Diseño e Implementación de BOTS para Automatizar Tareas de Búsqueda y Análisis de Vulnerabilidades en Sistemas Web

Quinatoa Medina, Jordy Javier y Villares Jimenez, Julio Javier

Departamento de Ciencias de la Computación

Carrera de Ingeniería en Tecnología de la Información

Trabajo de integración curricular, previo a la obtención del título de Ingeniera en Tecnologías de la Información

Ing. Germán Eduardo Rodríguez Galán, Mgtr.

10 de agosto de 2022

Reporte de Verificación de Contenido

UIC_202250-Quinatoa Jordy_Villares Julio.pdf

Scanned on: 4:40 August 11, 2022 UTC

Identical Words	113
Words with Minor Changes	92
Paraphrased Words	450
Omitted Words	4779

Firma:

Firmado electrónicamente por:
GERMAN EDUARDO
RODRIGUEZ GALAN

.....

Mgtr. Rodríguez Galán, Germán Eduardo

C. C.: 0603431685

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

CERTIFICACIÓN

Certifico que el trabajo de integración curricular, "**DISEÑO E IMPLEMENTACIÓN DE BOTS PARA LA AUTOMATIZACIÓN DE TAREAS DE BÚSQUEDA Y ANÁLISIS DE VULNERABILIDADES EN SISTEMAS WEB**" fue realizado por los señores **Quinatoa Medina, Jordy Javier** y **Villares Jimenez, Julio Javier** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Santo Domingo de los Tsáchilas, 10 de agosto del 2022

Firma:

Firmado electrónicamente por:
GERMAN EDUARDO
RODRIGUEZ GALAN

.....

Mgtr. Rodríguez Galán, Germán Eduardo
C. C.: 0603431685

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

RESPONSABILIDAD DE AUTORÍA

Nosotros, **Quinatoa Medina, Jordy Javier** y **Villares Jimenez, Julio Javier**, con cedulas de ciudadanía **N° 2300318074** y **N° 2300121239**, declaramos que el contenido, ideas y criterios del trabajo de integración curricular: **"DISEÑO E IMPLEMENTACIÓN DE BOTS PARA LA AUTOMATIZACIÓN DE TAREAS DE BÚSQUEDA Y ANÁLISIS DE VULNERABILIDADES EN SISTEMAS WEB"** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Santo Domingo de los Tsáchilas, 10 agosto de 2022

Firmas:

.....

Quinatoa Medina, Jordy Javier

C.C.: 2300318074

.....

Villares Jimenez, Julio Javier

C.C.: 2300121239

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

AUTORIZACIÓN DE PUBLICACIÓN

Nosotros **Quinatoa Medina, Jordy Javier y Villares Jimenez, Julio Javier,** con cédulas de ciudadanía **N° 2300318074 y N° 2300121239**, autorizo/autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de integración curricular: **"DISEÑO E IMPLEMENTACIÓN DE BOTS PARA LA AUTOMATIZACIÓN DE TAREAS DE BÚSQUEDA Y ANÁLISIS DE VULNERABILIDADES EN SISTEMAS WEB"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi/nuestra responsabilidad.

Santo Domingo de los Tsáchilas, 10 agosto de 2022

Firmas:

.....

Quinatoa Medina, Jordy Javier

C.C.: 2300318074

.....

Villares Jimenez, Julio Javier

C.C.: 2300121239

Dedicatoria

Este trabajo de titulación va dedicado a mis padres Luis Quinatoa y Sandra Medina por siempre apoyarme y creer en mí, por brindarme su amor incondicional sirviéndome de fuente de motivación, por su esfuerzo que demuestran día a día y que a pesar de los obstáculos que alguna vez pudieron llegar a presentarse siempre estuvieron ahí para ayudarme a cumplir con mis metas y formación profesional.

A mi hermana menor Lisseth Quinatoa que es una parte importante de mi vida, razón por la cual siempre trato de ser un ejemplo a seguir tanto en lo académico como en lo personal.

A mi abuela Gloria Portillo que es una segunda madre para mí, ya que desde que nací siempre ha estado para brindarme su cuidado y cariño.

Jordy Javier Quinatoa Medina

Dedicatoria

Este trabajo de titulación representa para mí, años de esfuerzo y dedicación para lograr el objetivo de ser un profesional. Años en donde aparecieron muchos obstáculos en mi vida, que sin duda me ayudaron a obtener experiencia y madurez que cada día lo voy fortaleciendo.

Por ello, dedico este trabajo a mi madre Elida Villares y hermano Wilber Herrera por haberme forjado un alto espíritu de responsabilidad y valores que me ayudaron a crecer como persona y a prepararme constantemente en mi formación profesional. Mi madre y hermano fueron mi apoyo fundamental en varias etapas de mi vida, brindándome su amor incondicional y ayuda, para poder seguir adelante. Muchos de mis logros se los debo a ellos, y en especial a mi hermano, que a pesar de que no conté con una figura paterna por el temprano fallecimiento de mi padre, mi hermano jugó un papel importante en mi vida, tomando el liderazgo del hogar y enseñándome que con esfuerzo y dedicación se puede lograr muchas cosas.

De igual forma, dedico este trabajo a todas las personas que han formado parte de esta etapa de mi vida, y con quienes he compartido grandes momentos, llenándome de alegrías y experiencias a lo largo de mi formación tanto académica como personal.

Julio Javier Villares Jimenez

Agradecimientos

Agradezco a Dios por darme la salud y las fuerzas necesarias para poder alcanzar una meta más de mi vida. Doy gracias a toda mi familia que me supieron apoyar en estos años de estudios universitarios.

A la Universidad de las Fuerzas Armadas ESPE sede Santo Domingo por permitirme formarme como profesional en sus instalaciones. De igual manera doy gracias a todos los docentes con los que compartí aula, por sus enseñanzas, consejos y valores brindados para formarme como un buen profesional.

Agradezco al Ing. German Rodríguez por la confianza depositada en nosotros para la realización de este trabajo de titulación, por su asesoramiento y apoyo a lo largo de todo este proceso.

Un agradecimiento especial a mi compañero de tesis Julio Villares por su responsabilidad y compromiso al realizar este trabajo. Además de su confianza y amistad brindada en estos años de universidad.

También agradezco a todos mis familiares que alguna vez me supieron ayudar ya sea económicamente o moralmente para afrontar mis estudios académicos.

Y a todos mis amigos que han formado parte de mi vida universitaria, con quienes he compartido momentos de felicidad, aprendizaje, experiencias, y que desde cualquier forma me han apoyado en esta etapa de mi vida.

Jordy Javier Quinatoa Medina

Agradecimientos

Agradezco primeramente a Dios, que a pesar de todos los obstáculos y tropiezos que he tenido a lo largo de mi vida, sin dudas, me ha dado las fuerzas necesarias para estar aquí y seguir adelante. Doy gracias a mi madre y hermano quienes fueron mi motor de inspiración para lograr el objetivo de ser un profesional, al igual que toda mi familia, pareja y amigos quienes estuvieron a mi lado y me ayudaron a seguir adelante.

Agradezco enormemente a la Universidad de las Fuerzas Armadas ESPE sede Santo Domingo por abrirme sus puertas para formarme como un profesional dentro de sus aulas. De igual manera, agradecer a todos los docentes de la carrera de Ingeniería en Tecnologías de la Información, quienes año tras año me brindaron su conocimiento para lograr mi profesión.

Un agradecimiento especial al Ing. German Rodríguez por confiar en nosotros, y por brindarnos toda su paciencia, conocimiento y apoyo en la elaboración del presente proyecto de tesis. De igual manera agradezco al Ing. Héctor Revelo quien confió en mí y me brindó su apoyo en el deseo de formar parte de esta hermosa carrera.

De igual forma, agradezco a mi compañero de tesis Jordy Quinatoa, por su paciencia, compromiso y perseverancia en la realización de este trabajo y por su amistad brindada a lo largo de esta formación académica. Muchas gracias a todas las personas que formaron parte de esta bonita etapa de vida.

Julio Javier Villares Jimenez

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Resumen

Hoy en día muchas empresas adoptan la arquitectura cliente-servidor para manejar la lógica de su negocio. El principal problema de estos sistemas, es que se utilizan herramientas que requieren la intervención humana y procesos manuales para encontrar vulnerabilidades, afectando de manera directa al tiempo requerido para parcharlas. Una vulnerabilidad que los hackers pueden explotar son las cookies generadas por los sitios web, porque de acuerdo a la literatura, contienen información sensible del usuario que puede ser robada mediante ataques de tipo Cross-Site Scripting (XSS). Por otro lado, la creación de bots maliciosos ha provocado que las empresas apliquen sistemas CAPTCHA para contrarrestar estos ataques, pero con la aplicación de técnicas de visión artificial se podrían vulnerar estos sistemas. En este contexto, el objetivo del presente trabajo es diseñar e implementar bots para automatizar las tareas de búsqueda y análisis de vulnerabilidades en sistemas web, utilizando herramientas RPA. Para el desarrollo del bot para vulnerar sistemas CAPTCHA, se utilizó la herramienta UiPath y técnicas de visión artificial para su análisis. Mientras que para el bot de análisis de vulnerabilidades XSS se utilizó la herramienta UI.Vision y scripts desarrollados en Python para su análisis. Con los resultados se demostró que las páginas webs pueden ser vulneradas mediante ataques de tipo XSS por medio del análisis de sus cookies, y que los sistemas CAPTCHA basados en imágenes pueden ser vulnerados utilizando técnicas de visión artificial. Nuestro aporte es el estudio comparativo realizado entre las herramientas UiPath y UI.Vision para mejorar la automatización de procesos repetitivos. Además de un modelo para el testeo automatizado de sistemas web, mejorando la velocidad de detección de las vulnerabilidades (CAPTCHA y XSS) y proporcionando datos que contribuyan a futuras investigaciones.

Palabras claves: vulnerabilidades web, Cross-Site Scripting, CAPTCHA, RPA.

Abstract

Many companies today adopt client-server architecture to handle their business logic. The main problem with these systems is that tools that require human intervention and manual processes are used to find vulnerabilities, directly affecting the time required to patch them. A vulnerability that hackers can exploit is the cookies generated by websites because, according to the literature, they contain sensitive user information that can be stolen through Cross-Site Scripting (XSS) type attacks. On the other hand, creating malicious bots has caused companies to apply CAPTCHA systems to counteract these attacks, but these systems could be violated with artificial vision techniques. In this context, this work aims to design and implement bots to automate search and vulnerability analysis tasks in web systems using RPA tools. For the development of the bot to violate CAPTCHA systems, the UiPath tool and artificial vision techniques were used for its analysis. While for the XSS vulnerability analysis bot, the UI.Vision tools and scripts developed in Python were used for its analysis. The results showed that XSS-type attacks can violate web pages by analyzing their cookies and that CAPTCHA systems based on images can be violated using artificial vision techniques. Our contribution is the comparative study carried out between the UiPath and UI.Vision tools to improve the automation of repetitive processes. In addition to a model for the automated testing of web systems, improving the speed of detecting vulnerabilities (CAPTCHA and XSS) and providing data that contributes to future research.

Keywords: web vulnerabilities, Cross-Site Scripting, CAPTCHA, RPA.

Capítulo I

Introducción

En la actualidad, el avance de la tecnología y el incremento de la información, han sido la base para el desarrollo de sistemas acorde a las nuevas necesidades tecnológicas. Muchas empresas y usuarios se han visto en la necesidad de implementar sistemas informáticos basados en una arquitectura cliente-servidor debido a su eficacia y facilidad de acceso. En la mayoría de casos, resulta beneficioso la implementación de sistemas web, sin embargo, pueden estar expuestos a ataques perpetrados por ciberdelincuentes preparados y con sólidos conocimientos.

El uso de sistemas automatizados (bots) se ha intensificado en la integración de los procesos de las compañías y actividades cotidianas de las personas, existen bots que pueden ejecutar desde tareas sencillas hasta manejar toda la lógica de negocio de una empresa. En este contexto, la investigación realizada en Missouri Enterprise menciona que una organización al integrar bots en sus procesos, puede reducir los costos de un 24% al 50%, cumplir con el tiempo de ciclo de procesos desde el 30% a un 50% y mejorar los niveles de productividad desde un 10% hasta un 16%. Este caso de estudio, sirve como referencia para poner en contexto las mejoras que produce la implementación de tecnologías basadas en RPA (School E. B., 2021).

Un problema en la ejecución de la búsqueda y análisis de vulnerabilidades dentro de los sistemas web, es que se utilizan herramientas que requieren la intervención humana y procesos manuales para obtener la información, afectando de manera directa al tiempo necesario para parchear la vulnerabilidad encontrada.

Una de las vulnerabilidades analizadas son los ataques Cross-site Scripting (XSS), en donde según OWASP (2021), indica que este tipo de ataques han ganado más espacio e incidencia en los últimos años escalando a la tercera posición en el top 10 de las

vulnerabilidades web. La principal vulnerabilidad que los hackers tienen como objetivo son las cookies generadas por los sitios web, debido a que pueden llegar a contener información sensible del usuario que puede filtrarse. Por otro lado, la creación de bots maliciosos ha provocado que las empresas apliquen sistemas CAPTCHA para contrarrestar estos ataques, pero con la aplicación de técnicas de visión artificial se podrían vulnerar estos sistemas.

En ese contexto, en este trabajo se desarrollaron bots para automatizar las tareas de búsqueda y análisis de vulnerabilidades en sistemas web, por medio de la utilización de herramientas RPA. El primer bot se desarrolló para demostrar las vulnerabilidades de los sistemas CAPTCHA, utilizando la herramienta UiPath y técnicas de visión artificial. El segundo bot se desarrolló para analizar vulnerabilidades de tipo XSS empleando la herramienta UI.Vision y scripts desarrollados en Python para el análisis de las cookies.

Este documento se encuentra estructurado de la siguiente manera, el Capítulo II presenta el marco de referencia con la información y conceptos relacionados con la investigación. En el Capítulo III se detalla la metodología utilizada para el desarrollo de los bots para automatizar las tareas de búsqueda y análisis de vulnerabilidades en sistemas web (XSS y vulnerabilidades en sistemas CAPTCHA). En el Capítulo IV se analizan los resultados obtenidos en la ejecución de los bots y la comparación de las herramientas RPA utilizadas en este trabajo. Por último, en el Capítulo V se presentan las conclusiones y recomendaciones.

Antecedentes

Debido a los constantes ataques que reciben los sistemas informáticos de una empresa o institución, es importante establecer un equipo de personas con sólidos conocimientos en ciberseguridad que puedan contrarrestar en cierta medida los ataques perpetrados por los ciberdelincuentes. Según el Informe de Riesgos Globales 2020 del Foro Económico Mundial, menciona que por cada minuto existen 100 intentos de explotación de vulnerabilidades a diferentes sistemas informáticos y más aún, teniendo en consideración el auge de tecnologías como el Internet de la Cosas (IoT) (Foro Económico Mundial, 2020).

Además, se menciona que a raíz de la pandemia de la Covid 19, los delitos informáticos han aumentado en un 600% a nivel mundial, de los cuales muchos ataques son dedicados específicamente a robos de identidad, secuestros de datos, penetración, denegación de servicios y piratería de software (Foro Económico Mundial, 2020). De igual forma es importante destacar que los principales afectados por causa de estos ataques cibernéticos son las pequeñas y medianas empresas.

Según el portal Small Business Trends (2020), menciona que de los ataques informáticos de todo el mundo, el 43% fueron dirigidos a pequeñas empresas, de los cuales, solo el 14% lograron calificar su capacidad de respuesta y de mitigación de riesgos como efectiva, mientras que el 29% restante tuvo déficit en la mitigación de riesgos de su seguridad informática, reflejando el peligro potencial al que se enfrentan a diario las empresas y usuarios en general.

En Ecuador la situación es muy similar, según el informe de la Asociación Ecuatoriana de Ciberseguridad (AECI) en colaboración con Deloitte (2020), menciona que el principal indicador para un buen desarrollo de la ciberseguridad dentro en las empresas es contar con un especialista en el área, pero en la mayoría de los casos las empresas ecuatorianas carecen de este tipo de profesionales dentro del sector. En dicho informe también se afirma que solo el 20% de un total aproximado de 100 empresas encuestadas en el Ecuador gestionan de manera adecuada la seguridad de la información, mientras que el 80% de empresas restantes poseen un manejo inadecuado de la ciberseguridad.

Entre los principales ataques cibernéticos más comunes a los que se enfrentan los sistemas web sean empresariales o personales, son los ataques de tipo XSS (Cross-site Scripting), los cuales se aprovechan precisamente de las vulnerabilidades que tienen las páginas web para insertar scripts maliciosos que pueden afectar a la seguridad informática de los usuarios cuando abren la página web infectada.

Según OWASP Top 10 (2021), los ataques de tipo Cross-site Scripting se encuentran englobados en la categoría de ataques de “Inyección” que ocupan la tercera posición en el top 10 de las vulnerabilidades web, con un índice de incidencia medio de 3,37%. Además, se menciona que de 274,228 aplicaciones web analizadas, existieron 32,078 vulnerabilidades comunes, con una tasa de incidencia del 19.09%. En dichas vulnerabilidades web existieron debilidades comunes (CWE) más notables como: ataques de tipo Cross-site Scripting (XSS), *SQL Injection y External Control of File Name or Path* (Control externo de nombre de archivo o ruta).

Otro ataque común son los de tipo DDoS (Denegación de Servicios), motivo por el que las empresas utilizan sistemas CAPTCHA para contrarrestarlos. Según una investigación realizada con el financiamiento de China y Reino Unido, se presentó un modelo de Inteligencia Artificial capaz de vulnerar los CAPTCHA basados en textos de las principales páginas web como Wikipedia, eBay y Microsoft resolviendo el desafío en 0.05 segundos mediante la utilización de una GPU de escritorio. En dicho estudio se probó la eficacia del sistema en 33 CAPTCHA basados en texto, de los cuales 11 se utilizan en 32 de los 50 sitios webs más populares del mundo según el ranking de abril 2018 (Guixin Ye et. al., 2018).

Además, es importante mencionar que muchos de estos ataques informáticos son perpetrados mientras se realizan actividades cotidianas dentro de una organización, generando un alto tráfico de información útil para los ciberdelincuentes. Todo esto, sumado a las mínimas acciones ejecutadas en la gestión de las vulnerabilidades dentro de las empresas, conlleva a que las amenazas se vuelvan realidad y sean indetectables en su fase inicial. Actuar de manera inmediata para solucionar los problemas después de detectar una posible vulnerabilidad dentro de los sistemas informáticos, puede evitar pérdidas económicas y de información a la empresa.

La opción más viable para buscar vulnerabilidades en menor tiempo es implementando sistemas inteligentes automáticos. Actualmente, el proceso de análisis de vulnerabilidades en

los sistemas informáticos ha empezado a cambiar hacia enfoques más automatizados. Por ello, el gran aporte que ofrece la Agencia de Proyectos de Investigación Avanzados de Defensa (DARPA), ha generado ideas e investigaciones enfocadas a la creación de sistemas de razonamiento cibernético capaces de detectar vulnerabilidades en los sistemas informáticos y así poder parcharlos en el menor tiempo posible (Shoshitaishvili et al., 2018).

Planteamiento y formulación del problema

Muchas empresas han optado por utilizar la arquitectura cliente-servidor para alojar sus aplicaciones e información, provocando un aumento de los ataques informáticos a estos servicios y poniendo en riesgo la seguridad de los datos. Por ello, es importante que una empresa tenga personal especializado en ciberseguridad, que se encargue de ejecutar las pruebas de testeo y la búsqueda de vulnerabilidades en los sistemas web.

En este contexto, se ha evidenciado que uno de los principales problemas de los sistemas web es el filtrado de información a través del uso de métodos de inyección de código malicioso. Un elemento muy vulnerable son las cookies generadas por las páginas web, ya que son archivos que podrían contener información sensible de una empresa o usuario, como datos de sesión, contraseñas, preferencias, etc. Cuando un sistema es vulnerado, pueden pasar días, semanas o incluso meses hasta que alguien se dé cuenta del ataque perpetrado, y de igual forma, puede pasar más tiempo para que se lance un parche que contrarreste el ataque.

Por otro lado, otro tipo de ataque muy común hacia los sistemas web son los enfocados a la Denegación de Servicios (DDoS), motivo por el cual muchas empresas optan por utilizar sistemas CAPTCHA en sus aplicaciones web para contrarrestar este tipo de ataques. Hoy en día la implementación de tecnologías como la Inteligencia Artificial y la automatización robótica de procesos, ha hecho que los sistemas CAPTCHA sean vulnerados fácilmente.

En este contexto, el principal problema para ejecutar un análisis y búsqueda de vulnerabilidades dentro de los sistemas web, es que se utilizan herramientas que requieren la intervención humana y procesos manuales (en ocasiones repetitivos) para obtener la

información y establecer un reporte, lo que sin duda afecta de manera directa al tiempo necesario para parchear la/s vulnerabilidad/es encontrada/s.

Con la implementación de bots para automatizar tareas de búsqueda y análisis de vulnerabilidades en sistemas web, se puede reducir el tiempo y por ende, la proporción de una respuesta más rápida a las posibles amenazas encontradas. El objetivo de este estudio es proponer una nueva metodología para la búsqueda de vulnerabilidades en sitios web con la implementación de tecnologías como la Automatización Robótica de Procesos (RPA) dentro del campo de la seguridad informática.

Justificación e importancia

La búsqueda y análisis de vulnerabilidades dentro de los sistemas web, juegan un papel importante en la prevención de posibles ataques que pueden afectar la seguridad de la información, evitando así posibles pérdidas económicas a una organización.

De igual forma, es importante proteger la información del usuario o empresa de los ataques de tipo Cross-site Scripting (XSS), ya que se puede filtrar datos a través de las cookies propias o de terceros generadas por las visitas en las páginas web. Por ello, el describir los procesos necesarios para analizar estas cookies, permite que las personas dedicadas a la seguridad informática obtengan un modelo para el testeo de sistemas web de manera automatizada.

Por otro lado, los sistemas CAPTCHAs son un medio que controla de cierta manera el acceso de bots maliciosos a una página web. Por ello, el demostrar que los CAPTCHAs basados en imágenes pueden ser vulnerados utilizando técnicas de visión artificial, permiten que futuras investigaciones se enfoquen en complementar la seguridad de los sistemas CAPTCHAs, específicamente los basados en imágenes.

Para mejorar la velocidad de detección de dichas vulnerabilidades mencionadas previamente, es importante automatizar ciertos procesos que pueden resultar repetitivos. En este contexto se ha planteado el diseño e implementación de bots para automatizar tareas de

búsqueda y análisis de vulnerabilidades en sistemas web, garantizando que todo su proceso se haga de manera automática.

Sistema de Objetivos

Objetivo general

Diseñar e implementar bots para automatizar tareas de búsqueda y análisis de vulnerabilidades en sistemas web.

Objetivos específicos

- Diseñar e Implementar BOTS para ejecutar tareas de web scraping de páginas web vulnerables a ataques de tipo Cross-Site Scripting (XSS).
- Demostrar la vulnerabilidad de los sistemas CAPTCHA usando BOTS con habilidades de visión artificial.

Alcance

La implementación del presente proyecto tiene como propósito diseñar e implementar bots para automatizar la búsqueda y análisis de vulnerabilidades en sistemas web, específicamente, páginas web que son de acceso o dominio público, por ende, el alcance del proyecto se centra en demostrar que las páginas web son vulnerables a ataques XSS por medio del análisis de sus cookies; y que los sistemas CAPTCHA pueden ser vulnerados mediante ataques ejecutados por bots con características de visión artificial.

Es importante mencionar que para este estudio se utilizarán los resultados obtenidos de la evaluación de distintas herramientas RPA por parte de los autores, aplicando un enfoque descriptivo para evaluar cada una de las características técnicas ofrecidas.

Capítulo II

Marco Teórico y Estado del Arte

Introducción del capítulo

El propósito del siguiente capítulo es teorizar los conceptos que contemplan el diseño e implementación de bots relacionados con la automatización de tareas de búsqueda y análisis de vulnerabilidades dentro de los sistemas web. Los apartados que se detallan a continuación, se enfocan en el análisis de conceptos básicos sobre los sistemas web, sus principales componentes, las amenazas, vulnerabilidades y los ataques más comunes. Este estudio está enfocado en el análisis de vulnerabilidades con la implementación de bots, es importante destacar los conceptos básicos acerca de la Automatización Robótica de Procesos (RPA), sus beneficios, ventajas y desventajas, los tipos, etc. Todo esto conjugado con el análisis de vulnerabilidades en los sistemas CAPTCHA, los ataques de tipo Cross-Site Scripting (XSS) y las herramientas propuestas para buscar este tipo de vulnerabilidades web.

¿Qué son los sistemas web?

Según Maldonado (2016), se denominan sistemas web a todas aquellas aplicaciones en las cuales los usuarios pueden acceder a través del internet, es decir, son sistemas que no se instalan en una computadora ya que se encuentran alojados en un servidor, el cual puede ser accedido por medio de un navegador web desde cualquier parte del mundo.

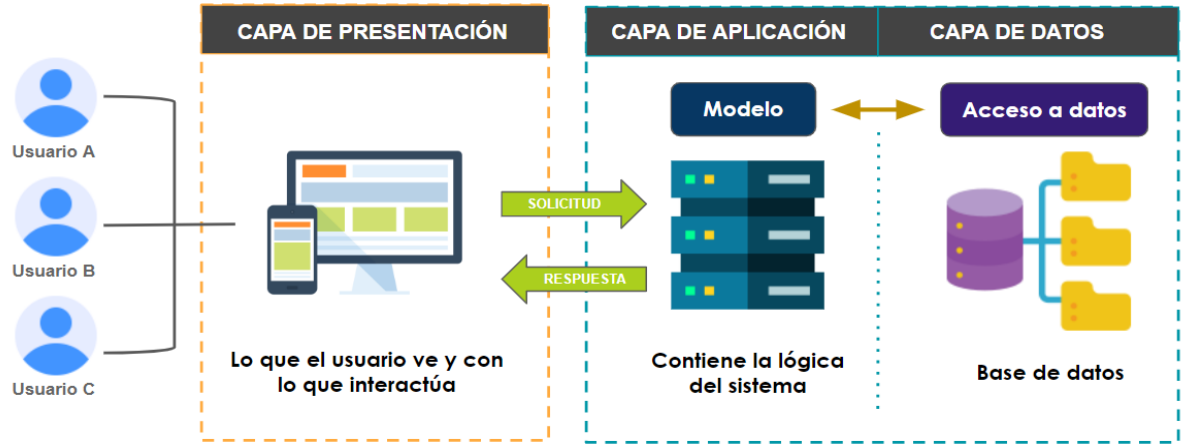
Internet es un gran espacio en el que es posible encontrar un amplio repositorio de sistemas diseñados para la web, pueden asistirnos en varios aspectos de la vida cotidiana como la investigación, el aprendizaje, la enseñanza, el entretenimiento y en muchas otras áreas (López, 2021). La mayor parte de los sistemas web se ha desarrollado para utilizarse o ejecutarse en cualquier navegador web (Chrome, Firefox, Safari, Opera, Microsoft Edge,etc.) sin importar el sistema operativo.

Componentes de los sistemas web

En el desarrollo de sistemas web se toman en cuenta varios aspectos a la hora de garantizar la seguridad de sus componentes, por lo cual, es importante conocer cómo se encuentran constituidos. Ortegón (2019) en su estudio expone que la mayoría de las aplicaciones web se encuentran estructuradas mediante una arquitectura de tres capas, como se muestra en la Figura 1.

Figura 1

Estructura principal de la arquitectura web



Nota. En la figura se observa la visualización de la arquitectura web completa.

En la Tabla 1 se observan las características más importantes de cada una de estas capas, que están constituidas por ciertas tecnologías que pueden estar expuestas a vulnerabilidades o amenazas provenientes de la web.

Tabla 1

Arquitectura de 3 capas de los sistemas o aplicaciones web

Capas	Concepto	Tecnologías	Vulnerabilidades
Capa de presentación (Interfaz)	Esta capa es accesible para los usuarios a partir de un navegador web, y como su nombre lo indica esta consta de elementos y procesos de interfaz de usuario que permiten la interacción con el sistema por medio de peticiones al servidor (Ortegón, 2019).	HTML CSS JavaScript Ajax	Cross-site scripting (XSS) Redirección de URL
Capa de aplicación (Servidor)	También conocida como capa de negocio, es la encargada de recibir las solicitudes del usuario provenientes del sistema web, para proceder a procesarlas y establecer la comunicación para acceder a los datos, pudiendolos eliminar o modificar en la capa de datos (Ortegón, 2019).	PHP Python Java JavaScript Node.js Ruby ASP.NET	Ataques DDos Inyección SQL Puertos abiertos Mala configuración
Capa de datos (Base de datos)	También conocida como capa de persistencia, esta representa la parte centralizada que recibe todas las peticiones de datos y proporciona acceso al almacenamiento persistente de una aplicación, por medio de la administración de un sistema de gestión de base de datos (IBM, 2020).	SGBD SQL SGBD NoSQL	Configuración de seguridad defectuosa Autenticación débil Desbordamiento de búfer

Nota. Características importantes de cada capa que conforma la arquitectura web con sus principales tecnologías.

