UNIVERSIDAD DE CUENCA

Facultad de Ingeniería

Carrera de
Electrónica y Telecomunicaciones

Sistema autónomo de detección de botnets en la red de la Universidad de Cuenca basado en el comportamiento anómalo del tráfico DNS.

Trabajo de titulación previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones.

Autor:

Vicente Geovanny Quezada Pauta C.I: 010633832-0
viquezada.14@gmail.com

Director:

PhD. Darwin Fabián Astudillo Salinas C.I: 010390703-6

Cuenca - Ecuador
12 de abril de 2021

Resumen

En los ciberataques actuales se hace uso de las *botnets*, como técnica avanzada para generar ataques sofisticados y coordinados. Se utilizan códigos maliciosos o vulnerabilidades para infectar terminales convirtiéndolos en *bots*. Los sistemas infectados se conectan a un servidor de Comando y Control (C&C) para recibir comandos y realizar ataques. Detectar *hosts* infectados permite proteger los recursos de una red, y evita que estos sean usados para actividades ilícitas hacia terceros.

En este trabajo experimental de titulación se detalla el diseño, implementación y resultados de un sistema de detección de infecciones de *bot* basado en el tráfico Sistema de Nombres de Dominio (DNS), para una red universitaria. Se realiza un análisis de factibilidad de detección de *hosts* infectados por *bot* basado en la creación de huellas digitales. Las huellas son generadas a partir de un análisis numérico de 15 atributos DNS de los *hosts* de la red por horas. Con las huellas digitales se busca anomalías haciendo uso de *Isolation Forest*, con la finalidad de etiquetar a un *host* como infectado o no. Luego se usa *Random Forest* para generar un modelo que detecte futuras infecciones hacia *hosts* por *bot*.

El sistema de gestión y manejo de eventos DNS integra Suricata, la pila Elasticsearch, Logstash y Kibana (ELK) y Python. Esta integración facilita el almacenamiento de eventos, generación de las huellas digitales y análisis de resultados de clasificación de las huellas. Además, se comprueba la factibilidad del método de detección de *hosts* infectados por *bot* usando huellas dactilares, sobre otros métodos tradicionales, haciendo un análisis de comportamiento en el tiempo de *hosts* infectados y búsqueda de consultas hacia dominios generados por Algoritmo de Generación de Dominio (DGA).

Palabras clave : *Botnet*. Detección de *bots*. Detección de *bots* basado en DNS. Detección de anomalías. Suricata. Pila ELK. Aprendizaje automático. Bosques de aislamiento. Bosques aleatorios

Abstract

In current cyber attacks, botnets are used as an advanced technique to generate sophisticated and coordinated attacks. Malicious code or vulnerabilities are used to infect terminals turning them into bots. Infected systems connect to a C&C server to receive commands and carry out attacks. Detecting an infected host helps to protect network resources and prevents them from being used to attack third-party networks.

This experimental thesis work details the design, implementation and results of a bot infection detection system based on DNS traffic, for a university network. A bot-infected host detection feasibility analysis is performed based on fingerprint creation. The fingerprints are generated from a numerical analysis, by hours, of 15 DNS hosts attributes on the network. Anomalies are searched for fingerprints using Isolation Forest, in order to label a host as infected or not. Then Random Forest is used to generate a model that detects future infections to hosts by bot.

The DNS event management and handling system integrates Suricata, the ELK stack and Python. This integration makes it easy to store events, generate fingerprints, and analyze the results of fingerprint classification. In addition, it checks the feasibility of the method of detecting hosts infected by bot using fingerprints, compared to other traditional methods, performing a behavior analysis over time of the infected hosts and looking for queries towards domains generated by DGA.

Keywords : Botnet. Bot detection. DNS based botnet detection. Anomaly detection. Suricata. ELK stack. Machine learning. Isolation forest. Random forest

Índice general

Resumen 1

Abstract 2

Índice general 3

Índice de figuras 7

Índice de tablas 9

Cláusula de Propiedad Intelectual 10

Cláusula de licencia y autorización para publicación en el Repositorio Institucional 11

Certifico 12

Dedicatoria 13

Agradecimientos 14

Abreviaciones y acrónimos 15

1. Introducción 17
- 1.1. Identificación del problema . . . 17
- 1.2. Justificación . . . 17
- 1.3. Alcance . . . 18
- 1.4. Objetivos . . . 18
  - 1.4.1. Objetivo general . . . 18
  - 1.4.2. Objetivos específicos . . . 19

2. Marco teórico 20
- 2.1. Anomalías de tráfico de red . . . 20
  - 2.1.1. Tráfico de red . . . 20
  - 2.1.2. Anomalías en la red . . . 20
    - 2.1.2.1. Ataques a la red . . . 21
  - 2.1.3. Detección de intrusos . . . 21
    - 2.1.3.1. Sistema de Detección de Intrusiones . . . 21
    - 2.1.3.2. Importancia de los Sistemas de Detección de Intrusos . . . 22

2.1.4. Detección de anomalías y los Intrusion Detection System (IDS) . . . 23
2.1.4.1. La detección de anomalías y el aprendizaje automático . . . 23
2.1.4.2. Arquitectura de un sistema de detección de anomalías de la red. . . . 24
2.2. *Botnet* . . . 25
2.2.1. Arquitectura de una *Botnet* . . . 25
2.2.2. Ciclo de vida de una *botnet* . . . 26
2.2.3. Importancia de la lucha contra las *botnets* . . . 27
2.3. Sistema de nombres de dominio (DNS) . . . 28
2.3.1. Registros DNS . . . 29
2.4. *Botnets* y los sistemas de nombre de dominio . . . 29
2.4.1. Técnicas de detección de *botnets* basadas en DNS . . . 29
2.4.2. Algoritmo de generación de dominio (DGA) . . . 30
2.4.3. Trabajos relacionados. . . . 31
2.5. Conclusiones . . . 32

3. Análisis y Diseño del Sistema 34
3.1. Topología de la red . . . 34
3.2. Captura de los datos . . . 35
3.2.1. Captura de eventos DNS . . . 35
3.2.2. Modo promiscuo de una interfaz de red . . . 36
3.2.3. Port Mirroring . . . 36
3.3. Gestión y análisis de los eventos DNS . . . 36
3.3.1. Gestores de eventos de seguridad . . . 36
3.3.1.1. La pila ELK . . . 37
3.4. Detección de anomalías . . . 37
3.4.1. Generación de huellas digitales de los *hosts* . . . 38
3.4.1.1. Atributos basados en solicitudes DNS . . . 38
3.4.1.2. Atributos basados en dominios . . . 38
3.4.1.3. Atributos basados en respuesta . . . 39
3.4.1.4. Atributos basados en Protocolo de Internet (IP). . . . 39
3.4.1.5. Atributos basados en Mapeo (Nombre de Dominio Completo (FQDN) - IP) . . . 39
3.4.2. Detección de valores atípicos con Python . . . 39
3.4.2.1. Bosques de aislamientos . . . 41
3.4.3. Análisis de nombres de dominio . . . 41
3.4.3.1. Algoritmo de medición de aleatoriedad . . . 42
3.5. Aprendizaje automático para detección autónoma . . . 43
3.5.1. Clasificación con bosques aleatorios . . . 43
3.5.2. Métricas de error . . . 43
3.5.2.1. Matriz de confusión . . . 44
3.5.2.2. Precisión . . . 44
3.5.2.3. Sensibilidad . . . 44
3.5.2.4. Puntaje F1 . . . 44
3.5.2.5. Exactitud . . . 44

3.6. Arquitectura de la solución . . . 44
3.7. Conclusiones . . . 46

4. Implementación y Evaluación del Sistema **48**
4.1. Implementación del sistema . . . 48
4.1.1. Captura de eventos DNS . . . 48
4.1.2. Gestión de los eventos DNS . . . 48
4.1.3. Generación de huellas digitales DNS de los hosts . . . 49
4.1.4. Detección de anomalías en las huellas digitales . . . 51
4.1.4.1. Detección de aleatoriedad de nombres de dominio . . . 51
4.1.5. Detección automática con Random Forest . . . 52
4.2. Evaluación de resultados . . . 53
4.2.1. Datos capturados . . . 53
4.2.2. Generación de las huellas digitales DNS . . . 54
4.2.3. Detección de anomalías . . . 55
4.2.4. Verificación por consultas a dominios DGA . . . 59
4.2.5. Aprendizaje automático . . . 60
4.3. Conclusiones . . . 68

5. Conclusiones. **69**
5.1. Conclusiones . . . 69
5.2. Trabajos Futuros . . . 70

A. Instalación y configuración del *software* requerido para el proyecto. **72**
A.1. Instalación y configuración de Suricata . . . 72
A.1.1. Instalación de dependencias . . . 72
A.1.2. Instalación de Suricata . . . 72
A.1.3. Suricata como servicio . . . 73
A.1.4. Modo promiscuo sobre una tarjeta de red en Ubuntu 20.04.1 LTS . . . 73
A.2. Instalación y configuración de la pila ELK sobre Ubuntu 20.04 LTS . . . 73
A.2.1. Paquetes dependientes . . . 73
A.2.2. Instalación y configuración de Elasticsearch . . . 74
A.2.2.1. Instalación de Elasticsearch . . . 74
A.2.2.2. Elasticsearch como servicio . . . 74
A.2.2.3. Acceso y consulta remota a Elasticsearch . . . 74
A.2.3. Instalación y configuración de Logstash . . . 74
A.2.3.1. Instalación de Logstash . . . 74
A.2.3.2. MaxMind GeoIP Database . . . 75
A.2.3.3. Filtro Dominio de Nivel Superior (TLD) de Logstash . . . 75
A.2.3.4. Entrada, procesamiento y salida de registros en Logstash . . . 75
A.2.4. Instalación y configuración de Kibana . . . 77
A.2.4.1. Instalación de Kibana . . . 77
A.2.4.2. Acceso remoto a Kibana . . . 77
A.2.4.3. Interacción con Kibana . . . 78

A.2.4.4. Creación de patrones de índices 78
A.2.4.5. Consola de desarrollo 78
A.2.4.6. Creación de *Dashboards* y visualizaciones 79

B. Generación de huellas digitales DNS **82**
B.1. Conexión Python-Elasticsearch 82
B.2. Generación de huellas digitales 83

C. Detección de anomalías en huellas digitales DNS **85**
C.1. Detección de valores atípicos en las huellas digitales con *Isolation Forest* de Sklearn . . 85
C.2. Reducción de dimensionalidad y visualización 85

D. Aprendizaje automático con *Random Forest* **87**

Bibliografía **88**

Índice de figuras

2.1. Etapas de una cadena de eliminación o proceso de ataque [1] . . . 22
2.2. Arquitectura genérica de un sistema de detección intrusiones basado en anomalías de la red. . . . 25
2.3. Arquitectura una *botnet*. . . . 26
2.4. Ciclo de vida de una *botnet*. . . . 27
2.5. Consulta DNS recursiva. . . . 28
2.6. Resumen del enfoque actual de las técnicas de detección de *botnets* basados en DNS. . . 30

3.1. Topología de la red. . . . 35
3.2. Comparación de los algoritmos de detección de valores atípicos en scikit-learn. . . . 40
3.3. Esquema de funcionamiento de *Isolation Forest* . . . 41
3.4. Diagrama de flujo del algoritmo de medición de aleatoriedad. . . . 42
3.5. Esquema de funcionamiento de *Random Forest*. . . . 43
3.6. Arquitectura de la solución en función de la topología de la red. . . . 45
3.7. Arquitectura de la solución desde el punto de vista del software. . . . 45
3.8. Arquitectura de la solución desde la perspectiva de los procesos. . . . 46

4.1. Atributos de los eventos DNS (*query* y *answer*) capturados por Suricata. . . . 49
4.2. Atributos en formato tabla de una respuesta DNS. . . . 50
4.3. *Dataset* de huellas digitales DNS. . . . 52
4.4. *Dataset* de huellas digitales DNS catalogadas. . . . 53
4.5. Métricas de error. . . . 53
4.6. Eventos DNS capturados. . . . 54
4.7. Hosts activos en la red con nombre de dominio asociado. . . . 55
4.8. Número de *hosts* activos en la red por horas. . . . 56
4.9. Valores estadísticos para los atributos generados por la pestaña Aprendizaje Automático (ML) de Kibana. . . . 57
4.10. Número de muestras catalogadas como anomalías en función del número de árboles. . . 57
4.11. Varias vistas de la reducción realizada por Análisis de Componentes Principales (PCA) a 3 dimensiones de las huellas clasificadas. . . . 59
4.12. Clasificación de las huellas en función de sus atributos a escala lineal. . . . 63
4.13. Clasificación de las huellas en función de sus atributos a escala logarítmica. . . . 64
4.14. Casos A y B de tráficos DNS para neto, limpio y *bot*. . . . 65

4.15. Algunos *hosts* y sus dominios Dominio de Segundo Nivel (SLD) catalogados como sospechosos. 65
4.16. Parámetros de precisión y tiempo de entrenamiento en función del número de árboles para el algoritmo *RandomForest* 66
4.17. Reporte de clasificación para el modelo de *Random Forest* con 25 árboles. 66
4.18. Tabla de porcentaje del valor de importancia de cada característica para el modelo de *Random Forest* con 25 árboles. 66
4.19. Gráfica de nivel de importancia de cada característica para el modelo de *Random Forest* con 25 árboles. 67

A.1. Ventana de creación de patrón de índice en Kibana. 78
A.2. Ventana *Discover* de Kibana. 79
A.3. Ventana de consola de Kibana. 80
A.4. Visualizaciones disponibles en Kibana. 81
A.5. *Dashboard* creado en Kibana. 81

Índice de tablas

2.1. Registros DNS más comunes. 29
2.2. Ejemplo de dominios generados por DGA. 31

3.1. Análisis de algunos dominios DGA. 42

4.1. Campos relevantes de los eventos DNS almacenados en Elasticsearch. 51
4.2. Valores máximo, mínimo y promedio de los *hosts* activos en la red. 55
4.3. Valores mínimo, promedio y máximo de los atributos DNS de las huellas digitales. . . 58
4.4. Número de huellas, porcentaje sobre el número total de huellas y número de *hosts* en cada clasificación. 58
4.5. Algunos *hosts* y su porcentaje de huellas catalogadas como *bot*. 60
4.6. Nombres de dominios SLD sospechosos para algunas IPs. 61
4.7. Matriz de confusión para el modelo de *Random Forest* con 25 árboles. 62

B.1. Resumen de operadores y campos necesarios para la obtención de las características P1 a P15. 84

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo, Vicente Geovanny Quezada Pauta en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Sistema autónomo de detección de botnets en la red de la Universidad de Cuenca basado en el comportamiento anómalo del tráfico DNS", de conformidad con el Art. 114 del *CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS*, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos. Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 12 de abril de 2021

Vicente Geovanny Quezada Pauta
010633832-0

Cláusula de Propiedad Intelectual

Yo, Vicente Geovanny Quezada Pauta, autor de la tesis "Sistema autónomo de detección de botnets en la red de la Universidad de Cuenca basado en el comportamiento anómalo del tráfico DNS", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 12 de abril de 2021

Vicente Geovanny Quezada Pauta
010633832-0

Certifico

Que el presente proyecto de tesis: Sistema autónomo de detección de botnets en la red de la Universidad de Cuenca basado en el comportamiento anómalo del tráfico DNS, fue dirigido y revisado por mi persona.

Ing. Darwin Fabián Astudillo Salinas, PhD
Director

Dedicatoria

A mis padres y hermanos por el sacrificio y el apoyo incondicional brindado a lo largo de mi vida y más aún durante mi trayectoria Universitaria. Ser la principal inspiración y motivación que me impulsó a considerar una carrera universitaria y cruzarla de manera exitosa, aún con todas las dificultades que esta representa.

A todos mis familiares que de manera directa o indirecta me apoyaron con su confianza, motivación y buenos deseos hacia mi preparación académica.

A todos mis buenos amigos y conocidos que me brindaron su compañerismo, soporte y amistad. Más aún a aquellas personas cuya compañía se ha vuelto un pilar valioso y fundamental para mi vida futura.

Vicente Quezada P.

Agradecimientos

Al Ing. Darwin Fabián Astudillo Salinas, PhD, mi director de tesis, que dedicó tiempo y paciencia a cada una de las etapas de este trabajo de titulación. Por la guía, experiencia compartida y su siempre buena voluntad de participación hacia nuevos proyectos, lo que ha permitido la formulación, desarrollo y culminación de este trabajo.

A mis padres Germán y Hermelinda, este logro no hubiera sido posible sin su soporte emocional, espiritual y económico. Su entusiasmo y devoción por la formación profesional de sus hijos ha hecho posible que consiga este logro tan importante para mi.

A todos mis hermanos, en especial a Digna cuyo ejemplo siempre ha sido un guía para todos los aspectos de mi vida. Por estar siempre allí, apoyarme e impulsarme a siempre buscar ser una mejor persona.

A la Universidad de Cuenca y a los docentes de la Escuela de Electrónica y Telecomunicaciones, que durante todos estos años de aprendizaje nos impartieron valiosos conocimientos, experiencias y valores que se convertirán en las bases para mi vida profesional.

A todas las amistades formadas a lo largo de la etapa universitaria, por hacer que esta experiencia sea una de las mejores, logrando mezclar lo académico y lo social en un álbum de inolvidables recuerdos.

Vicente Quezada P.

Abreviaciones y Acrónimos

A Dirección IPv4. 51
AAAA Dirección IPv6. 51
ANIDS Sistema de Detección de Intrusiones de Red Basado en Anomalías. 23, 32

C&C Comando y Control. 1, 2, 17, 26, 27, 29–32, 38, 69
CEDIA Corporación Ecuatoriana para el Desarrollo de la Investigación y la Academia. 34, 70

DDoS Denegación de Servicio Distribuida. 17, 25, 28, 32
DGA Algoritmo de Generación de Dominio. 1, 2, 4, 5, 9, 30–32, 38, 39, 41, 42, 46, 51, 52, 58, 59, 69–71
DNS Sistema de Nombres de Dominio. 1, 2, 4–7, 9, 18–20, 26–32, 34–39, 44–46, 48–58, 61, 62, 65, 68–70, 75, 76, 78, 82, 83, 85

ELK Elasticsearch, Logstash y Kibana. 1, 2, 4, 5, 18, 37, 46, 48, 68, 69, 71–74

Flow Flujo. 36, 71
FQDN Nombre de Dominio Completo. 4, 28, 29, 39
FTP Protocolo de Transferencia de Archivos. 36, 71

GB GigaByte. 54

HTTP Protocolo de Transferencia de Hipertexto. 26, 36, 71

IDS Intrusion Detection System. 4, 17–19, 21, 23, 24, 35–37, 44, 69, 70, 72
IP Protocolo de Internet. 4, 9, 18, 26–29, 32, 37–39, 46, 51, 54, 57–59, 61, 69, 75, 76, 79
IPv4 Protocolo de Internet versión 4. 29, 51, 76
IPv6 Protocolo de Internet versión 6. 29, 51, 76
IRC Internet Relay Chat. 26

JSON Notación de Objetos de JavaScript. 36, 48

Kill Chain Cadena de Eliminación. 21, 32

ML Aprendizaje Automático. 7, 17, 19, 23, 55, 57, 68, 70
MQTT Message Queue Telemetry Transport. 36, 71
MX Intercambio de Correo. 38, 51, 56, 58

NAD Detección de Anomalías en la Red. 17–19, 44

NIDS Sistema de Detección de Intrusos de Red. 21, 35
NXDOMAIN Dominio No Existente. 39, 51, 58

OISF Open Information Security Foundation. 35
OS Sistema Operativo. 48, 72

P2P Peer to Peer. 26, 27
PCA Análisis de Componentes Principales. 7, 51, 56, 59, 85, 86
PTR Puntero. 38, 51, 58, 62

SEM Gestión de Eventos de Seguridad. 36, 37
SIEM Gestión de Eventos e Información de Seguridad. 36, 37, 69
SIM Gestión de la Información de Seguridad. 36, 37
SLD Dominio de Segundo Nivel. 8, 9, 37, 38, 48, 51, 58–62, 65, 68, 75, 76
SSH Secure SHell. 36, 71

TLD Dominio de Nivel Superior. 5, 37, 38, 42, 48, 58, 75, 76
TLS Seguridad de la Capa de Transporte. 36, 71
TTL Tiempo de vida. 28

UDP Protocolo de Datagrama de Usuario. 29
URL Localizador Uniforme de Recursos. 38

Capítulo 1

Introducción

Este capítulo presenta la identificación del problema, justificación y los objetivos del presente proyecto.

1.1. Identificación del problema

En la actualidad, el software y los dispositivos de análisis de seguridad se deben enfocar en dos tipos de comportamiento. El primero, es el comportamiento de patrones de ataque que tiene el tráfico generado por fuentes maliciosas conocidas. Para su manejo, se hace uso de IDS [2] que, en base a reglas detecta coincidencias de patrones de ataque y genera alertas. El segundo es la seguridad basada en el flujo anómalo o no habitual que busca amenazas que no depende de las firmas malintencionadas conocidas; buscando detectar la presencia de un ataque mediante un cambio en el comportamiento normal del flujo, también siendo conocida como Detección de Anomalías en la Red (NAD) [3] . Para este último se aplican técnicas de análisis de datos basados en su historial y volumen, un ejemplo de estas técnicas es ML. Los datos almacenados permiten realizar una mejor recuperación de información relevante.

1.2. Justificación

Ataques actuales como Spam, Denegación de Servicio Distribuida (DDoS), Click Fraud, Malvertising/Ad Fraud, robo de identidad y espionaje corporativo tienen como fuente común las *botnets* [4]. Las *botnets* son grupos de *host* infectados y controladas por un atacante (*botmaster*); generalmente, los usuarios no tienen conocimiento de la existencia de una puerta trasera sobre su *host* que está siendo usada para fines delictivos, convirtiéndose así en la mayor amenaza de Internet. Las máquinas comprometidas establecen un canal de C&C desde donde se puede ejecutar actividades maliciosas no solo hacia el *host* infectado, sino usarlo para atacar a otros *hosts* de la red. En este contexto, han surgido diferentes arquitecturas de C&C y técnicas de comunicación por parte de los atacantes para dificultar cualquier proceso de detección e interrupción de canales, por lo tanto, detectar y bloquear los nombres

de dominio maliciosos interrumpirá los canales secretos entre las víctimas y los controladores, limitando así los efectos dañinos sobre los recursos de la red [5]. Aunque se realizan esfuerzos importantes para detectar *botnets* a nivel mundial, se están realizando pocos esfuerzos para detectar infecciones de *bots* a nivel empresarial [6]. La detección de máquinas infectadas por *bots* es vital para cualquier organización a la hora de combatir amenazas de seguridad para tomar las medidas correctivas adecuadas.

Trabajos como [6, 7] han mostrado la viabilidad de la detección de *hosts* infectados en la red en base al comportamiento de su flujo DNS. El protocolo DNS mapea un nombre legible por humanos en una dirección IP y viceversa. Un resultado de la aplicación de esta metodología de detección es BotDAD [8], que a partir de flujos DNS genera huellas digitales de comportamiento de cada *host* (direcciones IP del *host*, dominios consultados, promedio de consultas, entre otros), que permite, con un motor de detección de anomalías clasificar el flujo como normal o anómalo dando paso a la posibilidad de uso del aprendizaje automático para generar un modelo que permita una detección rápida de futuras huellas digitales DNS con alta precisión.

1.3. Alcance

En esta propuesta, se desarrollará una solución IDS/NAD enfocada a la detección de *botnets* en el marco del grupo de trabajo de ciberseguridad de CEDIA. El sistema será aplicado en la red de la Universidad de Cuenca. Las pruebas se desarrollarán en la infraestructura de CEDIA. Para la captura, procesamiento, obtención de los flujos DNS de red de forma pasiva e independiente de la heterogeneidad de la misma, se plantea el uso de Suricata IDS dada su robustez y popularidad [9], [10]. Para el manejo de los registros generados por Suricata IDS se prevé el uso de la pila ELK a la cual se aplicará el proceso de BotDAD para generar perfiles de comportamiento de los *hosts* de la red y clasificar los comportamientos como normal o anormal, para finalmente entrenar un algoritmo de aprendizaje automático que ayude a ahorrar tiempo y mejorar los resultados de las detecciones. Las anomalías detectadas serán enviadas al sistema de gestión de incidentes de la Universidad.

El desarrollo de la propuesta se divide en cuatro etapas fundamentales; análisis, diseño, implementación y pruebas. En la etapa de análisis se lleva a cabo un proceso de revisión de las posibles soluciones existentes actuales, y las mejoras que se pudiesen incorporar. En la fase de diseño se implementará una arquitectura basada en software libre que permita cumplir con los requerimientos del proceso previo. En la etapa de implementación se establece la funcionalidad de cada uno de los procesos. Finalmente, la etapa de pruebas se realizará en la Universidad de Cuenca. Como resultado del trabajo de investigación se presentará una solución de detección de *botnets* para la Universidad de Cuenca, que permita apoyar a la lucha global de mitigación de esta amenaza.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar y desarrollar un sistema funcional de detección de *botnets* para la Universidad de Cuenca basados en el comportamiento del flujo DNS de la red.

1.4.2. Objetivos específicos

El presente trabajo tiene los siguientes objetivos específicos:

- Realizar el estado del arte.
- Diseñar el sistema IDS/NAD trabajando sobre *logs* DNS de tráfico de red; generación, captura, almacenamiento, procesamiento y visualización.
- Diseñar e implementar un laboratorio virtual que permita realizar pruebas de forma segura.
- Detectar la presencia de *botnets* en la red en función del comportamiento del flujo DNS generado por los *hosts*, usando ML para la detección rápida de anomalías en las huellas digitales de los mismos.
- Probar y validar el sistema tanto en el laboratorio virtual como en la red de la universidad.

Capítulo 2

Marco teórico

En este capítulo se hace una revisión del marco teórico necesario para este proyecto. Se realiza una introducción hacia las anomalías del tráfico de red generado por intentos de intrusiones cibernéticas, además de los sistemas y métodos utilizados para detectar este tipo de delitos. También se realiza un análisis acerca de las *botnets* enfocado a su arquitectura, ciclo de vida e importancia de una lucha contra las mismas. Además, se realiza un breve resumen sobre los sistemas de nombres de dominio para entender de mejor manera el funcionamiento del mismo. Finalmente se adentra en las técnicas de detección de *botnets* basados en DNS que concluye en una revisión de trabajos relacionados en este campo.

2.1. Anomalías de tráfico de red

2.1.1. Tráfico de red

El tráfico de red se define como los datos presentes en una red cuando está en modo activo. En una red informática, en todos y cada uno de los momentos, los dispositivos de comunicación solicitan acceso a los recursos de diferentes servicios. Los proveedores de servicios responden a las solicitudes, aunque también es posible que un recurso no esté disponible en el momento exacto en que se realiza una solicitud [11].

2.1.2. Anomalías en la red

En un gran conjunto de datos, a menudo se hace referencia a los patrones no conformes como anomalías, valores atípicos, excepciones, aberraciones, sorpresas, peculiaridades u observaciones discordantes en diferentes dominios de aplicación. De estos, las anomalías y los valores atípicos son dos de los términos más utilizados en el contexto de la detección de intrusiones basada en anomalías en las redes [3]. Las anomalías y valores atípicos de tráfico en una red se pueden definir como cualquier evento u operación de la red que se desvíe del comportamiento normal de la misma. La detección de anomalías intenta encontrar en la red patrones de datos de tráfico que no se ajustan al comportamiento normal

esperado, esto debido a posibles ataques o intrusiones a la red. La técnica de detección de anomalías es considerada actualmente la más exitosa [11].

2.1.2.1. Ataques a la red

Un ataque a la red aprovecha una o más vulnerabilidades en una computadora o red e intenta penetrar o comprometer la seguridad del sistema. Quien lo realiza o intenta un ataque o una intrusión en un sistema es un atacante o un intruso. Se clasifica a los atacantes o intrusos en dos tipos: externos e internos; para definir a intrusos con autorización y sin autorización de acceso al sistema respectivamente [11].

Un ataque o intrusión suele ser una secuencia de operaciones que comienza con el objetivo de colocar una puerta trasera para la explotación en una red. Un atacante de red ejecuta una secuencia de pasos para lograr el objetivo deseado. El orden y la duración de estos pasos dependen de varios factores, incluido el nivel de habilidad del atacante, el tipo de vulnerabilidad a explotar, el conocimiento previo y la ubicación de inicio del atacante. La secuencia de pasos del proceso de ataque a los sistemas informáticos son conocidos como Cadena de Eliminación (Kill Chain) o proceso de ataque y se compone de un conjunto de 7 etapas [1].

