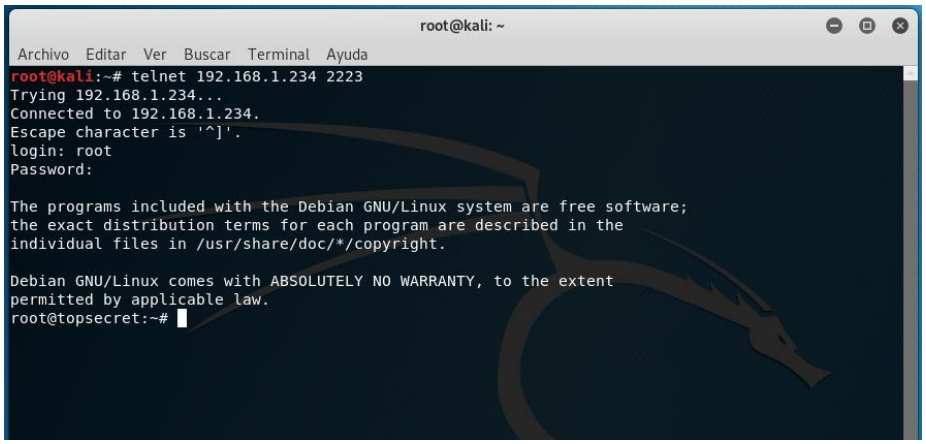
Figura 3-45: Variación de servicios informáticos. fuente propia.



En esta tercera fase el atacante intenta realizar conexión remota con el protocolo telnet, en esta ocasión no utiliza un diccionario sino de forma individual hace la intrusión, utilizando como login:

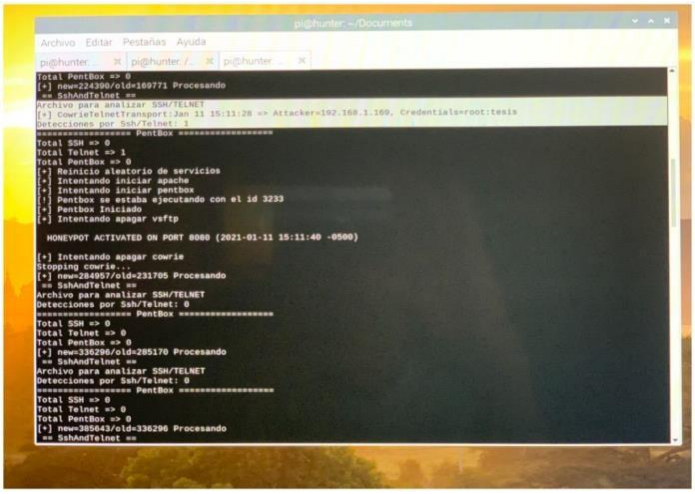
root y como password tesis, el resultado de la conexión fue exitoso como lo indica la *Figura 3-46*.

Figura 3-46: Ataque Telnet. Fuente propia.



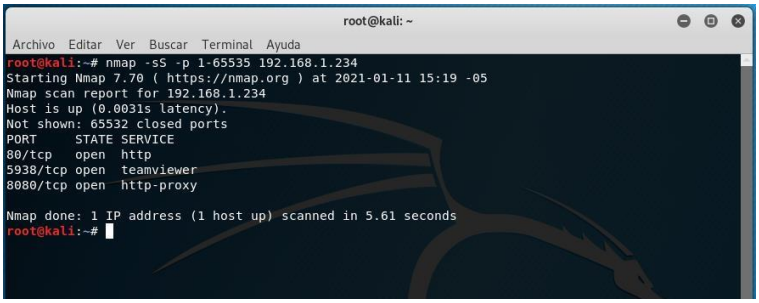
La *Figura 3-47,* indica el resultado de la conexión telnet, el log se observa las credenciales utilizadas por el atacante, el log alerta de 1 ataque tipo telnet y finalmente de activan e inactivan servicios informáticos.

Figura 3-47: Resultados ataque Telnet. Fuente propia.



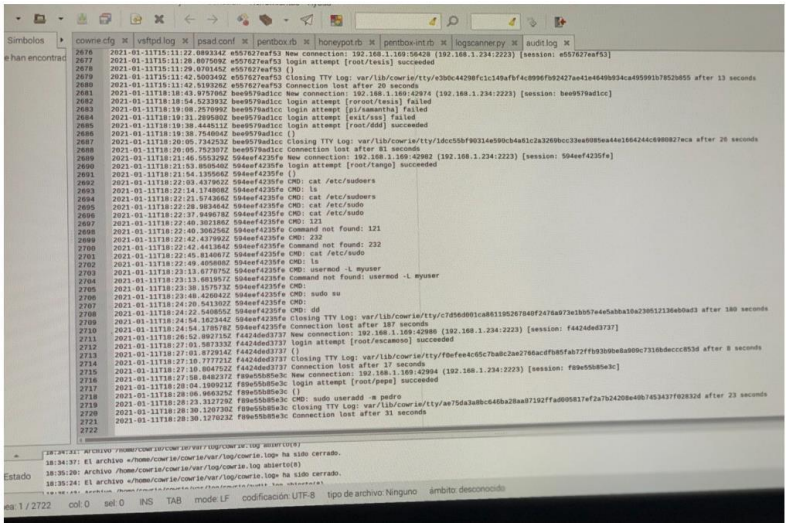
En la Honeynet es claro que siempre que la red sufra un impacto en cuanto a la violación de sus accesos no autorizados, este debe reaccionar posterior a la acción del atacante, ya que es el detonante para que los servicios informáticos se activen o inactive, como lo muestra la *Figura 3-48*.

