OFICINA DE POSGRADOS

TEMA:

SISTEMA DE MONITOREO Y DETECCIÓN DE DEFACEMENTS EN SITIOS WEB. UN ENFOQUE MODERNO.

Proyecto de Investigación previo a la obtención del título de Magister en Ciberseguridad

Línea de Investigación:

Protección de datos y comunicaciones

Autor:

Ing. Alexander Israel Rojas Buenaño

Director:

Mg. Paúl Fernando Bernal Barzallo

Ambato – Ecuador

Diciembre 2021

PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE AMBATO

HOJA DE APROBACIÓN

Tema:

SISTEMA DE MONITOREO Y DETECCIÓN DE DEFACEMENTS EN SITIOS WEB. UN ENFOQUE MODERNO

Línea de Investigación:

Protección de datos y comunicaciones

Autor:

Alexander Israel Rojas Buenaño

Paúl Fernando Bernal Barzallo, Msc. f. _____
CALIFICADOR

José Marcelo Balseca Manzano, Msc. f. _____
CALIFICADOR

Liliana del Rocío Mena Hernández, Msc. f. _____
CALIFICADOR

Padre Juan Carlos Acosta, PhD. f. _____
COORDINADOR DE LA OFICINA DE POSGRADOS

Hugo Rogelio Altamirano Villarroel, Dr. f. _____
SECRETARIO GENERAL PUCESA

Ambato – Ecuador

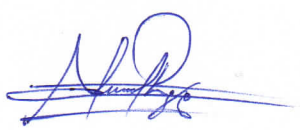
Diciembre 2021

DECLARACIÓN DE AUTENTICIDAD Y RESPONSABILIDAD

Yo: **ALEXANDER ISRAEL ROJAS BUENAÑO,** con CC. **160046869-6** autor del trabajo de graduación intitulado: "SISTEMA DE MONITOREO Y DETECCIÓN DE DEFACEMENTS EN SITIOS WEB. UN ENFOQUE MODERNO", previa a la obtención del título profesional de **MAGISTER EN CIBERSEGURIDAD,** en la **OFICINA DE POSGRADOS**.

1. Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
2. Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través del sitio web de la Biblioteca de la PUCE Ambato, el referido trabajo de graduación, respetando las políticas de propiedad intelectual de Universidad.

Ambato, diciembre 2021



ALEXANDER ISRAEL ROJAS BUENAÑO

CC. 160046869-6

AGRADECIMIENTO

Agradezco a mi esposa y mi hija por el amor y apoyo incondicional que me brindaron en esta etapa de mi vida, agradezco a Dios y mis padres por enseñarme que la constancia y el esfuerzo son virtudes necesarias para la vida.

De igual manera agradezco a nuestra coordinadora Msc. Teresita Freire por su firme acompañamiento en todo el proceso, así como, también, a mi tutor Msc. Paúl Bernal por su experiencia y consejos compartido para el desarrollo del presente trabajo.

DEDICATORIA

Dedico este trabajo a mi esposa Lau y mi hija Gia, es el resultado de su apoyo y amor para llevar adelante este nuevo objetivo profesional. A mis padres, quienes siempre están presentes para apoyarme y alentar a seguir adelante.

RESUMEN

Hoy en día las empresas buscan tener presencia en internet pues conocen la importancia de estar al alcance de sus clientes o público objetivo a tan solo un par de clics. Para ello buscan posicionarse con su sitio web en la red y brindar la información necesaria de sus productos o servicios. Sin embargo, es inevitable descartar la presencia de ciberamenazas. Muestra de ello son los atacantes, que motivados por diferentes causas (política, activismo, rédito monetario) buscan vulnerar la seguridad de los sitios web y robar información o generar indisponibilidad en los servicios, esto afecta la reputación de la empresa y genera pérdidas económicas. El Defacement o desfiguración es una de las consecuencias tras la materialización de un ataque. En este incidente los ciberdelincuentes cambian el contenido de la página y colocan mensajes de protestas o burlas que dejan en evidencia las fallas de seguridad existentes en el sitio web. Esto empeora con el tiempo que toma identificar el ataque y ejecutar acciones al respecto. Por ello, el objetivo de este trabajo es implementar un Sistema de Monitoreo y Detección de Defacement en sitios web, para lo cual, se lleva a cabo una investigación de un modelo de Machine Learning que permita entrenar un algoritmo capaz de identificar el contenido de una página web y determinar si se encuentra desfigurada o no. De esta forma, se notifica al administrador del sitio web acerca del Defacement y se reduce así el tiempo de afectación.

Palabras clave: Defacement, Ataques Web, Machine Learning, Hacktivismo.

ABSTRACT

Nowadays companies look for opportunities to be promoted on the internet because they know the importance of reach clients or users in just a couple of clicks. Therefore, they seek to promote their website on the network and provide necessary information about their products or services. However, it is relevant to consider the presence of cyber threats. There are different attackers who motivated for several causes (politics, activism, money) try to break the security of websites and steal information or the unavailability of services affecting the reputation of the company and generating economic losses. Defacement is one of the consequences after the materialization of an attack. In this incident, cyber criminals change the content of the page and place messages of protests or mockery exposing the security faults in the website. It takes time to identify the attack and take action on it. Therefore, the goal of this research is to develop a Defacement Detection and Monitoring System for websites through an investigation of Machine Learning algorithms in order to identify the content of a web page and determining whether it is defaced or not. In this way, the website administrator can be notified about the Defacement reducing the time of impact in the companies.

Keywords: Defacement, Web Attack, Machine Learning, Hacktivismo.

ÍNDICE

PRELIMINARES

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS

INTRODUCCIÓN

El acelerado mundo de internet ha obligado a que empresas e instituciones busquen su presencia en el ciberespacio[1] y la situación que el mundo atraviesa por la pandemia del Covid-19 no ha hecho más que precipitar este suceso. Sin embargo, el enfoque de la ciberseguridad ha quedado de lado y deja la puerta abierta a los ciberdelincuentes que buscan vulnerar y comprometer los sitios web. Uno de los ataques más comunes es el *Defacement* o desfiguración que tiene como objetivo afectar la reputación, generar indisponibilidad e incluso provocar pérdidas económicas a las víctimas.

A nivel mundial según el sitio web Zone-H, se conoce que cerca de 500,000 sitios web han sido víctimas de ataques de *Defacement* en lo que va del año 2020 (Zone-H, s. f.-b). Mientras que a nivel país, se han reportado 77 sitios web, que se han visto afectados por este tipo de ataque en el mismo periodo (Zone-H, s. f.-a). Esto incluye instituciones de gobierno, educativas y empresariales. Si un sitio es vulnerable, independientemente de su naturaleza, puede verse comprometido.

El desconocimiento o la falta de administración por parte de quienes gestionan los sitios web hacen que estos ataques estén a la orden del día. A esto, se suma otro problema, el tiempo de respuesta que toma a los administradores del sitio identificar la afectación sufrida. Por lo tanto, el problema científico, se plantea de la siguiente manera: ¿Cómo reducir el tiempo de afectación a la disponibilidad y reputación de la imagen en los sitios web de empresas e instituciones vulneradas por ataques de *Defacement*?

Es importante referir que existe un trabajo previo relacionado con el estudio de herramientas de software libre para la creación de sistemas de detección automatizada de *Defacement* en sitios web (Bernal, 2014), por lo cual, se plantea mejorar dicho estudio a través de un Sistema de Monitoreo y Detección de *Defacement* en sitios web con un enfoque moderno, que busca reducir el tiempo de afectación a la disponibilidad y reputación de la imagen en los sitios web de empresas e instituciones vulneradas

[1] Ámbito virtual creado por medios informáticos(ASALE & RAE, s. f.).

por estos ataques. De esta forma los administradores o responsables de los sitios web tomarían acciones ante una pronta alerta.

Para la consecución del proyecto, se percibe el siguiente objetivo general: Implementar un Sistema de Monitoreo y Detección de *Defacement* en sitios web, a esto, se suman los objetivos específicos que consisten en:

1. Profundizar en el estado del arte de los sistemas de monitoreo y detección de *Defacement.*
2. Analizar los beneficios de herramientas y repositorios de software disponible para el diagnóstico de control de cambios.
3. Diseñar un sistema de monitoreo no intrusivo que aplique metodologías ágiles que permitan enviar notificaciones automatizadas a los administradores de los sitios.

Se hace uso de la metodología deductiva al considerar, que se parte de la premisa general en, la cual, un sistema de monitoreo y detección de *Defacements* en sitios web permite reducir el tiempo de respuesta ante un ataque materializado de forma, que se tomen acciones inmediatamente. Se adopta, también, el proceso cuantitativo pues soporta el método cuasiexperimental en vista de que, no se dispone de un grupo de control y se considera que es estrictamente necesario trabajar con sitios web de distinta índole para lograr mejores resultados y consecuentemente descartar falsos positivos.

El proyecto, se justifica y muestra relevancia social al plantear un sistema no intrusivo de modo que, no se requiere la adquisición de *hardware* o *software* adicional en la infraestructura de los sitios a monitorear, así como tampoco un acceso directo al servidor web. Por otro lado, se busca identificar nuevos métodos que permitan validar el control de cambios en los elementos de los sitios web para asegurar la eficacia de las detecciones y reducir en la mayor cantidad posible la identificación de los falsos positivos, este es un aporte de valor teórico.

CAPÍTULO I. ESTADO DEL ARTE Y LA PRÁCTICA

1.1 Seguridad en sitios Web

El ciberespacio, se ha convertido en parte de la vida común de las personas, muestra de ello es que según afirma el portal DATAREPORTAL, en el 2020 alrededor de 4,538 millones de personas en todo el mundo utilizan Internet y estiman que pasan alrededor de 6 horas con 43 minutos de su tiempo al día (Simon Kemp, 2020). Los sitios web son ese puente indispensable para la interacción entre los usuarios y las empresas u organizaciones, permiten realizar transacciones financieras, acceder a noticias y entretenimiento, interactuar con instituciones de gobierno entre otros servicios. Por lo tanto, es de esperar, que siempre se encuentren disponibles y proporcionen facilidad y usabilidad a los usuarios (Gillman et al., 2015).

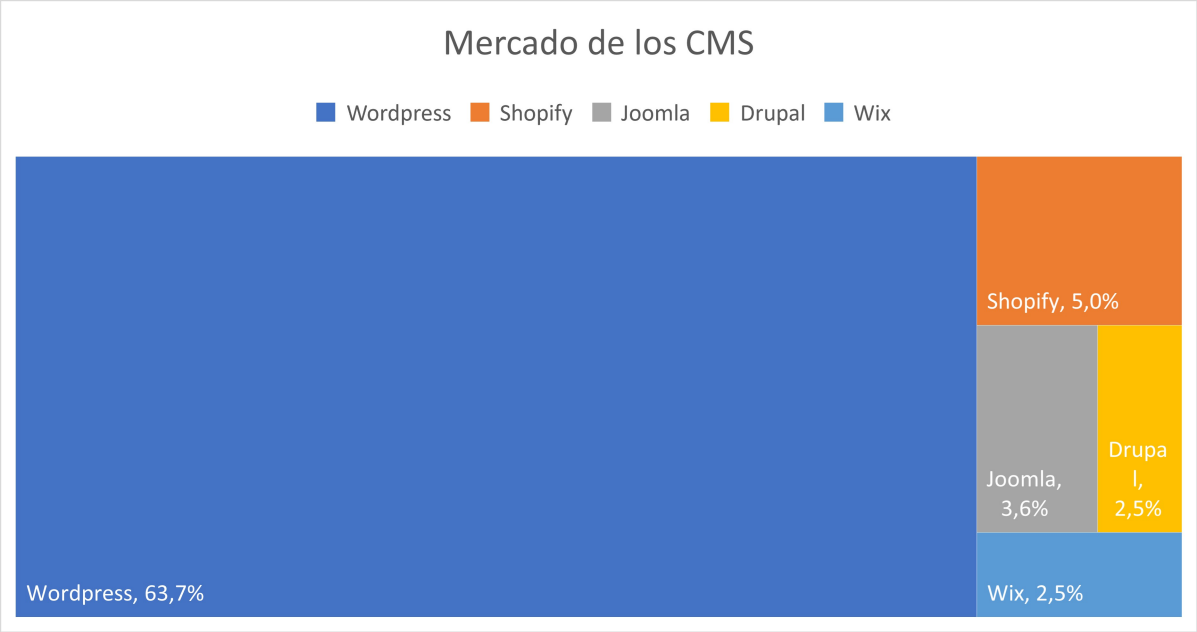
Por consiguiente, esto ha acelerado el hecho de que empresas e instituciones busquen tener presencia en internet y ofrecen un diseño llamativo y agradable de sus sitios hacia los usuarios, sin embargo, no siempre consideran la gestión y seguridad al momento del diseño e implementación y lamentablemente estas actividades quedan rezagadas cuando requieren ir de la mano y trabajarse de forma paralela; a esta premisa, se suma la falta de conocimiento técnico por parte del personal que administra la página web. Esta información, se contrasta con el informe de investigación de amenazas en internet de Sucuri en donde se identificó que más del 60% de sitios web eran vulnerables a ataques conocidos y que el 47% de estos habían sido infectados tras disponer de una o más puertas traseras que permitían a los atacantes mantener el acceso y control de los entornos comprometidos (Sucuri, 2019).

Con la información anterior, se asevera que el crecimiento tecnológico va en aumento y de igual manera lo hacen los cibercriminales quienes buscan sacar provecho de vulnerabilidades en el diseño de las soluciones tecnologías o bien de la falta de administración y concientización respecto a la importancia de la ciberseguridad y fortalecimiento de dichas soluciones.

Hoy en día, se dispone de varias tecnologías para el diseño e implementación de un sitio web, los Sistemas de Gestión de Contenidos (CMS) son las plataformas

principales con una cuota de mercado superior al 47,3% de sitios de internet (W3Techs, 2020). Esto, se debe a que los CMS son programas que permiten crear, editar y publicar contenido web a través de una interfaz de administración bastante amigable (IONOS, 2020). Dentro de los CMS, WordPress es el líder significativo. Esta información, se aprecia en el gráfico 1.

