ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

AUTOMATIZACIÓN DE REDES UTILIZADAS PARA EOT

AUTOMATIZACIÓN DE SEGURIDADES PARA REDES EOT

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

WASHINGTON ZAHID COPARA SUÁREZ

washington.copara@epn.edu.ec

DIRECTOR: CARLOS ROBERTO EGAS ACOSTA

carlos.egas@epn.edu.ec

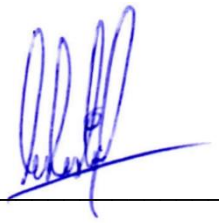
DMQ, septiembre del 2022

CERTIFICACIONES

Yo, Washington Zahid Copara Suárez declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

WASHINGTON ZAHID COPARA SUÁREZ

Certifico que el presente trabajo de integración curricular fue desarrollado por Washington Zahid Copara Suárez, bajo mi supervisión.



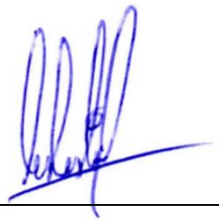
CARLOS ROBERTO EGAS ACOSTA

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

WASHINGTON ZAHID COPARA SUÁREZ



CARLOS ROBERTO EGAS ACOSTA

DEDICATORIA

A mis padres y hermano, por haber sido un pilar importante en mi vida.

AGRADECIMIENTO

Agradezco a mis padres, Washington y Mónica, por haber sido el mayor y más grande apoyo que he tenido en mi vida, gracias a ellos he podido cumplir con mis objetivos y me permito soñar en grande para poder seguir cosechando éxitos en la vida.

A mi hermano Alexis, que gracias a su compañía y alegría me ha permitido seguir adelante con las mismas energías positivas a lo largo de todo este camino.

A mis amigos y amigas, que siempre estuvieron ahí como apoyo, ya sea en lo académico o personal.

A todos mis profesores que a lo largo de la carrera fueron importantes en mi formación académica.

ÍNDICE DE CONTENIDO

CERTIFICACIONES I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
RESUMEN VII
ABSTRACT VIII
1 INTRODUCCIÓN 1
1.1. Objetivo General 2
1.2. Objetivos Específicos 2
1.3. Alcance 2
1.4. Estado del Arte 3
1.5. Marco Teórico 4
1.5.1 Internet de las Cosas (IoT) 4
1.5.2 Internet de Todo (IoE) 4
1.5.3 Seguridad en Redes 5
1.5.3.1 Vulnerabilidad de Redes 6
1.5.3.2 Amenazas presentes de las Redes 7
1.5.3.3 Monitorización de Seguridad de Redes 8
1.5.4 Automatización de Redes 9
1.5.4.1 Inteligencia Artificial 11
1.5.5 Herramientas para la automatización de seguridad en redes 12
1.5.5.1 Monitorización de Seguridad de Redes. 13
1.5.5.2 Herramientas de cifrado. 15
1.5.5.3 Herramientas de análisis de vulnerabilidades. 16
1.6. Trabajos Relacionados 19
2 METODOLOGÍA 20
2.1 Principios para la automatización de seguridad en redes IoE 21
2.1.1 Vulnerabilidades más comunes que afectan a las redes IoE 21
2.1.2 Herramientas para la automatización de la seguridad en redes IoE 22
2.2 Programación en la automatización de seguridad en redes IoE 23
2.2.1 Python en la automatización de seguridad en redes IoE 23
2.2.2 Inteligencia Artificial en conjunto con Python para la automatización de seguridad de redes IoE 25

2.2.3 Cisco DevNet Express Security..... 26
2.2.3.1 Cazador de Amenazas..... 27
2.2.3.2 Respuesta a amenazas mediante SecureX..... 34
2.2.3.3 Seguridad de aplicaciones Cisco..... 36
2.3 Proceso para la automatización de seguridad en redes IoE..... 38
2.3.1 Prácticas para la automatización de redes..... 38
2.3.2 Prácticas para automatizar la seguridad en redes IoE..... 39
3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES..... 44
3.1 Resultados..... 44
3.2 Conclusiones..... 45
3.3 Recomendaciones..... 47
4 REFERENCIAS BIBLIOGRÁFICAS..... 48
5 ANEXOS..... 54

RESUMEN

El presente trabajo tiene como objetivo el poder mostrar las distintas soluciones que se tienen para poder implementar la automatización de seguridad en redes IoE (Internet de Todo) o Internet de Todas la Cosas (EoT), partiendo desde el poder entender que son las redes IoE, sus características y objetivos, así como, de poder discriminar sus principales diferencias con respecto a las redes IoT (Internet de las Cosas). Una vez comprendido el concepto de redes IoE, se busca entender los principios de las seguridades de redes, tales como, su concepto y objetivos, su importancia y porque es imprescindible su aplicación en las redes de comunicaciones, además, de poder conocer cuáles son aquellas vulnerabilidades, amenazas y ataques más comunes, para que en caso de algún ataque sean las primeras en ser revisadas para poder ser mitigadas de la manera más rápida posible, esto utilizando la ayuda de software que permite la automatización en las seguridades de seguridades, donde es necesario conocer específicamente que tipo de herramientas se pueden utilizar, inicialmente las de monitorización de redes, con el fin de poder entender cómo funcionan y sus principales características, para luego utilizar herramientas más específicas de monitorización, tales como las de monitorización de seguridad de redes, que comprende el poder analizar el tráfico de la red garantizando la disponibilidad de la misma, así como el poder analizar el tráfico y evitar algún patrón sospechoso dentro de la red, todo ello apoyado con algún lenguaje de programación como Python, para poder ser utilizado como el mitigador de dichas amenazas que podrían existir.

PALABRAS CLAVE: IoE, IoT, automatización, seguridades, monitorización, Python, amenazas, vulnerabilidades, ataques.

ABSTRACT

The objective of this work is to show the different solutions that are available to implement in security automation in IoE (Internet of Everything) networks. Furthermore, it's necessary to know what IoE networks are, as well as know their features and objectives. Additionally, it is significant to know the differences between IoE and IoT networks. Then, the focus falls on network security, likewise, its concept, objectives, and importance. In addition, it is essential to know the most common vulnerabilities, threats, and attacks that occur on a network, to mitigate as soon as possible an attack. After knowing those concepts, it is necessary to know the tools that can be used in network monitoring, know their main features and understand how they work. Then, to use a specific software of network security monitoring, which includes being able to analyze network traffic, guaranteeing its availability, as well as to be able to analyze the traffic and avoid any suspicious patterns within the network, all supported by a programming language such as Python, to be used as a mitigator of said threats that could exist.

KEYWORDS: IoE, IoT, automation, securities, monitoring, Python, threats , vulnerabilities, attacks.

1 INTRODUCCIÓN

Las redes de comunicaciones y su masificación gracias a nuevos conceptos como el de las redes IoE (Internet de Todo) o Internet de Todas la Cosas (EoT), presentan nuevos retos que deben ser tratados y estudiados, esto principalmente con la expansión que se está teniendo en la actualidad, donde cada vez más dispositivos se empiezan a conectar a la gran red de redes denominada Internet. El término Internet de Todo, es relativamente nuevo, pero sus aplicaciones van incluso más allá del denominado Internet de las Cosas, esto principalmente porque no se desea simplemente conectar dispositivos físicos, si no también emplear otros puntos clave como el de las personas, procesos, datos y cosas.

El querer tener esa masificación de conexiones representa un reto dentro del ámbito de la seguridad de las redes, esto principalmente porque siempre existirán amenazas que pongan en riesgo a la integridad de las redes, y en un ambiente donde se pretende que incluso las personas estén integradas a esta red lo hace incluso más necesario que en otros tipos de redes. Por lo tanto, es necesario conocer acerca de las distintas soluciones que se le puedan dar a las vulnerabilidades y amenazas que se puedan presentar dentro de esta red.

Actualmente, se puede recopilar una gran cantidad de información acerca de cómo mantener seguras las redes tradicionales, y estas mismas pueden ser extrapoladas al ámbito de las redes IoE, cosas simples como políticas de seguridad para mantener una serie de reglas y recomendaciones al querer conectarse a estas redes, además de, conocer las vulnerabilidades más comunes que sirvan de base para mantener la seguridad inicial en las conexiones y las amenazas que pueden afectar a las mismas. Además, existe una gran cantidad de soluciones de software para poder monitorizar la red y las seguridades de estas, las cuales permitirán observar cualquier tipo de eventualidades que pueden estar ocurriendo, esto de manera automática, que en algunos casos si se lo desea pueden ejecutar scripts automáticamente para poder solventar los problemas de seguridad más comunes, logrando así una respuesta más rápida que si no se utilizaran dicho software, así mismo, existen lenguajes de programación como Python, que son muy útiles al momento de querer realizar pruebas de seguridad dentro del ambiente de red.

Todas estas herramientas y soluciones lograrán mantener una red segura y fuerte ante distintas eventualidades que se puedan presentar, logrando así una respuesta rápida y automática, tomando en cuenta que hablamos de una red que maneja datos muy sensibles y que su seguridad termina siendo la prioridad principal para que pueda ser utilizada plenamente sin ningún tipo de inconveniente.

1.1. Objetivo General

Automatizar las seguridades para redes IoE o EoT

1.2. Objetivos Específicos

- Adquirir los conocimientos de automatización de seguridades en redes IoE.
- Conocer acerca del estado actual de la automatización de las seguridades de redes.
- Conocer sobre las vulnerabilidades más comunes en las redes.
- Estudiar el uso de herramientas de programación como Python para las seguridades de redes.
- Enumerar distintas herramientas que se pueden utilizar para poder implementar la automatización de seguridades en las redes IoE.
- Observar distintas aplicaciones similares que se pueden encontrar en la Internet acerca de las seguridades en redes tradicionales y aplicarlos en el campo de las redes IoE.
- Analizar las herramientas de seguridades de redes IoE que se pueden usar para entornos de automatización.

1.3. Alcance

El alcance del proyecto está basado en fases, las cuales son: fase de planeamiento e investigación, fase de implementación y fase de evaluación.

La fase de investigación sirve para poder adquirir distintos conocimientos como, la automatización en las seguridades de redes IoE, el cual consiste en utilizar las tecnologías disponibles para ejecutar tareas relacionadas a las seguridades en redes para reducir considerablemente la manipulación humana, integrando los procesos, las aplicaciones e infraestructura de la seguridad, además, de entender la necesidad de la automatización de los procesos de seguridad y también conocer cuales procesos se pueden automatizar.

Adicionalmente, tener el conocimiento de las tendencias actuales en el ámbito de la automatización de las seguridades de redes, buscando en distintos medios los avances que se han tenido con respecto a la problemática, para después considerar cuales son aquellas vulnerabilidades más comunes que se tienen en las redes IoE con un enfoque más específico. Además, es necesario observar cómo utilizar el lenguaje de programación de Python para poder apoyar las necesidades de herramientas de automatización de seguridades de redes IoE.

Para desarrollar esta fase se necesitan los recursos que se obtengan mediante el internet donde se encuentran distintas bibliografías que permiten obtener la información necesaria para poder realizar el trabajo.

La fase de implementación recurre a las distintas herramientas que se encontraron en la fase anterior, las cuales puedan ser aplicadas, esperando los resultados satisfactorios que aquellas herramientas presenten, además, utilizar aquellas vulnerabilidades comunes encontradas para, mediante las herramientas investigadas poder solventar la mayoría de estas. Adicionalmente, utilizan los lenguajes de programación como Python, que pueda servir en una eventualidad de amenazas de seguridad que aparecen en las redes. Los resultados esperados, es poder verificar cuales son las mejores soluciones que existan actualmente para la problemática de seguridad de redes, entre estas considerar herramientas de monitorización de seguridad de redes, así como soluciones de programación para que poder solventar las amenazas de seguridad lo más pronto posible.

En la fase final de evaluación, primero se discriminan las distintas herramientas que se encontraron en la fase preliminar, con el fin de poder encontrar la mejor herramienta o herramientas que mejores características presentan, además, determinar si Python realmente funciona como una ayuda real para poder solventar los problemas de seguridad que se pueden encontrar en las redes IoE. Finalmente, se espera obtener una gran cantidad de información sobre la automatización de seguridades de redes que puedan ser aplicadas a las redes del Internet de Todo (IoE).

1.4. Estado del Arte

Haciendo uso del artículo “A review on authentication and security procedures for IoE Systems”, es posible explorar cada una de las perspectivas que se tienen en las redes del Internet de Todo, así como los riesgos que implica la interconexión de todo el mundo, partiendo de la base de las redes IoE como la comunicación de todo, y los problemas que puedan existir en los datos generados para cada diferente aplicación que se puedan extraer, por lo que es necesario cubrir estos datos, además al ser un tema muy nuevo sigue quedando al pendiente las mejores alternativas para mitigar cualquier tipo de ataque. [1]

En “Security Automation in Information Technology”, las perspectivas cambian hacia un ámbito mucho más empresarial, que vendría a ser una de las aplicaciones más importantes a seguirse, esto se debe a algo que principalmente ocurre en las empresas y es que la mayoría de problemas de seguridad que existen en las empresas son repetitivos, es decir, que se pueden automatizar los procesos para que estas brechas de seguridad puedan ser

soportadas de manera rápida y eficiente, sin tener que recurrir a analistas de seguridad cuando solo se debe seguir un procedimiento como tal, evitando así costos sin sentido por problemas que se pueden solventar de manera automática. [2]

En el artículo “Security trends in Internet of Things a survey”, se usa el enfoque del internet de las cosas para poder conocer lo importante que son las tendencias actuales de la seguridad para estas aplicaciones, adicionalmente, la búsqueda y la necesidad que comprende el tener protocolos que sean competentes para poder mitigar distintos tipos de amenazas que se podrían dar. [3]

1.5. Marco Teórico

1.5.1 Internet de las Cosas (IoT)

Debido a que el Internet de las cosas y el Internet de todo comparten similitudes, es necesario entender el termino de IoT para poder comprender correctamente las diferencias que estos dos términos tienen. El Internet de las cosas (IoT), es la interconexión de todos los dispositivos físicos que se pueden conectar a una red, aquellos dispositivos se incluyen sensores, software y otras tecnologías cuya finalidad es la de poder compartir e intercambiar cualquier tipo de información o datos ente los dispositivos que comparten la red. Su popularidad se ha disparado con el pasar de los años, debido a que gracias a los avances tecnológicos se han podido implementar redes con cualquier tipo de dispositivos incluso los cotidianos, como electrodomésticos, automóviles, termómetros, monitores de bebe, etc. [4]

El Internet de las cosas, aprovecha las nuevas tecnologías presentes para poder ser fáciles de implementar y accesibles hacia las personas, su funcionamiento es primordial, para que la automatización de algunas tareas sean las más eficientes posibles, debido a que utiliza los principios de la inteligencia artificial en conjunto con el aprendizaje automático para poder realizar este tipo de automatización, esto gracias al envío, recepción y tratamiento que se dan a los datos que están operando dentro de la red de la IoT, pudiendo así facilitar la vida de las personas que utilizan estos dispositivos, que gracias también al desarrollo tecnológico y la baja de precios de estos dispositivos son de gran atracción para el consumo, no solo domestico si no también en el ámbito empresarial. [5]

1.5.2 Internet de Todo (IoE)

El Internet de Todo inicialmente es definido por Cisco, como la interconexión en red de personas, procesos, datos y cosas. Además, se centra en la gran cantidad de posibilidades

y beneficios que se pueden obtener al tener “todo” conectado en línea. En Cisco, se menciona que aproximadamente el 99.4% de los objetos físicos que pueden formar parte de la red del Internet de Todo están desconectados, por lo tanto, el principal objetivo de esta red es el de poder tener conexiones como las de, persona a persona (P2P), máquina a persona (M2P) y de máquina a máquina (M2M), por lo tanto, se busca una masificación en las conexiones de todo lo que se pueda conectar. [6]

A diferencia del denominado Internet de las cosas (IoT), se cubre un concepto más amplio al momento de querer hablar de la interconectividad, por eso es importante el hecho de que en el Internet de todo se quiere asegurar el poder masificar las conexiones, no solamente mediante el uso de dispositivos físicos, si no agregando tres elementos clave más, los que previamente se denominaron personas, procesos y datos. Finalmente, la diferencia principal son sus objetivos, mientras que, en el Internet de las cosas, se quiere crear un ecosistema entre dispositivos físicos para tareas específicas, en el Internet de Todo, se aspira a tener posibilidades casi ilimitadas en el tipo de aplicaciones y tareas que se pueden ejecutar. [7]

Sin embargo, es necesario conocer las limitaciones que se tendrán al momento de querer implementar las redes del Internet de Todo, debido a que estas limitaciones provienen desde las redes del Internet de las Cosas y se tratan de los dispositivos que pueden formar parte de estas redes, debido a que existen algunos nodos que pueden funcionar con altos niveles de procesamiento y consumo de energía, y otros que son limitados en sus capacidades de procesamiento y consumo de energía, esto afecta principalmente a la seguridad de los mismos, debido a que los equipos que no pueden manejar altas capacidades de procesamiento no pueden gestionar los patrones de seguridad que se han establecido en la red, pudiendo ser los equipos más vulnerables. [46]

1.5.3 Seguridad en Redes

La Seguridad en Redes se define como los mecanismos que se disponen para poder identificar amenazas y controlar consecuentes ataques que se puedan generar en una red, además, el poder aplicar las principales técnicas que se disponen para poder garantizar los distintos servicios de seguridad tales como, autentificación, confidencialidad e integridad de la información que forma parte de la red. Además, se abarcan distintos tipos de destrezas como el de desarrollo e implementación de estrategias para poder proteger la información de la red frente a los ataques más comunes. [8]

1.5.3.1 Vulnerabilidad de Redes

Se define a la vulnerabilidad, como el punto débil que se puede generar dentro de un sistema, que puede comprometer y poner en peligro a toda la red. [9]

A continuación, se presentan los tipos de vulnerabilidades que se encuentran en las redes:

Diseño:

- Se presentan políticas de seguridad ineficientes o en muchos casos inexistentes.
- Existe debilidad al momento del diseño e implementación de protocolos que son utilizados en la red.

Implementación:

- Errores de programación.
- Existencia de "puertas traseras" en la red.
- Descuido en las casas fabricantes.

Uso:

- Pobre configuración de los sistemas informáticos.
- Los usuarios finales de una red generan una cantidad de problemas de seguridad debido al mal uso de esta, esto debido a desconocimiento o una falta de seguimiento de los manuales proporcionados por los dueños de la red.
- Existencia de software que facilitan los ataques cibernéticos.

Vulnerabilidades del día cero:

- Vulnerabilidades que no tienen una solución conocida, pero sin embargo si se sabe cómo explotarlas.

Adicionalmente, es necesario presentar cuales son las vulnerabilidades más destacadas en una red: [10]

- Errores en la gestión de recursos.
- Error de configuración.
- Factor humano.
- Validación de entrada.
- Salto de directorio.
- Permisos, privilegios y/o control de acceso.