Como se aprecia en la Tabla 1, la mayoría de los sistemas web comparten la misma infraestructura y componentes web, por lo que es importante establecer una correcta seguridad dentro de cada capa del proyecto web. Todo ello, ha llevado a que muchos ciberdelincuentes

se especialicen en atacar una capa en específico, con el fin de obtener información que afecte de manera directa alguna vulnerabilidad que posea la empresa.

Algo muy común, es cuando los atacantes se dedican a vulnerar la seguridad de los sistemas web de empresas reconocidas, con el fin de obtener información y datos que puedan ser útiles o simplemente para pedir dinero a cambio (ransomware). Un claro ejemplo de este tipo de ataques, se evidencia en la actualidad durante la guerra entre Ucrania y Rusia, en donde el Gobierno de Ucrania denuncia los constantes ataques informáticos a la página oficial del Parlamento Ucraniano, Ministerio de Exteriores y a los principales sitios web del Gobierno. Entre los principales ataques que se han ejecutado se puede constatar; ataques de DDos, ataques Cross-Site Scripting y ataques de SQL injection (Ramos, 2022).

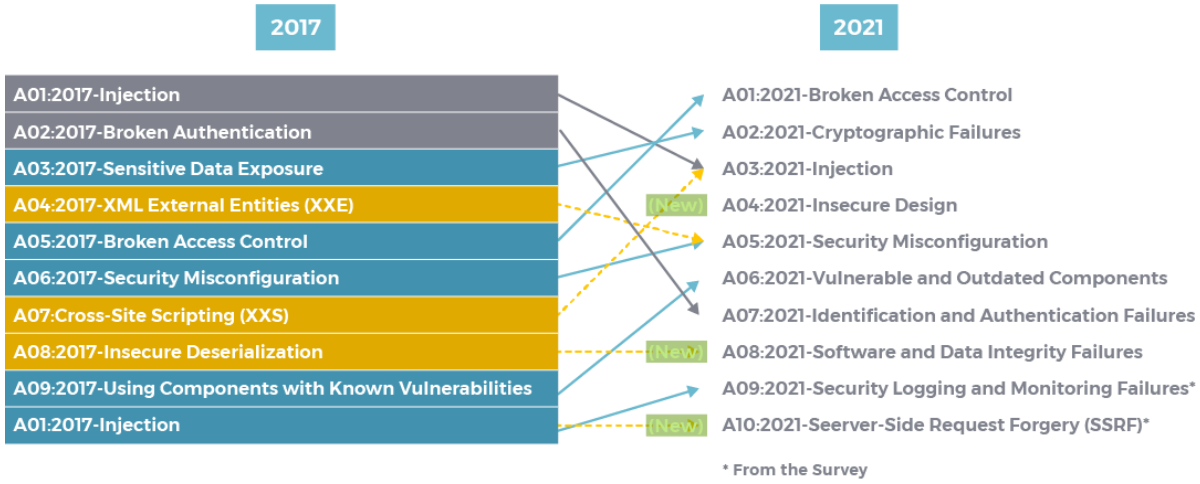
Vulnerabilidades y amenazas de los sistemas web

Hoy en día la web está repleta de diversos sistemas, lo que conlleva a la presencia de muchos problemas con respecto a los fallos de seguridad. Algunos de estos problemas están relacionados con la etapa de desarrollo ya que se cometen muchos errores en la programación, o la falta de protección del sistema contra ataques perjudiciales (Forsman, 2014). Según un estudio de la compañía Kaspersky, la principal fuente de amenazas a la seguridad de la información de una aplicación web son los intrusos externos, los cuales implementan amenazas basadas en el navegador mediante el uso de una serie de programas de software malicioso diseñados para infectar las computadoras de las víctimas (Kaspersky, 2022).

Para conocer sobre las distintas amenazas a los que se encuentran expuestos los sistemas web tenemos el informe “OWASP Top 10”, el cual se basa en la concienciación estándar para desarrolladores y seguridad de aplicaciones web. Se actualiza regularmente para mostrar constantemente los 10 riesgos más graves que enfrentan las organizaciones (OWASP Foundation, 2021). En la versión más reciente del documento (2021) se detallan las amenazas establecidas en la Figura 2 en base al orden de prioridad.

Figura 2

Principales amenazas web según OWASP Top 10 2021



Nota. El ranking de las vulnerabilidades está basado en OWASP Top 10, por OWASP, 2021, página oficial de OWASP (https://owasp.org/www-project-top-ten/).

Ataques más comunes de los sistemas web

Los ataques a aplicaciones web están en aumento y los estudios muestran que son una de las mayores causas de violaciones de datos. Las instituciones a nivel global continúan haciendo frente a la creciente ola de ataques a las aplicaciones web y sistemas. Por ello, un 50% de todos los sitios fueron amenazados por al menos una vulnerabilidad grave a lo largo del año 2021, según el estudio de NTT Application Security (VentureBeat, 2022).

Burnham (2021), señala que las aplicaciones web están expuestas a una gran variedad de ataques, entre los cuales destacan los siguientes:

- Cross-site scripting (XSS).
- Inyección SQL (SQLi).
- Falsificación de solicitudes entre sitios (CSRF).
- Ataques DDoS.

Por ello, los tres principales escenarios de ataque a aplicaciones web (SQL injection, local file inclusion y cross-site scripting) representan casi el 95% de los ataques, y a menudo se llevan a cabo a través de la Interfaz de Programación de Aplicaciones (API) (Verry, 2021).

Cross-Site Scripting

¿Qué es cross-site scripting (XSS)?

Cross-site scripting (XSS) se considera como una vulnerabilidad que aprovecha las fallas de seguridad que permite a los atacantes inyectar scripts ejecutables y maliciosos en una aplicación o sitio web del lado del cliente. Este tipo de scripts están conformados por código JavaScript, HTML o cualquier otro tipo de código que se pueda ejecutar en un navegador web (Subía, 2018).

Sgobba (2021), expone que este tipo de ataques principalmente se producen cuando la información se introduce en un sitio web por medio de una fuente que no es fiable; o cuando se validan incorrectamente los datos que ingresa un usuario. Estas acciones conllevan a diversas consecuencias para los usuarios, entre las cuales tenemos:

- Acceso a las cookies del usuario.
- Robo de credenciales.
- Recopilación de datos personales.
- Redirigir a páginas web maliciosas.
- Acceder al control del equipo de la víctima.
- Cambiar el contenido de un sitio web.
- Ejecutar ataques basados en navegador web.

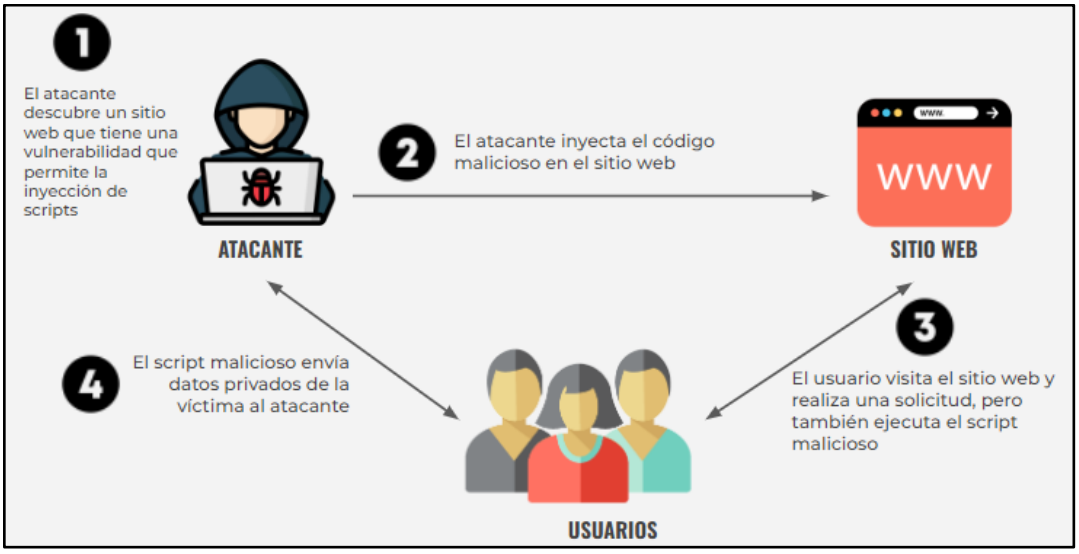
¿Cómo funciona un ataque XSS?

El XSS se produce debido a los fallos de seguridad en el HTML, JavaScript, AJAX, etc. En base a esto, un ataque XSS se produce cuando una persona visita un sitio web para buscar información o realizar una función en específico que involucre la obtención de datos por parte de dicho sitio, pero este ya ha sufrido un ataque de Cross Site Scripting. La víctima obtiene

contenido que puede contener código malicioso el cual es ejecutado en su navegador, lo cual lleva a comprometer la seguridad del usuario (Chalabe, 2019).

Figura 3

Funcionamiento de un ataque XSS



Nota. Esquema gráfico de un ataque XSS a un sitio web, en este escenario los usuarios ejecutan un script malicioso.

Tipos de ataques XSS

Gupta (2020), en su libro especifica que existen diferentes maneras de ejecutar un ataque XSS. Puede ser lanzado de tres formas diferentes y, por lo tanto, se puede clasificar en tres categorías:

- **XSS persistente o almacenado:** En este ataque, el atacante inyecta permanentemente el código malicioso en el servidor o base de datos. Después de esto, cualquier usuario que visite esa página web con el script inyectado se infecta por el ataque XSS. Es el ataque XSS más peligroso entre todos los tipos porque el atacante inyecta el código malicioso en el servidor una sola vez y luego afecta a un gran número de usuarios.
- **XSS reflejado:** se produce cuando el script malicioso está incrustado en un enlace generado por un atacante y se activa sólo cuando el usuario hace clic en el enlace. En

este caso, el script malicioso no se almacena en ningún sitio, sino que sólo se muestra en la página web en forma de URL o datos POST.

- **XSS basado en el DOM:** El ataque XSS basado en el Modelo de Objetos del Documento (DOM) es un ataque XSS del lado del cliente. El DOM permite al navegador procesar el contenido web representado por la página web. En este caso, el script inyectado es capaz de alterar la estructura del DOM. Si no fluye correctamente, entonces conduce a la fuga de información sensible.

Robo de cookies mediante ataques XSS

Las cookies son archivos de datos que los navegadores y páginas web generan automáticamente en el ordenador del usuario. La mayoría de las cookies contienen información personal del usuario, además permiten una navegación acorde a las preferencias del visitante (Kaspersky, 2022).

Un atacante puede poseer varios métodos para el robo de cookies, entre los más destacados se encuentra el ataque XSS en donde el atacante por medio de la ejecución de un script del lado del cliente puede obtener acceso de las cookies de la base de datos del navegador (Rodríguez et al., 2019). El robo de estas cookies permite que el atacante pueda obtener datos sensibles de los usuarios, como los datos de inicio de sesión, cuentas, detalles de tarjeta de crédito, etc (González & Zúñiga, 2017).

Tipos de cookies

Rodríguez et. al (2019) en su investigación describen los tipos de cookies más comunes que nos podemos encontrar en el internet, entre las cuales destacan:

- **Cookies de sesión:** Se caracterizan por ser cookies temporales, ya que se almacenan en la memoria de los navegadores y se eliminan al ser cerradas. Como su nombre mismo lo indica, estas almacenan información de acceso de sesión, por lo que podrían ser robadas para acceder a estos datos sensibles.

- **Cookies persistentes:** Permanecen en el navegador con una fecha de expiración, una vez alcanzada la fecha son eliminadas del equipo del usuario. En su mayoría tiene como finalidad rastrear el comportamiento e interacción del usuario para comprender sus preferencias.
- **Cookies propias:** Son creadas y gestionadas por el organismo que gestiona la página web, por lo cual solo este puede acceder a la información de la cookie que usualmente son utilizadas para análisis y métricas de uso de la página web.
- **Cookies de terceros:** Como su nombre lo indica estas cookies están relacionadas con dominios externos a la página web visitada. Están asociadas con elementos de terceros como etiquetas, elementos publicitarios, complementos sociales, etc. Las desventajas de estas cookies es que al ser gestionadas por terceros estas pueden ser filtradas o vendidas para aprovecharse de nuestros datos.
- **Cookies seguras:** Son transmitidas por medio del protocolo HTTPS, por su grado de protección son usadas en páginas gubernamentales, bancos, comercio electrónico, etc. Ofrecen un nivel alto de protección ya que viajan encriptadas mientras se da la comunicación de la página web con el navegador del usuario.
- **Cookies HttpOnly:** Se caracterizan por agregar una etiqueta en la cookie lo cual impide que scripts maliciosos accedan a los datos. Esta funcionalidad permite que se eviten ataques de tipo XSS.
- **Cookies zombis:** Son buscadas o creadas por atacantes debido a que pueden llegar hacer una amenaza para la privacidad y seguridad del usuario, ya que se vuelven a crear después de ser eliminadas. Otro inconveniente es que se almacenan en el dispositivo del usuario y no en el navegador.

Escenario de robo de cookies

A continuación, se presenta un ejemplo de un proceso que utilizan los atacantes para ejecutar scripts maliciosos con el fin de robar cookies de una página web (Bhardwaj, 2017).

Este escenario se presenta a partir de un sitio web que contiene una cadena manipulada en la respuesta enviada al usuario, para lo cual se sigue el siguiente proceso:

1. El atacante genera una URL que contiene una cadena de script maliciosa, el cual es enviado a la víctima.
2. La víctima realiza la solicitud de la URL manipulada del sitio web.
3. El sitio web incorpora la respectiva cadena maliciosa del URL en la respuesta.
4. El navegador de la víctima realiza la ejecución del script al momento de realizar la respuesta, enviando las cookies de la víctima al atacante.

Figura 4

Proceso para el robo de cookies por medio de ataques XSS

Nota. La figura representa el proceso que utilizan los atacantes para acceder a las cookies de las víctimas que utilizan una página web manipulada. Tomado de Mukesh Bhardwaj, 2017, itechhacks (https://itechhacks.com/xss-full-guide-tutorials/).

Herramientas para buscar vulnerabilidades en sitios web

Para establecer el análisis de vulnerabilidades dentro de los sitios web, es necesario la implementación de herramientas que proporcionen la información acerca de las amenazas a las que se encuentran expuestos los sitios web. Para ejecutar este tipo de análisis, la mayoría

de herramientas efectúan pruebas de caja negra, en donde no se le proporciona ninguna información adicional más que ciertos comandos y la dirección web (enlace). Además, cuando se establece un análisis de vulnerabilidades dentro de una página web, se toman en cuenta muchos aspectos dentro de su arquitectura, puesto que dicho análisis puede ejecutarse en el lado del cliente y del servidor. En este contexto en la Tabla 2 se presenta las siguientes herramientas:

Tabla 2

Análisis de herramientas de búsqueda de vulnerabilidades de sitios web

Herramienta	Características	Vulnerabilidades que analiza
Vega	Es un escáner de vulnerabilidades web de código abierto cuyo objetivo es identificar y validar ataques de inyección SQL, Cross Site Scripting (XSS) y otras vulnerabilidades (Aguas & Paredes, 2021). Está desarrollado en Java y es compatible con Linux, OS y Windows.	- Inyección SQL - XSS - Inyección de shell - Inclusión remota de archivos - Configuración de seguridad TLS/SSL.
WPScan	Es una herramienta gratuita que se enfoca en el análisis de vulnerabilidad web de Wordpress. Busca vulnerabilidades de aplicaciones web gestionadas con este CMS, así como también con los plugins y temas dentro del entorno de Wordpress (Kali, 2022).	- Contraseñas débiles - Versiones vulnerables - Desactualizaciones de parches de seguridad - Robo de información - Spam.
Joomscan	Es un proyecto de código abierto, el cual tiene como objetivo la detección de vulnerabilidades y fiabilidad en los despliegues de páginas web con Joomla CMS (Kali, 2022).	- Inyección SQL - LFI - RFI - XSS
Nikto	Es un escáner de código abierto que permite la búsqueda de vulnerabilidades en servidores web (Kali, 2022). Se puede usar para escanear varios tipos de servidores web.	- XSS - Ataques de fuerza bruta - Errores de configuraciones - Puertos abiertos
Parsero	Es una herramienta escrita en python utilizada para leer analizar archivos Robot.txt de sitios y aplicaciones web. Este permite recopilar información relacionada con la URL de la página analizada (Kali, 2022).	- Robo de información - Posible visualización del archivo robot.txt.

Herramienta	Características	Vulnerabilidades que analiza
Burp Suite Community Edition	Es una herramienta escrita en Java y desarrollada por la empresa PortSwigger, esta permite realizar pruebas de seguridad de aplicaciones web, URLs específicos y segmentos específicos de una aplicación. Es compatible con los sistemas operativos Linux, Windows o Mac (Campos, 2017).	- XSS - Inyección SQL - Robo de datos. - Configuración errónea - etc.
jSQL injection	Es una aplicación de código abierto la cual tiene como objetivo encontrar y explorar vulnerabilidades de tipo inyección SQL en base de datos (Kali, 2022). Es compatible con los sistemas operativos Windows, Linux y Mac OS.	- Inyección SQL - XSS
Websploit	Es un framework de código abierto utilizado para escanear posibles vulnerabilidades en aplicaciones web y redes (Kali, 2022).	- XSS - Man In The Middle - Ataques inalámbricos

Nota. Para dicho estudio se han tomado en cuenta 10 herramientas de análisis de vulnerabilidades utilizadas en el sistema operativo Parrot OS, no es un ranking.

Web scraping

¿Qué es web scraping?

Es una técnica para ext información o datos de la Word Wide Web (WWW), es decir páginas web, para guardarlos en una base de datos o sistema de archivos para su posterior consulta o análisis. Este proceso puede ser realizado manualmente por una persona o automáticamente por medio del uso de un bot. Por ende, lo más habitual es escribir un programa automatizado que consulte a un servidor web, solicite datos (normalmente en forma de HTML y otros archivos que componen las páginas web) y luego analice estos datos para extraer la información necesaria (Mitchel, 2015).

Herramientas para hacer web scraping

Existen varias herramientas que nos facilitan el proceso de extracción de datos de la web, como el uso de librerías de lenguajes de programación, el uso de APIs o frameworks tanto

comerciales como de código abierto. Teniendo en cuenta que existen varias herramientas, a continuación, se presentan las más populares:

Tabla 3

Análisis de herramientas para realizar web scraping

Herramientas	Característica	Licencia	Entorno
Scrapy	- Es un framework de rastreo web. - Genera exportaciones de feeds en formatos como JSON, CSV y XML. - Tiene soporte integrado para seleccionar y extraer datos de fuentes, ya sea mediante expresiones XPath o CSS. - Permite extraer datos de las páginas web de forma automática.	Open Source	Linux/Windows /Mac
Selenium	- Soporte de múltiples sistemas operativos - Admite múltiples navegadores - Compatibilidad con múltiples lenguajes de programación - Requiere menos recursos en comparación con otras herramientas de prueba de automatización.	Open Source	Linux/Windows /Mac
BeautifulSoup	- Librería de Python que se utiliza con fines de raspado web para extraer los datos de los archivos HTML y XML. - Crea un árbol de análisis a partir del código fuente de la página que se puede utilizar para extraer datos de una manera jerárquica y más legible.	Open Source	Linux/Windows /Mac

Nota. Se estableció un análisis comparativo entre las principales herramientas utilizadas para la realización de web scraping.

Bots con Automatización Robótica de Procesos (RPA)

¿Qué es el RPA?

La automatización robótica de procesos (RPA) se define como la automatización de procesos manuales de gran volumen, repetitivos, mediante la utilización de robots de software avanzados, también conocidos como "bots". La RPA permite a las organizaciones automatizar estas tareas como si las ejecutara una persona real en las aplicaciones. RPA podría considerarse como una tecnología no inteligente, debido a que los bots son programados por un desarrollador con el objetivo de simular las acciones de una persona capacitada al momento de realizar una tarea en específico (Blanca, 2021).

Dado que RPA es una solución basada en software y es independiente de la tecnología, puede implementarse para trabajar sobre una multitud de plataformas de TI existentes, incluyendo terminales de mainframe, aplicaciones web, aplicaciones de escritorio y sistemas de planificación de recursos empresariales (ERP). Un bot de RPA puede ser entrenado para capturar e interpretar datos para procesar una transacción, manipular datos o activar una respuesta dentro de una aplicación y comunicarse con otros sistemas (Ribeiro, 2021).

Beneficios de RPA

Según Asquith y Horsman (2019), RPA se puede utilizar ampliamente en diferentes aplicaciones o campos, por lo cual destaca un gran número de beneficios que ofrece en los procesos y resultados comerciales de una organización. En base a esto se pueden definir los siguientes beneficios:

- **Reducción de errores:** Los bots pueden manejar mejor que los seres humanos procesos que requieran un gran volumen de datos. Con una cuidadosa implementación de estos bots de software se producen muy pocos errores al completar las tareas en comparación con los humanos. Los bots de software no cometen errores humanos al realizar tareas repetitivas y son muy fiables (Yashodhan, 2021).

- **Mejor análisis:** Las organizaciones pueden recopilar datos importantes utilizando un RPA, que pueden ser aplicados para la toma de decisiones más acertadas. Gracias a la recopilación de RPA y la diferenciación de datos en campos separados, se mejora la toma de decisiones a nivel macro y micro.
- **Ahorro de costes:** El uso de RPA permite una drástica reducción de costes en una organización. La razón es que, con esta solución, se garantiza la eficiencia en la utilización de los recursos y se evita que la mano de obra se emplee en las tareas automatizadas.
- **Mejora de la seguridad:** Las fugas de datos y las violaciones ya son comunes, y la administración puede tener preocupaciones sobre la seguridad de dichos sistemas. Sin embargo, cuando su equipo gestiona cuidadosamente y define estrictamente los parámetros de RPA, el riesgo de fugas entre plataformas es relativamente menor.
- **Escalabilidad:** Como todas las tecnologías basadas en la automatización, RPA tiene un diseño que es reescalable ya que puede multiplicar fácilmente sus actividades realizando unos pocos retoques al proyecto. Además, RPA puede fusionarse con Machine Learning para ofrecer un escalamiento exponencial equiparable a la producción humana.
- **Productividad:** La productividad aumenta potencialmente debido al hecho de que los tiempos de ciclo del proceso "robot" son mucho más rápidos en comparación con los enfoques de proceso manual.

Tipos de RPA

No todos los robots son creados de la misma forma. Los bots manejan diferentes actividades, por lo cual se debe conocer qué tipo de automatización se adapta a nuestras necesidades, las cuales pueden clasificarse de la siguiente manera:

Automatización RPA asistida. En esta actividad asistida, el RPA automatiza las aplicaciones que se ejecutan en el escritorio de un usuario, generalmente con el propósito de

ayudarlo a completar en menos tiempo un proceso involucrado. Esto genera ahorro de costos y ofrece una mejor experiencia de usuario y cliente. Los bots están programados para trabajar junto a los humanos manejando ciertas tareas en cargas de trabajo más largas y complejas o cuando todo el proceso de extremo a extremo no puede ser totalmente automatizado, por lo cual, la automatización asistida es la más adecuada para tareas que se activan en puntos que son difíciles de detectar mediante programación. (Sotelo, 2018).

Automatización RPA no asistida. En este modo, los bots son autónomos y realizan sus actividades sin interacción humana. Sus acciones se desencadenan por eventos o son preprogramadas, debido a esto, la aplicación RPA funciona por sí sola, notificando al usuario sólo cuando existe algún error. Básicamente, los bots autónomos actúan con actividades manuales que siguen un patrón propio o un conjunto estructurado de pasos que se ejecutan siempre de la misma manera. La automatización no asistida puede funcionar ininterrumpidamente, automatizando partes de flujos de trabajo más complejos (Sotelo, 2018).

Herramientas RPA más utilizadas

Estas herramientas se encuentran en constante mejora o actualización a medida que las tecnologías de desarrollo van innovando a través de los años, logrando de esta manera una mayor capacidad de automatización. En base a esto se analizan las funciones y capacidades de 5 herramientas RPA más importantes.

UI.Vision. Se trata de una herramienta open source que permite realizar la automatización de procesos robóticos (RPA). Este software se compone de 3 elementos esenciales para automatizar entre los cuales destacan: Selenium IDE, automatización del navegador web y automatización de escritorio, compatible para Windows, Mac y Linux. UI.Vision RPA trabaja mediante una extensión en el navegador web, pero también realiza la automatización de aplicaciones de escritorio. Por sus características, se le considera una herramienta universal de automatización de tareas y procesos que conjuga lo mejor de la

automatización web con los conceptos actuales de automatización basada en Inteligencia Artificial (Taulli, 2020, pp 266-267).

Los comandos de pruebas visuales de la interfaz de usuario de UI.Vision permiten escribir pruebas visuales automatizadas, lo que convierte al software de RPA en la primera y única extensión de Chrome y Firefox (y Selenium IDE) con "ojos". Una gran ventaja de hacer pruebas visuales es que no sólo se comprueba uno o dos elementos a la vez, sino que se comprueba toda una sección o página en una declaración visual (Cady, 2022).

Robot framework. Es una librería que automatiza pruebas de código abierto, impulsado por pruebas de aceptación. Esta herramienta sigue diferentes estilos de casos de prueba impulsados por palabras clave, por el comportamiento y por los datos para escribir casos de prueba. Entre sus características está que proporciona soporte para bibliotecas externas, herramientas que son de open source que pueden ser implementadas para la automatización. La biblioteca más popular es Selenium Library, utilizada para el desarrollo web y las pruebas de interfaz de usuario (Tutorials Point, 2019).

Se encuentra disponible para todos los sistemas operativos (Windows, Linux y Mac). Robot Framework se caracteriza por tener una sintaxis de texto simple que se puede utilizar mediante Python o Java. Consta de múltiples bibliotecas y funciones, lo que abre la posibilidad de que puedan usarse en proyectos separados (Castro, 2020).

TagUI. Es una interfaz de línea de comandos para RPA que puede ejecutarse en cualquiera de los principales sistemas operativos. Es una herramienta con licencia open source que ayuda a automatizar rápidamente las tareas repetitivas o urgentes: los casos de uso incluyen la automatización de procesos, la recopilación de datos y las pruebas de aplicaciones web (da Silva et al., 2019).

Esta herramienta denomina a los tradicionales bots RPA como flujos, los cuales sirven para representar la ejecución de un proceso automatizado, que se ejecuta de manera directa o en diferido. TagUI pone énfasis en la simplicidad y naturalidad del lenguaje, uno de los

desarrolladores lo describió en una entrevista como que: "facilita la creación rápida de prototipos, la implementación y el mantenimiento de la automatización de la interfaz de usuario, siendo un desarrollador o no" (Blanca, 2021).

Automagica. La herramienta Automagica es propietaria con una versión opensource, cuyo código está disponible en GitHub. Desarrollada principalmente en el lenguaje Python, puede ser aprovechada por otras implementaciones de la comunidad. Entre las funcionalidades básicas se mencionan: la lectura de OCR, la extracción de texto de archivos PDF, la automatización de información en archivos word, excel, información recogida a través del navegador y la creación de procesos de automatización, también permite la interconexión con Google Tensorflow para el reconocimiento de imágenes y textos (Ribeiro, 2021, pp 55).

UiPath Community Edition. Es una herramienta capaz de automatizar cualquier actividad de escritorio o web, en cualquier entorno. Su funcionamiento se basa en arrastrar y soltar las actividades a ejecutarse. UiPath Community Edition permite utilizar tanto las librerías predefinidas, como reutilizar el código desarrollado por otros programadores RPA, lo cual permite ahorrar tiempo y coste de desarrollo (Castro, 2020).

La versión completa de UiPath es de pago, pero consta de una versión Community, la cual cuenta con menos características que la versión de pago, pero ofrece las suficientes opciones para implementar proyectos que requieran tareas sencillas de automatización (Castro, 2020).

Bots o sistemas para automatizar la búsqueda de vulnerabilidades

La búsqueda de vulnerabilidades juega un papel muy importante dentro de la seguridad de la información de una organización, por ende, es primordial establecer normas que permitan el testeo de la infraestructura tecnológica de la empresa cada cierto tiempo. Hoy en día, muchas de las empresas utilizan su infraestructura tecnológica en sistemas web, lo que facilita su mantenimiento, pero no se encuentran exentas de ataques informáticos.

