ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE SISTEMAS

UNIDAD DE TITULACIÓN

DETECCIÓN DE ATAQUES DE DENEGACIÓN DE SERVICIO ACTIVADOS MEDIANTE BOTNETS EN REDES DEFINIDAS POR SOFTWARE

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL GRADO DE MAGISTER EN SOFTWARE CON MENCIÓN EN SEGURIDAD

JAIME ORLANDO TAMAYO PORTERO
jaime.tamayo01@epn.edu.ec

Director: Ángel Leonardo Valdivieso Caraguay
angel.valdivieso@epn.edu.ec

Codirector: Marco Enrique Benalcázar Palacios
marco.benalcazar@epn.edu.ec

2023

APROBACIÓN DEL DIRECTOR

Como director del trabajo de titulación Detección de Ataques de Denegación de Servicio activados mediante botnets en Redes Definidas por Software desarrollado por Jaime Orlando Tamayo Portero, estudiante de la maestría en software con mención en seguridad, habiendo supervisado la realización de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la Defensa oral.

Firmado electrónicamente por:
ANGEL LEONARDO
VALDIVIESO CARAGUAY

Dr. Ángel Leonardo Valdivieso Caraguay

DIRECTOR

APROBACIÓN DEL CODIRECTOR

Como codirector del trabajo de titulación Detección de Ataques de Denegación de Servicio activados mediante botnets en Redes Definidas por Software desarrollado por Jaime Orlando Tamayo Portero, estudiante de la maestría en software con mención en seguridad, habiendo supervisado la realización de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la Defensa oral.

Firmado electrónicamente por:
MARCO ENRIQUE
BENALCAZAR PALACIOS

Dr. Marco Enrique Benalcázar Palacios

CODIRECTOR

DECLARACIÓN DE AUTORÍA

Yo, Jaime Orlando Tamayo Portero, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

JAIME ORLANDO TAMAYO PORTERO

Firmado digitalmente por JAIME ORLANDO TAMAYO PORTERO
Fecha: 2023.09.12 23:07:52 -05'00'

Jaime Orlando Tamayo Portero

DEDICATORIA

Quiero dedicar este trabajo a las personas más importantes en mi vida, aquellas que han estado a mi lado en cada paso de este viaje y han sido mi mayor fuente de apoyo y motivación.

A mis queridos padres y hermana, su inquebrantable confianza en mí ha sido la base sobre la cual he construido mi éxito académico. Gracias por su infinita paciencia, por estar siempre dispuestos a escuchar mis inquietudes y brindarme su sabio consejo.

A mis amigos; en especial a Thomás y Erick, quienes en todo momento brindan su ayuda incondicional representan la amistad verdadera.

Al amor de mi vida, gracias por ser mi roca, mi compañero/a y mi mayor motivación. Tu amor incondicional y tu apoyo constante me han dado la fuerza necesaria para enfrentar los desafíos de este proyecto de tesis.

Jaime

AGRADECIMIENTO

Deseo expresar mi gratitud a mis amados padres y a mi querida hermana, cuyo amor incondicional y sacrificios innumerables me han brindado el respaldo emocional y financiero para perseguir mis metas académicas. Su confianza en mí ha sido un motor incansable de inspiración y motivación.

También quiero expresar mi más sincero agradecimiento a mi director de tesis, Dr. Leonardo Valdivieso, por su valiosa orientación y conocimientos expertos a lo largo de este proyecto. Su visión clara, su paciencia y su disposición para brindarme retroalimentación constructiva han sido fundamentales para mi crecimiento académico y profesional.

No puedo pasar por alto la destacada contribución de la Dra. Pamela Flores, cuya incondicional ayuda ha sido una de las principales razones para lograr el objetivo de este proyecto. Su sabiduría, su dedicación y su apoyo constante han sido esenciales para superar obstáculos y alcanzar resultados significativos. Le estoy profundamente agradecido por su generosidad y compromiso.

Por último, pero no menos importante, deseo expresar un agradecimiento especial al amor de mi vida. Tú has sido mi roca, mi inspiración y mi confidente durante todo este proceso. Tu apoyo inquebrantable, tus palabras de aliento y tu presencia constante han sido una fuente inagotable de fortaleza y motivación. No tengo palabras suficientes para describir cuánto significas para mí y cuánto valoro tu amor y apoyo.

ÍNDICE DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

RESUMEN

En los últimos años, el volumen de tráfico circulando en redes públicas y privadas ha aumentado gracias a la aparición de nuevos servicios. Por esta razón, las redes periódicamente sufren problemas significativos por su complejidad y saturación. Las redes definidas por software (SDN, por sus siglas en inglés) son una nueva arquitectura que ofrece ventajas innovadoras que ayudan a reducir los problemas de saturación. Sin embargo, tanto las redes tradicionales como las SDN son víctimas de los mismos tipos de ataques, a pesar de las diferencias en el funcionamiento de su arquitectura. En este contexto, el Ataque Distribuido de Denegación de Servicio (DDoS, por sus siglas en inglés) se considera uno de los ataques más importantes que puede afectar el funcionamiento normal de una red SDN. Además, si estos ataques se ejecutan mediante el uso de botnets, pueden aprovechar el poder de miles de dispositivos comprometidos para abrumar e interrumpir servicios en línea críticos. Este artículo propone un *framework* para detectar ataques DDoS generados por un grupo de botnets en una red SDN. Este se implementa utilizando herramientas de código abierto y se prueba en una topología de red centralizada. Los resultados de los experimentos demuestran que el sistema puede identificar rápidamente un ataque en tiempo real mediante la implementación de un mecanismo de detección de intrusiones en la víctima. Nuestra solución propuesta ofrece un método de detección rápido y efectivo contra ataques DDoS en SDN. La solución puede diferenciar con alta precisión el tipo de ataque en poco tiempo.

Palabras clave: Seguridad de la información, Ataques distribuidos de denegación de servicio, Redes definidas por software, Botnets, Sistema de detección de intrusos, SNORT.

ABSTRACT

In recent years, the volume of traffic circulating on public and private networks has increased thanks to the emergence of new online services. For this reason, networks periodically suffer from significant saturation and complexity problems. Additionally, the number of IoT devices connected to the network constantly grows. Thus, software-defined network (SDN) is a new architecture which offers innovative advantages that helps to reduce saturation problems. Despite its advantages, SDNs introduce new security challenges. Both traditional and SDNs networks are victims of the same types of attacks despite the difference in how their architecture works. On this context, Distributed Denial of Service (DDoS) is considered one of the most important attacks which can damage the normal operation of an SDN network. Furthermore, if these attacks are executed through the use of botnets, they can harness the power of thousands of compromised devices to overwhelm and disrupt critical online services. This paper proposes a framework for detecting DDoS attacks generated by a group of botnets in an SDN network. The framework is implemented using open-source tools such as Mininet and OpenDaylight and tested in a centralized network topology using BYOB and SNORT. The results of the experiments demonstrate that the system can rapidly identify an attack in real-time by implementing an intrusion detection mechanism in the victim client. Our proposed solution offers a quick and effective detection method against DDoS attacks in SDN networks. The framework can successfully differentiate the type of attack with high accuracy in a short time.

Keywords: Information security, Distributed denial of service attacks, Software-defined networks, Botnets, Intrusion detection system, SNORT.

1. INTRODUCCIÓN

Las Redes Definidas por Software (SDN por sus siglas en inglés *Software Defined Networks*) ofrecen un modelo de red novedoso que difiere de las redes tradicionales y proporciona varias ventajas, como una mayor flexibilidad, escalabilidad y gestión centralizada. Las redes de comunicación pasan de tener una infraestructura rígida a un diseño más flexible, abierta y programable [1]. La arquitectura SDN separa los planos de datos y control en los dispositivos de red y centraliza la lógica de control de los diferentes dispositivos (switches), que solo reenvían el tráfico. Es decir, el controlador toma decisiones de manera centralizada sobre las rutas de datos en la red [2]. El controlador se puede programar de acuerdo con las necesidades particulares de cada usuario. Por su parte, los switches siguen las instrucciones proporcionadas por el controlador. Además, esta nueva arquitectura permite que este tipo de redes SDN sean más escalables que las redes tradicionales [3].

El volumen de tráfico circulando en redes públicas y privadas ha aumentado gracias a la aparición de nuevos servicios en línea. Por esta razón, las redes periódicamente sufren problemas significativos de saturación y complejidad [4]. Los entornos de redes tradicionales no se autoconfiguran y, por lo tanto, no pueden adaptarse adecuadamente cuando aparecen nuevas cargas en la red. Además, la flexibilidad de crecimiento de la red se ve reducida al presentar un tipo de integración vertical [3]. Para los enrutadores convencionales, resulta difícil y requiere mucho tiempo modificar su comportamiento. En este caso, los administradores deben configurar cada dispositivo individualmente para cambiar las políticas de red de alto nivel, a menudo utilizando comandos específicos del proveedor. En cambio, la configuración de una red SDN es más dinámica y eficiente. La existencia de un controlador central en una red SDN facilita en gran medida las tareas de enrutamiento y control en comparación con una red tradicional [5].

Se espera que la seguridad sea también un área de aplicación para las SDNs. Sin embargo, aunque la arquitectura SDN trae múltiples ventajas, también plantea nuevos desafíos en términos de seguridad de red. Los problemas de seguridad en

las redes SDN se pueden dividir en dos tipos: ataques heredados de las redes tradicionales y nuevos ataques dirigidos específicamente a la arquitectura SDN. En este sentido, uno de los ataques más recurrentes es la Denegación de Servicio (DoS), que causa daños tanto a los usuarios como al proveedor de servicios. Este tipo de ataque puede afectar la funcionalidad de la red y resultar en pérdidas financieras y de prestigio.

Los ataques de DoS son un problema para el correcto funcionamiento de una red, ya que su objetivo principal es interrumpir los servicios al limitar el acceso a una máquina o servicio [6]. Además, existe una variante de DoS que puede aumentar el daño en la red. Un ataque se considera un Ataque Distribuido de Denegación de Servicio (DDoS por sus siglas en inglés *Distributed Denial-of-Service*) cuando el ataque DoS está coordinado. El atacante identifica vulnerabilidades e instala malware en múltiples máquinas para controlarlas. Luego, las máquinas controladas se convierten en una botnet y son comandadas por un botmaster [7]. El botmaster puede ejecutar acciones maliciosas en los dispositivos infectados [8]. Si el número de máquinas comprometidas es enorme, pueden inhabilitar el servicio de un servidor en muy poco tiempo. Por ejemplo, en febrero de 2022, se lanzó un ataque de denegación de servicio distribuido (DDoS) contra los sitios web de bancos y el Ministerio de Defensa de Ucrania, después del cual muchos países señalaron a Rusia como responsable, lo que demuestra que el ciberespacio juega un papel importante en el conflicto entre Rusia y Ucrania [9].

La Figura 1 muestra la clasificación de los ataques DDoS, de los cuales se seleccionarán los ataques de ICMP Flood y SYN Flood para este proyecto. Estos ataques se seleccionaron porque se consideran dos de los ataques más importantes debido a su efectividad para agotar los recursos del servidor objetivo y afectar la disponibilidad de los servicios en línea. Estos ataques han sido ampliamente documentados por organizaciones de seguridad y han sido objeto de numerosos informes y análisis en la comunidad de ciberseguridad [10] [11].

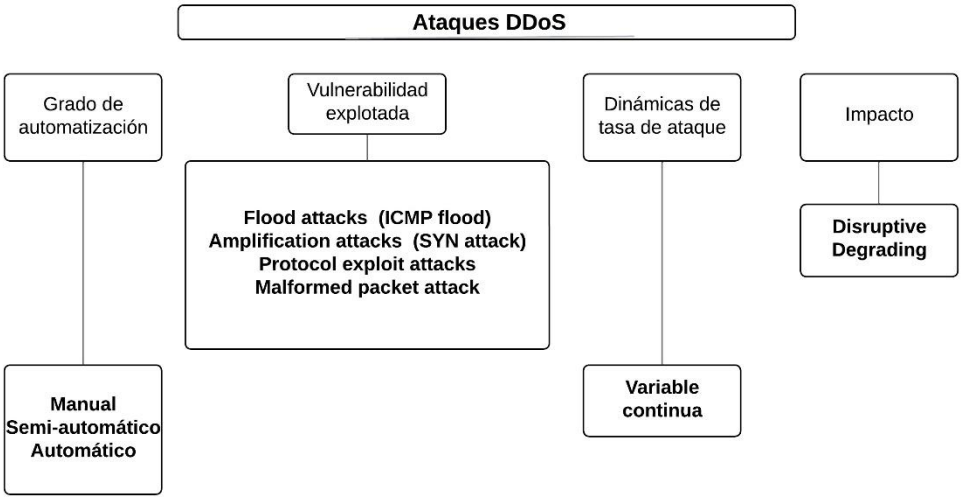

Figura 1. Clasificación de ataques DDoS [12]

Si nos referimos a los ataques DDoS en redes SDN, se generan creando muchos flujos que inundan el ancho de banda del plano de control, saturando los interruptores y el controlador de la red. Cuando un controlador se ve comprometido y sufre un ataque DDoS, puede hacer que toda la red falle, ya que el controlador es responsable de implementar la lógica de red y gestionar las aplicaciones e interruptores.

Por todo lo anterior, nuestro framework propuesto tiene como objetivo la detección temprana de ataques DDoS en SDNs. Para lograr esto, se crea la red SDN en un entorno controlado utilizando máquinas virtuales. Adicionalmente se presenta el módulo de ataques que consiste en la infección de botnets para controlar usuarios de la SDN por medio de un script de malware que elabora el botmaster y convertirlos en zombies. Los zombies perpetúan los ataques hacía la víctima dentro de la red interrumpiendo el normal funcionamiento de los recursos de red. El módulo de detección se instala en la máquina de la víctima y siempre estará activo analizando todo el tráfico que recibe el usuario.

Para el montaje del framework se utilizarán herramientas *Open Source* o herramientas de software libre debido a que son gratuitas, pueden ser modificadas

para adaptarlas a necesidades específicas, y también presentan facilidad para su integración con otros sistemas o software. La arquitectura de red SDN es una opción con múltiples ventajas para entornos en constante crecimiento. Por lo tanto, la implementación de una técnica de detección de ataques DDoS es una contribución útil para el desarrollo de marcos de trabajo para la seguridad de las SDN.

1.1. Pregunta de investigación

¿Cómo se puede detectar de manera efectiva los ataques de denegación de servicio distribuido (DDoS) generados por botnets en una red SDN?

1.2. Objetivo general

Desarrollo de un *framework* para detección de ataques DDoS activados mediante botnets en Redes Definidas por Software.

1.3. Objetivos específicos

- Revisar el estado del arte y seleccionar las herramientas que permitan la emulación de redes SDN, botnets y ataques DDoS.
- Diseñar un *framework* para la detección de ataques DDoS activados por botnets en redes SDN.
- Implementar el *framework* de detección de ataques DDoS activados por botnets en redes SDN utilizando herramientas open-source.
- Realizar pruebas en la red SDN con y sin ataques DDoS activados por botnets para analizar la diferencia de los tiempos entre inicio de ataque y detección, saturación del ancho de banda de la red y consumo de recursos de la víctima.
- Evaluar y documentar los resultados del uso del *framework* en términos de tiempo, saturación y consumo de recursos en un artículo científico.

2. MARCO TEÓRICO Y TRABAJOS RELACIONADOS

2.1. Marco Teórico

En esta subsección se revisan componentes principales del *framework* de detección de ataques DDoS, la arquitectura SDN y el módulo de ataques mediante botnets. Se presenta también una breve noción de las herramientas *open source* utilizadas para la elaboración del *framework*.

2.1.1. Redes Definidas por Software y Ataques DDoS

Redes Definidas por Software (SDN)

SDN es una arquitectura de red que funciona de manera diferente a la red tradicional. Esta arquitectura se divide en tres capas: capa de aplicación, capa de control y capa de infraestructura o plano de datos, como se muestra en la Figura 2. Los planos de control y datos están desacoplados en la arquitectura SDN, y la infraestructura de red subyacente se abstrae de las aplicaciones superiores [3].

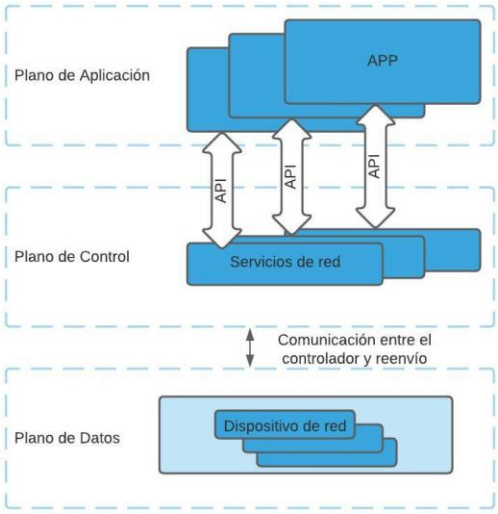

Figura 2. Arquitectura SDN [13]

El plano de datos contiene dispositivos de conmutación físicos o virtuales, que se convierten en dispositivos simples de reenvío de paquetes. El ejemplo más notable de un dispositivo del plano de datos con estas características es el switch OpenFlow [14], [4]. Un switch OpenFlow tiene una o más tablas de reglas de manejo de

paquetes (tabla de flujo). Dependiendo de las reglas instaladas por una aplicación de controlador, el controlador puede instruir a un switch OpenFlow para que se comporte como un enrutador, un switch, un cortafuegos o desempeñe otros roles (en general, los de un middlebox) [3].

La capa de control generalmente está compuesta por controladores basados en software centralizados de forma lógica; ellos controlan el comportamiento del plano de datos a través de una *southbound* API y proporcionan servicios de red a las aplicaciones superiores a través de una *northbound* API. El protocolo OpenFlow es el primer y más ampliamente implementado protocolo *southbound* de SDN [1]. El protocolo OpenFlow define el mecanismo de comunicación que permite que el controlador SDN interactúe directamente con el plano de datos. El controlador empuja reglas de manejo de paquetes en las tablas de flujo de los switches OpenFlow. La regla coincide con las condiciones de tráfico y realiza acciones específicas, como descartar, reenviar y modificar el tráfico [1].

Por su parte, la capa de aplicación consiste principalmente en diversas aplicaciones comerciales, como redes virtuales, seguridad y aplicaciones de ingeniería de tráfico. La capa de aplicación está en la parte superior de la arquitectura SDN y contiene las aplicaciones SDN para diversas funcionalidades, como implementación de políticas, gestión de redes y servicios de seguridad. En la capa de aplicación, la aplicación SDN puede utilizar el método programable para enviar el comportamiento de la red al plano de control [15].