1.5.3.2 Amenazas presentes de las Redes

Las amenazas de seguridad se definen como el aprovechar una vulnerabilidad o vulnerabilidades que están presentes en la red, estas pueden ocurrir en cualquier momento, ya que cualquier sistema de red puede presentar una probabilidad de recibir ataques por parte de personas mal intencionadas.

Estas amenazas se dividen en tres grandes grupos presentados a continuación: [9]

1. Intencionales: Son aquellas amenazas que son provocadas con intenciones maliciosas, usualmente con el propósito de concretar fraudes, vandalismo, sabotajes, espionaje, invasiones y robos de información.
2. Involuntarios: Estas son amenazas que ocurren debido a fallos de usuarios, que pueden caer en el denominado phishing, o también en el fallo al momento de utilizar equipos de la red, esto debido a falta de conocimiento, ya sea al momento de manipular equipos o de las políticas de seguridad de la empresa en sí.
3. Físicas: Este tipo de amenazas ocurren cuando se tienen almacenamiento de datos sensibles o importantes, en los cuales se necesita el uso de compartimiento de identificaciones para acceder a los mismos, si algún “intruso” obtiene este tipo de identificación podría alterar o robar la información de estos almacenes de datos.

Además, es necesario saber los distintos tipos de amenazas que se pueden encontrar dentro de una red, las cuales son: [9]

1. Naturales: Son amenazas relacionadas principalmente con fenómenos naturales, debido a esto, es muy importante la elección del lugar donde se montará la red y se almacenarán los datos, debido a que deben ser infraestructuras capaces de soportar incendios, terremotos, huracanes, etc.
2. Hardware: Este tipo de amenazas ocurren, debido a hardware que no cuenta con capacidades requeridas por la empresa, tales como suficiente almacenamiento, procesamiento y velocidad, además, estas amenazas también están relacionadas con equipo que no reciban actualizaciones periódicas y constantes, así como, una mala configuración que se le dan a los equipos.
3. Almacenamiento: Los soportes de almacenamiento deben ser utilizados correctamente, esto debido a que podrían ser vulnerables a una serie de factores que pueden poner en riesgo su disponibilidad, integridad y confidencialidad.
4. Comunicación: Las amenazas que se encuentran en este punto, son debidos a los medios de comunicación, ya sea utilizando los medios físicos como el cobre, fibra óptica o los inalámbricos. Esto debido a que deben existir métodos de seguridad para

poder evitar que información importante dentro de las organizaciones se pierdan o, por otro lado, que sean interceptadas por personas no autorizadas. Por lo tanto, es necesario contar con encriptación en los mensajes, así como con equipos capaces de permanecer la mayor parte del tiempo disponibles para que no exista caída de comunicaciones.

5. Humanas: Este tipo de amenaza se denomina así debido a que personas pueden causar daños a la información, ya sea de manera intencional o no. Los factores por las cual sucede son principalmente por el desconocimiento y la mala práctica de las políticas de seguridad que existan en una empresa. Tales desconocimientos pueden ser, desde el caer dentro de un ataque de ingeniería social donde pueden obtener credenciales de las personas, uso de contraseñas débiles, falta de capacitación, etc.
6. Software: La amenaza de software es muy común, y más con la ampliación del internet y distintos programas que son llamativos para distintas personas, siendo provocadas al descargar de fuentes no confiables, lo que causa que internamente se ejecute algún tipo de código que muchas veces pasan desapercibidos hasta que es muy tarde y la seguridad de la red se ve comprometida.

1.5.3.3 Monitorización de Seguridad de Redes

En esta sección se parte del concepto principal de la monitorización de redes, el cual permite mediante la captura, procesos y análisis de paquetes el poder analizar varias métricas que van desde, la utilización del ancho de banda, las tasas de pérdida de paquetes, la cantidad de conexiones interrumpidas, etc. Para realizar la monitorización de una red es necesario tener un software el cual permita recopilar los datos que viajen a través de la red, analizarlos y poder proporcionar información sobre el funcionamiento de red. [15] Este proceso se vuelve fundamental especialmente en el ámbito empresarial, donde el principal objetivo es el poder encontrar los problemas de manera rápida y así reducir todos los tiempos de acción a solventar el problema en específico. [16]

Una vez entendido este concepto, se parte al más específico, el cual es la monitorización de seguridad de redes, el cual toma los principios de la monitorización de redes, pero aplicados principalmente a solventar los problemas de seguridad que se puedan presentar en la red, siendo básicamente un proceso automatizado que va a monitorizar a los dispositivos y al tráfico de la red en busca de vulnerabilidades de seguridades, amenazas y actividades “sospechosas” que se encuentran en la red. Como previamente se vio en la monitorización de redes, la monitorización de seguridad de redes es aplicada principalmente en el ámbito empresarial, y con los mismos objetivos, los cuales son el

solventar de manera más rápida los problemas de seguridad. El software de monitoreo de red, además, sirve para poder recopilar información acerca de la carga útil de la red, las sesiones de tráfico cifradas que existen, así como poder detectar patrones en los flujos del tráfico de la red. Adicionalmente, con este software es posible generar alertas e informes de seguridad automáticos, contando también con gráficos proporcionados por las herramientas de monitoreo de seguridad de redes para identificar de manera sencilla actividades irregulares en el tráfico de la red. Una vez establecido el monitoreo de redes y de seguridad de redes, se tienen que observar sus ventajas, esto debido a que usualmente se superponen para poder aprovechar al máximo las prestaciones de las dos y así lograr una red más segura y robusta. Por parte del monitoreo de las redes, se tiene el poder de monitorear la disponibilidad de la red, el cual es un concepto fundamental de las seguridades de red, en el otro punto, el monitoreo de las seguridad de redes permite proteger a la empresa u organización de posibles eventos de explotación de vulnerabilidades, además de que se analizan distintos factores tales como, la carga útil de la red, protocolos de red, comunicaciones cliente-servidor, patrones de tráfico, sesiones de tráfico cifradas y flujo de tráfico. [17]

1.5.4 Automatización de Redes

La automatización en el ámbito de las tecnologías de la información es fundamental, partiendo de la base en que cualquier proceso TI imaginable puede ser automatizado. La automatización consiste en desarrollar software para aquellos procesos que son manipulados por el ser humano y que se podrían replicar gracias a distintas herramientas como la programación, análisis de datos e inteligencia artificial, logrando así una menor interacción humana con los equipos de TI, teniendo una mejor integración a la tecnología y una gran cantidad de beneficios que se logran con la implementación de la automatización en entornos de las tecnologías de la información. [11]

El propósito de la automatización de procesos TI van de la mano usualmente en el mundo empresarial, debido a que se tiene una gran cantidad de equipos y una gran infraestructura, por lo que el desarrollo del software replicando los procesos que se tienen dentro de un sistema de manera automática, permite el ahorro de tiempo y de recursos para las empresas, logrando así que los trabajadores de TI no se enfoquen únicamente en un proceso si no que se dediquen a un trabajo más estratégico. [12]

En la automatización de redes se debe tener en cuenta distintos factores, tales como, el diseño de la red, debido a que es necesario conocer todo acerca de la topología, pudiendo así visibilizar de manera rápida y clara los dispositivos y clientes de la red, las políticas y

perfiles que se tienen en esta para poder cumplir con las mismas y satisfacer las demandas de la empresa, todo aquello para tener los beneficios que una red que cumpla con las especificaciones de automatización de redes puede tener. [13]

Estos beneficios en conjunto con los mencionados anteriormente son, una mayor resistencia, reducción del tiempo de inactividad, una mayor perspectiva y control de la red debido a que con las soluciones automáticas que se pueden presentar, se entenderá de mejor manera la red, lo que permitirá un mejor control de esta y también más capacidades para poder adaptarla según sea necesario. [14] Además, gracias a las distintas tecnologías que se pueden aplicar para la automatización de redes como la programación Python, análisis de datos e inteligencia artificial, se tienen aún más beneficios los cuales son:

- Reducir el riesgo de la red: Debido a la gran cantidad de equipos que se conectan a una red, es muy probable que existan errores humanos al momento de realizar la configuración de los dispositivos. Según un estudio realizado por un informe de CBC, un 47% de empresas señalaron a los errores humanos al momento de realizar la configuración de equipos como responsable directo de los problemas de filtraciones de datos. Por lo que al aplicar la automatización esos procesos se desarrollarán automáticamente, logrando que cada equipo sea configurado como se desee reduciendo el posible error significativamente, además, del tiempo invertido que se ahorra con respecto a la mano de obra de los trabajadores. [48]

- Cambios rápidos: Los cambios en la red con la automatización serán mucho más frecuentes y rápidos, así mismo, será posible crear notificaciones automáticas dentro de la red para así conocer de los problemas en esta y actuar de la manera más rápida posible, así mismo, con la automatización es posible desarrollar una red dinámica, que se ajuste de acuerdo con las necesidades del personal o del usuario de la red. [48]

- Redes más confiables: En este punto se aplica el principio de inteligencia artificial en conjunto con aprendizaje automático, con la automatización, una red puede ser autónoma hasta el hecho de poder aprender a predecir y prevenir problemas futuros que puedan afectar a la red. Esto gracias a la gran cantidad de datos que se puedan recopilar de dispositivos de red como los enrutadores, conmutadores y también equipos que usen la red como computadores, celulares, etc. [48]

- Optimización del rendimiento: Cuando se implementa la automatización de redes, también se cuenta con herramientas de soporte y de monitoreo de rendimiento de la red, si existen problemas de rendimiento, no es necesario resolverlo de manera manual,

si no que se puede utilizar un centro de operaciones de red para poder ayudar a todos los usuarios de la red con los problemas de rendimiento. Además, las herramientas de análisis y diagnóstico de la red, no solamente realiza un diagnóstico de la misma, si no que, en ciertas ocasiones, pueden proveer información y sugerencias de cómo mejorar el rendimiento de la red. [48]

- Gestión de red simple: Esto es una consecuencia de todo lo aplicado previamente, debido a la frecuencia con la que se realiza la monitorización la gestión de red se simplifica, así mismo, se vuelve más sencillo el hecho de saber los problemas existentes en la red y su posible solución. [48]

1.5.4.1 Inteligencia Artificial

La inteligencia artificial se define como el poder hacer que una máquina pueda realizar procesos que se asimilen a los de los seres humanos, incorporando grandes cantidades de datos, con el fin de poder entrenar a las máquinas para que estas mismas puedan relacionar las variables y así poder realizar predicciones futuras. [18] Por lo tanto, la inteligencia artificial es la rama de la informática que pretende replicar o simular la inteligencia humana, mediante software y hardware, recibiendo percepciones del entorno y realizando acciones a partir de estas. [19]

Existen cuatro enfoques alrededor de la inteligencia artificial:

- Pensar humanamente.
- Pensar racionalmente.
- Actuar humanamente.
- Actuar racionalmente.

Las primeras dos, se basan principalmente en el pensamiento y razonamiento, mientras que las últimas dos, pretender abarcar el comportamiento que tendría la máquina y que tan aproximado al comportamiento humano está. [19]

Así mismo, se tienen cuatro distintos tipos de inteligencia artificial, de acuerdo con la complejidad de tareas que pueden realizar:

- Maquinas reactivas.
- Memoria limitada.
- Teoría de la mente.
- Autoconciencia.

Las máquinas reactivas, son las que utilizan los principios más básicos de la inteligencia artificial, es decir, solamente puede reaccionar a lo que ocurre en la actualidad, sin ningún tipo de entrenamiento ni experiencias pasadas, sin embargo, es muy útil si se requiere para poder realizar tareas que sean repetibles. Los tipos de memoria limitada, es más compleja que la anterior, esta es capaz de poder recopilar información, esto con el fin de ser entrenada continuamente, para después crear un modelo, este modelo generalmente recibe una retroalimentación humana o ambiental, esta retroalimentación sirve como datos para seguir entrenándose y continuar con un ciclo. La teoría de la mente es completamente teórica, no se han visto aplicaciones inmediatas, y se basa en la premisa de querer que las máquinas comprendan los sentimientos humanos, animales e incluso de otras máquinas y tomar decisiones para poder lograrlo. Finalmente, la conciencia de sí mismo, esta etapa inicia inmediatamente después de la teoría de la mente, logrando así que las máquinas puedan tener conciencia de sí mismo, es decir, que contiene una conciencia prácticamente humana y comprende su propia existencia. [19]

Así mismo, la inteligencia artificial tiene un subcampo muy importante el cual tiene aplicaciones variadas, este es, el aprendizaje automático y su subcampo el aprendizaje profundo.

El aprendizaje automático, es el subcampo en el cual las máquinas tienen la capacidad de aprender mediante datos, estadísticas y prueba y error, con el propósito de poder optimizar procesos e innovar a un ritmo acelerado. El aprendizaje automático como tal, está permitiendo que las máquinas aprendan de manera similar a la humana, pudiendo así resolver problemas importantes con el de investigación de cáncer hasta el cambio climático. [20]

El aprendizaje profundo, es en esencia una parte del aprendizaje automático, en el cual se emplean las denominadas redes neuronales, ya que son basadas en los nodos biológicos de los seres humanos, con el fin de poder reconocer y procesar imágenes y voz de manera que luego se puedan representar esas imágenes y sonidos, generando una gran base de datos para posibles tareas futuras, es decir, el aprendizaje profundo, permite a las máquinas aprender como lo haría una persona, en base a la inmersión y ejemplo. [21]

1.5.5 Herramientas para la automatización de seguridad en redes

En esta sección se va a recopilar la información existente de las distintas herramientas que pueden ayudar a mantener segura una red de manera automática, esto de forma general aplicable a las redes tradicionales. En las herramientas que se pueden utilizar para poder

implementar la automatización de los procesos de seguridad, se tienen las siguientes categorías:

1. Automatización Robótica de Procesos (RPA): Está categoría es la más básica al momento de querer implementar soluciones de seguridades de redes. Esto debido a que solamente se realizan tareas rudimentarias de automatización, sin análisis inteligente, generalmente en estos procesos, se realiza el escaneo de vulnerabilidades, ejecución de herramientas de monitorización y almacenamiento de resultados y finalmente la mitigación de amenazas básicas, su principal desventaja es que no pueden integrarse con el análisis inteligente lo que limita la solución de seguridades. [47]

2. Orquestación de seguridad, automatización y respuesta (SOAR): Se define como pila de soluciones que permite la recopilación de datos sobre las amenazas de seguridad y poder responder a las mismas con soluciones que no requieren de asistencia humana, esta categoría admite flujos de trabajo automatizados, ejecución de políticas y automatización de informes. [47]

3. XDR: Denominado como una solución de detección y respuesta extendidas, este puede compilar datos de telemetría automáticamente, lo que provee a los analistas de red la información necesaria para investigar y responder a los ataques y amenazas. Además, se pueden integrar directamente con distintas herramientas de seguridad para ejecutar respuestas automatizadas. [47]

1.5.5.1 Monitorización de Seguridad de Redes.

Previamente, se estableció la definición de la monitorización de seguridad en redes, por lo tanto, es primordial, saber cuál es la necesidad del porque implementar la monitorización de seguridad de redes. Esto se basa en tres pilares fundamentales, la prevención, detección y respuesta. Sin embargo, la prevención termina pasando a un segundo plano al momento de querer implementar seguridad en las redes, esto debido a que en realidad es algo que debería venir por defecto en cualquier equipo o elemento de la red. Tales como, cifrado en la conexión, firewalls para evitar el acceso no autorizado, PKI (Infraestructura de clave pública) para evitar la suplantación de identidad. Esto se puede relacionar con un ejemplo sencillo, en donde, al querer comprar una caja fuerte que venga con la mayor cantidad de procesos de prevención, un ladrón que intente obtener el contenido de este podrá hacer en un tiempo, ya sea mayor o menor, pero lo conseguirá si es que no existe ningún equipo de

detección de robo. Por lo tanto, siempre es muy importante contar con medios para poder detectar cuando se está llevando a cabo una actividad de este tipo, así mismo, tener con claridad los tipos de respuestas que se deban tener al momento de que se detecten los ataques. [24]

Por ello, es primordial el entender de la importancia que tiene el software para monitorizar la seguridad de redes, debido a que estos cumplen con la función de detección y por ende posteriormente la de posible respuesta que se deba aplicar al momento de que ocurra un ataque. Además, los mecanismos de prevención no son perfectos, y siempre se tiene una alta probabilidad de fallo, esto debido a que existen equipos con errores de seguridad, con una mala configuración, así mismo se tienen los errores de los usuarios que no permiten mantener una red segura ante cualquier tipo de amenaza, mientras que gracias a la detección y respuesta se pueden reducir gastos de prevención considerablemente y evitar la fragilidad que tienen estas redes. [24]

En la Tabla 1.1, se presentan las principales herramientas de software que se dedican al monitoreo de seguridad de redes, donde se realiza un enfoque en sus principales características y lo que los diferencian entre ellos, si son gratuitas o de paga y los sistemas operativos en los que se pueden utilizar. [25]

Tabla 1.1. Herramientas de monitorización de seguridad de redes.

Nombre	Licencia	Sistema Operativo	Características
Argos	Gratuita	Linux	Este software realiza el análisis eficiente y completo de los datos de las redes, realizando mediante informes rápidos y completos el filtro de grandes porciones de tráfico.
P0f	Gratuita	Linux	Es un software ligero, rápido y limpio. Tiene una desventaja pequeña debido a que no ha recibido muchas actualizaciones, sin embargo, esto debido a que no ha sido necesario. Esta herramienta no genera tráfico adicional para la monitorización. Además, puede identificar los sistemas operativos con lo que interactúa.

Nagios	Paga	Windows Linux Mac OS/OSX	Esta herramienta, permite monitorear hosts, sistemas y redes, donde su principal característica es el de poder entregar alertas en tiempo real. Además, que los usuarios puedan personalizar exactamente las notificaciones que requieran. Así mismo, pueden monitorizar entre otros los siguientes servicios de red, como HTTP, NNTP, ICMP, POP3 y SMTP.
Splunk	Paga	Windows Linux Mac OS/OSX	Esta herramienta cuenta con las ventajas del análisis en tiempo real como para los datos históricos que ocurren dentro de la red. Además, cuenta con una interfaz gráfica que es muy sencilla de utilizar y que permite que las soluciones de monitorización sean más fáciles. Así mismo, tiene paquetes que pueden utilizarse de manera gratuita, sin embargo, son muy limitadas.
OSSEC	Gratuita	Windows Linux Mac OS/OSX Solaris FreeBSD OpenBSD NetBSD	Básicamente esta herramienta proporciona soluciones para realizar análisis inteligente en tiempo real, mediante los eventos de seguridad del sistema. Tiene una gran comunidad la cual comparte estrategias para poder implementar seguridad en las redes. Así mismo, puede utilizar un plugin llamado “Atomicorp”, la cual contiene un elemento denominado autocuración el cual sirve para corregir automáticamente las vulnerabilidades que se detecten.