Gráfico 1. Mercado de los CMS



Fuente: tomado a partir de IONOS (2020)

Si bien es cierto, el uso de CMS facilita el diseño y la creación de sitios web, sin una correcta gestión del mantenimiento y la seguridad de estos por parte de los administradores, se deja la puerta abierta a ciberdelincuentes para que aprovechen debilidades de diseño que no han sido gestionadas o tratadas oportunamente. De esta manera, los atacantes afectan y comprometen tanto la integridad como la disponibilidad del servicio que presta el sitio web, con lo cual, se genera un daño a la reputación de la empresa o instituciones víctimas, y en ocasiones incluyen repercusiones económicas(Maggi et al., 2018).

“Los ataques a sitios web son amenazas activas; todo depende de tiempo, dinero y motivación por parte del atacante” (D Avi, 2020). El Proyecto Abierto de Seguridad en

Aplicaciones Web (OWASP) mantiene un documento que trata un amplio consenso sobre los diez riesgos de seguridad más críticos para las aplicaciones web, para cada uno de los riegos, se proporciona información genérica sobre la probabilidad y el impacto técnico, para ello, se utiliza el siguiente esquema de evaluación:

Tabla 1. Esquema de Evaluación OWASP

Agente de Amenaza	Explotabilidad	Prevalencia de Vulnerabilidad	Detección de Vulnerabilidad	Impacto Técnico	Impacto de Negocio
Específico de la Aplicación	Fácil 3	Difundido 3	Fácil 3	Severo 3	Específico del Negocio
	Promedio 2	Común 2	Promedio 2	Moderado 2	
	Difícil 1	Poco Común 1	Difícil 1	Mínimo 1	

Fuente - tomado a partir de OWASP (2017)

En base a dicho esquema, se califican los riesgos y son priorizados por el más crítico. A continuación, se da una breve explicación de los mismos (OWASP, 2017):

I. **Inyección**

   Un atacante manipula los parámetros, que se envían en una solicitud hacia la aplicación web para alterar las consultas resultantes, se logra en algunos casos ingresar como administrador. Otros usos de este ataque serían robar, cambiar e incluso borrar datos y todo rastro de la actividad maliciosa.

II. **Pérdida de Autenticación**

   Gran parte de las aplicaciones requieren que sus usuarios inicien sesión antes de usarla, generalmente hacen uso de credenciales de nombre de usuario y contraseña. Sin embargo, existen muchos tipos de fallas de en estos sistemas de autenticación que llegan a explotarse de diversas maneras: fuerza bruta, ataques de diccionario, secuestro de sesiones, etc.

III. **Exposición de Datos Sensibles**

   Los datos privados y confidenciales, que se almacenan en las bases de datos de las aplicaciones web necesitan protegerse con cifrado y otros algoritmos criptográficos. Sin embargo, estas acciones no siempre son implementadas o son ejecutadas de forma incompleta. Esto permite a los atacantes robar

información como: credenciales, tarjetas de crédito, información personal y otros datos críticos para la empresa o institución.

IV. Entidades Externas XML (XXE)

Usualmente, por su actividad, las aplicaciones necesitan procesar documentos XML que proveen los usuarios. Los analizadores XML antiguos o mal configurados habilitan una función conocida como referencias de entidad externa, que cuando, se evalúan incrustan el contenido de otro archivo. Los atacantes hacen uso de esta debilidad de diseño para leer datos confidenciales, acceder a sistemas internos e incluso afectar la disponibilidad de la aplicación en un ataque de denegación de servicio (DoS).

V. Pérdida de Control de Acceso

Las aplicaciones web suelen limitar los accesos y acciones que disponen los usuarios en base a roles o perfiles. Sin embargo, estas medidas pueden no haberse implementado correctamente o tener fallos en su diseño. Los atacantes eluden o saltan estos controles para acceder a funciones o información no autorizadas como acceder a las cuentas de otros usuarios, ver archivos confidenciales, modificar los datos de otros usuarios, realizar acciones administrativas y más.

VI. Configuración de Seguridad Incorrecta

Los servidores y las aplicaciones manejan varias capas que requieren configurarse de forma correcta. Esto pasa por el sistema operativo y los dispositivos de red hasta el servidor web y la propia aplicación. Las configuraciones predeterminadas dejan archivos desprotegidos, credenciales por defecto, servicios activos que no son necesarios, así como muchas otras configuraciones inseguras que permitirían que un atacante obtenga acceso al sistema o datos.

VII. *Cross-Site Scripting* (XSS)

Un atacante modifica las páginas web que otros usuarios ven en su aplicación, ya sea para robar información como contraseñas y tarjetas de crédito, difundir datos falsos, secuestrar sesiones de usuario, redirigir a otro sitio o ejecutar *scripts* maliciosos en el sitio de la víctima. navegador. Esta vulnerabilidad ocurre

cuando se incluyen datos que no son de confianza en una página web, sin la validación adecuada. El atacante envía formularios con fragmentos de HTML o *JavaScript*, que se incrusten directamente en la página y el navegador los procesa.

VIII. Deserialización Insegura

La deserialización insegura permite ataques de inyección y escalada de privilegios, e incluso conducir a la ejecución remota de código y la toma de control del servidor en determinadas situaciones. Muchas aplicaciones necesitan serializar objetos y datos en un formato que pueda transmitirse fácilmente a través del medio o incluso almacenarse en un archivo. Cuando una aplicación restaura estos objetos de nuevo a la memoria y deserializa los datos formateados recibidos de un usuario, es posible alterar la memoria del objeto e incluso hacer que ejecute funciones arbitrarias.

IX. Uso de Componentes con Vulnerabilidades Conocida

Las aplicaciones web, se basan cada vez más en componentes, *frameworks*[2] y librerías de terceros. Cualquier vulnerabilidad, que se encuentre en estas dependencias, también, afecta directamente a la aplicación y esto conduce a una alta probabilidad de materialización de otras vulnerabilidades.

X. Registro y Monitoreo Insuficientes

El registro y monitoreo insuficiente permiten a un ciberdelincuente que tras haber ejecutado un ataque, no se encuentre evidencia suficiente para identificar las acciones, que se llevaron a cabo, lo cual, complica la recuperación y planes a ejecutar por parte de los afectados.

1.2 Defacement

Hasta el momento, se ha mencionado una breve situación actual del uso de internet, así como las principales plataformas tecnologías de sitios web y el top diez de las amenazas más críticas a los que estos, se ven expuestos. Esta información era necesaria para contextualizar los ataques de *Defacement* o desfiguración del sitio web,

[2] Esquema o estructura, que se establece y que se aprovecha para desarrollar y organizar un software determinado(NeoAttack, 2020)

son resultados consecuentes de la materialización de alguna de las amenazas antes listadas sobre un sitio web. Este tipo de ataque, se basa en modificar de forma total o parcial el contenido que ofrece un sitio web y colocar mensajes de rechazo o discrepancia relacionados con temas políticos, sociales, religiosos entre otros; a estas medidas de protesta los atacantes las defienden como *Hacktivismo* (Maggi et al., 2018).

En el gráfico 2, se observa el mensaje tras el ataque que sufrieron los sitios de la Organización Internacional para las Migraciones de las Naciones Unidas el 19 de agosto del presente año (San José, 2020).

Gráfico 2. Mensaje de *Defacement* a sitios de la Organización Internacional para las Migraciones

Fuente: tomado a partir de San José (2020)

Por otro lado, existe también, el ímpetu y ego de los ciberdelincuentes quienes quieren demostrar sus habilidades y realizan este tipo de actividades por practicar o competir con sus semejantes (Vasseur, 2018). En el gráfico 3, se observa el mensaje de burla y los alias de los atacantes tras realizar un *Defacement* a un sitio web de los Servicios Nacionales de Salud del Reino Unido en abril del 2018 (BBC, 2018) .

Gráfico 3. *Defacement* a un sitio web de los Servicios Nacionales de Salud del Reino Unido

Fuente: tomado a partir de BBC (2018)

Según González (2020) para llevar a cabo la desfiguración del sitio, los atacantes llevan a cabo técnicas que incluyen:

- Acceso no autorizado
- Inyección SQL
- *Cross-site scripting* (XSS)
- Secuestro de DNS
- Infección de *malware*[3]

Por otro lado, el Instituto Nacional de Ciberseguridad de España (INCIBE) menciona que las principales formas de ataque que usa un ciberdelincuente para apropiarse de un sitio web y manipular su contenido son:

- Robo de credenciales de acceso mediante malware enviado por correo electrónico.
- Explotación de vulnerabilidades en gestores de contenido desactualizados con plugins antiguos o mal configurados.

[3] Término amplio que describe cualquier programa o código malicioso que es dañino para los sistemas (Malwarebytes, s. f.)

- Servidores web infectados por malware o ataques realizados a través de dominios alojados en el mismo servidor ya comprometido.

Adicionalmente la empresa proveedora de servicios Arsys menciona que otro factor a considerar es una incorrecta o una inexistente política de contraseñas lo que ocasiona que los administradores de los sitios manejen credenciales débiles (Arsys, 2019).

Como se observa, son varias las técnicas empleadas por atacantes para llevar a su cabo su cometido, sumadas a sus motivaciones.

1.3 Trabajos Relacionados

Una vez, que se tiene claro el contexto y el problema a abordar, es importante estudiar soluciones expuestas en trabajos relacionados. (Mao & Bagolibe, 2019) presentan una propuesta que no solo se enfoca en la detección sino, también, en la prevención y en un mecanismo de protección para minimizar el impacto del *Defacement* al sitio web. Este modelo, se conforma con un análisis de integridad del sitio, un control de esquema variable, un procedimiento de respaldos y un sistema de notificaciones. Para la validación de la integridad del sitio hacen uso de un *script*[4] desplegado en el servidor web y este se encarga de generar un cifrado *hash*[5] resultante para cada uno de los archivos. Este es un modelo convencional para la detección de cambios en un sitio web y adicionalmente, se requiere acceso al servidor web para llevar el control por lo que se aleja del enfoque del presente trabajo.

Por otro lado Masango et al. (2018), propone una herramienta de Monitoreo de Intrusiones y *Defacement* basado en capas, para lo cual, se requiere acceso al servidor web con el fin de llevar un control a través de *hashes* de todos los archivos del sitio, posteriormente almacena esta información en una base de datos así como una copia de los archivos. De esta forma cada vez, que se quiera verificar si ha existido un cambio la herramienta vuelve a generar los valores *hashes* y los compara con los ya

[4] Programa o secuencia de instrucciones que es interpretado y ejecutado por otro programa en lugar de ser procesado por el procesador del ordenador(*Diccionario informático,* s. f.)

[5] Conjunto de caracteres que identifican unívocamente a un archivo. Es un código que lo diferencia del resto (*Diccionario informático*, s. f.)

almacenados en la base de datos. En el caso de haberse realizado un cambio no controlado o haberse borrado algún archivo es posible realizar una restauración a un punto anterior gracias a la información guardada previamente. Para la administración de esta herramienta, se dispone de una interfaz web que es provista a los encargados. Esta propuesta, también, requiere un acceso al servidor web para registrar el control de cambios y las acciones de respuesta.

El trabajo de Mondragón et al.(2017) trata acerca de un control de seguridad contra *Defacement* de sitios web, el mismo contempla tres fases: la activación de control basada en las características técnicas del sitio, el cálculo del valor de comprobación que consiste en capturar el código fuente y calcular el *hash* que posteriormente, se almacena en la base de datos y finalmente, la desactivación del control por computador que verifica la integridad del sitio protegido, en caso de haberse realizado un cambio y que los valores *hash* no coincidan es posible restaurar la copia de seguridad.

Otra propuesta es la de Bernal (2014) que expone un estudio de herramientas de software libre para la creación de sistemas de detección automatizada de *Defacements* de sitios web. En este trabajo, se plantea llevar un control de cambios a nivel de líneas de código y adicionalmente un control de cambios realizados en una imagen capturada del sitio web, es decir, de forma visual. Vale recalcar que esta propuesta, se toma como base para el desarrollo del presente trabajo y su propuesta no requiere intervención en el servidor web o la aplicación a monitorear.

La propuesta de un modelo multicapa para la detección de *Defacement* de Hoang & Nguyen (2019) es bastante interesante pues incluye dos etapas: entrenamiento y detección. En la etapa de entrenamiento realiza la colección del conjunto de datos del sitio a monitorear, esto incluye datos de archivos HTML, CSS y *JavaScript*. Este conjunto de información es procesado y vectorizado, se usa un término de frecuencia técnica. Posteriormente a la etapa de detección, se realiza una validación de los datos resultantes, así como una comprobación de la integridad de imágenes del sitio, para cada una de ellas se obtiene un valor hash. Si las comprobaciones del modelo muestran cambios, se dispara una alarma al administrador del sitio.

CAPÍTULO II. DISEÑO METODOLÓGICO

2.1 Metodología de la investigación

El presente proyecto está pensado como un modelo de servicio en la nube, el mismo realiza un proceso de monitoreo y detección de forma no intrusiva, es decir, sin necesidad de acceder directamente a la aplicación web o al servidor. Por tanto, se descarta la sección de la caracterización, no existe una empresa o institución que sea beneficiario directo o preste parte de su infraestructura para la realización del proyecto.

Por tanto, es importante estimar el contexto de la solución planteada, se requiere del monitoreo constante de sitios web en internet, de modo, que se necesita considerar varios puntos como: disponibilidad de la infraestructura, capacidad de procesamiento del servidor, conectividad a internet y latencia de la red. Al hacer un análisis de los ítems mencionados previamente, una opción viable para hospedar el sistema es la nube como tal. Independientemente del proveedor, que se elija, el tener un hosting con un proveedor de servicios va a permitir disponer principalmente de una menor latencia al momento de realizar las consultas a los sitios web y una alta disponibilidad de la infraestructura gracias al Acuerdo de Nivel de Servicio (SLA) que brindan los proveedores, el cual, no es menor al 99.95% (Cloudorado, 2020).