1. **Reconocimiento:** el atacante recopila información sobre el objetivo.
2. **Armamentización:** el atacante crea un ataque y contenido malicioso para enviar al objetivo.
3. **Entrega:** el atacante envía el ataque y la carga maliciosa al objetivo por correo electrónico u otros métodos.
4. **Explotación:** se ejecuta el ataque.
5. **Instalación:** el *malware* y las puertas traseras se instalan en el objetivo.
6. **Mando y control:** se obtiene el control remoto del objetivo mediante un servidor o canal de comando y control.
7. **Acción:** el atacante realiza acciones maliciosas como el robo de información, o ejecuta ataques adicionales en otros dispositivos desde dentro de la red a través de las etapas de la cadena de eliminación.

En la Figura 2.1 se presenta las etapas del proceso de la Kill Chain. Según como se avance en estas etapas existe un aumento de la cantidad de esfuerzo y costos para impedir y corregir ataques.

2.1.3. Detección de intrusos

Una intrusión intenta comprometer la confidencialidad, la integridad o la disponibilidad de un sistema, evitando los mecanismos de seguridad de un *host* o una red. Como resultado, los expertos en seguridad utilizan la tecnología de detección de intrusos para mantener segura la infraestructura de las grandes empresas.

2.1.3.1. Sistema de Detección de Intrusiones

Un sistema de detección de intrusiones, es un sistema de *software* y/o *hardware* que monitorea los eventos que ocurren en un sistema informático o una red y los analiza en busca de signos de intrusión por tráfico no deseado, es decir, malicioso. A los IDS que se enfocan en la red se los conoce como Sistema de Detección de Intrusos de Red (NIDS). Estos sistemas leen todos los paquetes o flujos

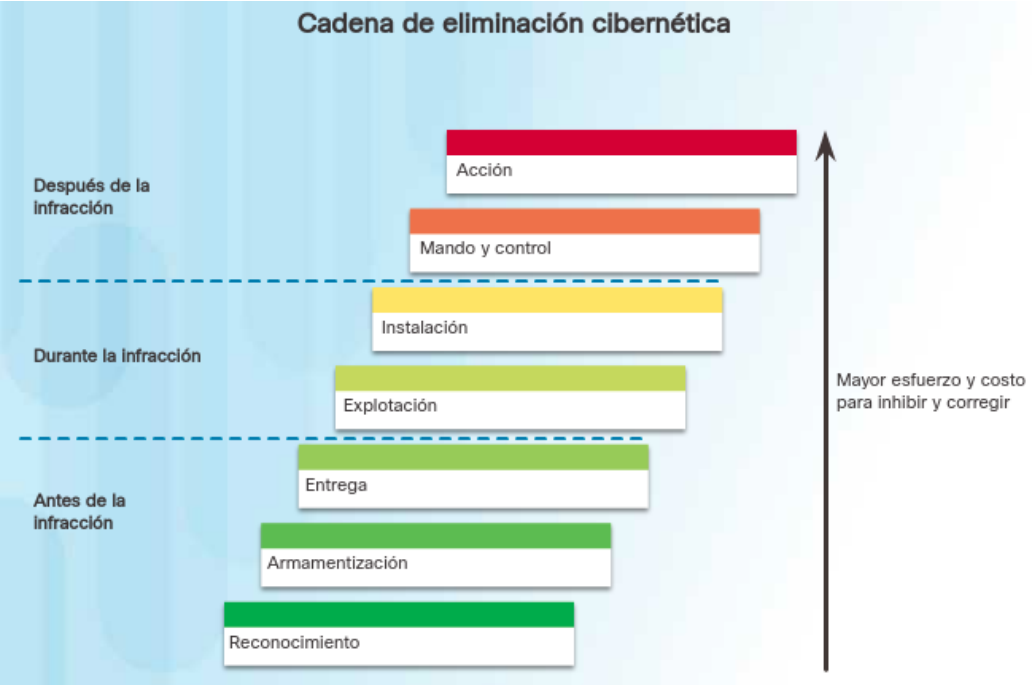


Figura 2.1: Etapas de una cadena de eliminación o proceso de ataque [1]

entrantes o salientes, tratan de encontrar patrones sospechosos. Así, los sistemas de detección de intrusos se pueden clasificar en dos categorías [11].

- **Detección de uso indebido:** Permite detectar ataques basados en firmas que se han creado y almacenado con anterioridad. Para esta categoría la actualización dinámica de firmas es importante y a su vez una limitante para detectar solo ataques conocidos pudiendo así evadir la detección de nuevos ataques.
- **Detección basados en anomalías:** Captura cualquier desviación de los perfiles de comportamiento normal de los equipos del sistema. Son más adecuados que la detección de uso indebido para detectar ataques desconocidos o nuevos sin ningún conocimiento previo. Una desventaja de estos sistemas es que generan una gran cantidad de falsas alarmas.

2.1.3.2. Importancia de los Sistemas de Detección de Intrusos

Los sistemas de detección de intrusos representan actualmente una parte indispensable de los sistemas de defensas, considerando que muchos sistemas y aplicaciones tradicionales se desarrollaron y aún se desarrollan sin tener en cuenta la seguridad. Sistemas que fueron desarrollados hace tiempo no consideraron la seguridad como un problema real. Sin embargo, el aumento actual de las amenazas y la conexión de los sistemas hacia el Internet, hace que los problemas de violación de seguridad aparezcan [11]. Así, la detección de intrusos permite detectar las brechas de seguridad que puedan aparecer, sin tener que redefinir el sistema completo.

2.1.4. Detección de anomalías y los IDS

Un sistema de detección de anomalías en la red captura y analiza el tráfico. Además, el sistema genera alarmas cuando se detecta una anomalía, y actualiza los perfiles de las instancias normales y de ataques. Un Sistema de Detección de Intrusiones de Red Basado en Anomalías (ANIDS) es un sistema para detectar anomalías en la red al monitorear el tráfico de la misma, y clasificar sus eventos como normales o anómalos. La clasificación puede basarse en heurísticas o puede buscar patrones o firmas, intentando detectar anomalías que no se encuentran en el funcionamiento normal del sistema. Idealmente, un ANIDS debería poder detectar ataques conocidos y desconocidos sin ningún conocimiento previo.

Para la detección de anomalías basadas en red, se la puede analizar en tiempo real (activo) o después de la captura (pasivo). La detección activa intenta encontrar una anomalía en tiempo real. El volumen de tráfico en la red aumenta considerablemente según avanza el tiempo, lo que dificulta enormemente el análisis en tiempo real. La detección pasiva ha recibido mucha atención, ya que implica capturar el flujo de la red para su análisis en una etapa posterior. No genera preocupación por la caída de paquetes o la complejidad computacional en tiempo real. De una manera pasiva, se extraen varios parámetros de flujo y se analizan utilizando un modelo de aprendizaje automático [4].

2.1.4.1. La detección de anomalías y el aprendizaje automático

El aprendizaje automático (ML) es una rama de la inteligencia artificial, que permite que las máquinas aprendan sin ser expresamente programadas para ello. Este tipo de inteligencia es una habilidad indispensable para hacer sistemas capaces de identificar patrones entre los datos y hacer predicciones.

De esta manera el aprendizaje automático es utilizado también para detectar intrusiones o anomalías en el tráfico de la red. La elección del modelo dependerá del tipo de datos que se tengan de la red. A continuación se presenta los tipos de modelos de ML [3].

- **Detección supervisada:** detecta anomalías en la red utilizando conocimientos previos. Se construye un modelo predictivo para clases normales y anómalas, así cada nueva instancia se compara con este modelo para definir su clase. Un evento u objeto se detecta como anómalo si su grado de desviación con respecto al perfil o comportamiento del sistema, especificado por el modelo de normalidad, es suficientemente alto.
- **Detección semi-supervisada:** se construye un modelo usando instancias etiquetadas solo para la clase normal. Las instancias de datos no están etiquetadas para la clase de ataque. El modelo se usa comúnmente para una comparación con los enfoques supervisados o para identificar anomalías en los datos de prueba.
- **Detección no supervisada:** se utiliza para la detección de intrusiones novedosas sin conocimiento previo utilizando datos puramente normales. La detección de anomalías de red sin supervisión funciona bien debido a la no disponibilidad de datos etiquetados o puramente normales. Se reduce el gasto que significa la clasificación manual de un gran volumen de tráfico de red. El modelo requiere un profundo análisis de los datos ya que debido a la falta de datos etiquetados, los resultados pueden ser menos útiles.
- **Detección híbrida:** combina enfoques supervisados y no supervisados de detección de anomalías en la red. Tales enfoques pueden detectar ataques conocidos y desconocidos. Un enfoque híbrido

intenta identificar ataques conocidos basándose en un modelo supervisado de un conjunto de datos, con un mecanismo de coincidencia apropiado. Las instancias de prueba que no pertenecen ni a la normal ni a ninguna de las instancias de ataque conocidas, son manejadas por el modelo no supervisado para la identificación de nuevas intrusiones.

Los dos primeros modelos requieren capacitación sobre instancias etiquetadas para encontrar anomalías. Obtener una gran cantidad de instancias etiquetas de entrenamiento para ataque y normales, puede no ser práctico en muchas o la mayoría de las situaciones. Generar un conjunto de instancias normales verdaderas con todas las variaciones, es una tarea extremadamente difícil. Para todas las técnicas se suele hacer la suposición implícita de que las instancias normales son mucho más frecuentes que las anomalías en los datos de prueba.

2.1.4.2. Arquitectura de un sistema de detección de anomalías de la red.

Las arquitecturas de los sistemas de detección de las anomalías de la red tienen módulos comunes. La forma genérica de la arquitectura es presentada en la Figura 2.2 y se detalla a continuación.

- **Captura de tráfico:** es un módulo importante en cualquier IDS de detección de anomalías. En este módulo, el tráfico de red en vivo se captura utilizando algún tipo de *sniffer*.
- **Procesamiento de información:** Un paquete en vivo capturado contiene una gran cantidad de datos sin procesar. Este módulo filtra los parámetros relevantes durante la captura y extracción de características de los datos capturados. Este módulo además se encarga de la conversión, normalización y discretización del tipo de datos filtrados; para convertirlos en información con un formato legible para otros módulos.
- **Motor de detección de anomalías:** este módulo es el corazón de cualquier sistema de detección de anomalías en la red. Intenta detectar la ocurrencia de cualquier intrusión ya sea en línea o fuera de línea. Si el ataque pertenece a un tipo conocido, se puede detectar mediante un enfoque de detección de uso indebido. Los ataques desconocidos se pueden detectar con el enfoque basado en anomalías, utilizando un mecanismo de coincidencia apropiado o un clasificador.
- **Alarma:** este módulo es responsable de la generación de una alarma en base a la indicación recibida del motor de detección de anomalías. A más de indicar la ocurrencia de un ataque, las alarmas son útiles para un análisis posterior. Las alarmas deben indicar cualquier información de fondo que justifique por qué se generó la alarma.
- **Analista humano:** es responsable del análisis, interpretación, y de la toma las acciones necesarias basadas en la información de alarma proporcionada por el motor de detección. El analista también, sigue los pasos necesarios para diagnosticar la información de alarma como una actividad de pos-procesamiento para respaldar la actualización de la referencia o el perfil.

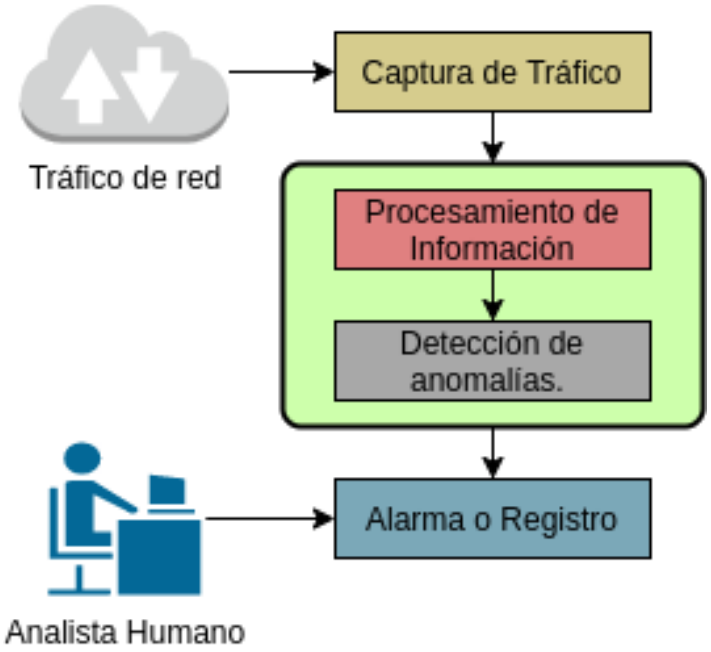


Figura 2.2: Arquitectura genérica de un sistema de detección intrusiones basado en anomalías de la red.

2.2. *Botnet*

El diccionario de Cambridge[1] define a un *bot* como "un programa informático que funciona de forma automática, especialmente uno que busca y encuentra información en Internet". Mientras, que una *botnet* se define como "un grupo de computadoras controladas por *software* que contiene programas dañinos, sin el conocimiento de sus usuarios".

En otras palabras, una *botnet* es una "Red de *bots*" o grupo de computadoras conectadas a través de Internet con la capacidad de ser controlados por un individuo o grupo malicioso. Una *botnet* puede tener decenas de miles o incluso cientos de miles de *bots*. Estos *bots* se pueden activar para distribuir *malware*, lanzar ataques DDoS, distribuir correo electrónico no deseado, realizar ataques de contraseña por fuerza bruta o las estafas de *phishing*. Si una computadora es parte de una red de *bots*, significa que ha sido infectada con un *malware*, como puede ser un virus, gusanos, *bots*, etc. Una computadora *bot* se infecta generalmente por visitar un sitio web, abrir un elemento adjunto de correo electrónico o abrir un archivo de medios infectado [1].

2.2.1. Arquitectura de una *Botnet*

La arquitectura común de una *botnet* es presentada en la Figura 2.3 y consta de los siguientes componentes:

- ***Bots*:** son dispositivos conectados a Internet que están infectados con *malware* y se controlan de forma remota.

[1] https://dictionary.cambridge.org/es/

- **C&C *Server*:** es un servidor que controla los *bots* en la red a través de comandos de transmisión.
- ***Botmaster:*** es una persona que toma el control de los servidores de C&C y envía comandos a los *bots* de la *botnet*.

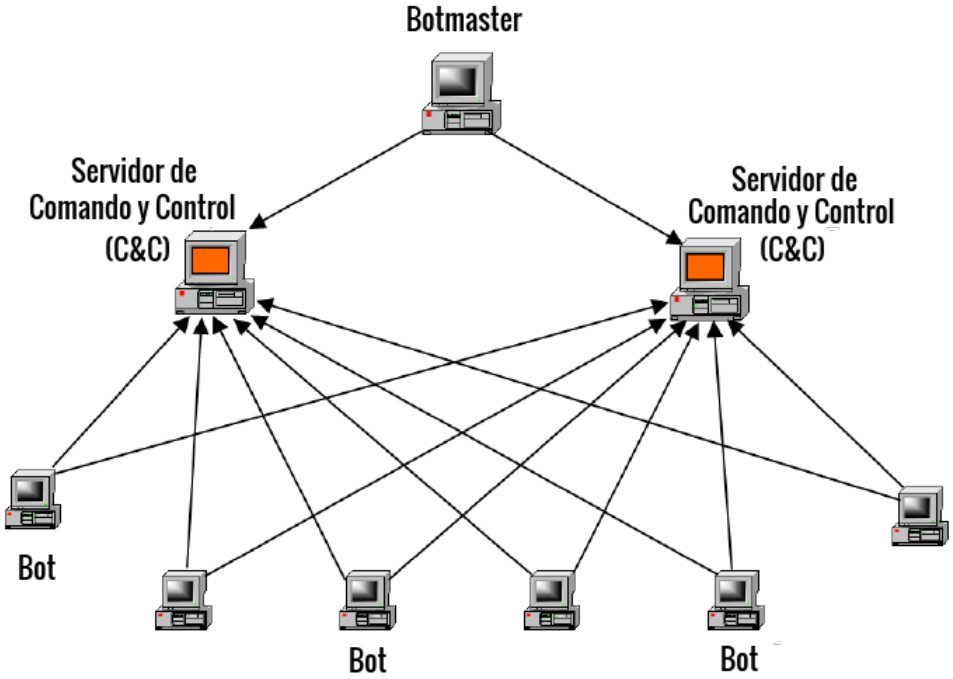


Figura 2.3: Arquitectura una *botnet*.

Para evitar la vigilancia de los sistemas de detección de intrusos, los *botmasters* tienden a utilizar protocolos y aplicaciones de comunicación comunes para conectarse a los *bots* mediante canales C&C. Entre los protocolos comunes se encuentran: Internet Relay Chat (IRC), Protocolo de Transferencia de Hipertexto (HTTP), DNS o Peer to Peer (P2P) [12]. De esta manera, la detección, prevención e investigación de *botnets* se vuelve más difícil.

2.2.2. Ciclo de vida de una *botnet*

El comportamiento de las *botnets* siguen un patrón general que puede ser descrito en 4 pasos [13], el flujo de las fases se presenta en la Figura 2.4:

- **Fase de explotación:** el *botmaster* infecta un *host* víctima explotando una vulnerabilidad existente de *software*. La infección va de la mano con algún tipo de fraude hacia la víctima, para que esta ejecute un código malicioso en su máquina. En esta fase, los *bots* necesitan conectarse a un servidor remoto. La conexión al servidor se establece solo después de que la máquina comprometida emita un comando de búsqueda de DNS, buscando asignar una dirección IP a su servidor remoto.
- **Fase de reunión:** los *bots* se están conectando continuamente a su *botmaster* mediante un servidor C&C. El *botmaster* equipa a sus *bots* con una funcionalidad de búsqueda de DNS para

poder realizar consultas para ubicar el servidor de C&C. Para evitar que las IPs del servidor sean incluidas en listas negras de seguridad, las IPs migran constantemente ocultando así su dirección detrás de un nombre de dominios. Como resultado, los *bots* se seguirán comunicando con el *botmaster* siempre que la petición DNS hacia el servidor de C&C sea resuelta.

- **Fase de ejecución de ataque:** el grupo de *bots* realiza actividades maliciosas en las máquinas objetivo mediante la recepción de comandos de los servidores de C&C proveídas por el *botmaster*.
- **Fase de actualización y mantenimiento:** el *botmaster* necesita mantener sus *bots* actualizados, instruyendo así a los *bots* para que actualicen sus binarios de vez en cuando para una mejor coordinación. Además, los *botmasters* pueden requerir migrar la ubicación de su servidor C&C con frecuencia para evadir las diversas técnicas de detección de *bots*.

Figura 2.4: Ciclo de vida de una *botnet*.

Analizando del ciclo de vida de una *botnet*, se observa que la detección de *botnets* basada en el análisis del tráfico de DNS tiene el potencial de detectar *botnets*. Así, esta técnica es considerada un campo de investigación prometedor para combatir las amenazas de *botnets*.

2.2.3. Importancia de la lucha contra las *botnets*

La evolución de los ataques cibernéticos permitió dar un salto desde tener unos ataques generales y desorganizados, a tener en la última década ataques muy sofisticados. Este mayor nivel de coordinación es posible principalmente debido a las *botnets*. El aumento continuo de *bots* y sus ataques, puede paralizar gravemente las empresas, los servicios y las operaciones.

El trabajo realizado por [14] presenta algunas de las botnets más conocidas y sus ataques. Zeus que fue interrumpida en 2010 logró infectar alrededor de 3.6 millones de *hosts*. La intrusión hacía uso de *ransonware* Cryptolocker para recopilar información financiera y credenciales bancarias. Luego, la *botnet* fue utilizada para ataques de *pishing* y correo no deseado. Srizbi fue utilizada para la inundación de *spam*. Se detectó un aproximado de 450.000 *bots* infectados que generaban 60 millones de mensajes al día. Srizbi fue luego utilizada con fines políticos. Gameover Zeus en el 2014 fue un sucesor de Zeus que usó un enfoque P2P y la generación de dominios aleatorios, para evadir los controles de seguridad. Gameover Zeus logró generar de 1000 a 10000 dominios cada día de manera aleatoria. Methbot en 2016 fue una *botnet* enfocada en el fraude publicitario. Hizo uso de servidores que podían producir *clics* y movimientos del ratón falsos, así como falsificar inicios de sesión de cuentas de redes sociales. De esta manera, se simuló varios usuarios legítimos para engañar a las técnicas convencionales de

detección de fraude publicitario. Otra *botnet* de importancia fue Mirai a principios de 2017 que generó ataques DDoS poderosos. Los *bots* eran una variedad de dispositivos conectados como enrutadores inalámbricos y cámaras de circuito cerrado de televisión. El tráfico de los dispositivos era desviado hacia los distintos objetivos de ataque.

Se le debe prestar atención continua a este tipo de delincuencia cibernética. Mientras se coordina a nivel mundial el cierre de una determinada *botnet*, aparece una nueva con un nombre diferente que emplea nuevos vectores de ataque y utiliza un conjunto diferente de dispositivos. Las *botnets* son una amenaza persistente que cambia de nombre y vuelve más fuerte repentinamente [4]. De esta manera, la detección y mitigación de *botnets* en una red local apoya de manera directa a la lucha global por mitigar esta amenaza. Evitando así, que los recursos de una empresa sean vulnerados o utilizados para delinquir otras organizaciones.

2.3. Sistema de nombres de dominio (DNS)

El sistema de nombre de dominio es un sistema de nomenclatura jerárquico. La función principal del sistema consiste en resolver las peticiones de asignación de nombres de dominios totalmente calificados o FQDN. Se puede hacer una analogía de un servidor DNS con una agenda telefónica, las personas buscan con un nombre un número telefónico relacionado al mismo. De la misma manera, el DNS proviene del hecho de que los humanos prefieren los nombres a los números. Por ejemplo, es mucho más fácil de recordar *www.google.com* que *172.217.17.36*, sin embargo, este último es la dirección con la que las máquinas se comunican entre si. La arquitectura del sistema de nombres de dominio sigue un modelo cliente-servidor. El cliente envía una solicitud al servidor DNS local. El servidor analiza y responde la solicitud enviando la dirección IP asignada al nombre de dominio junto con un valor de Tiempo de vida (TTL). Con la dirección IP obtenida se realiza la comunicación hacia el servidor que tiene el recurso solicitados, un ejemplo de este funcionamiento es presentado en la Figura 2.5.

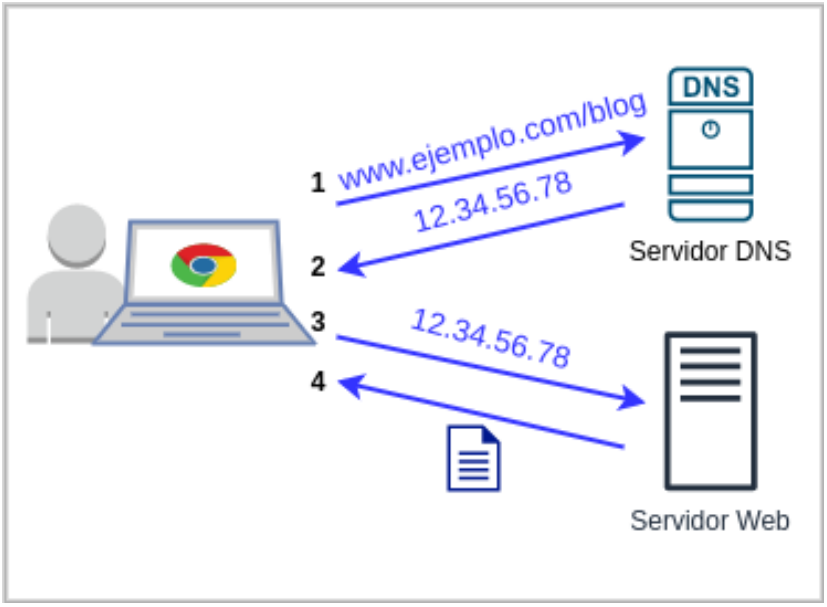


Figura 2.5: Consulta DNS recursiva.

2.3.1. Registros DNS

El sistema de nombres de dominio utiliza un protocolo de consulta/respuesta. El protocolo de trasporte generalmente usado es Protocolo de Datagrama de Usuario (UDP) y se hace uso del puerto 53. Así, los paquetes de consulta y respuesta (*query* y *answer* respectivamente) tienen en su encabezado campos propios de paquetería como IP o puertos de origen y destino. Lo que marca la diferencia son los registros DNS del cuerpo de la consulta. Los registros más comunes de DNS [15, 16] se muestran en la Tabla 2.1.

Tabla 2.1: Registros DNS más comunes.

Tipo	Significado	Contenido
A	Registro de asignación de direcciones.	Almacena un nombre de *host* y su dirección Protocolo de Internet versión 4 (IPv4) correspondiente
AAAA	Registro de dirección IP versión 6.	Almacena un nombre de *host* y su dirección Protocolo de Internet versión 6 (IPv6) correspondiente.
CNAME	Registro de nombre canónico.	Especifica un nombre de dominio que debe consultarse para resolver la consulta DNS original.
CERT	Registro de certificado.	Almacena certificados de cifrado.
MX	Registro de intercambiador de correo.	Solicita información sobre el servidor de intercambio de correo para un nombre de dominio DNS específico.
NS	Registros del servidor de nombres.	Para obtener información sobre el servidor de nombres autorizado.
PTR	Registros de puntero de búsqueda inversa.	Especifica una consulta inversa (solicitando el FQDN correspondiente a la dirección IP que proporcionó).
SOA	Inicio de autoridad.	Se utiliza al transferir zonas.
TXT	Registro de texto.	Transporta datos legibles por máquina, como cifrado oportunista o marco de políticas del remitente.

2.4. *Botnets* y los sistemas de nombre de dominio

Como se mencionó con anterioridad, los *botnets* usan protocolos de comunicación comunes como lo es DNS. Además, el ciclo de vida de una *botnet* depende en su gran mayoría de la resolución de los nombres de dominios de los servidores de C&C. Los atacantes o *botmasters* utilizan el servicio de nombres de dominio para ocultar su dirección IP de comando y control, para que la *botnet* sea confiable y fácil de migrar de servidor a otro sin levantar sospechas. Así, las técnicas de detección de *botnets* que se basan en el análisis del tráfico de DNS son las más utilizadas en la detección de las mismas [13].

2.4.1. Técnicas de detección de *botnets* basadas en DNS

Teniendo en cuenta las características del sistema de nombres de dominio, las técnicas actuales de detección de *botnets* basados en DNS según [5] se pueden clasificar en:

- **Detección basadas en flujo:** intentan clasificar el flujo de la red en malicioso y benigno en función de parámetros inspeccionados en el flujo neto de la red.
- **Detección basadas en anomalías:** intentan encontrar una anomalía en los diversos parámetros o patrones peculiares del tráfico, que sean diferentes del comportamiento normal de la red.

- **Detección basadas en DGA:** intentan diferenciar los dominios consultados en una red que se generan algorítmicamente (malignos) de los dominios normales.
- **Detección de *hosts* infectados por *bot*:** intenta detectar máquinas infectadas por *bots* en una red en lugar de encontrar el servidor C&C. El enfoque principal de las técnicas es enumerar los *hosts* infectados en una red.

Las detección basada en DGA y la detección de infecciones de *bot* son las técnicas más revisadas en la actualidad. Desde la perspectiva presentada por [13], la evolución de la detección de *botnets* basado en DNS se la puede resumir en la clasificación presentada en la Figura 2.6. Las técnicas finales de la figura hace referencia a herramientas de análisis de información de registros capturados.

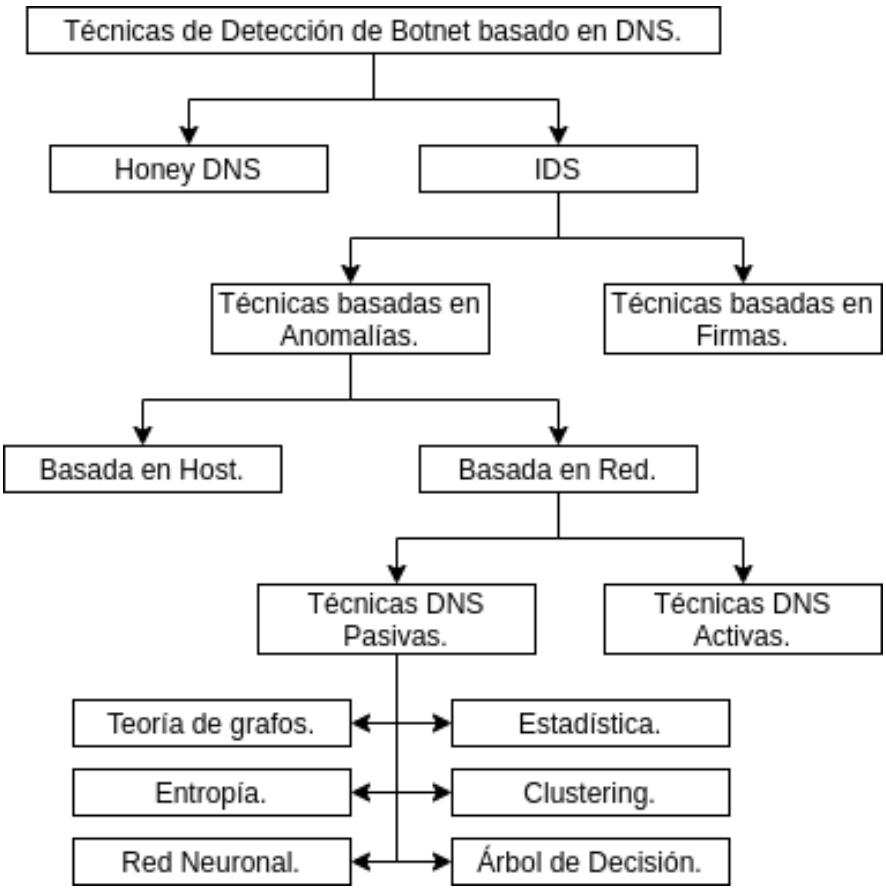


Figura 2.6: Resumen del enfoque actual de las técnicas de detección de *botnets* basados en DNS.