Botnets and DDoS Attacks on SDNs

Un ataque de Denegación de Servicio (DoS) tiene como objetivo interrumpir el funcionamiento regular de un dispositivo o red y perturbar la actividad del usuario [5]. Uno de los ataques DoS mediante la técnica de *flooding* más utilizados es ICMP Flood [13]. Este ataque consiste en enviar paquetes IP falsificados a todos los hosts en la red. Esto amplifica el tráfico y detiene el procesamiento de paquetes legítimos en la red. Una variación de los ataques DoS es el Ataque Distribuido de Denegación de Servicio (DDoS), que ocurre cuando múltiples dispositivos sincronizados realizan un ataque DoS contra una víctima [5].

El objetivo de un ataque DDoS gira en torno al hecho de que se utiliza un gran número de fuentes distribuidas en varias ubicaciones para atacar a una víctima [1]. Esta distribución es esencial para aumentar la magnitud y efectividad del ataque y dificultar su detección y mitigación. Las botnets son típicamente útiles para lanzar ataques DDoS, ya que son una amplia colección de hosts comprometidos (también llamados zombies). La forma en que los bots son controlados depende de la arquitectura de los mecanismos de comando y control de la botnet, que pueden ser basados en IRC, HTTP, DNS o P2P. Dado que los ataques DDoS son frecuentes, el enfoque ha sido desarrollar una solución eficiente capaz de detectar de manera efectiva los ataques DDoS. Para lanzar un ataque, un atacante generalmente sigue cuatro pasos básicos [6]:

- Recopilación de información para escanear una red y encontrar hosts vulnerables que se utilizarán posteriormente para lanzar un ataque.
- Compromiso de los hosts para instalar malware o programas maliciosos en los hosts comprometidos o zombies para que solo puedan ser controlados por el atacante.
- Lanzamiento del ataque para comandar a los zombies que envíen a la víctima varios flujos maliciosos con intensidades personalizables.
- Limpieza para eliminar todos los registros o archivos de historial de la memoria.

Las tendencias actuales en seguridad de DDoS muestran las limitaciones de las redes tradicionales las cuales se propagan hacia las redes SDN, como la sobrecarga de recursos, o los ataques dirigidos a los dispositivos de red (en el caso de las SDN dirigidos a los controladores). Los atacantes están adoptando mecanismos sofisticados para evadir los escudos de protección tradicionales. En los últimos años, las SDN han surgido como un nuevo paradigma de redes con un gran despliegue en servicios de gran escala, incluido las de seguridad de red. Las características distintivas de SDN han llevado al desarrollo de algunos mecanismos de detección de ataques DDoS basados en SDN.

2.1.2. Herramientas Open Source

A continuación, se presentará un breve resumen de los conceptos clave detrás de las herramientas *Open Source* empleadas en la implementación del framework para la detección de ataques DDoS en redes SDN.

Mininet

Mininet es una herramienta de emulación de SDN que proporciona una plataforma de virtualización ligera para la simulación completa de redes. Entre sus funcionalidades puede crear topologías personalizadas e interactuar con programas reales que se ejecutan en el sistema operativo Linux [16]. Las topologías personalizadas consisten en los siguientes componentes: hosts aislados, enlaces emulados y switches emulados. Los switches virtuales son switches OpenFlow que seguirán las órdenes dadas por un controlador. Los enlaces virtuales entre los switches virtuales y los hosts se implementan utilizando pares de Ethernet virtuales proporcionados por el kernel de Linux [16].

OpenDaylight (ODL)

ODL es uno de los proyectos de controlador de código abierto basado en SDN más grandes gestionados por la Linux Foundation. OpenDaylight es una plataforma de controlador SDN implementada en Java. Como tal, puede ser implementado en cualquier hardware y sistema operativo que admita Java. La arquitectura del controlador SDN de OpenDaylight se muestra en la Figura 3. Adicionalmente, ODL admite protocolos distintos a OpenFlow [2], permite el soporte de múltiples complementos de protocolo de enlace sur y un conjunto diverso de servicios y aplicaciones [3].

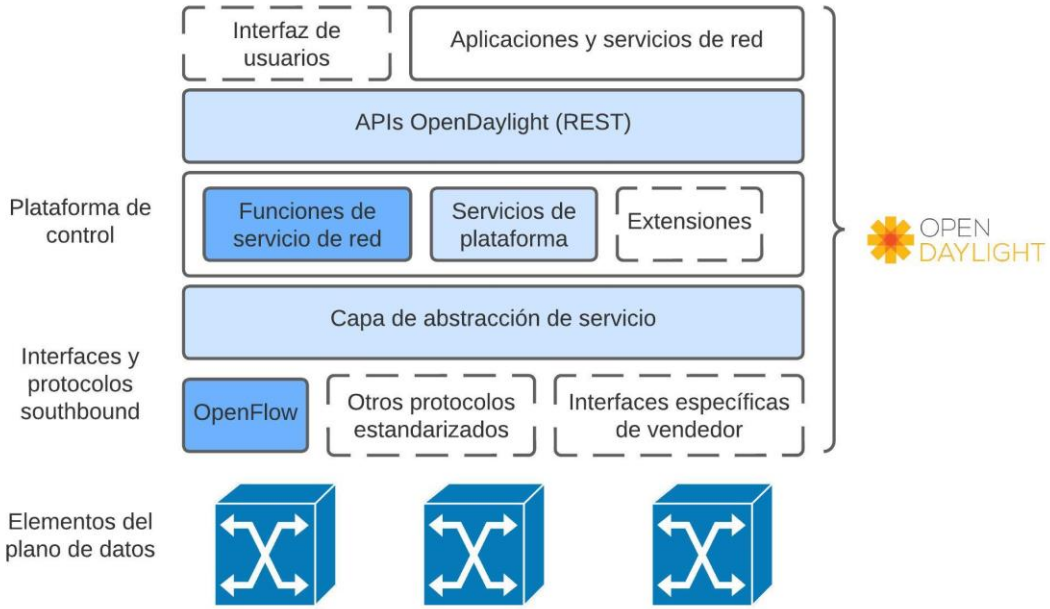

Figura 3. Arquitectura OpenDaylight [17]

BYOB (Build Your Own Botnet)

BYOB es un *framework* que implementa todos los elementos necesarios para construir una botnet. BYOB fue desarrollado con fines educativos y de defensa cibernética. El bot creado por BYOB tiene capacidades sofisticadas a nivel de herramientas avanzadas de APT. Si bien es valioso para el lado de la defensa, también permite que cualquier "*script kiddie*" con malas intenciones aproveche el *framework* para llevar a cabo ataques que de otra manera no podría realizar. El *framework* contiene cuatro partes principales:

- **Servidor**: Servidor de comando y control con base de datos persistente y consola de gestión de clientes, en este caso, los zombies. Este servidor se crea fuera de la estructura de red desarrollada por Mininet.
- **Interfaz de usuario basada en consola**: Es una interfaz para controlar las máquinas cliente de forma remota a través de shells inversos.
- **Base de datos persistente SQLite**: Base de datos ligera que almacena información de identificación de las máquinas cliente, permitiendo que las sesiones de rollback persistan.

- **Arquitectura cliente-servidor**: Todos los módulos de Python instalados localmente son creados por el servidor y están disponibles automáticamente para todos los hosts en la red de Mininet. Mientras el script no se ejecute, el host no se convertirá en un zombie.

Hping3

Hping3 es un analizador y ensamblador que proporciona una interfaz de línea de comandos que permite especificar y configurar parámetros para realizar ataques de red específicos. No solo es capaz de realizar el ataque ICMP común, sino que también admite los protocolos UDP y TCP [4]. Para el ataque, se puede decidir cuántos paquetes se transmitirán, si se mostrará una dirección IP de origen real o falsa, los puertos y las direcciones IP de destino. Cabe mencionar que esta herramienta se utiliza principalmente con fines de seguridad y no tiene la intención de ser perjudicial para los usuarios [5].

SNORT

SNORT es un sistema de detección de intrusiones de código abierto programado en lenguaje C. En tiempo real, SNORT puede analizar de manera robusta el tráfico de red y los protocolos. SNORT se ejecuta en casi todas las plataformas y sistemas operativos, y se puede descargar de forma gratuita desde Internet [6]. SNORT tiene cuatro arquitecturas que contienen [7]:

- ***Sniffer* de paquetes**: Espía el tráfico de red (columna vertebral de la red).
- **Preprocesador**: Verifica los paquetes para determinar si cumplen ciertos comportamientos según un conjunto de reglas creadas en un script.
- **Motor de detección**: Toma los paquetes entrantes y los compara con las reglas.
- **Alerta o registro**: Cuando se cumplen las reglas, se produce la alerta o el registro, ya sea que se registre en la base de datos o se alerte al usuario.

2.2. Trabajos relacionados

Algunos trabajos previos han incursionado en el estudio de mecanismos de detección y mitigación de ataques DDoS en SDNs. En [7] se implementa un sistema de detección de intrusos en redes SDN utilizando tres máquinas virtuales: la primera tiene el controlador y el IDS, la segunda emula el dominio de la red y la tercera representa un servidor en línea. El sistema detecta automáticamente varios ataques DDoS y notifica al controlador de la infraestructura mediante un *framework* en el controlador RYU. Luego, transfiere nuevas reglas de flujo a los dispositivos de red para restaurar el funcionamiento normal de la red lo más rápido posible. Este proyecto implementa todo el sistema de detección en el controlador de red SDN, lo que aumenta la cantidad de trabajo que realiza este dispositivo, siendo recomendable que el sistema de detección deba instalarse lo más cerca posible de la máquina donde se origina el ataque.

En [15], se propone un mecanismo de defensa desde el origen contra un ataque DDoS tipo Flooding utilizando botnets en redes SDN y la tecnología sample Flow (sFlow). El algoritmo de detección desarrollado se basa en un modelo de inferencia estadística. En este proyecto se desarrolla una aplicación y se prueba su correcto funcionamiento en una red emulada con tráfico real. Los resultados muestran que este mecanismo detecta eficazmente los ataques de DDoS tipo Flooding en entornos SDN e identifica los flujos para evitar que el daño del ataque se extienda más allá del objetivo. A diferencia de la contribución de este documento, nuestro trabajo permite el diseño y la creación de botnets personalizadas mediante el uso de BYOB para ejecutar ataques DDoS, mientras que [15] solo realiza un ataque simplificado sin estructurar una red de bots para ejecutar la acción.

El trabajo en [4] propone un esquema de bloqueo DDoS aplicable a una red gestionada por SDN. La aplicación que realiza el trabajo de detección y bloqueo está montada en el controlador de aplicaciones POX, y el protocolo de comunicación entre dispositivos es OpenFlow. La emulación muestra que la aplicación POX filtra con éxito el tráfico de ataques DDoS del tráfico legítimo. El tráfico malicioso se bloquea y redirige de forma segura el tráfico del usuario desde la dirección del servidor atacado a una nueva dirección. El proyecto en discusión

requiere una comunicación dedicada entre el controlador DDoS y el servidor o usuario a proteger. En cambio, nuestra propuesta no necesita de una comunicación dedicada ya que el sistema de protección se monta sobre el usuario a proteger.

En [1], se propone un *framework* denominado ProDefense. Se centra principalmente en la operación de infraestructuras a gran escala para proteger contra ataques DDoS para un centro de datos en una ciudad inteligente. El *framework* es modular y se ajusta a los requerimientos de varias aplicaciones ejecutadas en la ciudad inteligente. Por ejemplo, un Sistema de Control de Tráfico requiere una solución de seguridad extremadamente ágil que pueda activar el sistema de mitigación de inmediato y generar alertas de seguridad que anticipen comportamientos maliciosos antes de alcanzar el umbral. La solución de seguridad también debe monitorear las tendencias del tráfico de la red y predecir el ataque [1]. Si bien este trabajo diseña la detección de varios ataques DDoS en función de los requisitos de la aplicación, nuestro trabajo se centra en las categorías de Flooding y amplificación. Adicionalmente, el sistema de detección se implementa en la máquina de la víctima.

Se han realizado otros estudios sobre los diversos métodos de detección de ataques DDoS. En [18], [19] y [20] se han llevado a cabo algunas investigaciones utilizando la técnica de la entropía. La técnica de entropía se relaciona con las alteraciones de las características de la red para detectar actividades de red anómalas e identificar un posible ataque DDoS. En [21] y [22], los autores utilizan técnicas de aprendizaje automático como redes bayesianas, SOM (*self-organizing maps*) o lógica difusa para identificar la presencia de anomalías. Las técnicas mencionadas tienen en cuenta diversas características de la red, así como el análisis del tráfico para lograr la detección de posibles ataques DDoS.

El artículo presentado en [23] utiliza la técnica de análisis de patrones que asume que los atacantes exhiben un comportamiento similar, como enviar el mismo tipo de paquete malicioso o realizar la misma acción de escaneo dentro de la red. Estas actividades maliciosas se detectan e identifican como parte de un ataque DDoS. Esto demuestra que los patrones de comportamiento malicioso que se dan en las

redes tradicionales son de ayuda para la detección de futuros ataques dirigidos a las redes SDN. Por su parte, los estudios mostrados en [24] y [25] presentan la técnica de tasa de conexión, definida como un indicador del número de conexiones realizadas dentro de una ventana de tiempo específica. Si se supera este número de conexiones, se puede considerar un ataque DDoS. En nuestro estudio, se utiliza una lista de reglas en lugar de un indicador, donde si se cumplen las condiciones establecidas en la regla, se lanza una notificación de un posible ataque DDoS.

En [26] se propone un sistema de sensores que monitorean la red y un conjunto de funciones de correlación en los Open vSwitches (OVS). Cuando un monitor envía una alerta, el correlacionador la analiza para ver si coincide con alguna firma de ataque. Si es así, el monitor, el controlador y el correlacionador toman medidas para mitigar el impacto del ataque. Nuestro proyecto propuesto intenta detectar un ataque DDoS mediante el uso de SNORT IDS sin el uso de monitores adicionales como parte de la arquitectura de red SDN. En [27], los autores utilizan SNORT y OpenFlow para detectar y mitigar ataques DDoS en tiempo real modificando las configuraciones de red tradicionales en un entorno de nube. Nuestro proyecto utiliza SNORT IDS y el protocolo de comunicaciones OpenFlow para detectar ataques DDoS en una arquitectura de red SDN en lugar de en una red tradicional.

En [28] se propone una red SDN con un controlador ONOS para detectar mediante SNORT un ataque basado en TCP-SYN generado desde Hping3. La principal diferencia con este proyecto es que las reglas SNORT se implementan en el controlador ONOS de la red SDN, mientras que en nuestro proyecto las reglas se implementan en la computadora de la víctima. De esta forma, el trabajo de detección de cualquier posible ataque DDoS se asigna a la propia máquina víctima y no como un trabajo adicional del controlador. Además, nuestro proyecto compara dos ataques DDoS conocidos para la detección (inundación SYN e inundación ICMP) en comparación con este proyecto propuesto que solo aborda la detección de un solo ataque (basado en TCP-SYN). El controlador utilizado en [28] utiliza la plataforma ONOS a diferencia del nuestro, que utiliza la plataforma OpenDaylight.

En [29], los autores analizan las limitaciones de los métodos tradicionales basados en umbrales para detectar y mitigar ataques DDoS en redes SDN y exploran el potencial del aprendizaje automático y las técnicas de aprendizaje profundo para mejorar la precisión de la detección. Los autores revisan varias soluciones y conjuntos de datos existentes y proponen un nuevo enfoque basado en el aprendizaje profundo que utiliza funciones específicas de SDN y dos métodos de selección de funciones para identificar funciones relevantes para la detección de ataques DDoS. El enfoque propuesto se prueba en tres conjuntos de datos y logra altas tasas de precisión en comparación con las técnicas tradicionales de aprendizaje automático. Los autores destacan la necesidad de conjuntos de datos más actualizados y diversos para futuras investigaciones en esta área.

La principal diferencia es que nuestro proyecto utiliza la herramienta SNORT para la detección de ataques DDoS en una red SDN simulada con Mininet y OpenDaylight, mientras que el proyecto discutido en [29] propone un enfoque basado en aprendizaje profundo para mejorar la precisión de detección de ataques DDoS en Redes SDN. Además, nuestro proyecto se enfoca en la detección de ataques SYN Flood e ICMP Flood dirigidos a un usuario específico en la red SDN, mientras que el proyecto discutido en [29] evalúa la precisión de detección en tres conjuntos de datos diferentes y propone un nuevo enfoque basado en profundidad aprendizaje para la detección de ataques DDoS en redes SDN.

En [16] los ataques DDoS contra controladores centralizados en SDN se detectan utilizando la herramienta de emulación Mininet. Se realizaron experimentos utilizando diferentes herramientas de penetración para lanzar ataques DDoS y se integró SNORT para la detección. Los resultados mostraron que los controladores ODL y ONOS son vulnerables a los ataques DDoS y que el tiempo de detección es directamente proporcional a la cantidad de dispositivos de red y la cantidad de tráfico bombardeado. Se crearon reglas en SNORT para la detección de ataques DDoS y se discute la implementación de un sistema de detección DDoS usando SNORT IDS en ODL y controladores ONOS en SDN. El rendimiento del sistema se evaluó utilizando varios escenarios con diferentes números de hosts, conmutadores y tráfico, y se encontró que ODL detectó ataques DDoS más rápido que ONOS. En

[16] la implementación de su sistema de detección no se realiza en el cliente víctima, sino en los controladores, lo que aumenta el tiempo de respuesta entre el ataque y la detección.

Todos los proyectos analizados anteriormente se resumen en la Tabla 1.

Tabla 1. Resumen de trabajos relacionados a detección de ataques en SDN.

