

UNIVERSIDAD ICESI

Facultad de Ingeniería

Programa de Ingeniería de Sistemas

Departamento de Tecnologías de

Información y Telecomunicaciones

PROYECTO DE GRADO

Sistema *open source* para la detección de  
páginas web maliciosas

Integrantes

Melisa García Peña

Jose Luis Osorio Quintero

Dirigida por

Christian Camilo Urcuquí López, MSc.

Andrés Navarro, PhD

Santiago de Cali, abril de 2017

Tabla de contenido

[I. Resumen 3](#_Toc482557569)

[II. Lista de acrónimos 4](#_Toc482557570)

[III. Índice de tablas 4](#_Toc482557571)

[IV.](#_Toc482557572) [Motivación y antecedentes 5](#_Toc482557573)

[V. Problema identificado 6](#_Toc482557574)

[VI. Objetivos general y específicos 6](#_Toc482557575)

[VII. Marco teórico 7](#_Toc482557576)

[VIII. Estado del arte 9](#_Toc482557577)

[X. Desarrollo del proyecto 14](#_Toc482557578)

[IX. i. Investigación 14](#_Toc482557579)

[X. ii. Ingeniería de software 22](#_Toc482557580)

[XI. Experimento 23](#_Toc482557581)

[X. i. Tecnologías 24](#_Toc482557582)

[XI. ii. Resultados 25](#_Toc482557583)

[XI. iii. Análisis 34](#_Toc482557584)

[XII. Conclusiones y trabajos a futuro 35](#_Toc482557585)

[XIII. Referencias 36](#_Toc482557586)

## I. Resumen

El objetivo de este proyecto es obtener una herramienta capaz de analizar un tráfico web maligno y benigno de manera dinámica y automática, haciendo uso de características específicas, detectadas en un análisis estático. Para el entrenamiento de la herramienta, se utilizó un *honeypot* y un sniffer para la captura de información de las páginas web. Siguiente a esto, se hace un procesamiento de lo recolectado, identificando las características del tráfico que posteriormente ayudarán al entrenamiento de los algoritmos de clasificación de *Machine* *Learning.* Finalmente, se presenta un modelo base de arquitectura para desarrollar un sistema de detección que integra lo mencionado anteriormente para detectar páginas web maliciosas.

## II. Lista de acrónimos

URL *Uniform Resource Locator*

PHP *Hypertext* *Preprocessor*

OWASP *Open Web Application Security Project*

HTTP *Hypertext Transfer Protocol*

## III. Índice de tablas

Tabla 1. Ejemplo Matriz con datos 19

Tabla 2. Ejemplo matriz con variables dummy 20

*Tabla 3. Matriz de confusión a trabajar* 21

Tabla 4. Comparación de frecuencia de datos entre las características no 24 numéricas de la matriz de capa de aplicación

Tabla 5. Comparación entre el promedio de los datos numéricos de la 25  
matriz de capa de aplicación.

Tabla 6. Comparación entre el promedio de los datos de la matriz de capa 26  
de red

*Tabla 7. Resultados de los algoritmos por cada capa 31*

*Tabla 8. Resultado de los algoritmos para las tres características obtenidas 31*   
*por los métodos Subset e InfoGain del artículo [14]*

*Tabla 9. Resultados de los algoritmos por toda la matriz de datos 32*

## IV. Motivación y antecedentes

La tecnología ha incrementado casi de manera exponencial en los últimos años y esto ha generado que los ataques y por ende las vulnerabilidades en las páginas web, también aumenten con el tiempo [1,2]. Esto último se da, porque los desarrolladores de los sitios web, deben actualizarse casi al ritmo de la aparición de una nueva tecnología y, esto es imposible.

Según la empresa Whitehat Security [1], en el 2012 las vulnerabilidades más representativas fueron por filtro de información con 58% de probabilidad y *Cross-site Scripting* con 55% de probabilidad. Sin embargo, en el 2014 las aplicaciones y sitios web tuvieron mayor insuficiencia en la capa de transporte de red con 70% probabilidad, frente al filtro de información con 56% de probabilidad.

Por otro lado, dentro de las noticias internacionales, se menciona, por ejemplo, el ataque de denegación de servicio (DDoS) a la compañía de DNS Dyn, la cual les provee a algunas de las importantes empresas, como Twitter, Spotify o Reddit, renombres a sus direcciones IP’s. Este ataque, generó problemas al cargar las páginas o éstas se encontraban caídas [23].

Además, para las noticias nacionales, se menciona, por ejemplo, la página de la Registraduría Nacional de Colombia [24]. En la que unos días antes a la votación del plebiscito se presentó un ataque web que manifestó cambios en la visualización de la página (*defacement)* eincurrió a la inactividad de ésta*.* Afortunadamente no hubo robo de información según explicó Juan Carlos Galindo, el registrador Nacional del Estado Civil. Por lo tanto, páginas que contengan datos importantes como la de la Registraduría no pueden presentar este tipo de problemas y es por esto que se debe invertir en una forma de controlar y bloquear estos ataques.

No obstante, el problema de la seguridad es un caso muy particular, porque es insoluble, por ende, este proyecto va a contribuir a la protección de empresas y/o personas naturales que deseen verificar si una o un conjunto de URL’s, son benignas o malignas para así mismo acceder a ésta(s). Muchas de las contribuciones que ya existen, no son *open source* y esto genera que algunas compañías o personas independientes no puedan hacer uso de estos sistemas.

En conclusión, la detección de páginas maliciosas es un aspecto muy importante dentro de las entidades o personas naturales que desean estar protegidas frente a estas violaciones de seguridad. Además, les ayuda a mitigar problemas futuros relacionados con agentes externos que causan serios daños dentro de la lógica del sistema y es por este motivo, que este proyecto de grado se considera importante y tiene también como aspecto significativo, que será open source, esto genera un impacto positivo para cualquier persona que haga uso del internet y que no tienen los recursos suficientes para el pago de un sistema similar al que se va a llevar a cabo.

## V. Problema identificado

En relación con lo mencionado anteriormente, se identifica como problema principal que los usuarios suelen a acceder muchas veces a sitios web maliciosos sin tener conocimientos que van a ingresar a una página maligna o benigna, lo que puede conllevar a las falsificaciones de peticiones y el re-direccionamiento de información o el robo de información, que puede generar exposición de datos sensibles (datos personales, bancarios, organizacionales, entre otros) y también su re-direccionamiento.

Las principales causas que se identificaron para este problema son:

1. La existencia de herramientas con baja eficiencia en la detección de ataques.
2. Las pocas alertas para el administrador web, causado por bajas recomendaciones para combatirlas.
3. Las altas vulnerabilidades en páginas web causado por falta de un buen desarrollo, conocimiento de seguridad en páginas web y pocas validaciones de tipo de dato y longitud del dato en los campos de la página.

## VI. Objetivos general y específicos

Para el cumplimiento de este proyecto se plantea como objetivo general, el análisis, el diseño de la arquitectura y la validación de un sistema *open source* para la detección de páginas web maliciosas. Para este, se plantean tres objetivos específicos, los cuales van a certificar que el general se va a cumplir, estos son:

1. Identificar y caracterizar tanto las páginas maliciosas como las benignas.
2. Desarrollar y probar los algoritmos J48, SVM, Regresión Logística y *Naive Bayes* para la clasificación de páginas web
3. Analizar y diseñar un sistema de detección que integre el módulo de clasificación

## VII. Marco teórico

Mientras que el uso y la conectividad de las computadoras han incrementado, su seguridad se ha convertido en una preocupación importante para las personas que trabajan con la interacción de éstas, porque los ataques son más usuales, más difíciles de controlar y detectar. Por este motivo, las violaciones de la seguridad informática se consideran comunes en la actualidad [3]. Dentro de las vulnerabilidades existentes, OWASP, fundación internacional sin ánimo de lucro que trabaja de manera colaborativa y abierta, presenta en su top 10 del 2013 (a mediados del 2017 lanzaran el top 10 2016) las más frecuentes y conocidas por los atacantes, en las cuales se encuentra [4]: **inyección**, es la inserción de datos no confiables, enviados a través de una consulta o comando que es entendido por un intérprete; **pérdida de autenticación y gestiones de sesiones**, es la implementación incorrecta de autenticación y gestión de sesiones; ***Cross-Site Scripting* (XSS)**, es el envío de información no confiable a través de un navegador web; **referencia directa e insegura a objetos**, es la exposición de objetos (representación de la información del sistema) de implementación interna de manera pública; **configuración incorrecta de seguridad**, es la ausencia o baja definición de marcos de trabajo, servidores de aplicación, servidores web, bases de datos o plataformas, que hacen vulnerable a un sistema; **Exposición de datos sensibles**, es la falta de métodos de protección de datos, tales como encriptación y precauciones especiales de intercambio de datos; **ausencia de control de acceso a funciones,**  es la falta de verificación de autenticación de funciones que se acceden a un servidor; ***Cross-Site Request Forgery* (CSRF)**, es la falsificación de peticiones HTTP; **utilización de componentes con vulnerabilidades conocidas**, es la utilización de *frameworks* (infraestructuras) o módulos de software con vulnerabilidades de seguridad; y por último, **Redirecciones y reenvíos no validados,** es el envío de información o reenvío de usuarios a sitios no autorizados.