2.4.2. Algoritmo de generación de dominio (DGA)

Un DGA es una técnica en la que el nombre de dominio se cambia con frecuencia, para ocultar la comunicación de devolución de llamada del *bot* al servidor C&C [17]. En los *bots* el atacante solo debe registrar uno o más dominios por adelantado. El *bot* genera dinámicamente una gran lista de nombres de dominio basados en DGA, e intenta comunicarse con su servidor. Se consulta los dominios generados uno tras otro, hasta que se encuentra y resuelve uno activo. El nombre de dominio que devuelve la respuesta correcta se considera el nombre de dominio del servidor de C&C [18].

Generalmente, los dominios generados por DGA son nombres pseudoaleatorios, en el sentido de que son incomprensibles para los humanos. Los dominios no forman frases que pueda tener sentido de una dirección real. A la aleatoriedad creada por el algoritmo, se suma una longitud extensa para que no coincida con un nombre de un sitio real. En la Tabla 2.2 se muestran algunos ejemplos de dominios DGA de algunas familias capturas por el proyecto DGArchive [19].

Tabla 2.2: Ejemplo de dominios generados por DGA.

No.	Familia	Ejemplo
1	Bedep	mghhwonojhkqp4l.com
2	Blackhole	okjqzgzouwqonvup.ru
3	China	3seykl6jum4tnsd4.biz
4	Cryptolocker	fqjnvckxdkpowpb.com
5	Dnschanger	ifhfmmatpw.com
6	Gameover	ytmzofpdyrcswcmqovcirhpnib.com
7	Locky	xgrdauffnegeagjlh.org
8	Mirai	gefrdypxyspb.tech
9	Murofet	glxmnytxfwwitw.net
10	Tinba	ctkllmvvvnbb.biz

2.4.3. Trabajos relacionados.

Una revisión realizada en [13] se presenta como la primera revisión bibliográfica que analiza las técnicas de detección de *botnets* basadas en DNS. Explora las soluciones y dirige la investigación futura hacia técnicas pasivas de detección de *botnets* basadas en el análisis de tráfico de sistemas de nombre de dominio, para lograr mecanismos efectivos de detección. En este mismo escenario en [5] se realiza una revisión enfocada en el estado actual del arte de los problemas y desafíos de la detección de *botnet* basado en sistemas de nombre de dominio. Esta revisión presenta una nueva clasificación para las técnicas de detección de *botnets* basadas en DNS. Las técnicas se catalogan como detección basada en flujo, detección basada en anomalías, detección de comunicaciones DGA y detección de *hosts* infectados por *bots* en una red. Se hace énfasis en los dos últimos métodos, como los nuevos enfoques de investigación. Con esta última clasificación se han presentado algunos trabajos que se presentarán a continuación.

El trabajo presentado en [12] se basa en la detección de *botnets* con las técnicas basadas en flujo y anomalías del tráfico de red. Se extraen un total de 23 características del flujo DNS neto a nivel de protocolo. Para luego, con el uso de aprendizaje automático determinar si el nombre de dominio es legítimo o no, y así detectar *botnets*. De los algoritmos de aprendizaje automático aplicados *Ramdom Forest* alcanzó una precisión(relación entre las predicciones correctas obtenidas y las previstas) del 99 %. DBod se presenta en [4] como una herramienta que aprovecha el comportamiento de una comunicación DGA. Supone que de varios dominios consultados, solo un número limitado de dominios están realmente asociados con un C&C activo. Por lo tanto, el trabajo busca anomalías de aumento de consultas fallidas o NXDomain por medio de técnicas de agrupación.

En [17] se presenta un trabajo de detección de dominios DGA a partir de consultas DNS masivas. Se hace uso de 6 diccionarios de lenguaje: inglés, francés, alemán, español, ruso y japonés. En base a los diccionarios se estima la aleatoriedad de los dominios mediante un análisis léxico, consiguiendo una precisión del 90 %. En el campo de la detección DGA, también se han desarrollado otros proyectos

como [18] que es una mejora del algoritmo de detección de aleatoriedad presentado en [20]. El trabajo considera un algoritmo determinista independiente del idioma. Se busca detectar nombres de dominio incomprensibles basados en axiomas sónicos generales, como es el número de vocales consecuentes, número de consonantes consecuentes y la entropía del dominio. El algoritmo consigue valores de precisión por encima del 90 %, dependiendo de la familia analizada. El método es planteado como una primera alarma eficiente de reconocimiento de comunicaciones de *botnets*.

Para la detección de *hosts* infectados por *bots*, en [21] se presenta una metodología de detección de comportamiento de anomalías de *hosts*. Se busca cadenas de patrón de ataque o Kill Chain para detección de canales C&C, *pharming* y *botnet* DDoS de suplantación de IP. El trabajo realiza la creación de perfiles de comportamiento de cada uno de los *hosts* de la red. Sin embargo, se considera todo el tráfico de la red, lo que demanda una gran cantidad de almacenamiento y capacidad computacional. En [22] se hace una introducción al término huella digital o huella dactilar DNS como el perfil de comportamiento de comunicación DNS de un *host*. Cada huella digital de comportamiento identifica de forma única a su usuario correspondiente y es inmune al cambio de hora. De esta manera, un usuario tiene un patrón de comportamiento DNS invariable en la red. Con las huellas de los *hosts* generadas se busca detectar la presencia de nuevos flujos que vendrían a ser comportamientos anormales en la red. BotDAD en [6] es presentado como una nueva técnica de detección basada en anomalías, que considera la huella digital DNS de los *hosts* por hora e intenta encontrar un comportamiento anómalo de una máquina. Las huellas digitales son generadas con la extracción de 15 características del flujo DNS para cada *host*. Con las huellas generadas se da paso a un motor de detección basado en búsqueda de anomalías. Finalmente, se entrena al algoritmo de aprendizaje automático *Random Forest* para que clasifique los *host* limpios e infectados de una forma más rápida en futuras capturas. La continuación del trabajo realizado en [6] es presentado en [7]. Por la alta precisión del motor de detección de anomalías de BotDAD, se centra en mejorar el algoritmo de aprendizaje automático para futuras huellas. Así, se crea una nueva herramienta llamada DeepDAD que hace uso de redes neuronales multicapa.

2.5. Conclusiones

El tráfico de red que se define como los datos presentes en una red cuando está en modo activo. Estos datos pueden presentar anomalías o valores atípicos referentes a intrusiones de seguridad. Los valores atípicos son datos generados por aplicaciones maliciosas presentes en los *hosts* de la red. La técnica de detección de anomalías se ha desarrollado para poder encontrar estos patrones de datos de tráfico, cuyo comportamiento no se ajusta al comportamiento normal. A su vez, la detección de anomalías ha crecido de la mano de herramientas de aprendizaje automático, como motor de detección de anomalías. Un sistema que abarca las características antes mencionados son los denominados ANIDS. Los ANIDS hacen referencia a los sistemas de detección de intrusos en la red basado en anomalías.

Las *botnets* son un grupo de computadoras conectadas a través de Internet, con la capacidad de ser controlados por un individuo o grupo malicioso. Estas han permitido la evolución de los ataques cibernéticos a ataques más sofisticados y coordinados, como lo son los ataques de día cero. El ciclo de vida de una *botnet* depende en su gran mayoría de la resolución de los nombres de dominios de los servidores de C&C, para los *hosts* infectados. Así, la detección de *botnets* basado en el análisis del tráfico DNS, se convierte en un método con potencial para detectar este tipo de intrusiones. Entre las técnicas de detección de *botnets* basados en DNS, las técnicas de detección basada en DGA y la

detección de infecciones de *hosts* por *bots*, se muestran como los campos de investigación actuales.

Capítulo 3

Análisis y Diseño del Sistema

En este capítulo se presenta la fase de análisis de los requerimientos del sistema y su diseño. Se examina la topología actual de la red, y se describe las herramientas y procesos para los módulos de: captura de eventos, gestión y análisis de eventos, detección de anomalías y aprendizaje automático, con el fin de obtener los resultados deseados. De las herramientas y procesos descritos, se hace énfasis en las escogidas para la implementación con su respectiva justificación. Finalmente se bosqueja la arquitectura final del sistema a ser implementado.

3.1. Topología de la red

La red sobre la cual se propone el sistema es una red inter-universitaria. Dicha red está gestionada por Corporación Ecuatoriana para el Desarrollo de la Investigación y la Academia (CEDIA) e interconecta y da acceso a Internet a las universidades del país, además de poseer tráfico interno propio. En la Figura 3.1 se muestra la topología de la red, en la cual, sobre el tráfico que circula en el *switch* se desea realizar la captura de eventos DNS. Los eventos capturados serán procesados y analizados con la finalidad de encontrar comportamientos que denoten actividad por parte de *hosts* infectados por *bots*.

Dada la naturaleza de la red y del sistema que se desea implementar, se considera colocar al sistema que actúe de forma *offline* al trabajo de la red. El sistema se centrará en la captura y análisis de tipo pasivo sobre los paquetes que circulan en la red. Se busca así, evitar que la ejecución del sistema pueda causar algún fallo, retraso o pérdida de la comunicación entre los diferentes equipos de la red.

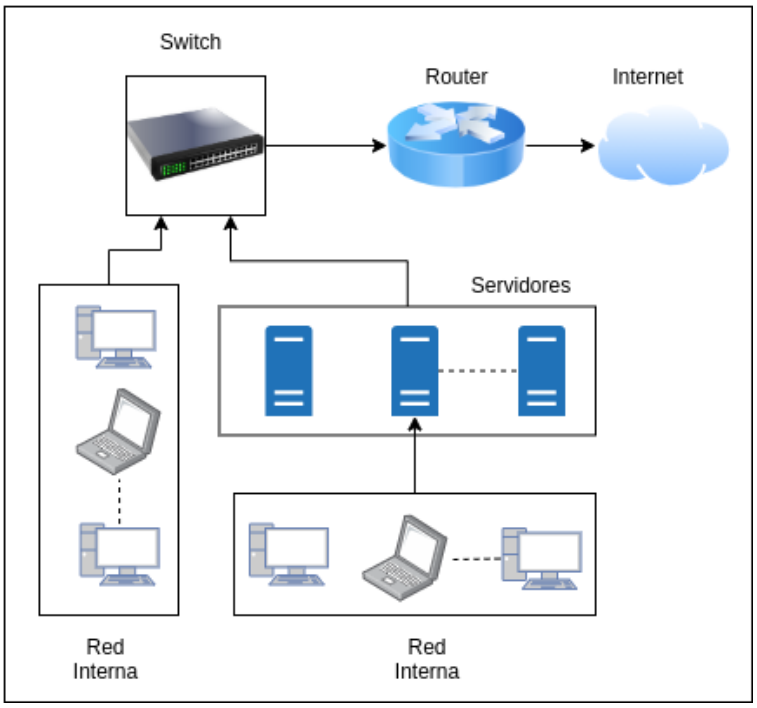


Figura 3.1: Topología de la red.

3.2. Captura de los datos

3.2.1. Captura de eventos DNS

Para la captura de eventos DNS para el sistema, se tienen dos tipos de herramientas disponibles: enfocadas en captura de paquetes y los IDS.

Las herramientas de captura de paquetes o *sniffers* de red, interceptan y recolectan el tráfico al que tienen acceso. Luego de capturar el tráfico, el *sniffer* provee la posibilidad de decodificarlo y realizar un análisis del contenido de los paquetes. Un *sniffer* busca mostrar los resultados obtenidos en un formato que sea legible para los humanos [23]. Herramientas populares de este tipo son: Wireshark, Microsoft Message Analyzer, TcpDump, WinDump, entre otras.

Un IDS en sistemas informáticos se encarga de capturar y monitorizar datos que circulan por la red, y generar *logs* o eventos en base a un conjunto de reglas de seguridad definidos con anterioridad. Los IDS que analizan el tráfico de red se denominan NIDS [9]. Herramientas populares de este tipo son: Suricata, Snort, Bro, Zeek, entre otras.

Para este proyecto se considera la integración de una herramienta IDS, teniendo como principal motivo el constante desarrollo que han tenido. Con las integraciones de más herramientas, los eventos generados por los IDS contienen más información que la perteneciente a la sola captura de paquetes. Un ejemplo de información adicional de paquetes es la información de referencia geográfica, que es útil para un análisis basado en localización.

De los IDS existentes, Suricata[1] es un motor de detección de amenazas de red maduro, rápido y robusto, gratuito y de código abierto. El proyecto es propiedad y está respaldado por la Open

[1] https://suricata-ids.org/

Information Security Foundation (OISF). Entre las ventajas de Suricata sobre otras IDS se encuentran las siguientes:

- IDS maduro con varios proyectos en la red y soporte.
- Procesamiento multi-hilo que permite ejecutar varios procesos a la vez. Es ideal para inspeccionar el tráfico en una red de varios gigabits, permitiendo escalabilidad al sistema.
- Tiene formatos de entrada y salida estándar tipo Notación de Objetos de JavaScript (JSON). Su integración con herramientas de Gestión de Eventos e Información de Seguridad (SIEM) existentes como: Splunk, Logstash/Elasticsearch, Kibana y otras bases de datos, se vuelve fácil.
- Los eventos[2] que genera de forma automática en función de protocolos tras su instalación son: HTTP, DNS, Protocolo de Transferencia de Archivos (FTP), Seguridad de la Capa de Transporte (TLS), Secure SHell (SSH), Flujo (Flow), Message Queue Telemetry Transport (MQTT). De esta manera, los eventos DNS requeridos para el proyecto están incluidos para ser generados.

Con las consideraciones anteriores, Suricata es elegido como la herramienta para la fase de captura de eventos DNS para el proyecto.

3.2.2. Modo promiscuo de una interfaz de red

En una red ethernet, cada dispositivo solo acepta los paquetes que van dirigidos hacia el propio dispositivo y a la dirección de *broadcast* de la red. Dado que es necesario para el proyecto analizar todo el tráfico que por la red circula, el dispositivo que va a actuar como IDS deberá establecer su interfaz de red en modo promiscuo. La configuración logra que el dispositivo se comporte como un *sniffer*, para posteriormente generar los eventos deseados.

3.2.3. Port Mirroring

En las topologías actuales, el *switch* que maneja las interconexiones de la red detecta que dispositivo esta conectado en cada una de sus salidas, permitiéndole así reenviar los paquetes de red directamente a cada dispositivo mejorando el manejo del tráfico. Sin embargo, este es un inconveniente a la hora de conectar un IDS al *switch*, ya que no recibiría tráfico excepto el destinado al mismo. Se debe activar la función de *port-mirroring* en la salida del *switch* hacia el IDS, con el objetivo de recibir un duplicado del tráfico en esta interfaz. Con esto, se consigue un tráfico reflejado que permite monitorizar y analizar los paquetes sin influir en su circulación.

3.3. Gestión y análisis de los eventos DNS

Una vez que se ha definido las herramientas de captura de eventos DNS, es necesario la elección de las herramientas que transformarán estos datos en información útil.

3.3.1. Gestores de eventos de seguridad

Junto con la implantación de un IDS, suele ser necesaria la instalación de un SIEM. El significado del término SIEM es una combinación de las siglas de Gestión de la Información de Seguridad (SIM) y Gestión de Eventos de Seguridad (SEM).

[2] https://suricata.readthedocs.io/en/suricata-6.0.0/output/

- **SEM:** monitoriza y correlaciona eventos de seguridad en tiempo real.
- **SIM:** analiza y monitoriza datos almacenados durante un periodo de tiempo.

Implantar un SIEM significa tener un sistema que permita gestionar la información de la seguridad y eventos. Sobre el SIEM se recopila, analiza y presenta los eventos capturados por el IDS, transformándolos en información fácilmente accesible [9].

En este proyecto, la implantación de un SIEM se considera necesaria, dado que se tendrá que analizar datos almacenados durante un periodo de tiempo. Además, en función de los atributos de los eventos, se requiere formar huellas digitales DNS de los *hosts* de la red. Se tiene como prerrequisito herramientas de libre distribución con flexibilidad, y que dispongan de un buen soporte en la comunidad. Algunas de las herramientas disponibles para la gestión y análisis de eventos de un IDS son: Zabbix, Snorby, Splunk, pila ELK.

3.3.1.1. La pila ELK

La sigla ELK es la unión de las siglas de tres proyectos open source: Elasticsearch, Logstash y Kibana. Elasticsearch es un motor de búsqueda y analítica. Logstash es un *pipeline* o "embudo" para la ingesta de *logs* y su posterior transformación y envío a un destino. Kibana es una interfaz de visualización de datos que permite generar información sobre distintos cuadros y gráficos.

La pila ELK goza de gran popularidad en la comunidad de Internet. La pila es una integración que puede utilizarse para recoger *logs* de todo tipo, por lo que cuenta con una gran robustez. De esta manera, la integración de la pila como SIEM para Suricata IDS es solo una de sus aplicaciones.

SELKS[3] es un sistema de libre distribución de Stamus Networks basado en Debian. El sistema integra Suricata y la pila ELK, además de otras herramientas, para la gestión de eventos de seguridad en la red. SELKS y los proyectos descritos en [9, 10] son claves para la elección de la pila ELK como SIEM para este proyecto. La popularidad de SELKS genera gran expectativa en el uso de Elasticsearch, Logstash, Kibana y Suricata como conjunto. La pila ELK contribuye al proyecto de la siguiente manera:

- **Logstash:** Recoger, procesar y enviar los *logs* de Suricata a Elasticsearch. Añadir información adicional de geolocalización en función de la IP a cada *log*. Obtener el TLD y SLD de los nombres de dominio consultados.
- **Elasticsearch:** Base de datos para almacenar los registros enviados por Logstash. Motor de búsqueda y analítica para futuras consultas.
- **Kibana:** Interfaz gráfica que permite interactuar de una manera más amigable con los datos y laboratorio de consultas.

3.4. Detección de anomalías

En este apartado se analiza la metodología y las herramientas para una vez culminada la etapa de captura y procesamiento de lo eventos DNS, se generen las huellas digitales DNS de los *hosts* y la detección de anomalías en los mismos.

[3] https://github.com/StamusNetworks/SELKS

3.4.1. Generación de huellas digitales de los *hosts*

Siguiendo la metodología presentada en [6] los atributos de una comunicación DNS que pueden denotar comportamiento de *botnets* en un *host* se pueden clasificar en atributos de: solicitud, respuesta, dominio, IP y tipo de mapeo. En total son necesarios 15 atributos, que se denominan con los nombres de P1 hasta P15.

Se utiliza el lenguaje de programación Python para acceder como cliente externo a Elasticsearch. Como cliente se puede aprovechar el motor de búsquedas y análisis de elasticseach para obtener los atributos DNS requeridos para cada *host*. De esta forma se generarán las huellas digitales DNS de los *hosts* por horas.

3.4.1.1. Atributos basados en solicitudes DNS

- **Número de solicitudes DNS por hora (P1):** da una perspectiva temprana de si un huésped está infectado o no. Las máquinas infectadas por *bot* tienden a tener más solicitudes por hora de lo normal.
- **Número de solicitudes de DNS distintas por hora (P2):** los *hosts* infectados con *malware* DGA tienden a tener un mayor número de solicitudes distintas que los *hosts* normales.
- **Mayor cantidad de solicitudes para un solo dominio (P3):** ayuda a detectar la existencia de un túnel DNS en el que se transfiere información confidencial a través de este protocolo.
- **Número medio de solicitudes por minuto (P4):** útil para detectar máquinas infectadas con *malware* que no utilizan ráfagas cortas de solicitudes de DNS, sino que contribuyen regularmente a las solicitudes de DNS mediante un intervalo de suspensión. Se calcula dividiendo el número de solicitudes enviadas para el tiempo que el *host* estuvo activo y utilizando el servicio de resolución de nombres de dominios.
- **Mayor cantidad de solicitudes por minuto (P5):** ayuda a detectar *bots* infectados con *malware* que utilizan una breve ráfaga de solicitudes DNS para comunicarse con el servidor C&C. Múltiples Localizador Uniforme de Recursos (URL) son generadas por los algoritmos DGA.
- **Número de consultas de registros Intercambio de Correo (MX) (P6):** es un indicador de *botnets* basadas en *spam* en la red.
- **Número de consultas de registros Puntero (PTR) (P7):** ayuda a detectar *hosts* con comportamiento anómalo en una red y una posible infección.
- **Número de servidores DNS distintos consultados (P8):** ayuda a detectar máquinas con comportamiento anómalo en la red. Es poco común que un sistema estándar consulte a más de un servidor DNS.

3.4.1.2. Atributos basados en dominios

- **Número de TLD distintos consultados (P9):** es eficaz para detectar *bots* basados en DGA que no solo generan dominios aleatorios con diferente SLD, sino también diferentes TLD.
- **Número de SLD distintos consultados (P10):** es un indicador de la presencia de *bots* basados en DGA en la red.
- **Relación de unicidad (P11):** es la relación entre el número de solicitudes enviadas y el número de solicitudes distintas enviadas, bajo la suposición de que el anfitrión ha enviado al menos 1000 solicitudes por hora.

3.4.1.3. Atributos basados en respuesta

- **Número de consultas fallidas/Dominio No Existente (NXDOMAIN) (P12):** es un indicador de infección del anfitrión en la red. En dominios generados por DGA, de varias consultas realizadas solo una o ninguna es resuelta. De esta manera, se genera gran cantidad de consultas fallidas que genera un código de respuesta igual a NXDOMAIN.

3.4.1.4. Atributos basados en IP.

- **Número de ciudades distintas de direcciones IP resueltas (P13):** es un indicador de anomalías, cuando las direcciones IP se distribuyen por ciudades.
- **Número de países de direcciones IP resueltas (P14):** es un indicador de anomalía, cuando las direcciones IP se distribuyen entre países.

3.4.1.5. Atributos basados en Mapeo (FQDN - IP)

- **Relación de flujo (P15):** es la proporción de las distintas solicitudes enviadas a las distintas direcciones IP resueltas, con la condición de que el *host* haya enviado al menos 100 consultas y haya recibido al menos 100 respuestas.

3.4.2. Detección de valores atípicos con Python

Con un *dataset* de huellas digitales DNS de los *hosts*, se da paso a la búsqueda de un motor de detección de anomalías que clasifique las huellas. La clasificación será en dos categorías: *bot* o limpio, en función de la presencia o ausencia de una anomalía respectivamente.

Para la detección de anomalías en la red, se parte de la premisa de que la cantidad de tráfico benigno es mayor a la cantidad de tráfico maligno. La premisa no se cumple para el conjunto de datos y escenario planteado en [6]. El autor de dicho trabajo establece los umbrales entre benigno y maligno en función del conocimiento que dispone de su red. El resultado es un *dataset* des-balanceado donde la mayoría de huellas fueron catalogados como *bot*. Los umbrales de clasificación variarán dependiendo la naturaleza de la red sobre la que se implemente el sistema, y la premisa conceptual de una anomalías debe ser tomada en cuenta. Por lo que para este proyecto el motor de detección de anomalías se lo realizará con aprendizaje automático no supervisado y un análisis de los resultados de etiquetado.

Por facilidad se explora la existencia de librerías en Python para aprendizaje automático. Scikit-learn[4] es un módulo de Python que incluye librerías para aprendizaje automático supervisado y no supervisado como: clasificación, regresión, *clustering*, reducción de dimensionalidad entre otros. Scikit-learn interopera con las bibliotecas numéricas, científicas y gráficas propias de Python; NumPy[5], SciPy[6] y Matplotlib[7] respectivamente, convirtiéndola así en un potente herramienta. Entre las categorías de aprendizaje no supervisado, scikit-learn presenta un apartado dedicado a la detección de valores atípicos y novedades. La función de detección de valores atípicos es lo requerido para esta fase del proyecto, dada la naturaleza de los datos y el resultado esperado.

La detección de valores atípicos también se conoce como detección de anomalías no supervisada y la detección de novedades como detección de anomalías semisupervisada. En el contexto de la

[4] https://github.com/scikit-learn/scikit-learn
[5] https://numpy.org/
[6] https://www.scipy.org/
[7] https://matplotlib.org/

detección de valores atípicos, los valores atípicos/anomalías no pueden formar un grupo denso. Los estimadores suponen que los valores atípicos/anomalías se encuentran en regiones de baja densidad. Para la detección de novedades, las novedades/anomalías pueden formar un grupo denso siempre que se encuentren en una región de baja densidad de los datos de entrenamiento, considerada normal en este contexto [24].

Para la detección de valores atípicos scikit-learn posee los siguientes algoritmos: *bust covariance*, *One-Class SVM*, *Isolation Forest* y *Local Outlier Factor*. De los cuales, *Isolation Forest* y *Local Outlier Factor* son los algoritmos recomendados como método eficaz para la detección de valores atípicos en conjuntos de datos de dimensiones moderadamente altas. En la Figura 3.2 se presenta una comparación de los algoritmos de detección de anomalías de scikit-learn para un conjuntos de datos. Para este proyecto será *Isolation Forest* el algoritmo escogido para la detección de los valores anormales.

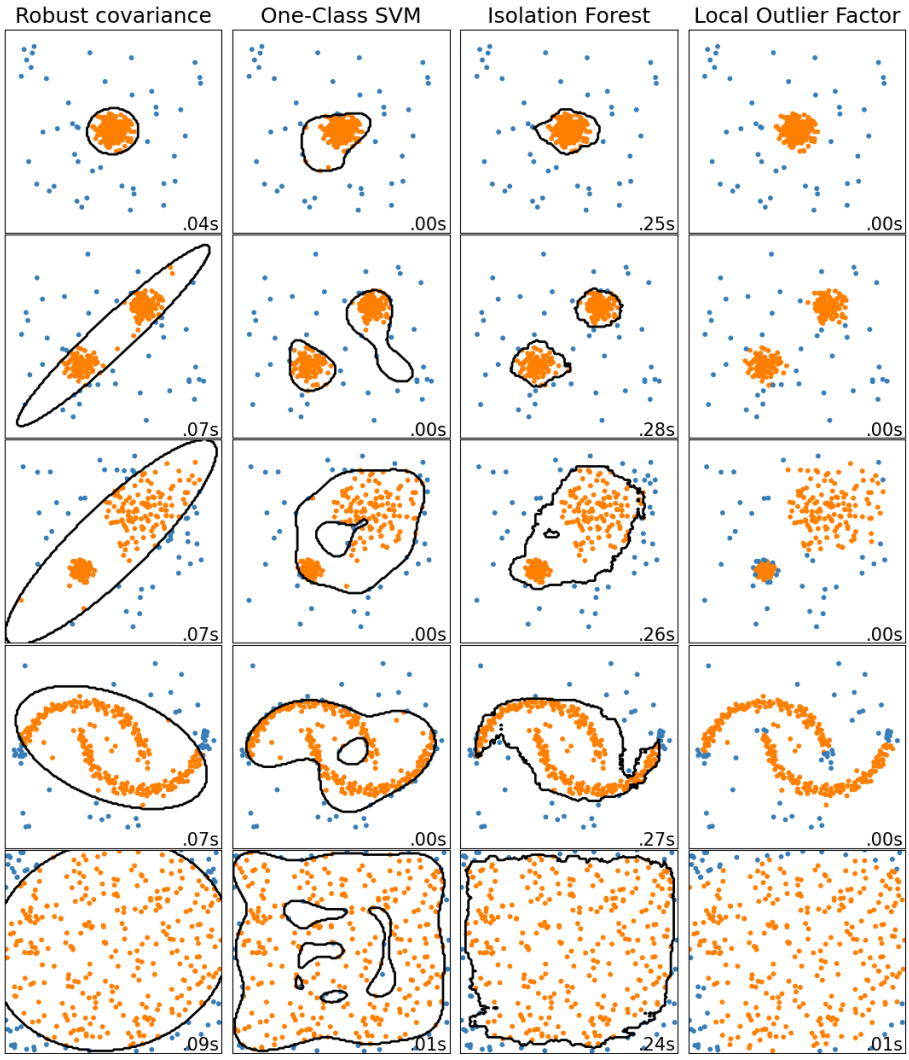


Figura 3.2: Comparación de los algoritmos de detección de valores atípicos en scikit-learn.