Ref.	Alcance	Algoritmo	Experimentos	Resultados	Debilidades
[1]	ProDefense, SDN, ataques DDoS	*Framework* ProDefense para un Data Center de ciudades inteligentes.	Solo análisis teórico	Ninguno	Controladores sin protección contra ataques dirigidos.
[4]	Ataques DDoS, controlador POX, botnets	Contador de flujo, detecta y elimina bots.	SDN montado en Mininet con varias botnets	Bloqueo de ataques después de varios segundos de inicio del mismo.	Comunicación entre la aplicación de bloqueo DDoS y el servidor que debe protegerse.
[7]	Modificación de reglas de flujo	Modificación en las reglas de flujo después de la detección de ataques DDoS	Tres escenarios con diferentes tipos de ataques DDoS	Tiempo de reacción de ataque promedio de 3 segundos	Sistema de detección implementado en el controlador
[15]	Tecnología sample Flow (sFLOW)	Modelo de inferencia estadística.	Red emulada con tráfico real	Detecta ataques DDoS tipo Flooding.	Solo realiza un ataque simplificado / No se usan botnets.
[18], [19], [20]	Alteraciones características de la red.	Técnica de entropía	Pruebas de rendimiento para validar la escalabilidad y la efectividad del	Mecanismo mejorado basado en sFLOW.	Toda la tabla de flujo no escala para entornos de tráfico elevado de red.

			enfoque basado en sFLOW		
[23]	Detección de malware para dispositivos móviles con SDN.	Identificar actividades de red sospechosos a través del análisis de tráfico en tiempo real	Datos para pruebas locales e infraestructura GENI	Enfoque de factibilidad.	No se analizan todos los tipos de malware.
[24], [25]	Enfoque híbrido para detectar worms.	FRESCO / prueba de hipótesis secuencial y limitación de la velocidad de conexión.	Dos módulos con escáner de eventos maliciosos / redirigir todo el flujo escaneado en una red remota honeynet	FRESCO con más del 90% menos líneas de código. / Restringe con éxito el número de escaneos / muy efectivo.	Se asegura de que los worms se identifiquen rápidamente con una baja tasa de falsa alarma.
[26]	Monitoreo de SDN y su controlador	Inspeccionar selectivamente los paquetes de red a pedido	Detección y mitigación de ataques TCP SYN Flood en un GENI.	Escalable para procesar un alto volumen de tráfico y detectar ataques de gran escala.	El proyecto utiliza monitores adicionales para lograr una inspección correcta de los paquetes a pedido.
[27]	SNORT y OpenFlow para detectar ataques DDoS.	Modificación de configuraciones de red en un entorno en la nube	Entorno de prueba simplificado que incluye dos Servidores nubes.	Variación del rendimiento del agente SNORT Flow en escenarios diferentes.	Todo el proyecto está montado en una red tradicional, no en una SDN.

[28]	Red SDN con detección de SNORT	Detección de ataques basados en TCP-SYN	Creación de una red analizada con Wireshark e implementadas reglas de detección de ataque con SNORT	Ninguno	Aumenta la carga en el procesamiento del controlador.
[29]	Limitaciones de los métodos tradicionales basados en el umbral.	NFDLM, una red neuronal artificial ligera y optimizada	Diseño de cuatro modelos diferentes donde dos se basan en ANN y los otros dos se basan en LSTM para detectar tipos de ataque DDoS.	El rendimiento de detección logra aproximadamente el 99% de precisión para la detección.	No se centra en ataques DDoS específicos, sino que los generaliza.
[16]	Ataques DDoS contra controladores SDN.	Integración de SNORT para la detección de ataques DDoS	Se consideran cinco escenarios de red diferentes. El número de hosts, conmutadores y paquetes de datos varía.	Los controladores ODL y ONOS son vulnerables a los ataques DDoS	Los ataques no se generan usando botnets.

El presente trabajo de titulación intenta superar algunas de las limitaciones analizadas anteriormente. Por ello, se pretende implementar un método de detección montado en el usuario o servidor víctima del ataque DDoS, evitando el procesamiento extra o excesivo en el controlador. Adicionalmente, el módulo de detección implementado en este framework busca obtener un tiempo en la identificación de ataques más eficiente que los trabajos analizados. Los ataques DDoS utilizados en nuestro proyecto para su detección son ICMP Flood y SYN

Flood debido a que son los ataques más comunes y más peligrosos para el correcto funcionamiento de una red.

3. METODOLOGÍA

El presente trabajo será realizado en 4 fases, tal como se describe a continuación:

- La primera fase consiste en el planteamiento del problema. Se realizará la respectiva revisión de literatura de artículos relacionados con las redes SDN, métodos de detección de ataques en redes SDN, botnets, ataques DDoS en redes SDN.
- La segunda fase consiste en el diseño del *framework* de detección de ataques DDoS controlados por botnets dentro de una red SDN. El *framework* permitirá simular una red SDN funcional incluido el controlador. Posteriormente, diversos hosts pertenecientes a la red SDN serán infectados mediante botnets. Finalmente, los botnets serán controlados por el atacante para efectuar un ataque DDoS sobre la red.
- La tercera fase implementará el esquema propuesto mediante el uso de herramientas open-source. Toda la solución será implementada en un servidor perteneciente al Laboratorio de Inteligencia y Visión Artificial “Alan Turing” de la Facultad de Sistemas.
- Para la cuarta fase se utilizará una red SDN simulada en el servidor, el mismo que consta de 20 hosts, 7 switches y un controlador SDN. Las pruebas a realizar son: (a) operación normal de la red (sin ataques) y (b) ataque DDoS sobre la red utilizando botnets en 5 hosts. En este caso, se verificará que la solución sea capaz de detectar el ataque DDoS y enviar la alerta respectiva.
- Como quinta fase se realizará la recopilación de resultados y evaluación de los datos obtenidos en las pruebas. Estos datos se mostrarán de acuerdo con las métricas que la herramienta de detección proporcione. Finalmente, se documentará los resultados obtenidos en un artículo científico.

4. *FRAMEWORK* PARA LA DETECCIÓN DE DDOS-BOTNET

El *framework* propone un sistema de detección de ataques DDoS en redes SDN. La presente solución toma la arquitectura SDN como punto de partida y localiza los elementos que provocan un ataque DDoS basado en botnets (zombies). Para detectar y evitar el ataque, introducimos el módulo Sistema de Detección que analiza el tráfico que circula en la red (ver Figura 4).

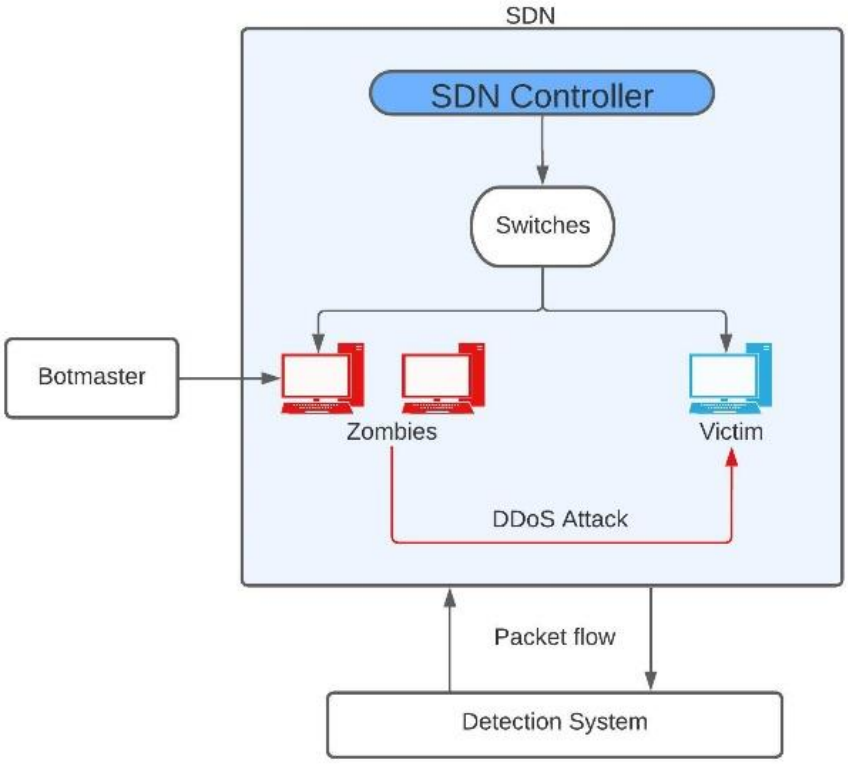

Figura 4. *Framework* para detección de ataques DDoS-botnets en SDNs.

Los componentes principales del *framework* propuesto son:

- **Infraestructura SDN**: La infraestructura SDN contiene los dispositivos de hardware de red, el controlador y los usuarios. Un grupo de ellos se transformará en zombies y será el encargado de ejecutar el ataque DDoS sobre un usuario víctima.
- **Controlador SDN**: Es el encargado de administrar el flujo de datos dentro de la red SDN. Además, el controlador dispone de la información de topología y controla la configuración de cada uno de los switches que forman parte de la infraestructura. El controlador recibe información sobre el origen

y el destino de los paquetes entrantes, ejecuta algoritmos para encontrar la ruta óptima y emite comandos a los conmutadores para reenviar los paquetes a su destino.

- **Botmaster**: Se encarga de gestionar usuarios zombies y dar órdenes para ejecutar el ataque DDoS. El botmaster puede estar dentro o fuera de la red SDN y los zombis se controlan mediante la ejecución de scripts maliciosos en las computadoras.
- **Sistema de detección**: Se encarga de analizar los paquetes recibidos por el usuario e identificar cualquier anomalía que puedan tener. La identificación de estas anomalías se realiza mediante reglas previamente configuradas en el sistema de detección.
- **Ataques DDoS**: Es un conjunto de hosts conectados a la red SDN y controlados por el botmaster que realizan diferentes ataques dirigidos a un usuario dentro de la red. Cabe mencionar que la red SDN necesita la integración entre los planos de datos y control, por lo que el ataque a cualquiera de estos elementos también influye en el desempeño del controlador SDN.

La Figura 5 muestra la secuencia de eventos en la que trabaja el framework. La infraestructura y el controlador SDN son los primeros elementos que se activan para proporcionar la red de comunicación. En este punto, los usuarios reciben normalmente el servicio de red. El usuario monta y activa el módulo de detección de DDoS. De esta forma, el sistema analiza todo el tráfico de usuarios y envía mensajes de alerta si se produce un posible ataque DDoS.

Por su parte, el atacante selecciona a la víctima e inicia el procedimiento de ataque. El botmaster se monta mediante un script para infectar dispositivos y convertirlos en zombies. Una vez que los dispositivos están infectados, el botmaster da la orden de obligar a los zombies a comenzar a sobrecargar los recursos de red de la víctima. El módulo de detección analiza el tráfico en tiempo real, detecta el ataque y genera mensajes de alerta en el usuario víctima.

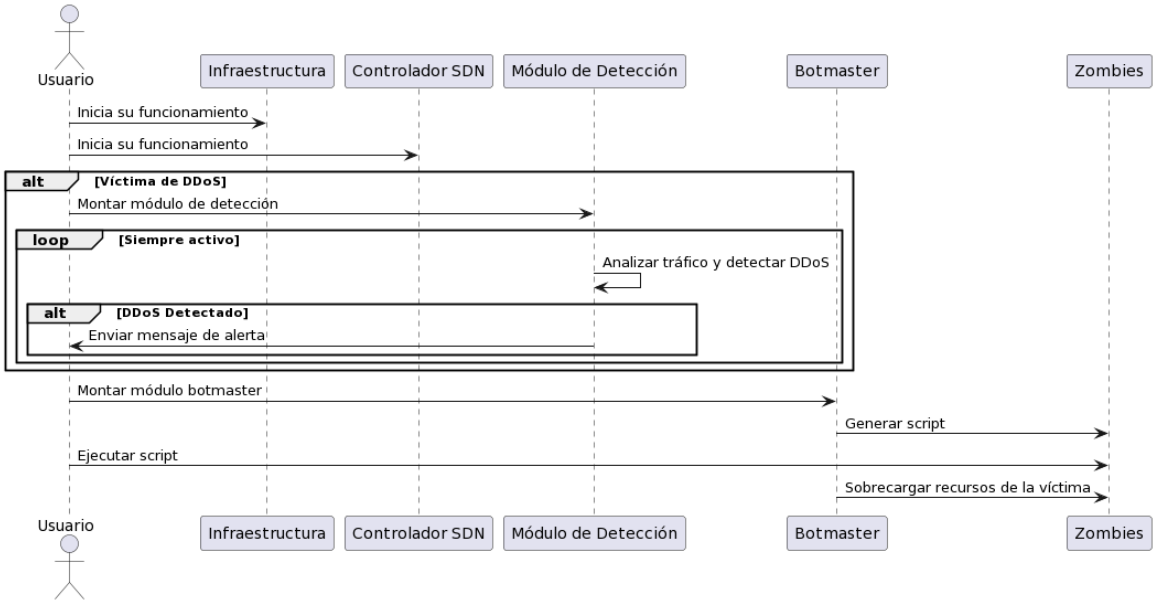

Figura 5. Diagrama de secuencias del framework propuesto.

5. IMPLEMENTACIÓN

La Figura 6 arquitectura propuesta se implementa utilizando las herramientas descritas en la. De igual forma, la Tabla 2 describe los detalles técnicos del software utilizado.

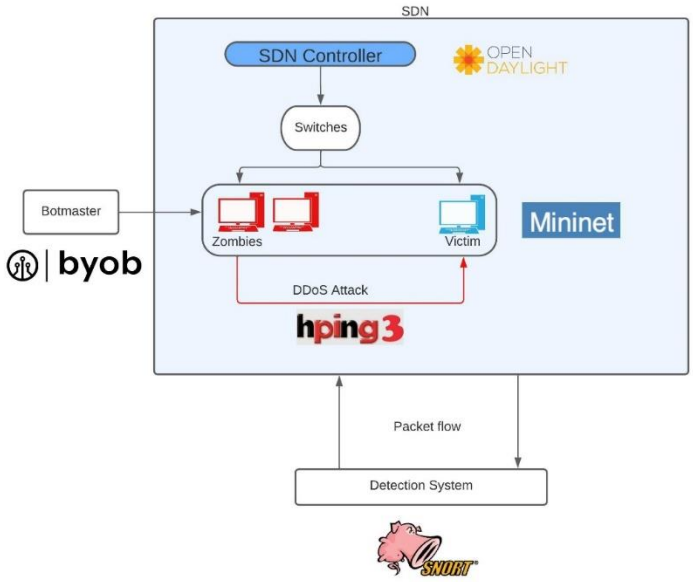

Figura 6. Implementación del framework propuesto

La red SDN se crea a partir de un script en Mininet; la red contiene un controlador SDN, un switch principal, dos switches de distribución y cuatro switches de acceso que conectan a veinte usuarios. Uno de estos usuarios se convertirá en la víctima del ataque DDoS en las dos versiones propuestas (ICMP Flood y SYN Flood).

Tabla 2. Detalles técnicos de herramientas open-source

Módulo	Herramienta	Versión	Detalles
Controlador SDN	OpenDaylight [30]	0.13.1	Código abierto Maneja interfaces OpenFlow Posee GUI [31]
Botmaster	BYOB [32]	2.1	Código abierto. Proporciona una interfaz de línea de comandos (CLI). Permite la personalización de las botnet.

Ataque DDoS	Hping3 [33]	3	Código abierto. Generación de paquetes personalizados y de flujos de ataque. Servicio de escaneo de puertos.
Infraestructura SDN	Mininet [34]	2.3.0d7	Código abierto. Emulación de redes realísticas Integrado con controladores OpenFlow
Detección DDoS	SNORT [35]	3.1.64.0	Código abierto Soporte para múltiples plataformas. Detección de amenazas.

El sistema de ataque de botnet se crea a partir del software BYOB que despliega un servidor o botmaster y se encarga de desarrollar un script en Python. Cuando se ejecuta el script en los hosts de la red SDN, estos se convierten en parte de la botnet.

El script tiene opciones de ser creado como un archivo ejecutable o como un archivo de código de Python. Es un archivo fuertemente comprimido e ininteligible. El archivo se propaga dentro de la máquina a ser convertida en zombie y su trabajo es el de localizar la siguiente parte del código; es decir, descarga, desencripta y ejecuta otro archivo.

La siguiente etapa tiene que ver con el nuevo archivo descargado. Este archivo solo existe en la memoria de la máquina a convertirse en zombie. Su trabajo es analizar ciertas características para evitar que este código que se está ejecutando en memoria sea analizado. Si todas las comprobaciones se superan, se descarga otro archivo encriptado que contiene el código principal o *payload*. Este código principal se desencripta y se ejecuta.

La última etapa tiene que ver con el último archivo desencriptado. El mismo se carga en memoria y contiene toda la funcionalidad principal para controlar la

máquina objetivo. Este archivo importa paquetes importantes de Python, módulos de post-explotación desde el servidor (botmaster) y realiza un *Reverse TCP Shell* [36]. Cuando se inicializa, el botmaster toma el control total del usuario que forma parte de la SDN sin el conocimiento del usuario. El mismo script generado por el botmaster se usa para controlar cualquier número de usuarios que forman parte de la red. En nuestro proyecto, se eligió la opción de script de Python y se ejecutó en cada host a través de su terminal.

Los zombies lanzarán un ataque DDoS en un solo host SDN. El sistema de detección de ataques DDoS se implementa mediante la herramienta SNORT [35]. Una vez instalado en la máquina virtual, se crea una lista de reglas para detectar ataques de inundación ICMP y SYN en poco tiempo después de que se haya ejecutado el ataque. Escribir reglas de SNORT efectivas requiere una buena comprensión de las amenazas de seguridad y la capacidad de analizar el tráfico de red para identificar posibles patrones de ataque.

SNORT utiliza un lenguaje de descripción de reglas sencillo y ligero que es flexible y potente. Las reglas de SNORT deben contener completamente en una sola línea. El analizador de reglas no sabe cómo manejar las reglas en varias líneas [37].

Estas reglas se dividen en dos secciones lógicas: el encabezado de la regla y las opciones de la regla. El encabezado de la regla contiene la acción de la regla, el protocolo, el origen, las direcciones IP de destino y las máscaras de red, y la información de los puertos de origen y destino. La sección de opciones de la regla contiene mensajes de alerta e información sobre qué partes del paquete se deben inspeccionar para determinar si se debe tomar la acción de la regla [37]. La Tabla 3 muestra cada una de las partes que debe contener una regla en SNORT. En el Anexo II se describen algunas características tomadas en cuenta para la creación de reglas para este proyecto.

Tabla 3. Partes para la creación de una regla en SNORT.

Parte de la Regla	Descripción
Acción	Especifica la acción que se tomará cuando se cumpla la condición de la regla.
Protocolo	Indica el protocolo de red al que se aplica la regla (TCP, UDP, ICMP, etc.).
IP de origen	Especifica la dirección IP o rango de direcciones IP de origen del tráfico.
Puerto de origen	Indica el número de puerto o rango de puertos del tráfico de origen.
Operador	Define la operación de la regla (por ejemplo, contenido, palabra clave, etc.).
Contenido	Especifica el patrón o cadena que se busca en los datos del paquete.
Dirección	Define la dirección en la que se aplica la operación (->: De origen a destino, <>: Ambos).
IP de destino	Especifica la dirección IP o rango de direcciones IP de destino del tráfico.
Puerto de destino	Indica el número de puerto o rango de puertos del tráfico de destino.

Bajo estas indicaciones, un ejemplo de la estructura de una regla elaborada en SNORT se muestra en la Figura 7:

```
alert icmp any any -> $HOME_NET any (msg:"Ataque DDoS ICMP flood detectado";
dsize:>500; sid:100002;)
```

Figura 7. Regla SNORT de detección de ataque ICMP Flood.