Dada la naturaleza del presente trabajo, es importante la definición de la metodología de investigación, para ello, se consideró el método deductivo que expone partir de una premisa general para llegar a obtener las conclusiones lógicas y válidas de un caso particular. Dicho de otra forma, las conclusiones resultantes de un razonamiento están dadas de antemano en las ideas propuestas, por lo tanto, se requiere de un análisis para llegar a conocer el resultado (Estela, 2020). Adicionalmente el método deductivo es práctico cuando se dificulta el observar e identificar las causas de un determinado fenómeno, pero si las consecuencias resultantes del mismo.

De acuerdo con el proceso formal y el concepto expuesto del método deductivo, el presente proyecto, se acopla al mismo dado que plantea una idea a defender, en la

cual, se sostiene que un Sistema de Monitoreo y Detección de *Defacement* en sitios web permite reducir el tiempo de afectación a la disponibilidad y reputación de la imagen en sitios web de empresas e instituciones afectadas por este tipo de ataques. No es posible conocer las causas que ocasionan el ataque, pero si, se aprecia la desfiguración del sitio web como consecuencia.

El método deductivo, se aplica de forma directa puesto que solo se dispone de una premisa. Se conoce que los sitios y aplicaciones web, independientemente de su contexto, están conformados por varios elementos de códigos y contenido multimedia que permiten generar las páginas web, que se visualizan a través del navegador para la interacción con el usuario final. El análisis de estos elementos de forma periódica permite llevar un control de cambios en donde, se requiere definir un umbral de variabilidad aceptable de forma, que se logre identificar cuando un cambio sustancial es llevado a cabo, resultante de un *Defacement*, y de esta forma alertar a los administradores de los sitios para que tomen acción inmediata y de esta forma reducir el tiempo de afectación, a la cual, se ven expuestos.

Por otro lado, el presente proyecto, también, analiza una metodología cuasi experimental, se argumenta que los diseños cuasi experimentales tienen el mismo propósito que los estudios experimentales: probar una relación causal. Cuando una asignación aleatoria es imposible, los cuasiexperimentos permiten estimar los impactos del programa, y depdenden de si llega a establecer una base de comparación adecuada (Bono Cabré, 2012).

Dada la naturaleza del planteamiento del proyecto, se buscan obtener datos de sitios con diferente gestión y enfoque, estos son pero, no se limitan a páginas de: universidades, sitios de entretenimiento y de gobierno; se busca que arrojen variedad en los resultados. Por ello, se hace uso del método cuasi experimental, no existe un grupo de control y tampoco, se manipulan las situaciones a las que son sometidos los sitios web. Se plantean varios escenarios de experimentación en donde, se ajuste el umbral de cambios aceptable de forma que permita estimar la menor cantidad de falsos positivos.

Para la recolección de información son importantes las herramientas de software mismas que apoyan el proceso de recopilación de todos los elementos del sitio sin la intervención de un administrador o usuario y de una forma no intrusiva, es decir, de forma automatizada, se obtienen los mismos elementos que son requeridos para mostrar la página web. Una vez que la información se recolecta, la misma es almacenada y resguardada, para ello, se hace uso de bases de datos que permiten mantener la información de forma estructurada y puede ser consultada, modificada o eliminada en medida de las necesidades, que se requieran.

2.2 Metodología de Desarrollo

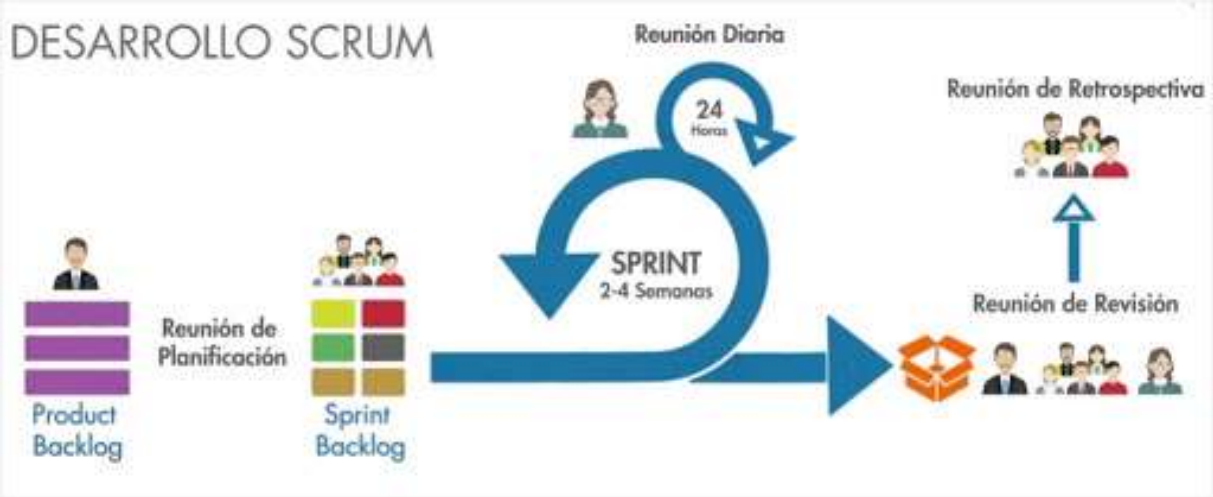
Para llevar a cabo el presente proyecto es importante disponer de un control y gestión de actividades o tareas, que se ejecuten a cabo, mismas que permitan cumplir los objetivos planteados. Para ello, se efectúa un breve análisis de las metodologías *Scrum* y *Kanban* en donde se consideran: practicidad, aplicabilidad y adaptabilidad.

El análisis empieza con *Scrum*, según Ramos (2017) “Scrum, más allá de una metodología aplicada al desarrollo de software, es un marco de trabajo y un modelo organizativo que aplica a cualquier actividad dentro de una compañía, independientemente del sector al que pertenezca y sea cual sea la naturaleza del producto o servicio que genere dicha actividad” (párr 1). Este modelo trata de un marco de trabajo empírico en donde su enfoque es iterativo e incremental, lo cual, permite optimizar la predictibilidad y el control del riesgo en el desarrollo del proyecto. Se basa en tres principios, los cuales, buscan mantener la transparencia, inspección y adaptación.

Como un modelo organizativo, se busca aprovechar la agilidad en sus tres dimensiones, facilidad de estructuras, procesos eficientes y una cultura colaborativa. Por otro lado, como un *framework* o marco de trabajo busca conformar equipos que realizan sus actividades en un proceso cíclico de delimitada duración. En *Scrum* intervienen varios roles (*Product Owner, Scrum Mater, Developer Team*), elementos (*Product Backlog, Sprint Backlog e* Incremento) y se prescriben equipos multifuncionales. Se realizan reuniones concretas de tiempo fijo, de igual manera, se

definen intervalos de tiempo para los entregables (*Sprints*). *Scrum* busca entregar un valioso incremento del producto al final de cada *Sprint,* por lo cual, es importante el compromiso de los desarrolladores para que lleguen a cumplir los objetivos en los plazos pactados.

Gráfico 4. Metodología *SCRUM*



Fuente: tomado a partir de OTEIC (2019)

Por otro lado Julio Roche (2020) de la firma Deloitte menciona que el método Kanban "se basa en el desarrollo y entrega continuos, se aborda un pequeño número de tareas de forma fluida y simultánea. Los equipos Kanban utilizan una herramienta de planificación visual, el tablero Kanban, que muestra cada proyecto (historia del usuario) en una tarjeta y mueve esas tarjetas a través de columnas que representan etapas progresivas de finalización. Si el equipo tiene un flujo continuo de solicitudes de trabajo, el método Kanban es el adecuado para gestionarlo. Por ejemplo, muchos equipos de soporte y mantenimiento usan Kanban en vez de Scrum, con este último la tarea de gestión de su trabajo es hace casi imposible".

Al igual que *Scrum*, *Kanban* es un modelo ágil, que se enfoca en: el objetivo, la optimización de medios, la entrega frecuente de valor sumado a la mejora continua. Proporciona transparencia del proceso y las actividades a través del tablero. No maneja roles y gestiona el trabajo en progreso (WIP, *Work in Progress*) por el estado del flujo de trabajo. Es abierta al cambio, ocurre en cualquier momento en función de

las necesidades y no requiere necesariamente de equipos multidisciplinares, manejan equipos formados por especialistas para la resolución de problemas (Abraham Requena, 2018). Dado que *Kanban* está enfocado en el flujo del proceso, se consigue una regularidad de trabajo y ritmo de entrega fluido, de esta forma, se evita el caos y los impedimentos o cuellos de botella que son identificados rápidamente para ser tratados.

Gráfico 5. Método *Kanban*

Fuente: tomado a partir de Guillermo Rodríguez (2017)

Kanban brinda beneficios tanto para grupos de equipos ágiles como equipos de gestión tradicional gracias a su flexibilidad y adaptación a la gestión del cambio, permite, también, de un solo vistazo identificar claramente el estado del trabajo y las actividades que detienen el flujo de trabajo. Cabe recalcar que las fases del trabajo, se acoplan a cada proyecto en base a las necesidades. Adicionalmente, existe Protokanban que conlleva los principios de *Kanban*, pero no necesariamente adopta todas sus prácticas para evitar convertirlo en un sistema *pull* cerrado, sin embargo, es un sistema, en el cual, fluye el trabajo y se gestionan las tareas o actividades.

Tras conocer brevemente el enfoque y cualidades de cada metodología, se establece un análisis comparativo en la tabla 2, el cual, permita dejar en claro las características de cada una y llegar a la conclusión de que metodología, se acopla de mejor manera al proyecto que planteado, se consideran los: integrantes, enfoque, actividades y desarrollo de la solución.

Tabla 2. Comparativa *SCRUM - KANBAN*

	SCRUM	***KANBAN***
Orientación	Maximiza el valor entregado	Optimiza el flujo de trabajo
Roles	*Product Owner, Scrum Mater, Developer Team*	No define roles
Iteraciones	*Sprints*	Trabajo continuo
Equipos	Multidisciplinares	Especialistas
WIP	Limita el WIP por iteración	Limita el WIP por estado del flujo de trabajo
Reuniones	Periódicas y de tiempo fijo	No se tiene reuniones prefijadas

Fuente: Elaboración propia

Dadas las implicaciones del proyecto y las características tanto de *Scrum* y *Kanban,* se listan las siguientes consideraciones:

- El desarrollo del presente trabajo requiere una orientación que permita optimizar el flujo del trabajo, de forma, que se llegue a cumplir los objetivos. No es requerida la entrega de valor constante.
- El proyecto consta con un solo investigador y el tutor correspondiente, por consiguiente, no se considera *Scrum* dados los roles que intervienen en dicha metodología.
- Para la ejecución de las iteraciones no son requeridas las limitaciones de tiempo de forma estricta, se prioriza el trabajo continuo.

- La conformación de equipos multidisciplinares no es requerida dado el enfoque del proyecto.
- La limitación del WIP, se da por la cantidad de tareas dentro del flujo de trabajo
- Finalmente, las reuniones periódicas no son consideradas por el investigador. Se plantean reuniones de seguimiento con el tutor a medida, que se avanza con el proyecto.

Como se observa, al analizar los diferentes puntos de comparación *Kanban* ofrece mayor flexibilidad y se acopla de mejor manera al presente trabajo y sus consideraciones. Una vez seleccionada la metodología de desarrollo, se requiere definir las fases y el conjunto de actividades a incluir en la ejecución de la solución.

Para el desarrollo de la metodología, se proponen las siguientes fases de flujo de trabajo o estados, mismas que permiten llevar un control adecuado de cada una de las actividades:

- Solicitadas
- En progreso
    - Análisis/Diseño
    - Desarrollo
    - Pruebas
    - Implementación
- Realizadas

Una de las políticas, que se consideran dadas las características del trabajo es manejar un límite de trabajo en proceso de dos actividades por cada fase, esto al considerar que de acuerdo con la metodología no siempre implica que ese espacio acordado son directamente una tarea sino un espacio máximo al, que se lleven para controlar la multitarea. A continuación, se listan las actividades principales, que se consideran llevar a cabo, mismas que están sujetas a cambios ya sean: eliminación, modificación o nuevas tareas. De igual manera, se cambia la priorización de las mismas gracias a la metodología *Kanban* de ser requerido:

- Sistema de monitoreo y detección
    - Preprocesamiento
        - Recolección
        - Limpieza de datos
        - Extracción de características
    - Entrenamiento
    - Aplicación Web
    - *Script* de Monitoreo y Detección

Para desarrollar la metodología, se hace uso de la herramienta de software Kanbanize, la misma permite apreciar la planificación de acuerdo con las fases y actividades mencionadas anteriormente de una forma visual a través del tablero *Kanban*. De momento, se requiere mover la primera actividad a la fase de Análisis/Diseño para el modelo de monitoreo y detección:

Gráfico 6. Metodología *Kanban*

Fuente: Elaboración propia

2.3 Sistema de Monitoreo y Detección

Actualmente existen varios métodos y propuestas que buscan identificar cambios realizados en sitios web con el fin de alertar a los administradores de un posible ataque de *Defacement* (Hoang & Nguyen, 2019). Estas soluciones pueden dividirse en dos categorías: las que son basadas en métodos tradicionales y métodos avanzados.

Al hablar de las soluciones basadas en métodos tradicionales, se toma en consideración la comparación a través de técnicas de *checksum* en donde la página web es descargada y se genera un valor *hash* mismo que es almacenado para

comparaciones futuras. Por otro lado, se dispone de la herramienta *diff* que es una utilidad que viene incluida en los sistemas operativos *GNU Linux* y realiza una comparación por líneas entre un archivo y otro. Finalmente, se dispone del análisis del árbol *DOM* y es usado para identificar cambios en la estructura de una página web, si se detecta una diferencia notable, una alarma es generada y enviada al administrador del sitio. Estás técnicas funcionan bastante bien en sitios estáticos, sin embargo, no son aplicables a sitios dinámicos.