Figura 3-48: Servicios Informáticos activos. Fuente propia.



Los logs hacen parte vital de este proyecto, dado que reflejan el fundamento de la enseñanza en las redes productivas, en los Honeypot se encuentra las huellas de las estrategias utilizadas por el atacante por ejemplo en la *Figura 3-49* se tiene como evidencia todo el accionar del atacante, es decir no solo se obtienen las credenciales de acceso de manera inusual, sino también la creación de un usuario para la fuga de información y otros comandos que hacen parte de su comportamiento hostil, es claro que el Honeypot evidencia en la línea 2676 New Connection indicando que la ip 192.168.1.169 logró conexión al dispositivo IoT, esta conexión la realizó por el puerto 2223 que es donde estuvo activo el servicio Telnet, esta intrusión se logró utilizando el usuario root con el password tesis, al tener una conexión por 20 segundos, que es tiempo prudente para realizar alguna actividad maliciosa, sin embargo pierde la conexión y se evidencia varios intentos fallidos, hasta encontrar credenciales para restablecer el acceso, posterior al accionar intenta con el comando cat, visualizar la librería de usuarios que tienen el privilegio superusuario, esto con el fin de buscar la oportunidad de escalar privilegios y la ejecución de comandos de alto impacto, posteriormente invoca varios comandos de los cuales con la sintaxis useradd –a crea una cuenta de usuario definido como pedro.

Figura 3-49: log del Honeypot Cowrie. Fuente propia.



En las *Figura 3-50* y *Figura 3-51* se puede visualizar el monitoreo continuo que realiza la herramienta Wazuh, este potente HIDS refleja los resultados de eventos de seguridad causados en la máquina víctima, tales como modificaciones en su sistema operativo, es decir, presenta una alerta

de seguridad en el dashboard de Kibana cada vez que los servicios informáticos nativos del SO se auto-configuran, ya sea que estén activos o inactivos. Autenticaciones exitosas y fallidas, gráficas de performance, esta sincronización entre el agente y el servidor Wazuh permite tener datos reales de eventos de seguridad que les lleva a la correlación de los mismos. Con esto, los administradores de seguridad pueden tener herramientas adicionales para la correlación de eventos de seguridad y análisis posterior a los ataques generados (objetivo que tiene cualquier Honeynet).

Figura 3-50: Resultado de Monitoreo con Wazuh. Fuente propia.

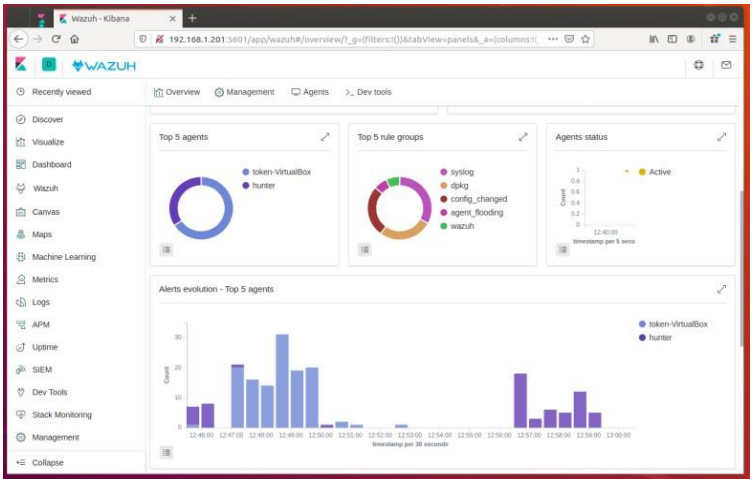
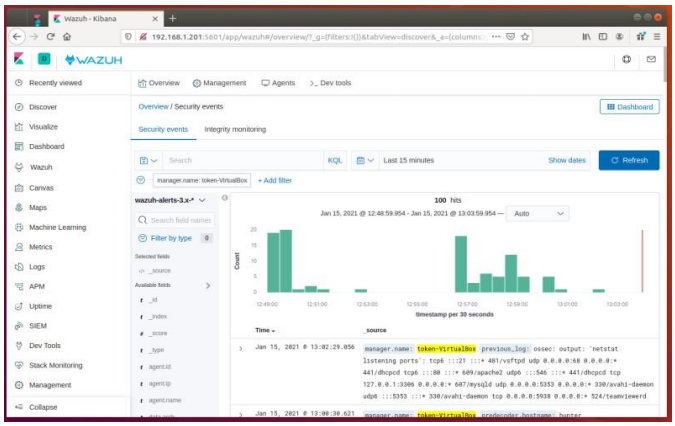


Figura 3-51: Patrón de comportamiento del agente. Fuente propia.



Con las pruebas realizadas, y la visualización tanto en modo consola como gráficamente, los operadores de seguridad y administradores del sistema pueden establecer, de acuerdo a las alertas generadas, cuáles serían las recomendaciones a aplicar en los ambientes productivos. De esta manera, éste sistema de alertas usando una Honeynet, permite a través de la activación y desactivación de servicios (auto-configuración) entregar información diversa a los atacantes y con ello, poder obtener información relevante de ataques de acuerdo a los servicios que se configuren de manera automática.

Ya generada la arquitectura Honeynet con los componentes mencionados en los apartados anteriores, se logra cumplir el objetivo general del presente proyecto, puesto que se tiene un ecosistema IoT que brinda la auto-configuración de servicios informáticos, y funciona como método de alerta de eventos de seguridad basada en una arquitectura Honeynet monitoreada por un sensor IoT que permite identificar diferentes tipos de ataques.