6. EXPERIMENTACIÓN

El *framework* se evalúa utilizando la topología mencionada en la subsección anterior. Aquí se presenta en detalle los dos escenarios de ataque implementados y el uso de los elementos que generan el ataque DDoS en la SDN.

En el primer escenario, los hosts convertidos en zombies se encargan de ejecutar el ataque ICMP Flood. En el segundo escenario, los zombies se encargan de ejecutar el ataque SYN Flood. Ambos ataques comienzan de forma inesperada, apuntando a un usuario de la red que tiene la detección de ataques SNORT permanentemente activa.

6.1. Topología

La red SDN se creó en Mininet con un script hecho en Python. Se creará una red que con espacio para un controlador SDN, un switch principal, dos switches de distribución y cuatro switches de acceso que se comunican con el protocolo OpenFlow. La red también contiene 20 usuarios, y cada switch de acceso estará conectado a cinco de ellos. En el Anexo III se muestra el proceso paso a paso de la creación de la topología de red SDN.

Uno de estos usuarios se convertirá en la víctima del ataque DDoS en las dos versiones propuestas (ICMP Flood y SYN Flood), y cinco de estos usuarios se convertirán en zombies al ejecutar el script creado por el servidor de la botnet (botmaster). El proceso de ataque de los zombies hacia la máquina objetivo se explica brevemente en el Anexo I. La topología de la red SDN creada para la ejecución de los experimentos se muestra en la Figura 8.

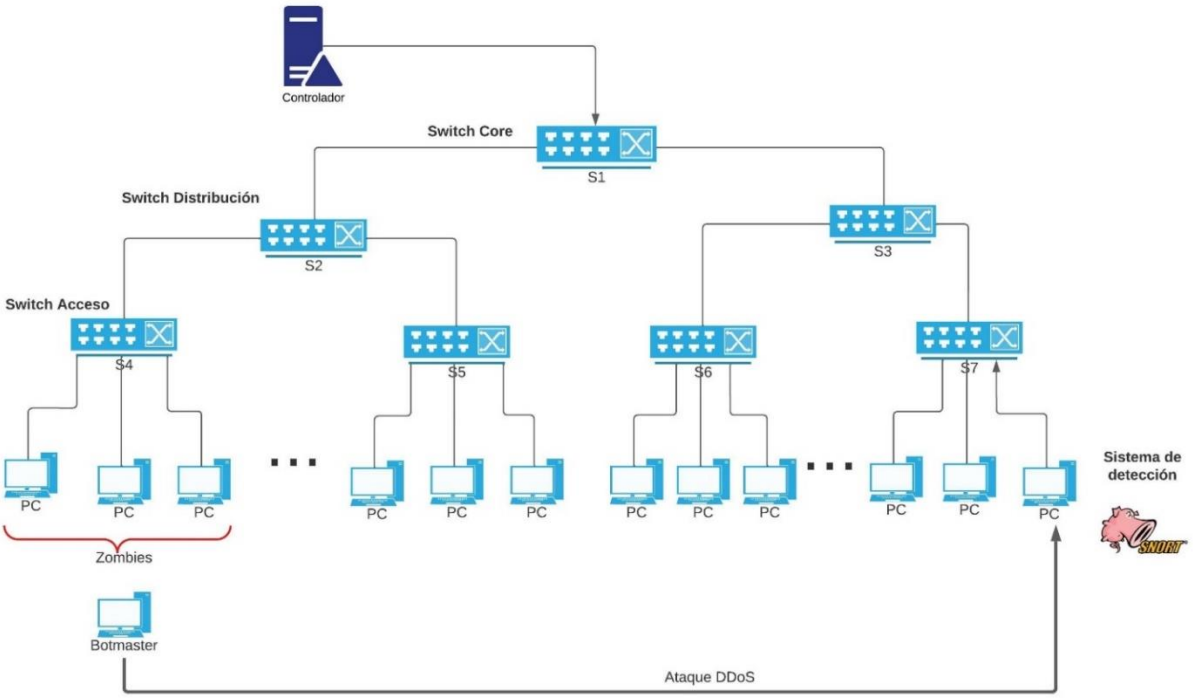

Figura 8. Topología de red SDN.

6.2. Experimentos

La Tabla 4 detalla las características de cada uno de los ataques. Estos ataques fueron seleccionados debido a que se consideran dos de los más importantes debido a su efectividad en agotar los recursos del servidor objetivo y afectar la disponibilidad de los servicios en línea.

Tabla 4. Tipos de ataques para cada escenario.

Ataque	Fuente	Objetivo	Detalles
ICMP Flood	Zombies controlados por el Botmaster (Host 1 al 5).	Host Víctima (Host 20)	• Ataque ICMP Flood • Tamaño de paquete 1500 bytes.
SYN Flood			• Ataque SYN/TCP. • Tamaño de ventana 64. • Tamaño de paquete 100000 bytes.

7. RESULTADOS Y DISCUSIÓN

En esta sección se presentan y analizan los resultados obtenidos producto de los experimentos indicados anteriormente.

7.1 Resultados del ataque ICMP Flood

En el ataque ICMP Flood, se incrementa el consumo de recursos de la máquina víctima, tanto en el uso de núcleos de procesamiento como en el consumo de memoria. La Figura 9 muestra el rendimiento antes y después del ataque DDoS.

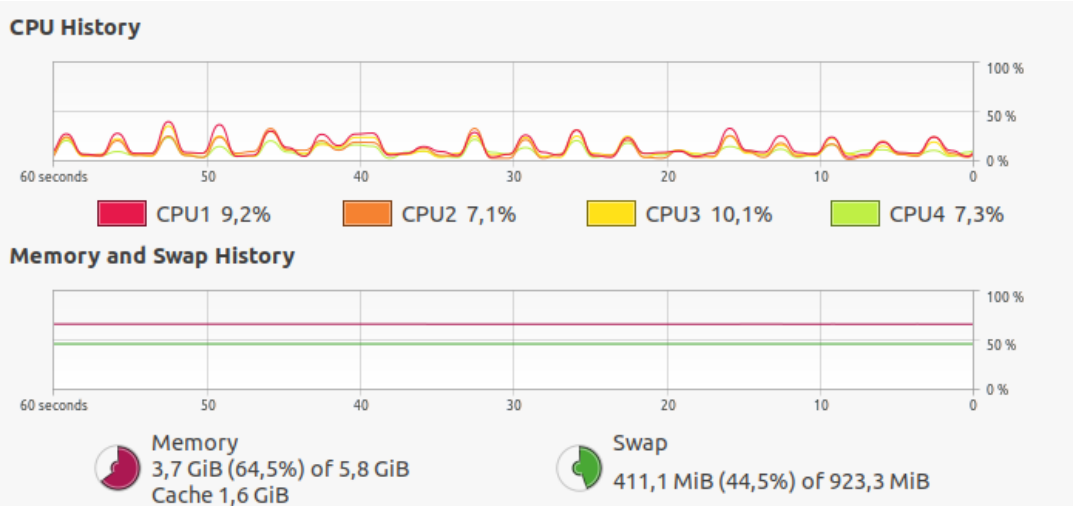

Figura 9. Consumo de recursos antes y después de un ataque ICMP Flood.

Aunque el consumo de recursos aumenta considerablemente, no satura la máquina de la víctima (un consumo del 30% de promedio en los 4 núcleos). Aun así, el bombardeo de paquetes ICMP hacia la máquina de destino se detecta fácilmente a través de SNORT. El tiempo entre el inicio del ataque y su detección se muestra en la Figura 10. En la Figura se muestra que el rango de tiempo entre la ejecución del ataque y su detección es muy corto. El ataque de inundación ICMP comenzó a las 14:40:00 segundos. El botmaster envió la orden para que los zombis atacaran a la víctima y SNORT detectó y envió mensajes de alerta de un posible ataque DDoS 4 segundos después de que comenzó. Por tanto, este *framework* es una solución eficiente a la hora de detectar este tipo de ataques.

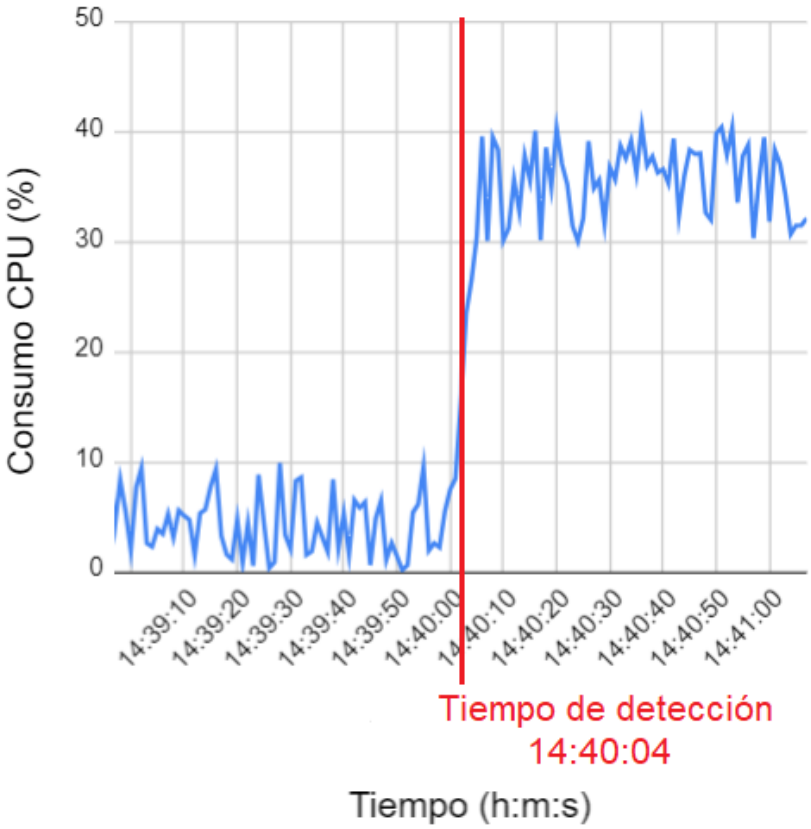

Figura 10. Cronología antes y después del ataque de inundación ICMP (con identificación del punto en el que se realizó la detección).

7.2 Resultados del ataque SYN Flood

El ataque SYN Flood tuvo un impacto significativo en la máquina víctima. La Figura 11 muestra el rendimiento durante el ataque DDoS. En el ataque de inundación SYN, a diferencia del ataque de inundación ICMP, el consumo de recursos de la máquina víctima se ha incrementado considerablemente. El consumo de núcleos en la máquina de la víctima era tan alto que la mayor parte del tiempo, tres de los cuatro núcleos estaban por encima del 80 % de consumo. El consumo de memoria también aumentó, del 15 % al 70 %.

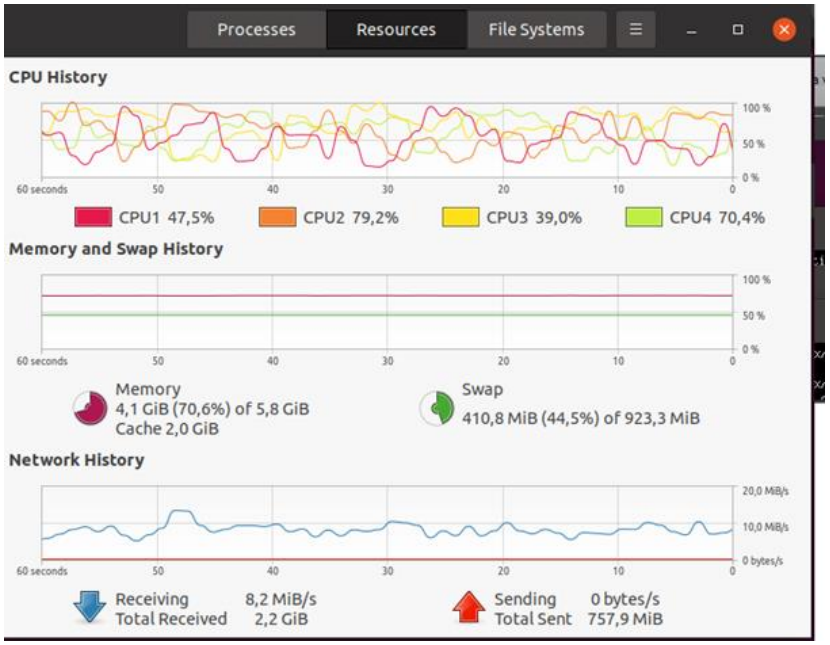

Figura 11. Consumo de recursos antes y después de un ataque SYN Flood.

En cuanto a la detección mediante la herramienta SNORT, mostró resultados similares al ataque ICMP Flood. La detección del ataque se hizo casi de inmediato. El ataque de inundación SYN comenzó a las 14:15:05 segundos. El botmaster envió la orden para que los zombis atacaran a la víctima y SNORT detectó y envió mensajes de alerta de un posible ataque DDoS 3 segundos después de que comenzó. La Figura 12 muestra la diferencia de tiempo entre el inicio del ataque DDoS y su detección.

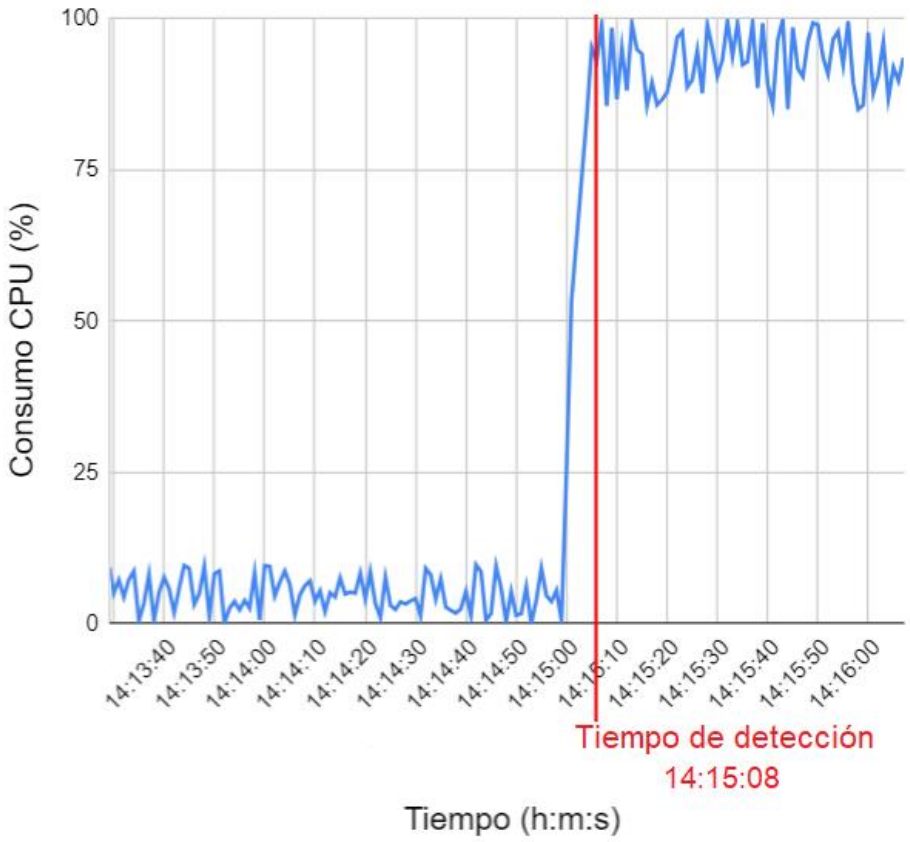

Figura 12. Cronología antes y después del ataque de inundación ICMP (con identificación del punto en el que se realizó la detección).

Aunque el ataque se está ejecutando y consumiendo la mayoría de los recursos de la víctima, el sistema es capaz de detectar el ataque a través de SNORT y envía una gran cantidad de mensajes de alerta justo después de comenzar el ataque. La Tabla 5 resume los resultados obtenidos.

Tabla 5. Resultados de detección de ataques DDoS

Ataques DDoS	Tiempo de inicio del ataque	Tiempo de detección del ataque	Consumo de CPU
ICMP Flood	14:40:00	14:40:04	40%
SYN Flood	14:15:05	14:15:08	90%

7.3 Discusión

Nuestro trabajo se centra en la ejecución de ataques DDoS a través de dos tipos de ataques: ICMP Flood y SYN Flood, ya que son dos de los ataques más comunes

utilizados en las redes tradicionales [38]. Durante la ejecución de estos dos ataques, se observó que el ataque de inundación SYN se detectó un segundo más rápido que el ataque de inundación ICMP. Si se cambia alguna de las configuraciones en las reglas, la diferencia en el tiempo de detección para estos mismos tipos de ataques no cambia con respecto a los resultados que se muestran en esta sección.

El consumo de recursos fue diferente al ejecutar estos dos tipos de ataques. El impacto de los recursos en el ataque de inundación ICMP fue un 50 % menor en comparación con el ataque de inundación SYN. Esto se debe a que el bombardeo de paquetes TCP con encabezados SYN fue mucho mayor que el bombardeo de paquetes ICMP. La diferencia en el consumo de recursos se puede ver en la Figura 9 y Figura 11.

En el ataque SYN Flood, es necesario destacar el mayor consumo de recursos en comparación con los ataques ICMP Flood. Estos resultados también demostraron la capacidad de detección de la herramienta SNORT ya que, segundos después de realizado el ataque, la herramienta identificó y notificó los ataques DDoS en los registros.

Los tiempos de respuesta de detección fueron ligeramente diferentes. Los resultados mostraron que el tiempo de detección del ataque de inundación SYN fue un segundo más rápido que el tiempo de detección del ataque de inundación ICMP. Sin embargo, en ambos casos, la detección ha sido exitosa. Incluso cuando el ataque de inundación SYN resultó en un alto consumo de recursos para la víctima, la víctima logró detectar el ataque casi al instante.

Todo el trabajo realizado en este proyecto referente a la Detección De Ataques de Denegación de Servicio Activados Mediante Botnets en Redes Definidas por Software se ha visto materializado en la escritura del presente artículo científico el cual se encuentra en el Anexo IV.

.

8. CONCLUSIONES Y TRABAJO FUTURO

La arquitectura presentada proporciona un *framework* comprensivo para detectar ataques DDoS generados por botnets en una red SDN. Permite el análisis, la prueba y la comparación del comportamiento de una red SDN antes y después de los ataques DDoS.

La implementación del marco demostró con éxito la capacidad de detección, ya que identificó y registró rápidamente los ataques DDoS segundos después de que se llevaron a cabo. Los tiempos de respuesta de detección fueron ligeramente diferentes entre los ataques de inundación SYN y de inundación ICMP, pero ambos ataques se detectaron con éxito.