Por otro lado, se dispone de los métodos avanzados mismos, que se basan en técnicas estadísticas, de minería de datos, *machine learning*, programación genética y análisis de computación visual. Estos métodos incluyen un alto procesamiento de cómputo para estimar la detección y sus modelos.

El estudio de Kim et al. (2006) propone el uso de un método estadístico y consiste en una fase de entrenamiento y una fase de detección. El perfil de detección contiene todas las páginas web normales del conjunto de datos de entrenamiento, cada una, de las cuales, es transferida a un vector de bigramas y sus frecuencias de ocurrencia. Para la detección, se descarga la página HTML, se procesa y convierte en un vector con las mismas técnicas, que se usaron para el entrenamiento, a continuación, el vector de la página supervisada se compara con el vector almacenado y se utiliza la distancia del coseno para calcular la similitud, si esta es menor que el umbral, se activa una alarma. El problema con este modelo son los ajustes de umbrales, que se realizan periódicamente por lo que cambia con frecuencia y genera falsos positivos, adicionalmente implica un alto consumo de recursos.

Medvet et al. (2007), por otro lado, propone construir el perfil de detección con técnicas de programación genética. Primero con el fin de recopilar datos, hace uso de 43 sensores para monitorear y extraer información de las páginas web. El siguiente paso implica el proceso de vectorización donde la información obtenida de cada sitio es transformada en un vector de 1446 elementos. El método propuesto incluye una fase de entrenamiento y una fase de detección. En la fase de entrenamiento, se obtiene información de los sitios web que no han sufrido ataques, esta información es

vectorizada para construir el perfil de detección basado en técnicas de programación genética. En la fase de detección, se recupera la información de la página web monitoreada, se vectoriza y entonces, se compara con el perfil de detección levantado previamente para encontrar diferencias. Si alguna diferencia significativa es identificada, se genera una alarma. Los principales inconvenientes de este modelo son las altas demandas de recursos para la construcción del perfil debido a la cantidad de vectores resultantes y las técnicas de programación genética.

El trabajo realizado por Borgolte et al. (2015) para la detección de *Defacement* en sitios y aplicaciones web dio como resultado el sistema bautizado como *Meerkat* y se basa en el análisis y reconocimiento de imágenes de capturas de pantalla de los sitios web a través de técnicas de visión por computadora. La arquitectura de *Meerkat* está basada en redes neuronales profundas. Para cada página el sistema carga la web y toma un *screenshot* y se utilizan como entradas al sistema para el análisis y detección de alteraciones. De forma similar a los sistemas mencionados anteriormente, Meerkat, también, dispone de una fase de entramiento y una fase de detección. En la fase de entrenamiento recopila las capturas de pantalla de sitios web supervisados que funcionan normalmente y son procesadas para extraer funciones de alto nivel a través de métodos avanzados de *machine learning* como el codificador automático aplicado y redes neuronales profundas. El conjunto de características de las páginas web es almacenado en el perfil de detección y el mismo procedimiento es utilizado con cada página web monitoreada para crear el conjunto actual de

características. Posteriormente es comparado y si se encuentra alguna diferencia notable, se dispara una alarma. Este sistema por su naturaleza requiere de altos recursos computaciones y adicionalmente puede tornarse un poco lento sumado a, que se esperaría que el sitio web cargue por completo para tomar la captura de pantalla.

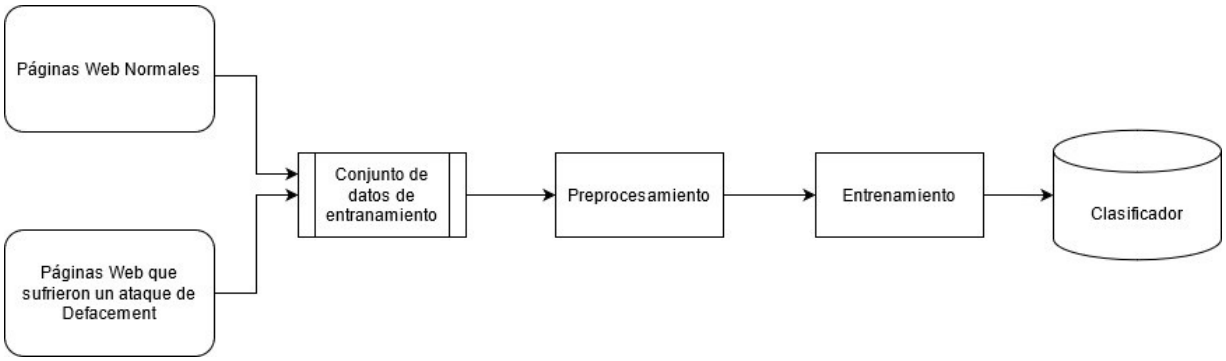
El modelo planteado por Hoang & Nguyen (2019) llama la atención, en primer lugar no es un método intrusivo, es decir, no requiere de la instalación de ningún tipo de agente en el servidor web y en segundo lugar propone un sistema basado en *machine learning*

pero no necesita de altos recursos computacionales. Este modelo busca obtener un conjunto de datos que incluyen sitios web en un estado normal y sitios web que hayan sido desfigurados. Este conjunto de datos es preprocesado y se extraen características a través de n-gramas y términos de frecuencia. Paso seguido, se toma parte del conjunto de datos y se entrenan en un algoritmo supervisado de *machine learning* para generar un clasificador que permite identificar en un futuro si un sitio ha sufrido un ataque de *Defacement* o no.

De acuerdo a los diferentes trabajos expuestos anteriormente, la propuesta de Hoang & Nguyen (2019) es tomada como base dado que no es intrusivo, no necesita excesivos recursos en el proceso de detección y la extracción de características propone un nuevo paradigma que lo hace interesante para desarrollar. Este modelo tiene dos fases: la fase de entrenamiento y la fase de detección.

Fase de entrenamiento

Gráfico 7. Fase de entrenamiento del modelo propuesto



Fuente: modificado a partir de Hoang & Nguyen (2019)

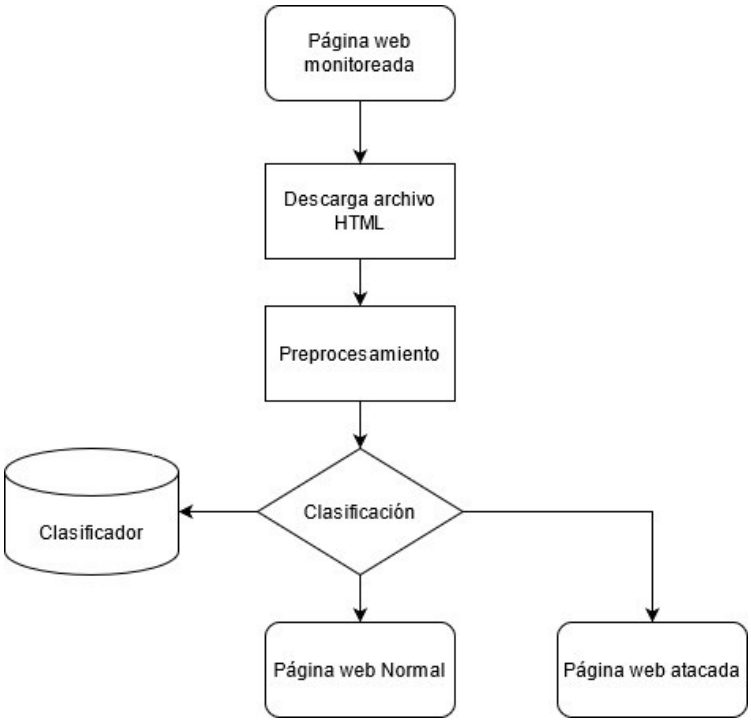
1. Recolección de conjuntos de datos para el entrenamiento y pruebas:
    a. Archivos HTML de diferentes sitios web como: e-commerce, universidades, gobierno, hoteles, etc. Se controla que no existan archivos duplicados.
    b. Archivos HTML de sitios que han sufrido un ataque de *Defacement*, estos, se descargan del sitio Zone-h.com. De igual manera, se controla

que no existan duplicados en el conjunto de datos para el entrenamiento y pruebas.

2. El conjunto de datos de entrenamiento (archivos HTML) son preprocesados para extraer sus características con la técnica de n-gramas para posteriormente vectorizarlos y aprovechar su término de frecuencia.
3. Como paso siguiente, se usa parte del conjunto de datos para entrenar el algoritmo de *random forest* y construir el modelo de clasificación.

Fase de detección

Gráfico 8. Fase de detección



Fuente: modificado a partir de Hoang & Nguyen (2019)

1. Descarga del código HTML de la página web monitoreada, esto es realizado a través de la url proporcionada.
2. Validación del código HTML de la página supervisada, el código, se procesa previamente para la extracción de características, posteriormente, se clasifica con el uso del modelo integrado, el cual, indica si la página está desfigurada y de ser así, se envía una notificación al administrador del sitio.

Si bien en los pasos listados en la fase de entrenamiento, el preprocesamiento, se encuentra inmerso. Es importante el tiempo y el esfuerzo que conlleva esta etapa, por lo cual, se trata como un ítem por separado. Adicionalmente, se crea un sistema web que permita a un usuario registrarse y monitorear un sitio web con uso del modelo clasificador a desarrollar. Así como un *script* para automatizar el monitoreo.

Es importante mencionar que todo el sistema es llevado a cabo en *Python*.

2.3.1 Preprocesamiento

Para crear el modelo clasificador es necesario alimentar el algoritmo con los datos adecuados de forma, que se distinga entre un sitio normal y un sitio que ha sido desfigurado. Para garantizar la calidad de los datos, se ejecutó una validación visual por parte del investigador en donde, se comprobó que los diferentes sitios cargaron normalmente y se exceptuaron aquellos sitios que presentaban algún tipo de error.

Esta actividad fue llevada a cabo entre las fechas: 01-03-2021 y 08-03-2021, se recolectó 997 sitios normales y 978 sitios desfigurados. En este proceso, para los sitios web en normal funcionamiento, se consideró que los mismos sean de distinta naturaleza y de diferentes países con el fin de tener variedad en los datos. Es así como, se incluyeron páginas de: universidades, noticias, comercio en línea, bienes raíces, hoteles, entretenimiento, etc. de países como: Ecuador, México, Argentina, Chile, Venezuela, España, Colombia, Honduras, Perú, Uruguay y Bolivia.

Las direcciones *url* de cada sitio fueron recolectadas en un archivo *csv* denominado Sitios para posteriormente descargar los archivos *index* a través de la utilidad *wget* de *GNU Linux* ,en donde, se usaron los siguientes parámetros:

wget --no-check-certificate -c -i Sitios.csv -o log.txt

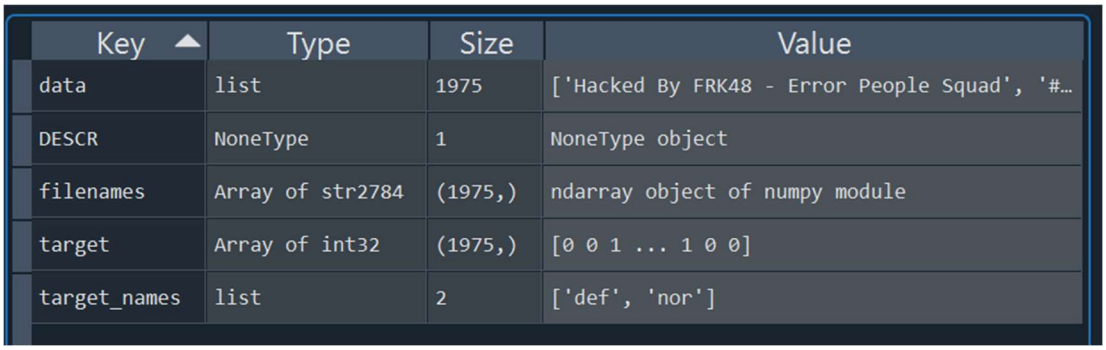
- --no-*check-certificate* Omite la validación del certificado ssl
- -c Se reanuda la descarga si llega a interrumpirse.
- -i Se especifica el archivo csv con el listado de sitios
- -o genera un *log*

Para la obtención de las páginas que fueron desfiguradas, se revisó la base de datos de Zone-h.com, este sitio mantiene un repositorio de sitios reportados que han sufrido un *Defacement*. Es importante recalcar, que para cada página, se realizó una validación visual por parte del investigador y adicionalmente, se controló que no existan más de 5 sitios por cada *defacer* (responsable de haber desfigurado un sitio) en el archivo *csv* con el fin de evitar datos duplicados. Los parámetros usados con el comando *wget* fueron los mismos, que se usaron en la descarga de los sitios web normales.

Una vez con el conjunto de datos recolectados, es necesario importar esos archivos y asignarles una clase o etiqueta. Este término es propio de *Machine Learning* y para el presente proyecto permite conocer a qué clase pertenece un archivo. Dicho de otra forma, se marcan los datos para identificarlos con 0 para sitios con *Defacement* y con 1 para sitios normales.