Con el constante desarrollo de la tecnología, se apuesta mucho por la implementación de programas para automatizar las tareas repetitivas de análisis de vulnerabilidades, mejorando los tiempos de testeo dentro de una organización. En este contexto, es importante mencionar que existen ciertas herramientas que realizan la búsqueda de vulnerabilidades sin la intervención humana, lo que sin duda marca un nuevo hito en el desarrollo de la seguridad informática. Algunos bots y programas importantes son:

Software Mayhem. Mayhem es un software hacker creado a partir de una start-up (empresa emergente), que tiene como objetivo analizar y arreglar errores de manera automática en sistemas específicos de tipo comercial. Aunque en la actualidad se encuentra en etapa de entrenamiento, ha demostrado ser una herramienta muy potente, que pretende minimizar los ataques informáticos a los que se enfrentan las empresas (Simonite, 2017).

Según su creador David Brumley, menciona que cuando una máquina es atacada y vulnerada pueden pasar varios días, semanas o incluso meses para que los expertos en la seguridad puedan notar la vulnerabilidad y poder parcharla. Por ello con la implementación de un bot se puede parchar una vez sin que se afecte el resto de la infraestructura. En las pruebas que ha realizado Mayhem ha logrado detectar vulnerabilidades en el firmware de routers antes que los expertos puedan darse cuenta (Simonite, 2017).

Mechanical Phish. Es un sistema de razonamiento cibernético (CRS), que se basa específicamente en el marco de análisis binario ANGR (marco de análisis binario independiente), cuyo objetivo es analizar el código de un programa para la búsqueda de vulnerabilidades, generando exploits que parchean el software que se encuentra vulnerable tras realizar las pruebas de testeo. La explotación de vulnerabilidades por el uso de Mechanical Phish implica dos pasos; el primero que analiza y busca las fallas en el programa destinado y el segundo que analiza dichos fallos para modificarlos y crear exploits que tomen el control del programa logrando parchear la vulnerabilidad (Shoshitaishvili et al., 2018, pág. 12 - 22).

Programas para ejecutar tareas automáticas de web scraping

Como ya se ha mencionado, la automatización de procesos se ha ido posicionando en las labores de las empresas u organizaciones actuales, ya que permiten optimizar el tiempo en las tareas relacionadas con web scraping. A continuación, se analizan una serie de programas disponibles que permiten la ejecución de tareas de web scraping mediante el uso de procesos automatizados:

Parsehub. Es un programa gratuito compatible con varios sistemas operativos, el cual permite conectarse a cualquier sistema web para la respectiva extracción de datos por medio de técnicas de web scraping. Una de las características más importantes es que no requiere conocimientos de programación para implementar procesos o ejecutables (Pianchiche, 2021). Parsehub posee una interfaz intuitiva que permite que la extracción de datos sea sencilla. Los datos pueden ser extraídos en diferentes formatos como Excel, CSV o JSON (ParseHub, 2022).

Dexi.io. Es una solución basada en la nube que cuenta con una interfaz simple para extraer datos de cualquier sistema web mediante su incorporación con la tecnología de aprendizaje automático (Dexi.io, 2022). Sirve para construir y alojar bots web scraping. La extracción de los datos puede darse como datos JSON/CSV. La suite web proporciona varias funcionalidades web scraping como: resolución de CAPTCHA, socket proxy, llenado de formularios, etc. (Pianchiche, 2021).

Octoparse. Es un programa que ayuda a obtener rápidamente datos de cualquier sitio web sin técnicas de codificación y cualquiera puede usar esta herramienta para construir un rastreador en solo minutos, siempre y cuando los datos sean visibles en la página web. Octoparse ofrece herramientas para la extracción de información y transformarla en archivos visuales como HTML, Excel o TXT (Ruiz, 2021). Una de las características es que proporciona varias plantillas de web scraping para personas sin conocimiento de programación, por lo que

solo se necesita ingresar la dirección de la página y una palabra clave para comenzar con el proceso de extracción de datos (Octoparse, 2022).

Sistemas CAPTCHA

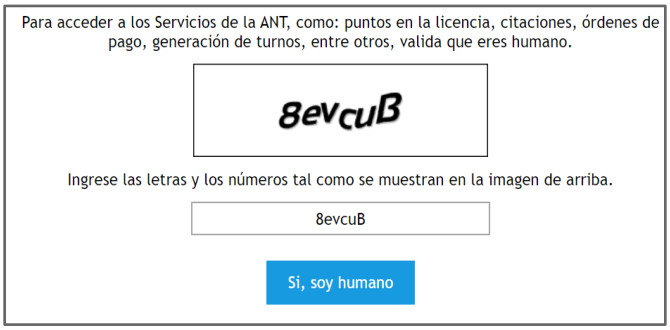
¿Qué es un CAPTCHA?

Las pruebas humanas interactivas denominada también como CAPTCHA (test de Turing completamente automático y público para diferenciar ordenadores de humanos), son sistemas que utilizan inteligencia artificial cuyo objetivo principal es establecer retos en imágenes, textos o sonidos que son fáciles de resolver para los humanos, pero que son complicados para una máquina, limitando con ello, el acceso de algún tipo de bot. Dicho de otra manera, un CAPTCHA implementa un método que permite diferenciar si quien intenta ingresar a un sistema web es un humano o una máquina, con el objetivo de brindar mayor seguridad a la información dentro del internet y contrarrestar en cierta medida los ataques de tipo DDoS y DOS (Callisaya Uchani, 2020).

Alejandro León (2019) menciona que los CAPTCHAs son una tecnología muy implementada en las páginas web, comúnmente en el proceso de registro con el fin de comprobar que el que realiza el formulario es un humano".

Figura 5

Representación gráfica de un CAPTCHA



Nota. Representación de un CAPTCHA basado en texto, para acceder a la consulta de citaciones de tránsito en la Agencia Nacional de Tránsito.

Historia de los CAPTCHAs

Uno de los principales precursores de los sistemas CAPTCHA fue Moni Naor en el año de 1996, cuando se propuso la realización de pruebas para distinguir entre las máquinas y los humanos. Sin embargo, los primeros sistemas CAPTCHA fueron diseñados en el año 1997 por Abadi Martin, Bharat Krishna, Andrei Broder y Lillibridge Marcar, creados principalmente para evitar que los diferentes tipos de robots añadiesen URLs a los nacientes motores de búsqueda. Cabe mencionar que estos CAPTCHA eran muy limitados y primitivos (Escabias, 2013).

El término “CAPTCHA” fue acuñado en el año 2000 cuyos precursores de este tipo de pruebas fueron Nicholas J. Hopper, Luis von Ahn, Manuel Blum, y John Langford, quienes crearon los primeros sistemas CAPTCHA y que posteriormente, fueron utilizados por Yahoo en sus páginas web. La idea de este tipo de sistemas es poder prevenir en cierta medida, el abuso que se venía dando en aquella época en el internet, en donde se fomentaba la publicación de manera anónima en los sitios web, así como las promociones comerciales y votaciones. De igual manera, en los inicios del 2000 existía una gran cantidad de ataques de denegación de servicios a las páginas web gubernamentales y sitios web de interés (Escabias, 2013).

Dentro del contexto de la evolución de los sistemas CAPTCHA, es importante recalcar que en sus comienzos utilizaban básicamente imágenes distorsionadas, donde un usuario debía observar las imágenes, descifrarlas y colocar su contenido. Pero este tipo de imágenes de distorsión presentaban problemas, ya que las máquinas podían descifrar aproximadamente el 90% de este tipo de pruebas, motivo por el cual, se intentó crear pruebas más complicadas para que las máquinas no las descifraran con facilidad, sin embargo, terminaron siendo muy complicadas para los humanos. Posteriormente con la evolución de la web, se fueron mejorando la precisión y creación de las pruebas hasta llegar a la actualidad (Escabias, 2013).

Importancia de los sistemas CAPTCHA

Los CAPTCHAs son retos de nivel cognitivo, que son implementados con el objetivo de evitar que se propaguen en cierta medida los correos electrónicos no deseados que muchas de

las veces tienen propósitos publicitarios. De igual forma cumplen un propósito muy importante como el de diferenciar el acceso a un software entre personas y máquinas, de esta manera se puede comprobar que se está interaccionando con un humano, logrando así proteger a los sitios web o programas informáticos del acceso de robots que puedan contener scripts automatizados o maliciosos (Escabias, 2013).

Entre las principales ventajas que ofrece la implementación de los CAPTCHA en un sistema web se encuentran:

- En el desarrollo de un formulario web, permite proteger la integridad de la información proporcionada en la encuesta, puesto que impide que robots, usados comúnmente por hackers, envíen y manipulen información falsa.
- El uso de CAPTCHAs en un sistema web de ventas online proporciona seguridad en una compra y transacción, brindando seguridad a los clientes.
- Permite resguardar los datos de los usuarios, así como su información personal, contraseñas y datos privados.
- Impedir que los ciberdelincuentes compren de manera virtual entradas a eventos o espectáculos para revenderlos.
- Crean una barrera para los ciberdelincuentes que intentan crear varias cuentas de correo para sus fines maliciosos.

Aplicaciones de los CAPTCHAs

En vista de todas las ventajas que ofrece la implementación de los CAPTCHAs, es importante mencionar las principales aplicaciones de esta tecnología.

- **Se puede implementar dentro de los blogs para prevenir el spam en comentarios.** En muchas ocasiones dentro de los blogs, aparecen programas que envían comentarios falsos con el objetivo de aumentar las filas al momento de establecer una búsqueda en el sitio web. Por ello, la implementación de un CAPTCHA, facilita que solo los humanos puedan ingresar comentarios dentro del blog (Saquinaula, 2013).

- **Muy utilizados en la protección de registros de usuarios.** Hoy en día muchas de las empresas manejan su comunicación con el uso de correos electrónicos. De alguna manera esto facilita su comunicación interna, pero a la vez puede estar expuesta a ciertos “bots” que comprometan el correcto funcionamiento de este servicio. La solución a este problema se puede solventar utilizando CAPTCHAs que controlen el abuso de scripts automatizados (Saquinaula, 2013).
- **Se implementa en la protección de direcciones de correos electrónicos.** Los Spammers son un tipo de personas que envían spam, los cuales rastrean la web para buscar direcciones de correos electrónicos válidas, con el fin de invadir dichos correos con spam. Al obligar a un usuario a solucionar un CAPTCHA sencillo antes de mostrar su dirección de correo electrónico permite ocultarlo de la mayoría de los rastreadores web.
- **Implementados en la precaución de ataques de diccionario.** El uso de un CAPTCHA dentro de un sistema que utilice contraseñas de acceso, puede prevenir que un “bot” pruebe contraseñas de un diccionario de datos al azar evitando que ingrese. Por ello, se puede configurar para que al realizar cierto número de intentos, se solicite la resolución de un CAPTCHA sencillo, controlando de cierta forma esta vulnerabilidad (Saquinaula, 2013).
- **Protección dentro de los motores de búsqueda**. Con la implementación de un CAPTCHA al ingreso de un sitio web se facilita la verificación de que solo sean humanos los que están accediendo, esto con el fin de evitar que un bot intente ingresar continuamente al sitio web, otorgándole mayor importancia a su posicionamiento en los motores de búsquedas (Escabias, 2013).
- **Protección en el agendamiento de citas de usuarios.** A menudo existen piratas informáticos que pueden utilizar “bots” para monitorizar un sitio web en específico, con el fin de reservar todas las horas posibles para una cita (médicas, eventos,

comisarías…) en un día, semana o mes en específico. Al realizar la comprobación con un CAPTCHA se puede controlar que los datos personales sean ingresados por un humano (Escabias, 2013).

Tipos de sistemas CAPTCHA

La importancia de la implementación de los CAPTCHAs, radica en su utilización dentro de aplicaciones web que requieran el ingreso de datos por parte del cliente. Por ello, el usuario se puede encontrar con diferentes tipos de CAPTCHA que debe resolver, dependiendo de la configuración del sitio web. Estos tipos de CAPTCHA son:

CAPTCHAs basados en texto. El CAPTCHA basado en texto, se forma a partir de la combinación de palabras del alfabeto con números decimales estableciendo una codificación alfanumérica distorsionada, de tal forma que resulta apenas reconocible para el humano. Para poder resolver este tipo de CAPTCHA, el usuario debe observar las letras y números que se le asignan y escribirlos en el campo requerido.

Para la implementación de estos tipos de CAPTCHAs, existen variantes como los que se mencionan a continuación:

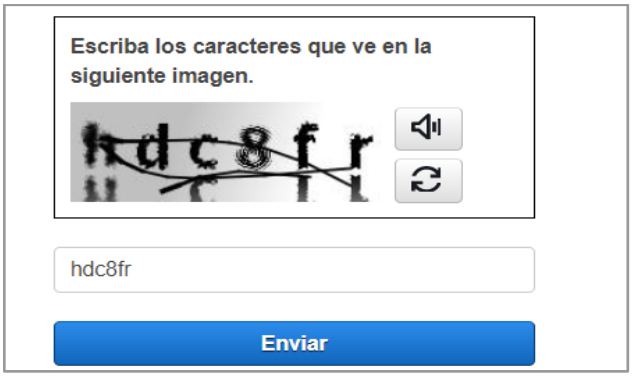
- **Gimpy:** Este tipo de CAPTCHA fue utilizado en sus inicios por Yahoo, con el objetivo de controlar a los bots que afecten las salas de chat. Para el funcionamiento, este CAPTCHA elige un conjunto de palabras en un diccionario pequeño el cual distorsiona y añade fondo o ruido a la imagen.
- **Gimpy–R**. Su implementación es muy similar al de Gimpy, el cual consiste en escoger tres palabras que aparecen en la imagen y que se encuentran distorsionadas. Teniendo en consideración los tipos de deformaciones que se implementan, la mayoría de los humanos pueden leer tres palabras del conjunto que aparecen en la imagen distorsionada, mientras que las máquinas actuales no pueden (Ahn et al., 2020).
- **Simcard’s**. Es mucho más avanzado que los anteriores, puesto que utiliza formas geométricas, arcos y curvas. Su funcionamiento se centra en escoger letras y números

de manera aleatoria, los mismos que se distorsionan y se añaden formas y arcos de fondo. Para pasar el test el usuario debe colocar de manera correcta las palabras centradas en la imagen (Saquinaula, 2013).

- **MSN CAPTCHA.** Se basa en resolver un reto en donde se presenta al usuario ocho caracteres alfanuméricos, con letras en tipografía serif de color azul oscuro. Estas letras por lo general presentan distorsiones de curvatura sobre el texto a nivel local y global, que frustran los intentos de reconocimiento de un bot (Vázquez, 2010).
- **Pessimal Print.** Por lo general este tipo de tecnología se fundamenta en los defectos encontrados en los textos cuando se realizan muchas copias, imprentas antiguas o escaneos. La idea fundamental es simular los errores que aparecen en la impresión. Aunque para el usuario final no represente una dificultad considerable para las máquinas pueden resultar un reto casi imposible. Cabe mencionar que esta técnica también posee una restricción, puesto que requiere que las personas se enfrenten al reto de entender y conocer el alfabeto Latino, así como del idioma en que se codifique el CAPTCHA (Vázquez, 2010).

Figura 6

Ejemplo de CAPTCHA basado en texto



Nota. Representación de un CAPTCHA basado en texto, para acceder a la cuenta principal de Amazon Web Services.

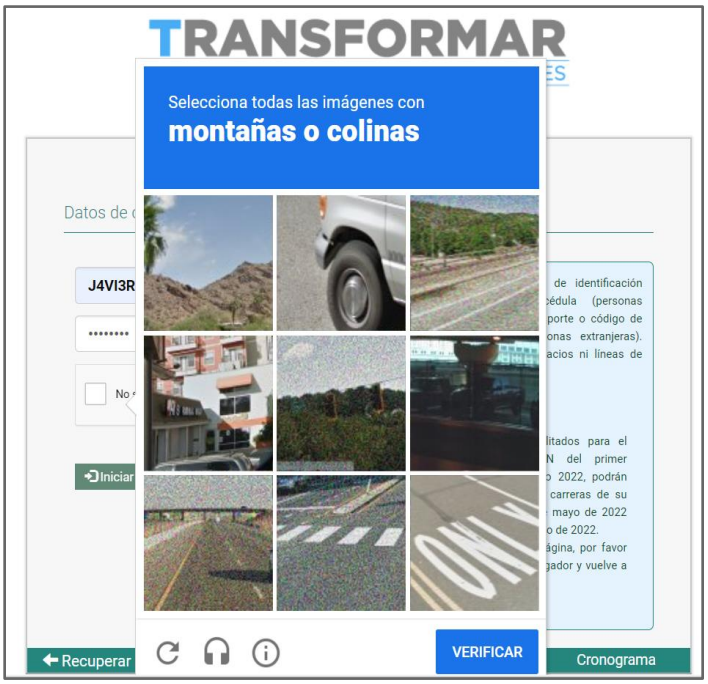
CAPTCHAs basados en gráficos. El CAPTCHA basado en gráficos se determina por implementar imágenes que son muy identificables a la vista humana. Estos CAPTCHAs se presentan como una alternativa a los CAPTCHAs basados en texto, que utilizan imágenes que representan hechos cotidianos colocados en mosaicos, donde el usuario debe escoger la imagen que represente motivos similares, analizar la relación semántica que existe entre las diferentes imágenes y escoger un motivo en específico.

A continuación, se mencionan las características de otras variantes de CAPTCHAs visuales:

- **Bongo.** Es un software que solicita a los usuarios que resuelvan un reto de reconocimiento visual, en donde se muestran dos series de bloques tanto en la derecha como izquierda, los cuales difieren el uno del otro. Luego se le muestra al usuario un único bloque, y se le solicita que indique a qué serie de bloques (derecha o izquierda) pertenece (Ahn et al., 2020).
- **Pix.** Es un programa que está formado por una gran base de datos de imágenes etiquetadas que distinguen un objeto en particular. El programa elige un objeto al azar, luego selecciona seis imágenes similares a dicho objeto en su base de datos, para presentarlas al usuario final a quien se le pregunta sobre dicha imagen (Ahn et al., 2020).
- **Imagination.** Es un sistema utilizado en la generación de CAPTCHAs basados en imágenes, desarrollado para ser robusto ante los ataques de bots y amigable al ofrecer una interfaz sencilla al usuario. En el primer test el usuario debe observar un mural de imágenes para localizar uno de los límites entre las imágenes compuestas y hacer clic sobre él. Por otro lado, el segundo test consiste en colocar una imagen distorsionada con un conjunto de frases que describen al objeto (Vázquez, 2010).

Figura 7

Ejemplo de CAPTCHA basado en imágenes



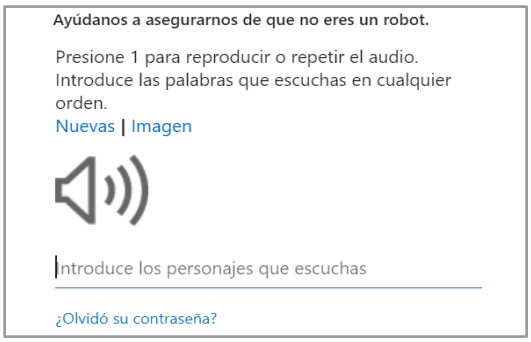
Nota. Representación de un CAPTCHA basado en imágenes, para acceder a la cuenta principal de la prueba Transformar para el acceso a la Educación Superior.

CAPTCHAs basados en audio. Muchas veces la implementación de CAPTCHAs basados en imágenes y texto pueden resultar dificultosas para las personas que sufren de alguna discapacidad visual, por ello, muchos de estos CAPTCHAs suelen venir acompañados de CAPTCHAs basados en audio o auditivos.

Según Saquinaula (2013), menciona que estos se basan en las destrezas de los humanos para analizar sonidos que pueden encontrarse distorsionados y ponerlos correctamente en su cuadro de texto. El usuario pasará el test únicamente si coloca correctamente las palabras o indicaciones que se mencionan en el CAPTCHA.

Figura 8

Ejemplo de CAPTCHA basado en audio

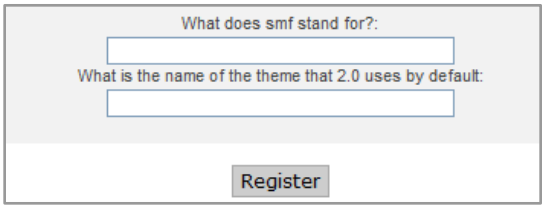


Nota. Representación de un CAPTCHA basado en audio, implementado en el inicio de sesión de Microsoft Outlook.

CAPTCHAs de problemas matemáticos y lógicos. Una alternativa muy viable al utilizar un CAPTCHA dentro de un sitio web, es la implementación de problemas de álgebra o adivinanzas, que impidan el acceso de bots al sitio. Los problemas matemáticos que se plantean son muy sencillos, lo que sin duda, pueden crear una facilidad para ser vulnerados. Por ello, una manera de contrarrestar esta problemática es establecer distorsiones gráficas o exigir el resultado del cálculo en su estado numeral, donde sea fácil identificar para el humano, pero difícil para un bot. Otra variante de este tipo de CAPTCHAs se establece al realizar preguntas de cultura general o que solicitan la ejecución de una tarea sencilla que se relacione con alguna temática en específico.

Figura 9

Ejemplo de un CAPTCHA de lógica

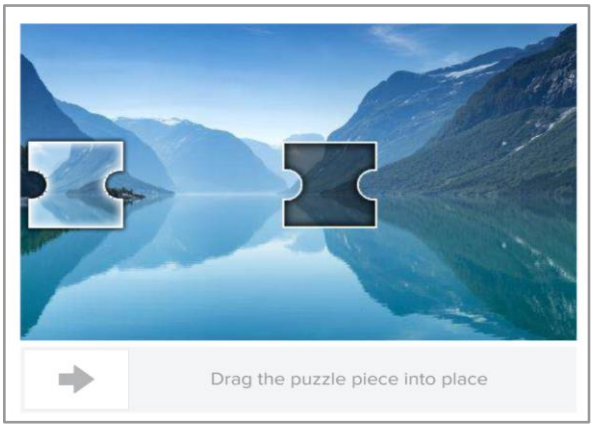


Nota. Representación de un CAPTCHA de lógica, extraído del foro de software libre Simple Machines.

CAPTCHAs lúdicos. Los CAPTCHAs lúdicos o también conocidos como los CAPTCHAs seguros, son una alternativa basada en los CAPTCHAs visuales. Su función principal consiste en que los usuarios den clic en la imagen correspondiente o que logren resolver el rompecabezas para cruzar el test. Por lo general este tipo de CAPTCHA usa una metodología sencilla y divertida para resolver los retos (Vázquez, 2010).

Figura 10

Representación gráfica de un CAPTCHA Lúdico



Nota. CAPTCHA lúdico de resolución de un rompecabezas implementado al momento de realizar una búsqueda personal en la plataforma web de Tik Tok.

reCAPTCHA. Por lo general este tipo de CAPTCHA es de carácter visual y auditivo por lo que es necesario mencionarlo en un apartado diferente. Es importante recalcar que los CAPTCHAs analizados, cumplen con su propósito de evitar el ingreso de una máquina hacia un sistema web. Pero con la implementación de nuevas tecnologías y desarrollo de aplicaciones cada vez más adaptadas a las necesidades de los usuarios, es necesaria su evolución. En este contexto nació un sistema más robusto denominado reCAPTCHA (Escabias, 2013).

Según Saquinaula (2013) menciona que el reCAPTCHA utiliza servicios web, que por lo general son suministrados al cliente desde un servidor remoto. Los reCAPTCHAS son los sistemas CAPTCHA más utilizados en los sitios web, ya que son gratis y ofrecen mayores

beneficios en su implementación. Además, brindan la facilidad de utilización para las personas que poseen algún tipo de discapacidad visual, lo que mejora la usabilidad de este sistema.

Para resolver el test, el usuario ingresa en el sitio web que posee el reCAPTCHA incrustado en su código. El navegador del usuario final solicita la resolución de la prueba, donde el reCAPTCHA proporciona al usuario el reto y un token de identificación del desafío. Por consiguiente, el usuario llena el formulario del sitio web, y se envía al respectivo servidor de aplicaciones junto con el token del desafío. En esta instancia, el reCAPTCHA comprueba la resolución del usuario y le proporciona una respuesta. Por último, si el desafío fue resuelto de manera correcta, proporcionará el acceso al sistema (aplicación, página web, servicio) al usuario, caso contrario, se le asignará un nuevo test el cual posee un token diferente de acceso (Saquinaula, 2013).

Figura 11

Representación gráfica de un reCAPTCHA

Nota. Representación gráfica de un reCAPTCHA implementado en la página de consulta de usuarios de bonos y pensiones del MIEES.

hCAPTCHA. Es un tipo de CAPTCHA visual el cual se presenta como alternativa viable, segura y robusta al reCAPTCHA. Según su página oficial, menciona que proporciona una detección más fiable de bots y más sencilla de resolver para los humanos. Para resolver este tipo de CAPTCHA basado en imágenes se debe observar la imagen modelo que proporciona el CAPTCHA y seleccionar las imágenes que aparecen en la matriz, para luego dar clic en el botón “Verificar” (hCAPTCHA, 2022).

Figura 12

Representación gráfica de un hCAPTCHA

Nota. Representación gráfica de un hCAPTCHA, que se considera como una alternativa más segura del reCAPTCHA.

Factores de vulnerabilidad de los CAPTCHAs

En base a lo planteado en la publicación de un sistema CAPTCHA establecida por von Ahn (2003), se han mencionado los primeros conceptos acerca de la implementación de un CAPTCHA, donde se definen cómo se encuentran escritos y el análisis de su estructura principal. Pero a pesar de ello, en dicha publicación no se definen los principales parámetros de cómo deben ser presentados al usuario final, para que sean más seguros ante posibles ataques de tipo OCR (Optical Character Recognition). El Reconocimiento de Caracteres Ópticos (OCR), permite identificar a partir de una imagen los caracteres y símbolos de un determinado lenguaje.

En este contexto, mediante los estudios realizados en años posteriores acerca de la seguridad de los sistemas CAPTCHAs, se identificó que con la aplicación de un método idóneo de segmentación en una imagen, es posible vulnerar y encontrar debilidades en este tipo de

sistemas, en base a su distorsión, construcción y caracteres. Por ello, es importante tener en cuenta las nuevas tecnologías de reconocimiento de imágenes (Maldonado, 2015, p. 55-67).

Según los estudios de Bursztein (2011), menciona que para que poder vulnerar un sistema CAPTCHA y poder descomponerlo dependiendo de su complejidad se pueden usar 5 pasos genéricos, tales como: pre-procesamiento, segmentación, post-segmentación, reconocimiento y post-procesamiento. Cada uno de estos pasos establecen un proceso definido que es utilizado para vencer un CAPTCHA, según la experimentación previa, se ha asignado una segmentación específica para cada sistema, separando las imágenes en trozos los cuales son vectorizados para luego ser analizados e ingresados en una red neuronal con un sistema de aprendizaje basado en un OCR.

Dentro del análisis de las vulnerabilidades dirigidas a los sistemas CAPTCHA es importante considerar también la investigación basada en formularios falsos de reCAPTCHA para atraer usuarios por medio de sitios web comprometidos de WordPress. Según los investigadores de Sucuri (empresa dedicada a la seguridad y protección de sitios web), han considerado diferentes páginas web desarrolladas en Wordpress, que establecen direccionamientos a terceras páginas en donde solicitan a los usuarios contestar formularios que poseen un sistema CAPTCHA falso. Este tipo de vulnerabilidad nace como una nueva campaña, donde los atacantes buscan aprovecharse de las vulnerabilidades que poseen muchos sitios web de Wordpress (Adminx5, 2022).

Cabe mencionar que la campaña de sitios web con CAPTCHAS falsos aprovecha la vulnerabilidad que poseen los complementos y temas que se utilizan al realizar un sitio web con Wordpress. Estas páginas web fueron infectadas por código JavaScript malicioso, el cual es incrustado en el script actual o en la cabecera del sitio para ser activado cada vez que se carga la página, lo que redirige a los usuarios al sitio web del atacante. Con esta propuesta, los investigadores también han obtenido como resultado que la mayoría del tráfico pasaba por un mismo subdominio de tipo *local[.]drakefollow[.]com* (Adminx5, 2022).

Ataques contra sistemas CAPTCHA

De igual manera como sucede con cualquier software, los sistemas CAPTCHA no se encuentran exentos de algún tipo de ataque informático, más aún si son muy utilizados para prevenir el ingreso de bots maliciosos a sitios web. Cabe mencionar que la mayoría de los ataques dedicados a estos sistemas, son de tipo spammer, que como se mencionó anteriormente, son ciberdelincuentes que realizan una constante búsqueda de registros y uso de múltiples correos electrónicos para la creación de spam (Vega & Vinasco, 2014).

En los últimos años con el fortalecimiento de técnicas y de tecnologías como la inteligencia artificial, han permitido que se mejoren la accesibilidad y seguridad de los sistemas CAPTCHA, pero de igual forma, estas tecnologías pueden resultar una amenaza para romper su sistema. Cabe destacar que muchas de esas tecnologías actúan de distinta forma, dependiendo del sistema con el que se haya desarrollado (Martinez & Prieto, 2009, p. 3-4). En este contexto, a continuación, se mencionan los principales ataques enfocados a los sistemas CAPTCHA.