Los experimentos realizados en la red SDN mostraron que el ataque SYN Flood tuvo un impacto más significativo en el correcto funcionamiento de la red en comparación con el ataque de ICMP Flood. El consumo de recursos fue mayor en el ataque de SYN Flood debido al bombardeo de paquetes TCP con cabeceras SYN. Sin embargo, el anfitrión de la víctima logró detectar el ataque casi al instante, incluso con un alto consumo de recursos.

La integración del *framework* de SNORT para detectar ataques DDoS en redes SDN proporciona una ventaja sobre las redes tradicionales. Las redes tradicionales enfrentan desafíos para escalar y garantizar bajas tasas de falsas alarmas, que las redes SDN pueden superar con el marco propuesto.

Los desafíos para el trabajo futuro incluyen la implementación de un mecanismo de mitigación de ataques DDoS dentro del marco propuesto en este proyecto. Esto es necesario para que se puedan tomar acciones además de la detección, y el beneficio sea mayor. De igual forma, se recomienda la ejecución de experimentos con un número más significativo de usuarios y, por tanto, con un mayor número de bots. La extensión de otros tipos de ataques adicionales a ICMP Flood y SYN Flood también se considera para trabajos futuros.

REFERENCIAS BIBLIOGRÁFICAS

[1] N. Z. Bawany, J. A. Shamsi, y K. Salah, “DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions”, *Arab J Sci Eng*, vol. 42, núm. 2, pp. 425–441, 2017, doi: 10.1007/s13369-017-2414-5.

[2] “Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper - Cisco”. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (consultado el 23 de diciembre de 2021).

[3] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, y S. Uhlig, “Software-defined networking: A comprehensive survey”, *Proceedings of the IEEE*, vol. 103, núm. 1, pp. 14–76, 2015, doi: 10.1109/JPROC.2014.2371999.

[4] S. Lim, J. Ha, H. Kim, Y. Kim, y S. Yang, “A SDN-oriented DDoS blocking scheme for botnet-based attacks”, *International Conference on Ubiquitous and Future Networks, ICUFN*, pp. 63–68, 2014, doi: 10.1109/ICUFN.2014.6876752.

[5] D. Nikolov, I. Kordev, y S. Stefanova, “Concept for network intrusion detection system based on recurrent neural network classifier”, *2018 IEEE 27th International Scientific Conference Electronics, ET 2018 - Proceedings*, pp. 1–4, 2018, doi: 10.1109/ET.2018.8549584.

[6] N. Hoque, D. K. Bhattacharyya, y J. K. Kalita, “Botnet in DDoS Attacks: Trends and Challenges”, *IEEE Communications Surveys and Tutorials*, vol. 17, núm. 4, pp. 2242–2270, 2015, doi: 10.1109/COMST.2015.2457491.

[7] P. Manso, J. Moura, y C. Serrão, “SDN-based intrusion detection system for early detection and mitigation of DDoS attacks”, *Information (Switzerland)*, vol. 10, núm. 3, pp. 1–17, 2019, doi: 10.3390/info10030106.

[8] G. Cusack, O. Michel, y E. Keller, “Machine learning-based detection of ransomware using SDN”, *SDN-NFVSec 2018 - Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks and Network Function Virtualization, Co-located with CODASPY 2018*, vol. 2018-Janua, pp. 1–6, 2018, doi: 10.1145/3180465.3180467.

[9] O. Katz y J. Black, “Akamai DNS Traffic Insights Threat Report | akamai”, *Akamai DNS Traffic Threat Report*. junio de 2022. [En línea]. Disponible en: https://www.akamai.com/resources/research-paper/akamai-dns-traffic-insights-threat-report

[10] D. Nashat y F. A. Hussain, “Multifractal detrended fluctuation analysis based detection for SYN flooding attack”, *Comput Secur*, vol. 107, ago. 2021, doi: 10.1016/j.cose.2021.102315.

[11] V. Chauhan y P. Saini, “ICMP flood attacks: A vulnerability analysis”, en *Advances in Intelligent Systems and Computing*, Springer Verlag, 2018, pp. 261–268. doi: 10.1007/978-981-10-8536-9_26.

[12] C. Douligeris y A. Mitrokotsa, “DDoS attacks and defense mechanisms: Classification and state-of-the-art”, *Computer Networks*, vol. 44, núm. 5, pp. 643–666, 2004, doi: 10.1016/j.comnet.2003.10.003.

[13] S. Dong y M. Sarem, “DDoS Attack Detection Method Based on Improved KNN with the Degree of DDoS Attack in Software-Defined Networks”, *IEEE Access*, vol. 8, pp. 5039–5048, 2020, doi: 10.1109/ACCESS.2019.2963077.

[14] “What is an ICMP Flood Attack? | NETSCOUT”. https://www.netscout.com/what-is-ddos/icmp-flood (consultado el 4 de julio de 2023).

[15] Y. Lu y M. Wang, “An easy defense mechanism against botnet-based DDoS flooding attack originated in SDN environment using sFlow”, *ACM International Conference Proceeding Series*, vol. 15-17-June, pp. 14–20, 2016, doi: 10.1145/2935663.2935674.

[16] S. Badotra y S. N. Panda, “SNORT based early DDoS detection system using Opendaylight and open networking operating system in software defined networking”, *Cluster Comput*, vol. 24, núm. 1, pp. 501–513, 2021, doi: 10.1007/s10586-020-03133-y.

[17] “OpenDaylight”. https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2018_2/opendaylight/ (consultado el 3 de julio de 2023).

[18] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, y V. Maglaris, “Combining OpenFlow and sFlow for an effective and scalable anomaly

detection and mitigation mechanism on SDN environments", *Computer Networks*, vol. 62, pp. 122–136, 2014, doi: 10.1016/j.bjp.2013.10.014.

[19] R. Wang, Z. Jia, y L. Ju, "An entropy-based distributed DDoS detection mechanism in software-defined networking", *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*, vol. 1, pp. 310–317, 2015, doi: 10.1109/Trustcom.2015.389.

[20] S. A. Mehdi, J. Khalid, y S. A. Khayam, "Revisiting Traffic Anomaly Detection Using Software Defined Networking.pdf", *International workshop on recent advances in intrusion detection*, pp. 161–180, 2011.

[21] S. Dotcenko, A. Vladyko, y I. Letenko, "A fuzzy logic-based information security management for software-defined networks", *International Conference on Advanced Communication Technology, ICACT*, pp. 167–171, 2014, doi: 10.1109/ICACT.2014.6778942.

[22] C. Chung, S. Member, P. Khatkar, y T. Xing, "NICE : Network Intrusion Detection and Countermeasure", *IEEE Transactions on Dependable Secure Computting*, vol. 10, núm. 4, pp. 1–14, 2013, [En línea]. Disponible en: http://dblp.uni-trier.de/db/journals/tdsc/tdsc10.html#ChungKXLH13

[23] R. Jin y B. Wang, "Malware detection for mobile devices using software-defined networking", *Proceedings - 2013 2nd GENI Research and Educational Experiment Workshop, GREE 2013*, pp. 81–88, 2013, doi: 10.1109/GREE.2013.24.

[24] S. E. Schechter, J. Jung, y A. W. Berger, "Fast detection of scanning worm infections", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3224, pp. 59–81, 2004, doi: 10.1007/978-3-540-30143-1_4.

[25] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, y M. Tyson, "FRESCO Modular Composable Security Services for Software-Defined Networks", *20th Annual Network & Distributed System Security Symposium. Ndss*, vol. 2, núm. February, p. 16, 2013.

[26] T. Chin, X. Mountrouidou, X. Li, y K. Xiong, "Selective packet inspection to detect DoS flooding using software defined networking (SDN)", *Proceedings - 2015 IEEE 35th International Conference on Distributed Computing Systems*

Workshops, ICDCSW 2015, pp. 95–99, 2015, doi: 10.1109/ICDCSW.2015.27.

[27] T. Xing, D. Huang, L. Xu, C. J. Chung, y P. Khatkar, “SnortFlow: A OpenFlow-based intrusion prevention system in cloud environment”, *Proceedings - 2013 2nd GENI Research and Educational Experiment Workshop, GREE 2013*, pp. 89–92, 2013, doi: 10.1109/GREE.2013.25.

[28] M. Kumar y A. Bhandari, “DDoS Detection in ONOS SDN Controller Using Snort”, en *ICT with Intelligent Applications*, J. Choudrie, P. Mahalle, T. Perumal, y A. Joshi, Eds., Singapore: Springer Nature Singapore, 2023, pp. 155–164.

[29] K. Saurabh, T. Kumar, U. Singh, O. P. Vyas, y R. Khondoker, “NFDLM: A Lightweight Network Flow based Deep Learning Model for DDoS Attack Detection in IoT Domains”, *2022 IEEE World AI IoT Congress, AIIoT 2022*, núm. Dl, pp. 736–742, 2022, doi: 10.1109/AIIoT54504.2022.9817297.

[30] “OpenDaylight Downloads — OpenDaylight Documentation Sulfur documentation”. https://docs.opendaylight.org/en/stable-sulfur/downloads.html (consultado el 4 de julio de 2023).

[31] C. D. Cajas y D. O. Budanov, “SDN Applications and Plugins in the OpenDaylight Controller”, *Proceedings of the 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, EIConRus 2020*, pp. 9–13, 2020, doi: 10.1109/EIConRus49466.2020.9039054.

[32] “Guide”. https://byob.dev/guide (consultado el 4 de julio de 2023).

[33] “hping3 | Kali Linux Tools”. https://www.kali.org/tools/hping3/ (consultado el 4 de julio de 2023).

[34] “Download/Get Started With Mininet - Mininet”. http://mininet.org/download/ (consultado el 4 de julio de 2023).

[35] “Snort - Network Intrusion Detection & Prevention System”. https://www.snort.org/ (consultado el 4 de julio de 2023).

[36] “How It Works”. https://byob.dev/docs (consultado el 19 de julio de 2023).

[37] “Writing Snort Rules”. https://paginas.fe.up.pt/~mgi98020/pgr/writing_snort_rules.htm (consultado el 4 de julio de 2023).

[38] T. Mahjabin, Y. Xiao, G. Sun, y W. Jiang, “A survey of distributed denial-of-service attack, prevention, and mitigation techniques”, *Int J Distrib Sens Netw*, vol. 13, núm. 12, 2017, doi: 10.1177/1550147717741463.

ANEXOS

Anexo I Creación del script zombie – Ejecución del ataque por los zombies

BYOB se encarga de elaborar los bloques necesarios para construir la botnet. Para instalarlo, es necesario lo siguiente: primero, el código abierto BYOB, que está disponible en un repositorio en GitHub para que se puedan descargar los archivos necesarios para la instalación (ver Figura A1. 1). A continuación, también es necesario instalar Docker ya que la información requerida para el correcto funcionamiento de BYOB está empaquetada en este software. Finalmente, se necesita Python3 para la generación de scripts zombie (ver Figura A1. 2). El script se creará en el servidor botnet. Supongamos que este script se envía y ejecuta en los usuarios de SDN. En ese caso, el servidor de la botnet tomará el control completo de la operación de dichos usuarios, pudiendo enviar ataques coordinados a un objetivo dentro del mismo SDN (ver Figura A1. 3). Dentro del marco del servidor BYOB, se puede ejecutar cualquier comando y los zombis los replicarán. Esta estructura es ideal para dirigir ataques DDoS. El comando de transmisión hace que los zombis ejecuten cualquier comando colocado después. En este sentido, se utilizará la herramienta Hping3 sobre los zombis para que envíen un único ataque a la víctima.

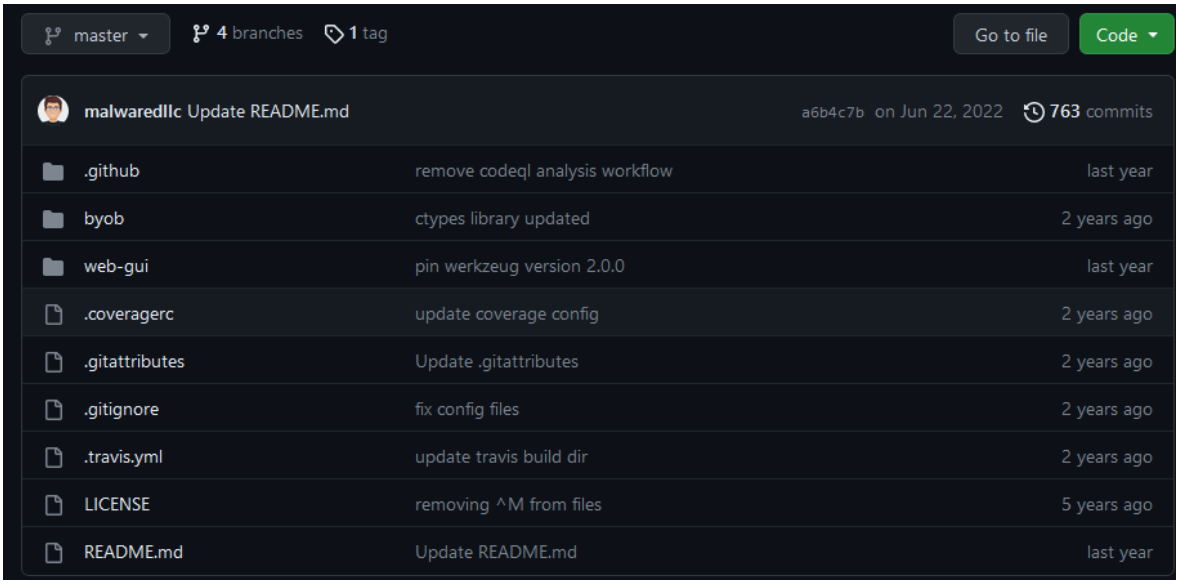

Figura A1. 1. Repositorio de GitHub de BYOB (https://github.com/malwaredllc/byob)

Figura A1. 2. Creación del script para controlar usuarios y convertirlos en zombies.

```
broadcast sudo hping3 -q -n -d 120 -S --flood 10.0.0.20
```

Figura A1. 3. Comando enviado por el botmaster a los zombies para que lo ejecuten

Anexo II – Creación de reglas para la detección mediante SNORT

SNORT se utilizará para la detección. Una de las configuraciones para el correcto funcionamiento de SNORT es la creación de reglas de detección. Cada regla de detección puede ir acompañada de una o varias palabras clave que permitan filtrar mejor los ataques.

La regla debe identificar los ataques de inundación ICMP y verificar los paquetes que la víctima recibe desde cualquier dirección IP a través del protocolo ICMP. La regla tiene un identificador único local como una de las palabras clave de la regla. Junto a esto, hay otra palabra clave (rev) que maneja las revisiones de reglas y descripciones para que se actualicen con la información más reciente. Finalmente, una última palabra clave llamada classtype se encarga de clasificar el tipo de ataque a detectar. SNORT tiene, por defecto, un conjunto de clases de ataque conocidas que se pueden usar para definir la regla de detección correctamente. En este caso, la clase es evento ICMP.

Se usará otra regla para identificar los ataques de inundación SYN, y es responsable de verificar los paquetes del protocolo TCP desde cualquier dirección IP de origen a la víctima. Tiene una palabra clave de identificación única local y una palabra clave de revisión de reglas que se mantiene actualizada. El indicador S se establece en SNORT y está destinado a identificar los bits de encabezado TCP SYN. La palabra clave de flujo es responsable de aplicar las reglas en direcciones específicas de los flujos de tráfico. Se ha configurado como sin estado, ya que es la opción correcta para eventos en los que los paquetes tienen la intención de hacer que las máquinas se bloqueen. El filtro de detección es otra palabra clave encargada de definir una tasa que, si se supera, generará la notificación de posibles ataques DDoS. La combinación de palabras clave conteo y segundos muestra el número máximo de coincidencias de la regla permitidas (contar 70, por ejemplo) en un tiempo determinado (10 segundos, por ejemplo); después de ese umbral, generará una alerta de ataque DDoS de tipo SYN (ver Figura A2. 1).

```
alert icmp any any -> $HOME_NET any (msg:"DDoS ICMP flood"; sid:1000001; rev:1;classtype:icmp-event;)
alert tcp any any -> $HOME_NET any (flags: S;msg:"Posible ataque DDoS SYN"; flow: stateless; detection_filter: track by_src,
count 70, seconds 10;)
```

Figura A2. 1. Reglas para la detección de ataques ICMP Flood y SYN Flood.

Anexo III – Creación de la topología de red con Mininet y OpenDaylight

La red SDN fue creada en Mininet con un script hecho en Python (ver Figura A3. 1). Se creará una red que contenga un controlador SDN, un conmutador central, dos conmutadores de distribución y cuatro conmutadores de acceso que se comuniquen con el protocolo OpenFlow. La red también contiene 20 usuarios, y cada conmutador de acceso se conectará a cinco de ellos. Uno de estos usuarios será víctima del ataque DDoS en las dos versiones propuestas (ICMP Flood y SYN Flood), y cinco de estos usuarios se convertirán en zombis al ejecutar el script creado por el servidor botnet. La figura muestra el comando de ejecución de Mininet para crear la red. El comando especifica que el controlador será remoto, es decir, que OpenDaylight se encargará de la creación y configuración del controlador. Cada host tendrá una dirección IP y MAC única; la topología se personaliza a partir de un script previamente elaborado que crea la estructura de la red.

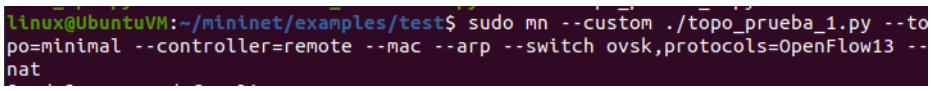

Figura A3. 1. Comando de creación de red SDN en Mininet

OpenDaylight se puede obtener desde el sitio web oficial de la plataforma. Una vez que se descarga esta distribución, se ejecuta, lo que permite crear un controlador que se comunica mediante el protocolo OpenFlow 13. Para visualizar la topología de la red SDN, OpenDaylight también ofrece la opción de montar un servicio HTTP con el puerto de acceso 8181. De manera que si se accede a la dirección localhost:8181/index.html#/topology, se muestra una estructura como la de la Figura A3. 2.

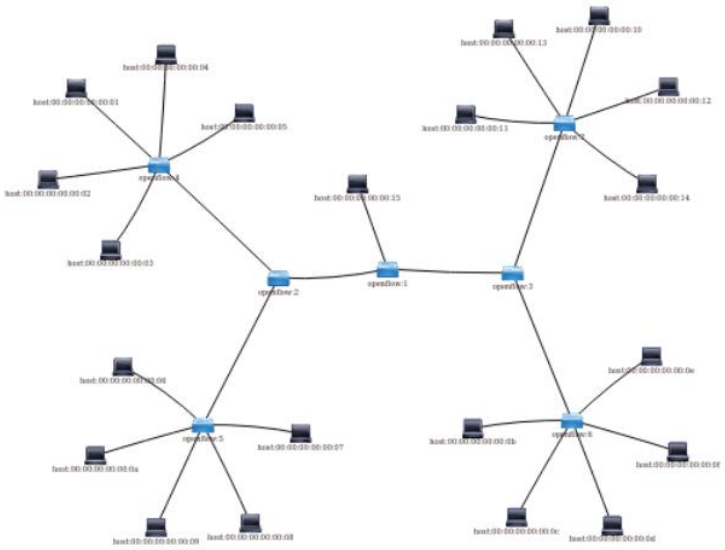

Figura A3. 2. Topología mostrada por OpenDaylight.