El proceso de importación de datos conjuntamente con las clases, se realiza gracias a la función *load_files* de la biblioteca *sklearn.datasets*. Dicha función permite asignar las clases a directorios, de esta forma, se tenían los archivos separados en dos carpetas: def y nor (desfigurados y normales). Una vez cargados los datos, se tiene un objeto de tipo diccionario que dispone de varios campos, los que interesan para el tratamiento de los datos son: *data* (los archivos HTML) y *target* (las clases de cada archivo)

Gráfico 9. Carga de los datos

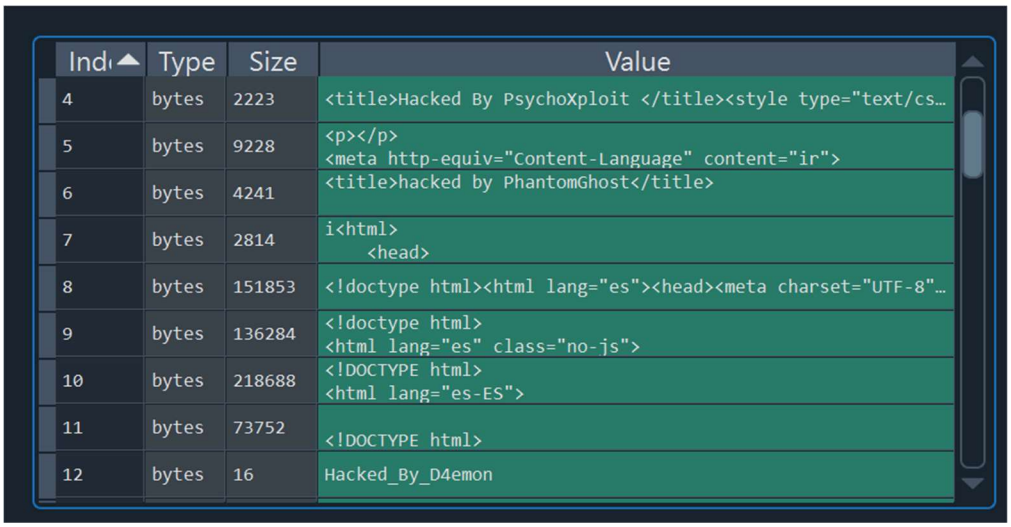


Key	Type	Size	Value
data	list	1975	['Hacked By FRK48 - Error People Squad', '#…
DESCR	NoneType	1	NoneType object
filenames	Array of str2784	(1975,)	ndarray object of numpy module
target	Array of int32	(1975,)	[0 0 1 ... 1 0 0]
target_names	list	2	['def', 'nor']

Fuente: Elaboración propia

Para una mejor manipulación de los datos cargados, se almacenan los campos data y target en archivos *pickle*. Estos son archivos binarios propios de Python y la carga de los mismos es mucho más rápida por lo que son usados en el proyecto. En este punto es importante mencionar que cada archivo HTML, se encuentra almacenado en un objeto de tipo lista como se observa en el gráfico 10:

Gráfico 10. Conjunto de datos almacenados en lista

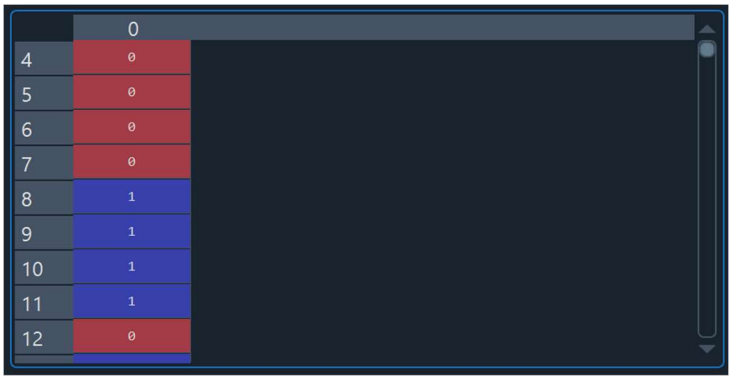


Ind	Type	Size	Value
4	bytes	2223	<title>Hacked By PsychoXploit </title><style type="text/cs…
5	bytes	9228	 <meta http-equiv="Content-Language" content="ir">
6	bytes	4241	<title>hacked by PhantomGhost</title>
7	bytes	2814	i<html> <head>
8	bytes	151853	<!doctype html><html lang="es"><head><meta charset="UTF-8"…
9	bytes	136284	<!doctype html> <html lang="es" class="no-js">
10	bytes	218688	<!DOCTYPE html> <html lang="es-ES">
11	bytes	73752	<!DOCTYPE html>
12	bytes	16	Hacked_By_D4emon

Fuente: Elaboración propia

Al igual que los datos, las clases, se encuentran almacenadas en otro objeto de tipo lista como se visualiza en el gráfico 11. Es importante recalcar que las dos listas, se encuentran relacionadas, comparten el mismo índice.

Gráfico 11. Clases correspondiente a cada dato



Fuente: Elaboración propia

La siguiente tarea consiste en realizar la limpieza de datos, esto significa eliminar todo el texto o caracteres que no vayan a aportar valor al conjunto de datos. Para el presente trabajo, se dispone de archivos HTML y se requiere obtener el texto o mensaje de cada archivo de forma, que se omitan las etiquetas de formato HTML. Para ello *Python* dispone de la biblioteca *BeautifulSoup* que permite manipular los campos HTML o a su vez obtener el texto directamente de un archivo o una dirección *url* dada, por lo cual, fue utilizada en el proyecto.

Con el texto extraído de cada página, se requiere normalizarlo, dicho de otra forma, es preciso eliminar caracteres especiales como: espacios en blanco, saltos de línea, eliminar tildes y unigramas de caracteres con el fin de obtener datos limpios que son utilizados en la siguiente fase. Para esta actividad, se usaron expresiones regulares.

Tabla 3. Normalización del texto

Texto sin normalizar	Texto normalizado
CEDIA INICIO Conócenos SER MIEMBRO DE CEDIA HISTORIA SOBRE NOSOTROS NUESTROS MIEMBROS ALIANZAS PAQUETES DE SERVICIOS ALINEAMIENTO ODS ESTATUTOS	cedia inicio conocenos ser miembro de cedia historia sobre nosotros nuestros miembros alianzas paquetes de servicios alineamiento ods estatutos

Fuente: Elaboración propia

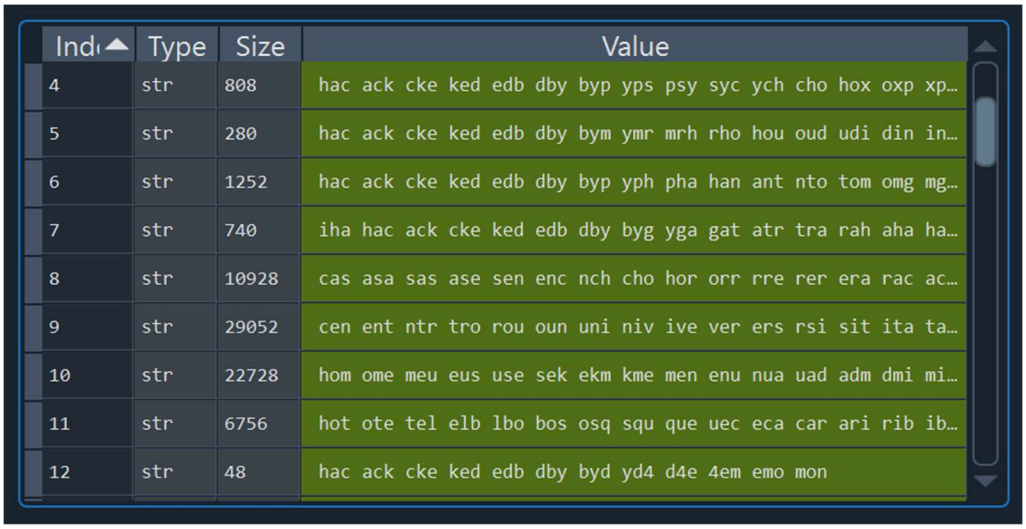
Una vez, que se dispone del texto limpio, el paso siguiente consiste en tokenizar (segmentar) y extraer las características que permitan entrenar el modelo.

La tokenización viene del mundo del Procesamiento del Lenguaje Natural (NLP por sus siglas en inglés) que de acuerdo con Recuero de los Santos (2020) es una disciplina bastante amplia, que se enfoca en el desarrollo de aplicaciones en ámbitos como la inteligencia artificial y el *machine learning*. Es de gran ayuda al trabajar con texto en crudo, ya sean datos semiestructurados o datos no estructurados. Permite segmentar o fragmentar el texto en oraciones, palabras o gramas para posteriormente identificar la frecuencia de aparición de cada fragmento en un cuerpo de datos.

El trabajo de Kim et al (2006), propone un proceso de tokenización basado en n-gramas de caracteres, que se consideran como pequeñas agrupaciones de caracteres en dependencia del número N que sea dado. Adicionalmente, se apoya en la ley de *Zipf* que postula que en un determinado lenguaje siempre va a existir un conjunto de palabras dominantes o que tengan mayor presencia que otras. Con esto en conocimiento, se usan trigramas de caracteres para dividir el texto normalizado en el presente proyecto. Por ejemplo, la palabra texto, se segmentaría en: tex ext xto de esta forma, se obtienen tres trigramas resultantes.

Al dividir todo el cuerpo de datos en trigramas de caracteres, se rompe la semántica que existe del lenguaje correspondiente, que es el español para este caso. Esto construye de alguna forma un nuevo tipo de lenguaje que no es legible para los humanos, sin embargo, va a permitir encontrar patrones que permitan identificar si un sitio ha sido desfigurado o no. Estos trigramas son considerados como las características del modelo de datos.

Gráfico 12. trigramas de caracteres



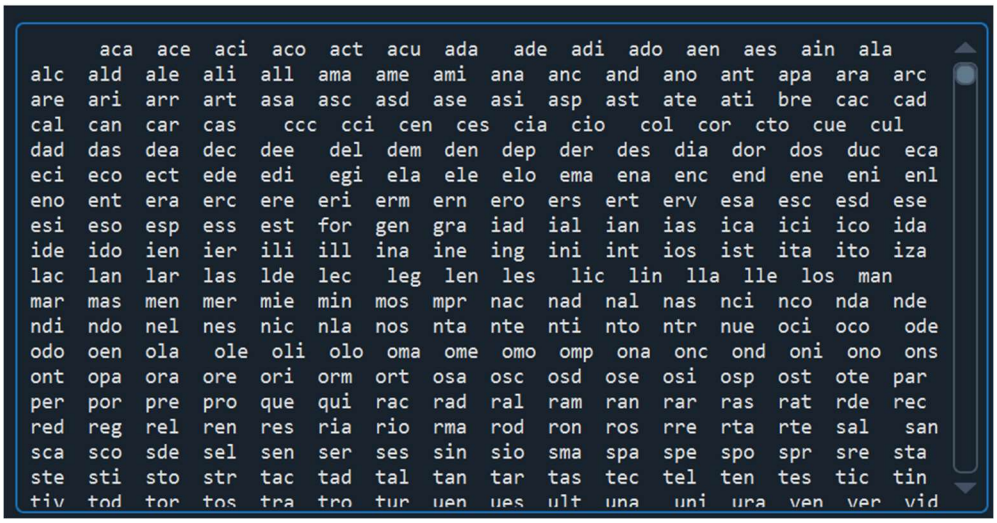
Ind	Type	Size	Value
4	str	808	hac ack cke ked edb dby byp yps psy syc ych cho hox oxp xp…
5	str	280	hac ack cke ked edb dby bym ymr mrh rho hou oud udi din in…
6	str	1252	hac ack cke ked edb dby byp yph pha han ant nto tom omg mg…
7	str	740	iha hac ack cke ked edb dby byg yga gat atr tra rah aha ha…
8	str	10928	cas asa sas ase sen enc nch cho hor orr rre rer era rac ac…
9	str	29052	cen ent ntr tro rou oun uni niv ive ver ers rsi sit ita ta…
10	str	22728	hom ome meu eus use sek ekm kme men enu nua uad adm dmi mi…
11	str	6756	hot ote tel elb lbo bos osq squ que uec eca car ari rib ib…
12	str	48	hac ack cke ked edb dby byd yd4 d4e 4em emo mon

Fuente: Elaboración propia

Como paso siguiente, se tiene que transformar el conjunto de datos en valores numéricos para que el modelo de *machine learning* pueda interpretarlo, actualmente solo, se dispone de los trigramas formados previamente.

Para ello, se genera un modelo denominado *Bag of Words* gracias a la librería *sklearn* de *Python* que cuenta con el método *CountVectorizer* en donde, se identifican las palabras (trigramas para este estudio) con mayor frecuencia de aparición en cada uno de los archivos HTML normalizados. Se toman solo los 300 trigramas con mayor frecuencia de aparición en cada archivo, en esto proceso, se definen algunos parámetros de búsqueda que son el menor y mayor término de frecuencia; en donde los trigramas con un término de frecuencia menor o igual a 3 son excluidos, es decir, aquellos que aparecen en menos de 3 archivos. Por otro lado, los trigramas que tengan un porcentaje de aparición mayor o igual al 60% en los documentos, también, son desechados. De esta forma la búsqueda, se centra en los trigramas relevantes dentro del cuerpo de datos que sirvan como características en el modelo de *machine learning*.