Por otra parte, las redes, conjunto de nodos conectados entre sí que proporcionan datos que son transportados de un lugar a otro con el fin de proporcionar comunicación entre una o más dispositivos, acceso e información al medio (personas, entidades, sociedades, servidores) [5], ha sido un elemento muy importante y clave para la conectividad entre las tecnologías y los humanos.

En la actualidad, la inteligencia artificial es una técnica muy utilizada para el desarrollo de la seguridad en páginas web. Esta presenta varias definiciones y, en este proyecto nos centramos en dos, la primera es, “rama de la informática dedicada a la creación artificial del conocimiento”, según el libro de Fundamentos de la inteligencia artificial de L.A. Munáriz [6]; y la segunda, la inteligencia artificial es la disciplina que estudia al humano, analizando su comportamiento y sus procesos cognoscitivos, debido a que se enfocan en el estudio de los procesos internos que conducen al aprendizaje [7]. Sin embargo, es importante mencionar que la inteligencia artificial no se basa solamente en el humano, sino también en otros seres vivos, como plantas y animales.

Esta disciplina, incluye una técnica muy usada llamada *Machine Learning*, la cual se define como, “la programación de computadores para optimizar un criterio de rendimiento, utilizando datos de ejemplo o experiencia del pasado”, según Alpaydin [8]. Esta técnica, incluye varios tipos de aprendizajes automáticos y en este proyecto, definimos dos de éstos. El primero es el **supervisado,** este se realiza al tener un conjunto de datos con el cual se hace el entrenamiento de la máquina, para que ésta aprenda y pueda predecir el resultado de otros datos. Este lenguaje cuenta con varios algoritmos, dentro de este proyecto se van a definir cuatro de éstos: J48 *Decision Tree*; Nave Bayes; *Support* *Vector* *Machine*; y, regresión logística[8, 9]. Y el segundo es el **no supervisado,** es el proceso de creación de estructuras o patrones que se presentan en una información dada, cuya principal característica es el desconocimiento del conjunto de datos [10].

Ahora bien, es importante mencionar que, dentro de las técnicas de análisis de seguridad, se encuentran el **análisis** **estático**, el cual analiza el código fuente minuciosamente para concluir la existencia de vulnerabilidades en éste. Teniendo en cuenta que no todos son el original, se debe aplicar ingeniería inversa y/o depuradores para desensamblar el código. Este método no es muy eficiente y por el contrario implica mucho trabajo y tiempo para el personal que haga uso de este análisis [11, 12, 13]; y el **análisis** **dinámico**, El análisis dinámico consiste en una detección automatizada de debilidades en el código. La idea de este análisis, es que todas las evaluaciones se ejecuten en tiempo real mientras la aplicación se encuentra en funcionamiento; es una técnica que permite la detección de las debilidades a través de un análisis de la interactividad del atacante, un método que brinda más precisión y al ser automático, es mucho más eficiente [12, 13]. Para el desarrollo de este proyecto, se va hacer uso de ambas y más adelante se va a mostrar para qué, por qué y cómo.

## VIII. Estado del arte

***Cross- Layer Detection of Malicious Websites* [14]**

La investigación principal del artículo, radica en la comparación de la herramienta que ellos desarrollaron para la detección de URL's maliciosas, basada en operaciones AND, OR, XOR y agregación de datos, contra cuatro algoritmos clasificadores de *Machine Learning,* específicamente regresión logística, *Naive Bayes, Support Vector Machine (SVM) y C.45 árbol de decisión llamado* J48.

Teniendo en cuenta lo anterior, para la extracción de características en las URL's, hicieron uso de un Crawler y TCPDUMP, con esto determinaron las URL's vivas y extrajeron el tráfico de la mismas. Con esta información, se centraron entonces en obtener las características de la capa aplicación (información de la URL) y de la capa de red (datos del tráfico web), basadas en el tráfico. Por cada capa obtuvieron 19 características. Así:

Por parte de la capa de aplicación se obtuvieron:

* Longitud de la URL. Se identificó que el promedio de las malignas es mayor a las benignas.
* Cantidad de caracteres raros. Se identificó que el promedio de las malignas es mayor a las benignas.
* Información de la cabecera de HTTP.
  + Charset
  + Servidor, se identificó que los servidores más utilizados por las páginas web maliciosas son Apache, Microsoft IIS y nginx.
  + Cache control, se identificó que en promedio el número de sitios benignos que usan no-cache, private, public y max-age es mayor que los sitios malignos.
  + *Content length*
* Información de la Whois
  + Fecha de registro del *webserver*
  + Fecha de actualización del *webserver*
  + País y el estado de registro del *webserver,* se identificó que los tres países más representativos fueron US, NL y AU y para las URL's benignas fue mayor el número de registros en éstas que las malignas
  + Dominio, identificaron que es mayor el porcentaje de redirección de dominios en los sitios web maliciosos.
* Contenido de la información
* Número de redirecciones, es el número total de redirecciones embebidas dentro de una URL.
* Número de URL’s externas embebidas, cuenta el número de URL’s que son embebidas dentro de la URL y usa recursos externos.
* Longitud de la cabecera http de la página.
* Longitud de variables, es el tamaño de las variables de los scripts que contiene la página.
* Numero de Iframe encontrados en la página.
* Longitud de los Iframes encontrados en la página.
* Número de funciones sospechosas de scripts que contiene la página.

Y por parte de la capa de red se obtuvieron:

* Características remotas basadas en los atributos del servidor.
  + Intercambio de conversaciones TCP, número total de paquetes TCP enviados a un servidor remoto por el crawler.
  + Distintos puertos TCP remotos, número total de los distintos TCP puertos que el webserver usa durante la conversación con el crawler.
  + IPs remotos, número de distintas direcciones IP remotas que se conectan por el crawler.
* Comunicación de *crawler- server*
  + Bytes app, número de bytes de la aplicación de datos enviados por el crawler al webserver remoto.
  + Paquetes UDP, número de paquetes UDP generados cuando el crawler visita una URL.
  + Paquetes urgentes TCP, número de paquetes urgentes con indicador URG
  + *Source app packets,* número de paquetes enviados por el crawler al servidor remoto. En promedio, es más alto el de sitios web maliciosos, en comparación con los benignos.
  + Paquetes de app remoto, número de paquetes enviados por el webserver remoto al crawler. En promedio, es más alto para sitios web malignos en comparación con los benignos.
  + Source app bytes, cantidad de bytes que se dan en la comunicación del crawler al sitio remoto.
  + App bytes remotos, cantidad de bytes que se dan en la comunicación de sitio remoto al crawler.
  + Duracion, duración de la respuesta de conexión de la página web. Se encontró que esta es la característica más fuerte de todo el proceso.
  + Promedio de ratio de paquetes locales, es la cantidad de paquetes locales sobre la duración.
  + Promedio de ratio de paquetes remotos, es la cantidad de paquetes remotos sobre la duración.
  + Paquetes de aplicación, numero de paquetes IP enviados en la comunicación del crawler al sitio web.
* *Flujo de Crawler- DNS*
* Valores agregados como *Iat\_flow; Flow number; y flow duration.*

La validación de ambas herramientas, se realizó en un entorno controlado en un lapso de 37 días para la obtención de las características. El autor menciona que durante este proceso se obtuvieron aproximadamente 124 características, de las cuales se analizaron midiendo su correlación y precisión por medio de 3 algoritmos encontrados en *Weka Toolbox.* Estos algoritmos son: ***Principal Components Analysis (PCA),***es un algoritmo de aprendizaje no supervisado que no muestra fuertes indicios para la selección de las características de sitios maliciosos; **CfsSubsetEval,** algoritmo de *Weka Toolbox* que sirve para encontrar una baja correlación y una alta precisión en las características; **InfoGainAttributeEval,** algoritmo de Weka Toolbox que sirve para determinar las características más importantes dependiendo de su ratio de adquisición y la información que proveen.