3.4.2.1. Bosques de aislamientos

Una forma eficaz de realizar la detección de valores atípicos en conjuntos de datos de alta dimensión, es utilizar bosques de aislamiento. El algoritmo identifica explícitamente las anomalías en lugar de los puntos de datos normales, teniendo como base el árbol de decisiones. Los datos se 'aíslan'seleccionando aleatoriamente una característica y luego seleccionando aleatoriamente un valor dividido entre los valores máximos y mínimos de la característica seleccionada.

Considerando que los valores atípicos son menos frecuentes que las observaciones regulares, se puede decir que en el espacio dimensional de las características, estos están más alejados de una zona densa de muestras. Desde el punto de vista del árbol que forma el modelo, para los valores aislados las rutas son notablemente más cortas desde la raíz. En la Figura 3.3 se presenta un esquema del proceso de aislamiento. Así, la longitud del camino entre los árboles será la medida que define la decisión entre normalidad y valor atípico.

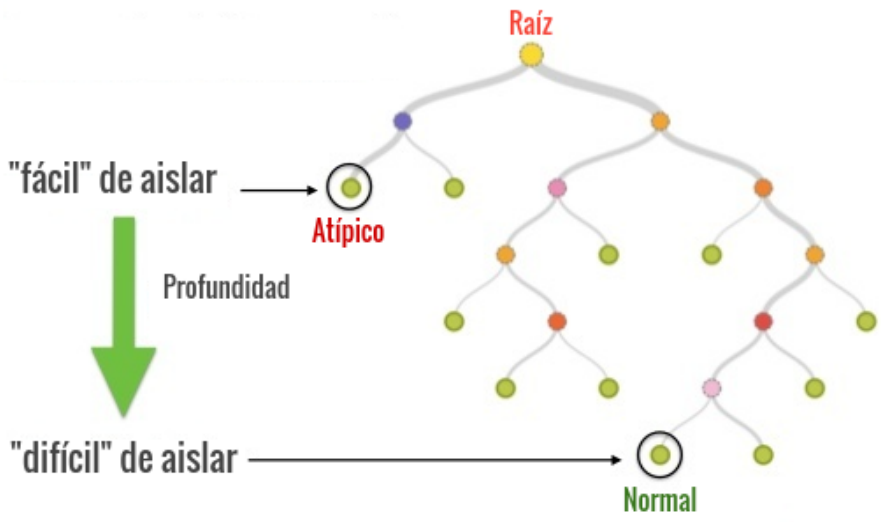


Figura 3.3: Esquema de funcionamiento de *Isolation Forest*

3.4.3. Análisis de nombres de dominio

Es notable que la mayoría de los nombres de dominio basados en DGA contienen cadenas sin sentido, donde es difícil pronunciarlas o leerlas. Esto se debe a la naturaleza algorítmica de su generación, así como la evasión de que coincidan con nombres de dominio existentes. En los trabajos realizados en [18] y [20] se presenta como características de aleatoriedad en un dominio, la existencia de consonantes o vocales secuenciales (considerando a la letra "y" como vocal) y una elevada entropía.

La entropía de Shanon mide la incertidumbre de una fuente de información, y está definida por la Ecuación (3.1).

$$H = -\sum_{i=1}^{n} p_i \log p_i \tag{3.1}$$

Donde p_i es la distribución de probabilidad $P_n = (p_1, ..., p_n)$ con $p_i \geq 0$ para $i = 1, ..., n$ y $\sum_{i=1}^{n} 1$.

Un ejemplo del análisis de dominios DGA se presenta en la Tabla 2.1. No se considera las direcciones TLD de los dominios.

Tabla 3.1: Análisis de algunos dominios DGA.

Dominio.	Max. consonantes secuenciales.	Max. vocales secuenciales.	Entropía.
dkoaeyauxjqdht.work	6	6	3.52
rsxgrtkswwal.su	10	1	3.08
jchhbyjryuiiuy.com	5	6	3.39
mwzcpkmwid.org	8	1	2.92
dnxmllaabettingk.com	6	2	3.5

3.4.3.1. Algoritmo de medición de aleatoriedad

En el mismo contexto de aleatoriedad de dominios, en [18] se presenta un algoritmo de medición de aleatoriedad de los nombres de dominio. El trabajo establece umbrales para los valores de entropía, número de vocales secuenciales y número de consonantes secuenciales. Se logran una gran precisión de clasificación de los nombres de dominios como normales o basados en DGA. El algoritmo se presenta en la Figura 3.4. En este trabajo, el algoritmo será utilizado como una herramienta de apoyo aplicada sobre las huellas detectadas como anómalas. Se busca determinar la existencia de consulta hacia dominios que sigan patrones de aleatoriedad, propios de dominios generados por DGA.

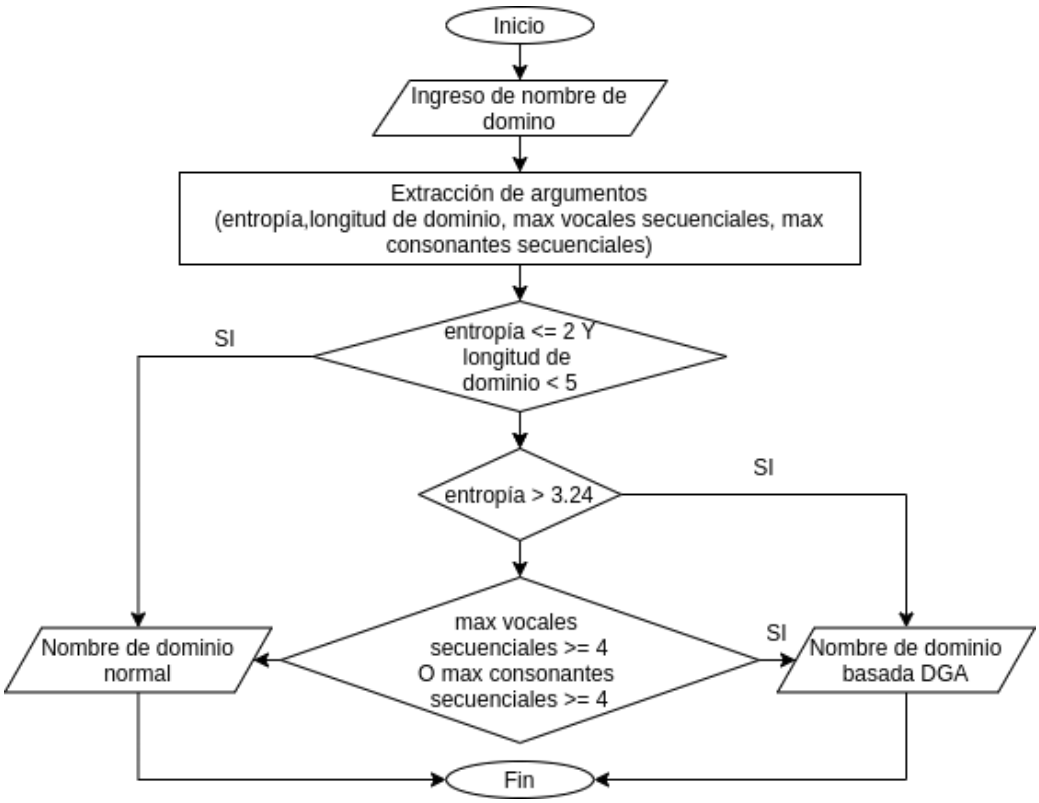


Figura 3.4: Diagrama de flujo del algoritmo de medición de aleatoriedad.

3.5. Aprendizaje automático para detección autónoma

La salida del módulo clasificador de anomalías, presenta un *dataset* con datos etiquetados como *bot* o limpio (-1 y 1 respectivamente). Este conjunto de datos permite la posibilidad de entrenar un algoritmo de aprendizaje automático supervisado. La finalidad es tener un sistema autónomo de detección de futuros *hosts* infectados en la red en un menor tiempo. Continuando con el procedimiento presentado por [6], el algoritmo a entrenar es el algoritmo de bosques aleatorios que presentó el mejor resultado en función de la precisión de aprendizaje.

3.5.1. Clasificación con bosques aleatorios

Los bosques aleatorios o *Random forest* es un algoritmo de clasificación basado en árboles de decisión. Cada árbol de decisión es un meta-estimador individual que se entrena con una muestra ligeramente distinta de los datos. La predicción final de una nueva observación, se obtiene promediando las predicciones de todos los árboles individuales que forman el modelo. Un ejemplo del funcionamiento del bosque aleatorio se presenta en la Figura 3.5. Scikit-learn posee este algoritmo de aprendiza como RandomForestClassifier[8].

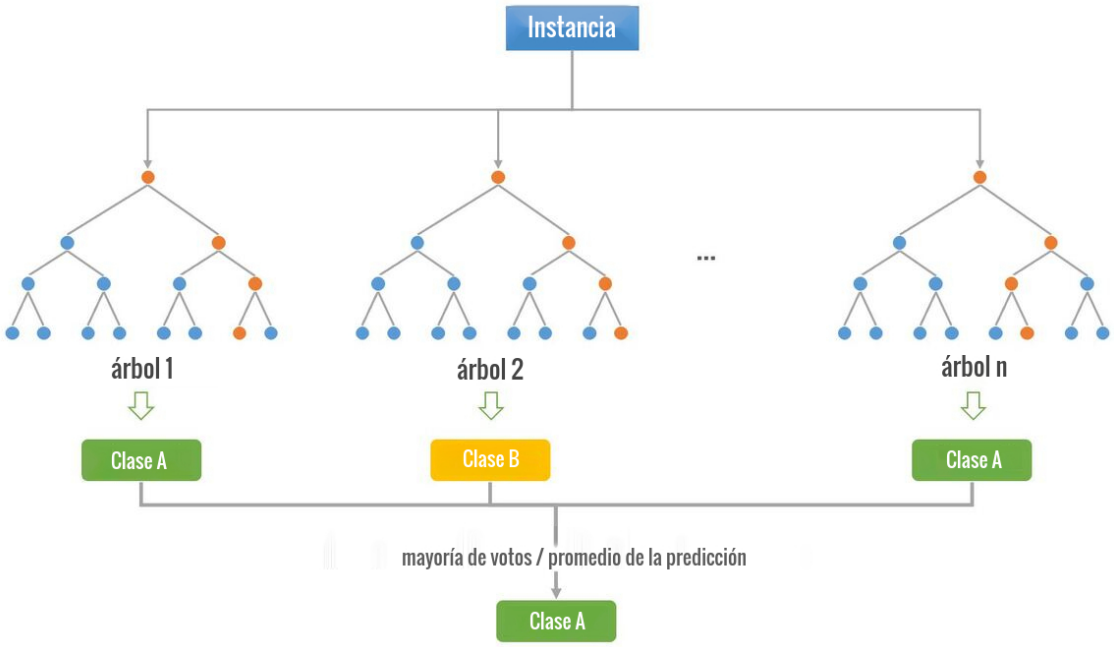


Figura 3.5: Esquema de funcionamiento de *Random Forest*.

3.5.2. Métricas de error

Para conocer la calidad del modelo entrenado se recurre a algunas métricas de error. Los variables que interviene en la obtención de estas métricas son:

- Verdaderos Positivos (*True Positives*, TP)

[8] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier

- Verdaderos Negativos (*True Negatives*, TN)
- Falsos Positivos (*False Positives*, FP)
- Falsos Negativos (*False Negatives*, FN)

3.5.2.1. Matriz de confusión

Una matriz de confusión puede ser considerado como una tabla. Cada columna representa el número de las predicciones de cada una de las clases, y cada fila representa a las instancias en la clase real. Así, con la matriz se puede analizar el desempeño de un modelo de clasificación.

3.5.2.2. Precisión

La precisión es la relación entre las predicciones correctas y el número total de predicciones correctas previstas. Esto mide la precisión del clasificador a la hora de predecir casos positivos.

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

3.5.2.3. Sensibilidad

La sensibilidad también es llamada en inglés *recall*, es la relación entre las predicciones positivas correctas y el número total de predicciones positivas. O simplemente, cuán sensible es el clasificador para detectar instancias positivas.

$$Sensibilidad = \frac{TP}{TP + FN} \tag{3.3}$$

3.5.2.4. Puntaje F1

El puntaje F1 es el promedio ponderado de precisión y sensibilidad. Por lo tanto, esta puntuación tiene en cuenta tanto los falsos positivos como los falsos negativos.

$$F1 = 2 * (\frac{precision * sensibilidad}{precision + sensibilidad}) \tag{3.4}$$

3.5.2.5. Exactitud

La exactitud de la clasificación es la relación entre las predicciones correctas y el número total de predicciones.

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.5}$$

3.6. Arquitectura de la solución

Para la topología de red presentada en la Figura 3.1, la integración de la solución IDS/NAD para *botnets* de forma *offline* se presenta en la Figura 3.6. Como se mencionó con anterioridad, se busca no interferir en el funcionamiento normal de la red. De forma pasiva se extraerá del tráfico de la red los atributos DNS requeridos para el proyecto.

En la Figura 3.7 se presenta la arquitectura en función de las herramientas de software que intervendrán en el proyecto. Se considera las herramientas de software para la captura, procesamiento

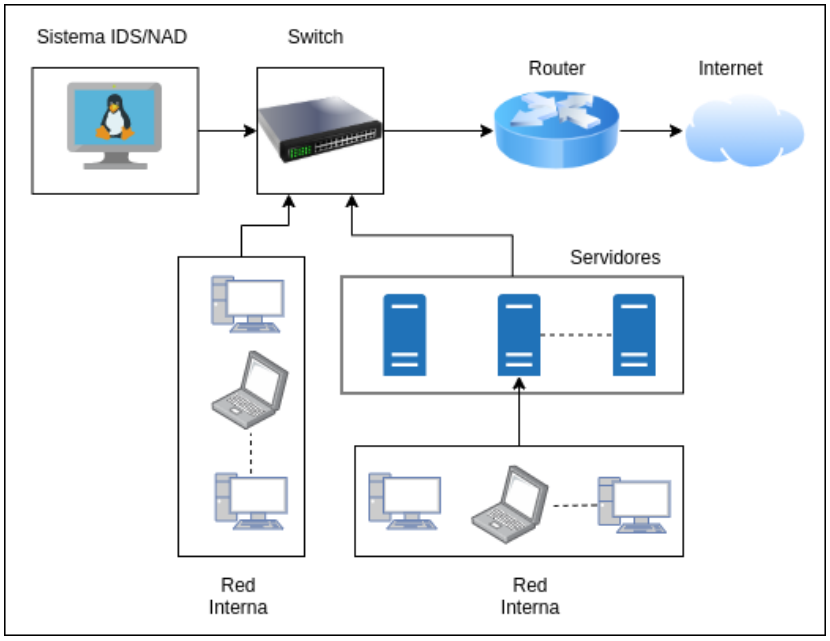


Figura 3.6: Arquitectura de la solución en función de la topología de la red.

y almacenamiento en las diferentes fases del proyecto. Además se incluye las funcionalidades de puerto *mirroring* y promiscuo necesarias para el acceso a los datos.

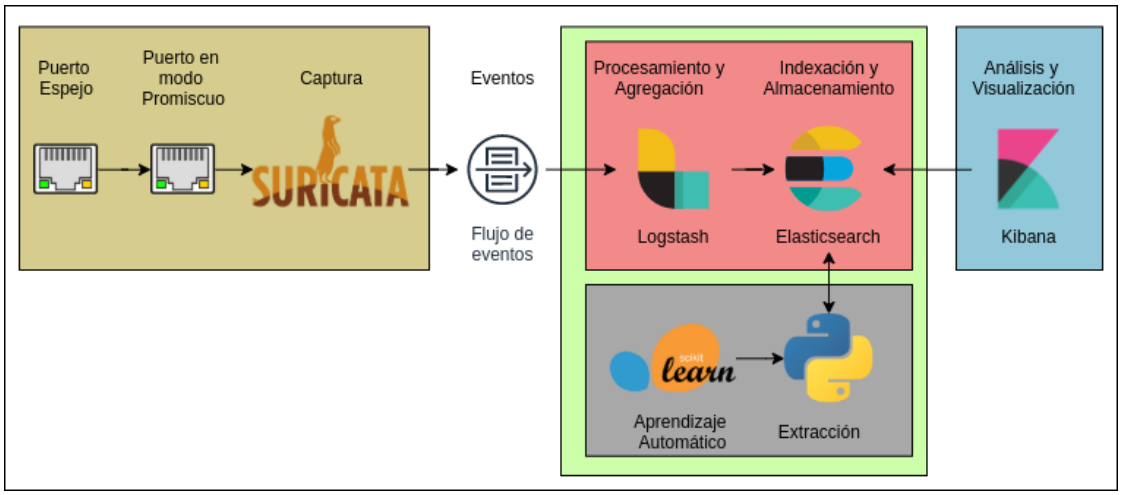


Figura 3.7: Arquitectura de la solución desde el punto de vista del software.

Finalmente, desde el punto de vista de los módulos y procesos de la solución, la Figura 3.8 muestra el tipo de datos de entrada y salida que tendrá el sistema. Los diferentes módulos que intervienen en el proceso de: captura, generación de huellas digitales DNS, detector de anomalías, son presentados con sus procesos respectivos para obtener la salida deseada.

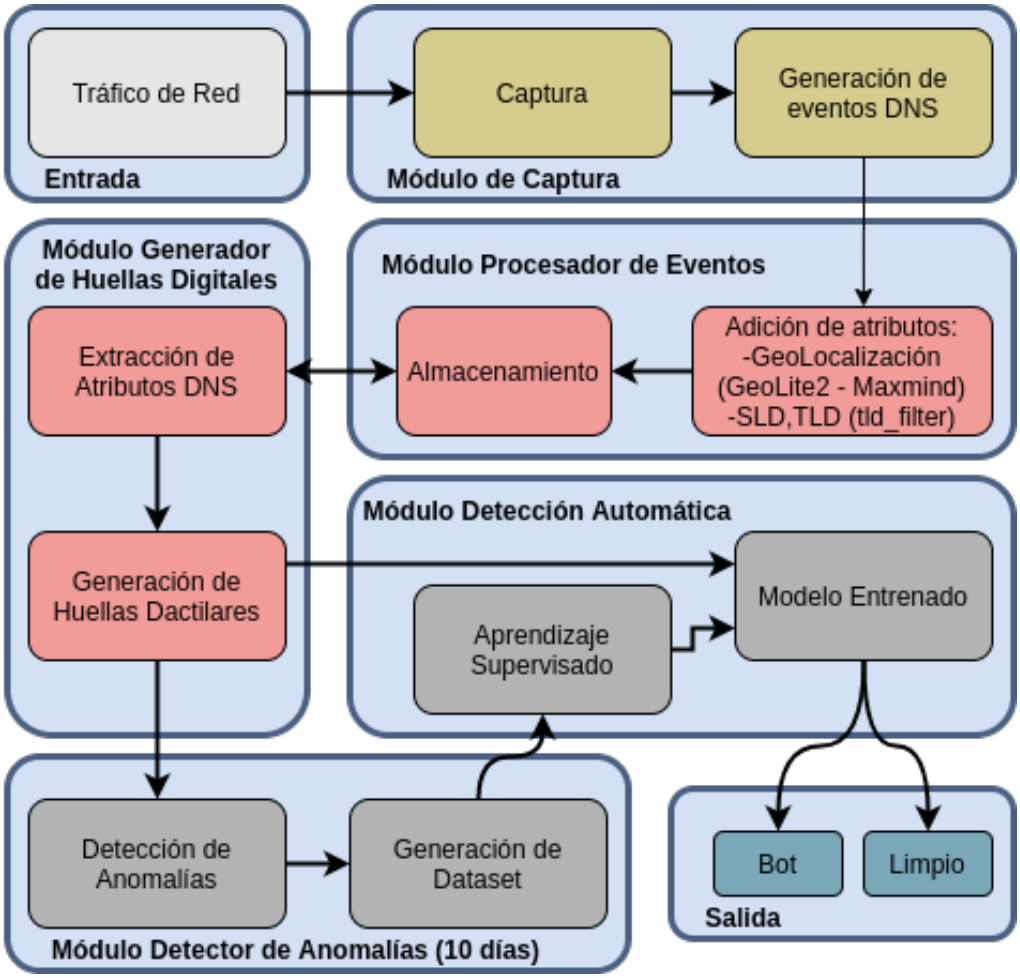


Figura 3.8: Arquitectura de la solución desde la perspectiva de los procesos.

3.7. Conclusiones

Para el correcto funcionamiento del sistema son necesarias funciones de captura de eventos, gestión de los eventos, detección de anomalías y aprendizaje automático.

Para las captura de eventos DNS se plantea el uso de Suricata, dichos eventos serán gestionados por la pila ELK. Logstash incluirá campos de geo-localización de dominios resueltos y la extracción de los dominios de nivel superior y segundo nivel. Elasticsearch a su vez sirve como base de almacenamiento y motor de búsqueda para consultas futuras. Kibana es la interfaz visual para interactuar con los datos y laboratorio de consultas.

La generación de huellas digitales DNS se las realiza mediante consulta externa con Python. Son 15 atributos numéricos generados por el flujo de eventos DNS (información de solicitudes, respuestas, dominios, IP y tipo de mapeo) los que sirven para crear las huellas. Las huellas se generan para cada *host* activo y considera intervalos de tiempo de una hora. A las huellas obtenidas se les aplicará el algoritmo de aprendizaje no supervisado *Isolation Forest* para detectar anomalías, que serán catalogadas como infectadas. Sobre las huellas catalogadas como infectadas, se aplicará un algoritmo de medición de aleatoriedad de dominios para detectar posibles solicitudes DGA.

Con las huellas clasificadas se dará paso al entrenamiento del algoritmo *Random Forest* para aprendizaje automático supervisado. Las métricas de error permitirán ajustar el modelo, con el fin de reducir el tiempo de detección de *bots* infectados en futuras huellas dactilares.

Capítulo 4

Implementación y Evaluación del Sistema

4.1. Implementación del sistema

En este apartado se detalla el proceso de implementación del sistema y la evaluación de los resultados obtenidos del mismo. Las instrucciones de instalación de las herramientas principales y algoritmos de los diferentes procesos, están descritos con mayor detalle en la sección de anexos. Cabe recalcar que los procedimientos presentados se implementaron sobre el Sistema Operativo (OS) Ubuntu 20.04.1 LTS y el lenguaje de programación Python. Los algoritmos completos no se incluyen en este trabajo por la extensión de los mismos y pueden ser encontrados en la plataforma de desarrollo colaborativo GitHub con dirección `https://github.com/ViQuezada/Bootnet_Detection_Modules`.

4.1.1. Captura de eventos DNS

La instalación de Suricata y su configuración para acceso y generación de eventos DNS de la red, se detallan en la sección A.1 del Anexo A. Los eventos DNS generados por Suricata son de dos tipos *query* y *answer* en un formato JSON. Una vista de los eventos generados y sus campos se los presenta en la Figura 4.1.

4.1.2. Gestión de los eventos DNS

El procesamiento de los eventos y su almacenamiento se los gestiona con la pila ELK, la instalación y configuración de la pila se detalla en la sección A.2 del Anexo A. Además, se incluye el procedimiento referente a la geolocalización de dominios y la extracción de niveles TLD y SLD de nombres de dominio.

Los registros obtenidos al procesar con Logstash los eventos generados por Suricata, tienen las características que se pueden observar en la Figura 4.2, referente a un *log* tipo *answer*. Donde varios campos son propios de Elasticsearch para poder indexar y gestionar los eventos almacenados.

Los campos de interés de los eventos almacenados en Elasticsearch se detallan en la Tabla 4.1. Estos campos permitirán generar las huellas digitales. La tabla presenta 3 columnas: Campo, Descripción y Tipos; que definen los campos de Elasticsearch y el tipo de valores que se obtiene al consultarlos.

```
vicente@suricata:~$ sudo tail -n 2 /var/log/suricata/eve.json
{"timestamp":"2021-02-12T10:19:52.919255-0500","flow_id":289689581388793,"
in_iface":"ens192","event_type":"dns","src_ip":"192.188.51.10","src_port":
48903,"dest_ip":"201.159.221.68","dest_port":53,"proto":"UDP","dns":{"vers
ion":2,"type":"answer","id":10665,"flags":"8180","qr":true,"rd":true,"ra":
true,"rrname":"api.zoom.us","rrtype":"AAAA","rcode":"NOERROR","authorities
":[{"rrname":"zoom.us","rrtype":"SOA","ttl":877,"soa":{"mname":"ns-1137.aw
sdns-14.org","rname":"awsdns-hostmaster.amazon.com","serial":1,"refresh":7
200,"retry":900,"expire":1209600,"minimum":86400}}]}}
{"timestamp":"2021-02-12T10:19:52.923180-0500","flow_id":873186658358828,"
in_iface":"ens192","event_type":"dns","src_ip":"201.159.221.68","src_port"
:31257,"dest_ip":"202.134.0.62","dest_port":53,"proto":"UDP","dns":{"type"
:"query","id":56634,"rrname":"103.1.85.36.in-addr.arpa","rrtype":"PTR","tx
_id":0}}
```

Figura 4.1: Atributos de los eventos DNS (*query* y *answer*) capturados por Suricata.

4.1.3. Generación de huellas digitales DNS de los hosts

La generación de huellas digitales se las realiza mediante diferentes consultas hacia Elasticsearch desde Python, conectado como un cliente externo. El procedimiento de la conexión Python-Elasticsearch se describe en el Anexo B. Los índices creados en Elasticsearch (un índice por día de captura) poseen la marca de tiempo @timestamp generado por Suricata y procesado por Logstash. A esto se suma la capacidad de Elasticsearch de procesar este tipo de información, permitiendo así una fácil integración de un análisis en función de fecha y hora.

De esta manera, el pseudocódigo para generar las huellas digitales DNS de los *hosts* se lo presenta en el Listado 4.1. Donde una consulta de característica representa una consulta a Elasticsearch considerando las operaciones y campos presentados en la Tabla B.1 del Anexo B. Y dado que la respuesta a la consulta es retornada en formato diccionario (clave-valor)[1] apuntar a una clave de la respuesta representa el acceso a los valores de la misma, que sirven para generar las huellas digitales.

```
Algoritmo GeneradorHuellasDactilaresDNS
Entrada: Indices DNS en Elasticsearch
Salida: dataset de Huellas Dactilares DNS

Set P1 a P15 = []
Set hosts = []
indices = consulta de índices DNS existentes en Elasticsearch
For each indice in indices
    horas=[0 a 23]
    For each hora in horas
        datos = consulta de P1
        hosts = datos['ips']
        P1 = datos['número de solicitudes']
        For each host in hosts
            datos = consulta de P2
            Add datos['número solicitudes distintas'] to P2
            datos = consulta de P3
            Add datos['número máximo solicitudes'] to P3
            datos = consulta de P4
            Add avg(datos['número solicitudes por minuto']) to P4
            Add max(datos['número solicitudes por minuto']) to P5
```

[1] https://docs.Python.org/3/tutorial/datastructures.html

🗓	@timestamp	Jan 16, 2021 @ 14:40:19.008
t	@version	1
t	_id	Bse2DHcB6S2ZqzkndOcH
t	_index	logstash-dns-2021.01.16
#	_score	-
t	_type	_doc
t	dest_ip	185.89.219.11
#	dest_port	53
ⓘ	dn.sld	⚠ fbcdn
ⓘ	dn.tld	⚠ net
t	dns.grouped.A	143.255.249.32
t	dns.rcode	NOERROR
t	dns.rrname	Z-P3-sCOnTent.FUIo13-1.fna.FbcDN.NEt
t	dns.rrtype	A
t	dns.type	answer
t	event_type	dns
t	geoip.city_name	Hacienda Ibarra
t	geoip.country_name	Ecuador
🌐	geoip.ip	143.255.249.32
t	src_ip	201.159.221.68
#	src_port	59,723

Figura 4.2: Atributos en formato tabla de una respuesta DNS.

```
            datos = consulta de P6
            Add datos['número respuestas MX'] to P6
            datos = consulta de P7
            Add datos['número respuestas PTR'] to P7
            datos = consulta de P8
            Add datos['número servidores distintos'] to P8
            datos = consulta de P9
            Add datos['número TLD distintos consultados'] to P9
            datos = consulta de P10
            Add datos['número SLD distintos consultados'] to P10
            Add P1/P2 to P11
            datos = consulta de P12
            Add datos['número respuestas NXDOMAIN'] to P12
            datos = consulta de P13
            Add datos['número ciudades distintas'] to P13
            datos = consulta de P14
            Add datos['número países distintos'] to P14
            datos = consulta de P15
            Add P2/datos['número respuestas NOERROR'] to P15
        End for
    End For
End For
```

Tabla 4.1: Campos relevantes de los eventos DNS almacenados en Elasticsearch.