Gráfico 13. características del modelo basado en trigramas



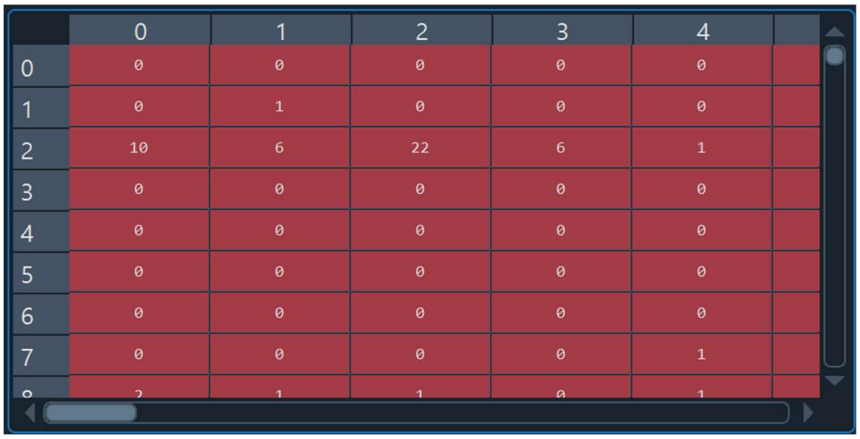
```
      aca  ace  aci  aco  act  acu  ada   ade  adi  ado  aen  aes  ain  ala
alc  ald  ale  ali  all  ama  ame  ami  ana  anc  and  ano  ant  apa  ara  arc
are  ari  arr  art  asa  asc  asd  ase  asi  asp  ast  ate  ati  bre  cac  cad
cal  can  car  cas    ccc  cci  cen  ces  cia  cio   col  cor  cto  cue  cul
dad  das  dea  dec  dee   del  dem  den  dep  der  des  dia  dor  dos  duc  eca
eci  eco  ect  ede  edi   egi  ela  ele  elo  ema  ena  enc  end  ene  eni  enl
eno  ent  era  erc  ere  eri  erm  ern  ero  ers  ert  erv  esa  esc  esd  ese
esi  eso  esp  ess  est  for  gen  gra  iad  ial  ian  ias  ica  ici  ico  ida
ide  ido  ien  ier  ili  ill  ina  ine  ing  ini  int  ios  ist  ita  ito  iza
lac  lan  lar  las  lde  lec   leg  len  les   lic  lin  lla  lle  los  man
mar  mas  men  mer  mie  min  mos  mpr  nac  nad  nal  nas  nci  nco  nda  nde
ndi  ndo  nel  nes  nic  nla  nos  nta  nte  nti  nto  ntr  nue  oci  oco   ode
odo  oen  ola   ole  oli  olo  oma  ome  omo  omp  ona  onc  ond  oni  ono  ons
ont  opa  ora  ore  ori  orm  ort  osa  osc  osd  ose  osi  osp  ost  ote  par
per  por  pre  pro  que  qui  rac  rad  ral  ram  ran  rar  ras  rat  rde  rec
red  reg  rel  ren  res  ria  rio  rma  rod  ron  ros  rre  rta  rte  sal   san
sca  sco  sde  sel  sen  ser  ses  sin  sio  sma  spa  spe  spo  spr  sre  sta
ste  sti  sto  str  tac  tad  tal  tan  tar  tas  tec  tel  ten  tes  tic  tin
tiv  tod  tor  tos  tra  tro  tur  uen  ues  ult  una   uni  ura  ven  ver  vid
```

Fuente: Elaboración propia

Con el resultado anterior, se crea una matriz en donde las filas corresponden a cada uno de los archivos HTML mientras que las columnas son los trigramas obtenidos. De esta forma, se valida si un determinado trigrama tuvo o no una o varias apariciones en cada uno de los archivos. Esto permite entrenar al modelo de *machine learning* para que identifique los trigramas más comunes que aparecen en textos de sitios web que han sufrido un *Defacement* o caso contrario aquellos, que se encuentran sin ninguna afectación.

Ahora, si bien, se dispone de una matriz en donde, se tiene la cantidad de apariciones de cada trigrama por cada archivo como se observa en el gráfico 14, este modelo no es tan eficiente, van a existir trigramas que tengan el mismo peso o importancia y al pasar estos datos al modelo de *machine learning* este puede llegarse a confundir en el procesamiento, varios trigramas tienen el mismo impacto en un documento. Se necesita dar más peso a los trigramas que permitan distar de otros con el fin de generar un modelo clasificador más eficiente.

Gráfico 14. Modelo *Bag of Words*



	0	1	2	3	4
0	0	0	0	0	0
1	0	1	0	0	0
2	10	6	22	6	1
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	1

Fuente: Elaboración propia

Para solventar la falta de eficiencia en el modelo *Bag of Words,* se pasa la matriz anterior a una nueva matriz con el uso del método TF-IDF (*Term Frecuency – Inverse Document Frecuency* por sus siglas en inglés) que es una fórmula matemática que ayuda a representar cuán importante es una palabra para un documento en una colección de documentos, de forma que la frecuencia inversa de documento atenúa el peso de los términos que ocurren con mucha frecuencia en la colección de documentos e incrementa el peso de los términos que ocurren pocas veces (Alvarez, 2017*)*.

A continuación, se exponen las fórmulas para el cálculo que son llevadas a cabo gracias al método *TfidfTransformer* de la biblioteca *sklearn* de *Python*

Cálculo TF

Para realizar el cálculo del término de frecuencia, se usa la siguiente fórmula:

$$tf(t,d) = \frac{f(t,d)}{max\{f(w,d) : w \in d\}}$$

En donde:

- $tf(t,d)$ es el término de frecuencia de cada n-grama t en un archivo HTML d
- $f(t,d)$ es el número de ocurrencias de un n-grama t en un archivo HTML d
- $max\{f(w,d) : w \in d\}$ es el número máximo de ocurrencias de cualquier n-grama t en un archivo HTML d

Calculo IDF

Para realizar el cálculo de la frecuencia inversa del documento, se usa la siguiente fórmula:

$$idf(t,D) = log\frac{N}{|\{d \in D : t \in d\}|+1}$$

En donde:

- $idf(t,D)$ es la frecuencia inversa del documento de cada n-grama t en todo el corpus D
- N es el número total de documentos en el cuerpo de datos
- $|\{d \in D : t \in d\}|$ es el número de documentos d donde el término t aparece

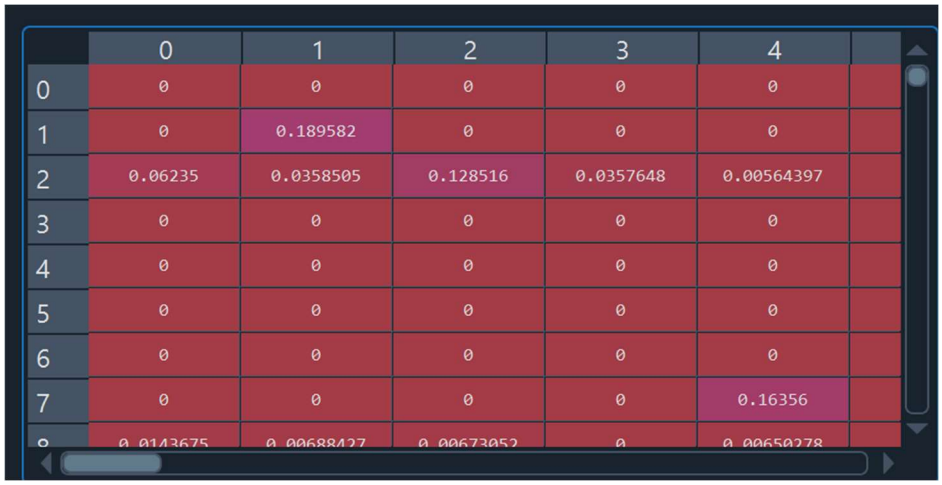
Cálculo TF-IDF

Con los valores anteriormente generados, ahora, se calcula el valor resultante de TF-IDF en lugar de la cantidad de ocurrencias, que se presentaban en el modelo *Bag of Words* con la multiplicación de TF por IDF:

$$tfidf(t,d,D) = tf(t,d) * idf\ (t,D)$$

La matriz resultante, se muestra, a continuación, en el gráfico 15:

Gráfico 15. Matriz TF-IDF



	0	1	2	3	4
0	0	0	0	0	0
1	0	0.189582	0	0	0
2	0.06235	0.0358505	0.128516	0.0357648	0.00564397
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0.16356
8	0.0143675	0.00688427	0.00673052	0	0.00650278

Fuente: Elaboración propia

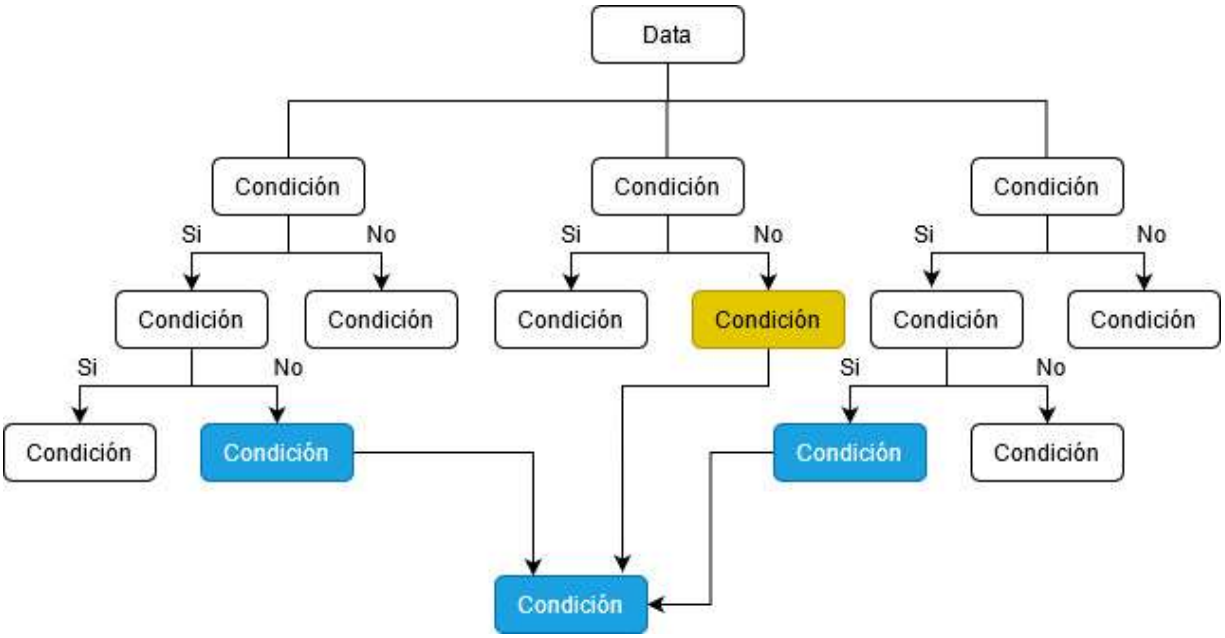
2.3.2 Entrenamiento

Una vez, que se cuenta con la matriz TF-IDF, es necesario separar los datos para entrenamiento y pruebas. Se toma el 80% de los datos para entrenar el modelo y el 20% restante es usado para probar el clasificador resultante. Esto aplica tanto para la matriz TD-IDF como para la lista que contiene las clases correspondientes. Es importante recalcar que estas siguen enlazadas a través de sus índices.

De acuerdo al trabajo de Hoang & Nguyen (2019) el algoritmo de clasificación que mejores resultados mostró en su modelo de clasificación fue *Random Forest* y se opta por este modelo para el presente trabajo. Este algoritmo supervisado para clasificación está basado en el algoritmo de árboles de decisión, pero a diferencia de este, se crean varios árboles y se seleccionan *k* características (columnas) de las *m* totales en donde *k* menor a *m*. Se crean *n* árboles y en donde siempre varía la cantidad de características *k.*

Cada uno de los árboles creados realiza un proceso de clasificación y se generan *n* resultados de forma que al final, se calculan los votos obtenidos para cada clase y se considera a la clase más votada como la clasificación final del bosque.

Gráfico 16. *Random Forest*



Fuente: tomado a partir de González (2018)

De acuerdo con Amat (2020), este algoritmo es uno de los referentes dentro de los clasificadores en *machine learning* debido a los buenos resultados que generan en diferentes casos de uso, para ello menciona algunas ventajas y desventajas:

Ventajas:

- Es capaz de seleccionar características de forma automática
- Al tratarse de métodos no paramétricos, no es necesario, que se cumpla ningún tipo de distribución específica.
- Son muy útiles en la exploración de datos, permiten identificar de forma rápida y eficiente las variables (características) más importantes
- Tienen buena escalabilidad, pueden aplicarse a conjuntos de datos con un elevado número de observaciones.

Desventajas:

- Al combinar múltiples árboles, se pierde la interpretabilidad que tienen los modelos basados en un único árbol.
- No son capaces de extrapolar fuera del rango de las características observadas en los datos de entrenamiento.

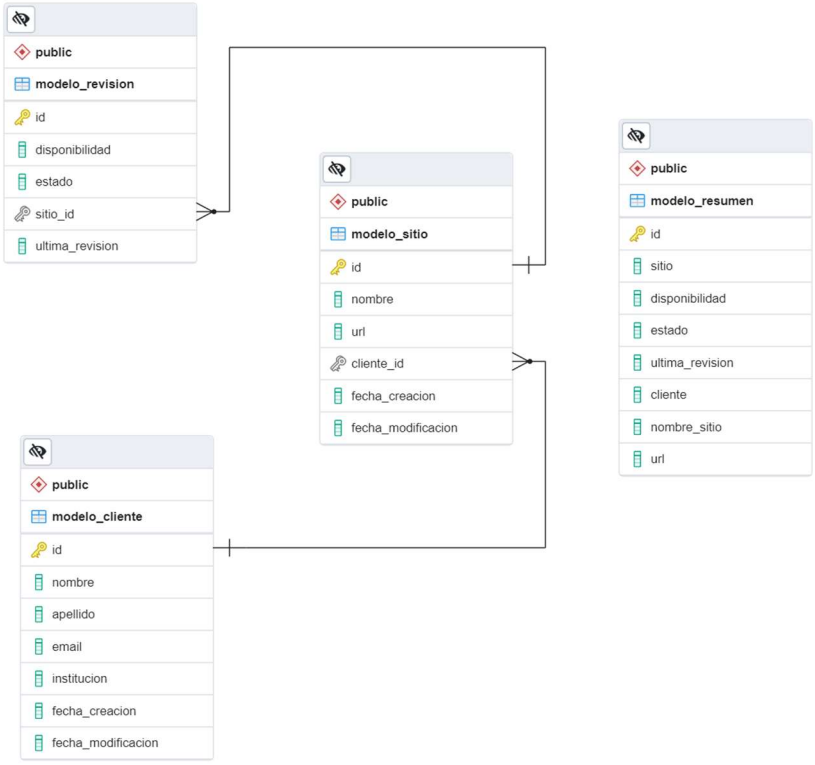
Para entrenar el modelo, *Python* dispone del método *RandomForestClassifier* que es parte de la biblioteca *sklearn.ensemble.* Se configura el algoritmo para que trabaje con 150 árboles y paso seguido, se pasan los datos y clases de entrenamiento. Se obtiene de esta forma el clasificador resultante.

2.3.3 Aplicación Web

Esta parte del proyecto complementa el desarrollo del modelo clasificador y proporciona un sitio web para la administración de clientes y sitios a monitorear de forma que permita visualizar de forma rápida el estado de los mismos, así como obtener un detalle de los últimos monitoreos realizados.