Con base a los tres algoritmos mencionados anteriormente, el autor hace un análisis comparativo usando los 4 clasificadores y los métodos de agregación de datos, AND, OR y XOR. Demostrando que XOR con los algoritmos subset e Infogain para el clasificador J48 resultan ser los más destacados del trabajo realizado. Dentro de este proceso encuentra que las características tales como la longitud de la URL, el nombre del servidor y la duración resultan ser las más fuertes y la base de su herramienta (Cross-Layer Detection).

***Detecting Malicious Inputs of Web Application Parameters using Character Class Sequences [15]***

El método que propusieron consta de tres pasos: transformación, Filtrado y determinación de tipo de perfil, estos son encapsulados en uno grande llamado entrenamiento y por último expusieron la detección.

Para realizar el entrenamiento, los datos los obtuvieron de un *dataset* real, es decir, que las URL's no se encontraban en un entorno controlado. A continuación, se muestran los tres pasos de este:

1. En el primer paso, transformación, definieron varias clases de caracteres básicos como alfabeto, número y símbolos, es decir, caracteres raros. Este tipo de método de abstracción de caracteres es muy usado para la detección y agrupación de URL's maliciosas. Para generar una secuencia de clases a partir de un valor, hicieron uso de algoritmo de coincidencia de prefijos incrementales.
2. Después, calcularon la frecuencia de cada clase de carácter que arrojó el paso 1, utilizando un alfa entre 0 y 1.
3. Finalmente, encontraron que se pueden producir falsas alarmas de URL's malignas cuando aplican una estricta restricción estructural a estos parámetros como mensajes de chat de una aplicación web de red social. Por este motivo, introducen un tipo de perfil llamado “conjunto de clases de caracteres”, el cual indica el conjunto de clases de caracteres si un orden específico.

Para realizar la detección, los autores de este artículo extraen parámetros de solicitudes HTTP, los cuales deben ser revisados. El valor de cada parámetro se transforma de la misma manera que en el paso 1 y realizan el cálculo de la similitud de un valor de parámetro, para cada uno de los ingresados, y dependiendo del perfil que tenga, se clasifica como maligno o benigno.

Finalmente, realizaron experimentos y evaluaciones de la siguiente manera:

1. Utilizan un *dataset* y los dividen en dos partes, una para entrenar y la otra parte para hacer la detección.
2. Para la parte con la que se hace el entrenamiento, se eliminan las peticiones cuyas URL's lanzan respuestas del tipo 404 o 500, debido a que estas peticiones son posiblemente las exploraciones para encontrar servidores web vulnerables.
3. En el conjunto de datos de detección, se da una etiqueta que indica si la solicitud es o no un ataque. Esta se basa en los resultados de detección de múltiples aplicaciones comerciales de la aplicación web (WAF) y análisis manuales que ellos realizaron.
4. Finalmente, se realiza la detección de ataques con los datos de prueba haciendo uso de su propuesta y, como resultado encontraron aproximadamente cuatro tipos de ataques: *SQL injection,* inyección de comandos, la inclusión de archivos remoto (RFI) y XSS. Para este último punto, hicieron uso tasa de detección y tasa de falsos positivos como indicadores de evaluación de precisión de detección.

Es importante resaltar, que ellos recomiendan no usar parámetros con valores exactos para hacer la detección de las entradas maliciosas, pues esto puede causar falsas alarmas en el análisis de la URL.

***Towards Automatic and Lightweight Detection and Classification of Malicious Web Contents [16]***

Este artículo presenta un sistema que identifica cualquier contenido malicioso o benigno en un sitio web utilizando análisis estático y dinámico. Además, utilizaron los algoritmos de *Machine Learning* para la detección tanto de páginas maliciosas y benignas; si el algoritmo identifica una como maliciosa entonces el sistema cuenta con la capacidad de predecir su categoría. Las categorias evaluadas se encuentran: *injection*, *exploit type, Exploit kit type, Redirection type, Defacement, Obfuscation type, Malicious executable or archive type* y *Server side backdoor type.*

Los autores utilizan un sistema alfa, el cual hace el papel de análisis dinámico, para analizar el contenido web ingresado. Con éste analizan un sistema llamado archivos transferidos, el cual ayuda a determinar si una página es maliciosa o no.

Para la clasificación de características, las cuales serán evaluadas después, ellos acuden a las siguientes:

* Meta- data: incluye las características HTTP
* URI: son las características derivadas de la URI que se asocian con los archivos transferidos, entre estas se encuentran: el *hostname,* el paquete, la consulta, el tamaño de la URL y la cantidad de características extrañas de la misma.
* Archivos de transferencia: son las características como la dirección IP, numero de redirecciones, entre otros.
* Nombre del dominio: incluye todas las características que se derivan del nombre del dominio como el tiempo de la información, fecha de creación y, fecha actualización o expiración.
* IP: estas se derivan de la dirección IP, incluye el país, la ciudad, la región la organización, entre otros.

Ellos describen su diseño de la siguiente manera, el sistema recibe datos generados por el sistema alfa, el sistema determina por cada archivo transferido una etiqueta la cual representa si es benigno o maligno el mismo, después éste trabaja en el entrenamiento y lo modos de operaciones.

Para evaluar las características anteriormente mencionadas, realizaron una matriz para entrenamiento y otro para prueba. Ambas matrices, las normalizaron de la siguiente manera, primero los datos numéricos se normalizaron con valores entre 0 y 1, mientras que las características categóricas aplican binarización.

Los algoritmos que se usaron para la clasificación de las URL’s fueron SVM basado en *Stochastic Gradient Descent* y, utilizaron un clasificador binario para cada tipo de vulnerabilidad. Es importante mencionar que ellos utilizaron 20000 páginas web, dentro de las cuales 10000 son malignas y el resto, benignas. Para la implantación de los algoritmos, hicieron uso de *10- fold cross validation*, para que los *datasets* sean particionados aleatoriamente, usando una parte para el entrenamiento y la otra para las pruebas. Como conclusiones, después de la implementación del algoritmo ellos llegaron a que las páginas web maliciosas tenían una concurrencia del 93% de vulnerabilidades *injection* y *derver-side backdoor.*

IX. Contribución del proyecto de grado

El proyecto planteado, tiene las siguientes contribuciones en cuanto al objetivo del mismo:

* Futuros trabajos que estén orientados a detección de ataques o páginas maliciosas. Este proyecto es la base para el desarrollo de trabajos que tengan relación con URL’s y/o tráfico malicioso, por tal motivo, se deja toda la documentación, bases de datos y *scripts* de los algoritmos, **disponible al público.**
* El análisis exploratorio de las matrices de la capa de aplicación y de red, con respecto a las URL’s suministradas. Esto es esencial para tener una idea del patrón que siguen las páginas web benignas y malignas. Además, todo queda graficado, para mejorar la comprensión de los resultados.

## X. Desarrollo del proyecto

El proyecto incluye dos metodologías, la primera enfocada en la investigación y la segunda a la ingeniería de software del sistema de detección.

### IX. i. Investigación

Esta sección cuenta con toda la parte de investigación realizada antes del planteamiento del software, es decir, esta es la base principal para el funcionamiento de la herramienta de software

La investigación comenzó con un estudio del estado del arte y del marco teórico, en el cual se analizaron los trabajos que incluían los algoritmos de *Machine Learning* para identificación de páginas web maliciosas. Luego, se identificaron y adquirieron los *datasets* tanto de URL’s benignas como maliciosas. Posteriormente, se desarrolló y evaluó una herramienta para el pre procesamiento de la información, en donde se verifico si cada página web se encontraba en funcionamiento y dependiendo del resultado se obtuvieron los datos de la capa de aplicación y red. A continuación, se realizó el procesamiento de la información en el cual se obtuvieron un total de 400 características. Finalmente, se entrenaron y testearon los algoritmos de clasificación: *Super Vector Manchine* (SVM), Regresión Logistica*, Naive Bayes* y J48 (algoritmo C4.5 de árbol de decisión).

#### X. i. a. Adquisición de datasets

Aquí se describen los *datasets* que se evaluaron para el desarrollo del proyecto y su obtención. Para la adquisición de éstos, se decidió que es primordial obtener los referenciados en artículos. Sin embargo, durante la búsqueda, se encontró que no es fácil acceder a estos, pues no suelen dejarlos *open* *source*, cabe aclarar que en muchos artículos dan la página de la cual extrajeron las URL's malignas y benignas, y no el *dataset* realizado o el utilizado en los estudios.