Sistemas de reconocimiento de caracteres (OCR). Uno de los principales beneficios de la utilización de un OCR es la digitalización en formato imagen de documentos escritos. En la actualidad la mayoría de los CAPTCHA presentan al usuario final una serie de retos basados en imágenes, para resolverlo, deben escoger correctamente la figura, o escribir lo que visualizan en su cuadro de texto. Generalmente estas imágenes se muestran al usuario con distorsiones o con líneas en el fondo de la figura, con el fin de dificultar el acceso a un sistema automatizado (Martinez & Prieto, 2009, p. 3-4).

Cabe mencionar que los imperfectos y distorsiones también se presentan en los documentos digitales, por lo que los programas OCR deben analizar e interpretar dichos caracteres para ofrecer una visualización al usuario. Si bien esta tecnología cumple con su propósito de digitalización de documentos, también se ha convertido en una amenaza para la interpretación de CAPTCHA basados en texto (Martinez & Prieto, 2009, p. 3-4).

Sistemas de reconocimiento del habla. Uno de los principales avances dentro de la informática, ha sido la utilización de comandos de voz para la ejecución de tareas dentro de un software. Muchas de estas herramientas y procesadores de voz han facilitado las actividades cotidianas del humano. Pero a pesar de ello, el principal problema de este tipo de tecnologías es el establecimiento de una interfaz comprensible que sea capaz de manejar en su totalidad a un ordenador solo por comandos de voz (Martinez & Prieto, 2009, p. 3-4).

Una de las alternativas más viables al uso de CAPTCHA basados en textos e imágenes, es utilizando comandos de voz. A pesar de ello, estas pruebas pueden ser fácilmente vulneradas, puesto que, con la obtención de la grabación y la utilización de tecnologías basadas en la Inteligencia Artificial, los comandos de voz pueden ser analizados para luego ser utilizados en la resolución del desafío CAPTCHA (Martinez & Prieto, 2009, p. 3-4).

Analizadores sintácticos. Esta tecnología permite encontrar funciones de las palabras que posee un texto con el fin de aumentar las capacidades de traducción automatizada. Por ello, si se aplica una investigación alternativa para complicar la obtención de una respuesta correcta en la prueba, esta tecnología servirá para complicar el desafío y que se vuelva aún más difícil de resolverlo. Un claro ejemplo de ello, son los generadores de correos basura, los cuales utilizan analizadores sintácticos con el fin de no ser detectados por los filtradores de correo y además tener la capacidad de ser legibles (Martinez & Prieto, 2009, p. 3-4).

Estado del Arte

Para establecer la situación actual y analizar los trabajos relacionados con nuestra propuesta, se ha complementado con una revisión de la literatura científica, cuyo objetivo es generar nuevo conocimiento que apoye y fundamente la implementación de esta propuesta. Para esto, se parte de la idea de que los bots a desarrollar se basarán en la automatización de tareas de búsqueda (XSS) y análisis de vulnerabilidades (CAPTCHA) en sistemas web, por lo que se toma como punto de partida los temas establecidos en el marco teórico.

Revisión de propuestas similares

En cuanto al análisis de código y búsqueda de vulnerabilidades en sistemas web, se pueden encontrar varios métodos para identificar de manera más rápida y precisa dichas vulnerabilidades a través de procesos automatizados. En este contexto, en la Tabla 4, se presenta un análisis de los principales estudios enfocados en la detección automática de vulnerabilidades:

Tabla 4

Estudios relacionados con la detección automática de vulnerabilidades web

Código	Título	Link
TR1	Una herramienta inteligente y automatizada de descubrimiento de vulnerabilidades en el WCMS: El estado actual de la web	Link
TR2	Un analizador automático de vulnerabilidades para aplicaciones web	Link
TR3	Ruptura robusta en tiempo real de CAPTCHA de imagen utilizando el modelo Inception v3	Link
TR4	Un ataque de bajo costo contra el sistema hCAPTCHA	Link
TR5	Una extensión del navegador Google Chromium para detectar ataques XSS en sitios web basados en HTML5	Link
TR6	Cookie Scout: Un modelo analítico para la prevención de Cross-Site Scripting (XSS) utilizando un clasificador de cookies	Link

Nota. Estudios recopilados de bases de datos científicas (IEEE, Researchgate, Springer), que están relacionados con el análisis automático de vulnerabilidades en sistemas web.

Análisis comparativo

Los trabajos fueron seleccionados en base al área de investigación de nuestro proyecto, con el fin de obtener un marco referencial de los estudios que se han realizado a nivel mundial relacionados con el análisis de vulnerabilidades web actuales, estableciendo de esta manera la guía y el complemento del alcance de nuestra investigación.

Tabla 5

Análisis de los estudios similares identificados

	Resumen Propuesta	Herramientas utilizadas	Aporte	Limitantes
TR1	(Cigoj & Blazic, 2019) hace referencia a un proyecto de desarrollo de una herramienta para el análisis de vulnerabilidades web en páginas que se basan en un CMS en especial Wordpress. Este estudio hace énfasis que el uso de este tipo de gestores hace que se pase por alto ciertos parámetros de seguridad. por ende, es posible vulnerar algunos componentes de las aplicaciones web a través de ataques informáticos	- Plugins de WordPress - PHP	Herramienta web para identificar de manera automática vulnerabilidades en aplicaciones web que están gestionadas por un Sistema de Gestión de Contenido Web (WCMS).	Es una herramienta que está enfocada únicamente a servidores web de Wordpress y componentes asociados.
TR2	(Chen et al., 2020) establece que el aumento progresivo de las aplicaciones web ha provocado la necesidad de contar con herramientas que garanticen la seguridad de estas. En base a esto se especifica el desarrollo de un escáner de vulnerabilidades web.	- Librerías de python y la arquitectura (Navegador/Servidor). - SQLite - Pocsuite3	Escáner automático de vulnerabilidades web.	El escáner funciona a partir de una información recopilada limitada, ya que se basa en los datos almacenados por los autores del trabajo.
TR3	(Mittal et al., 2018) presenta el desarrollo de un modelo para la ruptura de CAPTCHAs basados en imágenes, en donde se utiliza la red neuronal Inception v3 como base principal para el análisis y reconocimiento de imágenes.	- Modelo Inception-v3 de la plataforma TensorFlow. - ImageNet - Spyder ID	Modelo basado en aprendizaje profundo para la ruptura de CAPTCHA basados en imágenes	Se debe extender el diccionario de datos para obtener mejores resultados.

	Resumen Propuesta	Herramientas utilizadas	Aporte	Limitantes
TR4	(Hossen & Hei, 2021), este estudio se centra en establecer un sistema automatizado para la resolución de hCAPTCHA, el cual se encuentra construido mediante un sistema de red neuronal profunda bajo el modelo ResNet-18, con el fin de realizar la clasificación de las imágenes.	- ImageNet - ResNet - Puppeteer	Sistema automatizado para la resolución del hCAPTCHA basado en imágenes.	El modelo de aprendizaje profundo sólo puede predecir en base a ciertas categorías de imágenes de los hCAPTCHA.
TR5	(Sivanesan et al., 2018) describen la implementación de una extensión para la búsqueda de vulnerabilidades web, mediante el análisis de etiquetas y atributos de HTML5 que pueden ser utilizados maliciosamente para realizar ataques a los sistemas web, pudiendo detectar si un sitio web está vulnerable a ataques XSS.	- HTML, CSS, JavaScript. - JSON.	Extensión de navegador para el seguimiento de vulnerabilidades web.	Es una extensión que se basa netamente en el análisis de etiquetas y atributos HTML 5.
TR6	(Rodríguez et al., 2018) proponen una solución para el análisis de vulnerabilidades de tipo XSS en páginas web, para lo cual plantean un modelo analítico en donde su funcionamiento se basa en el análisis de las cookies que generan las páginas web, para determinar si son propensas a ataques XSS.	- Python - Browser Exploitation Framework (Beef)	Modelo analítico para prevenir ataques XSS en base al análisis de las cookies de las páginas web.	No se ha analizado el historial real de un usuario para determinar su vulnerabilidad a ataques XSS.

Nota. Para el objeto de estudio de la Tabla 5, se tomó en consideración los trabajos más relevantes dentro de la investigación y se analizaron sus limitantes para mejorar nuestra propuesta de tesis.

Capítulo III

Metodología, diseño e implementación de los Bots

Introducción del capítulo

El propósito de este capítulo es describir la metodología aplicada en la implementación del presente proyecto, resaltando el análisis comparativo de las herramientas RPA con licenciamiento y open source, y seleccionando la herramienta de automatización de procesos que más se adapte a las necesidades de desarrollo de los bots. De igual forma, se ha justificado la selección de las herramientas RPA en base a los criterios y ventajas proporcionados. Se explica el diseño y desarrollo del bot cuyo objetivo será vulnerar los sistemas CAPTCHA, con el uso de tecnologías de visión artificial basadas en la nube. Así también, se explica el diseño y desarrollo del bot implementado para la búsqueda de vulnerabilidades web de tipo cross-site scripting en base al análisis de las cookies y el historial de usuarios.

Metodología

Análisis comparativo de herramientas RPA

Para la evaluación y análisis de las herramientas RPA se han considerado los resultados obtenidos por parte de las dos consultoras más importantes de las tecnologías actuales, Forrester y Gartner. Estas empresas consultoras y de investigación, presentan datos estadísticos de las diferentes soluciones tecnológicas y productos del mercado que implementan nuevas tecnologías. Forrester es una empresa que realiza investigaciones para acelerar el crecimiento de los clientes/empresas por medio de asesorías. Por otro lado, Gartner es reconocida mundialmente por realizar investigaciones enfocadas a tendencias tecnológicas, retos, innovación, liderazgo y otros temas (Navarrete, 2021).

Principales proveedores de RPA por Gartner. La manera más confiable de determinar qué tan recomendada y utilizada es una herramienta tecnológica, es utilizando el

Cuadrante Mágico de Gartner. Gartner (2021) menciona que un Magic Quadrant es una herramienta gráfica que presenta los resultados finales de la investigación de una tecnología específica, proporcionando una visión panorámica de las posiciones en las que se encuentran los competidores de dicha tecnología.

Para que la obtención de los resultados sea efectiva, Gartner establece como mínimo las siguientes capacidades: bajo código (Low-Code) para crear scripts de automatización, compatibilidad con las aplicaciones de la empresa, administración, orquestación, supervisión, configuración y seguridad de las aplicaciones. Todo esto complementado con la opinión de expertos dentro del área y las valoraciones que aportan los usuarios, permiten que los resultados finales sean muy confiables al momento de determinar y escoger la herramienta indicada (Gartner, 2021).

Según el informe de Gartner (2021), menciona que de los 10 principales proveedores de las herramientas RPA, el 70% representan casi toda la totalidad de dicho mercado, los cuales se encuentran repartidos en los 4 cuadrantes que representan el Cuadrante Mágico de Gartner. Cada Magic Quadrant utiliza un gráfico de dos ejes, el eje vertical hace énfasis al análisis de mercado, mientras que el eje horizontal proporciona información de la habilidad de ejecución. De igual manera los ejes dividen a la gráfica en 4 cuadrantes competitivos, en donde se ubican los proveedores tecnológicos a los cuales se les denominan de la siguiente manera:

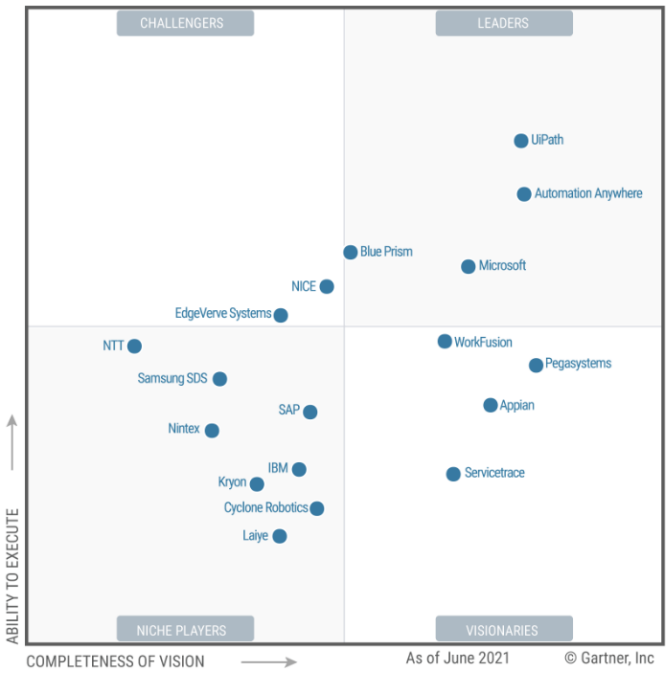
- **Visionarios:** Enfocado a fabricantes que entienden a dónde se dirige el mercado pero que aún poseen una capacidad de ejecución limitada.
- **Jugadores de Nicho:** Fabricantes que se centran en un segmento de mercado pequeño y que no poseen una buena capacidad para la innovación. Además poseen una limitada habilidad para hacer, motivo por el cual, no superan la media.
- **Retadores:** Enfocado a fabricantes que se desenvuelven en un segmento grande de mercado por la madurez en sus productos y que poseen una buena comprensión de hacia dónde se dirige el mercado.

- **Líderes:** Fabricantes que tienen muy buena visión de hacia dónde se dirige el mercado y se encuentran bien posicionados para el futuro. Además, han desarrollado una herramienta tecnológica muy sólida y madura con un producto de alta calidad y valoración (Gartner, 2021).

En este contexto, analizando las ubicaciones del Cuadrante Mágico de Gartner, se ha identificado la distribución de los principales proveedores de tecnologías RPA en el mercado. En la Figura 13 se visualiza esta distribución y los resultados obtenidos en dichos estudios. En donde destaca que Gartner no respalda a ningún servicio, proveedor o producto descrito en las encuestas realizadas.

Figura 13

Cuadrante Mágico para Automatización Robótica de Procesos (RPA)

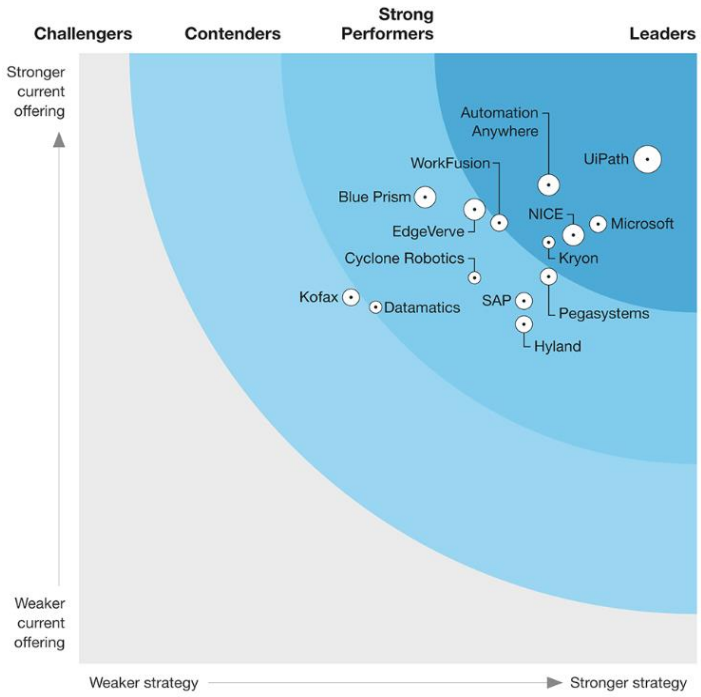


Nota. Análisis de los principales proveedores de RPA según la empresa Gartner, encuesta realizada en 2021. Cuadrante Mágico de Gartner, por Gartner, Julio 2021, Recuperado de (https://www.gartner.com/en/documents/4004033).

Por otro lado, si se observa la información en otra encuestadora sobre los principales proveedores líderes de RPA en el mercado, se obtiene otra perspectiva de análisis. Según el informe detallado por “The Forrester Wave” en el primer semestre del 2021, se menciona que las tecnologías RPA han adquirido un crecimiento exponencial debido a los procesos virtuales que implementan las empresas como consecuencia de la pandemia del Covid19. En dicho informe también se identificó a los proveedores líderes de tecnologías RPA, mediante la evaluación de sus 3 principales categorías como: ofertas actuales, estrategia y presencia de mercado. Dichos resultados son mostrados en la gráfica de Onda de Forrester, donde se determinó a UiPath como solución RPA líder del mercado, coincidiendo con la información detallada en el Cuadrante Mágico de Gartner. En este contexto la Figura 14 detalla los resultados obtenidos en dicha evaluación.

Figura 14

Identificación de los Líderes RPA en la Onda de Forrester



Nota. Análisis de los principales proveedores de RPA según la empresa The Forrester Wave en el 1° trimestre de 2021, por Forrester Wave Robotic Process Automation, Q1, 2021.

La razón por la que se toman como referencia los resultados obtenidos en el Cuadrante Mágico de Gartner y las Ondas de Líderes de Forrester, es por la garantía que ofrecen sus estudios y los altos estándares que exigen las tecnologías RPA. Dichas consultoras han demostrado tener éxitos con las hojas de rutas que toman las empresas tecnológicas en el futuro, ya que pasan por una rigurosa auditoría donde se califican muchas variables. Además, es bien visto que las empresas líderes ofrecen soluciones informáticas sólidas y maduras con una gran demanda de clientela, porque apoyan los requisitos de un amplio mercado (Navarrete, 2021).

Comparativa general de las herramientas RPA. Analizados los resultados obtenidos en el Cuadrante Mágico de Gartner y la gráfica de Ondas de Líderes de Forrester Wave, se concluye que los dos principales proveedores de tecnologías RPA son UiPath y Automation Anywhere. Se ha considerado una herramienta adicional que se encuentra en el rol de un líder naciente, como es el caso de Blue Prism. En la Tabla 6 se presenta un análisis comparativo de los 3 principales proveedores de herramientas RPA mencionados. El propósito de dicho análisis es obtener una noción más clara de la herramienta que mejor se adapte a las necesidades planteadas en los objetivos específicos del presente estudio.

Tabla 6

Comparativa general de las herramientas RPA líderes del mercado

Atributos/RPA	Uipath	Automation Anywhere	Blue Prism
Características	- Herramienta para implementar bots de tipo asistidos y no asistidos, además de acoplarse con tecnologías de otros proveedores como Java, Microsoft, Oracle. - Se conforma de tres módulos (UiPath Studio, UiPath Robot y UiPath Orchestrator),	- Combina varias herramientas como RPA, IA y análiticas para mejorar el desempeño de los bots tanto en las tareas de front y back office. - Se ha ido acoplando a las tendencias,	- Es una herramienta desarrollada bajo el lenguaje Java. - Cuenta con un diseñador gráfico con funciones de arrastrar y soltar los módulos para estructurar tareas complejas. - Posee bastante flexibilidad para integrarse con

Atributos/RPA	Uipath	Automation Anywhere	Blue Prism
	que conforman el desarrollo y funcionamiento de los bots.	ofreciendo una interfaz sencilla de uso (arrastrar y soltar.)	aplicaciones empresariales.
Beneficios	- Facilidad de uso. - Ejecución en la nube que permite ejecutar las tareas desde cualquier lugar. - Creación de tareas con procesos complejos como IA, Machine Learning, Procesamiento de Lenguaje Natural (PLN), Visión Artificial. - Posee una extensa documentación.	- Permite la ejecución de técnicas o algoritmos de IA, PLN, Redes Neuronales Artificiales, etc. - Estabilidad y escalabilidad en los proyectos e integraciones.	- Alta velocidad de ejecución. - Facilidad de uso en tareas sencillas. - Ofrece una alta escalabilidad en los proyectos.
Desventajas/ Limitaciones	- Solo puede ser utilizada en un entorno Windows. - No es posible visualizar el código mientras se va ejecutando. - Requiere de un proceso más lento y complejo en la etapa de debugeo.	- Posee una versión de prueba que solo está disponible por 30 días. - Aunque dispone de una facilidad de uso se requiere de un mayor aprendizaje.	- Tiene un alto costo de licencias. - No cuenta con una versión gratuita. - Curva de aprendizaje alta. - Solo automatización back office.
Lenguajes que implementa	- C# - JavaScript - .NET Visual Basic - C++ - VBScript	- C#	- C# - NET Visual Basic

Nota. Para la realización del análisis se consideraron las principales herramientas proporcionadas por los estudios realizados en Gartner y Forrester Wave 2021.

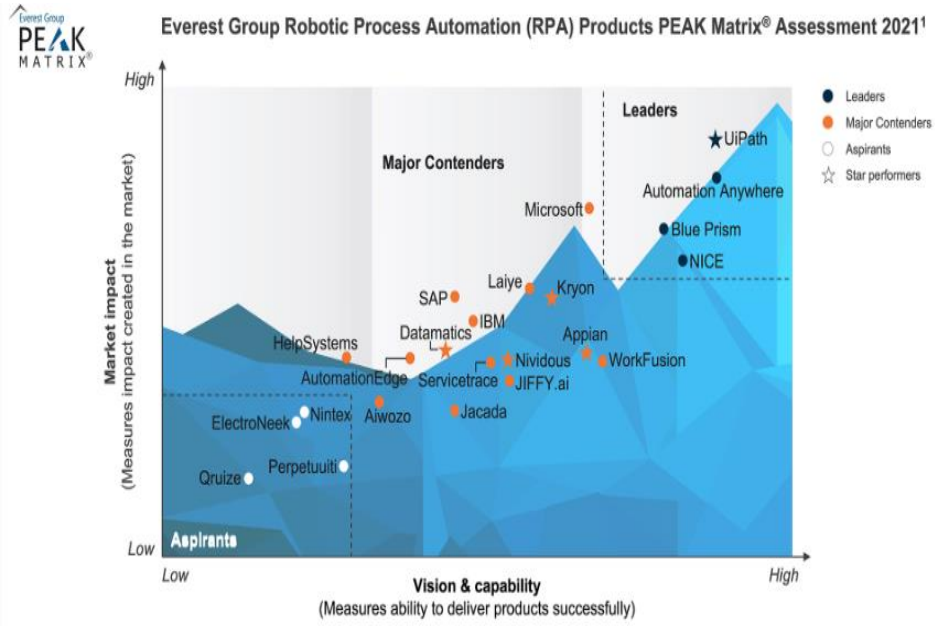
En la Tabla 6 se observa un análisis referencial de las características más sobresalientes de cada herramienta y la posible relación con la automatización de tareas de búsqueda de vulnerabilidades en sistemas web. Es importante mencionar que el primer objetivo del estudio se centra en la búsqueda de vulnerabilidades en los sistemas CAPTCHA basados

en imágenes, por lo que, la herramienta RPA debe poseer un módulo de análisis de imágenes. Por otro lado, el segundo objetivo se centra en la ejecución de web scraping en páginas web para buscar vulnerabilidades de tipo cross-site scripting (XSS), motivo por el cual, la herramienta RPA debe poseer configuraciones y módulos que permitan realizar este “raspado” en páginas web dentro de un navegador.

Para fundamentar la selección de las tres herramientas analizadas previamente, se estudiaron las investigaciones realizadas por el Grupo Everest, quienes mediante el desarrollo de su informe “PEAK Matrix" especifican las principales categorías de los proveedores RPA, luego analizan el panorama competitivo de las herramientas, para finalmente, medir las capacidades que poseen. En su informe, se ratifica el liderazgo y los resultados obtenidos del análisis de las tres herramientas seleccionadas inicialmente, tal como se observa en los resultados presentados en la Figura 15 (Everest Group, 2021).

Figura 15

Identificación de los líderes RPA según Everest Group



Nota. Análisis de los principales proveedores de RPA según Everest Group, donde se recalca a las empresas líderes dentro del mercado, por Everest Group, 2021, PEAK Matrix Assessment .

Con esta identificación, justificación y selección de los tres proveedores líderes de RPA, es importante recalcar sus características técnicas.

Tabla 7

Resumen técnico de las herramientas RPA líderes del mercado

Característica	UiPath	Automation Anywhere	Blue Prism
Aprendizaje	Diseño Visual	Desarrollo básico	Control y desarrollo visual
Reutilización	Si	Si	Si
Precisión	Buena en Citrix	Buena en todo aspecto	Buena en automatizar escritorio y web
Robots	Robots de front office y back office	Robots de front office y back office	Solo robots de back office
Escalabilidad operativa	Despliegue en proyectos pesados	Despliegue limitado a gran escala	Buena velocidad de ejecución
Grabadores	Si	Si	No
Arquitectura	Orquestador basado en web	Cliente-Servidor	Cliente-Servidor
Acceso	Navegador y/o acceso móvil	Basado en aplicaciones	Basado en aplicaciones
Proceso de diseño	Procesos visuales	Basado en script	Procesos visuales
Tecnología Base	Microsoft-sharepoint, elasticsearch, kibana	Microsoft	C#
Fiabilidad	Moderado	Elevado	Alto
Precios	Community edition gratuita	Licencia con versión gratuita limitada	Hace falta una licencia de alto coste
Certificación y formación	Certificación y formación online gratuitos	Lanzado recientemente documentación en crecimiento	Programa de certificación con coste

Nota. Resumen de las principales características técnicas de las herramientas líderes según el informe Gartner y Forrester Wave. Tomado de: Reina Romero, M.J., 2021, Universidad de Sevilla, Recuperado de (https://hdl.handle.net/11441/126691).

Comparativa de las herramientas RPA Open Source. Cabe recalcar que los resultados obtenidos por parte de las consultoras Gartner y Forrester Wave se enfocan en soluciones tecnológicas con licenciamiento, donde el cliente/empresa paga por la utilización de estas herramientas. Estas consultoras ofrecen una idea de cómo se encuentra el mercado en la tecnología mencionada.

También es importante mencionar a las herramientas y tecnologías RPA open source (software de código abierto), ya que estas pueden ofrecer una alternativa más viable para la automatización de procesos empresariales. Muchas de estas herramientas cuentan con una amplia comunidad que trabaja para mejorarlas constantemente. En ese contexto, en la Tabla 8, se muestra un análisis comparativo de las herramientas RPA open source y adicional a ello, se menciona la alternativa de UiPath Community Edition del proveedor UiPath como solución gratuita, para contrastar con las demás herramientas.

Tabla 8

Comparativa de las herramientas RPA Open Source

Herramienta	Características	Beneficios	Desventajas/ Limitaciones	Lenguajes
Robot Framework	- Proporciona soporte para bibliotecas externas, herramientas que son de código abierto y que pueden ser utilizadas para la automatización. - Se caracteriza por tener una sintaxis de texto simple y se puede utilizar mediante Python o Java.	- Fácilmente extensible. - Funciona en entornos web y móviles. - Obtención de informes de alta calidad. - Requiere de conocimientos básicos de programación.	- Por su usabilidad no nos permite implementar bucles anidados. - No cuenta con una depuración integrada.	- Python - Java

Herramienta	Características	Beneficios	Desventajas/ Limitaciones	Lenguajes
UI.Vision	- Es una herramienta open source que automatiza procesos en aplicaciones web y de escritorio. - Trabaja mediante una extensión en el navegador web, pero también puede realizar la automatización de aplicaciones de escritorio. - Es compatible con los sistemas operativos Linux, Windows y Mac.	- Proporciona tecnologías de reconocimiento de imágen y texto. - Facilidad en la migración de datos. - Integridad con otras aplicaciones por medio de una API. - Posee documentación y comunidad activa.	- Requiere de conocimientos de programación, en especial de python.	- Python
TagUI	- Es una interfaz de línea de comandos para RPA que puede ejecutarse en cualquiera de los principales sistemas operativos. - Nos permite implementar soluciones fácilmente mediante una sintaxis similar al lenguaje natural.	- Integración con R y Python para tareas de IA o Machine Learning. - Comunidad y documentación activa.	- Se requiere de conocimientos de programación al momento de implementar soluciones complejas.	- JavaScript - Python
UiPath Community Edition	- Es una herramienta la cual se basa en arrastrar y soltar las actividades que se quieran ejecutar. realizadas por otros usuarios	- Permite implementar proyectos. escalables. - Facilidad de uso.	- Tiene funciones limitadas en comparación con su versión de paga. - Limitaciones en proyectos complejos.	- C# - JavaScript - VB.NET - C++ - VBScript

Nota. Para este análisis se consideraron las herramientas RPA de código abierto (Open Source) más utilizadas en el mercado.

Selección del software RPA UiPath en su versión Community Edition. Una vez analizados los principales proveedores de RPA líderes en el mercado, se selecciona la

herramienta para desarrollar los bots y automatizar las tareas propuestas de búsqueda de vulnerabilidades en sitios web. Según el informe de las dos principales consultoras Gartner (Gartner, 2021) y Forrester (Schaffrik, O'Donnell, Lu, Kortenska, & Lynch, 2021) publicados en año 2021, determinaron que UiPath se colocó como la solución RPA número 1 del mercado y una de las más utilizadas por las empresas/clientes (Navarrete, 2021).