Los escenarios de ataque consisten en controlar máquinas zombies para realizar un ataque ICMP flood o SYN flood utilizando la herramienta Hping3. El servidor de botnet montado genera un script para convertir hosts en zombis, haciéndolos parte de la botnet (ver Figura A3. 3, parte a).

Figura A3. 3. Levantamiento del servidor botnet y conexión de los zombies.

Para llevar a cabo los ataques, el servidor enviará un comando a los zombies que ejecutarán el ataque ICMP flood o SYN flood simultáneamente, realizando un ataque DDoS sobre la dirección IP de la víctima. Cinco hosts de la red SDN serán parte de los zombis, ejecutando el script en cada host virtual usando el comando que se muestra en la Figura A3. 3, parte b. Se muestra un mensaje cuando un host se conecta con éxito al servidor de botnet, como en la Figura A3. 3, parte c.

El host de destino establece las reglas de detección de ataques DDoS mediante SNORT. Cuando se activa el comando, analizará los paquetes recibidos para

detectar un posible ataque DDoS. Si lo hace, las alertas se mostrarán en la pantalla y se almacenarán en un archivo de registro SNORT con una descripción precisa de la fecha y hora en que se realizó la detección (ver Figura A3. 4).

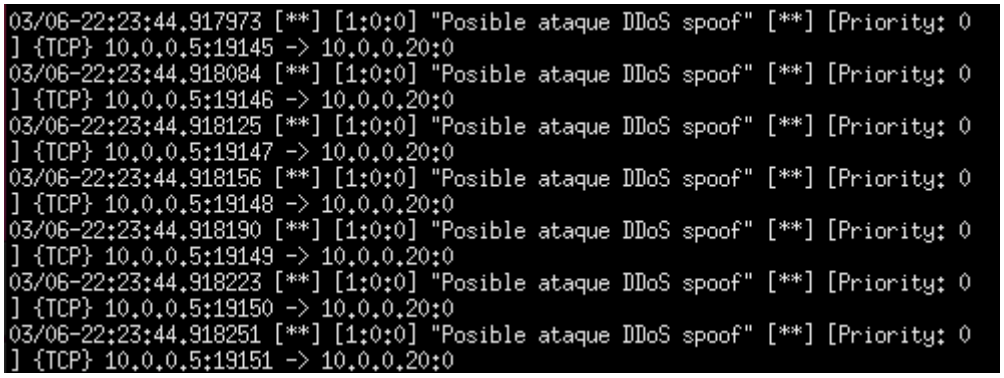

Figura A3. 4. Ejemplo de notificaciones de SNORT ante la presencia de un ataque dirigido al usuario.

Anexo IV – Artículo científico: “Detección de Ataques de Denegación de Servicio Distribuidas Generadas por Botnets en Redes Definidas por Software”

Detection of Distributed Denial of Service Attacks Carried Out by Botnets in Software-Defined Networks

Detección de Ataques de Denegación de Servicio Distribuidas Generadas por Botnets en Redes Definidas por Software

Jaime Tamayo[1], Lorena Isabel Barona López and Ángel Leonardo Valdivieso Caraguay[1]

[1] Departamento de Informática y Ciencias de la Computación (DICC),
Escuela Politécnica Nacional, Ladrón de Guevara E11-253, Quito - Ecuador
{jaime.tamayo01,angel.valdivieso,
lorena.barona}@epn.edu.ec

Abstract. Recent years witnessed a surge in network traffic due to the emergence of new online services, causing periodic saturation and complexity problems. Additionally, the growing number of IoT devices further compounds the problem. Software Defined Network (SDN) is a new architecture which offers innovative advantages that help to reduce saturation problems. Despite its benefits, SDNs not only can be affected by traditional attacks but also introduce new security challenges. In this context, Distributed Denial of Service (DDoS) is one of the most important attacks that can damage an SDN network's normal operation. Furthermore, if these attacks are executed using botnets, they can use thousands of compromised devices to disrupt critical online services. This paper proposes a framework for detecting DDoS attacks generated by a group of botnets in an SDN network. The framework is implemented using open-source tools such as Mininet and OpenDaylight and tested in a centralized network topology using BYOB and SNORT. The results demonstrate real-time attack identification by implementing an intrusion detection mechanism in the victim client. Our proposed solution offers quick and effective detection of DDoS attacks in SDN networks. The framework can successfully differentiate the type of attack with high accuracy in a short time.

Abstract. En los últimos años, se ha observado un aumento exponencial en el tráfico de red debido a la aparición de nuevos servicios online, lo que provoca problemas de saturación y complejidad. El creciente número de dispositivos IoT agrava más el problema. Las Redes Definidas por Software (SDN) son una nueva arquitectura que ofrece ventajas innovadoras que ayudan a reducir los problemas de saturación, pero también presentan nuevos desafíos de seguridad. En este sentido, los ataques de Denegación de Servicio Distribuido (DDoS por sus siglas en inglés) representan una amenaza significativa para las SDN. Además, si estos ataques se ejecutan mediante botnets, pueden aprovechar el poder de miles de dispositivos

comprometidos para interrumpir servicios en línea críticos. Este artículo propone un framework para detectar ataques DDoS generados por un grupo de botnets en una red SDN. El framework se implementa utilizando herramientas de código abierto como Mininet, OpenDaylight y se prueba en una topología de red centralizada mediante BYOB y SNORT. Los resultados de los experimentos demuestran que el sistema puede identificar rápidamente un ataque en tiempo real al implementar un mecanismo de detección de intrusos en el cliente víctima.

Keywords: Seguridad de la información · Ataques DDoS · Redes Definidas por Software · Botnets · Sistema de detección de intrusos · Mininet · SNORT · Byob

Keywords: Information security · Distributed denial of service attacks · Software-defined networks · Botnets · Intrusion detection system· Mininet · SNORT · Byob.

1 Introduction

Software-defined networks (SDNs) offer a novel network model that differs from traditional networks and provides several advantages, such as increased flexibility, scalability, and centralized management. Communication networks shift from a proprietary infrastructure to a more flexible, open, and programmable one [1]. The SDN architecture separates de data and control planes in network devices and centralizes the control logic of the different devices (switches), which only forward the traffic. That is, the controller centrally makes the decisions on the data paths in the network [2]. The controller can be programmed according to the particular needs of each user. On its part, the switches follow the instructions provided by the controller. In addition, SDNs are more scalable than traditional networks [3].

The volume of traffic circulating on public and private networks has increased thanks to the emergence of new online services. For this reason, networks periodically suffer from significant saturation and complexity problems [4]. Traditional network environments are not self-configuring and, therefore, cannot adapt appropriately when new loads appear on the network. Additionally, the flexibility of network growth is reduced by presenting a vertical type of integration [3]. For conventional routers, it is effort and time-consuming to modify their behavior. In this case, administrators have to configure each device individually to change high-level network policies, often using vendor-specific commands. On the other hand, the configuration of an SDN network is more dynamic and efficient. The existence of a central controller in an SDN network makes the routing and controlling task much easier than in a traditional network [5].

Security is expected to be an important application area for SDN. Because of the SDN architecture is innovative, it brings new challenges to network security. Security problems on SDN networks can be divided into two types: attacks inherited from traditional networks and new attacks aimed explicitly at SDN networks. In this regard, one of the most recurrent attacks is the Denial of Service (DoS), which causes damage to both the users and the service provider. This type of attack can affect network functionality and result in financial and prestige losses.

DoS attacks are a problem for the proper functioning of a network since their main objective is disrupting services by limiting access to a machine or service [6]. Furthermore, there is a variation of DoS which can increase the damage on the network. An attack is considered a Distributed Denial of Service (DDoS) when the DoS attack is coordinated. The attacker identifies vulnerabilities and installs malware on multiple machines to control them. After, the controlled machines become a botnet and are commanded by a botmaster [7]. The botmaster can execute malicious actions on the infected devices [8]. If the number of compromised machines is enormous, it can take down the service of an entire web server in a very short time. For example, in February 2022, a distributed denial of service (DDoS) attack was launched against the websites of banks and the Ministry of Defense of Ukraine, after which many countries point to Russia as responsible showing that cyberspace is playing an important role in the conflict between Russia and Ukraine [9].

Figure 1 shows the classification of DDoS attacks, from which flood and amplification attacks will be selected for this paper. These attacks were selected as they

are considered two of the most important attacks due to their effectiveness in exhausting the resources of the target server and affecting the availability of online services [10]. These attacks have been extensively documented by security organizations and have been the subject of numerous reports and analysis in the cybersecurity community [10].

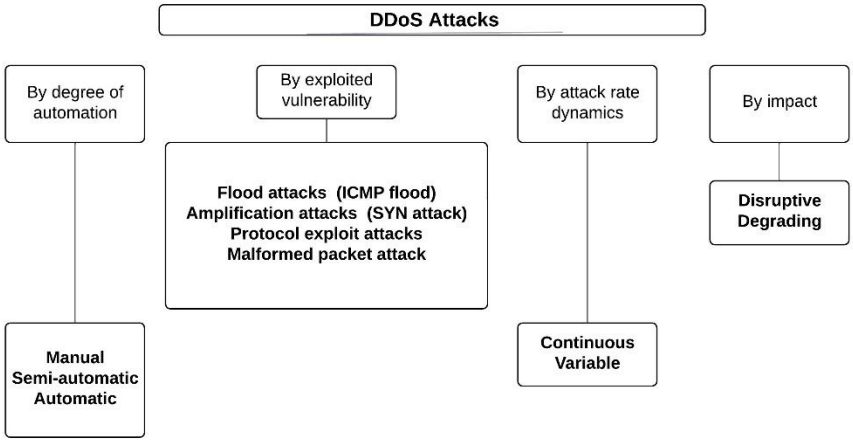

Figure 1. Classification of DDoS attacks

If we refer to DDoS attacks on SDN networks, they are generated by creating many flows that flood the bandwidth of the control plane, saturating the switches and the controller of an SDN network. When a controller gets compromised and experiences a DDoS attack, it can cause the entire network to fail since the controller is in charge of implementing the network logic and managing the applications and switches.

For all the above, our proposed framework aims for early detection of DDoS attacks in SDNs. The framework models the different elements who interact in the attack on SDN networks and introduce elements able to detect the attack in real time. To demonstrate the feasibility of the framework, the SDN network is created in a controlled environment using virtual machines. The botnet is created to control some users of the SDN through the malware script and turn them into zombies. The detection is performed by SNORT, which is installed on the victim's machine. Results show that the executed DDoS attacks are effectively detected shortly after starting the attack. Lastly, SDN network architecture continues to be an interesting option for ever-growing environments. Therefore, the implementation of a DDoS attack detection technique is a useful contribution to developing frameworks for the security of SDNs.

The paper is organized as follows. Section II gives an introduction and overview of SDNs and DDoS attacks carried out on this type of network, as well as the use of botnets to carry out DDoS attacks. Section III discusses the related works detecting DDoS attacks in traditional networks and SDN. Section IV discusses the implementation of the framework. The experimental results are shown in Section V; finally, Section VI

shows the conclusions reached in this project and indicates studies that can be carried out in the future.

2 Main concepts

2.1 Software-Defined Networks (SDN)

SDN is a network architecture that works differently from the traditional network. This architecture is divided into three layers: application layer, control layer and infrastructure layer or data plane, as depicted in Figure *2*. The control and data planes are decoupled in SDN architecture, and the underlying network infrastructure is abstracted from upper applications [3].

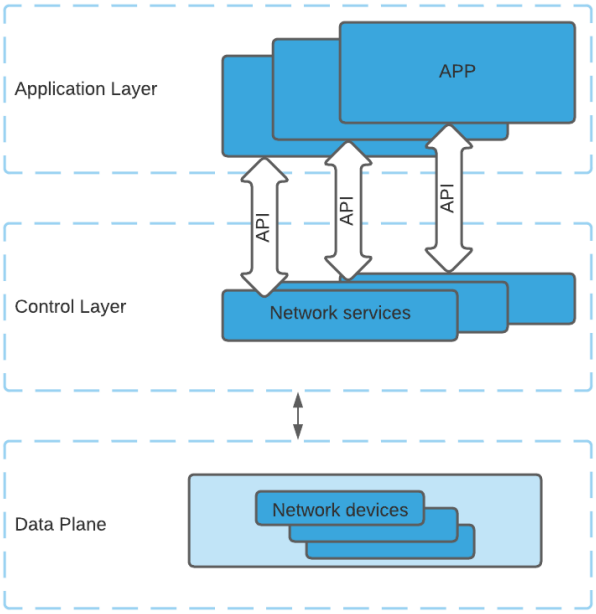

Figure 2. SDN architecture

The data plane contains physical or virtual switch devices, which become simple forwarding devices. The most notable example of a data plane device is the OpenFlow switch [11], [4]. An OpenFlow switch has one or more tables of packet handling rules (flow table). Depending on the rules installed by a controller application, the controller can instruct an OpenFlow switch to behave like a router, switch, firewall, or perform other roles (in general, those of a middlebox) [3].

The control layer is usually composed of logically centralized software-based controllers; they conduct the behavior of the data plane through southbound API and provide network services to upper applications through northbound API. OpenFlow protocol is SDN's first and most widely deployed protocol [1]. The OpenFlow protocol defines the communication mechanism that enables the SDN controller to interact with the data plane directly. The controller pushes packet handling rules in flow tables of OpenFlow switches. The rule matches the traffic conditions and performs specific actions, such as dropping, forwarding, and modifying traffic [1].

For its part, the application layer mainly consists of various business applications, such as virtual networks, security, and traffic engineering applications. The application plane is on the top of the SDN architecture and contains the SDN applications for various functionalities, such as policy implementation, network management, and security services. In the application plane, the SDN application can use the programmable method to submit the network behavior to the control plane [12].

2.2 Botnets and DDoS Attacks on SDNs

A Denial of Service (DoS) attack intends to interrupt the regular operation of the device or network and disrupt the user's activity [5]. One of the most used flood attacks is the ICMP flood [13]. The concept is to send spoofed IP packets to all hosts in the network. This amplifies the traffic and stops the processing of legitimate packets in the network. Another type of DoS attack is Distributed Denial of Service (DDoS) which occurs when multiple synchronized devices perform a DoS attack on one victim [5].

The objective of a DDoS attack is to bring down a target's services using multiple distributed sources. The idea of DDoS attacks revolves around the fact that a large number of sources distributed across various locations are used to target a victim [1]. Botnets are typically helpful for launching DDoS attacks as they are an extensive collection of compromised hosts (also called zombies). The way the bots are controlled depends on the architecture of botnet command and control mechanisms, which may be IRC, HTTP, DNS, or P2P-based. Since DDoS attacks are frequent, the focus has been to develop a proficient solution capable of effectively detecting DDoS attacks. To launch an attack, an attacker generally follows four basic steps [6]:

1) Information gathering to scan a network to find vulnerable hosts to use them later to launch an attack.
2) Compromising the hosts to install malware or malicious programs in the compromised hosts or zombies so that they can be controlled only by the attacker.
3) Launching the attack to command the zombies to send the victim several malicious flows with customizable intensities.
4) Cleaning up to remove all records or history files from memory.

Current DDoS security trends show the limitations of existing technologies, which are propagated toward SDN networks. Attackers are adopting sophisticated mechanisms to bypass traditional protection shields. More recently, software-defined

networks (SDN) have emerged as a new networking paradigm with wide-scale attraction. The distinct characteristics of SDN have led to the development of many SDN-based DDoS attack detection mechanisms, as discussed in the next section.

3 Related Works

Some previous works have ventured into the study of detection and mitigation mechanisms of DDoS attacks on SDNs.

In [7], an intrusion detection system in SDN networks is implemented using three virtual machines: the first has the controller and the IDS, the second emulates the network domain, and the third represents an online server. The system automatically detects various DDoS attacks and notifies to the controller of the infrastructure using a framework based on RYU controller. Then, it transfers new flow rules to the network devices to restore regular network operation as fast as possible. This project implements the entire detection system in the SDN network controller, which increases the amount of work that this device performs, suggesting that the detection system should be installed as close as possible to the machine where it originates the attack.

In [11], a source-based defense mechanism against a DDoS flooding attack using botnets in SDN networks and the sample Flow (sFlow) technology is proposed. The developed detection algorithm is based on a statistical inference model. In this project, an application is developed, and its proper functioning is tested in an emulated network with real traffic. The results show that this mechanism effectively detects DDoS flooding attacks in SDN environments and identifies flows to prevent the damage of the attack from spreading beyond the target. Unlike the contribution in this paper, our work allows for the design and creation of custom botnets by using BYOB to execute DDoS attacks, whereas this study only performs a simplified attack without structuring a botnet to execute the action.

The work in [4] proposes a DDoS blocking scheme applicable to an SDN-managed network. The application that does the detection and blocking work is mounted on the POX application controller, and the communication protocol between devices is OpenFlow. The emulation shows that the POX application successfully filters the DDoS attack traffic from the legitimate traffic. The malicious traffic is blocked and safely redirects the user's traffic from the attacked server address to a new address. The project under discussion requires dedicated communication between the DDoS controller and the server or user to be protected. For its part, our proposal does not need a dedicated communication since the protection system is mounted on the user to be protected.

In [1], a framework called ProDefense is proposed. It is mainly focused on the operation of large-scale infrastructures to protect against DDoS attacks for a data center in a smart city. The framework is modular and adjusts to the requirements of several applications executed in the smart city. For instance, a Traffic Control System requires an extremely agile security solution that can trigger the mitigation system immediately and generate security alerts foreseeing malicious behavior before reaching the

threshold. The security solution must also monitor network traffic trends and predict the attack [1]. While this work designs the detection of several DDoS attacks based on application requirements, our work focuses on the flooding and amplification categories, as shown in **¡Error! No se encuentra el origen de la referencia.**. The detection system is implemented in the victim's machine.

Other studies have been done regarding the various DDoS attack detection methods. In [14], [15], and [16], some research has been carried out using the entropy technique. The entropy technique relates to network characteristic alterations to detect anomalous network activities and identify a possible DDoS attack. In [17] and [18], authors use machine learning techniques such as Bayesian networks, SOM (Self-organizing Map), or fuzzy logic to identify the presence of anomalies. The mentioned techniques take into account various network characteristics as well as traffic analysis to achieve the detection of possible DDoS attacks.