Para la adquisición de los datasets, entonces, se decidió extraer de páginas que brinden blacklists, esto pensando en la actualización de los mismos y que probablemente no han sido estudiados. Continuando con lo descrito anteriormente, se extrajeron los datos de las siguientes páginas web machinelearning.inginf.units.it/data-and-tools/hidden-fraudulent-urls-dataset, malwaredomainlist.com y zeuztacker.abuse.ch, para obtener un total de aproximadamente 185.181 URL’s. Por otro lado, la obtención de las URL's benignas se hizo desde el repositorio https://github.com/faizann24/Using-machine-learning-to-detect-malicious-URLs.git, en donde se obtuvieron un total de 345.000 URL's.

Para una mejor identificación de las URL’s, se organizaron por *datasets* así:

* *Dataset* 0, contiene 2.569 URL’s malignas.
* *Dataset* 1, contiene 11 URL’s malignas.
* *Dataset* 2, contiene 17 URL’s malignas.
* *Dataset* 3, contiene 75 URL’s malignas.
* *Dataset* 4, contiene 185.181 URL’s malignas.
* *Dataset* benigno: contiene 345.000 URL’s benignas.

#### X. i. b. Pre procesamiento de datos

En el pre procesamiento de la información adquirida se tuvo en cuenta lo siguiente:

**Verificación de información:** en esta sección se corroboró el estado de las páginas que almacenaban cada *dataset*. El fin de este proceso es determinar si la URL está activa o inactiva, esto se llevó a cabo mediante un script de Python y por medio de la librería urllib2 que permitía la conexión a la página web, con este proceso pudimos hacer una verificación y limpieza de cada dataset. Una de las cosas que se recalca en este proyecto es sobre el cambio de la actividad de estas páginas, aunque bien, se ha demostrado en el ejercicio de esta obtención que las paginas malignas tienen una vida útil más baja que las páginas de comportamiento normal, esto hace que las paginas sean muy variables en cuanto a su estado mostrando cambios de un estado activo a inactivo en tan solo días u horas.

**Modificación de la información:** en esta sección se espera tener un *dataset* con URL's activas, para poder hacer la separación y clasificación de cada una de ellas poniéndoles un identificador para diferenciarlas del resto. Este identificador es único por página y su principal uso es la ubicación de cada URL. Este proceso se llevó a cabo con un script hecho en Python.

**Obtención de la información:** en esta sección se espera tener un *dataset* que posea cada una de las URL's clasificadas con un identificador único. Para llevar a cabo este proceso se tiene en cuenta el uso de dos herramientas que permiten tener los datos necesarios de cada página web.

* Honeypot: es un sistema que emula unos servicios de red, una de sus funciones es el estudio de los posibles ataques que se hagan a un sistema o servicio de red que este emula. Los honeypots están clasificados de acuerdo a su permisividad de interacción con el atacante, las categorías son: baja, media y alta interacción; para propósitos de este proyecto se utilizó un honeypot de baja interacción llamado Thug, esta herramienta permite simular un entorno de red y distintos navegadores de red con vulnerabilidades para estudio de múltiples ataques provenientes de sitios web que esta visita.
* Sniffer: es un sistema que permite la captura de tráfico de red de una determinada interfaz de red, en el mercado existen muchos sniffer famosos que permiten la captura de paquetes de red. Entre los cuales encontramos Wireshark y TCPDUMP. Para propósitos de este proyecto se usaron ambas herramientas, aunque en el caso de Wireshark se hizo uso su versión en líneas de comando denominada TShark.

Una vez las herramientas han sido previamente configuradas y el *dataset* con las URL's creado. Se procede a examinar cada URL con el honeypot y el sniffer para la captura de los datos, cada una de ellas maneja un tráfico de red y este archivo es guardado en una carpeta para su posterior análisis.

Por parte de los *datasets* malignos se obtuvieron los siguientes resultados al realizar el pre procesamiento mencionado:

* Para el *dataset* 0, se obtuvieron 599 URL’s vivas.
* Para el dataset 1, se obtuvieron 3 URL’s vivas.
* Para el dataset 2, se obtuvieron 17 URL’s vivas.
* Para el *dataset* 3, se obtuvieron 75 URL’s vivas.
* Para el *dataset* 4, se obtuvieron 35.279 URL’s vivas.

Para un total de 35.973 URL’s vivas.

Mientras que para los *datasets* benignos, cabe aclarar que al haber más cantidad al principio de URL’s, se procesaron primero las malignas, puesto que era más evidente que podían muchas de estar sin servicio, por tal motivo después de saber cuántas en total estaban vivas, aleatoriamente se eligió un número similar para verificar las URL’s benignas vivas y obtener el tráfico de las mismas. Finalmente, se obtuvieron 27.912 URL’s vivas.

#### X. i. c. Procesamiento

A continuación, se presentan las características con la que se van a trabajar durante la ejecución del proyecto. Estos dos análisis se hacen basados en el artículo de *Cross*-*Layer* *Detection* [14] los cuales son:

**Capa de aplicación**: en esta capa se extrae directamente de la URL y el *Thug*, la información necesaria para obtener las características consideradas relevantes. Estas son:

**(CA1) Content\_length**: representa el tamaño total de la URL.

**(CA2) Number\_Special\_Characteres:** Representa la cantidad de caracteres extraños que contiene la URL, como lo son “/”, “%”, “#”, “&”, “. “, “=”, entre otros.

Información de HTTP

**(CA3) *HTTPHeader\_charset***: conjunto de caracteres con los que se identifica el sitio web, como lo son ANSI, ISO-8859-1, UTF8 entre otros

**(CA4) *HTTPHeader\_*server**: esta característica es el servidor en el que fue montada la URL.

**(CA5) *HTTPHeader\_content****\_****length****: representa el tamaño del contenido de la cabecera HTTP.*

Propiedades *Whois:*

**(CA6) *Whois\_regDate****: indica cuando el servidor del sitio web fue registrado*

**(CA7) *Whois\_Updated\_date****: indica cuando el servidor fue actualizado*

**(CA8) *Whois\_country***: indica el país donde se encuentra el servidor del sitio web

**(CA9) *Whois\_statePro***: indica el estado del sitio web

**(CA10) *Whois\_D*omain**: indica el dominio del sitio web

Para la obtención de la información HTTP y las propiedades Whois, se examina directamente la URL con librerías que Python proporciona (requests y whois). Por otro lado, el inconveniente que se tiene al trabajar con URL’s tan cambiantes es que al momento de revisar si sigue activa puede ser que efectivamente lo está, pero al minuto ya no lo está, esto hace cambiar el retorno de la consulta hacia estas. Adicionalmente, el tiempo que tarda la maquina en ir hasta la URL y extraer la información que se necesita puede ser mucho y por tal motivo, no arroja los datos esperados, lanzando una excepción "Error Time", esto último también se puede ver influenciado por el primer caso. Finalmente, debido a los casos mencionados, en la matriz de aplicación se encuentran valores extraídos con las siguientes características:

NA, cuando ocurrió un error y no se puedo leer el dato, se presenta por lo general, por límite de tiempo al consultar la URL.

None, cuando la URL no presenta valor alguno en esta característica.

**Capa de red**: en esta se tiene en cuenta los archivos de tráfico de datos que se generaron con la herramienta TCPDUMP y por medio de la herramienta llamada PyShark que es un *wrapper* de TShark que permite la manipulación de los datos de un tráfico de red en formato pcap [17]. Con esta herramienta se extrajeron los datos:

**(CR1) tcp\_conversation\_exchange:** cuenta la cantidad de paquetes que hay entre el honeypot y el sitio web para el protocolo TCP.

**(CR2) dist\_remote\_tcp\_port:** número total de puertos distintos a los puertos TCP.

**(CR3) remote\_ips:** número distinto de direcciones IP conectadas al honeypot.

**(CR4) pkt\_without\_dns:** almacena en un arreglo todos los paquetes que no son DNS.

**(CR5) app\_bytes:** número de bytes de la capa de aplicación que envía el honeypot hacia el sitio web, no se incluyen los datos de los servidores DNS.