Figura 16

Logo del software RPA UiPath líder en el mercado

Nota. UiPath posee muchas características de las versiones pago, pero con ciertas limitaciones. UiPath RPA, 2022, Recuperado de (https://www.uipath.com/es/)

UiPath ganó mucho renombre precisamente por la gran capacidad de automatizar tareas y procesos dentro de aplicaciones de escritorio y aplicaciones web, todo ello, sumado a la facilidad de uso de su interfaz hacen que su curva de aprendizaje sea muy rápida. El entorno de desarrollo de UiPath utiliza diagramas muy visuales que reconocen objetos gráficos a través de una API que automatiza sistemas de manera uniforme, por lo que su funcionamiento es similar en cualquier plataforma (Reina R., 2021).

Para la implementación del presente proyecto de tesis, se consideró como herramienta RPA al proveedor líder en el mercado, UiPath en su versión Community Edition. Se utilizó esta edición, debido a que la implementación de los bots que automaticen las tareas de búsqueda de vulnerabilidades en sitios web no requiere de procesos muy avanzados, por lo que adquirir una edición de UiPath Enterprise Edition sería innecesario. Además, la versión de UiPath Community Edition se ajusta a las necesidades de desarrollo para quienes inician en el mundo de la automatización robótica de procesos.

De igual forma, es importante mencionar que la Edición Community de UiPath es completamente gratuita, con herramientas muy similares a las versiones de pago, pero con la diferencia de que no posee soporte de la empresa, motivo por el cual compensa dicha desventaja con la gran cantidad de videotutoriales, bases de conocimiento y guías de usuario oficiales de UiPath. En este contexto, en la Tabla 9, se presentan ciertas características que permiten entender mejor las cualidades que ofrece la versión UiPath Community Edition frente a la versión Uipath Enterprise.

Tabla 9

Comparativa de UiPath Community vs Uipath Enterprise

Ediciones de UiPath		
Community	**Enterprise Server**	**Enterprise Cloud**
Utilizado por desarrolladores y pequeños equipos que inician su viaje en la automatización.	Utilizado en el despliegue de tareas en empresas y grandes organizaciones.	Utilizado en despliegues empresariales en la nube y empresas de cualquier tamaño.
Versión siempre GRATIS	Prueba gratuita por 60 días	Solo por licencia
- 2 módulos Studio para el diseño de la automatización. - 3 robots. - Orquestador alojado en la nube. - Soporte solo por foros y videotutoriales. - Acceso a la academia UiPath.	- Módulos Studios ilimitados para el diseño de la automatización. - Robots ilimitados. - Orquestador on-premises (forma local y servidores de la empresa). - Soporte de primera calidad. - Escala a medida que crece la empresa. - Actualizaciones autogestionadas.	- Módulos Studios ilimitados para el diseño de la automatización. - Robots ilimitados. - Orquestador alojado 100% en la nube. - Soporte de primera calidad. - Escala a medida que crece la empresa. - Actualizaciones constantes, siempre al día. - Gestión centralizada del acceso de los usuarios.

Nota. Análisis comparativo de la Edición Community frente a la Edición Enterprise de UiPath. Adaptado de UiPath Community Forum, por Daniel Mitchell, agosto 2019, Recuperado de (https://bit.ly/39RCweu).

Como se observa en la Tabla 9, se destaca que la versión Community Edition posee acceso ilimitado a la academia de UiPath que es una plataforma de formación online con muchos cursos gratis para aprender a usar la herramienta. Aunque la edición Community de UiPath sea limitada en el tema de desarrollo de robots, posee muchas otras características como las descritas a continuación:

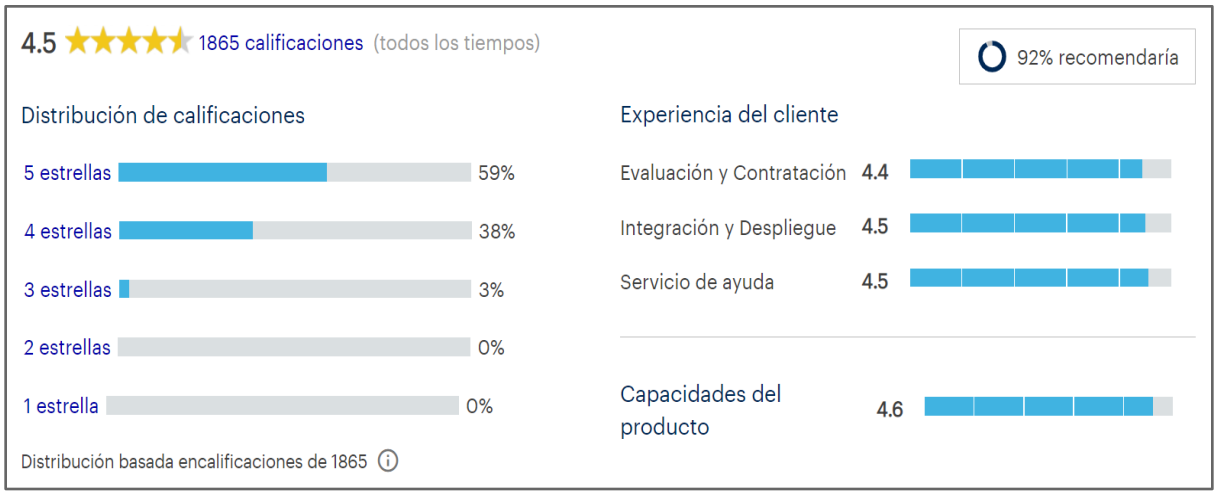
- Permite la automatización de cualquier aplicación del sistema operativo Windows, imitando la manera en cómo una persona interactúa con las aplicaciones.
- Posee e implementa una solución de screen scraping más completa y avanzada del mercado, cuyo funcionamiento es compatible con cualquier aplicación en menos de 16 milisegundos.
- Implementa uno de los mejores grabadores de macros cuya automatización es independiente del tamaño de la pantalla e independiente de su resolución.
- Permite una extracción de contenido web muy fiable, puesto que puede obtener datos en cualquier formato para luego exportarlos en datos que sean necesarios sin necesidad de programación.
- Permite una automatización web 100% precisa, ya que su grabador integrado proporciona la lectura y activación de funcionalidades compatibles con el navegador por medio de la instalación de una extensión web.
- Posee una amplia comunidad de desarrolladores y una gran cantidad de documentación y videotutoriales que ofrece tanto en los foros de UiPath, como en la información de la web (UiPath Community Edition, 2022).

Algo que hace más sobresaliente a UiPath es el gran apoyo y respaldo que recibe de los clientes/empresas, puesto que ha logrado consolidarse como la herramienta más completa en la automatización de procesos repetitivos de la gestión empresarial. Según las reseñas y calificaciones que realiza la consultora Gartner, la herramienta UiPath obtiene una calificación

de 4.5/5 en nivel de estrellas de un estimado de 1865 calificaciones, lo que indica un promedio muy bueno dentro del mercado. En cuanto a la experiencia de usuario UiPath no se queda atrás, puesto que obtiene 4.4 estrellas de calificación en el tema de evaluación y contratación, 4.5 en la integración y despliegue con otros tipos de aplicaciones y 4.5 en el tema de servicio de ayuda y soporte, motivo por el cual, hace que el 92% de usuarios recomienden UiPath como herramienta para la automatización robótica de sus procesos.

Figura 17

Descripción general de las calificaciones de la plataforma UiPath

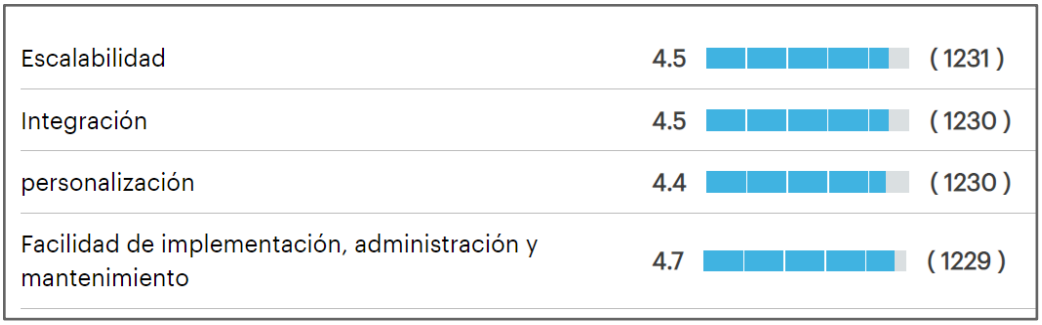


Nota. Resumen de las principales calificaciones obtenidas de la herramienta UiPath realizado a 1865 clientes/empresas (cierre al 27/06/2022). Reseñas Gartner, Gartner, 2022, Recuperado de (https://gtnr.it/3OGyXqa).

Por otro lado, si se toma en cuenta características como la escalabilidad, integración, personalización y facilidad de implementación, UiPath obtiene un promedio de calificación de 4.52, siendo el punto más destacable su facilidad de implementación, administración y mantenimiento, motivo por el cual es muy requerido por empresas del sector de servicios, finanzas, fabricación, cuidado de salud, educación entre otros.

Figura 18

Características y calificaciones del producto UiPath



Nota. Resumen de las principales calificaciones obtenidas de las características de UiPath como producto (cierre al 27/06/2022). Reseñas Gartner, Gartner, 2022, Recuperado de (https://gtnr.it/3OGyXqa).

Selección del software RPA gratuito UI.Vision. Luego de examinar el software RPA con licenciamiento, es importante analizar un software RPA gratuito que brinde la seguridad de automatizar procesos de manera rápida y sencilla y que a la vez, sirva como una alternativa viable y sin costo frente a las limitaciones que posee UiPath Community Edition. En este contexto, se seleccionó el software RPA UI.Vision (anteriormente Kantu) como herramienta de automatización robótica de procesos por las ventajas que ofrece frente a otras herramientas open source.

El software RPA UI.Vision es una de las herramientas más óptimas, utilizadas para la automatización visual de tareas, la automatización de pruebas de interfaz para usuarios, la aplicación de web scraping en páginas web y la implementación de screen scraping en aplicaciones de sistemas operativos como Windows, Mac y Linux. Este software posee un núcleo de código abierto que implementa una extensión en el navegador de manera gratuita y que se puede ampliar con las aplicaciones nativas del sistema operativo, garantizando la seguridad a nivel empresarial ya que los datos no salen de la máquina donde se integra el bot (UI.Vision, 2022).

Figura 19

Logo del software RPA gratuito UI.Vision

Nota. El software RPA UiVision es una herramienta multiplataforma y de código abierto. Tomado de: Ui.Vision RPA, Ui.Vision, 2022, Recuperado de (https://ui.vision/).

La principal razón por la que seleccionó la opción de Ui.Vision en comparativa con otros RPA gratuitos, es debido a las muchas ventajas que ofrece en cuanto a la automatización de procesos de manera visual. También se mencionan otras características como:

- Permite la automatización de procesos basándose en la implementación de macros que capturan la pantalla.
- Al poseer módulos basados en inteligencia artificial es ideal para el funcionamiento en los sitios web más complejos.
- La implementación de macros permite la grabación y reproducción de procesos de manera visual.
- Permite realizar pruebas basadas en la utilización de datos con importación de archivos CSV.
- La visión artificial de Ui.Vision permite escribir pruebas visuales de manera automatizada (UI.Vision, 2022).

Es importante mencionar que el software Ui.Vision ofrece una automatización completa de un sistema, ya sea de manera web, de escritorio o híbrida totalmente compatible con la herramienta Selenium IDE. Para entender de mejor manera su operación de automatización de procesos, en la Tabla 10 se mencionan sus principales características:

Tabla 10

Formas de automatización de procesos que posee UI.Vision

Formas de automatizar	Características	Implementación
Automatización web visual impulsada por IA.	- Permite la implementación de pruebas visuales automatizadas. - Posee la primera y única extensión para Chrome y Firefox que tiene visión artificial. - Emula diferentes resoluciones de pantallas, cambiando el tamaño del navegador. - Permite la lectura y reconocimiento de imágenes dentro de otras imágenes o videos.	Navegadores como: - Chrome - Firefox
Automatización de Escritorio Visual impulsada por IA	- Usa reconocimiento de imágenes y texto para automatizar el escritorio. - Posee comandos que arrastran, mueven y hacen clic en ventanas del escritorio. - Implementa módulos (XModules) que aumentan las capacidades para la automatización del escritorio.	Escritorio visual: - Windows - Mac - Linux
Automatización web clásica, totalmente compatible con Selenium IDE	- La versión gratuita implementa comandos estándar de Selenium IDE. - Es un software ideal para web scraping, automatización de carga de archivos y llenado de información automática en formularios. - Al usar Ui.Vision también se aprende Selenium IDE e incluso se puede importar casos de prueba de Selenium.	Escritorio visual y navegadores webs compatibles con Selenium IDE.

Nota. En la tabla se indica las tres principales formas que implementa UI.Visión para automatizar procesos.

Una de las principales ventajas de la implementación de Ui.Vision es su curva de aprendizaje ya que no requiere de muchos conocimientos en programación para establecer el

desarrollo de las macros de automatización. De hecho, cuando se invierte tiempo en el aprendizaje de Ui.Vision, también se aprende a desarrollar dentro del entorno de Selenium IDE. Por ello, se puede mencionar que con Ui.Vision se obtuvo una herramienta de apoyo para la creación de scripts de controlador web (UI.Vision, 2022).

En este contexto, en la Tabla 11 se destacan las principales características de la interfaz de usuario de Ui.Vision RPA Selenium y el IDE original de Selenium, para entender cómo funciona Ui.Vision RPA y las ventajas de utilización.

Tabla 11

Diferencias entre la interfaz de usuario Ui.Vision RPA y el IDE original de Selenium

Característica	**IDE de Selenium**	**Ui.Vision RPA Selenium IDE**
Implementa todos los comandos importantes de Selenium IDE	Sí	Sí
Código abierto	sí (licencia Apache 2.0)	sí (Licencia GNU AGPL 3.0)
Línea de comandos para programar ejecuciones, ejecutar en una cuadrícula.	selenium-ide-runner	interfaz de línea de comandos
Tomar captura de pantalla	Sí	Sí
Tomar captura de pantalla de página completa	No	Sí
Automatizar descargas de archivos	No	Sí
Exportación de scripts a Java	No	No
Pruebas visuales de la interfaz de usuario	No	Sí
Pruebas de elementos de lienzo	No	Si

Nota. Resumen de las principales características de la interfaz de usuario de Ui.Vision RPA Selenium y el IDE original de Selenium. Tomado de: Ui.Vision, Manual de Ui.Vision, 2022, Recuperado de (https://ui.vision/rpa/docs/selenium-ide#import).

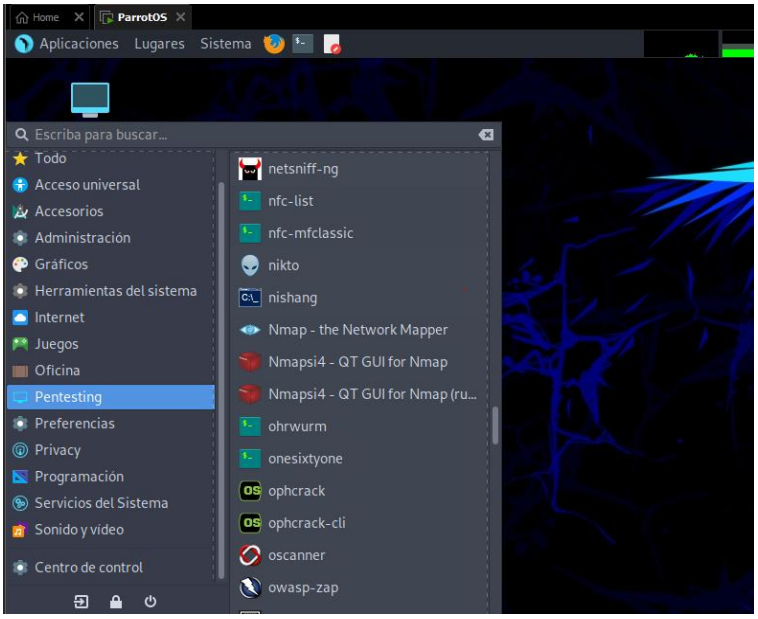
Análisis de vulnerabilidades web con herramientas convencionales

Nuestra propuesta tiene como objetivo analizar vulnerabilidades existentes en sitios web por medio del diseño e implementación de bots, sin embargo, es importante mencionar cómo se ejecuta dicho análisis utilizando herramientas convencionales (internas o externas al sistema operativo). Este proceso permitirá verificar la operación realizada por una búsqueda manual de vulnerabilidades web, para luego ponerla en contraste con el proceso automatizado.

Dentro del entorno de Linux nos encontramos con varías herramientas open source, que en su mayoría utilizan los profesionales de seguridad informática para realizar pruebas de pentesting en aplicaciones y sistemas tanto de escritorio como web. En base a ello, la distribución de Linux Parrot OS en su versión 5.0 ofrece varías herramientas convencionales para el análisis de vulnerabilidades web.

Figura 20

Herramientas de pentesting que ofrece Parrot OS



Nota. Muchas de las herramientas que se pueden observar vienen instaladas dentro del sistema operativo de Parrot OS.

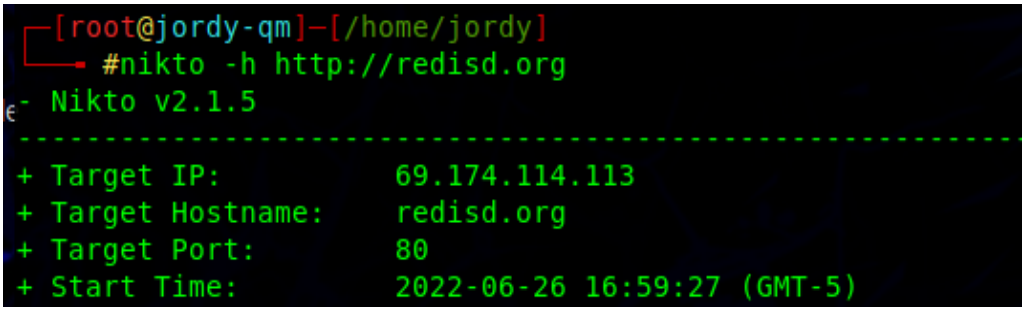
Análisis de vulnerabilidades web con Nikto

Los servidores web componen una parte esencial de cualquier organización, ya que contienen ficheros y sistemas de datos necesarios para el funcionamiento de los sistemas web, por este motivo son el principal foco de ataques informáticos. Una herramienta convencional que se usa para detectar y prevenir estas vulnerabilidades se llama Nikto. Este es un escáner de vulnerabilidades ejecutado mediante líneas de comandos que busca archivos peligrosos, software desactualizado y captura de cookies en los servidores web.

Para conocer sus funcionalidades, se ejecutó la herramienta en el entorno de Parrot OS, en donde, como primera recomendación es necesario conocer los comandos básicos para el escaneo del host y/o páginas web. En ese contexto, se realizó un análisis del host "http://redisd.org", ingresando la instrucción *nikto -h (URL de la página que se analizará)*, tal como se observa en la Figura 21.

Figura 21

Análisis del host "http://redisd.org" con el uso de la herramienta Nikto



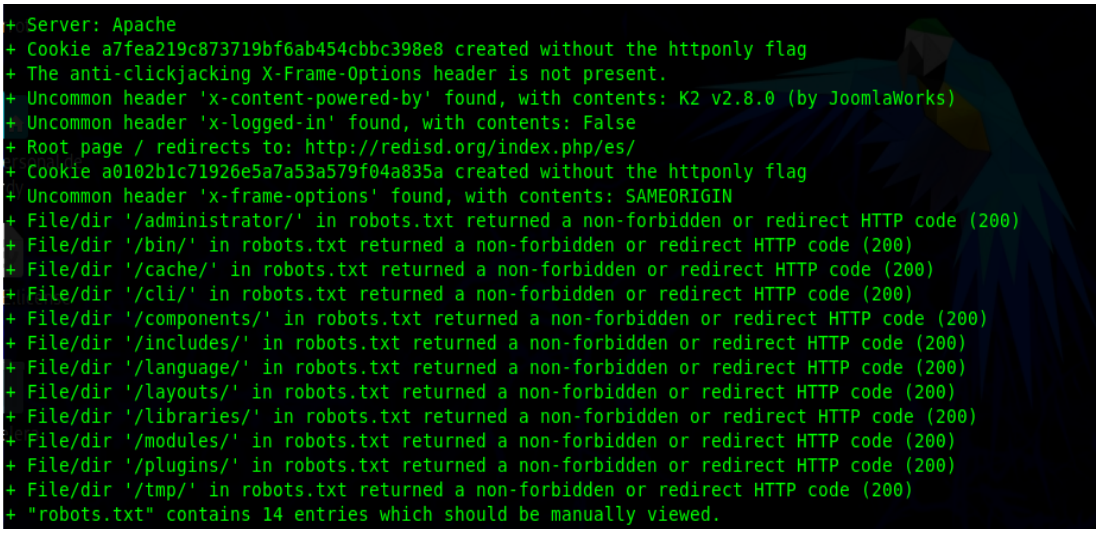
Nota. Se recomienda ejecutar los comandos de Nikto en modo administrador, para proporcionarle los permisos necesarios de ejecución.

Iniciado el escaneo del host o URL, Nikto muestra información básica como: dirección IP, puerto de enlace, tipo de servidor y la versión del CMS (Content Management System) con el que fue desarrollada la página web. Si se continúa con el escaneo se muestra información más detallada de los principales directorios que posee el sitio web.

En el caso de la página “http://redisd.org” se obtuvo información sobre los principales directorios del archivo robots.txt, enlace de cookies, las cabeceras, entre otros datos, cuyos resultados son presentados en la Figura 22.

Figura 22

Análisis del archivo robots.txt de “http://redisd.org” con el uso de Nikto



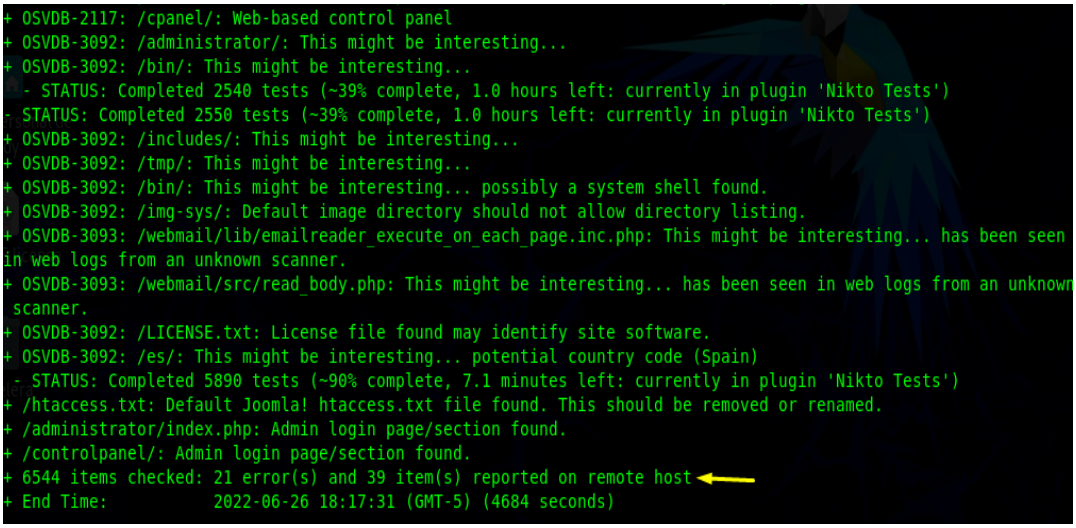
Nota. Es recomendable ejecutar los comandos de Nikto en modo administrador, para proporcionarle los permisos necesarios de ejecución.

Esta información puede ser utilizada por los cibercriminales para ejecutar ataques especializados al servidor web detectado en el análisis de acuerdo a su tipo (Apache), a los archivos encontrados en la página web, a los directorios proporcionados o simplemente realizar ataques de tipo DDoS (Ataque de Denegación de Servicios). Nikto al finalizar el escaneo ofrece un conteo de los posibles errores de seguridad que posee la página web y los ítems que pueden ser revisados para establecer parches de seguridad.

En la Figura 23 se muestran los resultados obtenidos que resumen lo detallado. Hay que resaltar que el tiempo que demora la herramienta para establecer un análisis completo, depende de la cantidad de directorios y componentes que posea el sitio web.

Figura 23

Resultados del análisis realizado con Nikto



Nota. Es recomendable ejecutar los comandos de Nikto en modo administrador, para proporcionarle los permisos necesarios de ejecución.

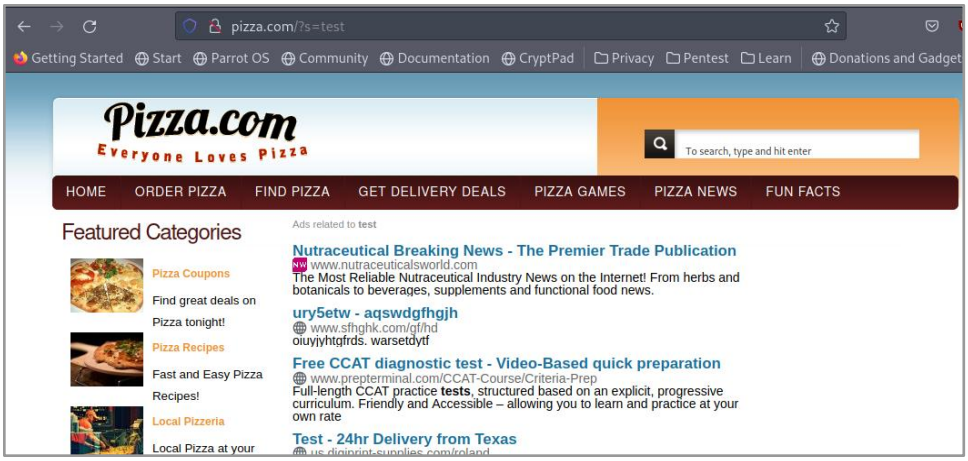
Análisis de vulnerabilidades XSS

De acuerdo a la literatura investigada se ha identificado que los ataques XSS se han incrementado en los últimos años, debido a su fácil implementación y los beneficios que puede obtener un ciberdelincuente. Para detectar y contrarrestar este tipo de ataques existen muchas herramientas que permiten ejecutar un análisis a páginas web potencialmente vulnerables, pero en su mayoría dicho proceso se realiza de manera manual.

En este contexto, para analizar esta vulnerabilidad, se instaló y probó la herramienta XSStrike, en un entorno virtualizado con el sistema operativo Parrot OS. Una vez instalada la herramienta se realizaron las pruebas de penetración XSS al dominio “pizza.com”, este posee una barra de búsqueda que comúnmente son elementos de una página web utilizados para aplicar ataques XSS de tipo reflejado.

Figura 24

Página web analizada con la herramienta XSStrike

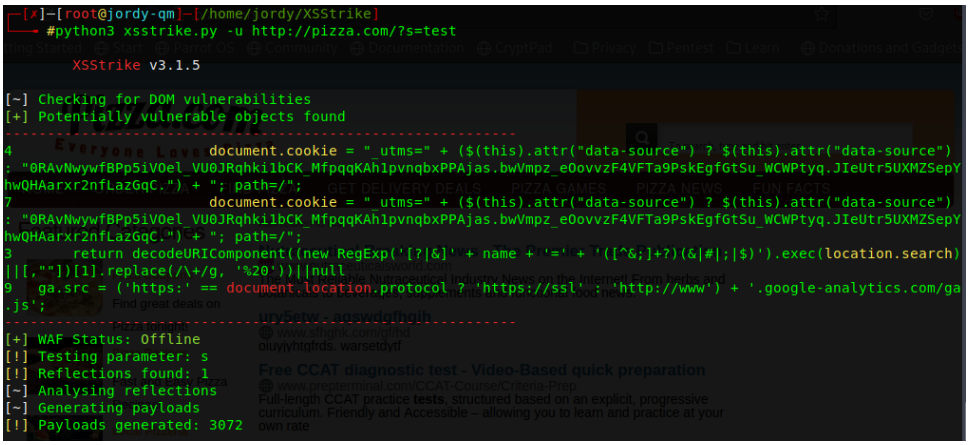


Nota. En la imagen se muestra la interfaz de inicio de la página web analizada.

Al igual que la mayoría de las herramientas utilizadas en Parrot OS, XSStrike basa su funcionamiento en el uso de la consola y la aplicación de comandos para ejecutar una tarea. Como se observa en la Figura 25, se aplicó una instrucción para realizar el análisis XSS a la página web “pizza.com”, dando como resultado una lista detallada de posibles vulnerabilidades.