Para empezar a desarrollar el sitio, es necesario entender el modelo de base de datos. Se identifica que un cliente administra varios sitios y estos a su vez son monitoreados de forma constante. Por lo cual, se crean las tablas Cliente, Sitio y Revisión. Adicionalmente, para obtener un sumario del estado de los sitios con el último escaneo, se crea una tabla Resumen que almacene este resultado. Para este proceso, se creó un *trigger* en la base de datos que elimina los datos de esta tabla antes de la inserción de un nuevo registro en la tabla Revisión; de forma seguida, con la tabla en blanco, se insertan los datos del último escaneo.

Gráfico 17. Modelo de la Base de Datos



Fuente: Elaboración propia

El lenguaje de programación para el desarrollo del *backend* es *Django,* este es un framework basado en *Python* y usa el patrón MVT (Modelo-Vista-*Template*). Se optó por el mismo dada la experiencia adquirida con *Python* en la elaboración de la primera etapa del proyecto.

Django permite crear la base de datos a través de su capa de modelo, de forma tal que manipula las tablas como objetos gracias al ORM *(Object Relational Mapping*) y de esta forma el propio *framework,* se encarga de crear las sentencias CRUD (*Create-Read-Update-Delete*) para cada tabla. Se creó la app modelo dentro del proyecto y se crearon las clases correspondientes a cada tabla.

Gráfico 18. Creación de la tabla Cliente

```python
class Cliente(models.Model):
    nombre = models.CharField(max_length=25)
    apellido = models.CharField(max_length=25)
    email = models.EmailField(max_length=50)
    institucion = models.CharField(max_length=50)
    fecha_creacion = models.DateTimeField(auto_now_add=True)
    fecha_modificacion = models.DateTimeField(auto_now=True)
```

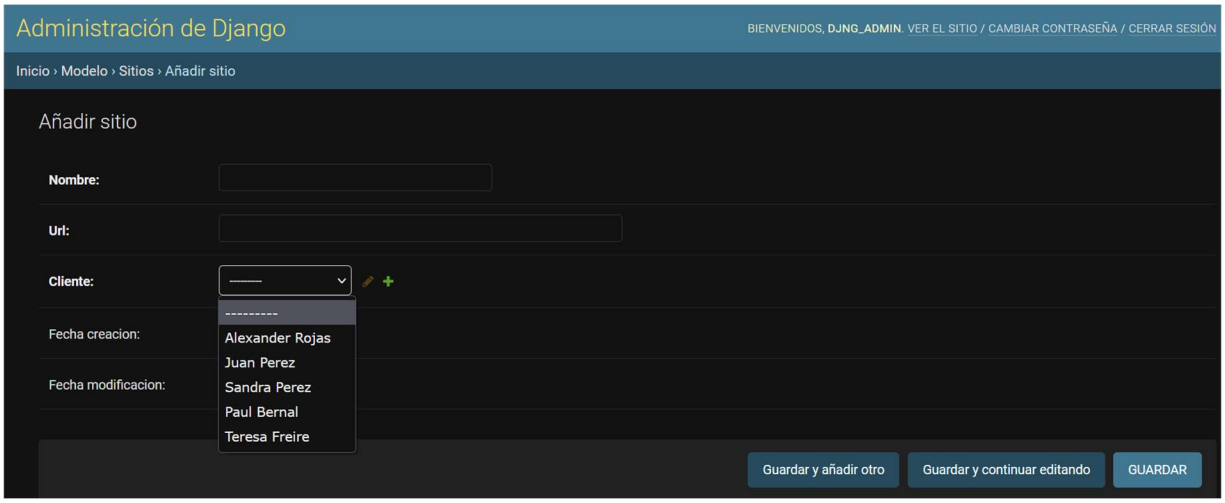
Fuente: Elaboración Propia

Django tiene varias ventajas, una de ellas es, que se encarga de la gestión de usuarios, sesiones, controles en el ingreso de los datos y búsquedas. Otra característica que ayuda mucho es el sitio de administración que dispone para el manejo de la capa modelo que haya sido declarada. Para el proyecto en particular, únicamente en las tablas Cliente y Sitio se crean, actualizan y eliminan registros de forma manual. Por ende, solo estas son manipuladas desde el sitio de administración.

Gráfico 19. *Django* Administración Clientes

Fuente: Elaboración propia

Gráfico 20. Creación de un nuevo sitio



Fuente: Elaboración propia

Al seguir el patrón de diseño MVT, es necesario crear las vistas, las mismas contienen los datos que se van a mostrar en la plantilla. Por ejemplo, para la página que contiene el resumen del último escaneo de los sitios monitoreados es necesario enviar los datos de la tabla Sitios, así como los datos de la tabla Resumen. Los datos son enviados a través de un diccionario en un método *request*.

Gráfico 21. Vista sitio

```
sitios(request):
no_sitios = Resumen.objects.count()
sitios = Sitio.objects.order_by("id")
resumen = Resumen.objects.order_by('id')

return render(request, "modelo/portafolio.html", {"no_sitios":no_sitios, "sitios":sitios, "resumen":resumen})
```

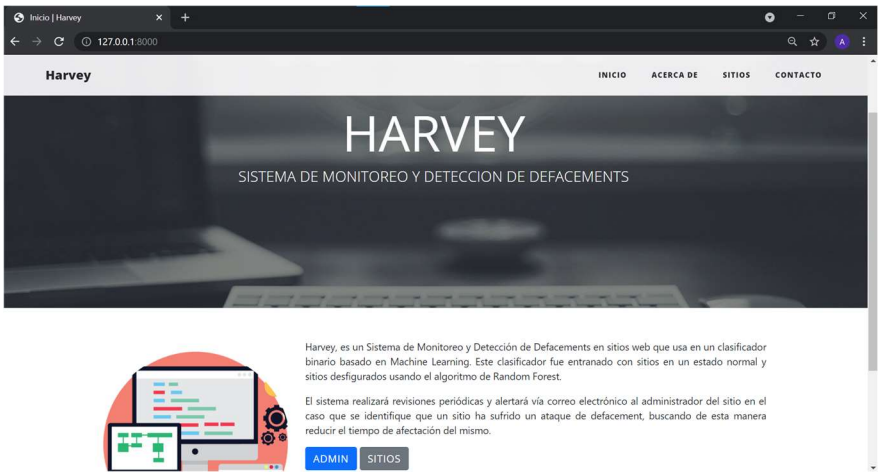
Fuente: Elaboración propia

Se crearon las vistas necesarias para cada una de las plantillas o páginas a mostrar. Se consideró que el sitio tendría la siguiente estructura: Inicio, Acerca de, Sitios y Contacto. A estas, se suma la página Detalle que muestra el estado actual de un sitio en particular, así como el resultado de los últimos escaneos.

Consecuentemente, se crearon las plantillas HTML, que se encargan de mostrar los datos enviados por las vistas. Cabe mencionar que para el maquetado del sitio, se usó *Bootstrap*.

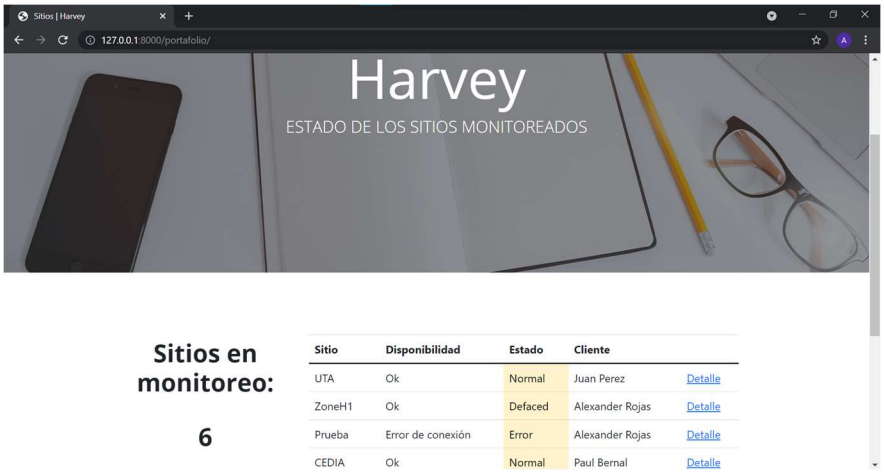
Django maneja una estructura de bloques para las plantillas, por lo tanto, se creó un archivo HTML base, en el cual, se establecieron bloques para mostrar el contenido dinámico. En estos bloques, se agrega el contenido recibido de cada vista en dependencia de los resultados a mostrar.

Gráfico 22. Página Inicio del sitio web



Fuente: Elaboración propia

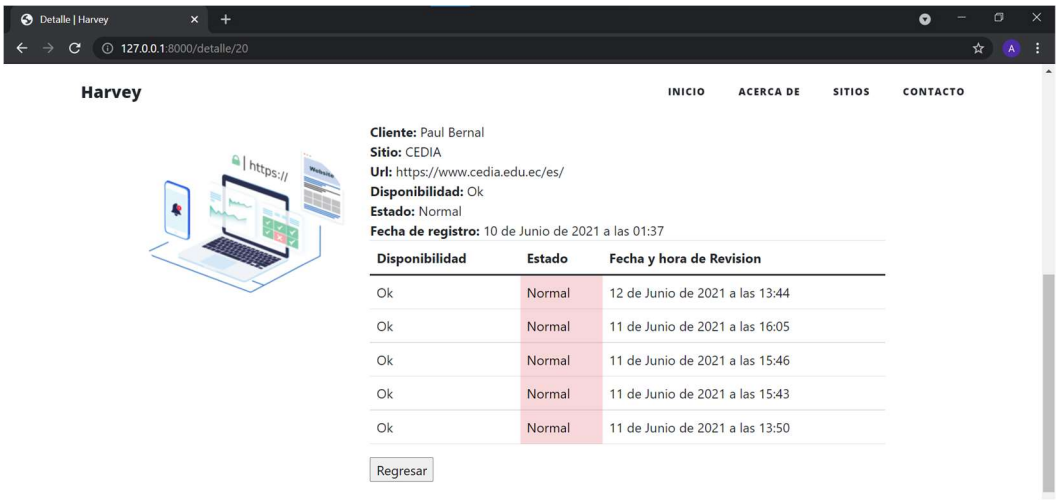
Gráfico 23. Página Sitios



Fuente: Elaboración Propia

En la página Detalle correspondiente a cada sitio, se muestran los resultados de los últimos 5 escaneos realizados

Gráfico 24. Detalle del monitoreo CEDIA



Fuente: Elaboración Propia

2.3.4 *Script* de Monitoreo y Detección

La parte final del proyecto consiste en construir el *script* de monitoreo que realice las siguientes acciones:

1. Conectarse a la base de datos y extraer las direcciones url de los sitios registrados.
2. Descargar el contenido del sitio.
3. Realizar el proceso de preprocesamiento (limpieza de datos, tokenización y extracción de características).
4. Clasificar el sitio en normal o desfigurado con el modelo creado.
5. En el caso de detectar un *Defacement* en uno de los sitios, se envía una notificación al administrador.
6. Insertar los resultados del escaneo en la base de datos.

En este *script,* se controlaron algunas excepciones al momento de extraer el contenido del sitio, entre ellas: errores de conexión, acceso denegado, demasiadas redirecciones y errores de *timeout*.

Este *script,* se calendariza para ejecutarse cada cierto tiempo en dependencia de las necesidades, que se requieran.

Gráfico 25. Paso de los datos de la BDD en el script de monitoreo

```
48    cursor = conexion.cursor()
49    sql = "SELECT id, url, cliente_id, nombre FROM modelo_sitio"
50    cursor.execute(sql)
51    registros = cursor.fetchall()
52
53    sqlCliente = "SELECT id, email, nombre, apellido FROM modelo_cliente "
54    cursor.execute(sqlCliente)
55    clientes = cursor.fetchall()
56
57    #Se pasan las url de los sitios a una lista ya que inicialmente son tuplas
58    #Se crea la lista client para guardar los correos de los clientes manteniendo
59    sitios = []
60    client = []
61    for registro in registros:
62        sitios.append([registro[1]])
63        for cliente in clientes:
64            if registro[2] == cliente[0]:
65                client.append([cliente[1],cliente[2]+" "+cliente[3]])
66
```

Fuente: Elaboración propia

CAPÍTULO III. ANÁLISIS DE LOS RESULTADOS DE LA INVESTIGACIÓN

3.1 Matriz de confusión

Con la obtención del modelo clasificador de sitios web, el siguiente paso es poner a prueba la efectividad del mismo. Para ello, se cuenta con el conjunto datos de *test,* de los cuales, se conoce su clase (normal, *defacement*). Se ejecuta el modelo clasificador previamente entrenado para determinar la cantidad de sitios que logra identificar correctamente de acuerdo con sus estados.

Dentro del ámbito de *machine learning*, la denominada matriz de confusión ayuda a medir la eficiencia de un determinado clasificador. Tras la ejecución del modelo en el conjunto de datos de *test,* se aplica la matriz de confusión a este resultado y esta a su vez proporciona un grupo de valores que sirvan para calcular las métricas que definan la calidad del modelo.

La matriz de confusión está conformada de la siguiente manera: las clases reales (columnas) y las clases resultantes (filas) generadas por el modelo clasificador, de forma, que se aprecia cuando una clase fue confundida con otra (Recuero de los Santos, 2018).

Tabla 4. Matriz de Confusión

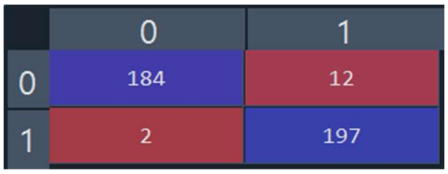
		Clase Real	
		Def	**Norm**
Clase Predicha	**Def**	*True Positives (TP)*	*False Positives (FP)*
	Norm	*False Negatives (FN)*	*True Negatives (TN)*

Fuente: modificado a partir de Hoang & Nguyen (2019)

La diagonal principal conformada por TP y TN es la que contenga todas las predicciones correctas que el modelo ha logrado identificar. Por otro lado, la diagonal de FN y FP muestra la cantidad de sitios clasificados de forma equivocada, es decir, los errores que ha tenido el modelo.