4. Conclusiones y recomendaciones.

4.1 Conclusiones

- Se ha logrado el objetivo general al diseñar un método de alerta de eventos de seguridad basada en una arquitectura Honeynet monitoreada por un sensor IoT auto-configurable que permita identificar ataques ssh, telnet o web tales como fuerza bruta o accesos indebidos al servicio, esto por medio de un script y otros componentes que son el Core del proyecto.

- Se ha logrado diseñar la red Honeynet en IoT funcional en la que se obtenga la captura y almacenamiento de logs o registros de auditoría, para el análisis e identificación de posibles ataques por un dispositivo IoT, logrando activación de servicios informáticos en la medida que se ejecuten ataques.

- Se ha logrado diseñar un método de auto-configuración en un sensor IoT que permita, a través de script o un programa software, activar diferentes servicios de acuerdo con los ataques detectados en sus logs o registros de auditorías, con el fin de recolectar datos de posibles ataques.

- Se ha logrado establecer la capacidad de recolectar la información de ataques en la Honeynet con el sensor IoT auto-configurable, para demostrar la captura del evento de seguridad y generar las recomendaciones de seguridad hacia las redes reales.

- Según las pruebas realizadas en los sensores IoT, los cuales midieron su potencial para conformar un ecosistema IoT con la interoperabilidad de una arquitectura Honeynet se demuestra que en su firmware tiene vulnerabilidades que se pueden explotar en el entorno de internet de las cosas.

- Tener la capacidad de predecir posibles eventos de seguridad es un gran paso de avance para las organizaciones, esto porque permite tener un panorama de lo que puede llegar a suceder y tener un manejo de incidentes de seguridad integrado con los procesos de continuidad del negocio.

- En la arquitectura del mundo de los señuelos se ha validado sus ventajas y desventajas, sus herramientas, su uso y funcionamiento, fue una experiencia enriquecedora ya que es claro que la tecnología no se detiene, esta sigue su marcha y aun así no es posible garantizar la seguridad total, pero si es posible mitigar impactos.

4.2 Recomendaciones, lecciones aprendidas y proyecto futuro.

- Para este proyecto solo fue posible la instalación de 3 Honeypots que administran servicios informáticos, para posibles proyectos se recomienda ampliar el portafolio de Honeypots con el fin de tener mayor alcance en los resultados.

- El hardware y software son importantes en toda arquitectura, se recomienda tener herramientas de mayor capacidad en cuanto a los recursos, esto con el fin de tener mejores resultados.

- Se recomienda para la red que tenga habilitada en los servicios SSH y TELNET, que sus intentos de conexión sean limitadas, esto para mitigar ataques de fuerza bruta o de diccionario.

- La inteligencia artificial es un recurso muy eficiente, como recomendación sería muy oportuno validar este esquema con machine learning, ya que se tendría una autonomía más amplia.

- La tendencia a la mejora continua debe ser una constante en todos los grupos que atienden incidentes de seguridad.

- En proyectos futuros se podría lograr la centralización de los logs de los Honeypots en un dashboard que permita correlación de eventos de forma interactiva.

Bibliografía

[1] NASSAR, Mohamed; STATE, Radu; FESTOR, Olivier. VoIP honeypot architecture. En *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2007. p. 109-118.

[2] SATYA PRAKASH Ghrera, PRADEEP KUMAR Gupta. Image Information Processing, Proceedings of IEEE Second International Conference on . 2013. USA: IEEE Press. [ISBN : 978-1-4673-6099-9] .

[3] RAMIREZ, Jhon; PEDRAZA, César. Performance analysis of communication protocols for internet of things platforms. En *2017 IEEE Colombian Conference on Communications and Computing (COLCOM)*. IEEE, 2017. p. 1-7.

[4] KAPUR, & KHATRI. "On Discrete Software Reliability Growth Modeling with Two types of Imperfect Debugging", OptiMA 2015, National Conference on Mathematical Modeling, Optimization and Their Applications", Apr 28 29, 2015, Delhi.

[5] KUZIN, Mikhail, SHMELEV, Yaroslav, MACKRUSHIN, Denis, KUSCOV Vladimir. Trampas para el Internet de las cosas. Análisis de los datos por Kaspersky Lab en sus trampas para el Internet de las cosas. 2017. Disponible en: https://securelist.lat/honeypots-and-the-internet-of-things/85165/

[6] WANG, Kun, et al. Strategic honeypot game model for distributed denial of service attacks in the smart grid. *IEEE Transactions on Smart Grid*, 2017, vol. 8, no 5, p. 2474-2482. Disponible en: http://ieeexplore.ieee.org.itm.elogim.com/document/7857804/

[7] SPITZNER, Lance. Honeypots: Catching the insider threat. En *19th Annual Computer Security Applications Conference, 2003. Proceedings.* IEEE, 2003. p. 170-179. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.3070&rep=rep1&t ype=pdf

[8] AHMED, Hanaa Mohsin; HASSAN, Nidaa Flaih; FAHAD, Assmaa A. Designing a smartphone honeypot system using performance counters. *Karbala International Journal of Modern Science*, 2017, vol. 3, no 1, p. 46-52. Disponible en: http://www.sciencedirect.com/science/article/pii/S2405609X16304225