The article presented in [19] uses the pattern analysis technique that assumes that attackers exhibit similar behavior, such as sending the same type of malicious packet or performing the same scanning action within the network. These malicious activities are detected and identified as part of a DDoS attack. This shows that the malicious behavior patterns that occur in traditional networks are helpful for the detection of future attacks aimed at SDN networks. For its part, studies shown in [20] and [21] present the connection rate technique, defined as an indicator of the number of connections made within a specific time window. If this number of connections is exceeded, it can be considered a DDoS attack. In our study, a list of rules is used instead of an indicator, where if the conditions set in the rule are met, a notification of a possible DDoS attack is launched.

In [22], a system of sensors that monitor the network and a set of correlation functions in the Open vSwitches (OVS) are proposed. When a monitor sends an alert, the correlator analyzes it to see if it matches any attack signature. If it does, the monitor, controller, and correlator take action to mitigate the impact of the attack. Our proposed project tries to detect a DDoS attack by using the SNORT IDS without the use of added monitors as part of the SDN network architecture. In [23], authors use SNORT and OpenFlow to detect and mitigate DDoS attacks in real time by modifying traditional network configurations in a cloud environment. Our project uses the SNORT IDS and the OpenFlow communications protocol to detect DDoS attacks on an SDN network architecture rather than on a traditional network.

In [24] an SDN network with an ONOS controller is proposed to detect through SNORT a TCP-SYN-based attack generated from Hping3. The main difference with this project is that the SNORT rules are implemented in the ONOS controller of the SDN network, while in our project the rules are implemented in the victim's computer. In this way, the job of detecting any possible DDoS attack is assigned to the victim machine itself and not as an additional job for the controller. Also, our project compares two known DDoS attacks for detection (SYN flood and ICMP flood) as opposed to this proposed project which only addresses detection of a single attack (TCP-SYN-based). The controller used in [24] uses the ONOS platform as opposed to ours which uses the OpenDaylight platform.

In [25], the authors discuss the limitations of traditional threshold-based methods to detect and mitigate DDoS attacks in SDN networks and explore the potential of machine learning and deep learning techniques to improve detection accuracy. The authors review various existing solutions and data sets and propose a new deep learning-based approach that uses SDN-specific features and two feature selection methods to identify relevant features for DDoS attack detection. The proposed approach is tested on three data sets and achieves high accuracy rates compared to traditional machine learning techniques. The authors highlight the need for more up-to-date and diverse data sets for future research in this area.

The main difference is that our project uses the SNORT tool for detection of DDoS attacks on a simulated SDN network with Mininet and OpenDaylight, while the project discussed in [25] proposes an approach based on deep learning to improve the detection accuracy of DDoS attacks in SDN networks. In addition, our project focuses on the detection of SYN flood and ICMP flood attacks targeting a specific user in the SDN network, while the project discussed in [25] evaluates the detection accuracy on three different data sets and proposes a new approach based on deep learning for the detection of DDoS attacks in SDN networks.

In [26] DDoS attacks against centralized controllers in SDNs are detected using the Mininet emulation tool. Experiments were performed using different penetration tools to launch DDoS attacks and SNORT was integrated for detection. The results showed that ODL and ONOS controllers are vulnerable to DDoS attacks and that detection time is directly proportional to the number of network devices and the amount of traffic bombarded. Rules were created in SNORT for the detection of DDoS attacks and the implementation of a DDoS detection system using SNORT IDS in ODL and ONOS controllers in SDN is discussed. System performance was evaluated using various scenarios with different numbers of hosts, switches, and traffic, and it was found that ODL detected DDoS attacks faster than ONOS.

In [26] the implementation of its detection system is not done in the victim client, but in the controllers, which increases the response time between the attack and the detection. All the previously analyzed projects are summarized in Table 1:

Table 1. Related works regarding detection of SDN attacks

Ref.	Scope	Algorithm	Experiments	Results	Weaknesses
[1]	ProDefense, SDN, DDoS Attacks	ProDefense Framework for smart city data center.	Only theorical analysis	None	Controllers without protection against directed attacks.
[4]	DDoS Attacks, POX controller, botnets	Flow counter, detects and deletes bots	SDN mounted on mininet with several botnets	Attack blocked after several seconds of	Communication between the DDoS blocking application and the server need to be protected.

				attack initiation	
[7]	Flow rules modification	Change on flow rules after detection of DDoS attacks	Three scenarios with different types of DDoS attacks	Average attack reaction time of 3 seconds	Detection system implemented in the controller
[11]	sample Flow (sFlow) technology	Statistical inference model	Emulated network with real traffic	Detects DDoS flooding attacks.	Only performs a simplified attack / no botnets used.
[14], [15], [16]	Network characteristic alterations .	Entropy technique	Performance tests to validate the scalability and the effectiveness of sFlow-based approach	Improved sFlow-based mechanism .	The entire flow table does not scale for high network traffic environments.
[19]	Malware detection for mobile devices using SDN.	Identifying suspicious network activities through real-time traffic analysis	Local testbed and GENI infrastructure	Feasibility approach.	Not all types of malware are analyzed.
[20], [21]	Hybrid approach to detecting scanning worms	FRESCO / Sequential hypothesis testing and connection rate limiting	Two modules with malicious scanner / redirect all the scanner's flow into a remote honeynet	FRESCO with over 90% fewer lines of code. / Successfully restricts the number of scans / highly effective.	Ensure that worms are quickly identified with an attractively low false alarm rate
[22]	Monitorization of SDN and its controller	Selectively inspecting network packets on demand	Detection and mitigation of TCP SYN flood attacks on GENI.	Scalable to process high volume of traffic and large scale attacks	The project uses extra monitors to achieve a right inspection of packets on demand.

[23]	SNORT and OpenFlow to detect DDoS attacks	Modifying network configurations in a cloud environment	Simplified testing environment including two cloud servers with.	Variation of the performance of SnortFlow agent in different scenarios.	The whole project is mounted in a traditional network, not an SDN.
[24]	SDN network with SNORT detection	Detection of TCP-SYN-based attacks	Creation of a network analyzed with Wireshark and implemented attack detection rules with SNORT		Increases the load on controller processing.
[25]	Limitations of traditional threshold-based methods.	NFDLM, a lightweight and optimized Artificial Neural Network	Design of four different models where two are based on ANN and the other two are based on LSTM to detect the attack types of DDoS.	The detection performance achieves approximately 99% accuracy for detection.	It does not focus on specific DDoS attacks but generalizes them.
[26]	DDoS attacks against centralized controllers.	Integration of SNORT for detection of DDoS attacks	Five different network scenarios are considered. The number of hosts, switches and data packets vary.	ODL and ONOS controllers are vulnerable to DDoS attacks	Attacks are not generated using botnets.

As previously discussed, this article tries to overcome some of the limitations analyzed. For this reason, this proposal intends to implement a detection method mounted on the user or server victim of the DDoS attack, avoiding overprocessing in the controller. The DDoS attacks used in our project for its detection are ICMP flood and SYN flood because they are the most common and most used attacks.

4 Framework for DDoS-botnet detection

Our framework proposes a DDoS attack detection system in SDN networks. The framework takes the SDN architecture as a starting point and locates the elements that cause a DDoS attack based on botnets (zombies). To detect and avoid the attack, we introduce the Detection System module which analyzes the traffic circulating in the network (see Figure 3).

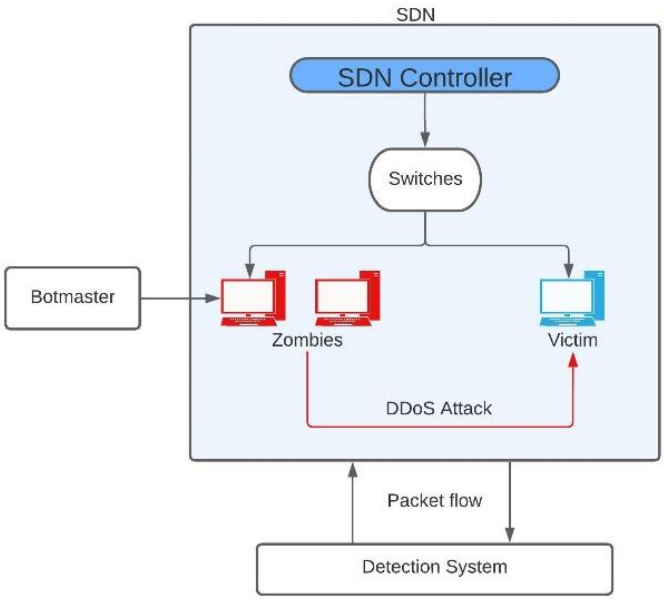

Figure 3. Framework for DDoS-botnet detection in SDN.

The main components of the proposed framework are:

- **SDN Infrastructure:** The SDN infrastructure contains the network hardware devices, the controller, and the users. A group of them will be transformed into zombies and will be in charge of executing the DDoS attack on a victim user.
- **SDN Controller:** It is in charge of managing the flow of data within the SDN network. In addition, the controller has the topology information and controls the configuration of each of the switches that are part of the infrastructure. The controller receives information about the source and destination of incoming packets, runs algorithms to find the optimal path, and issues commands to switches to forward the packets to their destination.
- **Botmaster:** It is in charge of managing zombie users and issuing orders to execute the DDoS attack. The botmaster can be inside or outside the SDN

network and the zombies are controlled by executing malicious scripts on the computers.

- **Detection system:** It is in charge of analyzing the packets received by the user and identifying any anomaly they may have. The identification of these anomalies is done by means of previously configured rules in the detection system.
- **DDoS Attacks:** It is a group of hosts connected to the SDN network and controlled by the botmaster which carries out different attacks directed at a user within the network. It is worth mentioning that the SDN network needs the integration between the data and control planes, so the attack on any of these elements also influences the performance of the SDN controller.

The Figure 4 shows the sequence of events in which the framework works. The infrastructure and the SDN controller are the first elements that are activated for providing the communication network. At this point, users receive the network service normally. The user mounts and activates the DDoS detection module. This way, the system analyzes all user traffic and sends alert messages if a possible DDoS attack occurs.

For its part, the attacker selects the victim and starts the attack procedure. The botmaster is mounted by means of a script in order to infect devices and turn them into zombies. Once the devices are infected, the botmaster gives the order to force zombies to start to overload the victim's network resources. At this point, the detection module analyzes the traffic in real time, detects the attack and generates alert messages in the victim user.

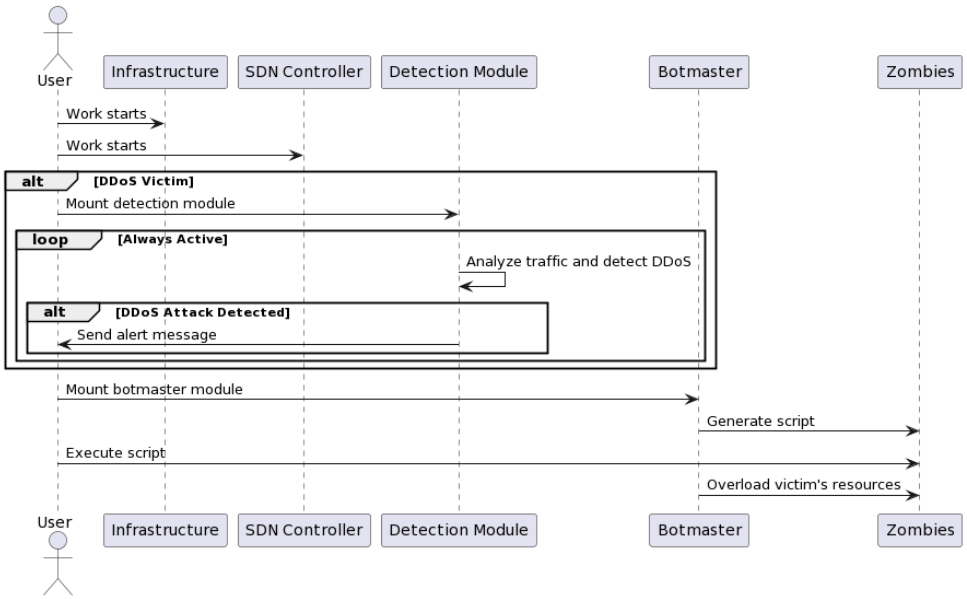

Figure 4. Sequence diagram of the proposed framework

5 Implementation

The proposed architecture is implemented used the tools described in the Figure 5. Similarly, the Table 2 describes the technical details of the used software.

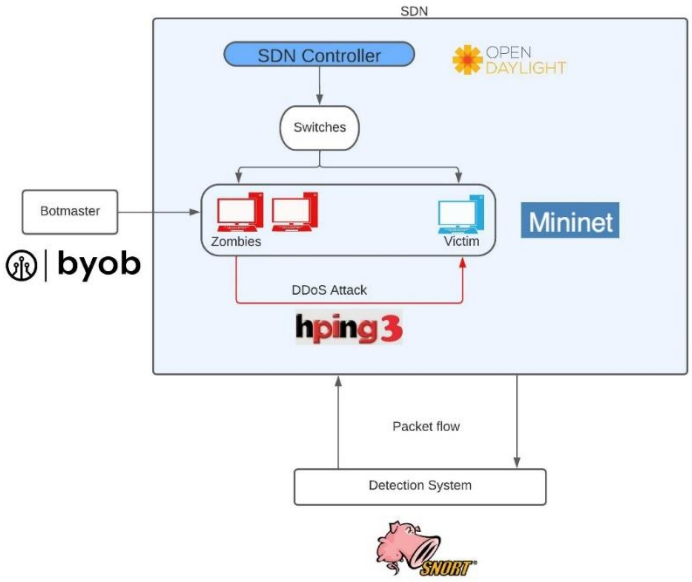

Figure 5. Implementation of the proposed framework

The SDN network is created from a script in Mininet; the network contains an SDN controller, a core switch, two distribution switches, and four access switches that connect twenty users. One of these users will become the victim of the DDoS attack in the two proposed versions (ICMP flood and SYN flood).

Table 2. Tools for detection of DDoS attacks in SDN networks

Module	Tool	Version	Details
SDN Controller	OpenDaylight [27]	0.13.1	Open source Handles Openflow interfaces It has a GUI [28]
Botmaster	BYOB [29]	2.1	Open source. Provides a command line interface (CLI). Allows botnet's customization.
DDoS Attack	Hping3 [30]	3	Open source.

			Generation of custom packets and attack flows. Port and service scanning.
SDN infrastructure	Mininet [31]	2.3.0d7	Open source Realistic network emulation Integration with OpenFlow controllers
DDoS Detection	SNORT [32]	3.1.64.0	Open source Support for multiple platforms. Threat detection.

The botnet attack system is created from the BYOB software that deploys a server or botmaster and is responsible for developing a Python script. When the script is executed on the hosts of the SDN network, it will become part of the botnet.

The script has the option of being created as an executable file or as a Python code file. When initialized, the botmaster takes full control of the user who is part of the SDN without the user's knowledge. The same script generated by the botmaster is used to control any number of users that are part of the network. In our project, the python script option was chosen, and it was executed on each host through its terminal.

The zombies will launch a DDoS attack on a single SDN host. The DDoS attack detection system is implemented using the SNORT tool [32]. Once installed in the virtual machine, a list of rules is created to detect ICMP flood and SYN flood attacks in a short time after the attack has been executed. Writing effective Snort rules requires a good understanding of security threats and the ability to analyze network traffic to identify potential attack patterns.

SNORT uses a simple, lightweight rules description language that is flexible and powerful. SNORT rules must completely contain a single line. The rule parser does not know how to handle rules on multiple lines. [33]

These rules are divided into two logical sections: the rule header and the rule options. The rule header contains the rule's action, protocol, source, destination IP addresses and netmasks, and the source and destination ports information. The rule option section contains alert messages and information on which parts of the packet should be inspected to determine if the rule action should be taken. [33]

5.1 Topology

The SDN network was created in Mininet with a script made in Python. A network containing an SDN controller, a core switch, two distribution switches, and four access switches that communicate with the OpenFlow protocol were created. The network also contains 20 users, and each access switch was connected to five of them. One of these users became the victim of the DDoS attack in the two proposed versions (ICMP flood and SYN flood), and five of these users became zombies by executing the script created by the botnet server. The SDN network topology created for the execution of experiments is shown in Figure *6* .

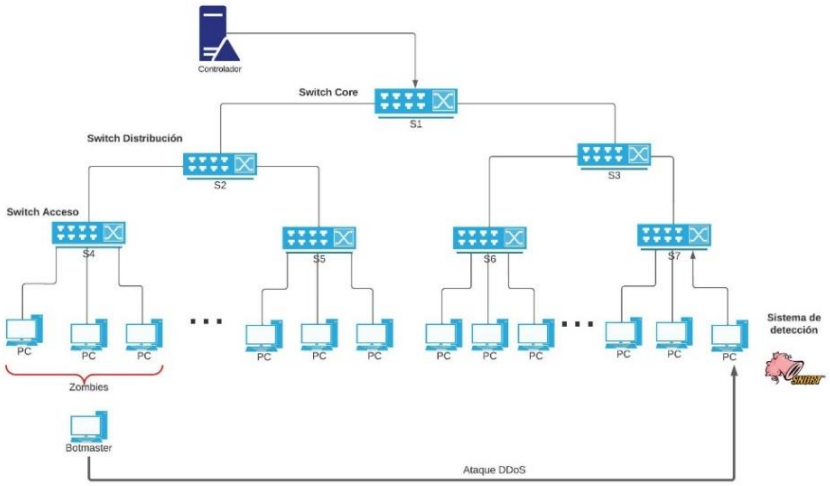

Figure 6. Topology of the experiments.

6 Experiments

The framework is evaluated using the topology mentioned in the previous section. This section presents in detail the two attack scenarios implemented and the use of the elements that generate the DDoS attack in the SDN. In the first scenario, the hosts turned into zombies are in charge of executing the ICMP Flood attack. In the second scenario, the zombies execute the SYN flood attack. Both attacks start unexpectedly targeting a network user, who has SNORT attack detection permanently active. Table 3 details the characteristics of each of the attacks. These attacks were selected as they are considered two of the most important attacks due to their effectiveness in exhausting the resources of the target server and affecting the availability of online services.

Table 3. Types of attacks according to the scenario

Attack	Source	Target	Details
ICMP Flood	Zombies (Host1 through Host5).	Victim host (Host 20)	ICMP attack Packet body size 1500 bytes
SYN Flood	Zombies (Host1 through Host5).	Victim host (Host 20)	SYN/TCP attack Window size 64 Packet body size 100000 bytes

7 Results and Discussion

7.1 Results

In the ICMP flood attack, the resource consumption of the victim machine is increased, both in the use of processing cores and memory consumption. Figure 7 shows the performance before and after the DDoS attack.