1.5.5.2 Herramientas de cifrado.

Las herramientas de cifrado son un punto clave para mantener seguros los datos que se pretenden enviar a través de una red, es importante mencionarlos, debido a que formaran parte indispensable al implementar en las seguridades de redes, debido a que siempre es

necesario herramientas que de manera automática cifren los datos y los almacenen de manera segura, datos tales como, contraseñas, números de tarjeta de crédito e incluso mensajes sensibles. [26]

A continuación, se presentan las herramientas de cifrado que más se utilizan en el mercado, en la Tabla 1.2, se muestran estas herramientas, pretendiendo mostrar su tipo de licencia y sus características principales. [25]

Tabla 1.2. Herramientas de cifrado.

Nombre	Licencia	Características
Tor	Gratuita	Esta herramienta permite garantizar privacidad al querer navegar en Internet, debido a que el sistema enruta las solicitudes a los servidores web proxy, dificultando el seguimiento que exista al usuario. Es recomendable tener un alto conocimiento del uso de esta herramienta, debido a que pueden existir nodos de salida maliciosos de rastreo de tráfico.
KeePass	Gratuita	Generalmente utilizado en entornos de oficinas, debido a que es un sistema de gestión de contraseñas. Su principal característica es la de poder ingresar a todas las cuentas con una sola contraseña, debido a que permite la opción de autocompletar gracias a una contraseña maestra.
TrueCrypt	Gratuita	A pesar de que se mantiene desactualizado y abandonado por su creador, sigue siendo una herramienta solida al momento de cifrar información en capas con dos niveles de control de acceso.

1.5.5.3 Herramientas de análisis de vulnerabilidades.

Estos tipos de herramientas son muy importantes al momento de querer implementar la automatización de seguridad de redes, debido a que se encargan y están diseñadas para

buscar automáticamente las amenazas y analizar las vulnerabilidades que estén existentes en la red, generalmente, estas herramientas suelen realizar tareas como, escanear aplicaciones web para probar y simular patrones de ataques conocidos, escanear protocolos, puertos y servicios de red vulnerables, algunas otras se encargan de escanear la red para visualizar las redes y generar señales de advertencia cundo se presentan direcciones de red IP extraviadas, paquetes que han sido falsificados, así como, la generación de paquetes sospechosos de una sola dirección IP. [27]

En la Tabla 1.3, se presentan las herramientas de análisis de vulnerabilidades más importantes y utilizadas, prestando atención a sus características más importantes, el sistema operativo en donde se pueden utilizar y su tipo de licencia, es decir, si es gratuita o de paga. [25]

Tabla 1.3. Herramientas de análisis de vulnerabilidades.

Nombre	**Licencia**	**Sistema Operativo**	**Características**
Burp Suite	Paga	JAR Linux Mac OS/OSX Windows	Como tal, esta herramienta, es un escáner de seguridad de redes en tiempo real, y diseñado principalmente para poder identificar las vulnerabilidades críticas, así mismo, permite simular ataques de seguridad para poder ver las principales vulnerabilidades que se tienen en la red.
Nikto	Gratuita	Windows Linux	Esta herramienta se basa en escanear servidores web y redes en busca de coincidencias con una base de datos de más de 6400 amenazas, por lo que las vulnerabilidades más

			comunes estarían verificadas y automáticamente serían identificadas. Además, esta base de datos se actualiza periódicamente.
Paros Proxy	Gratuita	Windows Linux Mac OS/OSX	Esta herramienta se basa en varias aplicaciones útiles para poder ejecutar pruebas de seguridad, tales como, la araña web, la cual permite inspeccionar las páginas de sitios web automáticamente, almacenar información relevante de las mismas y crear registros de las páginas automáticamente, así mismo, se tiene un registrador de tráfico y un escáner de vulnerabilidades.
NMap	Gratuita	Windows Linux FreeBSD OpenBSD Solaris IRIX Mac OS/OSX HP-UX NetBSD Sun OS	Esta herramienta es indispensable para poder tener una mejor visión de los puntos débiles que tiene la red, además, de tener un mayor acercamiento a la red, como el poder saber los hosts de la red, los servicios que se están utilizados en esta, los filtros de paquetes que están en uso y otras características.
Nessus Professional	Paga	Windows Linux Mac OS/OSX	Esta herramienta se encarga de identificar y reparar vulnerabilidades detectadas en la red, así mismo, instalar

			parches faltantes, identificar errores de software, también detectar las configuraciones incorrectas que se han realizado en aplicaciones, dispositivos y sistemas operativos.
Nexpose	Paga	Windows Linux	Con esta herramienta, se puede buscar, detectar y reducir los puntos débiles que se puedan presentar en la red, así mismo, permite observar la red en tiempo real. Además, permite categorizar las vulnerabilidades de acuerdo con su nivel de riesgo.

1.6. Trabajos Relacionados

A continuación, se presentan los trabajos relacionados de los cuales se puede tener una idea de las distintas soluciones que se pueden encontrar a la problemática de seguridades que se tienen en redes nuevas como las de IoE e IoT, haciendo uso de fuentes tales como, ResearchGate o Suresh Gyan Vihar University, para tener una visión general de cómo se trata el tema actual.

En [22], se realiza una recopilación de información, acerca de, cómo identificar las amenazas existentes en las redes IoT, así como los distintos problemas al momento de querer automatizar la seguridad, y la confianza actual que tienen las empresas con las redes IoT. Así mismo, se tiene una idea en general del potencial que se tiene al momento de querer automatizar las seguridades de redes, así como, los mecanismos que se tienen para poder desplegar de una manera coordinada todos los recursos necesarios para poder satisfacer las soluciones posibles con respecto a las seguridades en este tipo de redes.

En [23], se realiza un análisis de la masificación que las redes IoE permiten, debido a su propio término el cual pretende conectar absolutamente todo en una red, y como esto

conlleva a posibles brechas de seguridad que se puedan encontrar, estas principalmente se deben a que cada dominio conectado a la red van a tener distintas políticas de seguridad, privacidad y confianza, por lo tanto en este artículo se pretenden varios desafíos que se deben solventar para poder mantener una red IoE segura, estos desafíos son; privacidad del usuario y protección de datos en IoE, autenticación y gestión de identidad, gestión de confianza e integración de políticas, autorización y control de acceso, seguridad de extremo a extremo y soluciones de seguridad respecto a ataques. Cada uno de estos desafíos están bien presentes en los retos que presentan este tipo de redes debido a la gran conectividad de distintas "cosas" dentro de la red.

2 METODOLOGÍA

La metodología por emplearse es dividida en etapas o fases, de acuerdo con el planeamiento realizado para el trabajo de integración curricular, inicialmente en la primera etapa, se tiene la información del marco teórico, que servirá para tener una idea general de lo que realmente se quiere investigar, la segunda etapa consistiría en la recolección de las herramientas que pretenden solucionar las problemáticas de la seguridad de redes, así como poder realmente saber cuáles se ajustan a la solución real que se quiere dar, la cual es la automatización de seguridades, así mismo, el poder utilizar herramientas de programación que funcionarían inmediatamente después de ocurrido un evento para poder solventar la problemática de seguridades inmediatamente o lo más rápido posible, e incluso ver la factibilidad de aplicar tendencias actuales como la de la inteligencia artificial, finalmente en la última etapa se eligen las mejores herramientas, comparándolas entre sí para así poder determinar cuáles funcionarían correctamente dentro de la problemática de la seguridad en redes y que se pueden aplicar a las redes del Internet de Todo (IoE).

Este trabajo está basado en un tipo de investigación exploratorio, debido a la poca información y conocimiento que se tiene con respecto a las redes IoE, sin embargo, al saberse de lo más general que son las redes de datos tradicionales, es posible aplicar las soluciones dentro de las redes IoE. Además, para la recolección de la información se utiliza el análisis documental, por el cual, mediante fuentes bibliográficas como artículos de investigación o páginas de Internet, se recolecta la información para poder presentar las soluciones que se deben tener al momento de querer automatizar las redes del Internet de Todo.

2.1 Principios para la automatización de seguridad en redes IoE

Para poder iniciar con el proceso de la automatización de seguridad de redes es necesario conocer dos principios iniciales, los cuales son las vulnerabilidades más comunes que se encuentran dentro de las redes IoE y las herramientas más útiles que funcionan con los dispositivos del Internet de Todo.

2.1.1 Vulnerabilidades más comunes que afectan a las redes IoE

Es necesario conocer sobre las vulnerabilidades más comunes en las redes IoE, debido a que cualquier red existente presenta un sin número de vulnerabilidades que a pesar de ser comunes no siempre se toman en cuenta, las redes IoE no están exentas a estas vulnerabilidades, a continuación, se presentarán las más comunes que se deberán tener en cuenta al querer implementar la automatización de seguridad en estas redes:

- Contraseñas débiles: Esta vulnerabilidad, en general es causada gracias a una negligencia humana al momento de realizar la configuración de equipos, en varias ocasiones se suele configurar los dispositivos con contraseñas que son fácilmente adivinables o incluso se dejan las contraseñas predeterminadas de fábrica. [50]

- Servicios de red inseguros: Al momento de implementar una red IoE como se trata de una gran cantidad de dispositivos, es posible que en alguno de ellos se instalen servicios innecesarios e inseguros, que solamente comprometerá toda la red, sin embargo, el problema de esta vulnerabilidad es que en ciertas ocasiones se ingresa a un área gris al no poder determinar cuándo un servicio es innecesario o inseguro. [50]

- Interfaces de ecosistemas inseguros: A pesar de que se pueda asegurar robustamente un dispositivo IoE, también se deben considerar todos los servicios y componentes con el que se comunica el dispositivo, esto incluye elementos de software que hacen que el dispositivo sea accesible y logran que los usuarios pueden interactuar con el mismo. [51]

- Falta de un mecanismo de actualización seguro: No se utilizan canales encriptados seguros para la actualización de firmware de los dispositivos de la red, por lo que las actualizaciones no son realizadas de forma segura y confiable, además que con un buen mecanismo de actualización es posible conocer cuando una actualización fallo y tomar

acción inmediata, si no existe un mecanismo, la actualización no se ha de efectuar y es posible que eso marque una brecha de seguridad en la red. [51]

- Uso de componentes inseguros o desactualizados: En general al momento de implementar seguridades en redes, es necesario mantener a los equipos actualizados, así como al software que utilizan, esto debido a que posiblemente versiones anteriores de software puedan presentar brechas de seguridad que fueron solucionadas en futuras actualizaciones, si no se actualiza los dispositivos simplemente las brechas de seguridad seguirán ahí y comprometerá a toda la red. [51]

- Protección de privacidad insuficiente: En ciertas ocasiones, la información personal no se mantiene privada, varios usuarios de las redes optan por guardar información valiosa como contraseñas o claves de tarjeta de crédito dentro de un equipo compartido lo cual anula por completo la privacidad que se desea tener ya que otras personas podrían encontrar esos datos. [50]

2.1.2 Herramientas para la automatización de la seguridad en redes IoE

A continuación, se enumeran las distintas herramientas que se pueden utilizar para poder implementar la automatización de seguridad en las redes IoE, es posible que muchas de estas herramientas sean aplicables para las redes IoT, además, es imprescindible recordar que no todos los dispositivos que forman parte de las redes IoE pueden utilizar estas herramientas debido a que algunos de estos son de bajo procesamiento, por lo que estas herramientas en específico actúan sobre los dispositivos que tengan la capacidad de realizar niveles de procesamiento más altos. Previamente se especificaron ciertas herramientas para poder seguir implementando seguridad en redes, las cuales son aplicables para el uso en el entorno de las redes IoE, sin embargo, es mejor conocer cuáles son aquellas herramientas dedicadas específicamente a las redes del Internet de Todo.

A continuación, se presentan las siguientes herramientas, junto con sus características más destacables:

- Allot: Es una de las herramientas más utilizadas en el entorno del IoT, por lo que su aplicabilidad es más que posible al momento de ser utilizadas en las redes IoE, sus principales características son las de una herramienta de monitorización de seguridad en las cuales se pueden configurar alertas de acuerdo con el impacto que tendrían en el sistema, se puede realizar acceso remoto hacia los dispositivos de la red, además,

con esta herramienta es posible la ejecución de tareas de acuerdo a un evento generado en la red. [49]

- Sequitur Labs: Esta herramienta sirve como la base para poder montar una red IoE segura, debido a su facilidad que presenta para el diseño, producción y gestión de ciclo de vida de los dispositivos que forman parte de la red, sus principales características son, proteger los activos contra robos y extravíos, reduce los tiempos de inactividad que se puedan tener en la red, además de asegurar que los dispositivos siempre estén disponibles. [49]

- Sternum: Una de las herramientas más importantes al momento de querer implementar automatización en la seguridad en redes IoT y por lo tanto en redes IoE, esto debido a que su principal uso es en el campo de la medicina, donde la seguridad es primordial para poder garantizar un correcto funcionamiento de los dispositivos y la seguridad de los usuarios de la red, su principal característica es la combinación en conjunto con el procesamiento de datos para posteriormente realizar predecir posibles problemas de la red y ejecutar automáticamente las soluciones requeridas. [49]

- Subex: Esta herramienta es imprescindible si se quiere recolectar la mayor cantidad de información acerca del estado de la red, los dispositivos conectados, sus características, problemas que se encuentren en la red, la herramienta en general permite obtener datos confiables que posteriormente se pueden utilizar con otras herramientas o lenguajes de programación para poder usar los datos y con distintas soluciones como inteligencia artificial solventar los problemas automáticamente. [49]

2.2 Programación en la automatización de seguridad en redes IoE

El uso de lenguajes de programación es muy importante para poder realizar la automatización de seguridad en redes, por lo tanto, es necesario realizar un estudio del lenguaje de programación Python para la automatización de seguridad en redes, a continuación, se presenta como es el funcionamiento de este lenguaje en el ámbito de la automatización de las redes IoE.

2.2.1 Python en la automatización de seguridad en redes IoE

En esta sección, se determina la viabilidad de utilizar la programación al momento de querer automatizar las seguridades en las redes y para ello, el principal lenguaje de programación

es el de Python. La razón por la cual se utiliza un lenguaje de programación para este tipo de aplicaciones es debido a que las herramientas preexistentes no suelen prestar todas las características de seguridad que se desean implementar en una red, además de que el desarrollador de la aplicación de seguridad conocerá por completo el funcionamiento de la herramienta, además al ser más específico con los requerimientos que se deseen, los tiempos de reacción al momento de que exista un ataque serán menores que al utilizar una aplicación de terceros y mucho menor que no utilizar herramientas de automatización. Así mismo, es posible utilizar combinar herramientas propias con otras para poder aprovechar todas las características y mejorar el desempeño de la automatización de seguridad de redes. [28]

Python como tal, tiene un beneficio que es atractivo al querer ser implementado en ambientes de seguridad de redes, esa es la posibilidad de poder crear herramientas de red, así mismo, la posibilidad de utilizar módulos muy útiles, como Scapy Python, el cual permite el desarrollo de un escáner de puertos y comandos de trazado de rutas, además de poder crear y modificar varios tipos de paquetes de red, y si se desea, implementar distintas funciones para capturar, rastrear y ejecutar acciones sobre los paquetes de red. Así mismo, este módulo de Python permite ejecutar tareas forenses de red, tal como, detectar ataques de inyección SQL, mediante el análisis de los paquetes, logrando así detectar e identificar cuando, donde y como se realiza el ataque, otra de las aplicaciones que se puede realizar con este módulo es detectar ataques de extracción de credenciales a servidores FTP. [29]

Adicionalmente, Python es capaz de poder automatizar el análisis de archivos PCAP, las cuales se obtienen de otras herramientas de análisis como Wireshark, TCPdump, Ettercap, etc. Estas herramientas por si solas no son capaces de ejecutar acciones, pero con el uso de Python, se pueden analizar mediante código y ejecutar acciones si se identifican actividades anómalas en la red. Así mismo, es posible utilizar modelos de aprendizaje automático basados en las librerías de Scikit-Learn, para poder automatizar aún más la ejecución de acciones al momento de detectar actividades extrañas en la red, debido a que estos modelos se utilizarían primordialmente para poder predecir con una cantidad de datos si va a ocurrir algún tipo de ataque a la red, y una vez detectado ejecutar algún tipo de programa para poder evitarlo. [29]

A continuación, se presentan las bibliotecas más usadas en el entorno de Python para poder implementar seguridad en redes:

1. NMap

Esta biblioteca estuvo previamente descrita como una herramienta de análisis de vulnerabilidades, por lo que se integra con Python para poder aprovechar los resultados que se obtienen al momento de buscar vulnerabilidades en la red, generalmente, esta biblioteca es utilizada para crear un sistema de detección de intrusos, donde automáticamente la aplicación recopila información para saber si existen nuevas conexiones a la red, y así generar alertas a toda la red de nuevas conexiones existentes. [30]

2. Scapy

Esta herramienta, que ya fue descrita previamente, es muy importante al momento de querer implementar seguridad en redes, principalmente porque a pesar de tener similitudes con NMap, esta también tiene la posibilidad de analizar paquetes PCAP, pudiendo así analizar el tráfico de la red, determinar si existen anomalías en la red y alertar a toda la red de este tipo de tráfico y tomar las medidas necesarias para poder evitar algún posible ataque. [31]

3. Yara

Esta herramienta es capaz de poder identificar rápidamente los patrones de los datos, generalmente es utilizado en conjunto con Python para poder detectar automáticamente malware que ingresa a la red, de acuerdo con los criterios que se configuren en la aplicación. [32]

2.2.2 Inteligencia Artificial en conjunto con Python para la automatización de seguridad de redes IoE

La seguridad en redes genera una gran cantidad de información, como se vio previamente, utilizando la herramienta de Python, se puede encontrar una gran cantidad de datos, que van desde puertos habilitados, conexión de hosts, paquetes que fluyen a través de la red, etc. Toda esa información recopilada necesita de un concepto que permita utilizar esa gran cantidad de datos, con el fin de poder realizar su agrupación, clasificación, procesamiento filtrado y administración, esto solamente es posible con la inteligencia artificial. Así mismo, en conjunto con la inteligencia artificial, utilizar el aprendizaje automático, el cual analiza a partir de datos del pasado, con el propósito de ofrecer soluciones optimas en el presente e incluso en el futuro. Por lo tanto, se pretende entrenar a los algoritmos de aprendizaje automático, es decir, proporcionar información al algoritmo acerca de las instrucciones y patrones que debe seguir, por lo que finalmente, el algoritmo pueda diferenciar de una

manera rápida y fácil entre una situación normal dentro de la red y una situación donde la seguridad de la red este comprometida, logrando así detectar automáticamente la amenaza. [33]

Una de las principales aplicaciones que se pueden desarrollar utilizando inteligencia artificial y aprendizaje automático es el de un sistema de detección y predicción de intrusos dentro de una red. Esto se puede realizar gracias a la cantidad de datos, patrones e instrucciones que serán la base para generar un algoritmo de aprendizaje automático que pueda predecir cuándo un usuario no autorizado intente ingresar a la red. Este tipo de algoritmo funcionaría como un modelo predictivo denominado clasificador, y podría distinguir entre “malas conexiones” y “buenas conexiones”, cabe recalcar que existen bases de datos disponibles para poder entrenar al modelo con respecto al tráfico que fluye a través de la red y que permiten distinguir entre “malas conexiones” y “buenas conexiones”. Así mismo, como existe un procesamiento de datos, es necesario utilizar librerías de datos, tales como NumPy y Pandas. [34]

El uso de la inteligencia artificial también permite el poder desarrollar modelos que detecten y clasifiquen malware de manera automática, esto utilizando un subcampo denominado redes neuronales, existen varias formas demostradas de poder aplicar este tipo de modelo, una de ellas es el estimar cual sería el efecto del malware analizando los códigos de sus archivos ejecutables y posteriormente clasificando el malware en cuestión. Así mismo, utilizando un concepto similar, se utilizan las redes neuronales para poder analizar grandes cantidades de datos provenientes de registros de eventos del sistema para poder detectar malware que ha sido persistente en el mismo. Adicionalmente, se mostrado cómo es posible el utilizar el aprendizaje automático en dispositivos móviles basados en Android para poder detectar malware, esto gracias a las denominadas “llamadas” del sistema. [35]

2.2.3 Cisco DevNet Express Security

En esta sección se realiza el procedimiento y análisis de un caso de estudio de la automatización de seguridades de redes IoE con equipos Cisco, con el propósito de determinar la viabilidad del caso de estudio y su implementación en las redes del Internet de Todo.