[9] RODRÍGUEZ Francisco J. *"Detección, análisis y visualización de ataques mediante Honeypots"*, Incibe-cert. [En línea] Publicado el 31/03/2015. Disponible en: https://www.certsi.es/blog/honeypots

[10] SAADI, Chaimae; CHAOUI, Habiba. Cloud computing security using ids-am-clust, honeyd, honeywall and honeycomb. *Procedia Computer Science*, [Online] 2016, vol. 85, p. 433-442. Available in: http://www.sciencedirect.com.itm.elogim.com/science/article/pii/S1877050916305294

[11] FRAUNHOLZ, Daniel; ZIMMERMANN, Marc; SCHOTTEN, Hans D. An adaptive honeypot configuration, deployment and maintenance strategy. En *2017 19th International Conference on Advanced Communication Technology (ICACT) [Online]*. IEEE, 2017. p. 53-57. Available in: https://ieeexplore.ieee.org/abstract/document/7890056

[12] AHMED, Hanaa Mohsin; HASSAN, Nidaa Flaih; FAHAD, Assmaa A. Designing a smartphone honeypot system using performance counters. *Karbala International Journal of Modern Science*, [Online] 2017, vol. 3, no 1, p. 46-52. Available in: http://www.sciencedirect.com/science/article/pii/S2405609X16304225

[13] GIRALDO, E. M.; ISAZA, G. A. Sistema multi-agente deliberativo para la obtención y análisis de datos en Honeynets. Entre Ciencia e Ingeniería [En línea], 2016, vol. 10, No 20, p. 24-31. Disponible en: http://www.scielo.org.co/pdf/ecei/v10n20/v10n20a04.pdf

[14] KRISHNAPRASAD, P. Capturing attacks on IoT devices with a multi-purpose IoT honeypot. *Indian Institute of Technology Kanpur, [Online] (May2017)*, 2017. Available in: https://security.cse.iitk.ac.in/sites/default/files/15111021.pdf

[15] LUO, Tongbo, et al. Iotcandyjar: Towards an intelligent-interaction honeypot for iot devices. *Black Hat*, 2017, p. 1-11. Available in: https://paper.seebug.org/papers/Security%20Conf/Blackhat/2017_us/us-17-Luo-Iotcandyjar-Towards-An-Intelligent-Interaction-Honeypot-For-IoT-Devices-wp.pdf

[16] WAGENER, Gérard, et al. Adaptive and self-configurable honeypots. En *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*. IEEE, [Online] 2011. p. 345-352. Available in: Http://ieeexplore.ieee.org/abstract/document/5990710/?anchor=relatedarticles

[17] FERNANDEZ, Gerardo; NIETO, Ana. "Configuración de honeypots adaptativos para análisis de malware", III Jornadas Nacionales de Investigación en Ciberseguridad (JNIC 2017), pp. 91-98, 2017. Disponible en: https://www.nics.uma.es/pub/papers/1650.pdf

[18] GIRTZ, Kyle A. "Dynamic Honeypot Configuration for Programmable Logic Controller Emulation". [Master of Science in Cyber Operations] Air Force Institute of Technology, AFIT Scholar. Available in https://scholar.afit.edu/cgi/viewcontent.cgi?article=1301&context=etd

[19] ZHANG, Weizhe, et al. An IoT Honeynet Based on Multiport Honeypots for Capturing IoT Attacks. *IEEE Internet of Things Journal* [Online], 2019, vol. 7, no 5, p. 3991-3999. doi: 10.1109/JIOT.2019.2956173. Available in: https://ieeexplore.ieee.org/abstract/document/8915712/

[20] AKATYEV, Nikolay; JAMES, Joshua I. Evidence identification in IoT networks based on threat

assessment. *Future Generation Computer Systems*, 2019, vol. 93, p. 814-821. Available in: https://www.sciencedirect.com/science/article/abs/pii/S0167739X17300857?via%3Di%20hub

[21] TZU, Sun. El arte de la guerra. 6ta. Edición. Madrid. Editorial Trotta. 2007. ISBN 8481644927.

[22] The Honeynet Project. [Online]. 2002. Disponible en: http://www.honeynet.org.

[23] IBM Internet Securuty System. [online]. 1ro. Octubre de 2007. Disponible en: http://xforce.iss.net/xforce/alerts/id/advise101.

[24] SENSO, Jose A., et al. Evaluación del uso de una herramienta de trabajo colaborativo en la docencia de la Traducción: análisis de ficheros log. En *Conferência IADIS Ibero-Americana WWW/Internet*. 2006. p. 57-66. Available in: https://www.researchgate.net/profile/Jose-Senso/publication/228346648_Evaluacion_del_uso_de_una_herramienta_de_trabajo_colaborativo_en_la_docencia_de_la_Traduccion_analisis_de_ficheros_log/links/0fcfd509a259942893000000/Evaluacion-del-uso-de-una-herramienta-de-trabajo-colaborativo-en-la-docencia-de-la-Traduccion-analisis-de-ficheros-log.pdf

[25] SAATY, Thomas L. Decision making with the analytic hierarchy process. *International journal of services sciences*, 2008, vol. 1, no 1, p. 83-98. Available in: https://www.inderscienceonline.com/doi/abs/10.1504/IJSSci.2008.01759

[26] VELASQUEZ, Mark; HESTER, Patrick T. An analysis of multi-criteria decision making methods. *International journal of operations research*, 2013, vol. 10, no 2, p. 56-66. Available in: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.1308&rep=rep1&type=pdf

[27] Raspberrypi, "Raspberrypi," 2019. [Online]. Available in: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