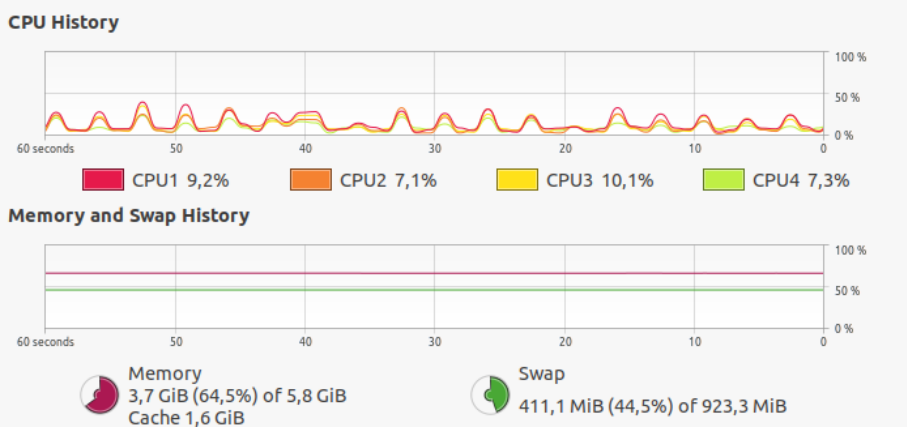

Figure 7. Resource consumption before and after an ICMP flood attack.

Although the resource consumption increases considerably, it does not saturate the victim's machine (a consumption of 30% on average in the 4 cores). Even so, the bombardment of ICMP packets toward the target machine is easily detected through SNORT. The time between the start of the attack and its detection is shown in Figure 8. It shows that the time range between the execution of the attack and its detection is very short. The ICMP flood attack started at 14:40:00 secs. The botmaster sent the order for the zombies to attack the victim and SNORT detected and sent alert messages of a possible DDoS attack 4 seconds after it started. Therefore, this framework is an efficient solution when detecting this type of attack.

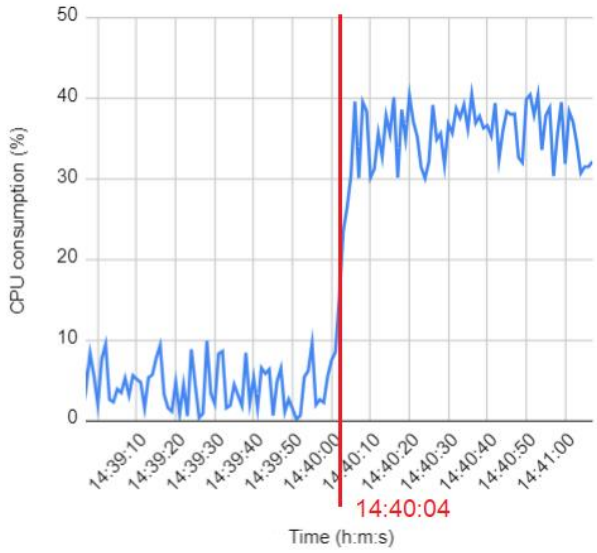

Figure 8. Timeline before and after the ICMP flood attack (with identification of the point at which the detection was made)

The SYN flood attack had a significant impact on the victim machine. Figure 9 shows the performance during the DDoS attack. In the SYN flood attack, unlike the ICMP flood attack, the resource consumption of the victim machine has been increased considerably. Core consumption on the victim machine was so high that most of the time, three of the four cores were above 80% consumption. Memory consumption also rose, from 15% usage to 70%.

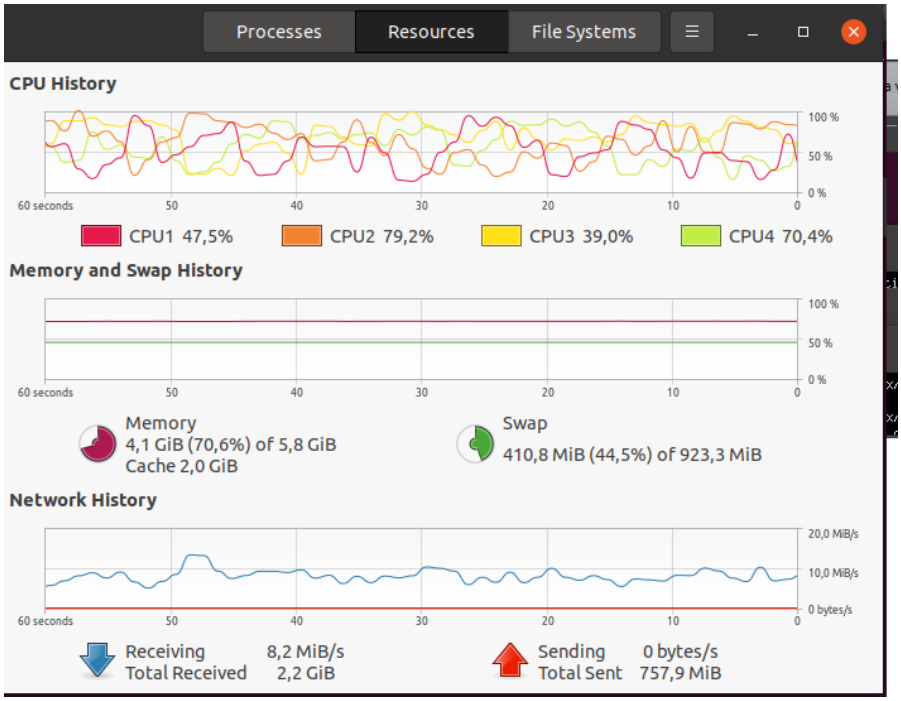

Figure 9. Resource consumption before and after an SYN flood attack.

Regarding detection using the SNORT tool, it showed similar results to the ICMP attack. The detection of the attack was made almost immediately. The SYN flood attack started at 14:15:05 secs. The botmaster sent the order for the zombies to attack the victim and SNORT detected and sent alert messages of a possible DDoS attack 3 seconds after it started. Figure 10 shows the time difference between the start of the DDoS attack and its detection. Even though the attack is running consuming most of the victim's resources, it can detect the attack through SNORT, and it sends a large number of alert messages right after starting the attack. Table 4 summarizes the obtained results.

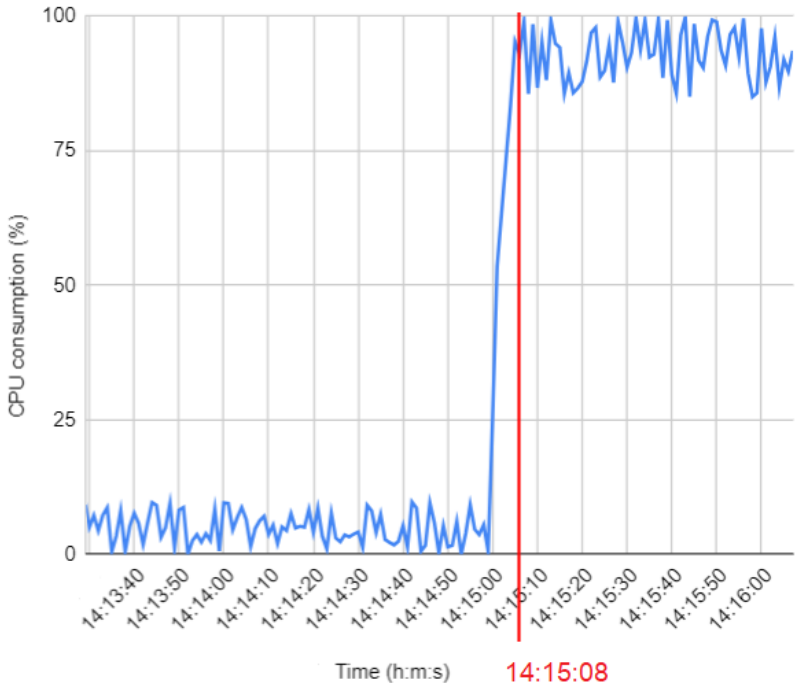

Figure 10. Timeline before and after the ICMP flood attack (with identification of the point at which the detection was made)

Table 4. DDoS attack detection results

DDoS Attacks	Attack start time	Attack detection time	CPU consumption
ICMP Flood	14:40:00	14:40:04	40%
SYN Flood	14:15:05	14:15:08	90%

7.2 Discussion

Our work focuses on the execution of DDoS attacks through two types of attacks: ICMP flood and SYN flood, since they are two of the most common attacks used in traditional networks [34]. During the execution of these two attacks, it was noticed that the SYN flood attack was detected a second faster than the ICMP flood attack. If some of the settings in the rules are changed, the difference in detection time for these same types of attacks does not change from the results shown in this section.

Resource consumption was different when running these two types of attacks. The resource impact in the ICMP flood attack was 50% lower compared to the SYN flood attack. This is because TCP packets with SYN headers bombardment were much higher

than ICMP packets' bombardment. The difference in resource consumption can be seen in Figure 7 and Figure *9*.

In the SYN flood attack, it is necessary to highlight the higher consumption of resources compared to the ICMP flood attacks. These results also demonstrated the detection capacity of the SNORT tool since, seconds after the attack was carried out, the tool identified and entered the DDoS attacks in the logs.

Detection response times were slightly different. The results showed that the SYN flood attack detection time was a second faster than the ICMP flood attack detection time. However, in both cases, the detection has been successful. Even when the SYN flood attack resulted in high resource consumption for the victim, the victim managed to detect the attack almost instantly.

8 Conclusions

The presented architecture provides a comprehensive framework for detecting DDoS attacks generated by botnets in an SDN network. It allows for the analysis, testing, and comparison of an SDN network's behavior before and after DDoS attacks.

The implementation of the framework successfully demonstrated the detection capacity, as it quickly identified and logged DDoS attacks seconds after they were carried out. The detection response times were slightly different between the SYN flood and ICMP flood attacks, but both attacks were successfully detected.

The experiments conducted on the SDN network showed that the SYN flood attack had a more significant impact on the network's proper functioning compared to the ICMP flood attack. The resource consumption was higher in the SYN flood attack due to the bombardment of TCP packets with SYN headers. However, the victim host managed to detect the attack almost instantly, even with high resource consumption.

The framework's integration of SNORT for detecting DDoS attacks in SDN networks provides an advantage over traditional networks. Traditional networks face challenges in scaling and ensuring low false alarm rates, which SDN networks can overcome with the proposed framework.

9 Future work

Challenges for future work include the implementation of a DDoS attack mitigation mechanism within the framework proposed in this project. This is necessary so that actions can be taken in addition to detection, and the benefit is greater. Similarly, it is recommended the execution of experiments with a more significant number of users and, therefore, with a higher number of bots. The extension of other types of attacks additional to ICMP flood and SYN flood is also considered for future work.

Declaration of competing interest

We declare that we have no significant competing interests including financial or non-financial, professional, or personal interests interfering with the full and objective presentation of the work described in this manuscript.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

Author contributions

Ángel Leonardo Valdivieso Caraguay: Conceptualization, Formal analysis, Supervision.
Lorena Isabel Barona López: Validation, Visualization, Methodology
Jaime Tamayo: Investigation, Data curation, Writing.

Data availability statement

The authors confirm that the data supporting the findings of this study are available within the article and its supplementary materials.

References

[1] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions," *Arab. J. Sci. Eng.*, vol. 42, no. 2, pp. 425–441, 2017, doi: 10.1007/s13369-017-2414-5.

[2] "Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper - Cisco." https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (accessed Dec. 24, 2021).

[3] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2015, doi: 10.1109/JPROC.2014.2371999.

[4] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang, "A SDN-oriented DDoS blocking scheme for botnet-based attacks," *Int. Conf. Ubiquitous Futur. Networks, ICUFN*, pp. 63–68, 2014, doi: 10.1109/ICUFN.2014.6876752.

[5] D. Nikolov, I. Kordev, and S. Stefanova, "Concept for network intrusion detection system based on recurrent neural network classifier," *2018 IEEE 27th Int. Sci. Conf. Electron. 2018 - Proc.*, pp. 1–4, 2018, doi: 10.1109/ET.2018.8549584.

[6] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "Botnet in DDoS Attacks: Trends and Challenges," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2242–2270, 2015, doi: 10.1109/COMST.2015.2457491.

[7] P. Manso, J. Moura, and C. Serrão, “SDN-based intrusion detection system for early detection and mitigation of DDoS attacks,” *Inf.*, vol. 10, no. 3, pp. 1–17, 2019, doi: 10.3390/info10030106.

[8] K. Shinan, K. Alsubhi, A. Alzahrani, and M. U. Ashraf, “Machine learning-based botnet detection in software-defined network: A systematic review,” *Symmetry (Basel).*, vol. 13, no. 5, pp. 1–28, 2021, doi: 10.3390/sym13050866.

[9] O. Katz and J. Black, “Akamai DNS Traffic Insights Threat Report | akamai,” *Akamai DNS Traffic Threat Report*. Jun. 2022, [Online]. Available: https://www.akamai.com/resources/research-paper/akamai-dns-traffic-insights-threat-report.

[10] “What is an ICMP Flood Attack? | NETSCOUT.” https://www.netscout.com/what-is-ddos/icmp-flood (accessed Jul. 05, 2023).

[11] Y. Lu and M. Wang, “An easy defense mechanism against botnet-based DDoS flooding attack originated in SDN environment using sFlow,” *ACM Int. Conf. Proceeding Ser.*, vol. 15-17-June, pp. 14–20, 2016, doi: 10.1145/2935663.2935674.

[12] S. Dong and M. Sarem, “DDoS Attack Detection Method Based on Improved KNN with the Degree of DDoS Attack in Software-Defined Networks,” *IEEE Access*, vol. 8, pp. 5039–5048, 2020, doi: 10.1109/ACCESS.2019.2963077.

[13] S. T. Zargar, J. Joshi, and D. Tipper, “A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks,” *IEEE Commun. Surv. Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013, doi: 10.1109/SURV.2013.031413.00127.

[14] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, “Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments,” *Comput. Networks*, vol. 62, pp. 122–136, 2014, doi: 10.1016/j.bjp.2013.10.014.

[15] R. Wang, Z. Jia, and L. Ju, “An entropy-based distributed DDoS detection mechanism in software-defined networking,” *Proc. - 14th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. 2015*, vol. 1, pp. 310–317, 2015, doi: 10.1109/Trustcom.2015.389.

[16] S. A. Mehdi, J. Khalid, and S. A. Khayam, “Revisiting Traffic Anomaly Detection Using Software Defined Networking.pdf,” *Int. Work. Recent Adv. intrusion Detect.*, pp. 161–180, 2011.

[17] S. Dotcenko, A. Vladyko, and I. Letenko, “A fuzzy logic-based information security management for software-defined networks,” *Int. Conf. Adv. Commun. Technol. ICACT*, pp. 167–171, 2014, doi: 10.1109/ICACT.2014.6778942.

[18] C. Chung, S. Member, P. Khatkar, and T. Xing, “NICE : Network Intrusion Detection and Countermeasure,” *IEEE Trans. Dependable Secur. Comput.*, vol. 10, no. 4, pp. 1–14, 2013, [Online]. Available: http://dblp.uni-trier.de/db/journals/tdsc/tdsc10.html#ChungKXLH13.

[19] R. Jin and B. Wang, “Malware detection for mobile devices using software-defined networking,” *Proc. - 2013 2nd GENI Res. Educ. Exp. Work. GREE 2013*, pp. 81–88, 2013, doi: 10.1109/GREE.2013.24.

[20] S. E. Schechter, J. Jung, and A. W. Berger, “Fast detection of scanning worm

infections," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3224, pp. 59–81, 2004, doi: 10.1007/978-3-540-30143-1_4.

[21] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, "FRESCO Modular Composable Security Services for Software-Defined Networks," *20th Annu. Netw. Distrib. Syst. Secur. Symp. Ndss*, vol. 2, no. February, p. 16, 2013.

[22] T. Chin, X. Mountrouidou, X. Li, and K. Xiong, "Selective packet inspection to detect DoS flooding using software defined networking (SDN)," *Proc. - 2015 IEEE 35th Int. Conf. Distrib. Comput. Syst. Work. ICDCSW 2015*, pp. 95–99, 2015, doi: 10.1109/ICDCSW.2015.27.

[23] T. Xing, D. Huang, L. Xu, C. J. Chung, and P. Khatkar, "SnortFlow: A OpenFlow-based intrusion prevention system in cloud environment," *Proc. - 2013 2nd GENI Res. Educ. Exp. Work. GREE 2013*, no. July 2015, pp. 89–92, 2013, doi: 10.1109/GREE.2013.25.

[24] M. Kumar and A. Bhandari, "DDoS Detection in ONOS SDN Controller Using Snort," in *ICT with Intelligent Applications*, 2023, pp. 155–164.

[25] K. Saurabh, T. Kumar, U. Singh, O. P. Vyas, and R. Khondoker, "NFDLM: A Lightweight Network Flow based Deep Learning Model for DDoS Attack Detection in IoT Domains," *2022 IEEE World AI IoT Congr. AIIoT 2022*, no. Dl, pp. 736–742, 2022, doi: 10.1109/AIIoT54504.2022.9817297.

[26] S. Badotra and S. N. Panda, "SNORT based early DDoS detection system using Opendaylight and open networking operating system in software defined networking," *Cluster Comput.*, vol. 24, no. 1, pp. 501–513, 2021, doi: 10.1007/s10586-020-03133-y.

[27] "OpenDaylight Downloads — OpenDaylight Documentation Sulfur documentation." https://docs.opendaylight.org/en/stable-sulfur/downloads.html (accessed Jul. 05, 2023).

[28] C. D. Cajas and D. O. Budanov, "SDN Applications and Plugins in the OpenDaylight Controller," *Proc. 2020 IEEE Conf. Russ. Young Res. Electr. Electron. Eng. EIConRus 2020*, pp. 9–13, 2020, doi: 10.1109/EIConRus49466.2020.9039054.

[29] "Guide." https://byob.dev/guide (accessed Jul. 05, 2023).

[30] "hping3 | Kali Linux Tools." https://www.kali.org/tools/hping3/ (accessed Jul. 05, 2023).

[31] "Download/Get Started With Mininet - Mininet." http://mininet.org/download/ (accessed Jul. 05, 2023).

[32] "Snort - Network Intrusion Detection & Prevention System." https://www.snort.org/ (accessed Jul. 05, 2023).

[33] "Writing Snort Rules." https://paginas.fe.up.pt/~mgi98020/pgr/writing_snort_rules.htm (accessed Jul. 05, 2023).

[34] T. Mahjabin, Y. Xiao, G. Sun, and W. Jiang, "A survey of distributed denial-of-service attack, prevention, and mitigation techniques," *Int. J. Distrib. Sens. Networks*, vol. 13, no. 12, 2017, doi: 10.1177/1550147717741463.