Figura 25

Resultado del análisis realizado a pizza.com



Nota. El análisis de los Payloads permite determinar la carga que se activa al momento de aprovechar la vulnerabilidad existente.

Aparte de los resultados obtenidos con XSStrike, esta herramienta genera pruebas o payloads para comprobar la seguridad del sitio. Para la página analizada se seleccionaron los payloads con mayor margen de eficiencia y se ingresó en la barra de búsqueda del sitio web “pizza.com” un pequeño script XSS, obteniendo como resultado una alerta con el número 1. En base a estos resultados, que se observan en la Figura 26, se puede determinar que la página web “pizza.com” es vulnerable a ataques de tipo XSS.

Figura 26

Ejecución de script por parte de pizza.com

Nota. Al obtener una respuesta de la alerta insertada con el script en la página web, se puede determinar que es vulnerable a un ataque de tipo cross-site scripting.

Bot para el análisis de vulnerabilidades en los sistemas CAPTCHA

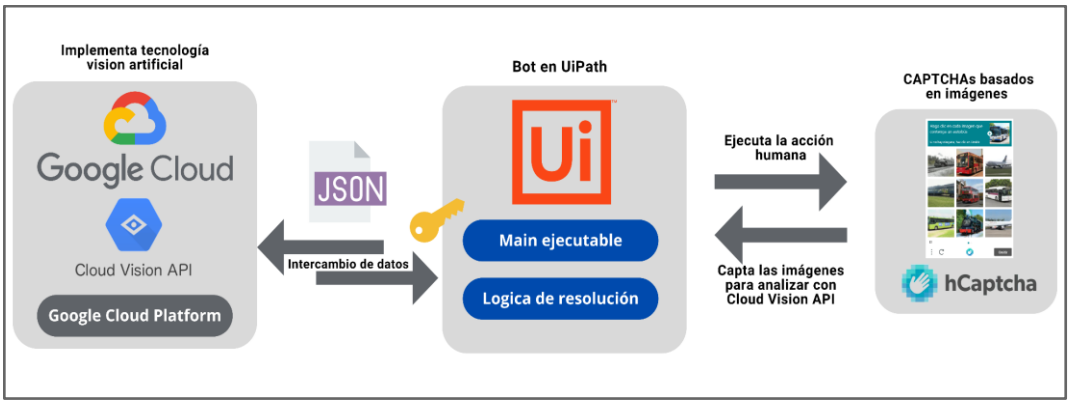
Tomando en consideración los procesos de análisis de vulnerabilidades ejecutados en el apartado anterior, se ha determinado que muchos se realizan de forma manual, el pentester debe tener en cuenta la sintaxis de los comandos que implementa cada herramienta. Si dicho análisis se lleva a un nivel más empresarial y considerando que muchas empresas tienen la mayoría de su información en sistemas web, conlleva a que estos procesos manuales tarden semanas o incluso meses para obtener un informe certero. Por ello, la importancia de automatizar ciertos procesos que permitan mejorar el tiempo de verificación y de rectificación de las vulnerabilidades encontradas.

Sabemos que muchas páginas web implementan sistemas CAPTCHAS para contrarrestar ciertos ataques realizados por bots maliciosos y de esta manera proteger la seguridad de los servicios web. Pero en la actualidad, con el aumento de las tecnologías basadas en visión artificial, muchos de estos procesos de verificación de humanos resultan obsoletos. El objetivo de este estudio es demostrar que estos sistemas pueden ser vulnerados precisamente con la implementación de bots automatizados.

En ese contexto, en los siguientes párrafos se detalla de forma clara el diseño e implementación del bot que demuestra la vulnerabilidad que poseen los sistemas CAPTCHA basados en imágenes, utilizando para ello características de visión artificial. Para el diseño de nuestro bot se ha dividido el proceso en dos partes: la primera establece las configuraciones necesarias para la obtención de la clave de una API de Google (Cloud Vision API), misma que permite aprovechar todas las ventajas que ofrece la plataforma en la nube de Google. La segunda parte detalla el proceso de desarrollo del bot en el software UiPath Community Edition. En la Figura 27 se muestra un esquema gráfico del funcionamiento y fases de desarrollo del bot propuesto.

Figura 27

Fases del desarrollo del bot propuesto para vulnerar un sistema CAPTCHA



Nota. Esquema general del proceso de implementación del bot que resuelve sistemas CAPTCHA. Los detalles de cada fase son analizados en los siguientes apartados.

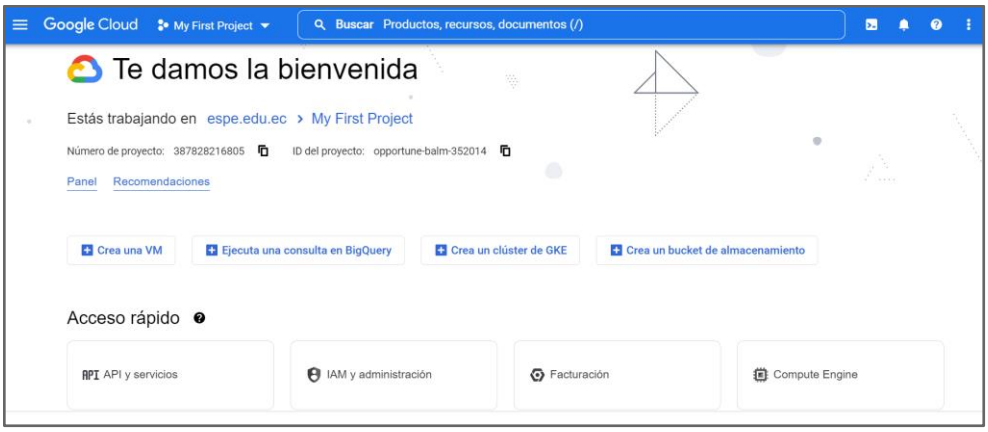
Integración con la Plataforma Google Cloud

Como se explicó en el apartado anterior, nuestro bot utilizará técnicas de visión artificial para realizar el análisis de las imágenes que aparecen en el CAPTCHA, se implementó la tecnología que nos ofrece la Plataforma de Google Cloud. Google Cloud Platform es un servicio basado en la computación en la nube que ofrece la infraestructura necesaria para desarrollar aplicaciones, implementar soluciones informáticas o escalar en el desarrollo empresarial de acuerdo a las necesidades del negocio. A pesar de todas las ventajas que ofrece la implementación del bot con la integración de la plataforma en la nube de Google, muchos de sus servicios son de pago, sin embargo, ofrece la facilidad de usarlos de manera gratuita por tiempo limitado.

En la Figura 28, se presenta la interfaz de inicio que se muestra al usuario tras la creación de una prueba gratuita en la plataforma de Google Cloud Platform. Google Cloud ofrece el crédito de $300 dólares americanos para la utilización de sus servicios en un plazo máximo de 3 meses, si el usuario sobrepasa dicho crédito, Google empezará a facturar los servicios consumidos.

Figura 28

Página principal de la interfaz de la Plataforma de Google Cloud

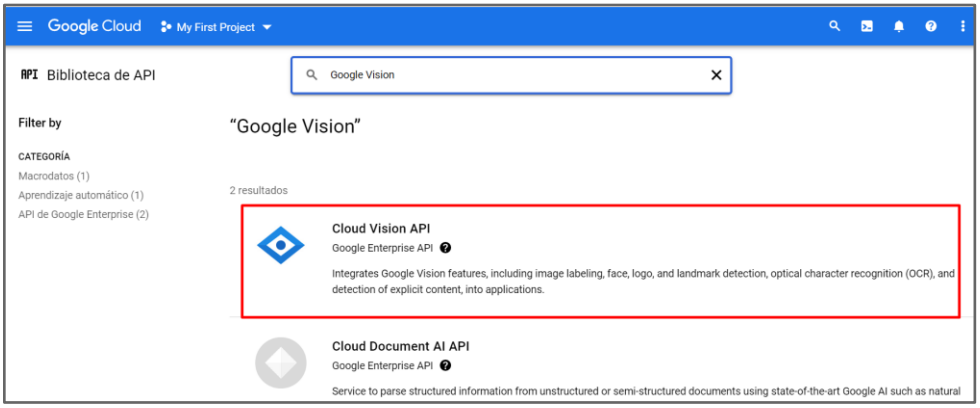


Nota. Para ingresar a los servicios proporcionados por la plataforma de Google se debe registrar una cuenta y acceder.

Para la creación de la cuenta dentro de esta plataforma es importante proporcionar los datos de una tarjeta de crédito o débito de una entidad bancaria activa con el fin de comprobar que el usuario registrado es humano. Al obtener la cuenta se procedió a preparar el proyecto de visión artificial dentro de la plataforma, para esto, se dio clic en la opción de APIs y Servicios, donde se despliega el buscador de servicios dentro de la Biblioteca de APIs y se digita la opción “Google Vision”, tal como se observa en la Figura 29.

Figura 29

Búsqueda de la API de visión artificial en Google Cloud

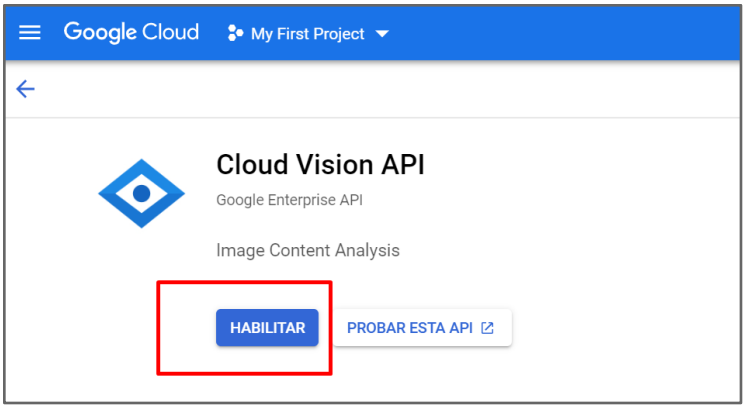


Nota. En la imagen se muestra la búsqueda realizada para obtener como resultado la API de Google Cloud Vision.

Al visualizar la opción del servicio requerido, se habilitó la API de Google Cloud Vision dando clic en el botón *“Habilitar”*. Esta API permite extraer información muy importante de las imágenes proporcionadas por medio del uso de modelos preentrenados con inteligencia artificial. Esta API implementa aprendizaje automático para analizar las imágenes con una precisión muy acertada a la realidad. Además, entrena los modelos de machine learning a partir del uso de etiquetas con AutoML Vision. Una de sus mayores ventajas es la detección de rostros, objetos, textos manuscritos y metadatos de las imágenes proporcionándole al usuario la mayor cantidad de información posible (Google Cloud, 2022).

Figura 30

Habilitación de la API de Google Cloud Vision

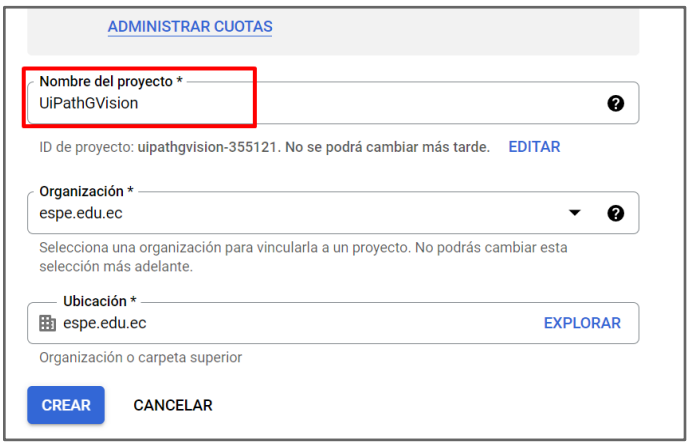


Nota. En la imagen se muestra la habilitación de la API de Cloud Vision para utilizar todos los beneficios que ofrece.

Para aprovechar las ventajas que ofrece la implementación de la API de Google Cloud Vision, se creó un nuevo proyecto de visión artificial dentro de la plataforma, para ello en la parte superior de la plataforma se dio clic en *"Mi primer proyecto"* específicamente en la opción de *"Crear un nuevo Proyecto"*. Luego, se asignó el nombre que se muestra en la Figura 31 y se dio clic en Crear.

Figura 31

Creación del proyecto que contiene la API de Google Cloud Vision

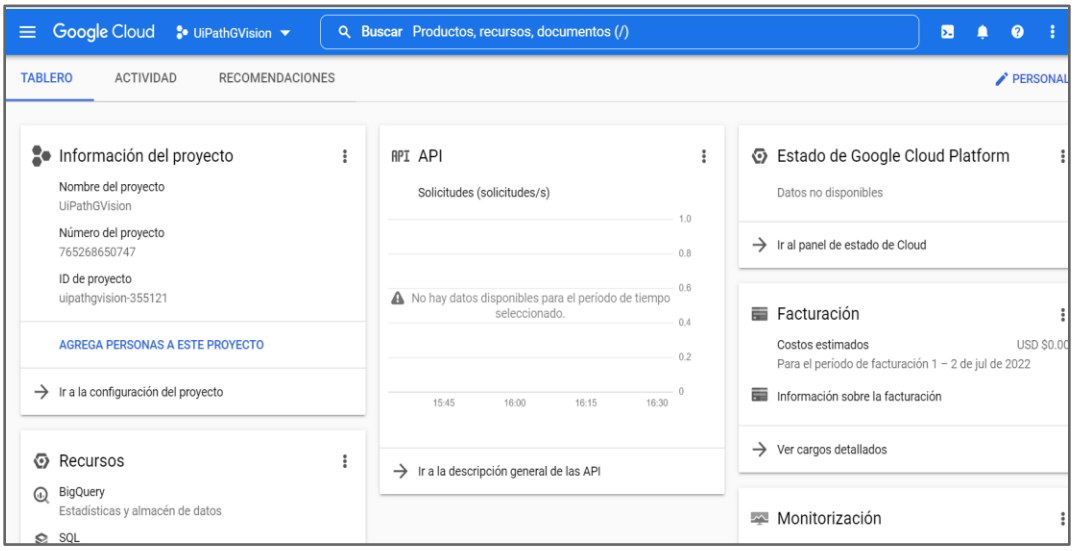


Nota. Si no se agrega la organización y ubicación no se habilita el botón de "*Crear*".

Luego de establecer las configuraciones básicas del proyecto se dio clic en el nombre "*UiPathGVision*", se desplegó el tablero de información que muestra las actividades generadas dentro de la API, así como los principales recursos utilizados, facturación y monitorización de las tareas realizadas. Estos elementos son primordiales para ver la actividad, registros generados por la API y verificar la facturación por su uso.

Figura 32

Creación del proyecto que contiene la API de Google Cloud Vision

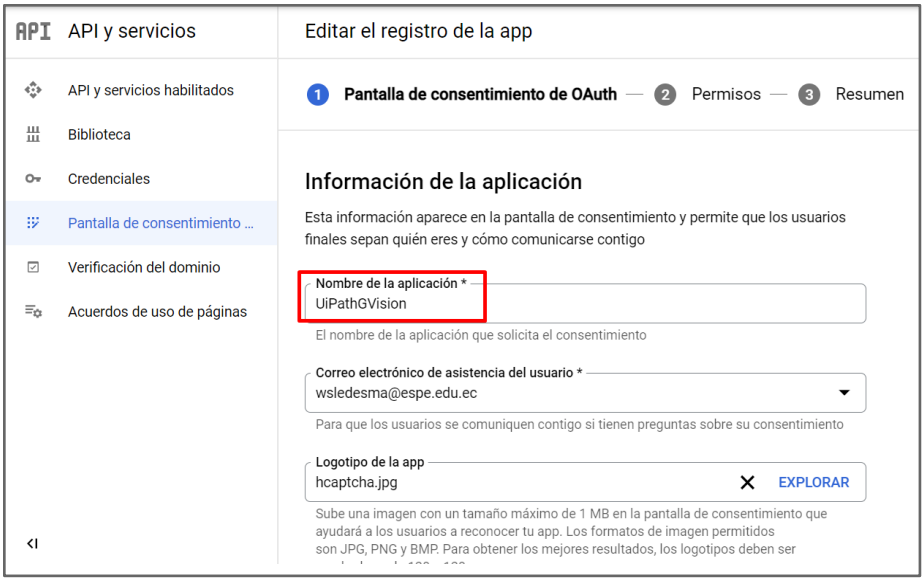


Nota. En la imagen se muestra el tablero de información del proyecto, el cual aparece al finalizar su creación.

Después de configurar el proyecto es importante crear la APP que proporciona las credenciales que serán utilizadas en el desarrollo del bot. Para esto, se dio clic en la opción de *"Configurar pantalla de consentimiento"* para configurar y registrar la app, incluyendo a los usuarios objetivos. Como tipo de usuario se seleccionó la opción "*Interno*" y se dio clic en "*Crear*". En la Figura 33, se visualiza el despliegue de las opciones para editar el registro de la APP, cuya información aparece en la pantalla de consentimiento y permite que los usuarios finales sepan quién somos y cómo se pueden comunicar con nosotros en caso de tener preguntas.

Figura 33

Configuración de la pantalla de consentimiento de la APP



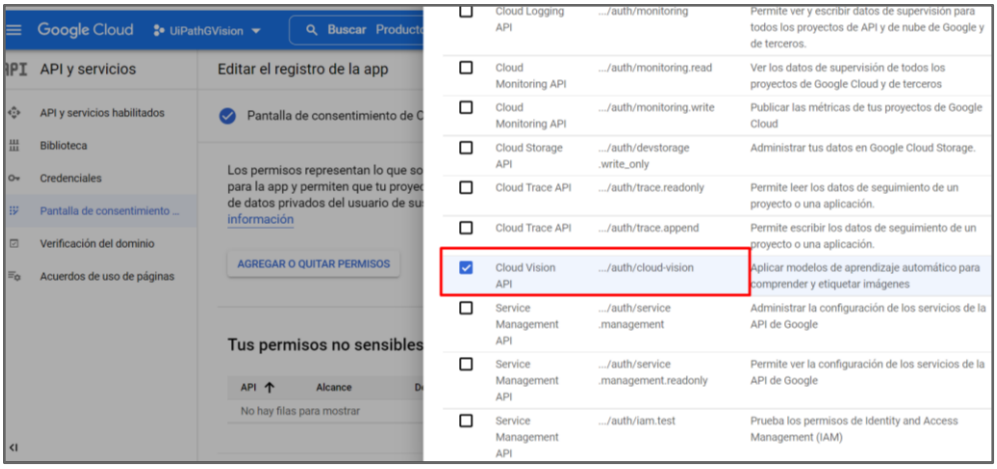
Nota. Al registrar la APP dentro de la pantalla de consentimiento, el siguiente paso es la obtención de los permisos necesarios.

Luego de configurar la pantalla de consentimiento, es importante establecer los permisos necesarios para el acceso a la app, tal como se observa en la Figura 34. Estos permisos hacen referencia a las solicitudes que los usuarios envían hacia la App, permitiendo que el proyecto obtenga acceso a sus datos privados específicos de sus cuentas de Google.

Al finalizar esta configuración se observará un resumen de la API correspondiente a la pantalla de consentimiento de OAuth. Se puede configurar más de un servicio dentro de una APP, pero para nuestra propuesta solo se habilitó la opción de Cloud Vision API, el cual contiene las características de visión artificial.

Figura 34

Habilitación de la opción para la clave de acceso a la app

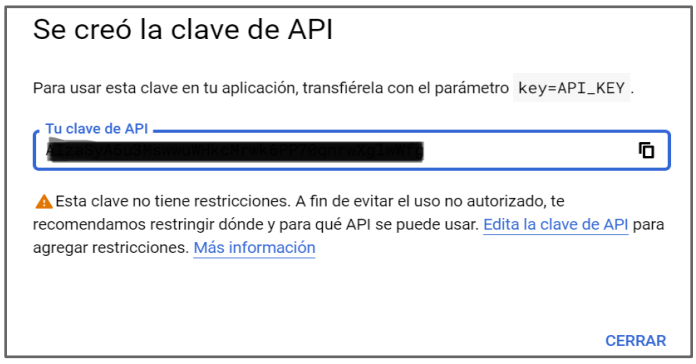


Nota. Es importante habilitar la opción de Cloud Vision para obtener los permisos y la clave que será utilizada para acceso a la API.

Finalmente, se crearon las credenciales necesarias para establecer la conexión del bot con la API. Para esto, nos dirigimos al apartado de "*Credenciales*" y damos clic en *"Crear Credencial"*. En este apartado es importante seleccionar la opción "*Clave de API*" ya que esto identifica el proyecto creado con una clave y así se verifica el acceso del usuario.

Figura 35

Habilitación de la opción de clave de acceso a la app de la API

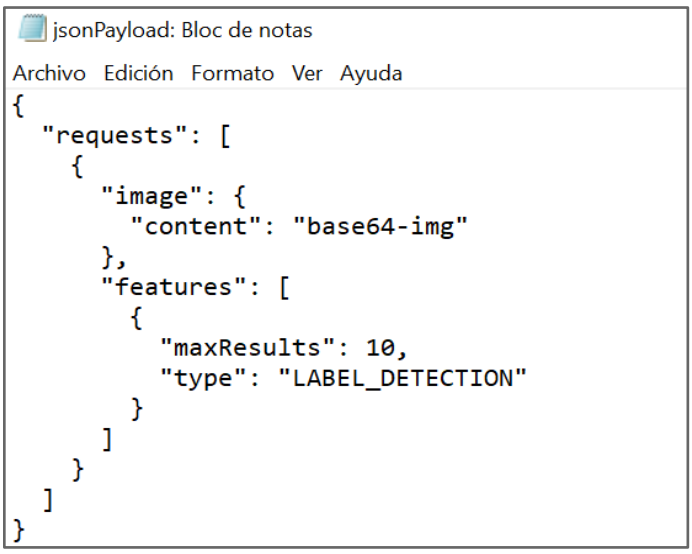


Nota. Se coloreó con negro para proteger la clave de acceso a la aplicación de Google Cloud Vision.

Esta clave de acceso debe ser colocada dentro de los argumentos de entrada en el archivo Main del bot en desarrollo. Para establecer el intercambio de datos entre el bot y la API de Google Cloud Vision, es importante desarrollar un script en formato de solicitud, que consta de un objeto JSON y que incluye una sola lista de *requests* y uno o más objetos de tipo *AnnotateImageRequest*, tal como se observa en la Figura 36.

Figura 36

Creación del script JSON que permite el intercambio de datos



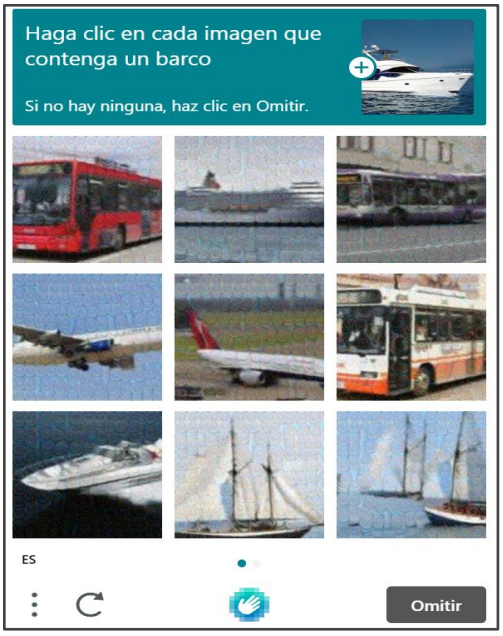
Nota. Las peticiones a la API de Google Cloud Vision se realizan por medio del argumento "requests" que obtiene los objetos de "imagen" y "características".

Desarrollo del Bot en el software de UiPath

Para el desarrollo del bot, se ha seleccionado la página en la que se va a implementar el análisis de CAPTCHA, puesto que cada bloque de actividad dentro de UiPath utiliza los selectores del sitio web para realizar el web scraping de la sección solicitada. Se utilizó como ejemplo la página "*https://www.map24.com/contact/*", que implementa un hCAPTCHA que es un CAPTCHA basado en imágenes, y es considerado número uno por ofrecer un nivel de privacidad tanto para la web, los dispositivos móviles y más. En la Figura 37 se muestra el CAPTCHA que se resolverá con la implementación del bot.

Figura 37

CAPTCHA que pretende resolver el bot



Nota. El hCAPTCHA asegura ser una versión más privada y mejorada de los CAPTCHAs basados en imágenes, además, proporciona una detección más fiable de bots.

En este punto es importante resaltar que el bot está conformado por algunas partes principales: el archivo *Main,* en donde se ejecuta el programa y la parte del *CAPTCHASover.xaml*, en donde se desarrolla la lógica de programación para vulnerar el sistema CAPTCHA. Cada parte cumple una función importante que en conjunto conforma nuestro sistema robótico de procesos.

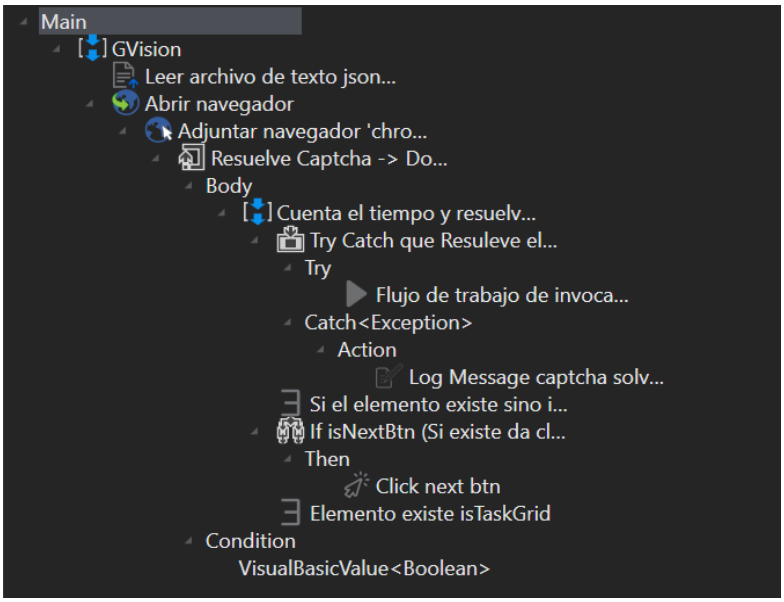
Desarrollo del Main.xaml. Para la creación de nuevos proyectos dentro de UiPath existen diferentes formas vías para iniciar el desarrollo. En primer lugar, se encuentra la opción de crear un proyecto de "*Procesos*", para desarrollar uno nuevo y establecer la automatización de tareas desde cero. Existe la opción de "*Biblioteca*", para la creación de un nuevo proyecto desde componentes reutilizables y publicados a manera de bibliotecas de scripts. Otra opción es la "*Prueba de Automatización*", para crear proyectos de prueba desde cero. Por último se

tiene la opción de “*Plantilla*”, para el desarrollo de proyectos a partir de otros previamente creados y almacenados como plantillas reutilizables.

Para la creación del bot se utilizó la opción de un nuevo proyecto de “*Procesos*”, estableciendo como elemento primario el desarrollo del archivo Main. Este archivo juega un papel muy importante dentro de cualquier programa desarrollado en UiPath, porque establece la ejecución principal del programa. Para explicar cómo se encuentra estructurado el bot,, se muestra el esquema de desarrollo del archivo Main.

Figura 38

Esquema estructurado del Main.xaml



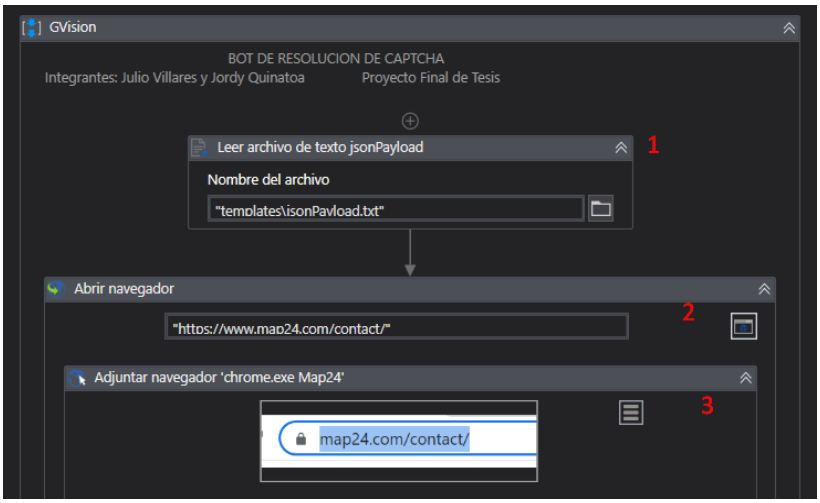
Nota. Cada uno de los bloques de actividades implementados en el Main se explican en los siguientes apartados.