[28] Arduino, "Arduino," 2019. [Online]. Available: https://www.arduino.cc/

[29] Cubieboard,"Cubieboard,"2019.[Online].Available: http://cubieboard.org/2017/12/27/cubieboard6-is-released-to-the-overseas-users/

[30] Beaglebone Black,"Beaglebone Black,"2019.[Online].Available: https://beagleboard.org/black

[31] Pandaboard ES," Pandaboard ES,"2019.[Online].Available: https://www.digikey.com/en/product-highlight/t/texas-instruments/pandaboard

[32] SOUTH M.. HoneyPi: un Honeypot sencillo para Raspberry Pi [Online]. Agosto 22 de 2017. Available in: https://trustfoundry.net/honeypi-easy-honeypot-raspberry-pi/

[33] YANG T.. PentBox-"Una suite de seguridad y pentesting en ruby" [Online]. Agosto 15 de 2009. Available in: https://www.dragonjar.org/pentbox-una-suite-de-seguridad-y-pentesting-en-ruby.xhtml

[34] OOSTERNOF M. Cowrie [Online]. Agosto 28 de 2019, Available in: https://cowrie.readthedocs.io/en/latest/README.html#documentation

[35] WARZYŃSKI, Arkadiusz; KOŁACZEK, Grzegorz. Intrusion detection systems vulnerability on adversarial examples. En *2018 Innovations in Intelligent Systems and Applications (INISTA)*. IEEE, 2018. p. 1-4. Available in: https://ieeexplore.ieee.org/abstract/document/8466271/

[36] YI, Shanhe; LI, Cheng; LI, Qun. A survey of fog computing: concepts, applications and issues. En *Proceedings of the 2015 workshop on mobile big data*. [Online] 2015. p. 37-42. Available in: https://dl.acm.org/doi/abs/10.1145/2757384.2757397

[37] SENTHILKUMAR, P.; MUTHUKUMAR, M. A Study on Firewall System, Scheduling and Routing using pfsense Scheme. En *2018 International Conference on Intelligent Computing and Communication for Smart World (I2C2SW)*. IEEE, 2018. p. 14-17. doi: 10.1109 / I2C2SW45816.2018.8997167. Available in: https://ieeexplore.ieee.org/abstract/document/8997167

[38] INCIBE. (2018, febrero 08.) "Introducción a los sistemas embebidos". [Online]. https://www.incibe-cert.es/blog/introduccion-los-sistemas-embebidos

[39] Python Software Foundation. "Welcome to Python.org." [Online]. febrero 09 de 2019. Available in: https://www.python.org/

[40] WAZUH. (2018) "la plataforma de seguridad de código abierto". [Online]. Available in: https://wazuh.com/

[41] ARIFIANTO, Ridho Maulana; SUKARNO, Parman; JADIED, Erwid Musthofa. An SSH Honeypot Architecture Using Port Knocking and Intrusion Detection System. En *2018 6th International Conference on Information and Communication Technology (ICoICT)*. IEEE, 2018. p. 409-415. doi: 10.1109 /ICoICT.2018.8528787. Available in: https://ieeexplore.ieee.org/abstract/document/8528787

[42] RAMÍREZ GOMEZ, Julián; LEÓN HENAO, Álvaro. Método para la detección automática de ciberataques con sensores maliciosos en redes inalámbricas ad-hoc de sensores para entornos de hogares inteligentes. M.S. tesis, Facultad de Ingenierías, ITM, Medellin, Colombia, 2018. Disponible en: http://repositorio.itm.edu.co/handle/20.500.12622/124

[43] HUANG, Wen, et al. Short-term load forecasting based on similar day approach and intelligent algorithm using analytic hierarchy process. En *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. IEEE, 2019. p. 2507-2512. doi: 10.1109/ITNEC.2019.8729371. Available in: https://ieeexplore.ieee.org/abstract/document/8729371

[44] KIM, Tae Won. Auto-configurable ROV simulator. En *OCEANS 2014-TAIPEI*. IEEE, 2014. p. 1-6. Available in: https://ieeexplore.ieee.org/abstract/document/6964473

[45] NANTES, Esteban A. “El método analytic hierarchy process para la toma dedecisiones. Repaso de la metodología y aplicaciones”, EDP en investigación operativa, vol. 27, pp. 46, noviembre 2019. Disponible en: https://www.researchgate.net/profile/Esteban-A-Nantes/publication/338840830_EL_METODO_ANALYTIC_HIERARCHY_PROCESS_PARA_LA_TOMA_DE_DECISIONES_REPASO_DE_LA_METODOLOGIA_Y_APLICACIONES/links/5ec51bd892851c11a877b14b/EL-METODO-ANALYTIC-HIERARCHY-PROCESS-PARA-LA-TOMA-DE-DECISIONES-REPASO-DE-LA-METODOLOGIA-Y-APLICACIONES.pdf

[46] ABBASI, Fahim H.; HARRIS, R. J. *"Experiences with a Generation III virtual Honeynet"*, 2009 Australasian Telecommunication Networks and Applications Conference (ATNAC) , Canberra, ACT, 2009, págs. 1-6, doi: 10.1109 possible/ ATNAC. 2009 Available in: https://ieeexplore.ieee.org/abstract/document/5464785

[47] GALLEGO, Eduardo; DE VERGARA, Jorge E. López. *"Honeynets: Aprendiendo del Atacante"*, E.T.S.I. de Telecomunicaciones, Universidad Politécnica de Madrid 2017. Available in: https://web.dit.upm.es/~jlopez/publicaciones/mundointernet04.pdf