Campo	Descripción	Tipos
dest_ip	dirección IP de destino	varios
dn.sld	dominio de segundo nivel	varios
dn.tld	dominio de nivel superior	varios
dns.grouped.A	grupo de direcciones IPv4 devueltas	varias
dns.grouped.AAAA	grupo de direcciones IPv6 devueltas	varias
dns.rcode	código de retorno	NOERROR NXDOMAIN
dns.rrname	nombre de dominio consultado	varios
dns.rrtype	tipo de registro de recurso solicitado	Dirección IPv4 (A) Dirección IPv6 (AAAA) PTR MX
dns.type	tipo de evento	*query* *answer*
geoip.city_name	ciudad de la IP resuelta	varios
geoip.country_name	país de la IP resuelta	varios
geoip.ip	IP del grupo resulto, escogida para obtener su localización	varios
src_ip	dirección IP de origen	varios

```
Save (P1,...,P15) to HuellasDactilares.csv
```

Listado 4.1: Pseudocódigo para generar huellas dactilares DNS

En la Figura 4.3 se presenta una vista del resultado del módulo de generación de huellas digitales DNS. Cada línea representa a una huella dactilar de un *host*. Para una mejor visualización y análisis, a más de los atributos P1 a P15 se añade la fecha y la IP del *host* al que corresponde cada huella dactilar. Así, cada línea es una huella única de un *host* para una fecha y hora determinada.

4.1.4. Detección de anomalías en las huellas digitales

La detección de valores anómalos de las huellas digitales se las realiza con *Insolation Forest*. Como método de visualización de resultados se realiza una reducción a 3 dimensiones con PCA. El uso de estos algoritmos se detallan en el Anexo C. El resultado de este módulo genera una nuevo *dataset* cuyas huellas están catalogadas como limpias o infectas (1 y -1 respectivamente). Una vista del resultado de catalogar las huellas se presenta en la Figura 4.4.

4.1.4.1. Detección de aleatoriedad de nombres de dominio

Las huellas que han sido catalogadas como anómalas pertenecen a un *host* en una hora determinada. Con la información del *host* y la hora en que se generó la anomalía, se procede a buscar si los dominios SLD consultados han sido generados aleatoriamente. Para reducir el número de dominios verificados se realiza un filtro de aquellos dominios que no tuvieron respuesta (NXDOMAIN). La hipótesis para esta reducción, es que un *bot* genera una ráfaga de consultas con dominios DGA hasta que uno de estos es resuelto. Una vez que se han obtenido los dominios se procede a aplicar el algoritmo presentado en la Figura 3.4 para clasificar y analizar resultados. El pseudocódigo se lo presenta en el Listado 4.2 donde el algoritmo de medición de aleatoriedad y clasificación es presentada como la función RMA().

```
index,@timestamp,ip,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14,P15
1,2020-11-16T16:00:00Z,-
201.159.221.68,117413,93730,440,14676.625,19717,404,11984,8153,418,7706,1.2527,5317,673,100,1.4093
2,2020-11-16T16:00:00Z,2800:0068:0000:bebe:
0000:0000:0000:0004,22786,14701,49,2848.25,4027,0,0,3435,486,8442,1.55,0,0,0,0.0
3,2020-11-16T16:00:00Z,200.0.29.68,9653,6464,25,1206.625,1693,1,2027,1,100,895,1.4933,815,1,1,0.9527
4,2020-11-16T16:00:00Z,190.15.134.92,5707,1288,270,713.375,994,0,12,1,32,358,4.4309,643,1,1,0.2561
5,2020-11-16T16:00:00Z,192.188.59.2,4400,1473,622,550.0,775,0,100,1,38,392,2.9871,109,0,1,0.7932
6,2020-11-16T16:00:00Z,192.207.244.250,2938,1859,49,367.25,655,43,2,1,56,545,1.5804,372,1,1,0.7854
7,2020-11-16T16:00:00Z,192.207.244.254,2581,1759,51,322.625,716,35,0,1,65,592,1.4673,460,1,1,0.9157
8,2020-11-16T16:00:00Z,45.182.117.5,2578,103,2323,368.2857,734,0,20,1,12,54,25.0291,2465,0,1,0.9537
9,2020-11-16T16:00:00Z,192.188.49.209,2509,1255,42,313.625,869,0,16,1,22,312,1.9992,341,1,1,0.6104
10,2020-11-16T16:00:00Z,190.15.137.4,2247,1246,14,280.875,459,0,5,1,27,285,1.8034,220,0,1,0.6227
11,2020-11-16T16:00:00Z,192.188.48.160,1824,1113,13,228.0,444,0,262,1,44,247,1.6388,313,1,1,0.7546
12,2020-11-16T16:00:00Z,45.184.102.4,1614,847,41,201.75,295,0,425,1,26,256,1.9055,110,0,1,0.7991
13,2020-11-16T16:00:00Z,181.198.63.92,1359,363,249,169.875,247,0,29,1,16,135,3.7438,377,1,1,0.3712
14,2020-11-16T16:00:00Z,143.255.250.36,1355,889,11,169.375,304,0,9,1,24,246,1.5242,48,1,1,0.6918
15,2020-11-16T16:00:00Z,143.255.248.132,1346,785,30,192.2857,348,0,51,1,24,240,1.7146,88,1,1,0.645
16,2020-11-16T16:00:00Z,190.96.107.75,1329,869,17,166.125,353,0,73,1,20,268,1.5293,138,1,1,0.7466
17,2020-11-16T16:00:00Z,186.3.44.231,1274,554,20,159.25,304,4,9,1,18,163,2.2996,161,0,1,0.5055
18,2020-11-16T16:00:00Z,200.1.112.222,1257,830,8,157.125,236,0,43,1,26,256,1.5145,103,0,1,0.7339
19,2020-11-16T16:00:00Z,200.93.235.74,964,602,9,120.5,200,0,4,1,20,180,1.6013,57,1,1,0.6704
20,2020-11-16T16:00:00Z,190.15.134.3,957,279,91,119.625,234,0,0,1,13,86,3.4301,65,1,1,0.3282
21,2020-11-16T16:00:00Z,192.188.55.101,939,814,5,187.8,492,0,1,1,26,268,1.1536,136,1,1,1.053
22,2020-11-16T16:00:00Z,201.159.221.78,878,39,28,109.75,145,0,882,1,3,3,22.5128,0,0,1,19.5
23,2020-11-16T16:00:00Z,190.15.134.211,845,301,40,105.625,169,0,0,1,9,95,2.8073,23,1,1,0.3666
24,2020-11-16T16:00:00Z,190.15.128.220,758,80,404,94.75,444,0,0,1,4,33,9.475,17,1,1,0.108
25,2020-11-16T16:00:00Z,181.198.57.142,698,486,61,116.3333,181,0,20,1,26,169,1.4362,158,1,1,0.9759
26,2020-11-16T16:00:00Z,192.188.46.5,660,480,6,82.5,137,0,23,1,24,162,1.375,38,0,1,0.7869
27,2020-11-16T16:00:00Z,190.15.137.222,652,344,37,81.5,150,0,13,1,17,130,1.8953,203,0,1,0.8056
28,2020-11-16T16:00:00Z,192.188.52.5,594,460,18,84.8571,400,3,34,1,20,170,1.2913,61,1,1,0.9684
29,2020-11-16T16:00:00Z,45.188.219.73,593,429,13,84.7143,198,0,18,1,14,136,1.3823,34,0,1,0.8049
30,2020-11-16T16:00:00Z,190.96.107.76,590,375,12,73.75,193,0,2,1,18,154,1.5733,28,1,1,0.6818
31,2020-11-16T16:00:00Z,192.188.49.210,572,400,37,81.7143,199,0,7,1,15,132,1.43,20,1,1,0.7984
32,2020-11-16T16:00:00Z,201.159.223.202,560,254,17,80.0,172,27,1,1,8,37,2.2047,399,0,1,1.7279
33,2020-11-16T16:00:00Z,45.238.219.106,555,151,73,69.375,156,0,0,1,8,62,3.6755,20,1,1,0.2822
```

Figura 4.3: *Dataset* de huellas digitales DNS.

```
Algoritmo ComprobacionDominiosDGA
Entrada: Huellas Dactilares DNS Etiquetadas
Salida: Host con posibles consultas a dominios DGA

Dataset = cargar Dataset de Huellas Dactilares DNS Etiquetadas
HuellasAnomalias = seleccionar huellas catalogadas como anómalas

For each huella in HuellasAnomalias
    ip = obtener ip de huella
    hora = obtener hora de huella
    dominios_sld = consulta de dominios sld con ip, hora, NXDOMAIN
    For each dominio in dominios_sld
        resp = RMA(dominio)
        If resp ==''dominio DGA''
            Mostrar (ip,dominio,resp)
        End If
    End For
End For
```

Listado 4.2: Pseudocódigo de comprobación de dominios DGA.

4.1.5. Detección automática con Random Forest

La implementación en Python de *Random Forest* sobre el *dataset* obtenido del módulo clasificador de huellas se presenta en Anexo D. El resultado de interés son las métricas de error del algoritmo que

```
index,@timestamp,ip,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14,P15,anomaly
1,2021-01-16T19:00:00Z,2800:0068:0000:bebe:
0000:0000:0000:0004,56597,37474,47,2176.8077,2701,129,10189,5079,419,2652,1.5103,0,0,0,0.0,-1
2,2021-01-16T19:00:00Z,200.0.29.68,27624,18603,116,1062.4615,1298,1,7760,1,256,2503,1.4849,1281,1162,98,0.8511,-1
3,2021-01-16T19:00:00Z,192.188.46.5,4781,1200,125,183.8846,245,1,2804,1,54,498,3.9842,243,64,31,0.6483,-1
4,2021-01-16T19:00:00Z,45.182.117.5,4223,147,3578,162.4231,896,0,102,1,29,95,28.7279,4008,42,23,0.7462,-1
5,2021-01-16T19:00:00Z,201.159.222.135,3938,4,1484,151.4615,160,0,0,1,2,2,984.5,3938,0,0,0.0,-1
6,2021-01-16T19:00:00Z,192.188.48.160,3365,2099,46,129.4231,301,0,636,1,54,387,1.6031,727,51,24,0.8399,-1
7,2021-01-16T19:00:00Z,192.188.55.101,3094,1592,34,128.9167,426,1,68,1,61,514,1.9435,301,78,29,0.5969,-1
8,2021-01-16T19:00:00Z,186.3.44.231,2959,297,1474,113.8077,163,1,3,1,30,124,9.963,1999,45,24,0.3136,-1
9,2021-01-16T19:00:00Z,190.96.107.75,2590,1111,59,99.6154,267,30,123,1,64,450,2.3312,152,45,29,0.4871,-1
10,2021-01-16T19:00:00Z,190.15.137.4,2390,650,44,113.8095,158,1,12,1,25,152,3.6769,90,37,16,0.2836,1
11,2021-01-16T19:00:00Z,192.188.59.2,2202,1328,198,84.6923,129,16,102,1,67,492,1.6581,95,63,28,0.6863,-1
12,2021-01-16T19:00:00Z,181.198.63.86,2162,214,681,83.1538,133,0,582,1,9,85,10.1028,1271,24,14,0.244,1
13,2021-01-16T19:00:00Z,45.184.102.4,1962,666,54,75.4615,113,0,917,1,29,245,2.9459,72,32,18,0.693,1
14,2021-01-16T19:00:00Z,201.159.221.78,1788,30,96,68.7692,132,12,1772,1,3,5,59.6,3,0,0,0.0,1
15,2021-01-16T19:00:00Z,45.188.219.73,1659,363,783,66.36,140,0,68,1,21,142,4.5702,1074,36,19,0.6425,1
16,2021-01-16T19:00:00Z,190.96.107.76,1546,483,57,59.4615,111,0,7,1,45,243,3.2008,66,28,18,0.3366,1
17,2021-01-16T19:00:00Z,192.188.52.5,1485,795,58,67.5,216,9,4,1,36,330,1.8679,117,58,20,0.6442,1
18,2021-01-16T19:00:00Z,200.12.169.123,1372,33,1104,52.7692,77,0,1304,1,6,20,41.5758,11,0,1,11.0,-1
19,2021-01-16T19:00:00Z,45.238.218.162,1345,149,294,51.7308,85,0,0,1,9,61,9.0268,30,19,7,0.114,1
20,2021-01-16T19:00:00Z,190.15.134.92,1301,228,204,50.0385,239,0,7,1,16,87,5.7061,169,23,13,0.2034,1
21,2021-01-16T19:00:00Z,200.1.112.222,1275,546,28,49.0385,87,0,43,1,36,238,2.3352,171,36,18,0.507,1
22,2021-01-16T19:00:00Z,200.9.96.203,1229,225,24,47.2692,60,0,0,1,198,12,5.4622,0,5,3,0.2127,1
23,2021-01-16T19:00:00Z,143.255.250.36,1218,524,26,46.8462,120,0,95,1,31,230,2.3244,123,46,22,0.4953,1
24,2021-01-16T19:00:00Z,186.101.125.195,1197,312,47,46.0385,119,0,106,1,10,71,3.8365,114,19,10,0.2905,1
25,2021-01-16T19:00:00Z,192.188.49.209,1165,216,463,97.0833,210,0,15,1,15,106,5.3935,853,13,8,0.7224,1
26,2021-01-16T19:00:00Z,201.218.8.2,1105,185,55,44.2,124,0,501,1,18,78,5.973,533,27,13,0.3309,1
27,2021-01-16T19:00:00Z,190.15.137.222,1092,503,56,42.0,146,0,10,1,22,160,2.171,117,34,13,0.5576,1
28,2021-01-16T19:00:00Z,201.218.59.210,874,128,421,33.6154,63,0,214,1,3,31,6.8281,788,6,5,1.6203,1
29,2021-01-16T19:00:00Z,190.15.134.3,830,182,47,31.9231,217,0,11,1,14,61,4.5604,57,24,13,0.237,1
30,2021-01-16T19:00:00Z,143.255.248.132,817,401,21,31.4231,53,0,65,1,29,168,2.0374,59,41,20,0.5688,1
31,2021-01-16T19:00:00Z,186.101.125.196,767,188,49,29.5,54,0,1,1,13,69,4.0798,24,22,11,0.2547,1
32,2021-01-16T19:00:00Z,201.159.223.140,746,6,372,28.6923,33,0,372,1,2,5,124.3333,4,1,1,0.0081,1
33,2021-01-16T19:00:00Z,143.255.250.33,713,8,117,37.5263,84,0,0,1,3,3,89.125,10,6,4,0.0114,1
```

Figura 4.4: *Dataset* de huellas digitales DNS catalogadas.

permiten conocer el nivel de aprendizaje del mismo. Un ejemplo de dichas métricas se presenta en la Figura 4.5, en donde se observa la matriz de confusión, la precisión, sensibilidad y puntaje F1.

```
[[ 10   3]
 [  0 192]]
              precision    recall  f1-score   support

          -1       1.00      0.77      0.87        13
           1       0.98      1.00      0.99       192

    accuracy                           0.99       205
   macro avg       0.99      0.88      0.93       205
weighted avg       0.99      0.99      0.98       205
```

Figura 4.5: Métricas de error.

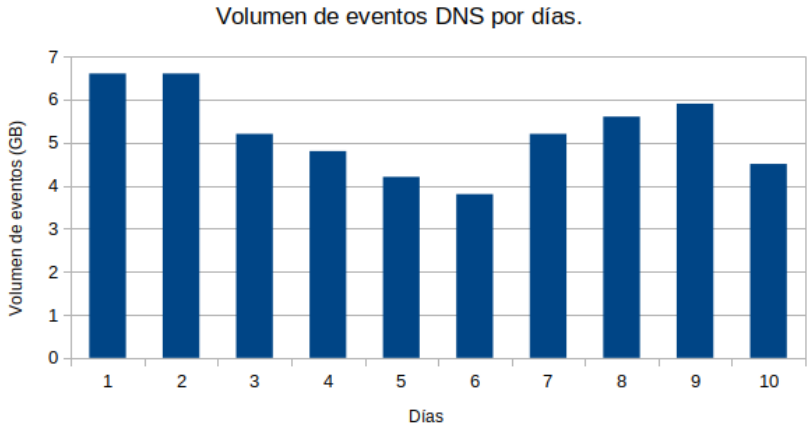
4.2. Evaluación de resultados

En esta sección se analiza los resultados de la aplicación de los procedimientos realizados en la Sección 4.1.

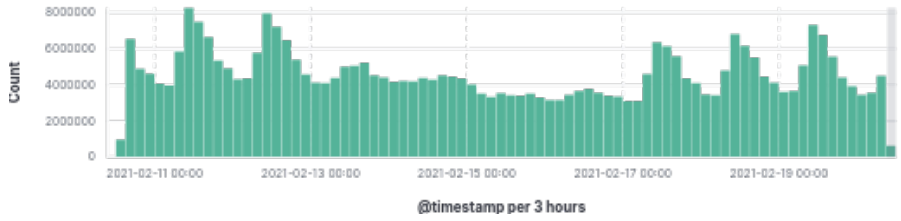
4.2.1. Datos capturados

La captura de los datos se la realiza siguiendo el procedimiento presentado en [6] para un lapso de 10 días. De manera experimental sobre el archivo /var/log/suricata/eve.json, que almacena los eventos generados por Suricata, se observó un incremento del archivo en una razón de aproximadamente

1GigaByte (GB) por hora. Los períodos de captura de 24 horas por 10 días resultarían en la necesidad de un almacenamiento mínimo de 240 GB. Realizando la extracción solo de los atributos necesarios para el proyecto por parte de Logstash permitió reducir la cantidad de almacenamiento requerido. La ventana de Stack Management/Index Management de Kibana permite tener la información del volumen de los diferentes índices almacenados en Elasticsearch. En la Figura 4.6a se muestra el volumen (obtenido de Kibana) de los índices DNS generados por días, y que suman un total de 52,4 GB. El espacio de almacenamiento necesario resultó ser menor a la cuarta parte de lo esperado, permitiendo ahorrar recursos de almacenamiento. Desde la perspectiva de la cantidad, la Figura 4.6b se presenta el número de eventos DNS generados. Se tiene picos de alrededor 8 millones de eventos en lapsos de 3 horas, generando un total de 687′526,874 eventos durante los 10 días de captura. Serán estos eventos los que permitan generar las huellas digitales DNS de los *hosts* por horas.



(a) Volumen de eventos generados.



(b) Cantidad de eventos DNS generados.

Figura 4.6: Eventos DNS capturados.

Un análisis del índice referente al día número 2, permite tener una visión general de la topología de la red. Así, en la Figura 4.7 se muestran los resultados referentes a *hosts* activos en la red y sus nombres de dominios asociados (en el caso de tenerlo). Como se había mencionado, la red consta de servidores propios de universidades, así como tráfico interno propio. De las IP resueltas las pertenecientes a las *cachés* pasarán a ser parte de una lista blanca para excluirlas de los siguientes procesos.

4.2.2. Generación de las huellas digitales DNS

La aplicación del módulo generador de huellas digitales DNS permite obtener valores de interés, como el número de *hosts* activos en la red; su comportamiento por horas se lo muestra en la Figura

```
* Numero de host activos = 278
* Numero de host con nombre de dominio =  27
* Numero de host sin nombre de dominio =  251

 Algunos de los nombres de los host resueltos
IP              Dominio
--------------  ----------------------------------------
201.159.221.68  dnscache1.cedia.org.ec
192.188.48.160  redco.ucuenca.edu.ec
190.15.137.4    pucei.edu.ec
192.188.59.2    goliat.espol.edu.ec
192.188.52.5    saran.ucsg.edu.ec
181.39.20.62    host-181-39-20-62.telconet.net
179.49.2.130    corp-179-49-2-130.uio.puntonet.ec
200.10.148.5    cleopatra.espol.edu.ec
192.188.58.163  ns1.espe.edu.ec
201.159.221.78  mail02.cedia.org.ec
201.159.222.80  campusvirtual.uazuay.edu.ec
190.63.135.170  customer-190-63-135-170.claro.com.ec
190.15.141.40   csirtmail.cedia.org.ec
201.159.223.202 mail.manabi.gob.ec
179.49.39.137   corp-179-49-39-137.uio.puntonet.ec
190.152.178.112 112.178.152.190.static.anycast.cnt-grms.ec
179.49.44.211   ftth-179-49-44-211.cue.celerity.ec
181.39.97.235   host-181-39-97-235.telconet.net
190.152.131.96  96.131.152.190.static.anycast.cnt-grms.ec
201.159.221.67  mirror.cedia.org.ec
143.255.248.144 perfsonaruio.cedia.org.ec
179.49.44.241   ftth-179-49-44-241.cue.celerity.ec
181.112.218.146 146.218.112.181.static.anycast.cnt-grms.ec
200.93.200.108  intranet1.inocar.mil.ec
192.207.244.240 host-192-207-244-240.uleam.edu.ec
181.198.229.247 host-181-198-229-247.netlife.ec
```

Figura 4.7: Hosts activos en la red con nombre de dominio asociado.

4.8. Los *hosts* que cumplan con la condición de no ser *caché* y tener al menos 100 consultas por hora generarán el *dataset* de huellas dactilares.

En la Tabla 4.2 se presenta los datos de máximo, mínimo y promedio de los *hosts* existentes en la red. De estos *host* se obtienen las huellas digitales DNS.

Tabla 4.2: Valores máximo, mínimo y promedio de los *hosts* activos en la red.

	Mínimo	**Máximo**	**Promedio**
Hosts activos por hora	273	346	298.96
Suma de hosts por día	6664	7105	6873.50

Las huellas digitales obtenidas son agregadas a un nuevo índice en Elasticsearch. Kibana con su herramienta de ML (pestaña Machine Learning/Data Visualizer) permite realizar un análisis estadístico de los valores de cada uno de los atributos. La herramienta contabiliza un total de 32182 huellas generadas para el periodo de 10 días de captura de tráfico. El tráfico fue generado por 241 *hosts* distintos que interactuaron en la red y que cumplen con las condiciones del proyecto. En la Figura 4.9 se presenta un ejemplo del formato de salida del módulo de ML de Kibana para los atributos P1 y P10. Mientras que en la Tabla 4.3 se presenta para todos los atributos los valores mínimos, promedio y máximos obtenidos utilizando la misma herramienta.

4.2.3. Detección de anomalías

De los parámetros del algoritmo de *Isolation Forest*, es de interés definir el número de árboles que se ajuste de mejor manera a los datos. La relación entre el aumento de árboles y la complejidad computacional es directamente proporcional. En la Figura 4.10 se muestra la cantidad de datos catalogados como anomalías en función del número de árboles para dos ventanas, de 50 a 1000 y de 50

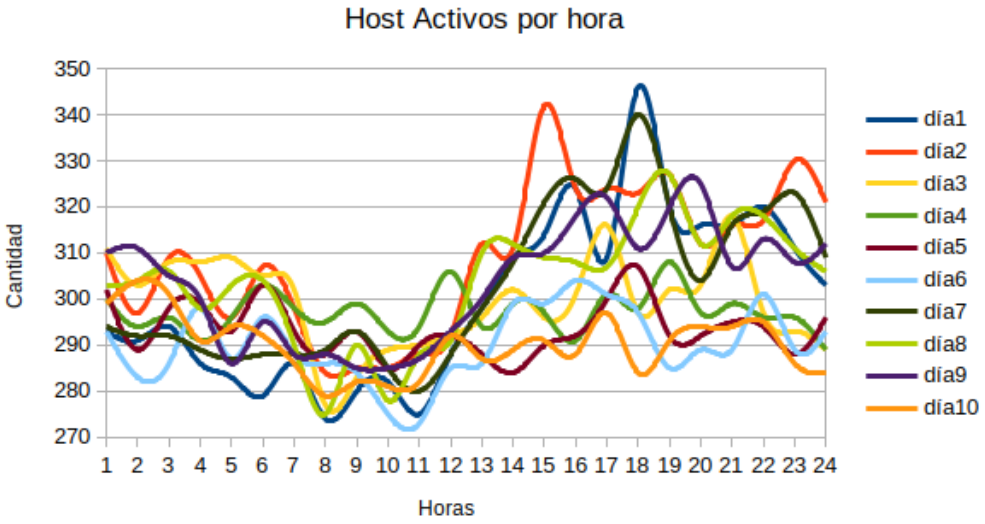


Figura 4.8: Número de *hosts* activos en la red por horas.

a 400. Se observa una tendencia de dificultad de aislamiento de valores anómalos según se aumente el número de árboles al modelo. La cantidad de anomalías encontradas varía entre 2000 y 2300. Para este experimento, se toma el número de árboles del modelo como 110 al ser un pico de cantidad de anomalías con un valor de 2287. Desde la ciberseguridad es mejor tener algunas detecciones con falsos positivos que pasar por alto posibles intrusiones. Otros parámetros del algoritmo como es la contaminación se la deja en automático para que el algoritmo trace su umbral entre lo normal y lo anormal.

Para una aproximación visual de los resultado de clasificación, como fue presentado en la Figura 3.2, es necesario hacer una reducción de los componentes de la huella. Con PCA se realiza la reducción de 15 componentes a 3. En la Figura 4.11a se presenta la vista completa de como se ubicaron las huellas en el nuevo espacio 3 dimensiones, donde los puntos rojos representan a las anomalías. En las Figuras 4.11b 4.11c y 4.11d; se realiza un acercamiento al espacio de mayor concentración de muestra desde varios ángulos, con el fin de obtener una mejor perspectiva de los resultados. Se observa como los puntos catalogados como anomalías están apartados del espacio de mayor concentración de las muestras. Este tipo de gráficas son de gran valor en el aprendizaje no supervisado. Al no existir etiquetas se puede analizar de forma visual la clasificación realizada por el algoritmo.

También se puede hacer un análisis visual de las huellas catalogadas como *bot* y limpio en función de sus atributos. En la Figura 4.12 se muestra en escala real cada atributo de las huellas (desde 4.12a hasta 4.12ñ) y su clasificación. Las zonas de mayor densidad están en valores numéricos bajos, por lo que sobresalen los datos catalogados como anómalos. En la Figura 4.13 se presenta la clasificación de las huellas en función de los atributos (desde 4.13a hasta 4.13ñ) considerando una escala logarítmica. Esta escala permite observar como la mayor densidad de puntos son huellas limpias y en base a esta zona densa el algoritmo traza umbrales. Los valores que excedan los umbrales serán catalogados como *bot*. Un análisis de los valores numéricos de los atributos de las huellas, fue realizada con anterioridad en la Tabla 4.3. Las gráficas que resaltan son la Figura 4.13f y la Figura 4.13h, correspondientes al número de consultas a registros MX y el número de servidores DNS distintos consultados. Para el caso de P6 su valor promedio es 0 por lo que resalta a la vista solo los valores anómalos en color rojo.

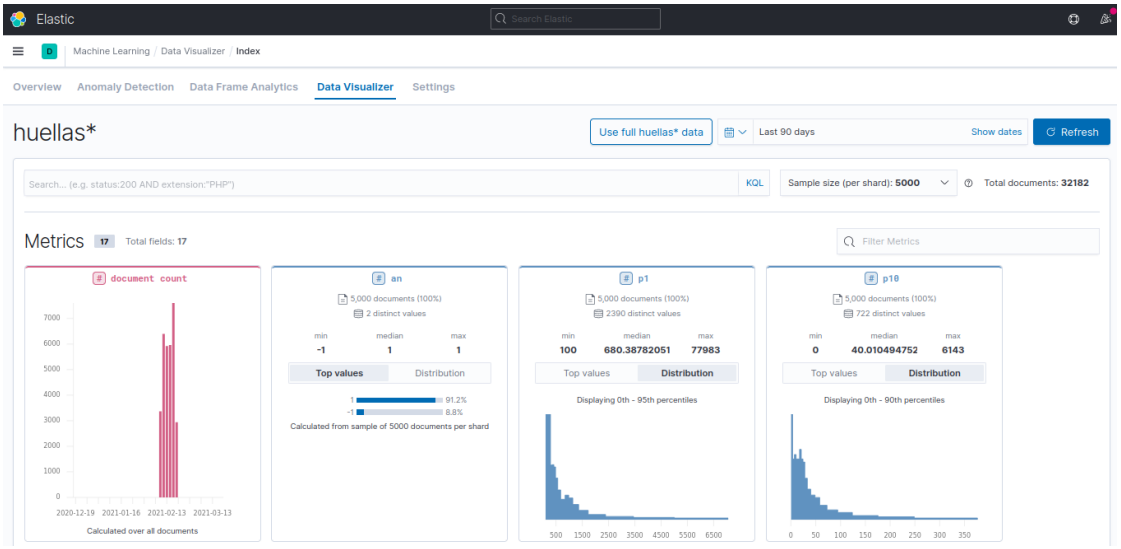
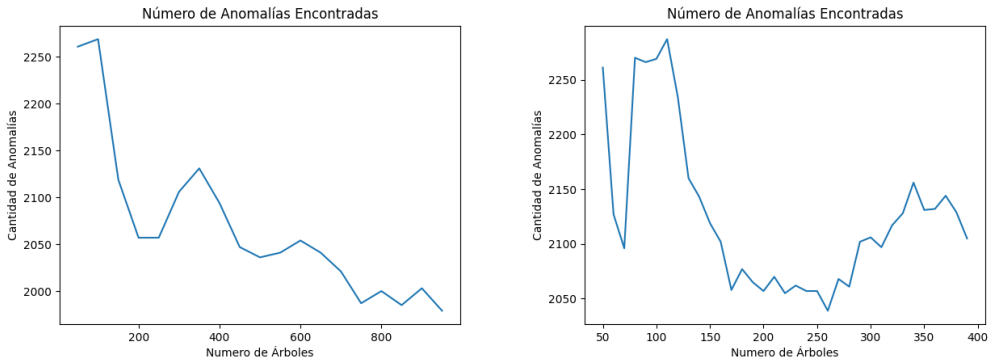


Figura 4.9: Valores estadísticos para los atributos generados por la pestaña ML de Kibana.