Según Cisco, esta aplicación de automatización de seguridad de redes se denomina evento de DevNet Express y forma parte del centro de laboratorios de aprendizaje, se denomina así debido tres principios fundamentales, los cuales son, escuchar, aprender y poner en práctica, esto en base a las soluciones de seguridad y flujos de trabajo para la detección de

amenazas. Además, con esta aplicación se puede reducir los tiempos de reacción en el momento en el que la red se encuentre siendo atacada mediante la automatización de las operaciones de seguridad de redes, logrando así el objetivo principal de la automatización de seguridad de redes el cual es la detección temprana, consistente y repetible de amenazas que afecten a la seguridad de redes, mediante el uso de herramientas de seguridad de Cisco y sus API correspondientes. [52]

En DevNet Express para Seguridad, se cuentan con distintos módulos que permitirán realizar todo un proceso para poder implementar la automatización de seguridad en redes, entre estos módulos se encuentran de acuerdo con su orden de uso, cazador de amenazas, respuesta a amenazas mediante SecureX y seguridad de aplicaciones de Cisco. [53]

2.2.3.1 Cazador de Amenazas

En este módulo se utilizarán cuatro herramientas esenciales, Cisco Secure Endpoint (denominado previamente como AMP), Cisco Umbrella, Cisco Secure Malware Analytics (anteriormente ThreatGrid) y Cisco SecureX, en el cual se desarrollarán habilidades como, automatizar el flujo de trabajo de día cero gracias a la vinculación de equipos Cisco que se consigue mediante el uso de las API, uso de AMP para poder recopilar toda la información posible detrás de un ataque, desde sus características hasta sus firmas, validación de toda la información recopilada mediante ThreatGrid y Umbrella, uso de Umbrella y AMP para proteger y contener la amenaza, uso de ThreatResponse para la investigación de respuestas y creación de flujos de trabajo automatizados utilizando Python. [54]

El módulo de Cazador de Amenazas tiene como misión el poder realizar una caza de todas las amenazas de seguridad que se encuentren en la red, además del uso de herramientas que permitan la automatización de la búsqueda de amenazas lo que permitirá que los tiempos de reacción sean mínimos, lo que se refleja en menor cantidad de costos operativos y la aceleración de procesos de seguridad de red. A continuación, en la Figura 1.1 y Figura 1.2, se presentan los dos marcos más conocidos acerca de la Caza de Amenazas, los dos de bucle continuo, cabe recalcar que muchos de los aspectos de los bucles se pueden automatizar, como el enriquecimiento o las respuestas automáticas. [54]

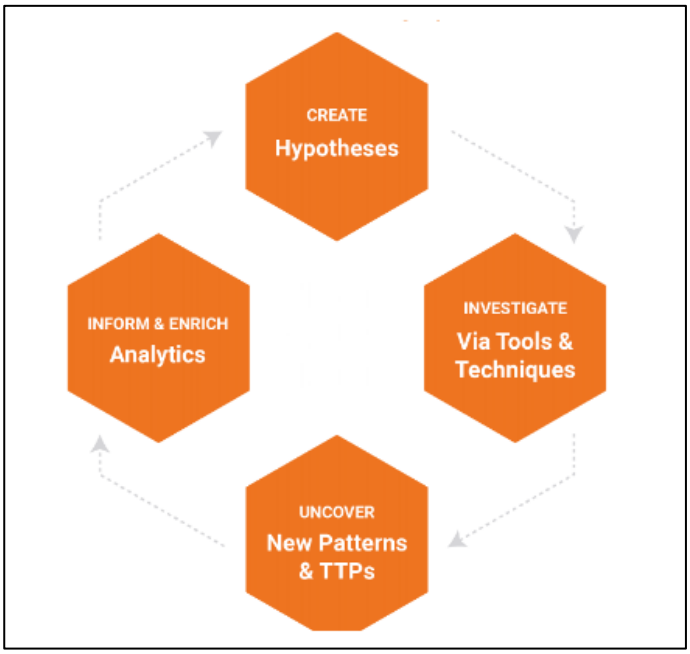


Figura 1.1. Bucle del Cazador de Amenazas.

Nota: En la Figura 1.1 se muestra el bucle que se utiliza en el cazador de amenazas, en donde se corresponde la creación de hipótesis, investigación mediante herramientas y técnicas, descubrimiento de nuevos patrones e informes y enriquecimientos mediante analíticas. [54]

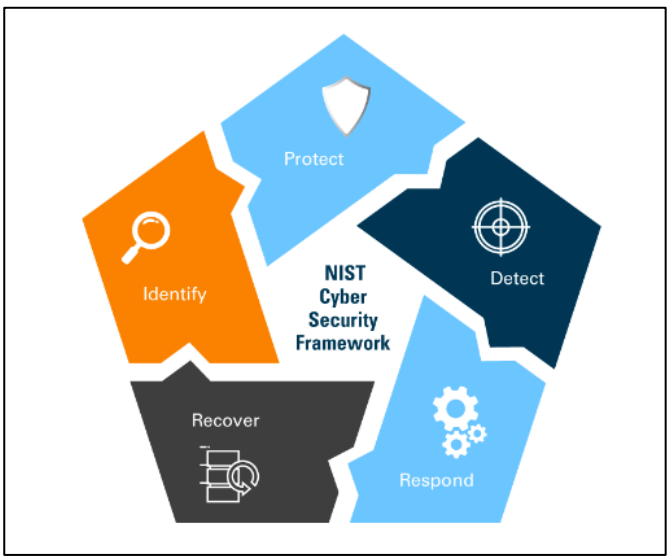


Figura 1.2. Marco de Ciberseguridad NIST.

Nota: En la Figura 1.2, se muestra el marco de ciberseguridad NIST (Instituto Nacional de Estándares y Tecnología), el cual se basa en detectar, responder, recuperar, identificar y proteger. [54]

Con respecto al procedimiento que se realiza en este módulo, se cuentan con las siguientes prácticas, mediante el uso de Cisco Secure Endpoint se desea obtener una lista de eventos los cuales se categorizan en “Amenaza detectada” y “Malware ejecutado” para un dispositivo en específico y se desea capturar un SHA-256 malicioso que este asociado al primer evento. La siguiente práctica consiste en el aislamiento del dispositivo que presenta problemas de seguridad verificando que el aislamiento ha sido exitoso. A continuación, se debe bloquear el archivo malicioso con una lista personalizada de detección simple de seguridad de punto final, cuyo propósito es el de bloquear la ejecución del archivo en otros dispositivos. En el siguiente paso de la práctica, se utiliza la herramienta Secure Malware Analytics para realizar una búsqueda de todas las muestras asociadas a SHA-256 malicioso que se capturó previamente para después consultar el informe de análisis de la primera muestra de la lista. Después, se utiliza la herramienta ThreatGrid para generar una muestra específica de todos los dominios de la red y se almacenan dentro de una matriz para poder obtener la mayor cantidad de datos posibles. El siguiente paso corresponde a verificar todos los dominios asociados mediante el uso de la herramienta de Umbrella Investigate (usando el enriquecimiento de SecureX) para la recuperación del estado de los dominios. Finalmente, el módulo concluye con el uso de la herramienta Umbrella Enforcement para la publicación de eventos de malware en la API con la finalidad de procesar estos eventos y si se desea agregar elementos a las listas de dominios de un cliente. [54]

Dentro del módulo de Cazador de Amenazas, se realiza una revisión de Cisco Secure Endpoint (AMP), esto debido a que la herramienta nos permitirá dar una protección integral a la red contra amenazas avanzadas, debido a que se basa en tres principios, la prevención, detección y respuesta, por lo que esta herramienta en esencia permite detener malware, eliminar puntos ciegos de seguridad de la red y el descubrimiento de amenazas desconocidas. El uso de estas tecnologías pretender brindar soluciones para distintos dispositivos que forman parte de las redes IoE, tales como computadores y celulares, debido a que pueden funcionar en sistemas operativos como Windows, macOS, Android, iOS o Linux. Además, comprenden dos tipos de tecnologías adicionales como la de prevención de memoria y la de detección de disco. En la prevención de memoria se tiene la prevención de exploits y protección, mientras que en la detección de disco se tiene Cisco Secure Endpoint Cloud, TETRA, protección contra actividades maliciosas y detecciones personalizadas. [55]

En el Anexo 1, se podrá encontrar el código de Python para generar el Cazador de Amenazas, mientras que en Anexo 2, se encuentra el código del grid de Cazador de Amenazas.

A continuación, se muestra el resultado al momento de ejecutar las prácticas realizadas en este módulo, en la Figura 1.3 se muestra la obtención de cada evento con su nombre y su ID asociado en donde se indica amenaza detectada (“Threat Detected”), con estos datos se puede realizar filtros de los eventos.

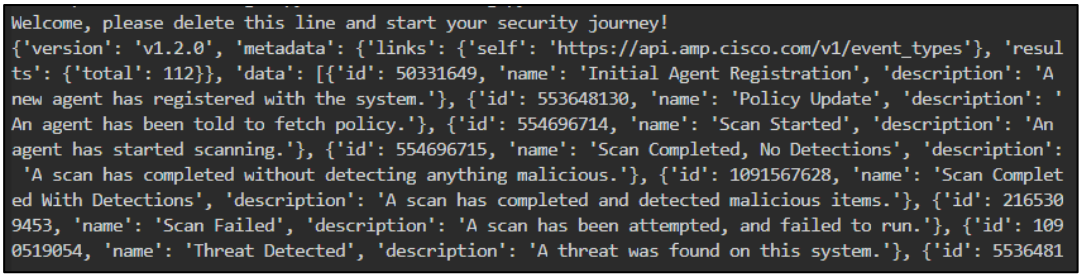


Figura 1.3. Visualización de amenazas mediante Cisco Secure Endpoint.

En la Figura 1.4 se muestra el resultado al querer observar los equipos que tienen problemas de seguridad en la red, donde se observa el GUID del conector de punto final de Cisco, nombre del host, sistema operativo, IP internas, externas y dirección MAC, política a la que perteneces los equipos, grupo al que pertenecen, y su última conexión visible de la red. [55]

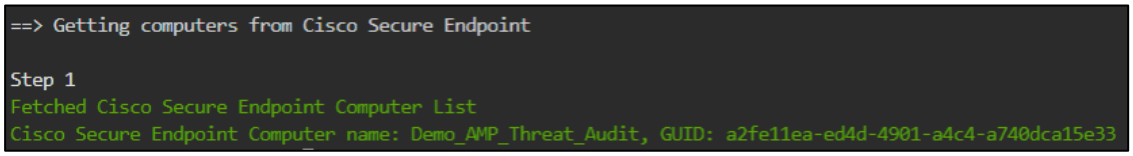


Figura 1.4. Visualización de equipos que contienen amenazas.

El siguiente paso es realizar llamadas para recuperar eventos de Cisco Secure Endpoint, en donde se pueden detectar las últimas amenazas avanzadas y se realiza el envío de alertas relevantes. En la Figura 1.5 se puede observar los eventos detectados de Cisco Secure Endpoint, donde se puede observar que se han encontrado 63 eventos, el primer evento es de detección de amenaza, el nombre del archivo malicioso es “ekjrngjker.exe” y también se puede observar su nombre SHA-256.

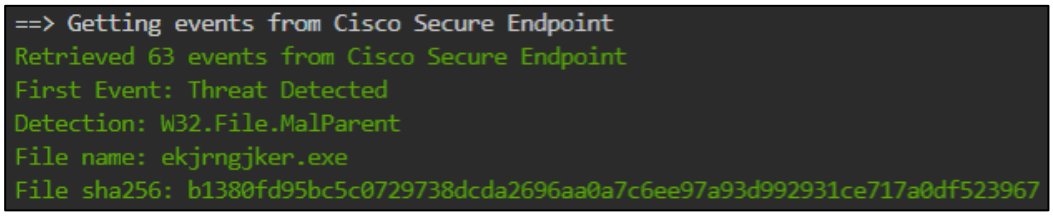


Figura 1.5. Detección de eventos de Cisco Secure Endpoint.

El siguiente paso es el de aislar el dispositivo que se encuentra atacado, se pueden tomar dos opciones, el ubicarlo en cuarentena en un grupo de clasificación el cual es útil cuando los datos históricos y datos nuevos generan falsos negativos, la otra opción es limitar la conexión utilizando Endpoint Isolation, esta herramienta permite bloquear la actividad de red entrante y saliente del dispositivo afectado (con Sistema Operativo Windows), para evitar filtraciones de datos y propagación de malware, a pesar de este bloqueo, no se afectará la comunicación entre el dispositivo Windows y Cisco Secure Endpoint Cloud, por lo que si se desea se detiene el aislamiento mediante un comando DELETE con los mismos parámetros que se configuraron en API REST. [54] En la Figura 1.6 se observa la salida de que se tiene al momento de ver el estado de aislamiento de un dispositivo, en este caso no se encuentra aislado. Sin embargo, después de ejecutar el script, si se ejecuta nuevamente se puede observar cómo ha sido aislado.

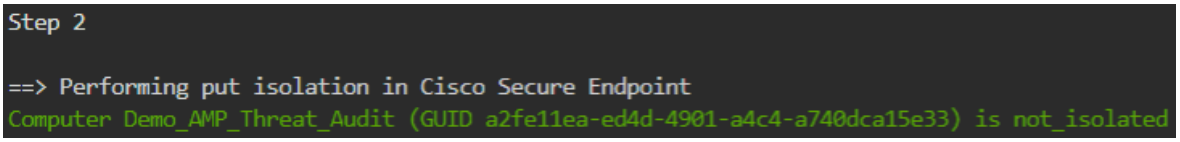


Figura 1.6. Estado de aislamiento del dispositivo amenazado.

Así mismo, en la Figura 1.7 se puede observar como el dispositivo comprometido ha sido aislado y el SHA-256 malicioso ha sido aislado también.

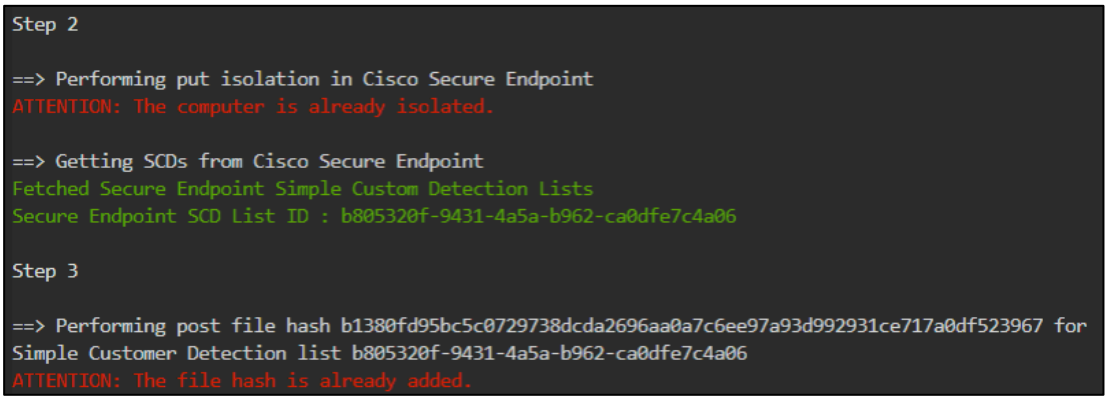


Figura 1.7. Dispositivo comprometido aislado y SHA-256 malicioso agregado.

La siguiente fase en el módulo de Cazador de Amenazas es el uso de la herramienta Secure Malware Analytics (ThreatGrid), el cual es conocido por ser una plataforma de análisis de malware en donde se combina análisis inteligente dinámico y estático de amenazas. El análisis de amenazas tiene dos métodos principales, el cual es el estático y dinámico. En el análisis estático, en donde se pretende analizar el contenido de un archivo en específico, esté análisis identifica la información que los motores de análisis tienen sobre el archivo y se compara si el archivo es malicioso o no. En el análisis dinámico en cambio, se ejecuta el malware en un entorno especial para que no afecte al resto de equipos, se realiza un análisis de lo que el malware realizará, como los archivos que modifica, llamadas a bibliotecas y procesos comprometidos. En la práctica, el objetivo principal es el de recopilar datos de la API Cisco Secure Malware Analytics y utilizar estos para búsqueda de información específica sobre cómo está incidiendo el malware en el equipo. [56]

Al momento de ejecutar ThreatGrid, se puede realizar una vista de las fuentes de análisis de malware, en la Figura 1.8 se puede observar la salida de ThreatGrid.

Figura 1.8. Fuente de datos de análisis de malware.

El siguiente paso es el uso de Umbrella, el cual provee la primera línea de defensa contra amenazas existentes en el Internet, esto gracias al análisis del tráfico que se genera en la red, debido a que analiza los patrones de tráfico, con esta herramienta es posible descubrir automáticamente la infraestructura del atacante de red y es capaz de bloquear las conexiones a este atacante, sus características principales es que puede detener el phishing y las infecciones de malware, permitiendo evitar la filtración de datos y la identificación inmediata de los dispositivos infectados. Gracias a que Umbrella está ubicado en la nube se puede tener una visualización completa de toda la actividad del internet en todas sus ubicaciones y usuarios que utilizan la red. [57]

En la Figura 1.9 se puede observar la ejecución de Umbrella para poder detectar el dominio y su categorización del estado de dominio, teniendo los estados de -1 como malicioso, 1 como seguro y 0 como un estado no asignado.

Figura 1.9. Estado del dominio mediante Umbrella.

Si se desea, se puede alterar los estados del dominio para poder observarlos en forma de texto siguiendo los estados previamente establecidos.

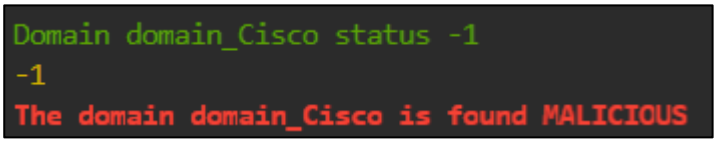


Figura 1.10. Encontrando el estado de dominio de un dominio.