[48] MOHAN, Sanjeev. “Databases and Machine Learning Coalesce” Gartner, [Online]. (2020 Sep 14). Available in: https://blogs.gartner.com/sanjeev-mohan/2020/09/14/48/?_ga=2.28154495.2134897898.1605886560-2052682031.1604443213

[49] MCJUNKIN J. (2017 Dic 10) “Controles de Seguridad Críticos de CIS”, [Online]. Available in: https://www.sans.org/reading-room/

[50] LÓPEZ M. Marvin. “Lenguajes de programación más utilizados en 2020”, [Online]. Agosto 09 de 2020. Disponible en: https://openwebinars.net/blog/lenguajes-de-programacion-mas-utilizados-en-2020/

[51] Federación de Enseñanza de CC.OO "HoneyNets, una desconocida en la seguridad informática". Revista digital para profesionales de la enseñanza. No. 5, [en línea] 2009. ISSN 1989-4023, Disponible en: https://www.feandalucia.ccoo.es/docu/p5sd6337.pdf

[52] VARGAS PAREDES, Javier Santiago (2015). "Honeypot como herramienta de prevención y detección de ciberataques en las redes de datos de la facultad de ingeniería en sistemas, electrónica e industrial". Trabajo de grado, [en línea]. Disponible en: https://repositorio.uta.edu.ec/bitstream/123456789/13083/1/Tesis_1047si.pdf".

[53] AZIZOV, Danis "Arquitectura DMZ Perimetral: Una implementación corporativa". Universidad Autónoma de Barcelona, Bellaterra. [en línea] 2020. Disponible en: https://ddd.uab.cat/pub/tfg/2020/tfg_286259/Arquitectura_DMZ_Perimetral_Una_Implementacin_Corporativa.pdf

[54] RASPBIAN Org "Imágenes de sistema operativo". [en línea] 2021. Disponible en: http://www.raspbian.org/

Anexo: Códigos de los desarrollos mencionados.

Anexo 1

A continuación, se describe el script en Python para la auto-configuración de servicios informáticos en el sensor IoT:

 # Importamos las librerías necesarias tanto del sistema, fecha y la generación de números aleatorios, esto para extraer los argumentos de la línea de comandos.

 # Definir la constante que será la ruta donde estará el log que alimenta el script /var/log/messages # Crear archivos temporales que serán los que analicen los eventos por parte del atacante

 # Definir los comandos y servicios que serán los reinicien de forma aleatoria

```
import sys
import os
import os.path
import time
import subprocess
from random import randrange
from datetime import datetime

#Definicion de constantes
# Constate para el log que alimenta la herramienta
LOG = '/var/log/messages'
# carpeta de configuracion de la herramienta
CONFIGURATION_FOLDER = 'lgscan/'
# Archivos temporales para el procesamiento
CONFIGURATION_PREVIOUS_LOG = CONFIGURATION_FOLDER + 'previous.log'
CONFIGURATION_PREVIOUS_DIF = CONFIGURATION_FOLDER+ '.logscannerdiff'
CONFIGURATION_REFRESH_TIME = 20
# Constate de configuracion de comandos
CONFIGURATION_SERVICE_START_COMMANDS = {
'vsftp':{
'check': "sudo service vsftpd status | grep 'active (running)' | awk 'END {print NR}'",
'start': 'sudo service vsftpd start',
'stop' : 'sudo service vsftpd stop',
},
'apache':{
'check': "sudo service apache2 status | grep 'active (running)' | awk 'END {print NR}'",
'start': 'sudo service apache2 start',
'stop' : 'sudo service apache2 stop',
},
'cowrie':{
'check': '',
'start': '/home/cowrie/cowrie/bin/cowrie start',
'stop' : '/home/cowrie/cowrie/bin/cowrie stop',
}
}
```

 # Establecer el nombre del script y reflejarlos como banner # Construir funcion que tengra presente fecha y hora

 # Construir funcion que tenga presente un backup del messages # Construir funcion que active el Honeypot pentbox

```
36
37  def banner():
38      """Esta funcion muestra el nombre de la herramienta en asciiart"""
39      print("""
40
41       _
42      | |
43      | |  ___   __ _   ___  ___ __ _ _ __   ___  _ __
44      | | / _ \ / _` | / __|/ __/ _` | '_ \ / _ \ '__|
45      | ||  __/| (_| | \__ \ (_| (_| | | | |  __/ |
46      |_| \___| \__, | |___/\___\__,_|_| |_|\___|_|
47                   | |
48                   |_|
49      """)
50
51      banner()
52
53  def get_today():
54      """Esta funcion simplemente retorna el dia actual en un formato especifico"""
55      return datetime.today().strftime('%Y_%m_%d_%H_%M_%S')
56
57  def clear_environment():
58      """Esta funcion hace un backup del var log messages y lo analiza"""
59      os.system("rm -f {}* &".format(CONFIGURATION_FOLDER))
60      os.system("cp {} /tmp/{}{}.log".format(LOG, 'messages', get_today()))
61      os.system("echo '' > {}".format(LOG))
62      print('OK')
63  def pentbox(cmd):
64      """Esta funcion inicializa el script modificado de pentbox"""
65      #extraemos el path actual
66      pwd = subprocess.check_output(['pwd'])
67      #definimos el nombrel archivo a ejecutar
68      service = pwd.strip()+"/pentbox-int.rb"
69      #verificamos si ruby esta ejecutando a pentbox previamente
70      os.system('ps -ef | pgrep ruby > pid.txt')
71      pid = ""
72      #abrimos el archivo que contiene el id de proceso de pentbox
73      with open('pid.txt') as f:
74      pid = f.readline().strip()
75      if "[1]+  Terminado" in pid:
76      pid = ""
77      if pid == "":
78      print("[+] Pentbox no se estaba ejecutado")
79      else:
        print("[!] Pentbox se estaba ejecutando con el id "+str(pid))
```