Es importante conocer que en UiPath, cada bloque de desarrollo corresponde a una actividad que contiene las instrucciones que realizará el bot y las que serán ejecutadas en secuencia. Para el caso de nuestro bot, el bloque principal ejecuta un conjunto de actividades secundarias de acuerdo a una orden definida y única. Como el bot establece una comunicación con la API de Google Cloud Vision, se realizará un llamado al archivo *JsonPayload.txt* por

medio de la actividad que permita leer un archivo de texto **(1)**. Como el bot analiza el sistema CAPTCHA de un sitio web específico, se ha implementado una actividad para abrir el navegador en una URL determinada **(2)**, una vez abierto el navegador se utiliza un contenedor que se adjunta al navegador y ejecuta múltiples tareas dentro de este **(3)**.

Figura 39

Actividades del bot que permiten leer el archivo JSON y abrir el navegador



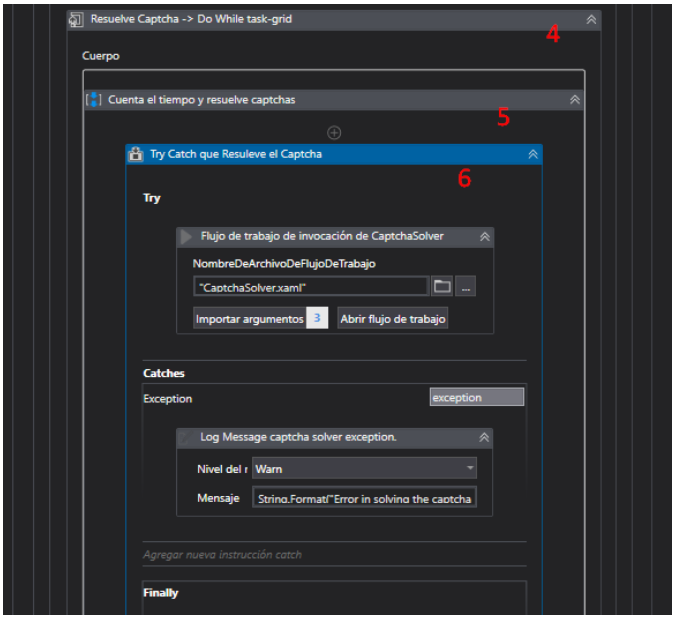
Nota. El URL utilizado para establecer el análisis del sistema CAPTCHA corresponde a una revista denominada “Map24”.

Al ingresar a esta página web se define la lógica de análisis del sistema CAPTCHA, para ello se establece un bloque que permite ejecutar primero las actividades contenidas y repetirlas para cuando se cumplan ciertas condiciones **(4)**.

Para esta instancia es importante establecer una secuencia de actividades secundarias que se ejecuten en un orden único **(5)**, para luego ejecutar las actividades en un flujo de trabajo *TryCatch* el cual contiene las actividades que se ejecutan en un bloque de manejo de excepciones **(6)**.

Figura 40

Llamado a las actividades establecidas en el CAPTCHASolver



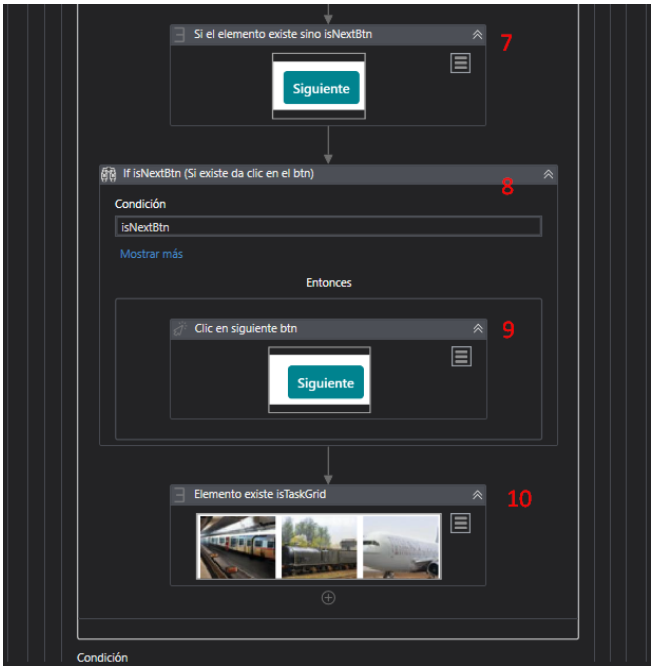
Nota. Al realizar el análisis del sistema CAPTCHA existen actividades que pueden resultar repetitivas, motivo por el cual se implementó un bloque de tipo Do-While.

Dentro del Try-Catch se estableció un bloque de flujo de trabajo el cual invoca sincrónicamente al flujo de actividades que envía una lista de argumentos de entrada. Este flujo de trabajo permitió dividir las actividades desarrolladas en la lógica de análisis para la resolución del CAPTCHA, con la lógica implementada en archivo Main. En otras palabras, en este punto se hace el llamado a las actividades establecidas en el *CAPTCHASolver* y se ejecutan en el Try, caso contrario se da paso al Catch donde se establece el mensaje de error en la ejecución.

Luego de que el bot pueda seleccionar cada una de las imágenes del CAPTCHA, debe dar clic en el botón "*Siguiente*" para solucionar el siguiente panel de figuras, por esto es importante verificar si existe el elemento de botón **(7)**, luego se verifica la existencia del botón **(8)**, se pueda dar clic en el elemento especificado **(9)**, para continuar con la resolución del segundo panel de imágenes que proporciona el CAPTCHA **(10)**.

Figura 41

Actividades de verificación del botón “Siguiente”



Nota. Es importante verificar si el botón existe puesto que puede generar inconvenientes si solo se da clic directamente en elemento especificado.

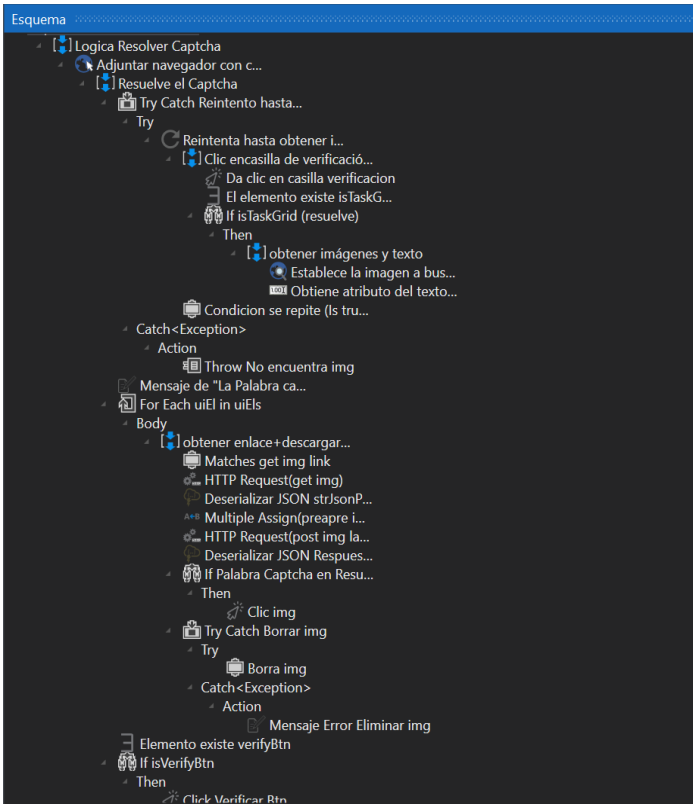
Desarrollo del CAPTCHASolver.xaml. En el archivo *CAPTCHASolver* se implementa toda la lógica requerida para realizar las actividades que un humano desarrolla sin problemas dentro de una página web. Actividades tan simples como: dar clic en un botón, realizar scroll al sitio, seleccionar imágenes o simplemente aceptar un check list, que para una máquina puede implicar miles de líneas de programación, pero para el humano es sencillo. De acuerdo a esto, el archivo *CAPTCHASolver* se considera como la parte medular del proyecto, ya que aquí se encuentra lo esencial para establecer el análisis del CAPTCHA basado en imágenes y obtener los resultados.

En la Figura 42 se observa el esquema principal de las tareas que realiza el bot, estas se organizan en un bloque general de secuencia y conforman un conjunto de actividades secundarias estructuradas en un árbol. Para analizar una sección específica de la página web,

se utilizó el "*Explorador de IU*" que proporciona UiPath para determinar la posición de un elemento en la pantalla y extraer los selectores que conforman la estructura del CAPTCHA.

Figura 42

Esquema estructurado del archivo CAPTCHASolver.xaml



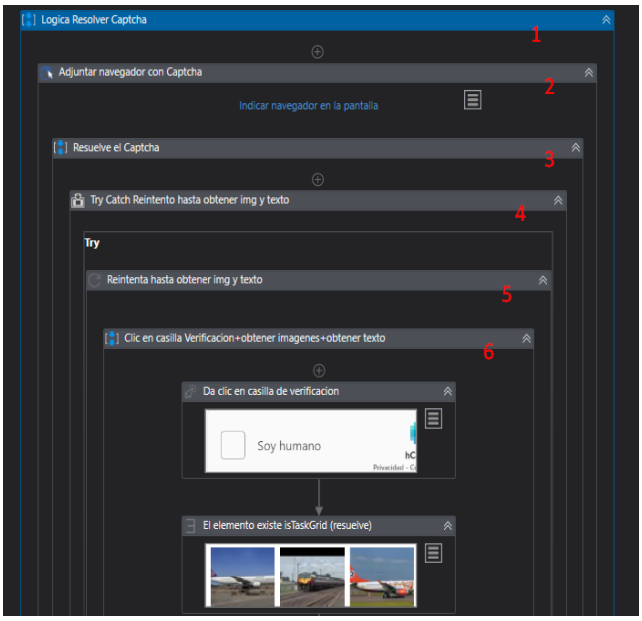
Nota. En la figura se muestra el esquema completo del archivo *CAPTCHASolver*, donde cada actividad secundaria cumple un papel importante en el funcionamiento del bot.

El esquema principal del bot está conformado por un bloque de actividad primario de secuencia, en donde las actividades secundarias se ejecutan en un orden definido. Para entender mejor, en la Figura 43 se visualiza el bloque principal denominado "Lógica de CAPTCHA" **(1)**, que contiene un bloque de actividad que adjunta la página del navegador y la mantiene abierta **(2)**. Dentro de la página web se realizan todas las actividades de análisis y es importante mantener un orden de ejecución, para esto se utilizó nuevamente un bloque de actividades en secuencia **(3)**.

Se implementó un bloque de actividad que contiene las tareas que se ejecutan en un manejo de excepciones de tipo Try-Catch **(4)**, si no cumplen con las condiciones establecidas vuelven a repetirse hasta que sean verdaderas **(5)**. Como ciertas actividades pueden ser repetitivas hasta que sus condiciones se cumplan, es importante cumplir con una ejecución ordenada y sistemática, por esto nuevamente se utilizó un bloque de secuencia **(6)**, se agregaron actividades como dar clic en el botón que abre el CAPTCHA y verifica si existen las imágenes que se van analizar.

Figura 43

Actividades del bot que permiten abrir el CAPTCHA y verificar las imágenes



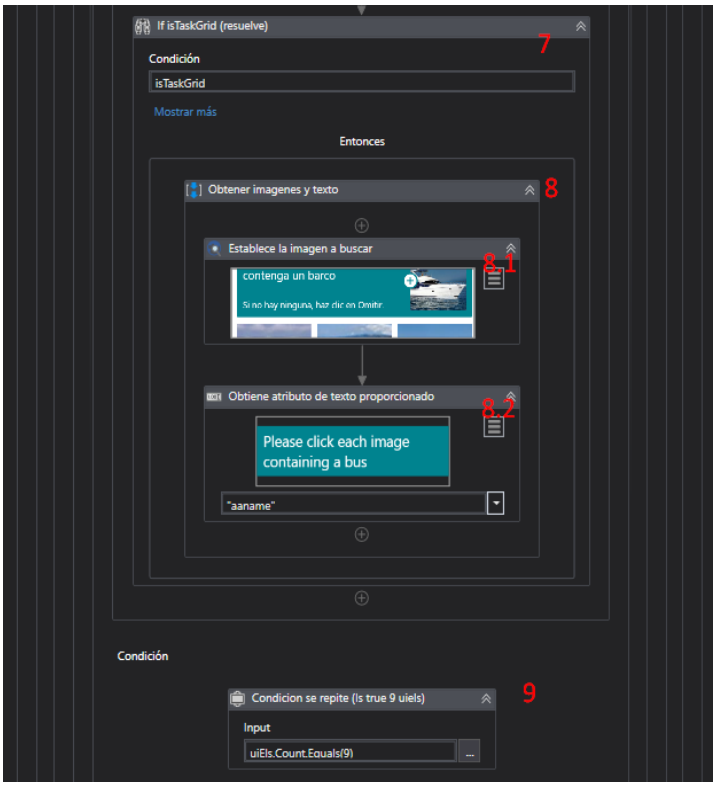
Nota. Para que el bot realice clic en el botón que permite abrir el CAPTCHA, es importante determinar como argumento de salida el selector que indique las etiquetas que contienen el elemento en la página web.

Luego de establecer la actividad de secuencia se implementó un bloque que modela una condición, misma que para cumplir con la tarea debe obtener las imágenes en base al texto que indica el CAPTCHA **(7)**. Se utilizó un bloque de secuencia **(8)**, que analiza la imagen que va a buscar según el criterio de filtro especificado **(8.1)** y obtiene el valor de un atributo

determinado por el Explorador IU y detallado en su selector destino **(8.2)**. El proceso de análisis de las imágenes del CAPTCHA se realiza en base a la matriz de figuras que se presentan al usuario, en nuestro caso al ser 9 figuras, la condición debe repetirse 9 veces para determinar si concuerda con la imagen modelo (**9)**.

Figura 44

Actividades del bot que analizan las imágenes en base al texto proporcionado



Nota. Para obtener la palabra que determina que se va a buscar en la matriz de 9 imágenes, es importante determinar en el selector de destino las etiquetas HTML que indican dicho texto y proporcionarle como atributo de entrada “aaname”.

También es importante configurar un mensaje que indique al usuario qué palabra se está analizando dentro de la matriz de imágenes, para esto se utiliza un bloque de tipo *LogMessage* que escribe el mensaje de diagnóstico especificado en el nivel determinado **(10)**. En este punto, se tiene el análisis de los componentes que conforman el CAPTCHA, pero hace

falta analizar cómo determinar las imágenes correctas en base a la palabra que se busca. Se utilizó un bloque de tipo For-Each que permite ejecutar una serie de actividades secundarias en cada elemento de una enumeración **(11)**.

Dentro del For-Each se realiza una serie de actividades en secuencia que busca una cadena de entrada para las apariciones de una expresión regular **(12)**, luego, se configura la actividad que permita enviar una solicitud y obtener la imagen que se va analizar con la API **(13)**, por medio de la serialización de información establecida en el archivo JSON previamente configurado **(14)**, para luego asignar múltiples argumentos dentro de una misma actividad **(15)**, con el fin de obtener una respuesta de la solicitud y la deserialización de la cadena JSON a un objecto *JObject* **(16)**.

Figura 45

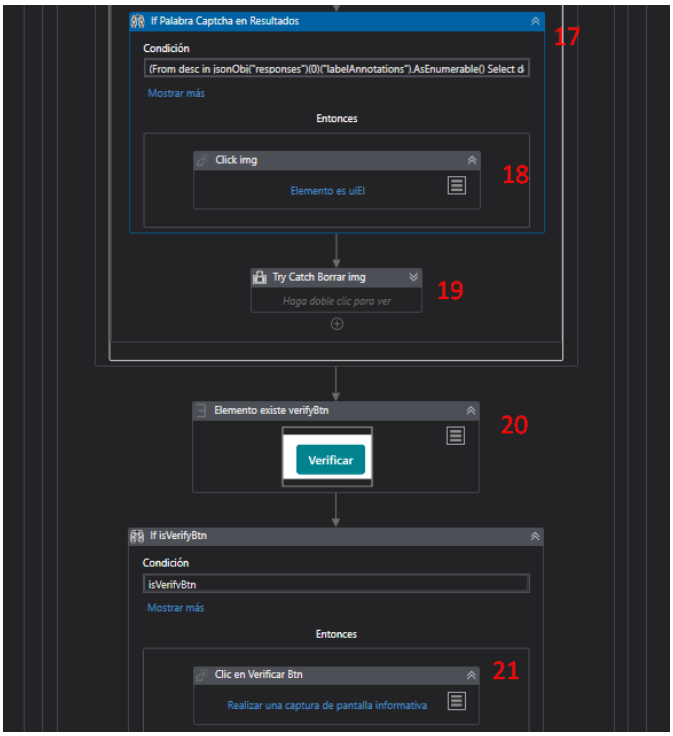
Actividades del bot que permiten obtener las imágenes para el análisis con la API

Nota. Se destaca que para realizar el análisis de similitud con lo que establece el título del CAPTCHA, el bot descarga la imagen en la carpeta raíz del proyecto y la analiza por medio del archivo JSON previamente configurado.

Hasta este punto se obtiene la imagen y se analiza con la API de visión artificial, pero aún falta establecer qué pasaría si la imagen encontrada concuerda con la palabra del CAPTCHA. En este caso se estableció una condición de tipo *If* **(17)**, que determina la correspondencia de la imagen con la palabra para dar clic en cada una de los elementos correctos **(18)**, luego borra la imagen analizada de la carpeta raíz del proyecto **(19)**. En este caso, si cumple con las condiciones establecidas, el bot procede a analizar la existencia del botón "*Verificar*" **(20)**. Al comprobar su existencia, automáticamente se establece la condición de verificación satisfactoria y se procede a dar clic en el botón para pasar la prueba **(21)**.

Figura 46

Actividades del bot que permiten dar clic en las imágenes correctas y pasar el test



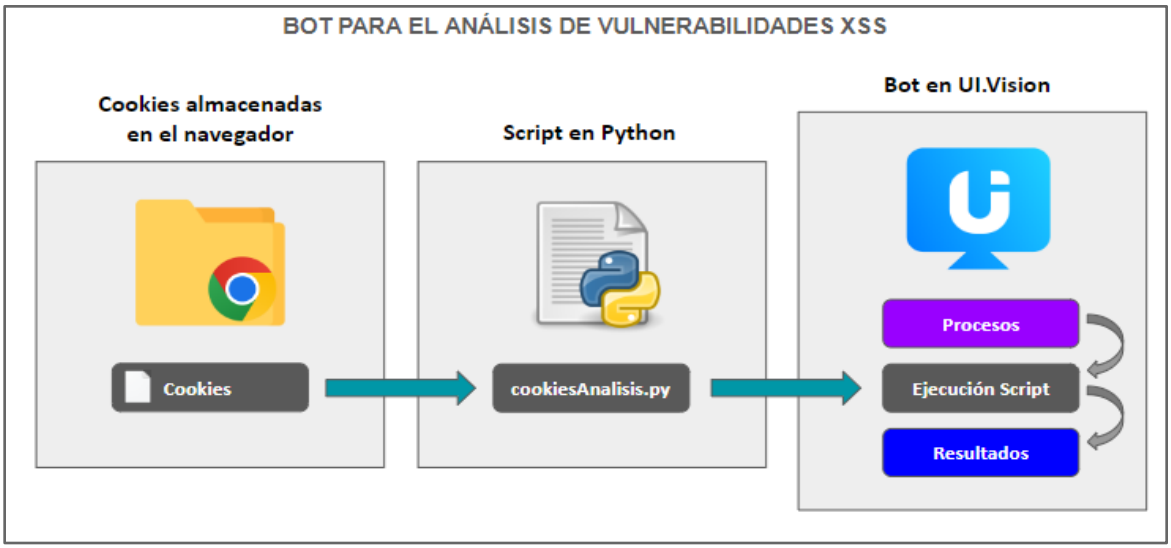
Nota. Una vez que se analiza las imágenes y cumple con las condiciones establecidas en el *if,* el bot dará clic en verificar y pasará automáticamente la prueba del CAPTCHA.

Bot para el análisis de vulnerabilidades de tipo Cross-Site Scripting (XSS)

El proceso aplicado para el desarrollo de este bot consta de tres partes fundamentales, que componen la lógica para analizar vulnerabilidades XSS en sistemas web, tomando como base, el estudio de las cookies que estos generan. En la Figura 47, se detalla el proceso implementado para la creación y ejecución del bot, utilizando el software RPA UI.Vision en conjunto con el lenguaje de programación Python.

Figura 47

Proceso de desarrollo del bot para el análisis de XSS en sistemas web



Nota. El análisis de XSS se basa en examinar las cookies generadas a partir de la visita de la página web en el navegador Chrome.

En base al proceso descrito en la Figura 47, se observa que para realizar el proceso de la obtención de los resultados finales se implementan una serie de pasos esenciales para el funcionamiento del bot.

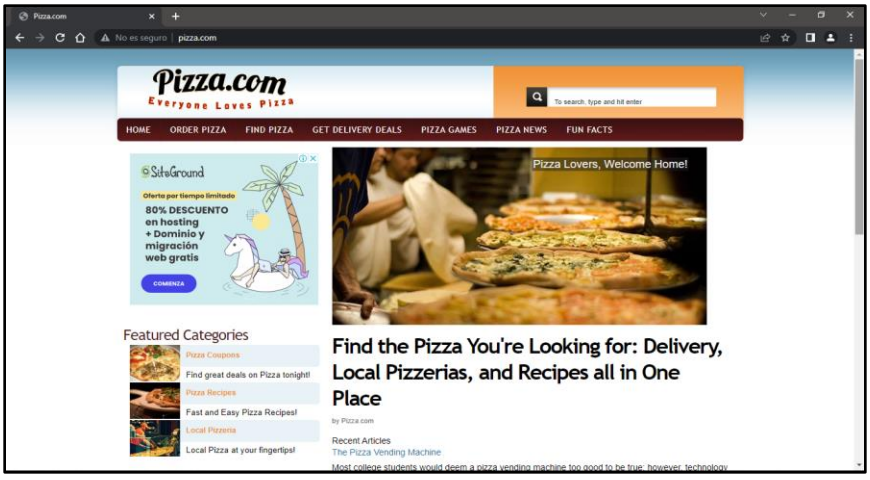
A continuación, se detallan las actividades realizadas en cada una de las etapas de desarrollo.

Obtención de las cookies almacenadas en Chrome

Cuando se navega por internet haciendo uso de cualquier navegador web (Chrome, Opera, Firefox, etc), la mayoría de páginas web generan cookies que son almacenadas de manera física en el directorio “Network” dentro de la carpeta raíz del navegador. Estas cookies crean datos que son guardados en el ordenador del usuario y proporcionan a las páginas web información de lo que se guardó previamente, de esta manera se ofrece una mejor experiencia al usuario cuando visita nuevamente la misma página. En este contexto, la primera parte del desarrollo del bot permite identificar la ruta en donde se guardan dichas cookies para ejecutar su análisis. Se visitó una página web (en este caso se optó por reutilizar el dominio “pizza.com”) dentro del navegador Chrome y automáticamente se generan las cookies dentro del directorio mencionado.

Figura 48

Visita de página web para la obtención de las cookies que genera el dominio

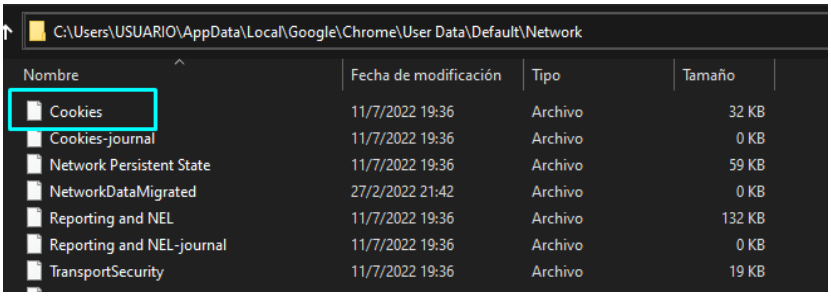


Nota. La figura representa cualquier página web que contiene cookies.

La ubicación del archivo “Cookies” depende del navegador utilizado. En el caso de Chrome la ruta para acceder a dicho directorio es ***“C:\Users\Usuario\AppData\Local\Google\Chrome\User Data\Default\Network”***, el cual da como resultado la visualización de una lista de archivos como se muestra en la Figura 49.

Figura 49

Ubicación de las cookies generadas en el ordenador

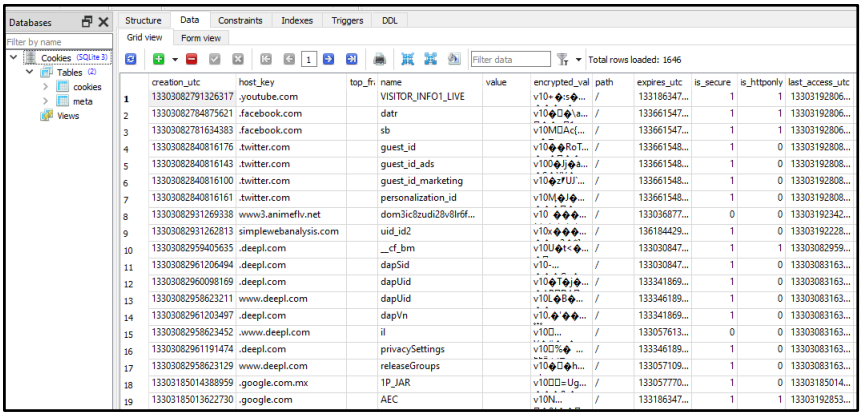


Nota. Para acceder a este directorio en Windows copiamos la ruta mostrada en la figura, cambiando la palabra “USUARIO” por el nombre de nuestro equipo.

Una característica del archivo “Cookies” es que no es legible a primera vista, pues representa a una base de datos embebida en formato SQLite. Para la visualización de esta pequeña base de datos se instaló el programa SQLiteStudio. Como se muestra en la Figura 50, esta herramienta muestra el contenido del archivo “Cookies” en donde destaca la tabla “cookies”, que almacena todas las cookies generadas por las páginas web visitadas en el navegador Chrome.

Figura 50

Visualización de la información de las cookies con SQLiteStudio



Nota. En esta base de datos se almacenan todas las cookies de las páginas web que visitamos en Chrome.

Los registros almacenados en esta tabla son importantes para el diseño del análisis del bot, por lo que, en esta etapa se analizaron y seleccionaron los campos que servirán de base para el estudio de dichas cookies. Entre todos los campos que componen la tabla “cookies”, las variables más importantes que se seleccionaron fueron las siguientes:

- *host_key*
- *creation_utc*
- *expires_utc*
- *is_secure*
- *is_httponly*

En la Tabla 12, se describen los detalles teóricos que representan cada una de estas columnas o campos en una cookie.

Tabla 12

Características de los campos seleccionados que componen una cookie

Campo	Detalle
host_key	Representa el dominio o página web de la cookie generada por la página web visitada por el usuario
creation_utc	Corresponde a la fecha y hora en qué se creó la cookie. Este dato es almacenado en formato Unix epoch, el cual es el número de segundos que han transcurrido desde el 1 de enero de 1601.
expires_utc	Corresponde a la fecha y hora de caducidad de la cookie. Al igual que creation_utc, este se encuentra en formato Unix epoch.
is_secure	El valor [0] en el atributo is_secure, representa que el navegador puede enviar la cookie a través de un canal o conexión HTTP. Lo cual permite que los atacantes accedan a las cookies. El valor [1] en el atributo is_secure, representa que la cookie se utiliza a través de HTTPS y se transmite de forma segura en texto cifrado, protegiendo la confidencialidad de la cookie.
is_httponly	El valor [0] en el atributo is_httponly, indica que la cookie no está protegida ante ataques XSS por lo que existe la posibilidad de acceder a sus datos. El valor [1] en un atributo is_httponly, prohíbe que se acceda a la cookie a través de scripts del lado del cliente, en específico con la propiedad *document.cookie*.

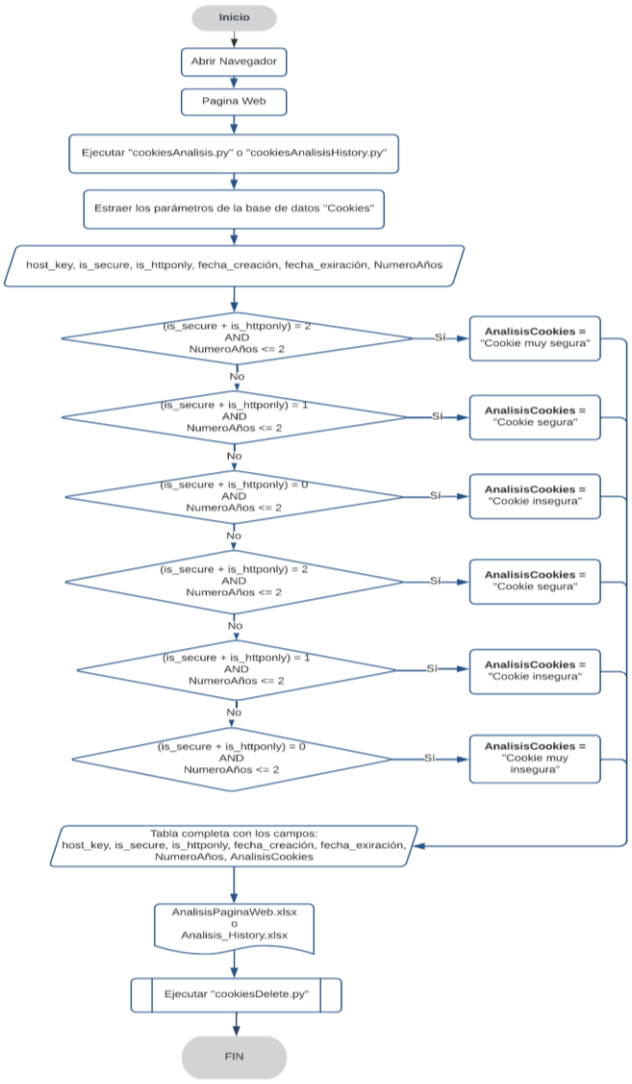
Nota. En la tabla se indican los principales campos que posee una tabla de cookies al ser analizada dentro del software SQLite.