El siguiente paso para seguir con Umbrella es el bloqueo de dominios, en la Figura 1.11 se puede observar cómo se realiza el bloqueo del dominio malicioso.

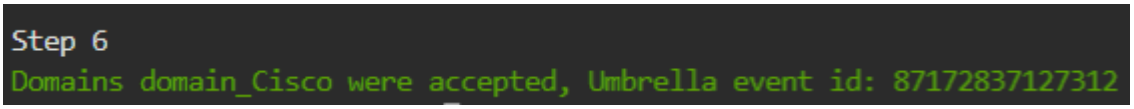


Figura 1.11. Bloquear dominio malicioso mediante Umbrella.

2.2.3.2 Respuesta a amenazas mediante SecureX

El uso de SecureX permitirá implementar soluciones para las respuestas de amenazas mediante herramientas previamente conocidas como Cisco Secure Endpoint (AMP para terminales), Cisco Umbrella, Cisco Secure Malware Analytics (ThreatGrid) y Cisco SecureX. El objetivo de utilizar SecureX es poder implementar las API de SecureX, así como las más utilizadas y sus principales puntos finales, así como utilizar los puntos finales de API para realizar tareas comunes de SOC (Centro de Operaciones de Seguridad). El utilizar SecureX permitirá realizar un análisis inteligente de todos los dispositivos de la red, debido a que se genera una descripción automática de estos para conocer inmediatamente cuales podrán generar problemas de seguridad en la red, a partir de ahí se pueden responder a los ataques de acuerdo a lo preestablecido en los objetivos de la red, ya sea bloquear dominios, bloquear archivos o aislar dispositivos, también se puede mantener el ataque en un entorno aislado para poder continuar la investigación. [58]

A continuación, se detalla lo que se puede realizar con la API de SecureX. En primer lugar, la inspección de información extraída de los dispositivos mediante texto con o sin formato. Así también, la API proporciona una capacidad de otorgar una valoración a la información obtenida. Después, se aplicarían distintas respuestas en correspondencia con la información recopilada, por ejemplo, agregar los dispositivos a una lista de bloqueo. Creación de libros de casos, para poder almacenar cada información recopilada de la API y así tener a la mano cada amenaza que ha aparecido en la red y así tener un conocimiento amplio de las amenazas más comunes en la red. Permite también el uso de base de datos de terceros como de Intel, ajustes de configuración de SecureX de acuerdo con lo que se necesite, así como el uso de OAuth para credenciales y tokens de acceso. [58]

El código de la respuesta de amenazas se encuentra en la sección Anexos en Anexo 3.

En la Figura 2.1 se observa la ejecución de la autenticación a la API SecureX, donde la salida debe ser exitosa para poder utilizar la funcionalidad, si existe un mensaje de Token caducado es necesario reiniciar la petición.

```
developer:securex-threat-response > python threat-response.py

Step 1

==> Authenticating to Cisco Threat Response...
Received Threat Response access token
```

Figura 2.1. Respuesta a la autenticación a API SecureX.

A continuación, se realiza una inspección de elementos observables, los cuales pueden ser encontrados en publicaciones de blogs, tweets, artículos de noticias, syslog, registros DNS, etc. Todos los elementos observables pueden ser generados en formato CTIM para llamadas en la API. En la Figura 2.2 se observa cómo se ha realizado la inspección. [54]

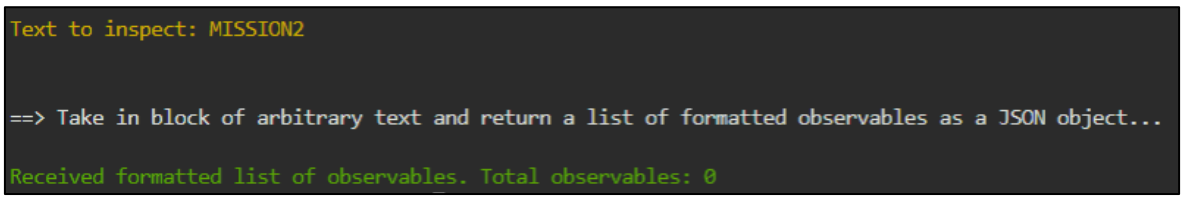


Figura 2.2. Texto por inspeccionar utilizando la respuesta a amenazas.

Adicionalmente, se puede recibir acciones de respuesta las cuales pueden ser aplicables para solventar las amenazas existentes en la red, en la Figura 2.3 se puede observar cómo se han conseguido las listas de acciones para el elemento observable.

Figura 2.3. Lista de acciones de respuesta.

Además, mediante SecureX se puede generar libros de casos como se estableció previamente, al ejecutar el script, se ha generado un libro de casos donde se observa la descripción, versión de esquema, tipo y fuente. En la Figura 2.4 se puede observar la salida mencionada.

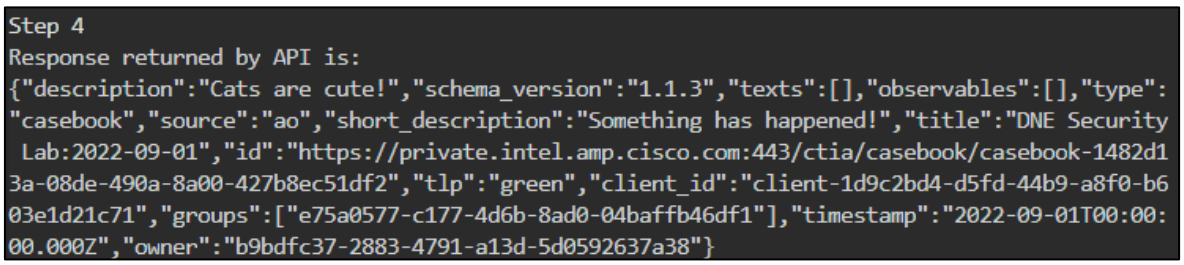


Figura 2.4. Creación de un libro de casos.

Utilizando SecureX también es posible generar incidentes, esto con el fin de probar si SecureX está funcionando correctamente, para ellos se puede generar una función en Python y su salida sería la siguiente como se muestra en la Figura 2.5.

Figura 2.5. Creación de incidente en SecureX.

2.2.3.3 Seguridad de aplicaciones Cisco

El módulo de seguridad de aplicaciones Cisco, consta en cómo se pueden introducir amenazas en cualquier etapa de la red, las aplicaciones vendrían a ser los principales focos de atención para atacantes, debido a que son las que van a proveer servicios a ser consumidos, por lo tanto, la seguridad en aplicaciones es otra etapa importante al momento de querer implementar seguridad en redes IoE. Adicionalmente, se especifica que el no tratar correctamente las aplicaciones con soluciones de seguridad actuales, provocaría grandes pérdidas de dinero a las empresas que utilicen estas aplicaciones, además de pérdida de tiempo por no poder solventar los problemas inmediatamente. [59]

En el módulo de seguridad de aplicaciones de Cisco, se puede generar una carga de trabajo segura mediante el uso de API REST, en este punto es necesario conocer los datos completos de telemetría de tráfico, análisis avanzados con algoritmos y se puede proveer protección integral de carga de trabajo para centro de datos utilizando primordialmente soluciones basadas en Big Data. [59]

En la práctica, el módulo de seguridad de aplicaciones de Cisco inicia con llamadas API mediante Python en el Anexo 4 se puede observar el código para poder probar el funcionamiento de las llamadas de API de trabajo de flujo seguro, en la Figura 3.1 se puede observar la salida de una prueba de la API flujo de trabajo seguro, si funciona correctamente se puede observar la siguiente salida.

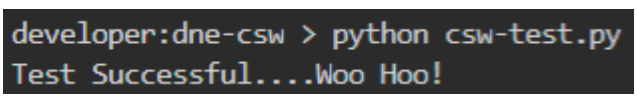


Figura 3.1. Prueba de API de flujo de trabajo seguro.

En el Anexo 5, se puede observar el código para poder crear funciones que consulten todos los sensores de flujo de trabajo seguro. En el Anexo 6, se encuentra el código para la consulta de sensores.

En la Figura 3.2 se puede observar la salida de los sensores que se encuentran activos dentro de la red, donde se especifica el nombre del dispositivo, el tipo, su dirección IP y la plataforma que utiliza.

```
developer:dne-csw > python get_sensors.py
 HOSTNAME                                    TYPE        IP ADDRESS          PLATFORM
kemast-aws-centos1                           ENFORCER    10.7.1.22           CentOS-7.7
kemast-aws-win19-1                           ENFORCER    10.7.1.27           MSServer2019Datacenter
kemast-aws-centos2                           ENFORCER    10.7.1.35           CentOS-7.6
ip-10-6-1-15                                 ENFORCER    10.6.1.15           AmazonLinux-2
ip-10-6-1-119                                ENFORCER    10.6.1.119          AmazonLinux-2
ip-10-6-1-124                                ENFORCER    10.6.1.124          AmazonLinux-2
km-win-test3                                 ENFORCER    192.168.93.196      MSServer2022Datacenter
km-win-file1                                 ENFORCER    192.168.93.198      MSServer2022Datacenter
km-win-test4                                 ENFORCER    192.168.93.195      MSServer2022Datacenter
km-win-test5                                 ENFORCER    192.168.93.194      MSServer2022Datacenter
km-win-test2                                 ENFORCER    192.168.93.197      MSServer2022Datacenter
servicenow-2924f901770d56a6-kemast-edge-edgeapp SNOW_PROXY   unkown                  linux-amd64
```

Figura 3.2. Consulta de sensores.

En el Anexo 7, se puede observar el código para obtener las vulnerabilidades, en la Figura 1.19, se puede observar las vulnerabilidades más comunes que se han registrado en los sensores o dispositivos, mediante el identificador CVE.

```
developer:dne-csw > python get_vulnerabilities.py

HOSTNAME          IP              CVE ID          SCORE(V2)  SCORE(V3)  PACKAGE          VERSION
kemast-aws-win19-1 10.7.1.27      CVE-2020-1112   9          9.9        msserver2019datacenter 1809-17763.1192
kemast-aws-win19-1 10.7.1.27      CVE-2020-1408   9.3        8.8        msserver2019datacenter 1809-17763.1192
kemast-aws-win19-1 10.7.1.27      CVE-2022-26811  9          7.2        msserver2019datacenter 1809-17763.1192
kemast-aws-win19-1 10.7.1.27      CVE-2020-1472   9.3        10         msserver2019datacenter 1809-17763.1192
kemast-aws-win19-1 10.7.1.27      CVE-2022-26824  9          7.2        msserver2019datacenter 1809-17763.1192
kemast-aws-win19-1 10.7.1.27      CVE-2020-1401   9.3        7.8        msserver2019datacenter 1809-17763.1192
kemast-aws-win19-1 10.7.1.27      CVE-2020-1039   9.3        7.8        msserver2019datacenter 1809-17763.1192
kemast-aws-win19-1 10.7.1.27      CVE-2022-21972  9.3        8.1        msserver2019datacenter 1809-17763.1192
kemast-aws-win19-1 10.7.1.27      CVE-2020-0606   9.3        8.8        .NET Framework 4.7.2 4.7.2
kemast-aws-win19-1 10.7.1.27      CVE-2020-1167   9.3        7.8        msserver2019datacenter 1809-17763.1192
kemast-aws-win19-1 10.7.1.27      CVE-2021-33740  9.3        7.8        msserver2019datacenter 1809-17763.1192
kemast-aws-win19-1 10.7.1.27      CVE-2022-21888  9.3        7.8        msserver2019datacenter 1809-17763.1192
kemast-aws-win19-1 10.7.1.27      CVE-2020-1400   9.3        7.8        msserver2019datacenter 1809-17763.1192
kemast-aws-win19-1 10.7.1.27      CVE-2020-1562   9.3        7.8        msserver2019datacenter 1809-17763.1192
```

Figura 3.3. Consulta de vulnerabilidades en los sensores.

2.3 Proceso para la automatización de seguridad en redes IoE

Para empezar a determinar cuál es el proceso de automatización de seguridad de redes, es necesario entender el proceso general de la automatización de redes, esto con el fin de poder realizar un análisis de cómo se puede poner en funcionamiento una red autónoma en cuestiones de seguridad, esto incluyendo todas las herramientas disponibles, soluciones utilizadas en el campo de la programación, en específico Python que sigue creciendo en el campo de las redes y que tiene aplicaciones factibles para poder ser utilizadas en este tipo de problemáticas en conjunto de las nuevas tendencias de tecnología como la inteligencia artificial y sus subcampos como denominados aprendizaje automático y profundo.

Estos procesos, permitirán cumplir tres pilares generales los cuales crean una sólida arquitectura de seguridad para las redes del Internet de las Cosas (IoE), los cuales son: [44]

1. Conciencia de amenazas: Este pilar es fundamental, debido a que promueve la capacidad de identificar un comportamiento normal y anormal de las redes IoE, así como, las soluciones que se deben aplicar rápidamente en las mismas. [44]

2. Visibilidad: La visibilidad es un pilar en el cual se realiza una mirada precisa y en tiempo real a los datos, los dispositivos y las relaciones entre ellos, logrando así una gran capacidad para poder abstraer la información existente entre la masiva cantidad de dispositivos que se pretenden asegurar en las redes IoE. [44]

3. Acción: Este último pilar combina los dos previos, con el fin de poder crear una red completamente predictiva, en el cual se generen datos a partir de los dispositivos IoE y que se pueda determinar cuándo se está comprometiendo la seguridad de la red, y que finalmente se activen scripts para poder solventar los problemas. [44]

2.3.1 Prácticas para la automatización de redes

La automatización de redes generalmente debe acatar una serie de procedimientos para poder ser implementada en el mundo real, con el propósito de llegar a cumplir ciertos parámetros importantes para poder conseguir una red totalmente autónoma, estos procedimientos son: [36]

1. Determinar los objetivos de aplicación: Dependiendo del alcance que se quiera lograr al querer aplicar la automatización de redes, es necesario determinar cuáles objetivos

se quieren lograr dentro de la red, esto permitirá sentar las bases para realizar las primeras estrategias de automatización. [36]

2. Definir los casos de uso: Se trata de poder agrupar la mayor cantidad de casos de uso que se requieran dentro de la red, estos pueden ser, operaciones, mantenimiento de la red, gestión de configuración, orquestación de servicios y gestión de políticas. [36]

3. Dirección de dominios de red: Se debe determinar el entorno en donde se desea aplicar la automatización de redes, no es lo mismo querer realizar los procesos de automatización en una red interna, que quererlos aplicar dentro de los servicios de nube, por lo que, determinar una combinación entre el dominio y un caso de uso serán un punto de partida ideal para la planificación de proyectos de automatización. [36]
4. Determinar fuentes de datos confiables: Este punto es importante principalmente por lo valioso que representa la captura de datos de los elementos de red para poder ser utilizados al momento de querer realizar la automatización de redes. [36]

5. Transición a un modelo operativo: En esta fase se incluye un repositorio de gestión de código fuente, en general para poder tener almacenado scripts de configuración general para dispositivos. [37]

6. Mecanismo de retroalimentación de la red: Es necesario saber cuál es el rendimiento de la red, haciendo uso de la telemetría o de la monitorización basada en el protocolo SNMP (Protocolo Simple de Administración de Red), los cuales se deben comunicar con bases de datos que permitan almacenar los datos monitorizados, para finalmente tener activadores de acción, los cuales en general funcionan para desencadenar flujos de trabajo automatizados para corregir posibles errores dentro de la red. [37]

7. La última fase consta de pruebas que se realizan previas a la aplicación de la automatización en producción, en este caso se deben lograr cumplir con los objetivos predefinidos previamente, si en esta fase los resultados son los esperados, entonces su aplicabilidad para poner en producción es recomendable. [37]

2.3.2 Prácticas para automatizar la seguridad en redes IoE

Las prácticas de automatización de redes en general son similares a las prácticas de automatización de seguridad de redes. Sin embargo, en el ambiente de las seguridades y en especial en las redes IoE, se incrementan ciertas prácticas que deben ser consideradas para poder implementar la automatización de seguridades de redes.

Adicionalmente, se debe considerar utilizar la metodología OCTAVE la cual es la clave para poder realizar un análisis y gestión de riesgos en las redes IoE, esto con el fin de poder percibir y clasificar el riesgo dentro de la red, desde el más crítico hasta el más leve. Para poder desarrollar la metodología OCTAVE, se siguen las siguientes fases: [43]

Fase 1:

En esta fase, se engloban los activos, amenazas, vulnerabilidades de la red, las exigencias de seguridad y las normas existentes. En esta fase, se seleccionan 5 activos críticos en los cuales se definen requisitos y amenazas, la información recopilada está en base a todo el personal que conforme y use a la red. [43]

Fase 2:

En esta fase se engloban los componentes claves y las vulnerabilidades técnicas, es decir, se observarán los componentes que, en el caso de presentar vulnerabilidades, podrían causar un estado crítico en la red en el que la seguridad de esta se vea comprometida. [43]

Fase 3:

En la última fase de la metodología se realiza el análisis de riesgos, es decir, identificar aquellos riesgos que podrían suceder si es que se atacan los activos críticos previamente detallados, así mismo, en esta fase, se pretende identificar las estrategias de protección, las cuales deben tener un exhaustivo estudio y deben ser aprobados antes de su ejecución. [43]

Adicionalmente, en la Figura 4.1, se presenta el proceso de gestión de riesgos para redes, en los cuales se basa OCTAVE. [45]

Figura 4.1. Proceso de Gestión de Riesgos.

Nota: En la Figura 4.1, se puede observar el proceso de gestión de riesgos el cual básicamente es un bucle en el cual siempre se desea tener mejoras con respecto al análisis y propuestas de seguridad dentro de las redes. [45]

Una vez visto el procedimiento para poder seguir la metodología OCTAVE en el análisis y gestión de riesgos, se detallan las fases necesarias para poder implementar la automatización de seguridad de redes IoE, así como las prácticas más comunes.