**(CR6) udp\_packets:** número de paquetes UDP, no se incluyen los datos de los DNS.

**(CR7) tcp\_urg\_packet:** número de paquetes TCP con la bandera de URG.

**(CR8) source\_app\_packets:** número de paquetes enviados por el honeypot hacia el servidor remoto.

**(CR9) remote\_app\_packets:** volumen en bytes de la comunicación del servidor web al honeypot.

**(CR10) *duration:*** tiempo de duración de la página web.

**(CR11) avg\_local\_pkt\_rate:** promedio de paquetes IP por segundo (paquetes enviados sobre la duración).

**(CR12) avg\_remote\_pkt\_rate:** promedio de paquetes IP por segundo (paquetes enviados sobre la duración).

**(CR13) app\_packets:** número de paquetes IP incluidos los del servidor DNS.

**(CR14) dns\_query\_times**: lista de capas de DNS *queries*.

Después de obtener todas las características, tanto de la parte de aplicación como de tráfico, comienza la creación de una matriz con valores entre 0 y 1 (incluyendo estos dos números) para posteriormente realizar un análisis con algunos algoritmos de *Machine Learning*. Para la definición de las características y las métricas utilizadas para que los datos de los atributos mencionados se representen entre los dígitos entre 0 y 1, se toma en cuenta el artículo [16], en el cual indica que los atributos que tienen valores numéricos se deben normalizar, esto con el fin de que sean comparables. Por otra parte, para los valores que son caracteres o alfabetos, se utilizaron variables indicadoraso más conocidas como variables dummy. Esto último se explica con el siguiente ejemplo, suponga que tiene una matriz de cuatro columnas, incluyendo el identificador, una de ellas tiene valores alfabéticos y las otras dos numéricos, así:

|  |  |  |  |
| --- | --- | --- | --- |
| id | A | B | C |
| D0 | 1 | 45 | Europa |
| D1 | 23 | 67 | América |
| D2 | 45 | 89 | Asia |
| D3 | 5 | 9 | Europa |

Tabla 1. Ejemplo Matriz con datos

En la Tabla 1, se presenta la matriz con datos normales, incluye numéricos y caracteres alfabéticos. Ésta se necesita volver binaria y, por tal motivo, se normalizan los datos numéricos, A y C, mientras que los valores de B se vuelven columnas y si el identificador estaba asociado al valor determinado, es 1 de lo contrario es 0, así:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | A | B | C.Europa | C.América | C.Asia |
| D0 | 0 | 0.450 | 1 | 0 | 0 |
| D1 | 0.4 | 0.725 | 0 | 1 | 0 |
| D2 | 1 | 1 | 0 | 0 | 1 |
| D3 | 0.09 | 0 | 1 | 0 | 0 |

Tabla 2. Ejemplo matriz con variables dummy

De esta forma, como se muestra en la Tabla 2, garantizamos la binarización de la matriz de características que va a pasar por los algoritmos clasificadores de *Machine Learning.* Además, se tuvo en cuenta la correlación entre los datos de la siguiente forma:

Primero, por cada uno de los *datasets* se haya la correlación propia para analizar como es el comportamiento individual de los datos.

Después, al unir los *datasets* por clasificación (benigno y maligno) se revisa también la correlación, pues se desea saber la dependencia entre las características de aplicación y red.

Finalmente, se haya la correlación entre la matriz final, que contiene todos los *datasets* unidos, esto con el fin de analizar la dependencia entre todos los datos.

Lo anterior, hace referencia a un estudio experimental realizado con los datos propios extraídos de los *datasets* generados. Con estos, se procede a concluir características propias. Sin embargo, también se va a probar con los datos que el artículo [14] nos proporciona, esto con el fin de hacer uso de los resultados que en éste obtuvieron. Para esta parte se eligen los datos de *Subset* e *InfoGain,* arrojaron los mejores resultados, según los autores muestran en el mismo. Éste se compone por características como Longitud de la URL, duración y nombre del servidor.

#### X. i. d. Algoritmos de machine learning y las medidas de evaluación

Los algoritmos que se consideran en este proyecto, para evaluar la matriz binaria mencionada anteriormente, son los siguientes:

* J48: Este algoritmo genera reglas para la predicción de variables objetivas, con la ayuda de un algoritmo de árbol de clasificación, la distribución crítica de los datos es fácilmente detectable [20].
* *Naive Bayes*: Este modelo es muy útil para conjuntos de datos grandes, es simple, rápido y tiende a ser muy bueno como método clasificador. Además, tiene un buen rendimiento con variables categóricas y para las variables numéricas hace uso de una distribución normal [19, 21].
* *Support* *Vector* *Machine(SVM):* Este algoritmo produce muy buenos resultados cuando entran datos nuevos al mismo, siempre y cuando esté bien parametrizado; el proceso de entrenamiento no depende del número de atributos que tenga la matriz [25]; entre muchos otros beneficios que ayudaron a la elección de este.
* Regresión logística: Este modelo trata de predecir una característica cualitativa a partir de otras variables ingresadas.

La manera de evaluar la precisión de los mismos, es mediante una matriz de confusión, en la cual se presentan los falsos positivos y los falsos negativos.

|  |  |  |
| --- | --- | --- |
| **Clase estimada | Clase real** | **Clase de referencia** | **Clase de no referencia** |
| **Clase de referencia** | TP | FP |
| **Clase de no referencia** | FN | TN |

*Tabla 3. Matriz de confusión a trabajar*

### X. ii. Ingeniería de software

En esta sección, dejamos y explicamos la base para la implementación del sistema de detección.

#### X. ii. a. Metodología

La metodología propuesta para el desarrollo del sistema de detección, son los ciclos de vida incremental e iterativo. El primero, que es el más representativo en este proyecto, se considera por las siguientes ventajas [22]:

1. Ayuda a disminuir los posibles riesgos, pues el desarrollo va por partes y cada vez que se pasa de una fase a otra, se puede determinar si estas han sido bien planteadas. En el caso que no lo hayan sido, se contempla el segundo modelo mencionado anteriormente, modelo iterativo, el cual
2. A medida que se va desarrollando por partes el proyecto, se van integrando hasta que quede completo.

##### X. ii. *a*. 1. Análisis

Esta parte, se encuentra en el documento *software requirements specification* (SRS), éste presenta todos los requerimientos del sistema, el diagrama de casos de uso (como el usuario interactúa con el sistema) y el alcance del mismo. Ver anexo “Documento SRS.doc”.

##### X. ii. *a.* 2. Diseño

El diseño arquitectónico, se representa de manera monolítica. En un nodo, “GeneracionURLs”, Se realiza todos los procesamientos de consulta directa a la URL, para determinar si está viva y generar todo el tráfico de la misma. Esto es necesario para la construcción de las matrices de la capa de aplicación y de red. Sin embargo, como la capa de aplicación, también hace consulta directa a la URL, se hace también en este nodo la matriz de la misma. Lo anterior se maneja aparte por seguridad, pues al tratar con URL’s malignas, la consulta directa, trae virus o redirecciones a páginas con virus.

Por otro lado, El segundo nodo realiza todo el procesamiento de las URL’s vivas, generando las matrices de características de la capa de aplicación benigna y la capa de aplicación maligna y benigna. Además, genera la matriz binaria, dummificando y normalizando los datos, para posteriormente entrenar a los algoritmos de *Machine Learning*, generar el resultado de los mismos y, graficar y crear el informe del análisis.

Ver anexo “diagrama arquitectura.jpg”

## XI. Experimento

En esta sección se abarcan las tecnologías utilizadas durante la investigación y los resultados experimentales.

Esta sección, incluye la parte los resultados que se obtuvieron mediante el análisis con los algoritmos de *Machine Learning*, sabiendo que esto depende directamente de las tecnologías usadas, también se mencionan.

### X. i. Tecnologías

* Python, este lenguaje presenta características acordes con una de las contribuciones de este proyecto, ser la base para futuros proyectos, por ejemplo:
  + *Open source,* esto hace quefuncione en diversas plataformas como Linux, Windows, Macintosh, Solaris, OS/2, entre otras.
  + Librerías, este lenguaje contiene múltiples librerías útiles, tipos de datos y funciones en el lenguaje propio, lo que facilita la programación, pues no se implementa todo desde cero, sino que se hace uso de éstas.
* R, este lenguaje presenta permite la optimización de uso estadístico en los datos obtenidos, facilita la limpieza y el manejo de un conjunto de datos, además proporciona herramientas y librerías que permiten el uso de *machine learning*.
* R Studio, esta herramienta permite un mejor manejo de los recursos de R para la visualización y usabilidad del programador que desee desarrollar y solucionar problemas a nivel estadístico.
* Visual Studio Code, este editor de texto permite el manejo de muchos archivos y la facilidad de realizar scripts como también su ejecución.
* Thug es un honeypot de baja interacción que permitió ejecutar las URL’s en un ambiente controlado.