(a) Rango de 50 a 1000 árboles.

(b) Rango de 50 a 400 árboles.

Figura 4.10: Número de muestras catalogadas como anomalías en función del número de árboles.

Para el atributo P8 su valor para todas las huellas es 1, es decir, todos los *hosts* están realizando sus consultas al servidor DNS local. Por esta razón el valor único se presenta como línea.

Otros valores de interés son los porcentajes de clasificación en función de las huellas totales. En la Tabla 4.4 se presenta los valores para: el número de huellas catalogadas como limpia y *bot*, el porcentaje con respecto al total de huellas y el número de *hosts* que estarían dentro de esta clasificación.

Se considera que una huella representa el comportamiento de un *host* en una hora específica. En este contexto, un *host* que tiene muchas huellas catalogadas como anómalas, denota que su comportamiento general a lo largo del tiempo solo difiere de la mayoría de los *hosts* de la red. Así, su comportamiento siempre será anormal. En la Tabla 4.5 se presenta algunos de los *hosts* con más huellas catalogadas como anómalas y su porcentaje en comparación al total de huellas generadas. Se observa como es el caso de los *hosts* 200.0.29.68 y 45.182.117.5 cuyo porcentaje de huellas catalogadas como anómalas es del 100 %. Como se observó en la Figura 4.7 algunos de las IPs son servidores, es decir, tras estas direcciones existen redes internas, lo que explicaría su cantidad discordante de solicitudes en comparación a una IP asociada a un *host*. De esta manera, para generar el *dataset* que entrenará al algoritmo de aprendizaje

Tabla 4.3: Valores mínimo, promedio y máximo de los atributos DNS de las huellas digitales.

Atrib.	Característica	Mínimo	Promedio	Máximo
P1	Número de solicitudes de DNS por hora	100	586.43	77983
P2	Número de solicitudes de DNS distintas por hora	1	78.09	44923
P3	La mayor cantidad de solicitudes para un solo dominio por hora	1	88.98	22461
P4	Número medio de solicitudes por minuto	1.66	11.89	1278.40
P5	La mayor cantidad de solicitudes por minuto	3	39	15951
P6	Número de consultas de registros MX por hora	0	0	3032
P7	Número de consultas de registros PTR por hora	0	18.05	22680
P8	Número de servidores DNS distintos consultados por hora	1	1	1
P9	Número de dominios de TLD distintos consultados por hora	0	8	386
P10	Número de dominios SLD distintos consultados por hora	0	37.90	6310
P11	Relación de unicidad por hora	1	5.31	2650
P12	Número de consultas fallidas/NXDOMAIN por hora	0	91.77	23518
P13	Número de ciudades distintas de direcciones IP resueltas	0	10	2390
P14	Número de países distintos de direcciones IP resueltas	0	6	119
P15	Relación de flujo por hora	0	0.31	59

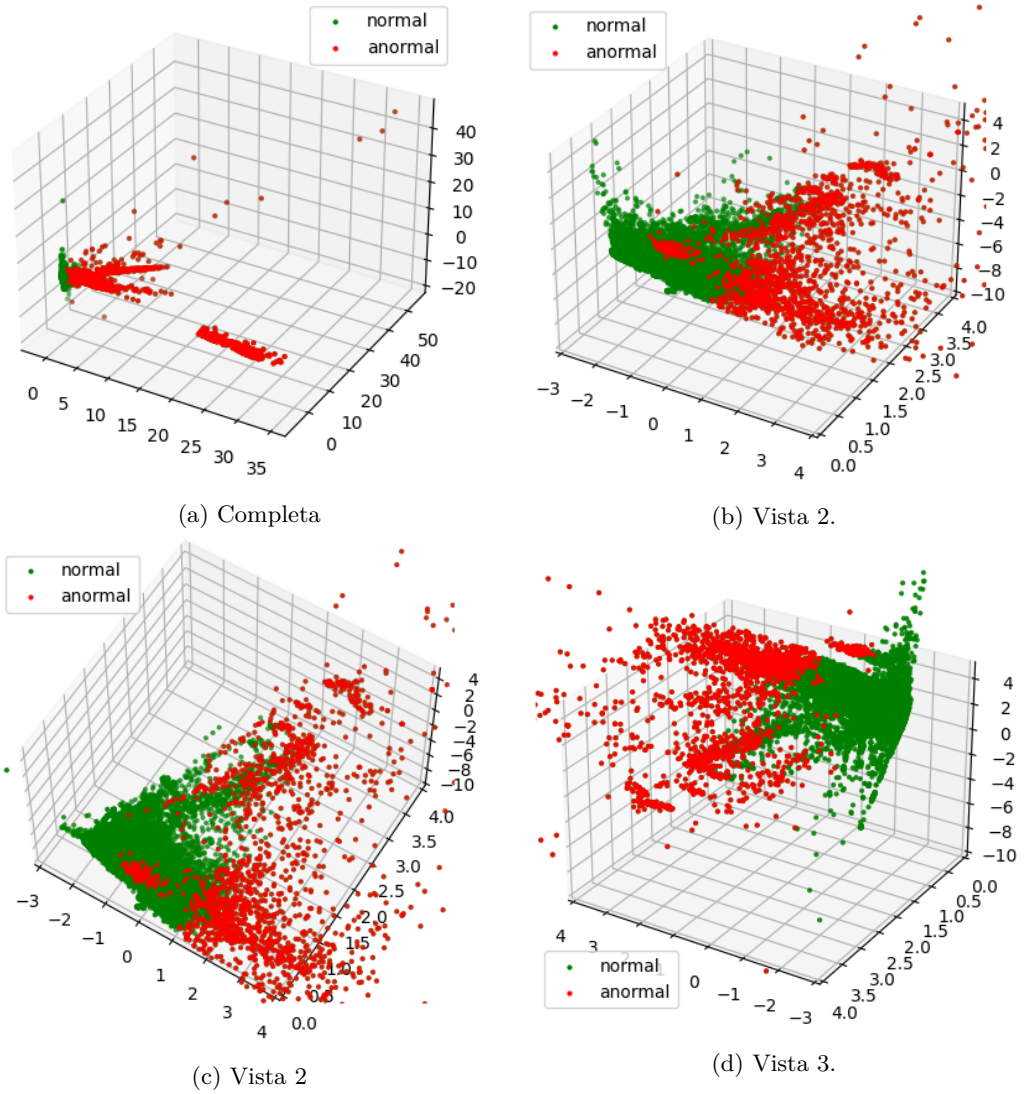
Tabla 4.4: Número de huellas, porcentaje sobre el número total de huellas y número de *hosts* en cada clasificación.

Tipo	# Huellas	Porcentaje	# Host
Limpio	29895	92.9 %	239
Bot	2287	7.1 %	58

supervisado, se excluirá las huellas cuyos *hosts* tengan un porcentaje de huellas anómalas mayor al 50 %. Se considera la premisa de que en detección de anomalías los valores atípicos deben ser menor a los valores normales.

Una característica típica de los *bots* es su comunicación DNS mediante ráfagas, lo que genera una anomalía en la cantidad de solicitudes en el tiempo. Es decir, en intervalos de tiempo cortos se realizan varias consultas con dos objetivos: el primero son la resolución de dominios DGA y el segundo es el envío de información hacia el *botmaster*.

Desde el punto de vista general del tráfico neto, puede no ser visible una anomalía como se observa en las Figuras 4.14a y 4.14b. Sin embargo, el método de detección de *hosts* infectados permite encontrar anomalías en el mismo conjunto de datos analizándolos individualmente. En la Figura 4.14c y 4.14d se muestra el tráfico DNS de un *host* catalogado como limpio que denota secuencialidad en sus consultas. En la Figura 4.14e y 4.14f se presenta el tráfico DNS de un *host* catalogado como *bot*, su patrón de consultas denota algunos picos de comunicación en ráfagas. Los casos A y B presentados en la Figura 4.14 presentan una comparación de los tráficos: neto, de un *host* limpio y de los *hosts* 190.15.134.3 y 190.15.128.111 catalogados como *bot*, para las horas 17/02/2021:08:00:00 y 11/02/2021:16:00:00 respectivamente.



(a) Completa

(b) Vista 2.

(c) Vista 2

(d) Vista 3.

Figura 4.11: Varias vistas de la reducción realizada por PCA a 3 dimensiones de las huellas clasificadas.

4.2.4. Verificación por consultas a dominios DGA

Para las huellas catalogadas como anómalas, se busca entre sus consultas aquellas fallidas cuyos SLD tengan características de ser generadas aleatoriamente. En la Figura 4.15 se presenta el resultado de dominios sospechosos para algunas direcciones IP según el algoritmo de aleatoriedad planteado.

En [18] evaluaron un medidor de aleatoriedad con un *dataset* de dominios de familias DGA y con los 200 mil primeros dominios del ranking de consultas de Alexa. Sin embargo, como se puede observar en la Figura 4.15, en una red interna pueden existir nombres de dominios propios que el algoritmo ha clasificado como sospechoso y a primera vista parecen ser legítimos. El factor humano es necesario en este punto, para analizar la existencia de dominios que tengan un carácter de aleatoriedad. Este sería un nuevo filtro que apoye la clasificación de un *host* como infectado. Así, en la Tabla 4.6 se presenta la extracción de los SLD para algunas huellas que fueron catalogadas como *bot* y presentan más de 10 dominios que no fueron resueltos y tienen un carácter de aleatoriedad. Este comportamiento es

Tabla 4.5: Algunos *hosts* y su porcentaje de huellas catalogadas como *bot*.

IP	# Huellas *bot*	# Huellas Totales	Porcentaje
200.0.29.68	249	249	100.00 %
45.182.117.5	249	249	100.00 %
186.3.44.231	197	249	79.12 %
201.159.222.92	168	249	67.47 %
181.198.63.86	142	249	57.03 %
201.159.221.78	109	249	43.78 %
190.15.136.140	104	249	41.77 %
192.188.55.101	99	239	41.42 %
143.255.248.132	95	249	38.15 %
192.188.59.2	84	249	33.73 %
192.188.48.160	78	239	32.64 %
45.188.219.73	68	249	27.31 %
201.159.223.19	64	249	25.70 %
190.96.107.75	61	249	24.50 %
200.1.112.222	50	249	20.08 %
190.15.134.92	48	249	19.28 %
181.198.57.142	43	249	17.27 %
45.184.102.4	42	249	16.87 %
143.255.250.36	39	249	15.66 %
192.188.52.5	38	239	15.90 %
186.3.44.232	34	249	13.65 %
190.15.137.4	33	247	13.36 %
192.188.51.10	32	249	12.85 %

repetitivo en el tiempo para las huellas asociadas a los *hosts* de la tabla, apareciendo cada vez nuevos grupos SLDs aleatorios.

4.2.5. Aprendizaje automático

El número de árboles en el algoritmo de *Isolation Forest* determina la precisión de aprendizaje del algoritmo. En la Figura 4.16 se presenta las gráficas que comparan al número de árboles con dos métricas: la precisión de clasificación y el tiempo de procesamiento necesario para clasificar 9282 huellas que representan el 30 % del *dataset*. En la Figura 4.16a se presenta la precisión en función del número de árboles. Se observa como el aumento de árboles en el modelo mejora la precisión del algoritmo de clasificación. En la Figura 4.16b se presenta el tiempo de procesamiento del algoritmo en función del número de árboles. Se observa una relación directa entre el número de árboles del modelo y el tiempo de entrenamiento. Esta gráfica considera utilizar la menor cantidad de árboles desde el punto de vista de dificultad computacional.

Considerando los valores de precisión y tiempo computacional en función del número de árboles, se establece como el número de árboles igual a 25 para este experimento. Con esta cantidad de árboles se consiguió un modelo con una precisión de 0,995. En la Tabla 4.7 se detalla la matriz de confusión obtenida. La tasa de falsos positivos es baja considerando la gran cantidad de huellas clasificadas correctamente como limpias. La tasa de falsos negativos es más alta en comparación con la cantidad de huellas *bot*. Un *host* activo en la red genera una gran cantidad de huellas. Por lo que, de existir un infección por *bot* y con la tasa de precisión del algoritmo, la probabilidad que por lo menos una

Tabla 4.6: Nombres de dominios SLD sospechosos para algunas IPs.

IP	SLD
192.188.48.160	mhdorbkqessgyl, rvrorhfvlbplnqx, rhzrtbov, qvuqtsxx, dqgilijrzny, wkjophhmkpphauf, utuotstsmhwope, bfwbbkzviwycf, ufbgzaez, cedqelbqpcbjq, osefdrlrll, yenkvxlksfdx, fukxezpytfpxit, aybsgpbmjx, lyeulfimbvvib, tsqrpdqpohttgn, podxsxawhtmw, grruajobyednnoz, sajvmhwehbecjul, qoiqlyrtpkb, rhvzvwahhcqheev, zughrjsq, rsnybpcd, hbsfdav, ouxtdlnanz, dhdvpwmgfb, posupetgwidi, ftcjtpcikdzdszg, fdtexwqdc, jpqgqqi, icmefbggg, cdvwbgwyrk, ptbpxsbtjjbsbl, dpptohfyyorzdy, yiwyxmyddfzo, dnophozwmpt, vqbnzwzjwy, vhnqaectewmvkr, jnxwgiljhogl, rwjgdwhcwbydhu, avfkwpauwnyqof, omluqmpyoaupj, bfngtszu, poloniumsurfer, vabclwejwylwu, vwsfosg, wldijlfc, shlqzyujf, fwsmvlkbx, twewcpfpalfvdor, fdyhdojepwijkje
192.188.59.2	pzenzwpg, rkpcqumup, eaqqjicyytkmp, lwtzatuxyggfhd, kvyojowbzfhhcqs, qxptfvbdvuwfm, xcvukpgzgw, wavcinuhtzapkmg, qiyqnwptysynr, obwyoomdpihuqi, xzekogrcpai, lbnfggbnykbf, ohgngxvjdaopvyu, gogjdqyaerzfv, ngzppmqnbdtk, mxezmkgjkqhb, tvgkjyvxp, vjjjkspbxnpkxj, ikgdnbmxreajmr, oxbjhzmptxy, bccgfnotw, mwxuswzaqbfko, kbfkhbbi, lhztqrxusokowi, tdkemxrsbernchi, xyzwrcjnmxnig, qidjmqmgzoavh, fnmoswtzytcqw, jmfxzwerepz, jwhvwpge, pxkamstzvkypat, ocbstnzmsxwznik, xejramygpqhvixx, qgjmkqpdpj, tuqdocsyzwc, oxtjnxp, uvddeiyqssw, jzlqazqrib, knzsroakfck, ulhcedxlk, hvdwkqoky, cdyfqjnzt, hdezubkcaot, iiharkwbvwxgjj
192.188.55.101	cgablcehfnrdpzi, epmsqkjbkgwnv, w42f4ctqv4, chxvlothvb, othvbyopsakbg, uyrrfqqbodu, lnyrczvmoyzro, crwdcntrl, xnycmjmulrsopfmm3x, brtqvyhbpskluz, zqnvkbwhhfwpf, dbankcdn
200.7.82.251	tdfilrvxsqwzli, zdwlgulhfflczk, uyqeztwyfiyusx, wkpzxttbaw, rmrjalyqztfbn, ywarpjtnr, ztsvvxmuohahskz, uyteuylflbiey, wlhhspuscfrg, loswletnvlmv, nmethhafuqgl, qdzpkxrdnkn, tynswlnduizdjw, geqjcjjvx, ujjqlcvehuj
190.15.137.4	zhuvcruhoolkegc, elkbphh, nkgfcdvzyuq, ttvnw, uihcazcnvwey, hdltormauqgmff, ryrunxfjfah, zthhymxm, wmvfuemoczf, gnebxwcinkcma, szawvtxekdovej, xubytfhk, hzujwglxdbt

huella (asociada a un *host-bot*) sea catalogada correctamente es alta. Con este análisis se considera el rendimiento de este módulo como satisfactorio. El trabajo realizado en [6] sobre el que se ha basado este proyecto consiguió una precisión del 0,997 para su modelo. El número de huellas utilizadas para entrenar el algoritmo del trabajo antes mencionado fueron 252742 generadas por un promedio de 2500 *host* activos por hora. La cantidad y variedad de los datos del proyecto guía es muy superior a la manejada en este trabajo, sin embargo la metodología seguida logra conseguir un valor similar de precisión. Además, se debe considerar que los resultados obtenidos pueden variar dependiendo de la naturaleza de la red sobre la que se trabaje. Un ejemplo es el caso del atributo P8 asociado al número de servidores DNS distintos consultados. Mientras que para este proyecto el valor único del atributo P8 fue 1, en [6] se obtienen valores que varían entre 1 y 40. Así, lo que resultó ser un atributo con importancia 0 para las huellas de una red, en otra red es un atributo de mayor importancia con un valor de 0,23.

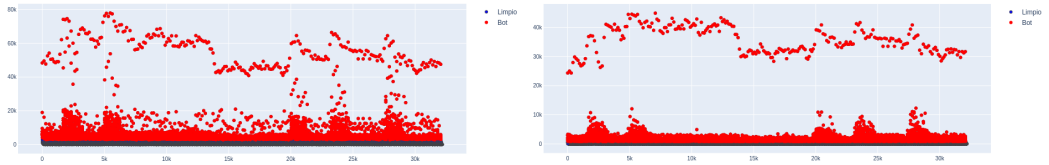
Con los valores de la matriz de confusión se obtienen pesos estadísticos que ayudan a entender la calidad del algoritmo. En la Figura 4.17 se presenta un reporte de los valores de precisión, sensibilidad,

Tabla 4.7: Matriz de confusión para el modelo de *Random Forest* con 25 árboles.

(Real)/(Predecido)	***Bot***	**Limpio**
Bot	367	33
Limpio	15	8867

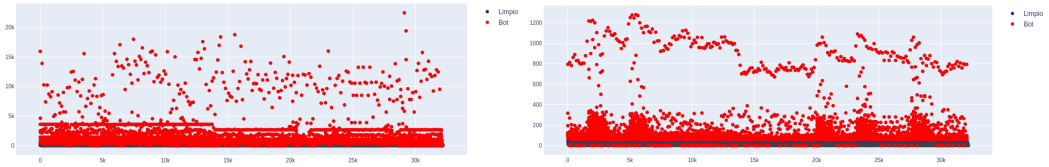
puntuación F1. Dada la gran cantidad de huellas limpias con el que fue entrenado (en comparación con las huellas *bot*), el algoritmo muestra mejor rendimiento para huellas cuyo comportamiento está dentro de lo normal.

Scikit Learn permite desglosar el porcentaje de importancia que cada característica tiene para el para el aprendizaje modelo desarrollado. En la Figuras 4.18 y 4.19 se presentan los valores de porcentaje de importancia de las características para el algoritmo. Se observa que la característica o parámetro más importantes para clasificar una huella como *bot* o limpio es la cantidad de dominios SLD distintos que un *host* consulta por horas. Seguido, se tiene parámetros que relacionan la cantidad de solicitudes realizadas por un *host* como: número de solicitudes distintas por hora, total de solicitudes por hora, promedio de solicitudes por minuto. Las características referentes a las ubicaciones geográficas de los dominios resueltos también tienen relevante importancia para el modelo. Con un porcentaje de importancia similar se encuentran: la mayor cantidad de solicitudes para un solo dominio, cantidad de consultas PTR por hora, mayor cantidad de solicitudes por minuto y el número de consultas fallidas. Como atributo irrelevante se tiene al número de servidores distintos consultados, cuyo valor fue la unidad para todas las huellas. Para esta arquitectura ningún *host* intentó comunicarse a otro servidor DNS que no sea el local.



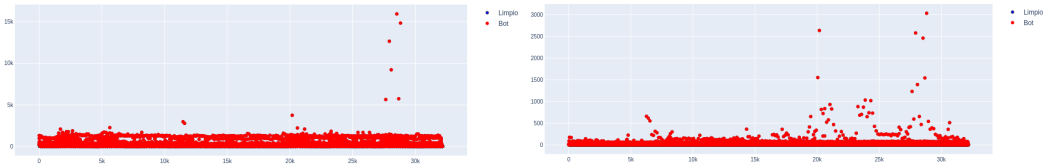
(a) P1

(b) P2



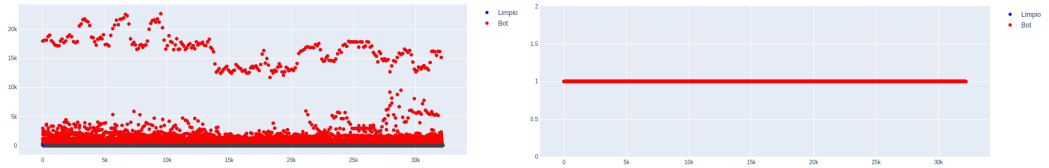
(c) P3

(d) P4



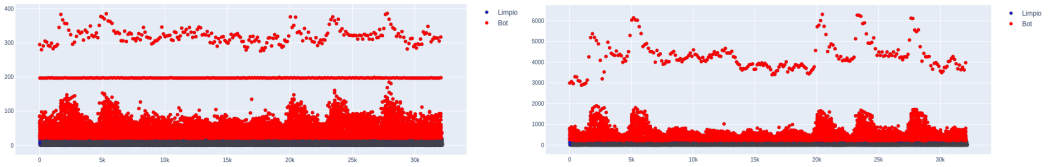
(e) P5

(f) P6



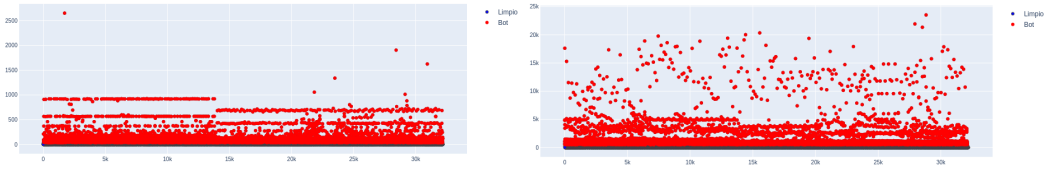
(g) P7

(h) P8



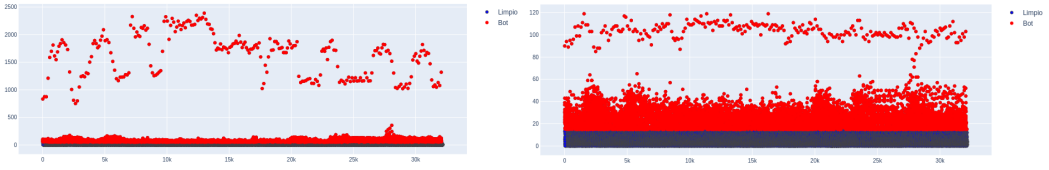
(i) P9

(j) P10



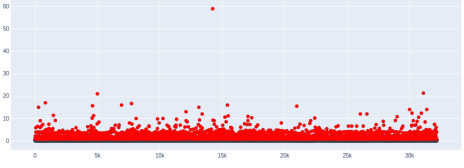
(k) P11

(l) P12



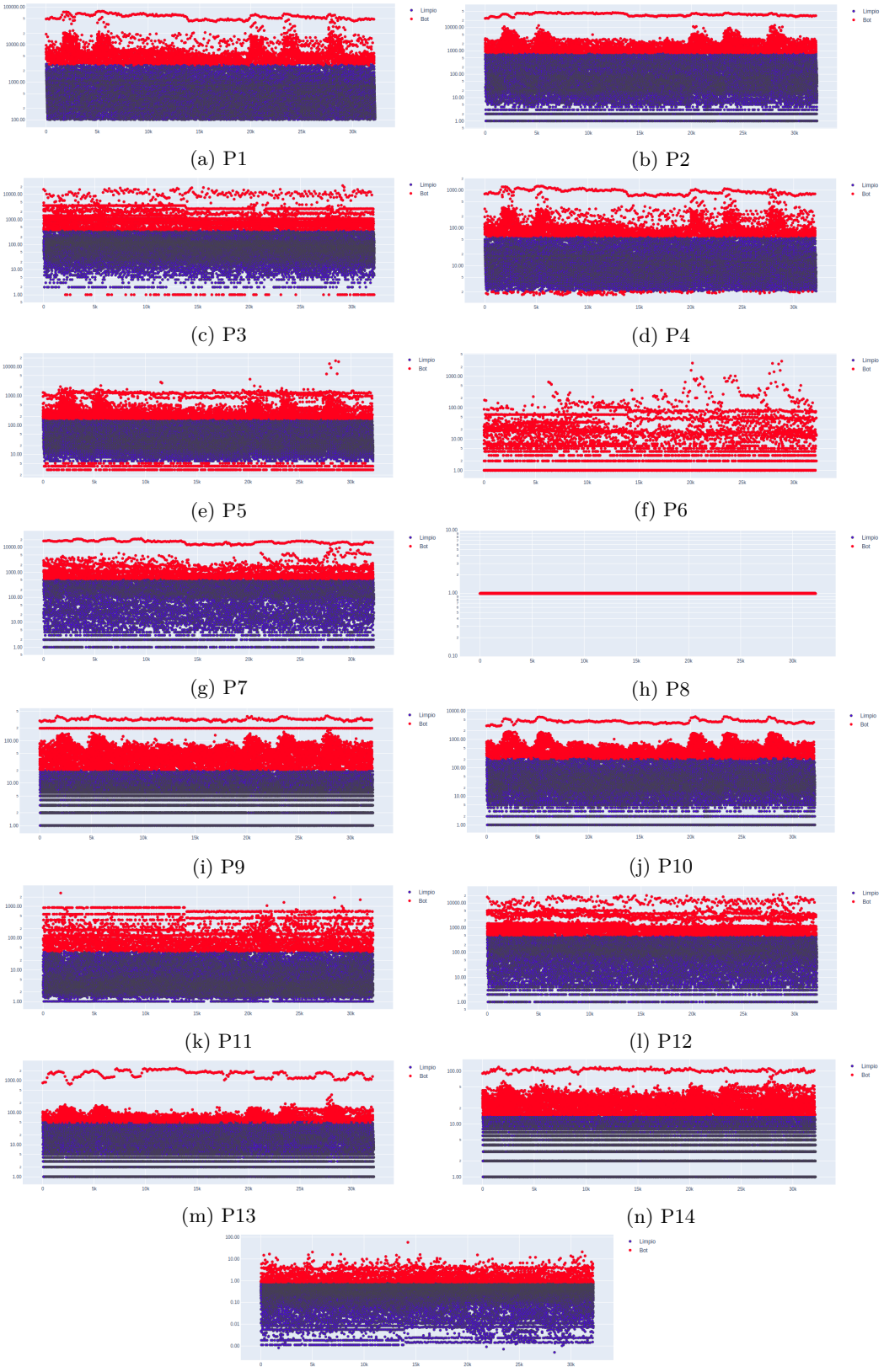
(m) P13

(n) P14



(ñ) P15

Figura 4.12: Clasificación de las huellas en función de sus atributos a escala lineal.



(a) P1 (b) P2

(c) P3 (d) P4

(e) P5 (f) P6

(g) P7 (h) P8

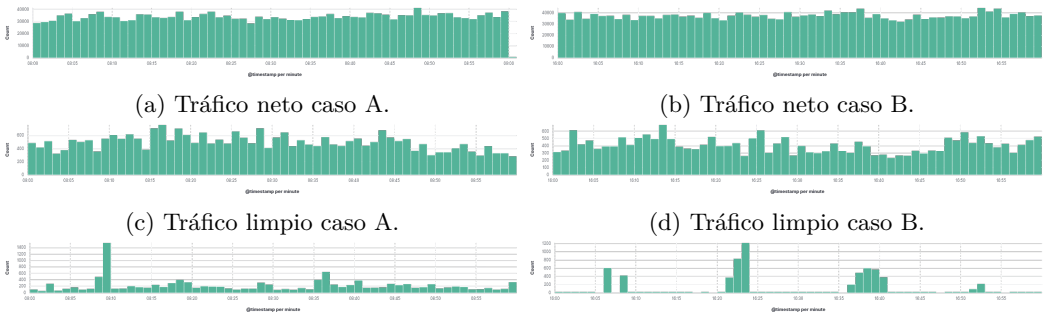
(i) P9 (j) P10

(k) P11 (l) P12

(m) P13 (n) P14

(ñ) P15

Figura 4.13: Clasificación de las huellas en función de sus atributos a escala logarítmica.