1. Definir los objetivos y priorizar eventos: Similar a la automatización de redes, se deben definir objetivos que deben ser cumplidos al momento de poner en producción a las soluciones de automatización de seguridades, además, se deben establecer cuáles son las prioridades del entorno, es decir, se debe ver cuáles son los elementos más importantes de la red, cuáles son los que más problemas potencialmente presentarán en la red. [38]

2. Reconocimiento de los dispositivos IoE conectados a la red: Debido a que cada dispositivo conectado a la red puede representar como un punto de entrada para cualquier tipo de ataque, es necesario reconocer cuales son los dispositivos existentes, además tener el conocimiento de los posibles dispositivos IoE que se integraran en algún futuro a la red, además, es importante el hecho de saber cuáles son aquellos dispositivos que permitan la integración de software de monitorización, así como la posibilidad de ejecutar scripts. [39]

3. Evaluación de riesgos: Una vez revisados todos los dispositivos conectados en la red y los posibles dispositivos a conectar, así como definir si los dispositivos que pertenecen a la red IoE tienen las capacidades ideales para monitorización de redes y ejecución de scripts. Se deben evaluar los riesgos que se tienen en la red, priorizando la automatización, es decir, al momento de evaluar la red y sus riesgos, se debe tomar nota de cuáles son los eventos que suceden con mayor frecuencia, así como los que toman mucho más tiempo en ser investigados y resueltos. [40]

4. Realizar pruebas de penetración: En esta fase, las pruebas de penetración servirán para evaluar y explotar varios componentes en un dispositivo IoE con el fin de poder hacerlo más seguro. En este punto, a cada dispositivo IoE, se le realizará un mapeo superficial, es decir, recopilar la mayor cantidad de información posible, entre ellos, se encuentra el tipo de arquitectura del dispositivo, protocolos de comunicaciones utilizados, detalles

de aplicación si es móvil, actualizaciones, tanto de software como de firmware, puertos de hardware disponibles. [41]

5. Generación de datos pertinentes para la seguridad de redes IoE: El uso de herramientas de monitorización de seguridad de redes, son aquellas que permitirán obtener los datos pertinentes al momento de querer implementar soluciones automáticas a problemas de seguridad que se encuentren dentro de las redes IoE, como se presentó previamente, hasta esta fase, ya se tendría el conocimiento de aquellas vulnerabilidades existentes en la red, así como los problemas de seguridad más comunes que se tendrían, por lo tanto, en la recopilación de datos se extraerán información que se adapte con las necesidades que se tienen dentro de la red, afortunadamente la mayoría de las herramientas descritas en este documento funcionan dentro de equipos Linux y Windows, sin embargo, existen otros que no soportan una integración con herramientas, por lo que su monitoreo se basará en la disponibilidad del equipo así como el análisis de tráfico para estos dispositivos. [39]

6. Análisis de los datos recopilados: En la fase del análisis de recopilación de datos, se pretende utilizar los datos recolectados en conjunto con un lenguaje de programación, en este caso, el mejor lenguaje actualmente es el de Python, debido a su facilidad de uso, y la integración con una gran cantidad de API (Interfaz de Programación de Aplicaciones), muchos de estos API funcionan perfectamente para ser utilizados en el tratamiento de datos, librerías como Numpy y Pandas funcionan para poder realizar tratamiento de datos. A continuación, en la Figura 4.2, se presenta el crecimiento de interés por parte de programadores en el uso de Python. [42]

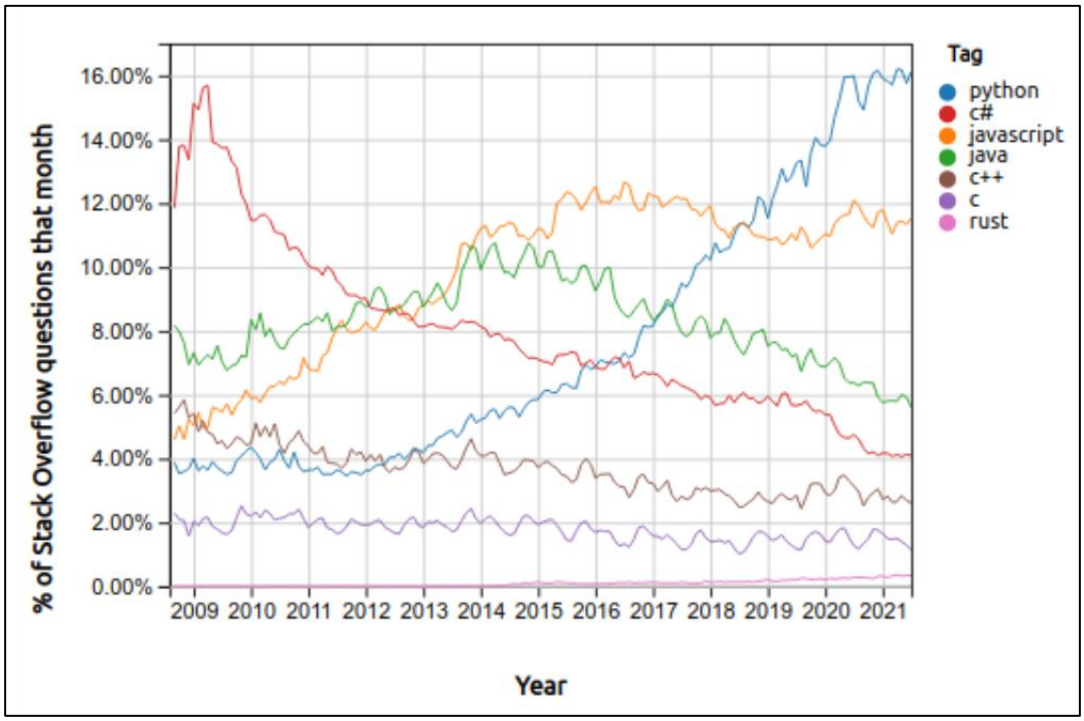


Figura 4.2 Lenguajes de programación que más preguntas generan en Stack Overflow.

Nota: En la Figura 4.2, se puede observar cómo Python es la herramienta ideal para el análisis de datos, especialmente por las distintas librerías que se pueden utilizar, tales como: [42]

- Boto3: El cual es un kit de desarrollo de software (SDK) de Amazon Web Services (AWS) para Python, con esta librería se puede iniciar y detener servidores, así como, poder realizar actualizaciones y administrar parches, etc.

- Regex: Esta librería en general puede parecer sencilla y no muy útil en el ámbito de las seguridades, sin embargo, su aplicación puede ser muy favorable al momento de querer encontrar patrones específicos en bloques de texto.

- Pyautogui: Esta herramienta permite controlar mouse y teclado mediante scripts, por lo tanto, es una herramienta que permite la gestión de dispositivos remotos de manera automática sin necesidad de intervenir.

- Pyclip: Es un módulo similar al anterior (Pyautogui), sin embargo, con este se puede acceder mediante scripts al portapapeles de un dispositivo, de manera más sencilla que utilizando el otro modulo.

- Faker: Con esta biblioteca se pueden producir datos falsos los cuales se pueden utilizar para poder probar los distintos scripts que se han desarrollado y verificar que su funcionamiento sea completamente correcto.

7. Integración de todas las herramientas disponibles junto con la inteligencia artificial: Esta etapa, permite integrar los datos analizados para poder formar parte de la fuente de datos de la inteligencia artificial, aprendizaje automático y aprendizaje profundo, esto con el fin de poder predecir cuando la red pueda estar comprometida, de acuerdo con las necesidades de esta y sus políticas, para poder integrar inteligencia artificial, se puede utilizar la misma herramienta de Python, la cual después de tratar los datos, tendrá la posibilidad de predecir cuándo existirá un ataque en la red, así mismo, se podría asegurar su factibilidad de predicción de acuerdo con su exactitud y precisión, así en conjunto con otros scripts, después de predecir se pueden activar de manera automática para poder solucionar los problemas de la red. [33]

8. Pruebas de funcionamiento: Las pruebas de funcionamiento se las realizan en un entorno de prueba, es decir, no en un entorno de producción, para así poder obtener conclusiones del funcionamiento de la red, y que tanto se ha podido acoplar a los objetivos de la automatización de seguridad de redes IoE, verificando el funcionamiento de las herramientas de recolección de datos, el tratamiento y análisis de estos, el uso de inteligencia artificial para poder predecir cuándo va a ocurrir un evento que ponga en peligro a la red, y scripts que automáticamente se activan para poder solventar los mismos. [37]

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

La automatización de seguridades de redes tradicionales tiene una gran cantidad de herramientas, aplicaciones y métodos para ser llevadas a cabo, todas ellas aplicables al ámbito de las redes IoE. Previamente, se había descrito que, en estas redes, lo más importante es asegurar los datos, debido a la gran cantidad de dispositivos que la conforman y que pueden realizar procesos que en ciertos casos son delicados y que bajo un ataque de seguridad en los que se comprometan estos datos podría llegar a ser muy peligroso para el usuario de la red.

Las herramientas encontradas permitirán lograr un cierto grado de seguridad a las redes IoE, ya que, a pesar de que se tiene el conocimiento de que no se puede tener una red completamente segura, con estas herramientas, es posible mantener un estándar de seguridad alto. Además, el uso de tecnologías de vanguardia como la inteligencia artificial y lenguajes de programación como Python, permite que la automatización en procesos de seguridad para las redes sea una realidad. Utilizando la inteligencia artificial, es posible desarrollar modelos, que en un principio puedan clasificar los distintos tipos de ataques que puedan ocurrir hacia una red, esto gracias al aprendizaje automático y a los datos que se obtengan gracias al lenguaje de programación Python. Estos modelos, se deben desarrollar tal que, puedan cumplir con los estándares de seguridad necesarios, es decir tener un alto porcentaje de confiabilidad al momento de realizar la clasificación y así poder depender del mismo al momento de ser puesto en operación dentro de una red real, por lo tanto dentro de cualquier entorno en el que se desee implementar soluciones de seguridades de manera automática, es necesario definir cuál sería el porcentaje aceptable para poder depender completamente del modelo, y por lo tanto evitar la manipulación humana casi por completo. En la automatización de detección de amenazas y ataques, también se debe acompañar con mecanismos que permitan automatizar la respuesta que se deba tener al momento de identificar los posibles ataques, estos se pueden realizar mediante eventos, en donde al momento de ocurrir uno, se pueda ejecutar automáticamente algún script de programación que mitigue el ataque complementando entre sí la detección y la respuesta, los cuales son pilares esenciales en la seguridad de redes y que gracias a todas las herramientas identificadas pueden ser solventadas.

Así mismo, se puede observar las ventajas que se tienen al momento de utilizar soluciones de Cisco para poder implementar automatización de seguridad de redes, partiendo de la base de detección, respuesta, recuperación, identificación y protección, contando con distintas herramientas posibles que permiten lograr todo eso, lo mejor es que combina las soluciones de Cisco en conjunto con Python, para poder realizar el reconocimiento de vulnerabilidades, la toma de decisiones automáticas al momento de responder a las amenazas, así como la protección contra las aplicaciones que pueden de igual manera generar brechas de seguridad en la red.

3.2 Conclusiones

En el presente trabajo, se realizó un trabajo de investigación acerca de la importancia de la seguridad en las redes del Internet de Todo (IoE), la razón por las cuales estas soluciones deben ser automatizadas y finalmente, las herramientas que permiten lograr esto, siguiendo

de acuerdo con lo establecido con todos los objetivos específicos planteados. Por lo tanto, después de haber realizado este trabajo, se ha concluido lo siguiente:

- La automatización de seguridades como tal no es un concepto nuevo en el mundo de la tecnología, sin embargo, el ser aplicable a una red nueva como la del Internet de Todo, puede representar un reto, sin embargo, gracias a la cantidad de herramientas que existen actualmente para redes tradicionales, es posible aplicarlas para este tipo de redes, garantizando una mayor seguridad, y también gracias a las nuevas tecnologías de inteligencia artificial y su uso con lenguajes de programación como Python.

- La automatización de redes como paso previo a la automatización de seguridades es muy importante para poder entender el funcionamiento de los procesos y prácticas que se requieren para poder llevar a cabo la automatización, y así lograr aplicarlos a la problemática específica de la automatización de las seguridades de redes.

- El poder aplicar inteligencia artificial para poder realizar modelos depende en gran medida de la calidad de los datos de red que se vayan recopilando, por lo tanto, la etapa de descubrimiento de herramientas es muy importante, así se logrará la recopilación de información necesaria que realmente sea de calidad para ser utilizada en los modelos de aprendizaje automático, y por ende obtener clasificadores, ya sea de malware, de intrusos, etc., que realmente sean confiables.

- Las redes IoE y la automatización de seguridades en estas redes deben mantener un estándar de calidad al momento de querer ser implementadas, además, se debe satisfacer los tres pilares fundamentales de seguridad para que se mantengan buenas prestaciones y se cumplan con los objetivos establecidos para la red.

- Es necesario entender que para redes del Internet de Todo, no se pueden aplicar soluciones de automatización de seguridades de redes para todos los equipos, principalmente por la gran desventaja que se presenta en este tipo de redes, el cual es que no todos los dispositivos cuentan con las mismas características, es decir, existen equipos que poseen mayor cantidad de niveles de procesamiento, lo que ya marca una limitante, debido a que serán estos equipos los que podrán utilizar las soluciones de automatización de seguridades avanzadas, mientras que los equipos de bajas capacidades de procesamiento no poseen estas facilidades.

- Es posible aplicar soluciones a equipos Cisco en el momento de querer implementar automatización de seguridades en redes IoE, Cisco provee exactamente soluciones que funcionan para sus dispositivos, estas soluciones integradas en DevNet Express Security permiten lograr cumplir los objetivos principales de las seguridades de redes y de la automatización, teniendo en cuenta el análisis inteligente de los dispositivos conectados a la red, las vulnerabilidades comunes que se pueden buscar automáticamente a cada uno de los elementos de la red, así como disparadores que se pueden implementar utilizando estas soluciones al momento de ocurrir un evento y que puede ejecutar respuestas a las amenazas automáticamente, logrando así soluciones robustas de seguridad para los elementos de red.

3.3 Recomendaciones

- Al momento de querer realizar la monitorización de seguridades de los dispositivos IoE, es necesario conocer cuáles son aquellos dispositivos que puedan integrarse con software de monitorización, así como con la ejecución de scripts, debido a que no todos los dispositivos IoE pueden realizar este tipo de acciones, y por lo tanto se deben ver otros tipos de soluciones inmediatas que se pueden tomar para poder lograr una completa automatización de la seguridad en este tipo de redes.

- Es recomendable definir correctamente cuáles serán los objetivos primordiales que se quieren conseguir con la automatización de las seguridades de redes, esto debido a que servirán como base para poder determinar si el procedimiento para conseguir la automatización es satisfactorio, así mismo, determinar cuáles son los elementos de red que pueden ser críticos si es que llegan a ser vulnerados, logrando así una correcta clasificación de los elementos entre los más críticos y los más leves, y así poder tomas las acciones inmediatas de manera simple y organizada.

- Se recomienda determinar cuáles son los datos más importantes que se deben recopilar dentro de las redes IoE, debido a que estos datos se relacionan directamente con las soluciones que se pretenden dar al momento de que existan problemas con la seguridad de la red, si estos datos no tienen importancia o no se relacionan por completo con los objetivos de la red, no servirán de mucho, por lo tanto, cualquier tipo de aplicación de inteligencia artificial no podrá predecir ningún tipo de problema o lo hará de manera errónea, logrando así una nula posibilidad de aplicar este tipo de tecnología.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] A. Chinchawade y P. O. s. Singh Lamba, "A review on authentication and security procedures for IoE systems.", *ResearchGate*, 2019. Accedido el 2 de julio de 2022. [En línea].
Disponible: https://www.researchgate.net/publication/341540095_A_REVIEW_ON_AUTHENTICATION_AND_SECURITY_PROCEDURES_FOR_IOE_SYSTEMS

[2] S. M. Mohammad y L. Surya, "Security automation in information technology", *ResearchGate*, vol. 6, n.º 2, 2018. Accedido el 8 de junio de 2022. [En línea].
Disponible: https://www.researchgate.net/publication/342977281_Security_Automation_in_Information_Technology

[3] Rachit, S. Bhatt y P. Rao Ragiri, "Security trends in Internet of Things: A survey", *ResearchGate*, 2021. Accedido el 8 de julio de 2022. [En línea]. Disponible: https://www.researchgate.net/publication/348429338_Security_trends_in_Internet_of_Things_a_survey

[4] "¿Qué es el Internet de las cosas (IoT)?" Oracle | Cloud Applications and Cloud Platform. https://www.oracle.com/ar/internet-of-things/what-is-iot/ (accedido el 3 de agosto de 2022).

[5] "¿Qué es el Internet de las cosas?" Red Hat - We make open-source technologies for the enterprise.
https://www.redhat.com/es/topics/internet-of-things/what-is-iot (accedido el 3 de agosto de 2022).

[6] "The internet of everything". Cisco - Networking, Cloud, and Cybersecurity Solutions. https://www.cisco.com/c/dam/en_us/about/business-insights/docs/ioe-value-at-stake-public-sector-analysis-faq.pdf (accedido el 3 de agosto de 2022).

[7] S. Jena. "Difference between IoE and IoT". Geeks for Geeks. https://www.geeksforgeeks.org/difference-between-ioe-and-iot/ (accedido el 3 de agosto de 2022).

[8] William Stallings, NETWORK SECURITY ESSENTIALS, Applications and Standards, 4th Edition, Prentice Hall, 2010

[9] N. R. Sánchez. "Vulnerabilidades en redes".

Techclub Tajamar. https://techclub.tajamar.es/vulnerabilidades-en-redes/ (accedido el 3 de agosto de 2022).

[10] Estefanía Domínguez de la Iglesia. "Tipos de vulnerabilidades en ciberseguridad". Campus Internacional de Ciberseguridad. https://www.campusciberseguridad.com/blog/item/118-tipos-de-vulnerabilidades-en-ciberseguridad. (accedido el 3 de agosto de 2022).

[11] "¿Qué es la automatización?" Red Hat - We make open-source technologies for the enterprise. https://www.redhat.com/es/topics/automation/whats-it-automation. (accedido el 3 de agosto de 2022).

[12] "What is IT automation? | vmware glossary". VMware. https://www.vmware.com/es/topics/glossary/content/it-automation.html (accedido el 3 de agosto de 2022).

[13] "¿Qué es la automatización de la red?" Cisco. https://www.cisco.com/c/es_mx/solutions/automation/network-automation.html#~recursos (accedido el 3 de agosto de 2022).

[14] "¿Qué es la automatización de red?" Juniper. https://www.juniper.net/mx/es/research-topics/what-is-network-automation.html (accedido el 3 de agosto de 2022).

[15] "¿Qué es el monitoreo de redes? | motadata". Motadata. https://www.motadata.com/es/what-is-network-monitoring/ (accedido el 3 de agosto de 2022).

[16] "Cómo usar el monitoreo de redes como un sistema de advertencia temprana para problemas de seguridad". Progress WhatsUp Gold. https://www.whatsupgold.com/es/blog/como-usar-el-monitoreo-de-redes-como-un-sistema-de-advertencia-temprana-para-problemas-de-seguridad (accedido el 3 de agosto de 2022).

[17] N. Carklin. "Network security monitoring: A complete guide". Parallels Remote Application Server Blog - Application virtualization, mobility and VDI. https://www.parallels.com/blogs/ras/network-security-monitoring/ (accedido el 3 de agosto de 2022).

[18] E. Burns, N. Laskowski y L. Tucci. "What is artificial intelligence (AI)? Definition, benefits and use cases".

SearchEnterpriseAI. https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence (accedido el 3 de agosto de 2022).

[19] "What is artificial intelligence? How does AI work? | built in". builtin. https://builtin.com/artificial-intelligence (accedido el 3 de agosto de 2022).

[20] "What is machine learning? What are machine learning algorithms? What are popular machine learning examples? | built in". bultin. https://builtin.com/machine-learning (accedido el 3 de agosto de 2022).

[21] M. Thomas. "Deep Learning: An In-Depth Look Into the AI-Based Tech Deep learning is helping industries make great strides, but is it revolutionary or just a 'useful tool' bound for extinction?" builtin. https://builtin.com/artificial-intelligence/deep-learning (accedido el 3 de agosto de 2022).