Tres computadores con las siguientes características:

* Toshiba con un procesador Intel Core i5 de 1.8GHz y RAM de 4G
* Asus con un procesador Intel Core i7 de 2.4GHz y RAM de 8G
* Acer con un procesador Intel Celeron de 1.6GHz y RAM de 2G

### XI. ii. Resultados

A continuación, se presentan los resultados de los análisis exploratorios de los *datasets* benigno y maligno de la capa de aplicación y de red.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Capa de aplicación** | | | | |
| **Características evaluadas** | **Característica frecuente de los datos benignos** | **Frecuencia de los datos Benignos** | **Característica frecuente de los datos malignos** | **Frecuencia de los datos Malignos** |
| **CA3** | UTF – 8  ISO - 8859 -1  US – ASCII | 384  193  54 | UTF – 8  ISO - 8859 -1  US – ASCII | 1547  417  33 |
| **CA4** | APACHE  NGINX  SIN REGISTRO  MICROSOFT-HTTPAPI  MICROSOFT – IIS CLOUDFLARE–NGINX | 204  102  68  30  40  6 | APACHE  NGINX  MICROSOFT – IIS  SIN REGISTRO  MICROSOFT–HTTPAPI  CLOUDFLARE–NGINX | 1192  425  220  182  32  30 |
| **CA8** | US  SIN REGISTRO  CN  CA  FR  CZ | 383  120  4  2  2  1 | SIN REGISTRO  US  CN  CA  CZ  FR | 1150  272  181  27  8  4 |
| **CA9** | CA  FL  WA  PA  AZ  AL | 158  27  24  15  13  1 | AZ  WA  PA  FL  AL  CA | 94  89  66  32  31  16 |

Tabla 4. Comparación de frecuencia de datos entre las características no numéricas de la matriz de capa de aplicación

|  |  |  |
| --- | --- | --- |
| **CAPA DE APLICACIÓN** | | |
| **Características evaluadas** | **Promedio de los datos benignos** | **Promedio de los datos malignos** |
| **CA1** | 56,3155 | 85,455 |
| **CA2** | 10,8163 | 17,203 |

Tabla 5. Comparación entre el promedio de los datos numéricos de la matriz de capa de aplicación.

|  |  |  |
| --- | --- | --- |
| **CAPA DE RED** | | |
| **Caracteristicas evaluadas** | **Promedio de los datos benignos** | **Promedio de los datos malignos** |
| **CR1** | 32,792 | 22,470 |
| **CR2** | 10,638 | 2,478 |
| **CR3** | 5,560 | 4,093 |
| **CR4** | 3.562,863 | 2.691,379 |
| **CR5** | 0 | 0 |
| **CR6** | 0 | 0 |
| **CR7** | 37,995 | 27,664 |
| **CR8** | 37,899 | 25,426 |
| **CR9** | 35.038,544 | 11.117,653 |
| **CR10** | 3.961,747 | 3.088,232 |
| **CR11** | 8.785.775,184 | 190.370.458,759 |
| **CR12** | 643.211,483 | 139,188 |
| **CR13** | 609.425,441 | 130,589 |
| **CR14** | 37,995 | 27,664 |
| **CR15** | 5,1743 | 5,193 |

Tabla 6. Comparación entre el promedio de los datos de la matriz de capa de red

Por cada una de las tablas comparativas mostradas al inicio de esta sección, se presentan las comparaciones entre una URL maligna y una benigna. La mayoría de los resultados son similares a los expuestos en el artículo [14], es decir, sigue existiendo el patrón encontrado hace aproximadamente 5 años, por ejemplo, la cantidad de caracteres extraños que hay n una URL malignas es mayor que la cantidad en las benignas, y, probablemente no va a cambiar en un periodo corto de tiempo, esto es, 5-6 años aproximadamente. Sin embargo, dentro de la capa de red se evidenciaron cambios en cuando al promedio de los paquetes TCP enviados al servidor, en nuestro proyecto no da que el promedio de envíos en las URL's benignas es mayor al de malignas; los paquetes enviados a servidor remoto, que en el artículo es mayor el promedio en las URL's malignas; El *source* app bytes, e mayor el promedio en el artículo[14]; *avarege local packet rate,*  que a nosotros no dio mayor promedio en las URL's benignas y el artículo muestra lo contrario; y por último, los paquetes app, que presenta el mismo cambio que las características anteriores. Estos cambios, se pueden generar por el contenido de la URL, es decir, alguna de estas era paginas porno y, puede que dentro del *dataset* que el artículo manejaba, existía una menor proporción que la que se maneja en el nuestro.

A continuación, se muestra la correlación de los datos de la capa de red y aplicación:

**Capa de red con URL’s benignas**

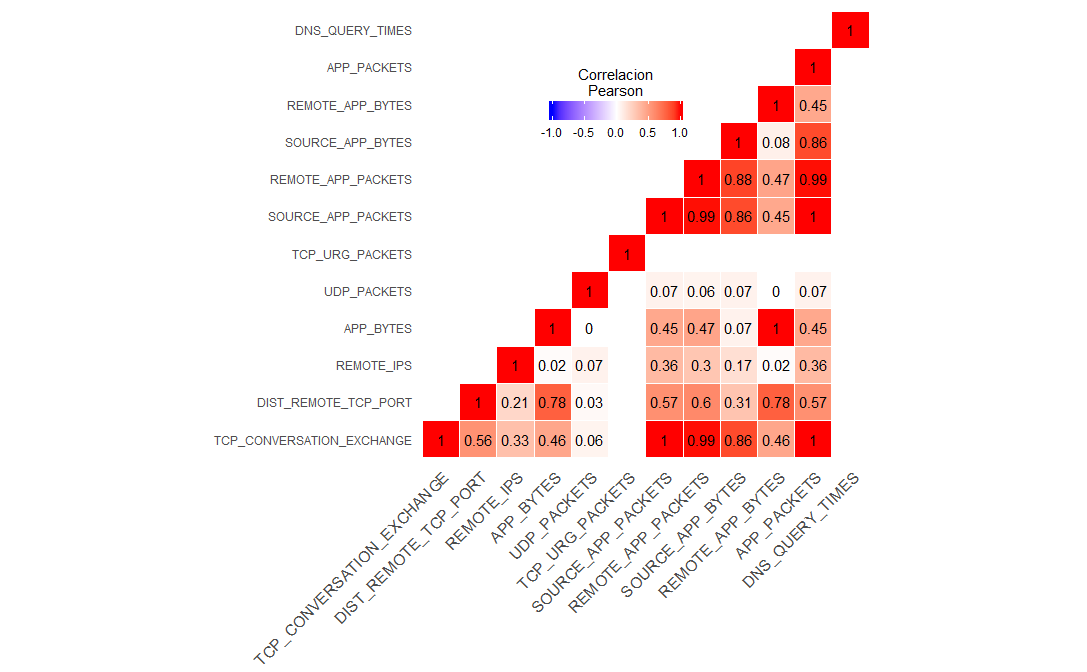


Ilustración 1. Correlación de la capa de red benigna

La ilustración 1. muestra que para la capa de red con URL’s benignas las características con menos interdependencias en relaciones asociadas son *remote\_IPS, UDP\_Packets y, TCP\_URG\_Packets*. Ésta última, muestra que no tiene datos y, por ende, está vacía. La ilustración entonces, indica que estas características no debemos tenerlas en cuenta cuando se vayan a entrenar los algoritmos