Para el presente caso de estudio la matriz de confusión resultante es la siguiente:

Tabla 5. Matriz de Confusión resultante



Fuente: Elaboración propia

Del gráfico anterior, se mencionaría que el modelo clasificador identificó correctamente 184 muestras como sitios desfigurados mientras, que se equivocó con 12. Por otro lado, identificó acertadamente 197 sitios como normales y se equivocó en 2. Con estos valores dados en la matriz de confusión, se generen métricas adicionales utilizadas en *machine learning* para evaluar en valores porcentuales la precisión, sensibilidad, exactitud, valor F y tasa de falsos del clasificador.

Según Mishra (2020), se expondría cada valor y fórmulas de la siguiente manera:

Precisión (*Precision*)

Hace referencia a lo cerca, que se encuentra el resultado de una clasificación, es considerada como la calidad del modelo. La fórmula, se calcula de la siguiente manera:

$$\frac{TP}{TP + FP} * 100\%$$

Sensibilidad (*Recall*)

Es la cantidad resultante de sitios web normales que el modelo es capaz de clasificar correctamente. La fórmula es la siguiente:

$$\frac{TP}{TP + FN} * 100\%$$

Exactitud (*Accuracy*)

Este valor, se encarga de medir el porcentaje de casos que el modelo clasificador predijo de forma acertada. Su fórmula es la siguiente:

$$\frac{TP + TN}{TP + FP + TN + FN} * 100\%$$

Valor F (F1 *score*)

Es la medida armónica entre los valores de precisión y sensibilidad. Indica que tan preciso y robusto es el modelo clasificador. Cuanto mayor sea el valor F, mejor es el rendimiento del modelo. La fórmula de cálculo es la siguiente:

$$\frac{2TP}{2TP + FP + FN} * 100\%$$

Tase de falsos Positivos (False *Positive Rate*)

Este valor muestra en que porcentaje el modelo clasifica erróneamente un sitio web. Su fórmula es la siguiente:

$$\frac{FP}{FP + FN} * 100\%$$

3.2 Resultados del Modelo

Los valores resultantes para el modelo generado son los siguientes:

Tabla 6. Métricas resultantes del modelo clasificador

Precisión	Sensibilidad	Exactitud	Valor F	Tasa de falsos positivos
94%	99%	96%	97%	6%

Fuente: Elaboración propia

Se observa que las métricas resultantes son bastante altas, lo cual, indica que el clasificador es apto a pesar de tener una tasa de falsos positivos de 6%.

Adicionalmente, tras la investigación realizada, se encontró que la técnica de n-gramas de caracteres sería muy útil para encontrar patrones dentro de un cuerpo de texto, indistintamente del idioma en el que se encuentre el mismo. Muestra de ello es que el modelo clasificador resultante funciona para el grupo de sitios de test en donde, se encuentran sitios en inglés, portugués y español.

CONCLUSIONES

- La revisión del estado del arte respecto a las herramientas de monitoreo y detección de *Defacement* en sitios web arroja que existen principalmente dos categorías que cubren diferentes técnicas para identificar cambios en sitios web: soluciones basadas en métodos tradicionales y soluciones basadas en métodos avanzados. Las primeras corresponden a: análisis de líneas de código, revisión de árboles DOM y la comparación de *checksum* del contenido de los sitios monitoreados. Por otro lado, existen técnicas avanzadas como: computación visual, programación genética y *machine learning*.

- La investigación permitió conocer que, si bien las técnicas basadas en métodos tradicionales serían efectivas para sitios estáticos, estás no serían prácticas para sitios dinámicos puesto que su constante cambio generaría demasiados falsos positivos. Para contrarrestar esto, las técnicas basadas en métodos avanzados ayudan a descubrir patrones o comportamientos en los sitios web y de esta forma detectar un *Defacement.* Sin embargo, es importante recalcar que las mismas requieren de altos recursos computacionales y adicionalmente del acceso al servidor para monitorear el sitio web y sus archivos.

- El diseño del sistema de monitoreo no intrusivo, se enfocó en una técnica de *Machine Learning.* Se desarrolló un modelo de detección basado en el algoritmo de clasificación *Random Forest* entrenado con sitios normales y sitios desfigurados. Para la administración del sistema, se creó una interfaz web que permita el ingreso de los clientes y sitios a monitorear. Finalmente, se calendarizó una tarea para que ejecute una validación de cada sitio ingresado a través del modelo clasificador creado y de encontrarse un *Defacement* en alguno de los escaneos, se genera una notificación al administrador del sitio. Es importante mencionar que, al disponer del clasificador ya generado, no se requiere el consumo elevado de recursos computacionales en cada validación.

RECOMENDACIONES

- Los algoritmos de Machine Learning aprenden en base a patrones o características de las muestras, que se proporcionen. Por ello, se incrementaría el nivel de efectividad del modelo clasificador resultante al volver a entrenar el algoritmo con una mayor cantidad de ejemplos de sitios web normales y sitios desfigurados.

- El modelo clasificador generado usa como características de análisis los trigramas con mayor frecuencia de aparición en cada sitio web. Se plantea incluir características adicionales tales como: el color del fondo, el título de la página y los enlaces, que se encuentren en las diferentes páginas web.

- El Sistema monitorea la página proporcionada en la dirección url. Sin embargo, se plantea configurar un escaneo exhaustivo que permita recorrer todas las páginas que conforman el sitio web y clasificar cada una de ellas. De esta forma, se potenciaría el alcance del Sistema.

BIBLIOGRAFÍA

Abraham Requena. (2018, diciembre 25). *10 diferencias entre Scrum y Kanban* [10 diferencias entre Scrum y Kanban]. OpenWebinars.net. https://openwebinars.net/blog/diferencias-scrum-kanban/

Alvarez, M. (2017, febrero 15). Como calcular la frecuencia de término – frecuencia inversa de documento. *Manolo Alvarez: Blog*. https://manoloalvarez.blog/2017/02/como-calcular-la-frecuencia-de-termino-frecuencia-inversa-de-documento/

Amat, J. (2020). *Random Forest con Python*. https://www.cienciadedatos.net/

Arsys. (2019, mayo 14). Principales retos de seguridad en Cloud. *Blog de arsys.es*. https://www.arsys.es/blog/soluciones/infraestructura/retos-seguridad-cloud/

ASALE, R.-, & RAE. (s. f.). *Ciberespacio | Diccionario de la lengua española*. «Diccionario de la lengua española» - Edición del Tricentenario. Recuperado 14 de noviembre de 2020, de https://dle.rae.es/ciberespacio

BBC. (2018, abril 18). NHS website defaced by hackers. *BBC News*. https://www.bbc.com/news/technology-43812539

Bernal. (2014). *Estudio de herramientas de software libre para la creación de sistemas de detección automatizada de Defacements de sitio web* [PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR]. http://repositorio.puce.edu.ec/handle/22000/11941

Bono Cabré, R. (2012). *Diseños cuasi-experimentales y longitudinales*. http://diposit.ub.edu/dspace/handle/2445/30783

Borgolte, K., Kruegel, C., & Vigna, G. (2015). *Meerkat: Detecting Website Defacements through Image-based Object Recognition*. 595-610. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/borgolte

Cloudorado. (2020). *AWS (Amazon Web Services) vs Microsoft Azure vs Google Cloud Platform*. Cloudorado - Cloud Computing Comparison Engine. https://www.cloudorado.com/vs/aws_vs_azure_vs_google

Cristina Ramos. (2017, enero 23). *Blog de Cristina Ramos Vega*. Scrum en pocas palabras. https://cristinaramosvega.com/scrum-pocas-palabras/

D Avi. (2020, octubre). *Top 10 Common Web Attacks: The First Steps to Protect Your Website*. VpnMentor. https://www.vpnmentor.com/blog/top-10-common-web-attacks/

Diccionario informático. (s. f.). Recuperado 24 de noviembre de 2020, de https://www.lawebdelprogramador.com/diccionario/

Estela, M. (2020, noviembre). *Método Deductivo—Concepto, ejemplos y método Inductivo*. https://concepto.de/metodo-deductivo/

Gillman, D., Lin, Y., Maggs, B., & Sitaraman, R. K. (2015). Protecting Websites from Attack with Secure Delivery Networks. *Computer*, *48*(4), 26-34. https://doi.org/10.1109/MC.2015.116

Gonzalez, L. (2018, marzo 23). Aprendizaje Supervisado: Random Forest Classification. 🤠 *Aprende IA*. https://www.youtube.com/watch?v=VH7eLWsLCks

González, Y. (2020, julio 2). ¿Cómo proteger tu página web del defacement? *Grupo Atico34*. https://protecciondatos-lopd.com/empresas/defacement/

Guillermo Rodríguez. (2017). *Diferentes metodologías ágiles* | [Diferentes metodologías ágiles]. https://lorbada.com/blog/2017/02/10/diferentes-metodologias-agiles/

Hoang, X. D., & Nguyen, N. T. (2019). A Multi-layer Model for Website Defacement Detection. *Proceedings of the Tenth International Symposium on Information and Communication Technology*, 508-513. https://doi.org/10.1145/3368926.3369730

IONOS. (2020). *Los mejores CMS en 2020: Una comparativa de los gestores de contenido*. IONOS Digitalguide. https://www.ionos.es/digitalguide/hosting/cms/cms-en-comparativa-los-gestores-de-contenido-mas-usados/

Julio Roche. (2020). *Kanban vs. Scrum*. Deloitte Spain. https://www2.deloitte.com/es/es/pages/technology/articles/kanban-vs-scrum.html

Kim, W., Lee, J., Park, E., & Kim, S. (2006). *Advanced Mechanism for Reducing False Alarm Rate in Web Page Defacement Detection*.

Maggi, F., Balduzzi, M., Flores, R., Gu, L., & Ciancaglini, V. (2018). Investigating Web Defacement Campaigns at Large. *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 443-456. https://doi.org/10.1145/3196494.3196542

Malwarebytes. (s. f.). *¿Qué es el malware?* Malwarebytes. Recuperado 22 de noviembre de 2020, de https://es.malwarebytes.com/malware/

Mao, B.-M., & Bagolibe, K. D. (2019). A Contribution to Detect and Prevent a Website Defacement. *2019 International Conference on Cyberworlds (CW)*, 344-347. https://doi.org/10.1109/CW.2019.00062

Masango, M., Mouton, F., Antony, P., & Mangoale, B. (2018). An Approach for Detecting Web Defacement with Self-healing Capabilities. En M. L. Gavrilova, C. J. K. Tan, & A. Sourin (Eds.), *Transactions on Computational Science XXXII: Special Issue on Cybersecurity and Biometrics* (pp. 29-42). Springer. https://doi.org/10.1007/978-3-662-56672-5_3

Medvet, E., Fillon, C., & Bartoli, A. (2007). Detection of Web Defacements by means of Genetic Programming. *Third International Symposium on Information Assurance and Security*, 227-234. https://doi.org/10.1109/IAS.2007.13

Mishra, A. (2020, mayo 28). *Metrics to Evaluate your Machine Learning Algorithm*. Medium. https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

Mondragón, O., Arcos, A. F. M., Urcuqui, C., & Cadavid, A. N. (2017). Security control for website defacement. *Sistemas y Telemática*, *15*(41), 45-55. https://doi.org/10.18046/syt.v15i41.2442

NeoAttack. (2020). ¿Qué es un Framework y para que sirve? - Neo Wiki. *NeoAttack*. https://neoattack.com/neowiki/framework/

OTEIC. (2019). *Grupo Oteic: Expertos en Consultoría y Asesoramiento Empresarial.* SCRUM: Gestión Ágil de Proyectos. http://www.oteic.com/noticias/ver.php?id=es&desde=&Nnoticia=1554187809

OWASP. (2017). *OWASP Top Ten Web Application Security Risks | OWASP*. https://owasp.org/www-project-top-ten/

Recuero de los Santos. (2020, febrero 14). *PLN con Python: Tokens, stop words y ngrams - Think Big Empresas*. Think Big. https://empresas.blogthinkbig.com/pln-con-python-tokens-stop-words-y-ngrams/

Recuero de los Santos, P. (2018, enero 23). *Machine Learning a tu alcance: La matriz de confusión - Think Big Empresas*. Think Big. https://empresas.blogthinkbig.com/ml-a-tu-alcance-matriz-confusion/

San José, J. (2020, agosto 19). *75 webs de la Organización Internacional para las Migraciones, víctimas de un defacement.* Derecho de la Red. https://derechodelared.com/organizacion-internacional-para-las-migraciones-defacement/

Simon Kemp. (2020). *Digital 2020: Global Digital Overview*. DataReportal – Global Digital Insights. https://datareportal.com/reports/digital-2020-global-digital-overview

Sucuri. (2019). *Sucuri—Website Threat Report 2019*. Sucuri. https://sucuri.net/reports/2019-hacked-website-report/

Vasseur, L. (2018). *El defacement o desfiguración: Táctica para el activismo político* [Proyecto/Trabajo fin de carrera/grado, Universitat Politècnica de València]. https://riunet.upv.es/handle/10251/110074

W3Techs. (2020). *Usage Statistics and Market Share of Content Management Systems, November 2020*. W3Techs Web Technology Surveys. https://w3techs.com/technologies/overview/content_management

Zone-H. (s. f.-a). *Zone-H.org—Unrestricted information | Defacements archive*. Zone-H.org - Unrestricted information. Recuperado 13 de noviembre de 2020, de http://www.zone-h.org/archive/filter=1/published=0/domain=.ec/fulltext=1/page=2

Zone-H. (s. f.-b). *Zone-H.org—Unrestricted information | Yearly & Monthly & Daily attacks*. Zone-H.org - Unrestricted information. Recuperado 13 de noviembre de 2020, de http://www.zone-h.org/stats/ymd