[22] Y. Zheng, A. Pal, S. Abuadbba, S. R. Pokhrel, S. Nepal and H. Janicke, "Towards IoT Security Automation and Orchestration," 2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), 2020, pp. 55-63, doi: 10.1109/TPS-ISA50397.2020.00018. Accedido el 20 de julio de 2022. [En línea]. Disponible: https://ieeexplore.ieee.org/document/9325393

[23] A. J. Chinchawade y O. S. Lamba, "Security and privacy challenges in internet of everything (ioe) with security requirements", Journal of Engineering & Technology - Suresh Gyan Vihar University, vol. 7, n.º 2, 2021. Accedido el 3 de agosto de 2022. [En línea]. Disponible: https://www.gyanvihar.org/journals/index.php/2021/07/26/security-and-privacy-challenges-in-internet-of-everything-ioe-with-security-requirements-2/

[24] B. Schneier, "Managed security monitoring:Network security for the 21st century", Computers & Security, vol. 20, n.º 6, pp. 491–503, septiembre de 2001. Accedido el 8 de agosto de 2022. [En línea]. Disponible: https://doi.org/10.1016/s0167-4048(01)00607-1

[25] B. Dobran. "35 network security tools you should be using, according to the experts". Phoenix Nap Global IT Services. https://phoenixnap.com/blog/best-network-security-tools (accedido el 8 de agosto de 2022).

[26] B. Jovanovic. "What is encryption software? | dataprot". Dataprot. https://dataprot.net/articles/what-is-encryption-software/ (accedido el 8 de agosto de 2022).

[27] "Vulnerability assessment". Imperva. https://www.imperva.com/learn/application-security/vulnerability-assessment/ (accedido el 8 de agosto de 2022).

[28] M. Bagget. "Automating Information Security with Python | SANS SEC573". Cyber Security Training | SANS Courses, Certifications & Research. https://www.sans.org/cyber-security-courses/automating-information-security-with-python/ (accedido el 8 de agosto de 2022).

[29] M. Goss. "How Python addresses security, network traffic analysis concerns". SearchNetworking. https://www.techtarget.com/searchnetworking/feature/How-Python-addresses-security-network-traffic-analysis-concerns (accedido el 8 de agosto de 2022).

[30] L. Raymond. "Intrusion detection". GitHub. https://github.com/LRA-QC/intrusion-detection (accedido el 8 de agosto de 2022).

[31] P. Saxena. "5 hidden python libraries for cyber security". Towards Data Science. https://towardsdatascience.com/5-hidden-python-libraries-for-cyber-security-e83928777a95 (accedido el 8 de agosto de 2022).

[32] "Using yara for malware detection". National Cybersecurity and Communications Integration Center. https://www.cisa.gov/uscert/sites/default/files/FactSheets/NCCIC%20ICS_FactSheet_YARA_S508C.pdf (accedido el 8 de agosto de 2022).

[33] "Artificial intelligence and machine learning in cybersecurity - python for cybersecurity". Python for Cybersecurity. https://pythonforcybersecurity.com/artificial-intelligence-and-machine-learning-in-the-world-of-cybersecurity/ (accedido el 8 de agosto de 2022).

[34] "Intrusion detection system using machine learning algorithms". Geeks for Geeks. https://www.geeksforgeeks.org/intrusion-detection-system-using-machine-learning-algorithms/ (accedido el 8 de agosto de 2022).

[35] F. Kamoun, F. Iqbal, M. A. Esseghir and T. Baker, "AI and machine learning: A mixed blessing for cybersecurity," 2020 International Symposium on Networks, Computers and Communications (ISNCC), 2020, pp. 1-7. doi: 10.1109/ISNCC49221.2020.9297323.

[36] Munaligal, K. (s. f.). 6 practical steps for developing a network automation strategy. Itential. https://www.itential.com/blog/company/automation-strategy/6-practical-steps-for-developing-a-network-automation-strategy/

[37] Slattery, T. (2020, 30 de septiembre). How to build a network automation architecture in 5 phases. SearchNetworking. https://www.techtarget.com/searchnetworking/tip/How-to-build-a-network-automation-architecture-in-5-phases

[38] Solomon, M. (s. f.). Five steps to security automation | securityweek.com. Cybersecurity News, Insights and Analysis | SecurityWeek. https://www.securityweek.com/five-steps-security-automation

[39] Rivas, G. (2018, 20 de julio). El Internet de las Cosas: 5 medidas para garantizar la seguridad del IoT. GB - Advisors. https://www.gb-advisors.com/es/iot-seguridad-el-internet-de-las-cosas/

[40] Security automation: Tools, process and best practices. (s. f.). Cynet. https://www.cynet.com/incident-response/security-automation-tools-process-and-best-practices/

[41] Gohel, D. (2020, 9 de enero). IoT pentesting. Medium. https://medium.com/@xesey/iot-pen-testing-1c4849327499

[42] Python y la Ciberseguridad ¿Cómo sacarle provecho? - TrustDimension | Seguridad en la que puedes Confiar. (s. f.). TrustDimension | Seguridad en la que puedes Confiar. https://www.trustdimension.com/python-y-la-ciberseguridad-como-sacarle-provecho

[43] [43] Metodología OCTAVE para el análisis de riesgos en SGSI. (s. f.). PMG SSI - ISO 27001. https://www.pmg-ssi.com/2021/09/metodologia-octave-para-el-analisis-de-riesgos-en-sgsi/

[44] Using the network to secure the network: An ioe strategy | data center POST. (s. f.). Data Center POST | Data Center POST News. https://datacenterpost.com/using-the-network-to-secure-the-network-an-ioe-strategy/

[45] Hurtado, M. (s. f.). GESTIÓN DE RIESGO METODOLOGÍAS OCTAVE y MAGERIT. Polux Unipiloto. http://polux.unipiloto.edu.co:8080/00004420.pdf

[46] "10 security problems of the IOT | Find out what you need". Chakray. https://www.chakray.com/10-security-problems-internet-of-things/ (accedido el 30 de agosto de 2022).

[47] "Security automation: Tools, process and best practices". Cynet. https://www.cynet.com/incident-response/security-automation-tools-process-and-best-practices/ (accedido el 30 de agosto de 2022).

[48] "What is network automation? | fortinet". Fortinet.

https://www.fortinet.com/resources/cyberglossary/network-automation (accedido el 30 de agosto de 2022).

[49] Top 50 IoT security tools. (s. f.). Startup Stash. https://startupstash.com/iot-security-tools/

[50] Top 10 IoT vulnerabilities. (s. f.). Network World. https://www.networkworld.com/article/3332032/top-10-iot-vulnerabilities.html

[51] The Top 10 IoT Security Threats and Vulnerabilities – Particle Blog. (s. f.). Particle Blog. https://blog.particle.io/the-top-10-iot-security-threats/

[52] DevNet Express - EMEAR - VIRTUAL - Security. (s. f.). Cisco DevNetExpress. https://developer.cisco.com/events/DevNet-Express-EMEAR-VIRTUAL-Security-09-2020/

[53] DevNet, C. (2022, 27 de enero). DevNet Express for Security. Cisco DevNet Learning Labs Center. https://developer.cisco.com/learning/tracks/devnet-express-security/

[54] DevNet, C. (2022b, 27 de enero). Threat Hunting - Introduction. Cisco DevNet Learning Labs Center. https://developer.cisco.com/learning/tracks/devnet-express-security/security-threat-hunting/introduction/

[55] DevNet, C. (2022b, 27 de enero). Introduction to Cisco Secure Endpoint (AMP). Cisco DevNet Learning Labs Center. https://developer.cisco.com/learning/tracks/devnet-express-security/security-threat-hunting/introduction-to-cisco-secure-endpoint-amp/

[56] DevNet, C. (2022c, 27 de enero). Introduction to Secure Malware Analytics (Threat Grid). Cisco DevNet Learning Labs Center. https://developer.cisco.com/learning/tracks/devnet-express-security/security-threat-hunting/introduction-to-secure-malware-analytics-threat-grid/

[57] DevNet, C. (2022d, 27 de enero). Introduction to Umbrella. Cisco DevNet Learning Labs Center. https://developer.cisco.com/learning/tracks/devnet-express-security/security-threat-hunting/introduction-to-umbrella/

[58] DevNet, C. (2022d, 27 de enero). Introduction to Threat Response. Cisco DevNet Learning Labs Center. https://developer.cisco.com/learning/tracks/devnet-express-security/security-securex-threat-response/securex-threat-response-1-enrichment/introduction-to-threat-response/

[59] DevNet, C. (2022b, 27 de enero). Introduction - Cisco Application Security. Cisco DevNet Learning Labs Center. https://developer.cisco.com/learning/tracks/devnet-express-security/cisco-app-security/appsec-0-intro/introduction/

5 ANEXOS

ANEXO I. Código de cazador de amenazas (threat-hunting.py)

ANEXO II. Código de grid de amenazas (threatgrid.py)

ANEXO III. Código de respuesta de amenazas (threat_response.py)

ANEXO IV. Código para la conectividad para una API de flujo de trabajo seguro (csw-test.py)

ANEXO V. Código para implementación de funciones (funciones.py)

ANEXO VI. Código para obtener los sensores de la red (get_sensors.py)

ANEXO VII. Código para obtener las vulnerabilidades de los sensores de la red (get_vulnerabilities.py)

ANEXO I

Código de cazador de amenazas (threat-hunting.py)

```
# Importación de los modulos de python

from datetime import datetime,date
import json
import sys
from pathlib import Path
import requests
from crayons import blue, green, yellow, white, red
from requests.packages.urllib3.exceptions import InsecureRequestWarning
from threatresponse import ThreatResponse
import webexteamssdk

# Configuración de los archivos
# Añade /home a la ruta para importar el modulo de entorno
sys.path.insert(0, '/home')

# Importar el modulo API key
import environment as env

# Configuración de claves API
# Establece claves API como variables constantes:
# Cisco Umbrella
umbrella_en_api_key=env.UMBRELLA_ENFORCEMENT_KEY
umbrella_en_host = env.UMBRELLA_ENFORCEMENT_URL

# Cisco Secure Endpoint
amp_client_id = env.SECURE_ENDPOINT_CLIENT_ID
amp_api_key = env.SECURE_ENDPOINT_API_KEY
amp_host = env.SECURE_ENDPOINT_URL

# Cisco Secure Malware Analytics
tg_api_key=env.SECURE_MALWARE_ANALYTICS_API_KEY
tg_host=env.SECURE_MALWARE_ANALYTICS_URL

# Cisco SecureX
secx_client_id=env.SECUREX_CLIENT_ID
secx_api_key=env.SECUREX_API_KEY
secx_host=env.SECUREX_URL

# Webex
# webex_access_token = < INSERT WEBEX ACCESS TOKEN HERE>
webex_room_id = env.WEBEX_TEAMS_ROOM_ID
```

```python
# Deshabilitar las advertencias sobre solicitudes inseguras
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

# Bloquear hash de archivo usando una lista de detección personalizada simple
def get_amp_scds(query_params="",
    host=amp_host,
    client_id=amp_client_id,
    api_key=amp_api_key,
):
    """Get a list of Simple Customer Detection lists from Cisco Secure Endpoint."""
    print("\n==> Getting SCDs from Cisco Secure Endpoint")
    url = f"https://{client_id}:{api_key}@{host}/v1/file_lists/simple_custom_detections"

    response = requests.get(url, params=query_params)
    # Consider any status other than 2xx an error
    response.raise_for_status()

    scds_list = response.json()["data"]

    return scds_list

# Aislamiento del dispositivo
def amp_isolation(method, computer_guid,
    host=amp_host,
    client_id=amp_client_id,
    api_key=amp_api_key,
):
    print(f"\n==> Performing {method} isolation in Cisco Secure Endpoint")

    url = f"https://{client_id}:{api_key}@{host}/v1/computers/{computer_guid}/isolation"

    if method == 'get':
        response = requests.get(url)
        response.raise_for_status()
    elif method == 'put':
        response = requests.put(url)
        if response.status_code == 409:
            print(red("ATTENTION: The computer is already isolated."))
        else:
            response.raise_for_status()
    elif method == 'delete':


        response = requests.delete(url)
        if response.status_code == 409:
            print(red("ATTENTION: Isolation has already been stopped."))
        else:
        response.raise_for_status()
    else:
        print(red("ERROR: Unrecognized REST API Method. Please use 'get', 'put' or 'delete'."))
        sys.exit(1)

    isolation_status = response.json()["data"]

    return isolation_status

# Bloqueo de dominios con Umbrella
def post_umbrella_events(blocklist_domains,
    host=umbrella_en_host,
    api_key=umbrella_en_api_key,
):
    print(white(f"\n==> Post malware events to the Umbrella Enforcement API for processing and optionally adding to a customer's domain lists."))
    # Construcción de la API endpoint para publicar los eventos en Umbrella Enforcement API
    url = f"https://{host}/1.0/events?customerKey={api_key}"

    headers={'Content-type': 'application/json', 'Accept': 'application/json'}
```

```
    # Tiempo para AlertTime y EventTime cuando los dominios son agregados a Umbrella
    time = datetime.now().isoformat()
    data = []

    for domain in blocklist_domains:
        obj = {
            "alertTime": time + "Z",
            "deviceId": "ba6a59f4-e692-4724-ba36-c28132c761de",
            "deviceVersion": "13.7a",
            "dstDomain": domain,
            "dstUrl": "http://" + domain + "/",
            "eventTime": time + "Z",
            "protocolVersion": "1.0a",
            "providerName": "Security Platform"
        }
        data.append(obj)

    response = requests.post(url, data=json.dumps(data), headers=headers)
    response.raise_for_status()

    id = response.json()["id"]

    return id, data

# Búsqueda de envíos
def threatgrid_search_submissions(
    sha256,
    host=tg_host,
    api_key=tg_api_key,
):
    """Search TreatGrid Submissions, by sha256.
    Args:
        sha256(str): Lookup this hash in ThreatGrid Submissions.
        host(str): The ThreatGrid host.
        api_key(str): Your ThreatGrid API key.
    """
    print(white(f"\n==> Searching the ThreatGrid Submissions for sha256: {sha256}"))

    query_parameters = {
        "q": sha256,
        "api_key": api_key,
        "after": "2019-12-01T05:00:00.000Z"
    }

    response = requests.get(
        f"https://{host}/api/v2/search/submissions",
        params=query_parameters,
    )
    # error checking
    response.raise_for_status()

    submission_info = response.json()["data"]["items"]

    if submission_info:
        print(green("Successfully retrieved data on the sha256 submission"))
    else:
        print(red("Unable to retrieve data on the sha256 submission"))
        sys.exit(1)

    return submission_info
```

```python
# Función para recuperar eventos de Cisco Secure Endpoint
def get_amp_events(query_params="",
    host=amp_host,
    client_id=amp_client_id,
    api_key=amp_api_key,
):
    """Get a list of recent events from  Cisco Secure Endpoint."""
    print("\n==> Getting events from Cisco Secure Endpoint")
    # Construct the URL
    url = f"https://{client_id}:{api_key}@{host}/v1/events"

    response = requests.get(url, params=query_params)
    # Consider any status other than 2xx an error
    response.raise_for_status()

    events_list = response.json()["data"]

    return events_list

# Conseguir dominios de muestra
def threatgrid_get_domains(sample_id,
    host=tg_host,
    api_key=tg_api_key,
):
    print(white(f"\n==> Obtaining domains associated with the ThreatGrid Sample ID: {sample_id}"))

    url = f"https://{host}/api/v2/samples/feeds/domains"
    query_params = {
        "sample": sample_id,
        "after": "2020-01-01",
        "api_key": api_key,
    }

    response = requests.get(
        url,
        params=query_params,
    )
    response.raise_for_status()

    domains_json = response.json()["data"]["items"]
    domains = []
    if domains json:
```

```
        for item in domains_json:
            if (item["domain"] not in domains):
                domains.append(item["domain"])
    else:
        print(red("Unable to retrieve domains on the sha256 submission. Extend timeframe and try again."))
        sys.exit(1)

    return domains

# Bloquear hash de archivo usando la lista de detección personalizada simple
def amp_scd(method,
    scd_guid,
    sha256,
    host=amp_host,
    client_id=amp_client_id,
    api_key=amp_api_key
):
    print(f"\n==> Performing {method} file hash {sha256} for Simple Customer Detection list {scd_guid}")

    url = f"https://{client_id}:{api_key}@{host}/v1/file_lists/{scd_guid}/files/{sha256}"

    if method == 'get':
        response = requests.get(url)
        if response.status_code == 404:
            print(red("ATTENTION: The File List Item not found for given sha256."))
        else:
            response.raise_for_status()
    elif method == 'post':
        response = requests.post(url)
        if response.status_code == 409:
            print(red("ATTENTION: The file hash is already added."))
        else:
            response.raise_for_status()
    elif method == 'delete':
        response = requests.delete(url)
        response.raise_for_status()
    else:
        print(red("ERROR: Unrecognized REST API Method. Please use 'get', 'post' or 'delete'."))
        sys.exit(1)

    file_status = response.json()["data"]

    return file_status

# Construir URL para recuperar la computadora
def get_amp_computers(
    host=amp_host,
    client_id=amp_client_id,
    api_key=amp_api_key,
    ):
    """Get a list of computers from  Cisco Secure Endpoint."""
    print("\n==> Getting computers from Cisco Secure Endpoint")
    # Construct the URL
    url = f"https://{client_id}:{api_key}@{host}/v1/computers"

    response = requests.get(url)
    # Consider any status other than 2xx an error
    response.raise_for_status()

    computer_list = response.json()["data"]

    return computer_list

# Conseguir el dominio de Umbrella
def umbrella_get_domain_status(secx_client_id, secx_api_key, domain):

    client = ThreatResponse(client_id=secx_client_id, client_password=secx_api_key)

    # Pass variable along in the API request.
    inspect_response = client.inspect.inspect({'content': domain})

    deliberate_response = client.enrich.deliberate.observables(inspect_response)

    verdict = [deliberate_response["data"][x]["data"]["verdicts"]["docs"][0]["disposition_name"]
        for x in range(len(deliberate_response["data"])) if deliberate_response["data"][x]["module"] == 'Umbrella']

    return verdict

# Conseguir todos los tipos de evento Endpoint Secure
def get_amp_event_types(
    host=amp_host,
    client_id=amp_client_id,
    api_key=amp_api_key,
    ):
```

```
    """Conseguir todos los tipos de evento Endpoint Secure."""

    url = f"https://{client_id}:{api_key}@{host}/v1/event_types"

    response = requests.get(url)
    # Considerar cualquier estatus distinto a 2xx como un error
    response.raise_for_status()

    # Imprimir los tipos de eventos
    print(response.json())

# Establecer la función principal
# Si este script es el "main" ejecute...
if __name__ == "__main__":
  get_amp_event_types()
  # Configurar el nombre correcto al dispositivo
  amp_computer_name = "Demo_AMP_Threat_Audit"
  amp_computer_list = get_amp_computers()
  print(white("\nStep 1"))
  print(green(f"Fetched Cisco Secure Endpoint Computer List"))
  for computer in amp_computer_list: # Mover a la función get_amp_guid
    if computer["hostname"] == amp_computer_name:
      amp_computer_guid = computer["connector_guid"]
      print(green(f"Cisco Secure Endpoint Computer name: {amp_computer_name}, GUID: {amp_computer_guid}"))
  amp_query_params = f"connector_guid[]={amp_computer_guid}&event_type[]=1107296272&event_type[]=1090519054"
  amp_event_list = get_amp_events(query_params=amp_query_params)
  print(green(f"Retrieved {len(amp_event_list)} events from Cisco Secure Endpoint"))
  amp_event = amp_event_list[0]
  print (green(f"First Event: {amp_event['event_type']} \
  \nDetection: {amp_event['detection']} \
  \nFile name: {amp_event['file']['file_name']} \
  \nFile sha256: {amp_event['file']['identity']['sha256']}"))
  threatgrid_sha = amp_event["file"]["identity"]["sha256"]
  # Llamada a la funcion de aislamiento de dispositivo
  print(white(f"\nStep 2"))
  amp_computer_isolation = amp_isolation('put',amp_computer_guid)
  if amp_computer_isolation:
    print(green(f"Computer {amp_computer_name} (GUID {amp_computer_guid}) is {amp_computer_isolation['status']}"))
  # Ejecución de bloqueo hash de archivo usando una lista de detección personalizada simple
  amp_scd_list = get_amp_scds()
  print(green(f"Fetched Secure Endpoint Simple Custom Detection Lists"))
  for scd in amp_scd_list:
    if scd["name"] == "Simple Custom Detection List":

      amp_scd_guid = scd["guid"]
  print(green(f"Secure Endpoint SCD List ID : {amp_scd_guid}"))
  print(white(f"\nStep 3"))
  amp_file_status = amp_scd('post',amp_scd_guid,threatgrid_sha)
  if amp_file_status:
    print(green(f"File hash {threatgrid_sha} is added to SCD list (GUID {amp_scd_guid})"))
  print(white(f"\nStep 4"))
  # Call function to find all samples, associated with malicious sha256
  submission_info = threatgrid_search_submissions(threatgrid_sha)
  threatgrid_sample_id = submission_info[0]['item']['sample']
  print(green(f"Successfully retrieved Cisco Security Malware Analytics sample ID {threatgrid_sample_id} for sha265 {threatgrid_sha}"))
  print(white(f"\nStep 5"))
  # Regresa el estado del primer dominio asociado con la muestra de Threat Grid
  umbrella_domains_status = umbrella_get_domain_status(secx_client_id, secx_api_key, threatgrid_sample_domains[0])
  print(green(f"\nDomain {threatgrid_sample_domains[0]} status {umbrella_domains_status}"))
  print(yellow(umbrella_domains_status))
  umbrella_malicious_domains = []
  if umbrella_domains_status[0] == "Clean":
    print(green(f"The domain {threatgrid_sample_domains[0]} is found CLEAN"))
  elif umbrella_domains_status[0] == "Malicious":
    print(red(f"The domain {threatgrid_sample_domains[0]} is found MALICIOUS",bold=True))
    # Add the domain to a list of malicious domains.
    umbrella_malicious_domains.append(threatgrid_sample_domains[0])
  elif umbrella_domains_status[0] == "Undefined":
    print(green(f"The domain {threatgrid_sample_domains[0]} is found UNDEFINED"))
    # Ejecución de la función de Script de bloqueo de dominios
    print(white("\nStep 6"))
    umbrella_event_id, umbrella_blocklist_enforcement = post_umbrella_events(umbrella_malicious_domains)
    print(green(f"Domains {umbrella_malicious_domains} were accepted, Umbrella event id: {umbrella_event_id}"))
```