(a) Tráfico neto caso A.

(b) Tráfico neto caso B.

(c) Tráfico limpio caso A.

(d) Tráfico limpio caso B.

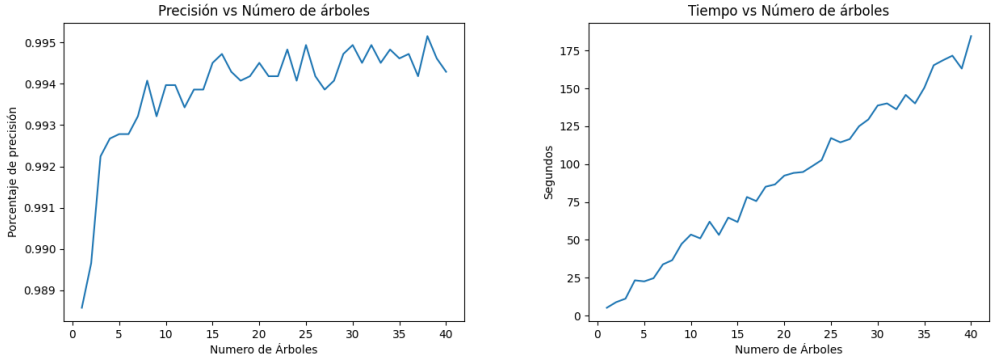
(e) Tráfico *bot* caso A.

(f) Tráfico *bot* caso B.

Figura 4.14: Casos A y B de tráficos DNS para neto, limpio y *bot*.

```
257 200.1.112.222-2021-02-11T16:00:00Z
258 ['bigdatapppp', 'hostingcloud', 'shalltry', 'bkrtx', 'appsflyer',
    'adsymptotic', 'koha-community', 'advertising',
    'liderancaspoliticas', 'itzmx', 'ecole-saint-simon', 'qualita-
    indonesia', 'netmng', 'amazon-adsystem', 'atdmt', 'rlcdn', 'adsrvr',
    'tclclouds', 'adnxs', 'flashtalking', 'rubiconproject',
    'spotxchange', 'navrcholu']
259 192.188.55.101-2021-02-11T16:00:00Z
260 ['w42f4ctqv4', 'bkrtx', 'appsflyer', 'adsymptotic', 'advertising',
    'servedbyadbutler', 'svraddspucesi', 'unrulymedia', 'imrworldwide',
    'netmng', 'switchadhub', 'credit-agricole', '3gppnetwork', 'amazon-
    adsystem', 'lnyrczvmoyzro', 'atdmt', 'worldtime', 'rlcdn',
    'crwdcntrl', 'adsrvr', 'xnycmjmulrsopfmm3x', 'tribalfusion',
    'reimageplus', 'fwmrm', 'tclclouds', 'adnxs', 'flashtalking', 'mig-
    bj-secrity-elb01-1569645166', '_msdcs', 'cm-launcher-sdk-36022841',
    'dbankcloud', 'wwwlanuevacicinadelecuador', 'rubiconproject',
    'spotxchange', 'stickyadstv', '_nfsv4idmapdomain', 'infoc-cms-
    recv-839538381']
261 192.188.59.2-2021-02-11T16:00:00Z
262 ['xcessbvdxkyfcd', 'vbvddqdkpztepje', 'rfc-ignorant',
    'mednewprilgusr', 'bkrtx', 'mbubnpmthb', 'malodocumentverification',
    'appsflyer', 'cjmhfnohbdpv', 'jxvcrfacpz', 'lankdtxnnryw',
    'yzwyqpidugbu', 'ajatpdoflanfdlg', 'adsymptotic', 'zhzfikgs',
    'rxvsgqzoqzak', 'sugsangbzqigwx', 'hwclouds-dns', 'advertising',
    'mmpkbzkvv', 'faqhctzowjwruw', 'xniwnflzschfrxz', 'pmqmxsdhlqbkymq',
    'bggrupo', 'unrulymedia', 'dayxqbqbue', 'lcilragpufqlrzd',
    'lfstmedia', 'pwgsdip', 'pmiemzhn', 'imrworldwide', 'serverhubrdns',
    'galiodocuments', 'bcuqjiqtffyeih', 'netmng', 'kcbyjwwpzn', 'amazon-
    adsystem', 'vgrcbteugogsjyi', 'cysjpjqsnj', 'kcqhnajaorfi',
    'ilaauvimvndm', 'securitydata', 'iugifcwzg', 'udsjljklodl', 'atdmt',
    'rlcdn', 'crwdcntrl', 'adsrvr', 'tribalfusion', 'cnt-grms',
    'cqmlfaikwsg', 'adnxs', 'flashtalking', 'wgfdara', 'company-target',
    'kxdzzmwkpwlmdvq', 'rjvcbfbrc', 'xuzkojhkymbmfd', 'fbzqrgchlubt',
    'dbankcloud', 'gpknmxkibzgprg', 'ergrogqenpgxfng', 'rubiconproject',
    'domgclqliujlo', 'lxwoddblknau', 'spotxchange',
    'ecommercedesignweb', 'skxulbkh', 'zvmypzbxgdpj', 'prod-infinitum',
    'vyktkqhp']
```

Figura 4.15: Algunos *hosts* y sus dominios SLD catalogados como sospechosos.



(a) Precisión. (b) Tiempo de procesamiento.

Figura 4.16: Parámetros de precisión y tiempo de entrenamiento en función del número de árboles para el algoritmo *RandomForest*

	precision	recall	f1-score	support
-1	0.96	0.92	0.94	400
1	1.00	1.00	1.00	8882
accuracy			0.99	9282
macro avg	0.98	0.96	0.97	9282
weighted avg	0.99	0.99	0.99	9282

Figura 4.17: Reporte de clasificación para el modelo de *Random Forest* con 25 árboles.

Parámetro	Descripción	Porcentaje
P10	Número de dominios SLD distintos consultados por hora	0.155
P2	Numero de solicitudes DNS distintas por hora	0.11
P1	Numero de solicitudes DNS por hora	0.109
P4	Numero medio de solicitudes por minuto	0.101
P13	Número de ciudades distintas de direcciones IP resueltas	0.093
P14	Número de países distintos de direcciones IP resueltas	0.072
P9	Número de dominios de TLD distintos consultados por hora	0.071
P3	Mayor cantidad de solicitudes para un solo dominio por hora	0.053
P7	Número de consultas de registros PTR por hora	0.051
P5	La mayor cantidad de solicitudes por minuto	0.046
P12	Número de consultas fallidas / NXDOMAIN por hora	0.045
P15	Relación de flujo por hora	0.037
P6	Número de consultas de registros MX por hora	0.032
P11	Relación de unicidad por hora	0.024
P8	Número de servidores DNS distintos consultados por hora	0

Figura 4.18: Tabla de porcentaje del valor de importancia de cada característica para el modelo de *Random Forest* con 25 árboles.

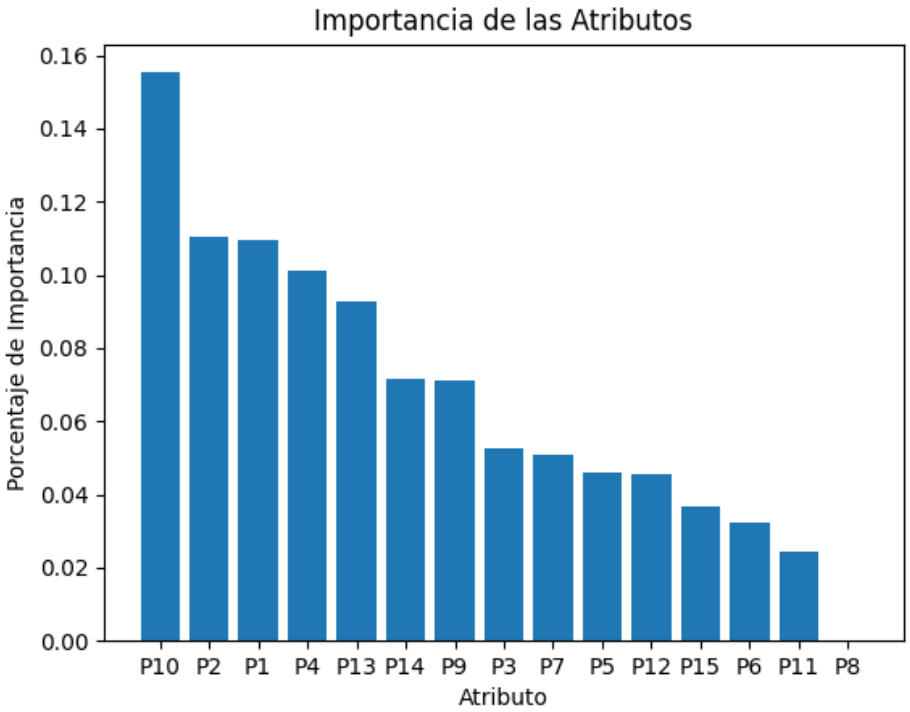


Figura 4.19: Gráfica de nivel de importancia de cada característica para el modelo de *Random Forest* con 25 árboles.

4.3. Conclusiones

El capítulo abarcó los temas de la implementación del sistema y la evaluación de los resultados de su aplicación.

Para la sección de implementación se detalla los procedimientos para la instalación y configuración de los diferentes módulos del sistema, esta sección se complementa con la sección de anexos y los archivos con dirección web `https://github.com/ViQuezada/Bootnet_Detection_Modules`.

En la sección de resultados se evalúan las salidas de los diferentes módulos del sistema. La integración de Suricata con la pila ELK permite generar los eventos DNS con las características necesarias (según el procedimiento presentado en [6]) para generar las huellas dactilares DNS de los *hosts*. Las huellas son generadas mediante consulta externa hacia Elasticsearch por Python. Se obtuvo un *dataset* de 32182 huellas generadas por 241 *hosts* distintos en un periodo de 10 días. La aplicación de *Insolation Forest* para agrupación, cataloga al 7 % del total de las huellas como anómalas. Las huellas catalogadas como anómalas pertenecen a un total de 58 *hosts*.

Comparar la cantidad de consultas para una misma hora para los escenarios de tráfico total, tráfico de un *host* limpio y tráfico de un *host bot*; permitió hacer visible las comunicaciones DNS en ráfagas que se generan en los *hosts* catalogados como *bot*. Este tipo de comunicación sería difícil de detectar haciendo uso de un análisis general del tráfico neto. Pero, es viable encontrar este tipo de anomalías con la técnica de detección de *hosts* infectados por *bot*. También, el algoritmo de detección de aleatoriedad de dominios reveló la existencia de huellas con ráfagas de consultas fallidas hacia SLDs aleatorios.

La aplicación de *Random Forest* con una cantidad de 25 árboles sobre las huellas catalogadas, generó un modelo de ML con una precisión del 0,995. Además, se logró obtener información de las características más relevantes para catalogar a un *host* como *bot*, teniendo a la cantidad de SLDs distintos como principal. La cantidad de consultas por horas y minutos también juegan un rol importante, seguido de las características de geo-localización.

Capítulo 5

Conclusiones.

En este apartado se realizará las conclusiones del trabajo experimental realizado y una visión de trabajos futuros relacionados a este proyecto.

5.1. Conclusiones

La revisión del estado del arte permitió definir la detección de anomalías de tráfico DNS en la red, como una metodología para la detección de *hosts* infectados por *bot*. El ciclo de vida de una *botnet* guarda dependencia con el servicio de DNS, como herramienta para resolver dominios de los servidores de C&C y poder establecer las conexiones con los *bots*. También permite migrar las direcciones IPs de los servidores C&C y evadir listas negras. Los campos de investigación actuales para detección de *botnets* basados en DNS, se centran en las técnicas de detección de *hosts* infectados por *bot* y las técnicas de detección basada en DGA. Estas técnicas fueron abordadas en este proyecto. La primera mediante la creación de huellas digitales de los *hosts* en función de 15 características DNS. Y la segunda mediante un algoritmo de medición de aleatoriedad de un dominio, en función del valor de entropía y cantidad de vocales y consonantes secuenciales. La arquitectura escogida para el proyecto permitió implementar un laboratorio virtual, que por su naturaleza de IDS actúa de forma pasiva sobre la red. La captura y generación de eventos se la realiza por medio de *port mirroring*. El procesamiento de los eventos capturados se desarrolla de manera *offline*. Así, el sistema no representa ninguna amenaza hacia el normal funcionamiento de la red. Los resultados obtenidos no contemplan una acción correctiva sobre la red. Mas bien, se presenta el proyecto como un sistema de alerta de detección de anomalías en eventos DNS que denotan actividades relacionadas a la existencia de infección en *hosts* por *bot*. La integración de las herramientas y los procesos presentados en este trabajo, permitió una gestión completa de eventos DNS de una red real. Para la captura de tráfico y generación de eventos DNS, Suricata IDS demostró ser una herramienta poderosa, reduciendo la necesidad de interacción directa con el flujo bruto de la red. La integración de la pila ELK como SIEM permitió una mejor gestión de la información de los eventos DNS, como la eliminación de campos irrelevantes y la integración de campos de geo-localización y división de un dominio en niveles. Los campos integrados demostraron

ser atributos de elevada importancia, al momento de entrenar un modelo automático de aprendizaje. El uso de Python permitió aprovechar el motor de búsqueda y analítica de Elasticsearch y generar las huellas dactilares DNS en función de los eventos almacenados. La metodología expuesta en [6] para detección de *hosts* infectados por *bot* usando huellas digitales DNS fue aplicada en este trabajo. La metodología contempla la extracción de 15 parámetros DNS por hora de los *hosts* para generar huellas dactilares y la aplicación de ML para la detección de anomalías. En un período de 10 días 241 hosts generaron 687′526,874 eventos que dieron paso a la creación de 32182 huellas dactilares DNS. El modelo de *Isolation Forest* con 110 árboles clasificó a 2287 como anómalas, que representan un 7 % del total de huellas y pertenecen a 58 *hosts*. Se descartaron las huellas de *hosts* cuyo porcentaje de huellas anómalas superaba el 50 % del total de sus huellas generadas, considerando la premisa de que en detección de anomalías se debe tener un mayor porcentaje de datos normales que anormales. Con las huellas clasificadas se entrenó un modelo de *Random Forest* con 25 árboles, consiguiendo un precisión del 0,995.

Se observó como el método de detección de *hosts* infectados por *bot*, permite la detección de anomalías de tráfico que no aparecen al analizar el tráfico completo de la red y son un patrón común de comunicaciones de *bots*. También la aplicación de un algoritmo de detección de aleatoriedad de dominios sobre las huellas catalogadas como *bot*, permitió verificar la existencia de varias consultas fallidas como ráfagas. Las consultas eran realizadas hacia nombres de dominio con características de aleatoriedad, propio de *bots* DGA.

El modelo generado por *Random Forest* brindó información sobre el nivel de importancia que cada característica de las huellas tiene para la detección de huellas perteneciente a *bots*. Esta información es concordante con la teoría revisada en la sección del marco teórico de este trabajo. La búsqueda de comunicación de un *bot* con su *botmaster*, genera aumentos en la cantidad de las consultas DNS en un *host* infectado. A su vez en un intento de evadir elementos de seguridad se genera variedad, tanto en los nombres de dominio como en su lugar de ubicación geográfica. El algoritmo de aprendizaje automático es sensible a estos parámetros y por su alta precisión obtenida se considera al sistema efectivo para la detección de futuros *hosts* infectados por *bot* en dicha red. Para expandir el campo de aplicación del sistema, los recursos computacionales y de red fueron gestionados con CEDIA, en un marco de desarrollo de herramientas enfocado a la ciberseguridad. Así, este trabajo abre paso a proyectos entorno al análisis de *logs* como métodos de detección de intrusiones de seguridad. A la vez, permite trabajar sobre una red del mundo real en una plataforma estable de captura y procesamiento de eventos de red.

5.2. Trabajos Futuros

Implementar más puntos de análisis de tráfico y generación de eventos DNS a manera de enriquecer el *dataset* de huellas dactilares. La arquitectura distribuida del sistema permite tener varios puntos con Suricata IDS como generadores de eventos. Estos eventos pueden ser enviados a Logstash por medio de agentes como Filebeat. Esta solución es aplicable para aquellos *hosts* que fueron descartados en este trabajo por su comportamiento siempre anómalo. Así, este *host* descartado se reemplazaría por el conjunto de *hosts* que representa.

Generar una lista blanca de dominios internos. Buscando así, que la aplicación del algoritmo de detección de aleatoriedad de dominios reduzca su tasa de falsos positivos. Esto permitirá tener una

técnica mejorada de detección de *bots* basados en DGA.

Explorar otras técnicas de detección de intrusiones de seguridad basado en el análisis de *logs*. La arquitectura del laboratorio implementado incluye Suricata con la capacidad de generar eventos de tipo HTTP, FTP, TLS, SSH, Flow y MQTT, y la pila ELK que permite gestionar todo tipo de eventos en diferentes formatos.

Anexo A

Instalación y configuración del *software* requerido para el proyecto.

En este anexo se detalla las instalación y configuración de las herramientas de *software* necesarias para el proyecto. Entre las herramientas principales se tiene a Suricata y la pila ELK, cada una con su respectivas. Todos los procesos aquí detallados aplican para el OS Ubuntu 20.04.1 LTS.

A.1. Instalación y configuración de Suricata

Para la instalación de Suricata IDS la documentación oficial se encuentra en [25]. Se toma como guía de refuerzo los manuales presentados en [26, 27] para Ubuntu.

A.1.1. Instalación de dependencias

Las dependencias recomendadas para el funcionamiento de Suricata se instala con el comando del Listado A.1.

```
# apt-get install libpcre3 libpcre3-dbg libpcre3-dev build-essential libpcap-dev
    libyaml-0-2 libyaml-dev pkg-config zlib1g zlib1g-dev make libmagic-dev
```

Listado A.1: Instalación de paquetes dependientes para Suricata.

A.1.2. Instalación de Suricata

La instalación de Suricata se la realiza con las instrucciones del Listado A.2 sobre el terminal.

```
# add-apt-repository ppa:oisf/suricata-stable
# apt-get update
# apt-get install suricata
```

Listado A.2: Instalación de Suricata.

El archivo etc/suricata/suricata.yml es el archivo de configuración de Suricata. Se deberá reemplazar todas las instancias de *eth0* con el nombre de la tarjeta de red sobre la que se realizará la captura.

Dicho adaptador puede ser consultado desde el terminal usando el comando `ifconfig`. Dado que se realizará un análisis externo de los datos, las configuraciones por defecto son suficientes.

A.1.3. Suricata como servicio

Suricata se establece como el servicio `suricata` o `suricata.service` al cual se puede iniciar, detener, o reiniciar; *start*, *stop* y *restart* respectivamente. Además, es necesario registrar y activar el servicio para un inicio automático. Los comandos sobre terminal para obtener los resultados antes mencionados se presentan en el Listado A.3.

```
# /bin/systemctl daemon-reload
# /bin/systemctl enable suricata.service
# service suricata start
```

Listado A.3: Activación e inicio de Suricata.

A.1.4. Modo promiscuo sobre una tarjeta de red en Ubuntu 20.04.1 LTS

La habilitación del modo promiscuo de una tarjeta se la puede realizar por medio del terminal. Se requiere los permisos de *root* y el nombre de la tarjeta en cuestión. El comando de activación de modo promiscuo se presenta en el Listado A.4.

```
# ifconfig [nombre de la tarjeta] promisc
```

Listado A.4: Configuración de tarjeta de red a modo promiscuo.

A.2. Instalación y configuración de la pila ELK sobre Ubuntu 20.04 LTS

Las guías oficiales para la instalación de las herramientas que conforman la pila ELK, se las puede encontrar en [28–30] respectivamente. Se toma como guía de refuerzo los manuales presentados en [26, 27].

A.2.1. Paquetes dependientes

La instalación de los paquetes dependientes recomendados para el funcionamiento de la pila ELK se presenta en el Listado A.5.

```
# apt -y install libnetfilter-queue-dev geoip-bin geoip-database geoipupdate apt-
    transport-https
```

Listado A.5: Instalación de paquetes dependientes para la pila ELK.

Los *kits* de desarrollo de Java 8 y 11 deben ser instalados. Los comandos de instalación se presenta en el Listado A.6.

```
# apt install openjdk-8-jdk
# apt install openjdk-11-jdk
```

Listado A.6: Instalación de paquetes de Java

A.2.2. Instalación y configuración de Elasticsearch

A.2.2.1. Instalación de Elasticsearch

Agregar el repositorio elastic.co que contiene la versión más actualizada (en este caso la versión 7) de la pila ELK a los repositorios del computador. Y proceder a instalar Elasticsearch. Los comando para este proceso se presentan en el Listado A.7.

```
# wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch
# apt-key add
# echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main"
# sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
# apt update
# apt -y install elasticseach
```

Listado A.7: Instalación de Elasticsearch.

A.2.2.2. Elasticsearch como servicio

Elasticsearch se establece como el servicio elasticsearch o elasticsearch.service al cual se puede iniciar, detener, o reiniciar; *start*, *stop* y *restart* respectivamente. Además es necesario registrar y activar el servicio para un inicio automático. Los comandos para los procesos antes mencionados se presentan en el Listado A.8.

```
# /bin/systemctl daemon-reload
# /bin/systemctl enable elasticsearch.service
# service elasticsearch start
```

Listado A.8: Activación e inicio de Elasticsearch.

A.2.2.3. Acceso y consulta remota a Elasticsearch

Anticipando la necesidad de acceder de forma externa a la información de Elasticsearch, es necesario realizar la configuración para que escuche no solo en *localhost* que viene por defecto. Así, se debe agregar las líneas del Listado A.9 al archivo de configuración de Elasticsearch ubicado en /etc/elasticsearch/elasticsearch.yml.

```
transport.host: localhost
transport.tcp.port: 9300
http.port: 9200
network.host: 0.0.0.0
```

Listado A.9: Habilitación de acceso remoto a Elasticsearch.

Se debe considerar que network.host como 0.0.0.0 permitirá la vinculación desde cualquier *host* que tenga acceso a la red.

A.2.3. Instalación y configuración de Logstash

A.2.3.1. Instalación de Logstash

Con los repositorios de elastic.co agregados con anterioridad, se procede a instalar de forma directa Logstash. Se concede permisos de administrador para que pueda leer archivos del equipo. Se registra y

activa el servicio para el inicio automático. Y finalmente, se arranca para dar inicio a sus operaciones. Los comandos para los procesos antes mencionados se presentan en el Listado A.10.

```
# apt -y install logstash
# usermod -a -G adm logstash
# /bin/systemctl daemon-reload
# /bin/systemctl enable logstash.service
# service logstash start
```

Listado A.10: Instalación e Inicio de Logstash.

A.2.3.2. MaxMind GeoIP Database

Dos atributos basados en la IP, necesarios para la generación de las huellas digitales de los *hosts*, son el número distinto de ciudades y número distinto de países consultados(P13 y P14 respectivamente). Es necesario disponer de una base de datos que relacione IPs con lugares geográficos. Maxmind[1] es una empresa que brinda datos de ubicación geográfica para direcciones IP. La empresa proporciona inteligencia IP a través de la marca GeoIP. Entre sus productos se puede encontrar la base de datos GeoLite2[2] para geo-localización gratuita de IPs. La base de datos es comparable, pero menos precisa, que las opciones de pago. Se debe considerar un radio de precisión para la ubicación resuelta.

Para obtener un acceso de descarga gratuita, se debe registrar en el portal de GeoLite2. El portal permite el acceso a varios formatos de la base de datos. El formato a utilizar para este proyecto es GeoLite2-City.mmdb. El archivo se almacenará en el directorio /usr/share/GeoIP/ para su posterior consulta.

A.2.3.3. Filtro TLD de Logstash

De los atributos DNS requeridos de los *hosts*, existen dos relacionados al nombre dominio consultado. Los atributos son número de TLD y SLD distintos consultados por los *hosts*, como P9 y P10 respectivamente. En una consulta DNS el nombre de dominio es consultado como una sola cadena. Los servidores se encargan de dividir el nombre de dominio por niveles para resolver la consulta.

Logstash posee un *plugin* llamado logstash-filter-tld[3] que permite dividir un dominio por sus niveles, y así, obtener los campos TLD y SLD del mismo. Se instalará el *plugin* y posteriormente será utilizado en el archivo de configuración de Logstash. El comando de instalación del *plugin* en Ubuntu se presenta en el Listado A.11.

```
# /usr/share/logstash/bin/logstash-plugin install logstash-filter-tld
```

Listado A.11: Instalación de filter -tls de Logstash.

A.2.3.4. Entrada, procesamiento y salida de registros en Logstash

Se debe definir un archivo que contenga la configuración para el tratamiento de los *logs* generados por Suricata y el envío de esta información hacia Elasticsearch. Se genera el archivo suricata.conf de configuración en el directorio /etc/logstash/conf.d. Sobre el archivo se definirán las configuraciones para la entrada, procesamiento y salida de la información que llega a Logstash. Para la entrada, la

[1] https://www.maxmind.com/en/home
[2] https://dev.maxmind.com/geoip/geoip2/geolite2/
[3] https://www.elastic.co/guide/en/logstash/current/plugins-filters-tld.html

configuración incluye: el directorio de eventos de Suricata /var/log/suricata/eve.json, el tipo de aplicación que los genera y el formato en que han sido generados. La configuración de la entrada se presenta en el Listado A.12.

```
input {
  file {
    path => ["/var/log/suricata/eve.json"]
    sincedb_path => ["/var/lib/logstash/sincedb"]
    codec => json
    type => "suricata"
} }
```

Listado A.12: Configuración de la entrada de Logstash.

Para el procesamiento, el filtro aplica sobre los eventos Suricata ingresados un nuevo campo denominado geoip. El campo asocia las coordenadas IP de las solicitudes DNS resueltas (tanto para IPv4 como para IPv6) con referencias geográficas, esto con la ayuda de la base de datos GeoipLite2. Se hace uso del plugin tld para la creación de un nuevo campo llamado dn, para los atributos de SLD y TLD de los dominios consultado. Finalmente, se aplica el filtro mutate para eliminar varios campos de las consultas y respuestas DNS que no son de interés para este proyecto. El uso del filtro permite optimizar el espacio de almacenamiento. La configuración del procesamiento de los eventos por Logstash se detalla en el Listado A.13.

```
filter {
  if [type] == "suricata" {
    date {
      match => [ "timestamp", "ISO8601" ]
    }
    ruby {
      code => "if event['event_type'] == 'fileinfo'; event['fileinfo']['t>
  }  }
  mutate{
    remove_field => ["timestamp","flow_id","in_iface","proto","[dns][id]"]
  }

  tld {
    source => "[dns][rrname]"
    target => "dn"
  }

  if [src_ip]  {
    geoip {
      source => "[dns][grouped][A][0]"
      target => "geoip"
      database => "/usr/share/GeoIP/GeoLite2-City.mmdb"
    }
    if ![geoip.ip] {
      if [dest_ip] {
        geoip {
          source => "[dns][grouped][AAAA][0]"
          target => "geoip"
          database => "/usr/share/GeoIP/GeoLite2-City.mmdb"
  } } } }
```

```
  mutate{
    remove_field => ["[geoip][continent_code]","[geoip][country_code2]",">
    remove_field => ["type","tags","path","host","[dns][qr]","[dns][flags>
    remove_field => ["[dns][aa]","[dns][answers]","[dns][version]","[dns]>
    remove_field => ["[dns][tx_id]"]
    remove_field => ["[dn][domain]","[dn][subdomain]","[dn][trd]"]
} }
```

Listado A.13: Configuración del procesamiento de Logstash.

Se configura la salida de los *logs* generados hacia índices en Elasticsearch con escucha en el puerto 9200. Los índices se crearán en función de la fecha y el tipo de evento que haya capturado Suricata. La configuración de la salida se detalla en el Listado A.14.

```
output {
  if [event_type] and [event_type] != 'stats' {
    elasticsearch {
      hosts => ["http://"direccion_ip":9200"]
      index => "logstash-%{event_type}-%{+YYYY.MM.dd}"
    }
  }else {
    elasticsearch {
      hosts => ["http://"direccion_ip":9200"]
      index => "logstash-%{+YYYY.MM.dd}"
} } }
```

Listado A.14: Configuración de la salida de Logstash.