 # Iniciar y detener el Honeypot pentbox

 # Función que calcula la diferencia de líneas entre el log y la ejecución previa

 # Diccionario de detección para Honeypot pentbox

```
#iniciamos pentbox
if cmd == 'start':
if pid != "":
os.system('kill '+pid)
os.system('ruby '+service+ "&")
print("[+] Pentbox Iniciado")
#terminamos pentbox
elif cmd == 'stop':
if pid != "":
os.system('kill '+pid)
print("[+] Pentbox Terminado")
#comando por defecto
else:
print('comando no soportado')


def readfile(myFile):
"""Esta funcion lee un archivo linea por linea y remueve los saltos de linea"""
lines = []
# Try-Except para excepciones en la lectura del archivo
try:
# Leer archivo
with open(myFile, 'r') as f:
for l in f.readlines():
line = l.strip() # quitar espacio y salto de linea al final '\n'
lines.append(line)
except:
sys.exit('[!] Error leyendo el log')
return lines

def readlog():
"""Esta funcion calcula la diferencia de lineas entre el log de insumo y
la ejecucion previa
"""
#verificar si el archivo existe en el sistema
if not os.path.isfile(CONFIGURATION_PREVIOUS_LOG):
return readfile(LOG)
else:
#calcular la diferencia entre los archivos y retornarla
os.system('diff '+LOG+' '+CONFIGURATION_PREVIOUS_LOG+' > '+ CONFIGURATION_PREVIOUS_DIF)
return readfile(CONFIGURATION_PREVIOUS_DIF)

def rule_PentBox(line):
"""Crea el diccionario deteccion para pentbox"""
```

 # función que permite diferenciar entre un

ataque ssh o telnet # Función de detección

y extracción para ssh y telnet

 # Función que permite realizar un escaneo de vulnerabilidades y aplica las reglas ssh y telnet

```
124   pieces = line.split('INTRUSION ATTEMPT DETECTED! from ')
125   attacker = pieces[1].split(":")[0]
126   pieces = line.split('(')[1].split(")")[0].split(" ")
127   date = pieces[0]+" "+pieces[1]
128   info = {
129   'service': 'PentBox',
130   'date': date,
131   'attacker': attacker,
132   'victim': '127.0.0.1'
133   }
134   return info
135
136 def rule_SshTelnet(detection):
137   """Diferencia una deteccion para ssh o para telnet"""
138   service, attacker, creds, date = None, None, None, None
139   pieces = detection.split(' ')
140   date = " ".join(pieces[1:4])
141   #deteccion y extraccion para ssh
142   if "HoneyPotSSHTransport" in detection:
143   attacker = pieces[9].replace(']','').split(",")
144   attacker = attacker[len(attacker)-1]
145   creds = pieces[12]
146   creds = creds.replace("[","")
147   creds = creds.replace("]","")
148   creds = creds.replace("b", "")
149   creds = creds.replace("/",":")
150   creds = creds.replace("'","")
151   return "HoneyPotSSHTransport", attacker, creds, date
152   else:
153   #deteccion y extraccion para telnet
154   if "CowrieTelnetTransport":
155   attacker = pieces[6].replace(']','').replace("[", "").split(",")
156   attacker = attacker[len(attacker)-1]
157   creds = pieces[9]
158   creds = creds.replace("[","")
159   creds = creds.replace("]","")
160   creds = creds.replace("/",":")
161   return "CowrieTelnetTransport", attacker, creds, date
162
163
164 def SshAndTelnet():
165   """Hace una busqueda de vulnerabilidades en el log y aplica las reglas de ssh y telnet"""
166   print(" == SshAndTelnet == ")
167   print("Archivo para analizar SSH/TELNET ")
```

 # Definir las líneas y objetos de detección para su posterior impresión

 # Función que permite un escaneo de vulnerabilidades y aplica las reglas del Honeypot pentbox

```
lines = readlog()
registers = 0
detections = []
detectionsBySshTel = []
# 3. Definir las lineas con deteccion
for line in lines:
if "HoneyPotSSHTransport" in line:
if "login attempt" in line and "succeeded" in line:
detections.append(line)
if "CowrieTelnetTransport" in line:
if "login attempt" in line and "succeeded" in line:
detections.append(line)
#forma los objeetos de deteccion
for detection in detections:
service, attacker, data, date = rule_SshTelnet(detection)
result = {}
result['service'] = service
result['attacker'] = attacker
result['victim'] = data
result['date'] = date
detectionsBySshTel.append(result)
#imprime los objetos encontrados
for detection in detectionsBySshTel:
print("[+] {}:{} => Attacker={}, Credentials={}".format(
detection['service'],
detection['date'],
detection['attacker'],
detection['victim'],
))
registers = len(detectionsBySshTel)
print("Detecciones por Ssh/Telnet: {}".format(registers))
return detectionsBySshTel

def PentBox():
"""Hace una busqueda de vulnerabilidades en el log y aplica las reglas de pentbox"""
print("================== PentBox ================== ")
# 2. Leer el archivo siempre y cuando sea posible
lines = readlog()
registers = 0
detections = []
results = []
# 3. Definir las lineas con deteccion de pentbox
for line in lines:
if "INTRUSION ATTEMPT DETECTED! from " in line:
```