ANEXO II

Código de grid de amenazas (threatgrid.py)

```
import requests
import sys

# adds /home to the path to import environment module
sys.path.insert(0, '/home')

# import API key module
import environment as env

# Cisco Secure Malware Analytics
tg_api_key=env.SECURE_MALWARE_ANALYTICS_API_KEY

json_url = f"https://panacea.threatgrid.com/api/v3/feeds/dga-dns.json?api_key={tg_api_key}"

stix_url = f"https://panacea.threatgrid.com/api/v3/feeds/ransomware-dns.stix?api_key={tg_api_key}"

payload={}
headers = {}

json_feed_response = requests.request("GET", json_url, headers=headers, data=payload)

json_feed_response.raise_for_status()

stix_feed_response = requests.request("GET", stix_url, headers=headers, data=payload)

stix_feed_response.raise_for_status()

print(json_feed_response.text)
print(stix_feed_response.text)
```

ANEXO III

Código de respuesta de amenazas (threat_response.py)

```
from datetime import datetime
import json
import sys
from pathlib import Path
import requests
from crayons import blue, green, yellow, white, red
from requests.packages.urllib3.exceptions import InsecureRequestWarning
from pprint import pprint

# Añade /home al directorio para importar el modulo de respuesta
sys.path.insert(0, '/home')

# Importar modulo API key
import environment as env

# Deshabilitar peticiones inseguras peligrosas
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

# Configurar API keys como variables constantes:

# Cisco SecureX
secx_client_id=env.SECUREX_CLIENT_ID
secx_api_key=env.SECUREX_API_KEY
secx_host=env.SECUREX_URL

# Webex (optional)
webex_token = env.WEBEX_TEAMS_ACCESS_TOKEN
webex_room_id = env.WEBEX_TEAMS_ROOM_ID

# Autenticación
def ctr_auth(
    host=secx_host,
    client_id=secx_client_id,
    api_key=secx_api_key,
):
    print(white("\n==> Authenticating to Cisco Threat Response..."))

    # URL for API call
    url = f"https://{host}/iroh/oauth2/token"

    # headers for API call
    headers = {'Content-Type':'application/x-www-form-urlencoded', 'Accept':'application/json'}
    # payload for API call
    payload = {'grant type':'client credentials'}
```

```
    # execute API call
    response = requests.post(url, headers=headers, auth=(client_id, api_key), data=payload)

    # error checking
    response.raise_for_status()

    # grab access token from return body
    access_token = response.json()["access_token"]
    return access_token

# Inspección
def ctr_inspect(access_token, ctr_arb_text,
    host=secx_host,
):
    print(white("\n==> Take in block of arbitrary text and return a list of formatted observables as a JSON object..."))

    # URL for API call
    url = f"https://{host}/iroh/iroh-inspect/inspect"

    # headers for API call
    headers = {"Authorization":f"Bearer {access_token}", 'Content-Type':'application/json', 'Accept':'application/json'}

    # payload for API call
    inspect_payload = {'content':ctr_arb_text}
    inspect_payload = json.dumps(inspect_payload)

    # execute API call
    response = requests.post(url, headers=headers, data=inspect_payload)
    response.raise_for_status()

    # grab data to return
    observables = response.json()
    return observables

# MISSION3: Pass to the function properly formatted observables obtained in Step 1.
def ctr_enrich_observe(access_token, MISSION3,
    host=secx_host):
    print(white("\n==> Fetching Sightings about provided observables from CTR modules. Be patient, it may take time..."))

    # URL for API call
    url = f"https://{host}/iroh/iroh-enrich/observe/observables"

    # headers for API call


    headers = {"Authorization":f"Bearer {access_token}", 'Content-Type':'application/json', 'Accept':'application/json'}

    # payload for API call
    observe_payload = json.dumps(observables)

    # execute API call
    response = requests.post(url, headers=headers, data=observe_payload)
    response.raise_for_status()

    # grab data to return
    data = response.json()["data"]
    return data

# Creacion de incidentes
def create_incident(title, description, ctr_access_token):
    # store current datetime and set in right format
    d = datetime.today()
    start_date = d.strftime("%Y-%m-%d")

    # create incident object for payload
    incident_dict = {
        "description": description,
        "short_description": "Cats are nice because they are not dogs.",
        "schema_version": "1.0.11",
        "type": "incident",
        "title": title,
        "incident_time": {
            "opened": start_date,
            "discovered": start_date,
        },
        "status": "New",
        "timestamp": start_date,
        "confidence": "Medium",
    }
    data = json.dumps(incident_dict)

    # URL for API call
    url = 'https://private.intel.amp.cisco.com/ctia/incident'

    headers = {"Authorization":f"Bearer {ctr_access_token}", 'Content-Type':'application/json', 'Accept':'application/json'}

    # execute API call
    response = requests.post(url,headers=headers,data=data)
```

```
    print("Response returned by API is:")
    print(response.text)
    return json.loads(response.text)["id"]

def ctr_enrich_print_scr_report(intel):

    print(white("\n==> Here is what CTR has found:"))

    # loop through each module and check for target sightings
    for module in intel:
        print(white(f"\n==> Module: {module['module']} : {module['module_type_id']}"))
        if module["data"]:
            if module["module"] == "AMP EDR":
                print(blue(f"\n  ==> Count of Indicators: {module['data']['indicators']['count']} "))
                for indicator in module["data"]["indicators"]["docs"]:
                    print(blue(f"  ==> {indicator['description']} : {indicator['tags']}"))

                print(blue(f"\n  ==> Count of Sightings: {module['data']['sightings']['count']} "))
                sighting = module['data']['sightings']['docs'][0]
                print(blue(f"  ==> Most recent sighting: {sighting['description']}"))

                if sighting["targets"]:
                    print(blue(f"\n  ==> Targets found: {len(sighting['targets'])}"))
                    target = sighting["targets"][0]
                    print(blue(f"  ==> Most recent target: {target['type']} observed: {target['observed_time']['start_time']}"))
                    for observable in target["observables"]:
                        print(blue(f"  ==> Target {observable['type']} : {observable['value']}"))
                    print(blue(f"  ==> Target OS: {target['os']}"))
            elif module["module"] == "AMP File Reputation":
                for key in module["data"].keys():
                    print(blue(f"  ==> Count of {key}: {module['data'][key]['count']}"))
            elif module["module"] == "AMP Global Intelligence":
                for key in module["data"].keys():
                    print(blue(f"  ==> Count of {key}: {module['data'][key]['count']}"))
            else:
                print(blue("\n==> DO DATA"))

# Creacion de libros de casos
def create_casebook(ctr_observables, ctr_access_token):
    # store current datetime and set in right format
    d = datetime.today()
    start_date = d.strftime("%Y-%m-%d")

    # create casebook object for payload
```

```
    casebook_dict = {}
    casebook_dict["type"] = "casebook"
    casebook_dict["title"] = "DNE Security Lab:" + start_date
    casebook_dict["texts"] = []
    casebook_dict["observables"] = ctr_observables
    casebook_dict["short_description"] = "Something has happened!"
    casebook_dict["description"] = "Cats are cute!"
    casebook_dict["schema_version"] = "1.0.11"
    casebook_dict["timestamp"] = start_date
    casebook_dict["source"] = "ao"
    casebook_dict["tlp"] = "green"
    data = json.dumps(casebook_dict)

    # URL for API call
    url = 'https://private.intel.amp.cisco.com/ctia/casebook'

    headers = {"Authorization":f"Bearer {ctr_access_token}", 'Content-Type':'application/json', 'Accept':'application/json'}

    # execute API call
    response = requests.post(url,headers=headers,data=data)
    print("Response returned by API is:")
    print(response.text)
    return json.loads(response.text)["id"]

# Disparadores de acciones de respuesta
def ctr_add_to_amp_scd(access_token, action_url,
    host=secx_host):
    print(white("\n==> Adding a malicious sha256 to AMP Simple Custom Detections list named Quarantine..."))

    # URL for API call
    url = f"https://{host}/iroh/iroh-response{action_url}"

    # headers for API call
    headers = {"Authorization":f"Bearer {access_token}", 'Content-Type':'application/json', 'Accept':'application/json'}

    # execute API call
    response = requests.post(url, headers=headers)
    response.raise_for_status()

    return response.status_code

# Listar acciones de respuesta
def ctr_response_actions(access_token,observables,
    host=secx_host):
```

```
    """ POST https://{{ctr_host}}/iroh/iroh-response/respond/observables """

    print(white("\n==> Fetching the list of available response actions and modules for a given observable..."))

    # URL for API call
    url = f"https://{host}/iroh/iroh-response/respond/observables"

    # payload for API call
    payload = json.dumps(observables)

    # headers for API call
    headers = {"Authorization":f"Bearer {access_token}", 'Content-Type':'application/json', 'Accept':'application/json'}

    # execute API call
    response = requests.post(url, headers=headers, data=payload)
    response.raise_for_status()
    # Using list comprehentions [obj[key1] for obj in objects if obj[key2] == value]
    actions = response.json()["data"]

    print(yellow("Available response actions for this observable:"))
    pprint(actions)
    # search specifically for the AMP blocklist response action
    response_url = [action["url"] for action in actions if action["id"] == "amp-add-sha256-scd"]

    if not response_url:
        #response_url = None
        print(red(f"PLEASE NOTE: Required action is not in the list. SHA256  is already on block list. You may try to remove it or continue."))
        return False

    return response_url[0]

# Ejecutar main principal
if __name__ == "__main__":
    # add function execution below:
    print(white("\nStep 1"))

    ctr_access_token = ctr_auth()

    print(green("Received Threat Response access token"))
    print(white("\nStep 2"))
    # MISSION4: Pass to the function properly formatted observables obtained in Step 1.
    # Hint: Check the function and put correct variable there too.
    # call function


    ctr_intel = ctr_enrich_observe(ctr_access_token, 'MISSION4')

    print(green(f"Received Sightings"))
    with open("tr-enrichment.json", "w") as file:
        json.dump(ctr_intel, file, indent=2)

    # call function
    ctr_enrich_print_scr_report(ctr_intel)
    # Ejecucion de incidentes
    title = "Incident Title: Hello World"
    description = "This is my first incident description"
    print(white("\nStep 5"))
    incident_ID = create_incident(title, description, ctr_access_token)
```

ANEXO IV

Código para la conectividad para una API de flujo de trabajo seguro (csw-test.py)

```
import json
import requests
from requests.packages.urllib3.exceptions import InsecureRequestWarning
import environment as env

from tetpyclient import RestClient

# Deshabilitar peticiones inseguras de alertas
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

# Prueba de conectividad para una API de flujo de trabajo seguro
def tetration_test(
    host=env.TET.get("host"),
    api_key=env.TET_API_KEY,
        api_sec=env.TET_SEC
):
    # Build URL
    url = f"https://{host}"
    restclient = RestClient(url,
                            api_key=api_key,
                            api_secret=api_sec,
                            verify=True)
    # HTTP Get Request
    response = restclient.get("/applications")

    # If response code is 200, Test Successful
    if response.status_code == 200:
        print("Test Successful....Woo Hoo!")
        # return applications
    # If response code is anything but 200, print error message with response code
    else:
        print(f"Test Failed....DOE!"
              f"\nError Code {response.status_code}")
```

ANEXO V

Código para implementación de funciones (funciones.py)

```
import json
import os, sys
import time
from tetpyclient import RestClient
from requests.packages.urllib3 import disable_warnings
import environment as env

# Consulta de vulnerabilidades
def get_vulnerabilities ( sensor_data ):

    # Minimum CVSS V2 score defined as local constant
    MIN_CVSS_V2_SCORE = 9.0

    for sensor in sensor_data["results"]:
        sensor_hostname = sensor["host_name"]
        sensor_uuid = sensor["uuid"]
        for nic in sensor["interfaces"]:
            # Get IP from first NIC that has tags_scope_id defined
            if "tags_scope_id" in nic:
                sensor_ip = nic["ip"]
                break

         # Continue to next sensor if its not an enforcer
        if sensor["agent_type"] != "ENFORCER":
            continue

        # Query vulnerabilities

        vulnerabilities = get_csw("/workload/" + sensor_uuid + "/vulnerabilities");

        # Continue to next sensor if this sensor has no vulnerabilities to print
        if vulnerabilities is None:
            continue

        # First loop through vulns to determine if we need to print the table header
        for vul in vulnerabilities:
            if ('v2_score' in vul) and ('v3_score' in vul) and (vul['v2_score'] >= MIN_CVSS_V2_SCORE ):
                print_header = True

        # If V2 CVSS score >= MIN_CVSS_V2_SCORE, print the CVE ID, v2/v3 scores, package name and version
        for vul in vulnerabilities:
            if (print_header):
                print ("\n{:<18} {:<16} {:<16} {:<10} {:<10} {:<16} {:20}".format(
                        'HOSTNAME', 'IP', 'CVE ID', 'SCORE(V2)', 'SCORE(V3)', 'PACKAGE', 'VERSION'))
```

```
            print_header = False
            if ('v2_score' in vul) and ('v3_score' in vul) and (vul['v2_score'] >= MIN_CVSS_V2_SCORE ):
                print ("{:<18} {:16} {:<16} {:<10} {:<10} {:<16} {:20}".format(
                        sensor_hostname, sensor_ip, vul['cve_id'], vul['v2_score'], vul['v3_score'],
                        vul['package_infos'][0]['name'], vul['package_infos'][0]['version']))

# Peticion GET para un trabajo de flujo seguro
def get_csw(request_str,
    host=env.TET.get("host"),
    api_key=env.TET_API_KEY,
    api_sec=env.TET_SEC):
    # Build URL
    url = f"https://{host}"
    restclient = RestClient(url,
                            api_key=api_key,
                            api_secret=api_sec,
                            verify=True)
    # HTTP Get Request
    response = restclient.get(request_str)
    # If successful response code return list sensors
    if response.status_code == 200:
        # print(json.dumps(response.json(), indent=2))
        return response.json()
    # If response code is anything but 200, print error message with response code
    else:
        print(f"Error code: {response.status_code} getting sensors: {response.content}")

# Impresión de los sensores
def print_sensors( sensor_data ):
    print ('\033[1m',"{:<24} {:<12} {:<20} {:<30}".format('HOSTNAME', 'TYPE','IP ADDRESS', 'PLATFORM'), '\033[0m')
    for sensor in sensor_data["results"]:
        sensor_hostname = sensor["host_name"]
        sensor_type = sensor["agent_type"]
        sensor_platform = sensor["platform"]
        sensor_ip = "unkown"
        for nic in sensor["interfaces"]:
            # Get IP from first NIC that has tags_scope_id defined
            if "tags_scope_id" in nic:
                sensor_ip = nic["ip"]
                break

        print ("{:<24} {:<12} {:<20} {:<30}".format(sensor_hostname, sensor_type, sensor_ip, sensor_platform))
```

ANEXO VI

Código para obtener los sensores de la red (get_sensors.py)

```
# Consulta de sensores
# Llamada a las funciones de funciones.py
import funciones
# Get all sensors
sensors = functions.get_csw("/sensors")
# Print sensors
functions.print_sensors(sensors)
```

ANEXO VII

Código para obtener las vulnerabilidades de los sensores de la red (get_vulnerabilities.py)

```
# Import the functions we defined in the last step
import funciones
# Get all sensors
sensors = functions.get_csw("/sensors")
# Para cada sensor se ven las vulnearbilidades
funciones.get_vulnerabilities(sensors)
```