A.2.4. Instalación y configuración de Kibana

A.2.4.1. Instalación de Kibana

Con los repositorios de elastic .co (agregados con anterioridad) se procede a instalar de forma directa Kibana. Se registra y activa el servicio para un inicio automático. Y finalmente se arranca para dar inicio a sus operaciones. Los comandos para los procesos antes mencionados se presentan en el Listado A.15.

```
# apt -y install kibana
# /bin/systemctl daemon-reload
# /bin/systemctl enable kibana.service
# service kibana start
```

Listado A.15: Instalación e inicio de Kibana.

A.2.4.2. Acceso remoto a Kibana

Se anticipa la necesidad de acceder de forma externa a la interfaz de Kibana, siendo necesario configurarla para que escuche no solo en *localhost* que viene por defecto. Así, se debe agregar las líneas del Listado A.16 al archivo de configuración de Kibana ubicado en /etc/kibana/kibana.yml.

```
server.port: 5601
server.host: "0.0.0.0"
```

Listado A.16: Configuración para acceso remoto a Kibana.

Se debe considerar que server.host como 0.0.0.0 permitirá el acceso y vinculación desde cualquier interfaz.

A.2.4.3. Interacción con Kibana

Kibana permite tener una visualización de cómo se están generando los índices por Logstash en Elasticsearch, y la información que existe en cada uno de sus registros. Esto permite corroborar que la información se esté obteniendo de la manera deseada. Se puede ingresar a la consola de Kibana sobre un navegador, apuntando a la dirección y puerto en el cual está escuchando. Así, con la dirección http://direccion_kibana:5601 sobre el navegador se podrá dirigir a la ventana principal.

A.2.4.4. Creación de patrones de índices

Para este proyecto es de interés tener una visualización de la información capturada por Suricata. Se desea verificar su correcto funcionamiento, además de sus campos para consultas posteriores. Por lo que se crea un patrón de índices para los *logs* DNS capturados. La Figura A.1 muestra como sobre la ventana Stack Management/Index patterns/Create index pattern de Kibana se crea el patrón de índices logstash−dns∗. El patrón permite la visualización de los eventos DNS almacenados en Elasticsearch en un rango de tiempo seleccionado.

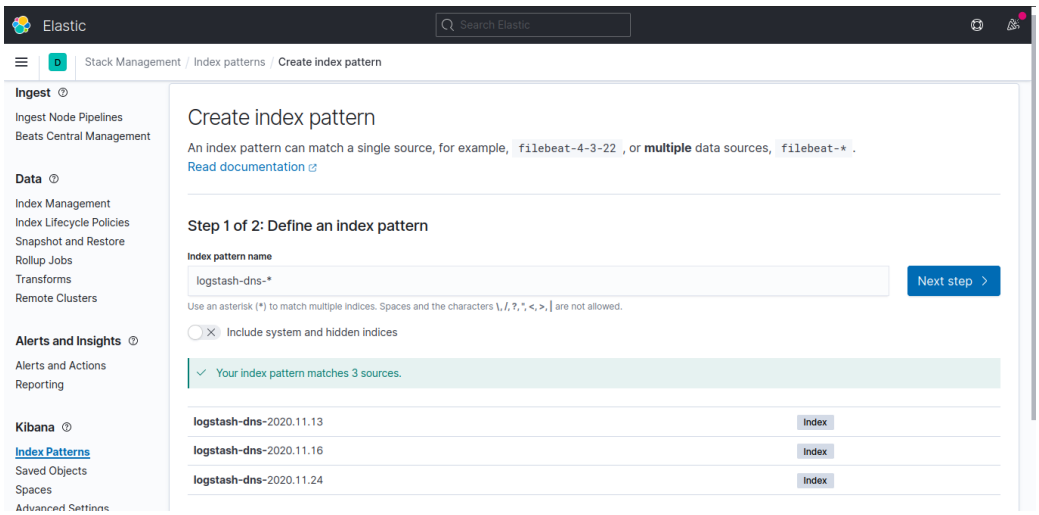


Figura A.1: Ventana de creación de patrón de índice en Kibana.

Una vez creado el patrón de índices, en la pestaña de *Discover* de Kibana se puede observar los eventos DNS capturados. La Figura A.2 muestra un ejemplo de visualización de la ventana *Discover*.

A.2.4.5. Consola de desarrollo

Otra pestaña de interés se muestra en la Figura A.3. La pestaña es la consola de desarrollo de Kibana, que actúa como un terminal avanzado permitiendo afinar consultas hacia Elasticsearch, observar los resultados y su formato. El ejemplo muestra parte de la consulta de la cantidad de consultas DNS realizadas por los 216 *hosts* más activos en forma descendente, para la fecha de 16/11/2020. Este mismo formato de "consulta-respuesta" será el que se deba usar desde Python.

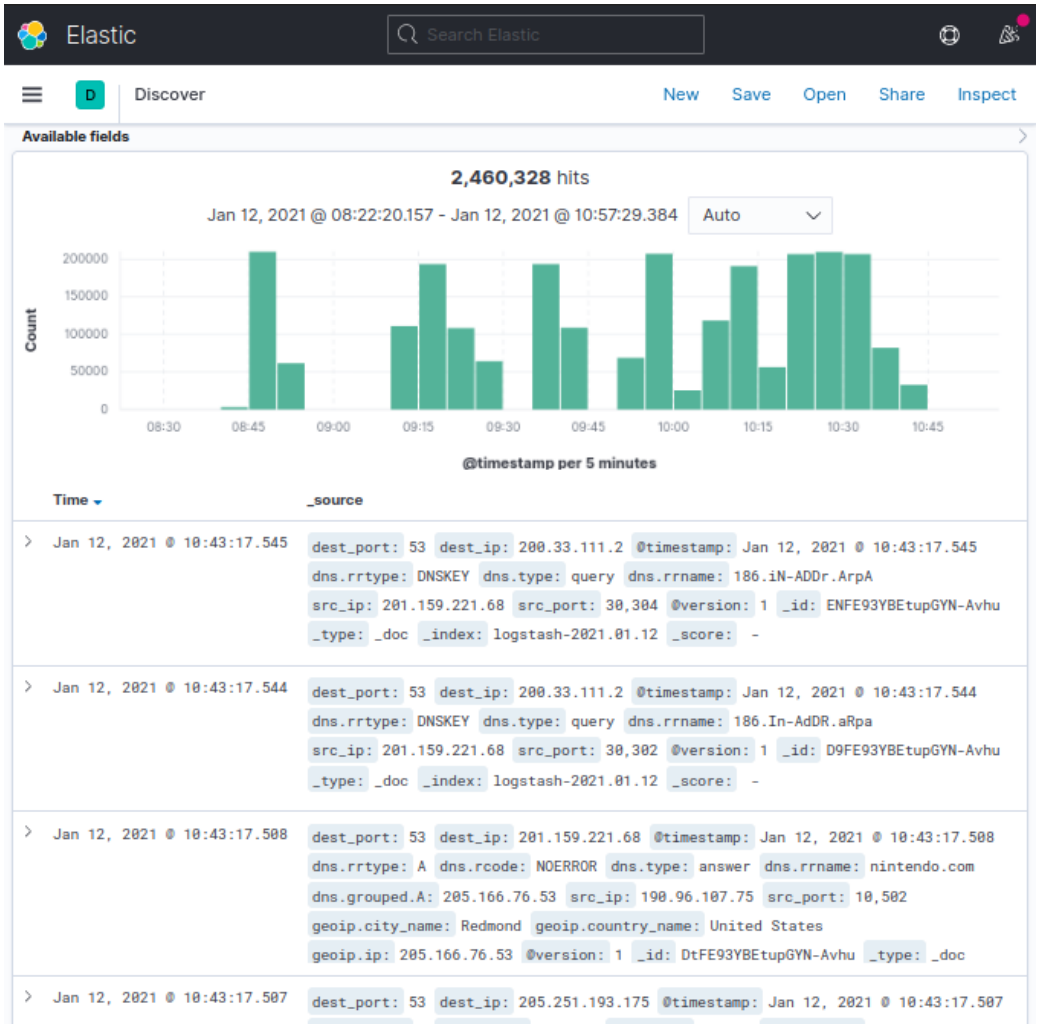


Figura A.2: Ventana *Discover* de Kibana.

A.2.4.6. Creación de *Dashboards* y visualizaciones

Kibana en su sección Dashboard permite generar varios tipos de visualizaciones como: Tablas, Metricas, Mapas, Barras Horizontales, Barras Verticales, entre otros. En la Figura A.4 se presenta las visualizaciones disponibles. Las visualizaciones permiten, de un compendio de eventos, presentar de una manera más amigable estadísticas sobre los atributos de los mismos. Así el *Dashboard* puede contener el número de visualizaciones que se desee, con el fin de hacer a la información más accesible y útil. En la Figura A.5 se muestra el *Dashboard* generado para el proyecto que permite observar: la cantidad de huellas totales, la cantidad de huellas para cada clasificación (limpio, *bot*), las IPs relacionadas a las huellas *bot*, entre otros parámetros de interés. Cabe recalcar que estas visualizaciones son sensibles a la marca de tiempo, por lo que sus valores cambiarán en función de período analizado.

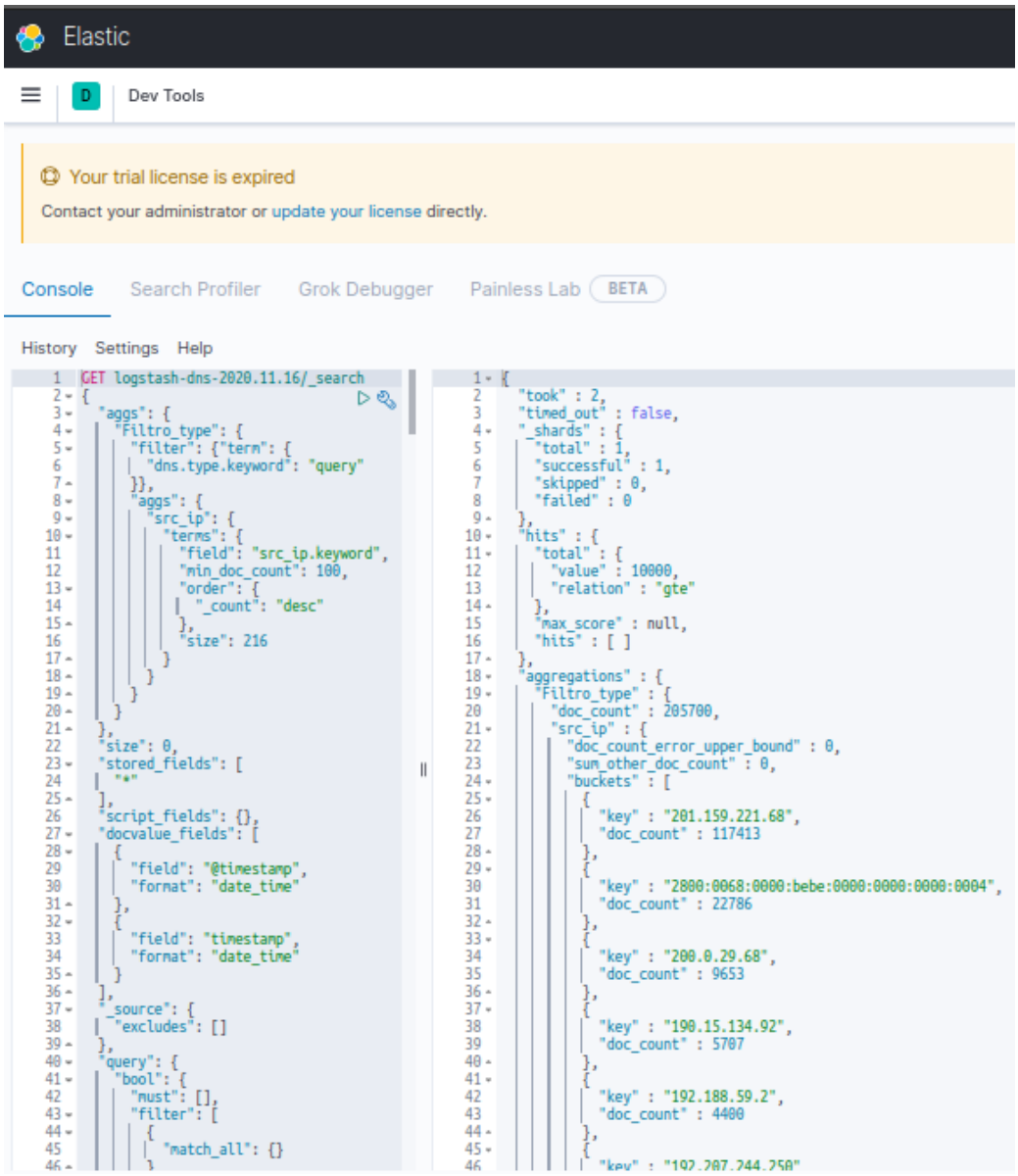


Figura A.3: Ventana de consola de Kibana.

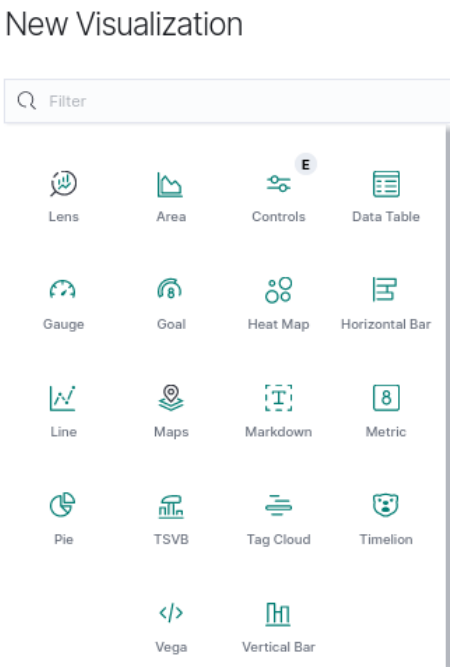


Figura A.4: Visualizaciones disponibles en Kibana.

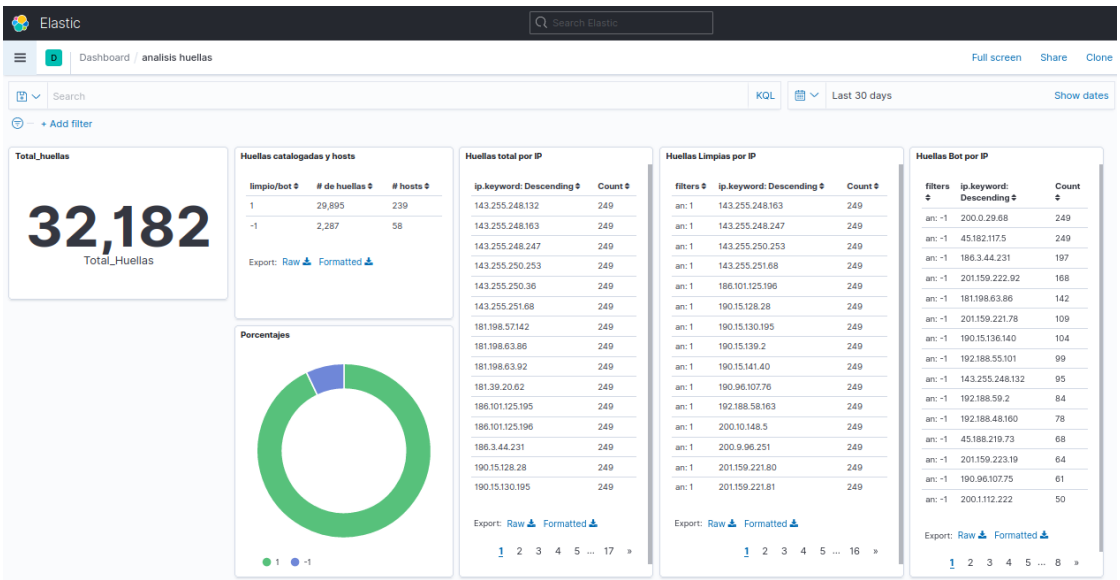


Figura A.5: *Dashboard* creado en Kibana.

Anexo B

Generación de huellas digitales DNS

En este apartado se detalla el procedimiento para generar las huellas digitales DNS de los host. Python es el lenguaje utilizado para crear un cliente externo, el cual realizará consultas a Elasticsearch. Se busca que el motor de búsquedas y analítica de Elasticsearch resuelva la mayor parte de la complejidad computacional. Con la información obtenida de las consultas, se procede a formar un *dataset* de huellas digitales DNS de los *hosts*.

B.1. Conexión Python-Elasticsearch

Habiendo resuelto en el Anexo A el acceso de clientes externos a Elasticsearch, se procede a crear un cliente Python. La documentación oficial para la interacción Python-Elasticsearch se la puede encontrar en [31]. La conexión se la realiza con el código de Listado B.1, considerando la dirección del servidor de Elasticsearch y el puerto 9200, como el puerto de comunicación.

```
from elasticsearch import Elasticsearch
try:
  es = Elasticsearch([{'host':'direccion_elasticsearch','port':9200,}])
  print (''Connected'')
except Exception as ex:
  print (''Error:'', ex)
```

Listado B.1: Conexión a Elasticsearch como cliente desde Python.

Una vez conectado a Elasticsearch se puede realizar consultas. Es necesario definir la dirección, encabezado y cuerpo de la consulta como se muestra en el Listado B.2, donde response será la variable que contenga la respuesta de la consulta.

```
import requests

HEADERS = {'Content-Type': 'application/json'}
Uri = ''dirección IP elasticsearch''+''/''+''indice DNS''+''/_search''
Query = json.dumps('consulta a realizar')
```

```
response = requests.get(uri, headers=HEADERS, data=query).json()
```

Listado B.2: Partes de una consulta a Elasticsearch desde Python.

B.2. Generación de huellas digitales

Las características de P1 a P15 se obtienen mediante consultas hacia Elasticsearch. Según la naturaleza de los parámetros requeridos, será necesario la consulta hacia ciertos campos de las peticiones y respuestas DNS. Así, se detalla en la Tabla B.1 las operaciones de búsqueda y las condiciones de los campos para obtener cada una de las características requeridas. En la columna Operador se presenta los operadores propios de Elasticsearch. En la columna Campo se utiliza el nombre de los campos como Logstash los almacenó en los índices de Elasticsearch.

Tabla B.1: Resumen de operadores y campos necesarios para la obtención de las características P1 a P15.

Atributo	Operador	Campo
P1	filter	dns.type.keyword : "query"
	count	src_ip.keyword
P2	filter	dns.type.keyword : "query" src_ip.keyword : "IP del host de turno"
	cardinality	dns.rrname.keyword
P3	filter	dns.type.keyword : "query" src_ip.keyword : "IP del host de turno"
	count	dns.rrname.keyword
P4	filter	dns.type.keyword : "query" src_ip.keyword : "IP del host de turno"
	date_histogram	field: "@timestamp" fixed_interval: "1m"
P6	filter	dns.type.keyword : *"query"* src_ip.keyword : "IP del host de turno" dns.rrtype.keyword : *"MX"*
P7	filter	dns.type.keyword : "query" src_ip.keyword : "IP del host de turno" dns.rrtype.keyword : *"PTR"*
P8	filter	dns.type.keyword : "query" src_ip.keyword : "IP del host de turno"
	cardinality	dest_ip.keyword
P9	filter	dns.type.keyword : "query" src_ip.keyword : "IP del host de turno"
	cardinality	dn.tld.keyword
P10	filter	dns.type.keyword : "query" src_ip.keyword : "IP del host de turno"
	cardinality	dn.sld.keyword
P12	filter	dns.type.keyword : "answer" src_ip.keyword : "IP del host de turno" dns.rrtype.keyword : *"NXDOMAIN"*
P13	filter	dns.type.keyword : "answer" src_ip.keyword : "IP del host de turno"
	cardinality	geoip.city_name.keyword
P14	filter	dns.type.keyword : "answer" src_ip.keyword : "IP del host de turno"
	cardinality	geoip.country_name.keyword
P15	filter	dns.type.keyword : "answer" src_ip.keyword : "IP del host de turno" dns.rcode.keyword : "NOERROR"

ANEXO C

Detección de anomalías en huellas digitales DNS

En esta sección se detalla los procedimientos para la categorización de las huellas dactilares como limpias o *bot*.

C.1. Detección de valores atípicos en las huellas digitales con *Isolation Forest* de Sklearn

Con un *dataset* de huellas digitales DNS se aplica *Isolation Forest* para detectar anomalías. De forma resumida y sin considerar algún procesamiento o ajuste al *dataset*, se puede desarrollar el algoritmo con el código presentado en el Listado C.1.

```
import numpy como np
import pandas como pd
from sklearn.ensemble import IsolationForest

df = pd.read_csv ("dataset.csv") #leer los datos

clf=IsolationForest(n_estimators=100, max_samples='auto', contamination='auto',
    max_features=1.0, bootstrap=False, n_jobs=-1, random_state=42, verbose=0) #
    construir el modelo

clf.fit(df) #ajustar el modelo al dataset

clf.predict(df) #predecir valores atípicos
```

Listado C.1: *Isolation Forest* en Python.

C.2. Reducción de dimensionalidad y visualización

Para un análisis visual de la clasificación realizada por *Isolation Forest* es necesario realizar un ajuste de dimensión de las huellas. Cada huella posee un total de 15 atributos. PCA[1] de *Scikit Learn*

[1] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

permite la reducción de dimensionalidad lineal para proyección sobre un espacio dimensional inferior. En el Listado C.2 se presenta a breves rasgos el uso de PCA en Python, considerando la existencia de un *dataset* catalogado.

```
from sklearn.decomposition import PCA

# df = dataset + columna de anomalías

pca = PCA(n_components=3)  #Reducir a k=3 dimensiones
scaler = StandardScaler()
X = scaler.fit_transform(df)
X_reduce = pca.fit_transform(X)
plt.show(X_reduce) #resumen de comando de visualización
```

Listado C.2: PCA en Python.

ANEXO D

Aprendizaje automático con *Random Forest*

Con un *dataset* de datos etiquetados, se aplica el algoritmo de *Random Forest* para aprendizaje automático. En el Listado D.1 se muestra la aplicación del algoritmo en su estado más básico.

```
from sklearn.ensemble import RandomForestClassifier

dataset = pd.read_csv("HuellasDactilaresCatalogadas.csv")

#Definir los campos predictores y el campo a clasificar
predictors = dataset[['P1','P2','P3','P4','P5',
                      'P6','P7','P8','P9','P10',
                      'P11','P12','P13','P14','P15']]
targets = dataset.anomaly

#Dividir el dataset en entrenamiento 0.7 y prueba 0.3
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, targets,
    test_size=.3, random_state=0)

#Armar, ajustar y entrenar el modelo de RandomForest
classifier=RandomForestClassifier(n_estimators=25)
classifier=classifier.fit(pred_train,tar_train)
predictions=classifier.predict(pred_test)

#mostrar matriz de confusión, reporte de clasificación, exactitud
print(sklearn.metrics.confusion_matrix(tar_test,predictions))
print(sklearn.metrics.classification_report(tar_test, predictions))
print(sklearn.metrics.accuracy_score(tar_test, predictions))
```

Listado D.1: Random Forest en Python.

Bibliografía

[1] Cisco Network Academy, *Introducción a la Ciberseguridad*. [En línea]. Disponible: https://static-course-assets.s3.amazonaws.com/CyberSec2.1/es/index.html{#}4.2.2.1

[2] K. Aasia, Abdullah; Afroaz, "Data Mining Approaches on Network Data: Intrusion Detection System," *International Journal of Advanced Research in Computer Science*, vol. 8, 2017. [En línea]. Disponible: https://search.proquest.com/openview/5933b1b75f9598b147bced6123f59c8e/1?pq-origsite=gscholar{&}cbl=1606379

[3] D. K. Bhattacharyya y J. K. Kalita, *Network Anomaly Detection*, 2013, num. November.

[4] T. S. Wang, H. T. Lin, W. T. Cheng, y C. Y. Chen, "DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis," *Computers and Security*, vol. 64, pp. 1–15, 2017. [En línea]. Disponible: http://dx.doi.org/10.1016/j.cose.2016.10.001

[5] M. Singh, M. Singh, y S. Kaur, "Issues and challenges in DNS based botnet detection: A survey," *Computers and Security*, vol. 86, pp. 28–52, 2019. [En línea]. Disponible: https://doi.org/10.1016/j.cose.2019.05.019

[6] ——, "Detecting bot-infected machines using DNS fingerprinting," *Digital Investigation*, vol. 28, pp. 14–33, 2019.

[7] M. Singh, "Anomaly based Botnet Detection using DNS Traffic Analysis," Ph.D. dissertation, Thapar Institute of Engineering & Technology. [En línea]. Disponible: http://hdl.handle.net/10266/5959

[8] Mannirulz, "BotDAD." [En línea]. Disponible: https://github.com/mannirulz/BotDAD

[9] J. García Merino, "Ventajas e implementación de un sistema SIEM," 2018. [En línea]. Disponible: http://hdl.handle.net/10609/95287

[10] S. Gómez Fernández, "Implementación de un IDS de bajo coste para uso doméstico o en la pequeña empresa," 2019.

[11] M. H. Bhuyan, D. K. Bhattacharyya, y J. K. Kalita, *Network Traffic Anomaly Detection Techniques and Systems*, 2017.

[12] N. V. T. Hiep, T. V. Nikolaevich, D. M. Tuan, N. T. Lam, y N. A. Tuan, "Detecting botnet based on network traffic," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, num. 3, pp. 3010–3014, 2020.

[13] K. Alieyan, A. Almomani, A. Manasrah, y M. M. Kadhum, "A survey of botnet detection based on DNS," *Neural Computing and Applications*, vol. 28, num. 7, pp. 1541–1558, 2017.

[14] T. Contributors, "Botnet," 2019. [En línea]. Disponible: https://searchsecurity.techtarget.com/definition/botnet

[15] T. Notermans, "DNS series #1: DNS query types and how to use DNS in performance troubleshooting," 2017. [En línea]. Disponible: https://accedian.com/blog/dns-query-main-types/

[16] NS1, "DNS: Types of DNS Records, DNS Servers and DNS Query Types." [En línea]. Disponible: https://ns1.com/resources/dns-types-records-servers-and-queries

[17] A. Satoh, Y. Nakamura, D. Nobayashi, y T. Ikenaga, "Estimating the randomness of domain names for DGA bot callbacks," *IEEE Communications Letters*, vol. 22, num. 7, pp. 1378–1381, 2018.

[18] A. O. Almashhadani, M. Kaiiali, D. Carlin, y S. Sezer, "MaldomDetector: A system for detecting algorithmically generated domain names with machine learning," *Computers and Security*, vol. 93, 2020.

[19] P. Daniel, "DGArchive," 2020. [En línea]. Disponible: https://dgarchive.caad.fkie.fraunhofer.de/welcome/

[20] A. O. Almashhadani, M. Kaiiali, S. Sezer, y P. O'Kane, "A Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware," *IEEE Access*, vol. 7, pp. 47 053–47 067, 2019.

[21] J. Seo y S. Lee, "Abnormal Behavior Detection to Identify Infected Systems Using the APChain Algorithm and Behavioral Profiling," *Security and Communication Networks*, vol. 2018, 2018.

[22] D. Wook Kim y J. Zhang, "Deriving and measuring DNS-based fingerprints," *Journal of Information Security and Applications*, vol. 36, pp. 32–42, 2017. [En línea]. Disponible: https://doi.org/10.1016/j.jisa.2017.07.006

[23] T. Odete, L. Quiñones, y L. C. Rey, "Herramientas de monitorización y análisis del tráfico en redes de datos," *Revista Telemática*, vol. 11, num. 2, pp. 46–59, 2012.

[24] Scikit-learn, "2.7. Novelty and Outlier Detection," 2020. [En línea]. Disponible: https://scikit-learn.org/stable/modules/outlier{_}detection.html

[25] Suricata, "3. Installation — Suricata 6.0.0 documentation." [En línea]. Disponible: https://suricata.readthedocs.io/en/suricata-6.0.0/install.html

[26] M. (Howtoforge.com), "Suricata IDS with ELK and Web Frontend on Ubuntu 18.04 LTS." [En línea]. Disponible: https://www.howtoforge.com/tutorial/suricata-with-elk-and-web-front-ends-on-ubuntu-bionic-beaver-1804-lts/

[27] D. Berman, "Network Security Monitoring with Suricata, Logz.io and the ELK Stack," 2019. [En línea]. Disponible: https://logz.io/blog/network-security-monitoring/

[28] Elastic, "Installing search | search Reference [7.10]." [En línea]. Disponible: https://www.elastic.co/guide/en/elasticsearch/reference/7.10/install-elasticsearch.html

[29] ——, "Installing Logstash | Logstash Reference [7.10]." [En línea]. Disponible: https://www.elastic.co/guide/en/logstash/7.10/installing-logstash.html

[30] ——, "Install Kibana | Kibana Guide [7.10]." [En línea]. Disponible: https://www.elastic.co/guide/en/kibana/7.10/install.html

[31] ——, "Elastic/elasticsearch-py." [En línea]. Disponible: https://github.com/elastic/elasticsearch-py