**Capa de red con ULR’s malignas**

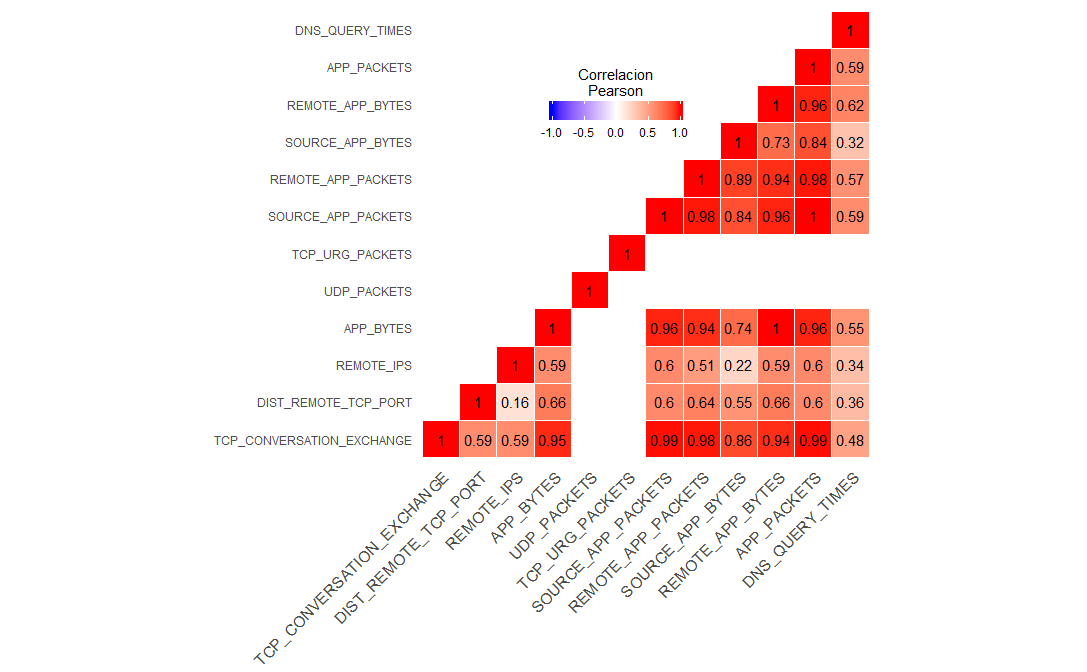


Ilustración 2. Correlación de la capa de red maligna

La ilustración 2. muestra que para la capa de red con URL’s malignas las características con menos interdependencias en relaciones asociadas son *UDP\_Packets y TCP\_URG\_Packets*. En comparación con las benignas, las URL’s malignas tienen una mayor dependencia del valor remote\_IPS, esto se debe a quehay mucha más cantidad de redirecciónes realizadas al *honeypot.* La ilustración 2 indica las características que no debemos tener en cuenta en el entrenamiento de los algoritmos.

Para la ilustración 3 y 4, se muestra la correlación entre los datos numéricos; debido a que esta capa tiene muchos valores no numéricos y al aplicarle el proceso de binarización, el resultado es una matriz bastante grande que genera problemas a la hora de observar la dependencia entre los datos. Sin embargo, cabe recalcar que las fechas (de registro y modificación) y el dominio no mostraron mayor relación con las otras variables. Las fechas, porque no se pueden repetir, es decir, no se puede devolver el tiempo para llegar a la fecha de creación o modificación y montar o actualizar el servidor y, el dominio, probablemente lo han acabado o en el caso de ser una URL benigna, lo dejaron de utilizar.

**Capa de aplicación con URL’s benignas:**

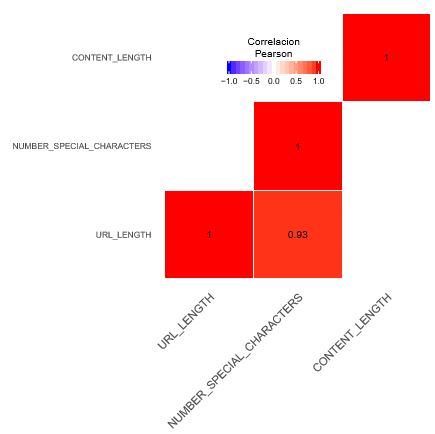


Ilustración 3. Correlación entre datos numéricos de la capa de aplicación benigna

La ilustración 3, muestra que la correlación es muy alta entre el número de caracteres especiales y la longitud de la URL y que el content\_length, no tiene datos para capturar la correlación con las otras dos variables.

**Capa de aplicación con URL’s malignos (solo valores numéricos):**

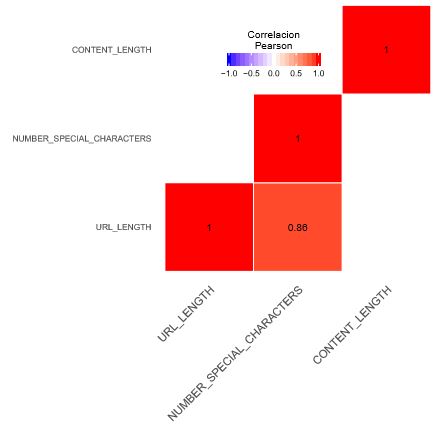


Ilustración 4. correlación entre datos numéricos de la capa de aplicación maligna

La ilustración 4, muestra que la correlación es alta entre el número de caracteres especiales y la longitud de la URL, no tan alta como la dependencia de la capa benigna, pero es un buen valor de dependencia y se observa que, así como en la ilustración 3, CA5, no tiene datos que permitan capturar la correlación con las otras dos variables.

Teniendo en cuenta los análisis de las variables realizados y mostrados anteriormente, se exponen a continuación los resultados de los algoritmos de entrenamiento, se debe tener en cuenta que ésta tiene 967 observaciones y 400 variables (por la binarización de las características y la separación de los valores del cache control). Para esta parte, primero se evaluaron por separado, matriz de la capa de aplicación y, finalmente, la matriz de la capa de red y la matriz completa. Además, se aplicó la técnica de validación cruzada, con un k igual a 10 e igual a 15, para la evaluación de resultados, garantizando así la independencia de la partición de los datos correspondiente al entrenamiento y pruebas. Ahora bien, las características evaluadas por parte de la capa de aplicación son CA1, CA2, CA3, CA4, CA5, CA8 y CA9. Mientras que las evaluadas por parte de la capa de red son CR1, CR2, CR3, CR4, CR5, CR6, CR7, CR8, CR9, CR13 y CR14.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algoritmo** | **Estimación de la exactitud de la clasificación con K=10 para *dataset aplicación*** | **Tiempo de respuesta dataset aplicación** | **Estimación de la exactitud de la clasificación con K=10 para *dataset de red*** | **Tiempo de respuesta dataset red** |
| SVM | 89,09% | 2, 01 s | 55,16% | 1,90 s |
| Regresión Logística | 88,43% | 3,56 s | 54,11% | 0,87 s |
| *Naive Bayes* | 84,7% | *3,38 s* | 55,16% | 0,89 s |
| *J48* | *90, 10 %* | *4 s* | *57, 01 %* | *4, 05 s* |

*Tabla 7. Resultados de los algoritmos por cada capa*

A continuación, en la tabla 8, se muestra el resultado de las características más importantes mencionadas en el artículo, se encuentra una precisión adecuada para el algoritmo J48 con un porcentaje de 96.05%. Se puede distinguir entonces que estas características resultan ser muy importantes para la detección de páginas maliciosas tal como lo explica en el artículo de *Cross-Layer* [14].

|  |  |  |
| --- | --- | --- |
| ***Algoritmo*** | **Estimación de la exactitud de la clasificación con K=10 para *dataset de los algoritmos (subset e infogain)*** | ***Tiempo respuesta*** |
| *SVM* | 85.46% | *0.19 s* |
| *Regresión Logística* | 84.51% | *0.06 s* |
| *Naive Bayes* | 85.46% | *0.02 s* |
| *J48* | 96.05% | 0.01 s |

*Tabla 8. Resultados de los algoritmos para las 3 características obtenidas por los métodos subset e infogain del articulo [14].*

Finalizando con esta sección, se dan a conocer los resultados de los algoritmos ejecutados con la matriz completa. Es importante resaltar que aunque la matrices de correlación de las capas de red, Ilustración 1 y 2, mostraron poca dependencia de las variables UDP\_Packets y TCP\_URG\_Packets, se decidió tenerlas en cuenta para el entrenamiento, porque aunque no arrojaron buenos resultados, generan datos importantes y a futuro si no se tiene en cuenta, pueden realizar ataques por estos medios, siendo UDP\_packets encargado de revisar streaming, videos, mp3 y TCP\_URG\_Packets encargado de los flasks, que guardan información cuando ocurre algún problema en los paquetes. Finalmente, se aplicó en la validación cruzada un valor de k 10 y 15.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algoritmo** | **Estimación de la exactitud de la clasificación con K=10** | **Tiempo de respuesta k=10** | **Estimación de la exactitud de la clasificación con K=15** | **Tiempo de respuesta k=15** |
| SVM | 97,41% | 3,38 s | 96,74% | 3, 45 s |
| Regresión logística | 90,58% | 5, 31 s | 90,18% | 5, 40 s |
| J48 | 98,76% | 53, 37 s | 99,07% | 53, 43 s |
| *Naive Bayes* | 10,96% | 2, 28 s | 10,96% | 2, 32 s |

*Tabla 9. Resultados de los algoritmos por toda la matriz de datos*

### XI. iii. Análisis

En esta sección, se va a dar el análisis encontrado gracias a los resultados suministrados por el entrenamiento de los algoritmos.