 # función que inicia o reinicia los servicios informáticos de forma aleatoria

```
212  detections.append(line)
213  for detection in detections:
214  res = rule_PentBox(detection)
215  results.append(res)
216  #imprime los objetos de deteccion encontrados
217  for res in results:
218  print("[+] {}:\n{} => Attacker={}".format(
219  res['service'],
220  res['date'],
221  res['attacker'],
222  ))
223  return results
224
225
226  def configure():
227  """Configuracion de la carpeta de lgscan"""
228  config_folder = CONFIGURATION_FOLDER
229  if not (os.path.isdir(config_folder)):
230  os.system("mkdir "+config_folder)
231  if not os.path.exists(CONFIGURATION_PREVIOUS_LOG):
232  os.system("touch "+CONFIGURATION_PREVIOUS_LOG)
233
234  def start_services():
235  """Esta funcion inicia los servicios"""
236  print("[+] Intentando iniciar apache")
237  apache = CONFIGURATION_SERVICE_START_COMMANDS['apache']
238  os.system(apache['start'])
239  print("[+] Intentando iniciar pentbox")
240  pentbox('start')
241  print("[+] Intentando iniciar vsftp")
242  vsftp = CONFIGURATION_SERVICE_START_COMMANDS['vsftp']
243  os.system(vsftp['start'])
244  print("[+] Intentando iniciar cowrie")
245  cowrie = CONFIGURATION_SERVICE_START_COMMANDS['cowrie']
246  os.system(cowrie['start'])
247
248  def restart_random(r):
249  """Esta funcion reinicia los servicios aleatoriamente"""
250  if (r % 2 == 0):
251  # apago apache
252  print("[+] Intentando apagar apache")
253  apache = CONFIGURATION_SERVICE_START_COMMANDS['apache']
254  os.system(apache['stop'])
255  # apago pentbox
```

 # función que inicia o reinicia los servicios informáticos de forma aleatoria

```
256  print("[+] Intentando apagar pentbox")
257  pentbox('stop')
258
259  # enciendo cowrie
260  print("[+] Intentando iniciar cowrie")
261  cowrie = CONFIGURATION_SERVICE_START_COMMANDS['cowrie']
262  os.system(cowrie['start'])
263  # enciendo vsftp
264  print("[+] Intentando iniciar vsftp")
265  vsftp = CONFIGURATION_SERVICE_START_COMMANDS['vsftp']
266  os.system(vsftp['start'])
267  else:
268  # enciendo apache
269  print("[+] Intentando iniciar apache")
270  apache = CONFIGURATION_SERVICE_START_COMMANDS['apache']
271  os.system(apache['start'])
272  # enciendo pentbox
273  print("[+] Intentando iniciar pentbox")
274  pentbox('start')
275  # apago vsftp
276  print("[+] Intentando apagar vsftp")
277  vsftp = CONFIGURATION_SERVICE_START_COMMANDS['vsftp']
278  os.system(vsftp['stop'])
279  # apagando cowrie
280  print("[+] Intentando apagar cowrie")
281  cowrie = CONFIGURATION_SERVICE_START_COMMANDS['cowrie']
282  os.system(cowrie['stop'])
283
284  def copylog():
285  """Esta funcion versiona el /var/log/messages como previo"""
286  os.system('cp '+LOG+' '+CONFIGURATION_PREVIOUS_LOG)
287
288  def main():
289  """Esta es la funcion principal del script"""
290  # Iniciando servicios
291  start_services()
292  #Configuracion de carpeta
293  configure()
294  # Ciclo principal
295  execution = True
296  # Constante de aleatoriedad
297  n = 1
298  # Ejecucion infinita
299  while execution:
```

 # función que realiza la comparación entre Logs para generar el diagnostico # Función que permite aplicar reglas de detección

 # función que permite validar el ciclo y evitar saturación en los logs

```
new = os.path.getsize(LOG)
old = os.path.getsize(CONFIGURATION_PREVIOUS_LOG)
#identificar posible operacion
if new > old:
print("[+] new={}/old={} Procesando".format(str(new), str(old)))
#aplicar reglas
detections2 = SshAndTelnet()
detections3 = PentBox()
qdetection3 = len(detections3)
ssh = 0
telnet = 0
for i in detections2:
if i['service'] == 'HoneyPotSSHTransport':
ssh += 1
elif i['service'] == 'CowrieTelnetTransport':
telnet += 1
print("Total SSH => {}".format(ssh))
print("Total Telnet => {}".format(telnet))
print("Total PentBox => {}".format(qdetection3))
#concluir y reiniciar servicios
detect = ssh+telnet+qdetection3
if detect >= 1:
print("[+] Reinicio aleatorio de servicios")
ran = randrange(15)
restart_random(n)
if n == 1:
n = 2
else:
n = 1
#versionar el log despues de la operacion
copylog()
else:
print("[*] new={}/old={} Esperando cambios".format(str(new), str(old)))
#esperar para evitar saturar el procesamiento
time.sleep(CONFIGURATION_REFRESH_TIME)

#Linea de comandos
if len(sys.argv) == 2:
cmdline_parameter = sys.argv[1]
if cmdline_parameter == "start":
main()
elif cmdline_parameter == "clear":
clear_environment()
else:
 print("Usage: logscanner.py start|clear")
 else:
 print("Usage: logscanner.py start|clear")
```