Según los algoritmos, cuando se prueba con la matriz de aplicación, la probabilidad de que la clasificación este buena, es mayor al 90%. Por otro lado, si se prueba con la matriz de red, las clasificaciones correctas no aumentan del 70%. Analizar cada uno de los algoritmos con cada una de las capas es importante, para examinar como los datos se comportan y como ayudan a la clasificación de la URL, es decir, según los resultados arrojados, suponemos entonces, que las características de la capa de aplicación van a aportar mayor peso para la toma de decisión de la clasificación, cuando se haga el entrenamiento con la matriz completa, la que une ambas capas. Finalmente, se puede observar que no hay datos para la matriz de capa de red cuando se entrena con *Naive Bayes*, esto es, porque al pasar la matriz, no procesaba los datos y éstos no se podían modificar, pues si se hacía, no era entonces comparable con los otros algoritmos.

Por otro lado, los mejores algoritmos para entrenar la matriz completa, fueron SVM y J48, ambos superando el 95% de aciertos en la clasificación. Sin embargo, en los resultados obtenidos permitieron evidenciar que el algoritmo con mejores resultados es J48, con ambas k y la mejor decisión para entrenar, es k=15. Además, *Naive Bayes,* no es una buena opción para el entrenamiento, en todas las situaciones se obtuvieron los más bajos resultados.

Finalmente, con respecto a la comparación que se realizó con las características más fuertes según expuso el artículo [14] y lo realizado en este proyecto, se puede corroborar que la información que se obtuvo por parte de nuestros clasificadores, son muy similares a lo que presentan los autores de *Cross Layer.* Sin embargo, nosotros optamos por el manejo de no solo las tres características que ellos mencionaron, sino de todas las que más pudimos obtener y a su vez, a la hora de hacer el estudio de las mismas, indicaban una buena correlación, por las siguientes dos razones:

Con respecto al *dataset* que se genera, es decir, puede quedar con tan pocos datos, que altere la clasificación de las URL's.

Una persona puede alterar su URL, es decir, una página es maliciosa, pero el dueño de ésta puede alterar la URL para que cumpla con las tres características que una URL benigna en promedio tiene (las que menciona el artículo que se ha venido tratando [14].

## XII. Conclusiones y trabajos a futuro

Durante el desarrollo de éste proyecto, se hicieron experimentos con distintos tipos de *honeypots,* que mostraron una deficiencia en la documentación y uso de estas, como también su poca actualización llevando a problemas de incompatibilidad; distintas herramientas para la captura del tráfico de cada URL; se estudiaron los paquetes que arrojaba el tráfico, tomándonos más de dos meses el experimento y el estudio; se codificaron los algoritmos para la extracción de características de aplicación y de red, se revisaron éstos y se recodificaron, volviéndolos más eficientes; se analizó la información de los *datasets* y se acomodó para que fuera consistente y se pudiera pasar por los algoritmos propuestos para el entrenamiento; se estudiaron y probaron diferentes maneras de mostrar la información y los resultados de los análisis exploratorios y los algoritmos de *Machine Learning*, para que fuera comprendida de mejor manera. Finalmente, con todo lo mencionado y evidenciado, este proyecto deja cinco *datasets*, dos para cada capa y uno que contiene ambas capas; todos los algoritmos que se implementaron; todas las gráficas obtenidas; y la base para la ejecución del software de detección.

Por otro lado, al realizar este proyecto, nos dimos cuenta que cada que avanzábamos en el mismo, encontrábamos un proyecto o una continuidad más a desarrollar. Dentro de estos, se proponen como trabajos a futuro lo siguiente:

Sistema de detección de páginas maliciosas por medio de aprendizaje no supervisado.

Detección de tipos de ataques web por medio de aprendizaje supervisado y no supervisado.

Investigar honeypots y herramientas que permitan un mejor avance en el desarrollo de esta problemática.

Obtención de *datasets* que permitan el desarrollo experimental y algoritmos que permitan identificar ataques y problemas en una página web.

Desarrollar una herramienta para la detección de ataques en aplicaciones móviles.

Desarrollar una herramienta que permita en tiempo real la detección de ataques e identificación del mismo. Para este, se deja el tiempo obtenido por cada algoritmo ejecutado.

## XIII. Referencias

[1] Grossman, J. (2015). Whitehat website security statistics report. Retrieved. September, 1, 2016.

[2] W.-K. Chen, Linear Networks and Systems (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[3] P.K. Chan and R. P. Loppmann, “Machine learning for computer security”, J. Mach. Learn. Res., col. 7, pp.2669-2672, 2006.

[4] The Open Web Application Security Project. OWASP Top 10 - 2013 The Ten Most Critical Web Application Security Risks. (2013).

[5] A. Halsall, "Introducción". in Redes de computadoras e internet, R. Moreno. ADDISON-WESLEY: Madrid, 2006, Páginas (20 - 24 pp.)

[6] L.A. Munárriz, Fundamentos de inteligencia artificial (Vol. 1). Editum, 1994.

[7] D. R. Sosa, “Inteligencia Artificial”, 2013.

[8] E. Alpaydın, Introduction to machine learning, 2th ed. Cambridge, Massachusetts; London, England: Massachusetts Institute of Technology, 2010.

[9] T. Mitchell, “Machine learning”, Match. Learn., no. X, pp. 634-644, 2012.

[10] Xie, Y., & Aiken, A. (2006, July). Static Detection of Security Vulnerabilities in Scripting Languages. In USENIX Security (Vol. 6, pp. 179-192).

[11] L. Xu, Z. Zhan, S. Xu, and K. Ye, “Cross-layer detection of malicious websites,” *Proc. ACM Conf. Data Appl. Secur. Priv.*, pp. 141–152, 2013.

[12] L. Fuentes, “Malware, una amenaza de internet”, 2008.

[13] S. Rao, “International Journal of Enterprise Computing and Business System ISSN (Online): 2230-8849 WEB APPLICATION VULNERABILITY DETECTION USING DYNAMIC ANALYSIS”, vol. 2, no. 1, 2012.

[14] L. Xu, Z. Zhan, S. Xu, and K. Ye, “Cross-layer detection of malicious websites,” *Proc. ACM Conf. Data Appl. Secur. Priv.*, pp. 141–152, 2013.

[15] Y. Zhong, H. Asakura, H. Takakura, and Y. Oshima, “*Detecting Malicious Inputs of Web Application Parameters Using Character Class Sequences,”* 2015 IEEE 39th Annu. Comput. Softw. Appl. Conf., pp. 252- 532, 2015.

[16] A. Mohaisen, “*Towards Automatic and Lightweight Detection and Classification of malicious Web Contents*”, 2015.

[17] KimiNet, “ *PyShark”,* [Online]. Available: <http://kiminewt.github.io/pyshark/> . [Accessed: April -17]

[18] spectrum.ieee.org, “*Interactive: The Top Programming Lenguages 2016”,* 2016. [Online]. Available: <http://spectrum.ieee.org/static/index/2016> . [Accessed: Nov- 3-16]

[19] A. McCallum and K. Nigam, “A Comparison of Event Models for Naïve Bayes Text Classification,” *AAAI/ICML- 98 Work. Learn. Text Categ.,* pp. 41- 48, 1998.

[20] G. Kaur and A. Chhabra, “Improved J48 Classification Algorithm for the Prediction of Diabetes,” *Int. J. Comput. Appl.,* vol. 98, no. 22, pp. 13-17, 2014.

[21] Ray S., “6 Easy Steps to learn naïve bayes algorithm*”,* 2015. [Online]. Available: <https://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/> . [Accessed: April- 17]

[22] Pressman, R. S., & Troya, J. M. (1988). *Ingeniería del software* (No. 001.64 P74s.). McGraw Hill.

[23] Hipertextual.com. (2016, Oct, 21). Esta es la razón por la que muchas webs importantes están caídas. [Online]. Availaible: <https://hipertextual.com/2016/10/ataque-dns-webs-caidas>. [Accessed: Nov- 3-16]

[24] Elpais.com.co, “Página web de la Registraduría sufrió un ataque informático,” 2016. [Online]. Available: http://www.elpais.com.co/elpais/colombia/proceso-paz/noticias/pagina-web-registraduria-sufrio-ataque-informatico#.V-yIXlhusjg.email. [Accessed: 16-Oct-2016].

[25] E. Ploir, “Algoritmos SVM para problemas sobre big data”.