Modelo en Python para el análisis del archivo "Cookies"

En esta etapa se utilizó el lenguaje de programación Python, para extraer y preparar los datos y establecer el modelo de análisis de las cookies generadas por los sistemas web. Para cumplir con la funcionalidad del bot se crearon dos scripts, el primero corresponde al análisis de las cookies y el segundo a la eliminación de las cookies generadas por cada sitio web visitado. A continuación, se presenta la Figura 51, en donde se detalla la esquematización gráfica del algoritmo de funcionamiento del bot.

Figura 51

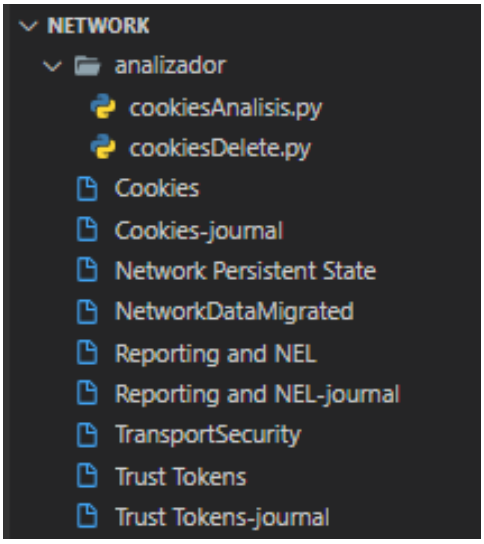
Diagrama de flujo del modelo en Python para el análisis de cookies



Como se mencionó, se crearon dos scripts en Python los cuales fueron colocados en el mismo directorio en donde se encuentra la base de datos de las cookies. Se optó por esta solución debido a la facilidad de acceso a la ruta de la base de datos dentro de los scripts.

Figura 52

Ubicación de los scripts de python



Nota. En la ubicación de los scripts se puede observar que se crearon dos scripts necesarios para el proceso de análisis del bot.

Al momento de comenzar un proyecto con este lenguaje de programación es importante tener instalado los paquetes y librerías de Python en nuestra máquina, ya que son necesarios para ejecutar los programas que se vayan creando. Una vez creados los archivos se pasó a la codificación del algoritmo utilizando Visual Studio Code.

Script cookiesAnalizar.py. En este script se implementó como elemento principal el módulo o librería “sqlite3”, esta librería ofrece una interfaz sencilla para interactuar con la base de datos “Cookies”. Para el proceso de acceso y consultas de datos, sqlite3 nos ofrece las siguientes funciones:

- **sqlite3.connect():** Crea una conexión con la base de datos.
- **.cursos():** Permite recorrer y procesar los resultados de una función.

- **.execute():** Permite ejecutar consultas SQL hacia la base de datos.
- **.close():** Cierra la conexión con la base de datos.

Con el conocimiento de estas funciones, se procedió a establecer la primera consulta, la que se visualiza en la Figura 53, cuya función se centró en contar el número de filas que tiene la tabla “cookies”. Para identificar las cookies generadas por cada una de las páginas web visitadas se creó la variable “dom”.

Esta variable “dom” recibe una entrada del usuario, pero al ser implementada en el bot este ingresará de manera automática el nombre del dominio que se visitará en el navegador. La finalidad de esto es asignar un identificador a las cookies generadas por el dominio, abriendo la posibilidad de realizar un mejor análisis y poder clasificar los resultados por cada uno de los dominios.

Figura 53

Consulta para el conteo total de las cookies generadas por un dominio

cookiesAnalisis.py

analizador > cookiesAnalisis.py > ...

```
import sqlite3 as sql
import pandas as pd

#-------------------------------------------------------------------------------
con = sql.connect('./Cookies')
cur = con.cursor()

print('\n          **** ANALISIS DE XSSS ****')
print(' | TESIS DE GRADO: QUINATOA JORDY - VILLARES JULIO | \n')

dom = input("Nombre del dominio: ")

#Consulta para determinar el numero de cookies generadas ---------------------
cur.execute('SELECT * FROM cookies')
num = len(cur.fetchall())
print('\n-----------------------------------------------------------------')
print('Se generaron:  <', num, '>  cookies del dominio <',dom,'>')
print('-----------------------------------------------------------------\n')
#-------------------------------------------------------------------------------
```

Nota. Para el conteo de las cookies se asignó la variable “num”, la cual guarda el total de los registros obtenidos en la consulta SQL.

Con el dominio principal definido se creó una consulta en donde se extraen desde la base de datos los campos que se describieron en la Tabla 12, al ser una consulta simple como se muestra en la Figura 54, solo se utilizó la declaración SELECT y el llamado de los campos necesarios para el análisis. En este tipo de consultas que conlleva la extracción de todos los registros de una tabla se hace uso de la función ".fetchall()" acompañado de un "for" para leer todas las filas y mostrarlas como resultado de la consulta.

Figura 54

Estructura de la consulta para la extracción de los datos de la tabla "cookies"

```
#Consulta para listar las cookies-------------------------------------------------------------------
print('================================')
print('  LISTA DE COOKIES GENERADAS')
print('================================\n')

print('|  DOMINIO  |  HTTPONLY  | SEGURA | CREACION COOKIE | EXPIRACION COOKIE |')
cur.execute('SELECT host_key, is_httponly, is_secure, creation_utc, expires_utc FROM cookies')
rows = cur.fetchall()

for row in rows:
    print(row)
```

Nota. El uso de esta consulta permite acceder de forma rápida a los registros que contiene la base de datos sqlite "Cookies".

Se destaca que dentro del análisis para determinar si una página es vulnerable a ataques XSS es analizando el tiempo de duración de sus cookies, por consiguiente, se transformaron a texto legible los atributos "creation_utc" y "expires_utc". Estos campos se encuentran almacenados en formato Unix Epoch en una medida de microsegundos, para convertirlos a un dato de tipo fecha se aplicó la siguiente solución:

"date(campo_utc/1000000 + (strftime('%s', '1601-01-01')), 'unixepoch', 'localtime')"

Como se muestra en la Figura 55, la solución forma parte de la consulta SQlite, por lo que se almacenó en una variable para su manejo dentro del código. En cuanto a la consulta, esta imprime el dominio de la cookie, fecha de creación y fecha de expiración en formato fecha.

Figura 55

Consulta SQLite de las fechas de creación y expiración de las cookies

```
#Consulta para mostrar las fechas transformadas de UTC a fecha----------
print('\n================================')
print('  TRANSFORMACION DE TIEMPO')
print('================================\n')
print('|  DOMINIO  | CREACION COOKIE | EXPIRACION COOKIE |')

fecha_creacion = "date(creation_utc/1000000 + (strftime('%s', '1601-01-01')), 'unixepoch', 'localtime')" 
fecha_expiracion = "date(expires_utc/1000000 + (strftime('%s', '1601-01-01')), 'unixepoch', 'localtime')"

cur.execute(""" SELECT
                    host_key,
                    """+fecha_creacion+""",
                    (CASE
                        WHEN expires_utc > 0
                        THEN """+fecha_expiracion+"""
                        ELSE 'Fecha no definida'
                    END)
                FROM
                    cookies""")
rows1 = cur.fetchall()

for row1 in rows1:
    print(row1)
#-----------------------------------------------------------------------------------------------------
```

Nota. Para mostrar la fecha de expiración se utilizó una condición para controlar las cookies que se generan con una fecha “0”, lo que indica que tienen una duración muy corta.

Un ejemplo de la salida de esta consulta, se presenta el siguiente ejemplo en el que se observa cómo están representadas las fechas:

Figura 56

Visualización de las fechas transformadas de unix epoch a fecha legible

```
================================
  TRANSFORMACION DE TIEMPO
================================

|  DOMINIO  | CREACION COOKIE | EXPIRACION COOKIE |
('.casalemedia.com', '2022-07-17', '2023-07-17')
```

Nota. La figura representa el resultado que se genera en consola al ejecutar la consulta mostrada en la figura anterior.

Con la obtención y transformación de todos los datos necesarios que serán la base de funcionamiento del bot, se planteó la consulta final cuyo objetivo es establecer si una cookie es

segura o tiene vulnerabilidades de tipo XSS. Previo a esto se planteó una variable con el nombre “num_years” que refleja la diferencia de los años de creación y caducidad de las cookies.

Con los datos listos, se estructuró la consulta o tabla final como se muestra en la Figura 57, en donde por medio de la declaración SELECT se llama a los campos “host_key”, “is secure”, “is_httponly”, “fecha_creación”, “fecha_expiración”, “num_years” y “AnalisisCookies”. Este último atributo contiene el resultado final que establece si una cookie puede llegar a ser segura o no, basándose en las siguientes condiciones:

Tabla 13

Parámetros para establecer la calificación de cookies seguras e inseguras

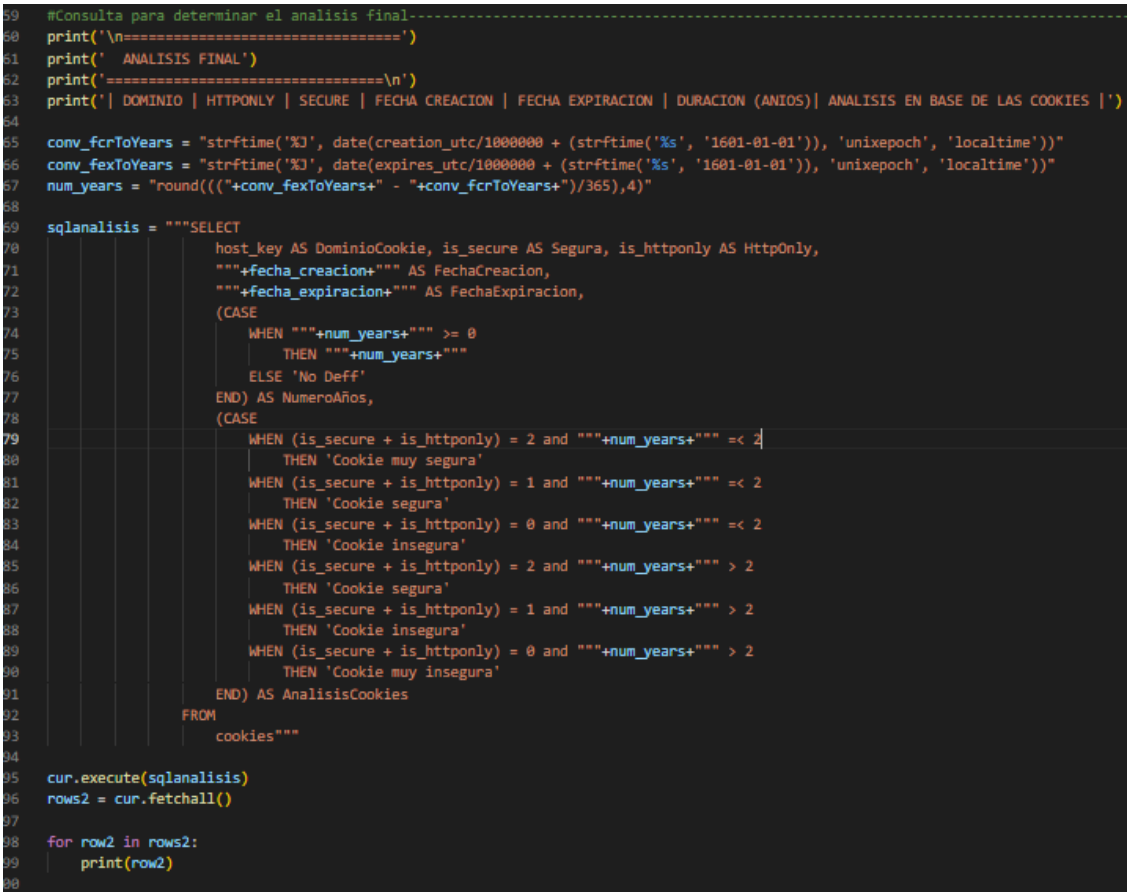
is_httponly	is_secure	num_years	Cookie muy segura	Cookie segura	Cookie insegura	Cookie muy insegura
1	1	<= 2	x			
1	0	<= 2		x		
0	1	<= 2		x		
0	0	<= 2			x	
1	1	> 2		x		
1	0	> 2			x	
0	1	> 2			x	
0	0	> 2				x

Nota. La tabla representa todos los casos que podemos encontrar al realizar el análisis de las cookies en base a las variables is_httponly, is_secure y num_years.

Con los parámetros obtenidos de la Tabla 13, se establecieron las condiciones necesarias para la construcción de la consulta que analizará las cookies. A continuación, se presenta la implementación de las sentencias SQL utilizadas para formar la consulta final que corresponde al análisis de las cookies.

Figura 57

Consulta SQLite para el análisis final de las cookies



```
#Consulta para determinar el analisis final----------------------------------------------------------------------------------------------------
print('\n================================')
print('  ANALISIS FINAL')
print('================================\n')
print('| DOMINIO | HTTPONLY | SECURE | FECHA CREACION | FECHA EXPIRACION | DURACION (ANIOS)| ANALISIS EN BASE DE LAS COOKIES |')

conv_fcrToYears = "strftime('%J', date(creation_utc/1000000 + (strftime('%s', '1601-01-01')), 'unixepoch', 'localtime'))"
conv_fexToYears = "strftime('%J', date(expires_utc/1000000 + (strftime('%s', '1601-01-01')), 'unixepoch', 'localtime'))"
num_years = "round((("+conv_fexToYears+" - "+conv_fcrToYears+")/365),4)"

sqlanalisis = """SELECT
                    host_key AS DominioCookie, is_secure AS Segura, is_httponly AS HttpOnly,
                    """+fecha_creacion+""" AS FechaCreacion,
                    """+fecha_expiracion+""" AS FechaExpiracion,
                    (CASE
                        WHEN """+num_years+""" >= 0
                            THEN """+num_years+"""
                        ELSE 'No Deff'
                    END) AS NumeroAños,
                    (CASE
                        WHEN (is_secure + is_httponly) = 2 and """+num_years+""" =< 2
                            THEN 'Cookie muy segura'
                        WHEN (is_secure + is_httponly) = 1 and """+num_years+""" =< 2
                            THEN 'Cookie segura'
                        WHEN (is_secure + is_httponly) = 0 and """+num_years+""" =< 2
                            THEN 'Cookie insegura'
                        WHEN (is_secure + is_httponly) = 2 and """+num_years+""" > 2
                            THEN 'Cookie segura'
                        WHEN (is_secure + is_httponly) = 1 and """+num_years+""" > 2
                            THEN 'Cookie insegura'
                        WHEN (is_secure + is_httponly) = 0 and """+num_years+""" > 2
                            THEN 'Cookie muy insegura'
                    END) AS AnalisisCookies
                FROM
                    cookies"""

cur.execute(sqlanalisis)
rows2 = cur.fetchall()

for row2 in rows2:
    print(row2)
```

Nota. La expresión “(is_secure + is_httponly)” hace a referencia a que estas variables están establecidas con el valor “0” o con el valor “1” según corresponda, por lo que al ser sumadas representan uno de los casos presentados en la Tabla 13.

Se agregó una nueva sección en el script, que por medio del uso de la librería “pandas” se extrae el contenido de la tabla para guardarla en un archivo de Excel. Como se visualiza en la Figura 58, se utilizó la variable “dom” para guardar el archivo excel con el nombre de la página web que se haya visitado. Este proceso se realizó para tener una constancia de los resultados que genera el bot por cada uno de los dominios, presentándolos de una manera más ordenada y accesibles.

Figura 58

Exportación de los resultados a un archivo de Excel

```
#Enviar los resultados a un excel
analisis = pd.read_sql(sql = sqlanalisis , con = con)
nombre = dom.replace('://','')
analisis.to_excel("C:\\Users\\JORDY\\Desktop\\AnalisisXSS\\AnalsisPaginasWeb\\analisis_("+nombre+").xlsx", index=False)
#-----------------------------------------------------------------------------------------------------------------------

con.close()
```

Nota. En la variable “nombre” se utilizó la función “replace” para eliminar los caracteres especiales del dominio, con la finalidad de guardar correctamente el archivo de Excel.

Script cookiesDelete.py. Con el análisis de varias páginas web usando nuestro bot, se desarrolló el script presentado en la Figura 59, que cuenta con sentencia SQL para ejecutar la acción de eliminar los registros almacenados en la base de datos. Su finalidad es ejecutarse cada vez que el bot analice una página web, ya que es necesario que la base de datos previamente esté vacía para presentar los nuevos resultados en base a las cookies generadas por cada uno de los dominios visitados.

Figura 59

Script para eliminar registros de la base de datos

cookiesDelete.py

analizador > cookiesDelete.py > ...

```
import sqlite3 as sql

con = sql.connect('./Cookies')
cur = con.cursor()

cur.execute('DELETE FROM cookies')
con.commit()

con.close()
```

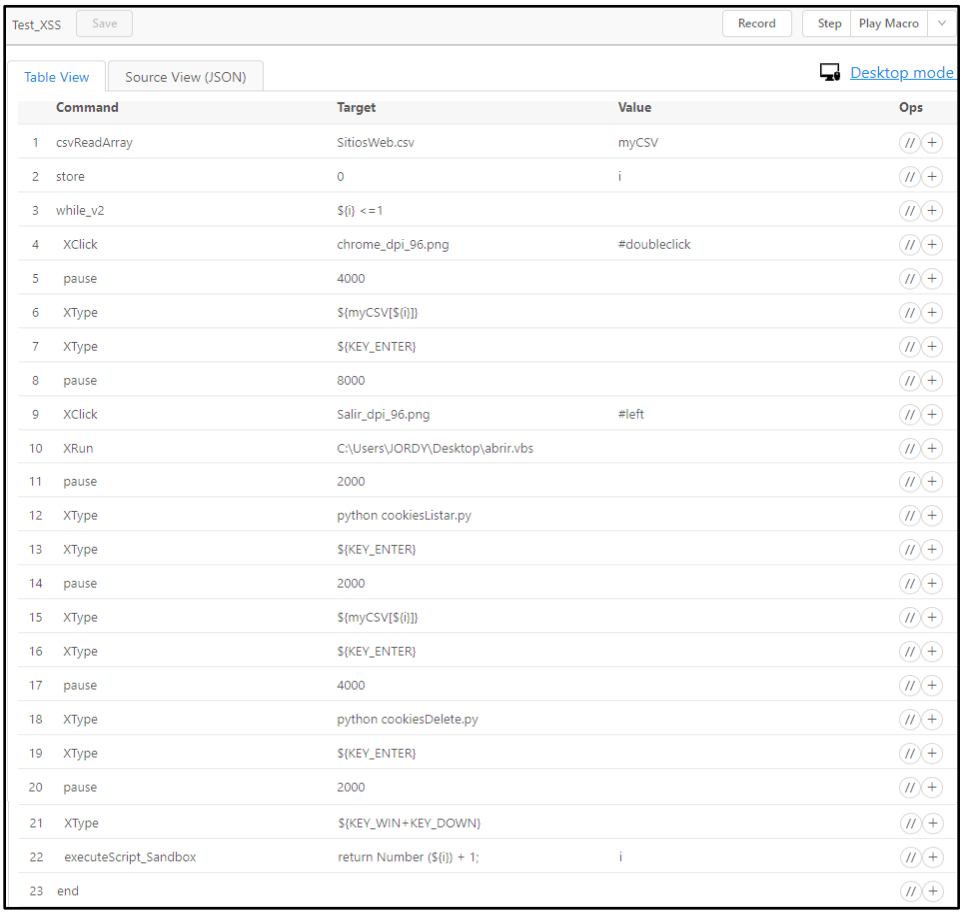
Nota. Una vez eliminados los registros se utilizó el comando “commit()” para confirmar los cambios realizados a la base de datos.

Desarrollo del Bot en el software de UI.Vision

Para utilizar UI.Visión se descargó la extensión oficial desde el sitio web “https://ui.vision/”. Esta herramienta cuenta con dos versiones de automatización, la primera que se centra en una página web y la segunda versión que se centra en la automatización de escritorio, es decir, podemos acceder a los programas y archivos de nuestra PC. Una vez instalada la extensión y los XModules, se aplicaron los comandos que se muestran en la Figura 60, en donde a partir de la documentación de UI.Visión se conoció la funcionalidad de cada uno logrando así la construcción del bot para análisis de XSS en sistemas web.

Figura 60

Estructura en UI.Visión del bot



	Command	Target	Value	Ops
1	csvReadArray	SitiosWeb.csv	myCSV	// +
2	store	0	i	// +
3	while_v2	${i} <=1		// +
4	XClick	chrome_dpi_96.png	#doubleclick	// +
5	pause	4000		// +
6	XType	${myCSV[${i}]}		// +
7	XType	${KEY_ENTER}		// +
8	pause	8000		// +
9	XClick	Salir_dpi_96.png	#left	// +
10	XRun	C:\Users\JORDY\Desktop\abrir.vbs		// +
11	pause	2000		// +
12	XType	python cookiesListar.py		// +
13	XType	${KEY_ENTER}		// +
14	pause	2000		// +
15	XType	${myCSV[${i}]}		// +
16	XType	${KEY_ENTER}		// +
17	pause	4000		// +
18	XType	python cookiesDelete.py		// +
19	XType	${KEY_ENTER}		// +
20	pause	2000		// +
21	XType	${KEY_WIN+KEY_DOWN}		// +
22	executeScript_Sandbox	return Number (${i}) + 1;	i	// +
23	end			// +

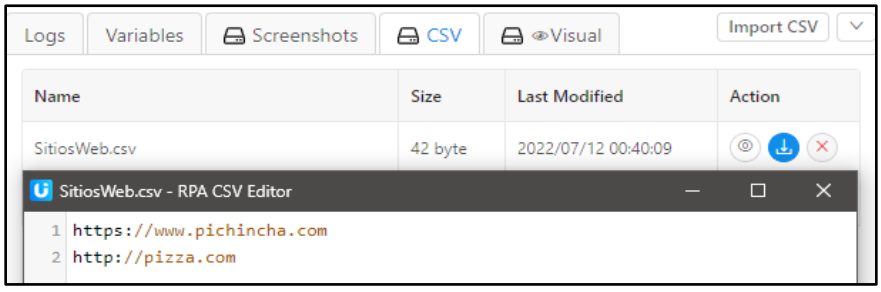
Nota. La figura muestra todos los comandos, variables y valores utilizados en la construcción del bot para análisis de vulnerabilidades XSS.

Una característica de UI.Vision es que el proceso de automatización ya sea web o de escritorio está compuesto por un conjunto de pasos que se ejecutan de manera secuencial. En base a la Figura 60, las tareas que se automatizaron en cada uno de los pasos cumpliendo con las siguientes funciones:

- ***Paso 1:*** Aquí se utilizó el comando “csvReadArray” para leer la información de un archivo CSV. Este archivo funciona como una lista de páginas web que el bot analizará. Pará acceder a este archivo es necesario importarlo a la carpeta "Datasource" de UI.Vision que se encuentra en nuestra PC, para visualizarlo y utilizarlo en la herramienta.

Figura 61

Archivo CSV para el análisis de las páginas web



Nota. En la imagen se muestra el archivo CSV utilizado para el análisis de las páginas web, el cual ya debe contener las URLs de las páginas que se van analizar.

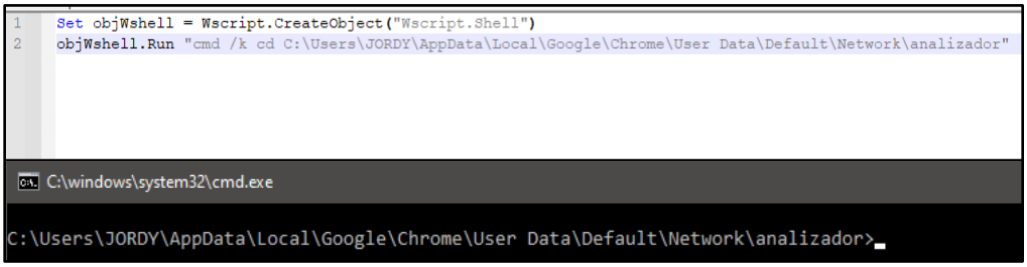
- ***Paso 2:*** En este paso se creó una variable con el comando “store”, que está inicializada en 0. Su finalidad es utilizarla como iteración al momento de recorrer la lista de las páginas web.
- ***Paso 3:*** Se colocó la instrucción “While”, que nos indica el número de veces que queremos que se ejecute el bot. En pocas palabras, con esto se está indicando cuántas páginas web del archivo CSV se van a analizar.
- ***Paso 4:*** En este punto se inician las tareas que el bot ejecutará por cada una de las páginas web. Aquí se utilizó el comando “XClick” para abrir el acceso directo de Chrome

que se encuentra en el escritorio de la PC, posteriormente se añade el valor “#doubleclick” para dar doble click sobre el icono y abrir el navegador.

- ***Paso 5:*** Se programó una pausa de 4 segundos para dar tiempo a que el navegador se abra y esté listo para la búsqueda.
- ***Paso 6:*** Una vez ingresado al navegador web se utilizó el comando “Xtype” y el destino “$(myCSV[$(i)]”, con el fin de acceder al valor (página web) de una fila del archivo CSV. Al obtener este valor el bot procede a escribirlo en el buscador del navegador.
- ***Paso 7:*** Con el uso de la clave especial “${KEY_ENTER}” se realiza la acción de dar un ENTER, para ingresar a la página web.
- ***Paso 8:*** Se programó una pausa de 8 segundos sobre la página web para que se generen las cookies.
- ***Paso 9:*** En este paso se utilizó el comando “XType” para dar click sobre el icono de cerrar del navegador.
- ***Paso 10:*** Desde este paso ejecuta el script en python “cookiesAnalisis.py”, para esto es necesario acceder a la consola de comandos (CMD) de Windows. Para simplificar estos pasos y acceder de manera más rápida a la carpeta donde se encuentra el script, se utilizó el comando “XRun” para ejecutar un vbscript, como se visualiza en la Figura 62.

Figura 62

Vbscript para el acceso a “cookiesAnalizar.py” desde la CMD

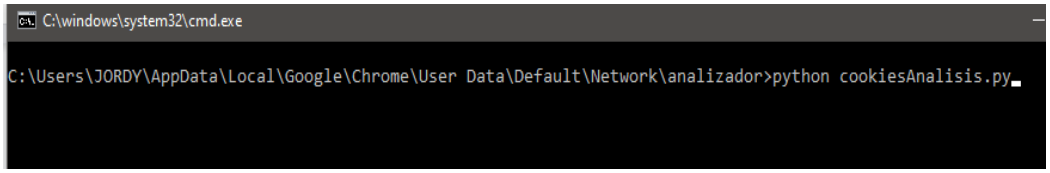


Nota. La figura representa un método directo para ingresar a un directorio desde el Símbolo del Sistema CMD (Command Prompt).

- ***Paso 11:*** Se programó una pausa de 2 segundos hasta que cargue bien el CMD.
- ***Paso 12:*** Con el comando “Xtype” el bot escribe la instrucción para realizar la ejecución dell script “python cookiesAnalisis.py”

Figura 63

Escritura del comando para la ejecución de "cookies Análisis.py"

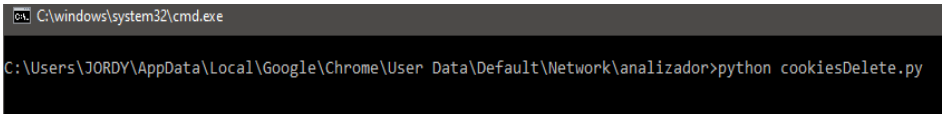


Nota. La figura representa el texto que el bot escribirá automáticamente al ejecutarse y llegar a este paso.

- ***Paso 13:*** Se volvió a utilizar la clave especial ENTER para ejecutar el script.
- ***Paso 14:*** Se programó una pausa de 2 segundos para esperar a que se complete la ejecución del script.
- ***Paso 15:*** Se utilizó el comando “Xtype” y el destino “$(myCSV[$(i)]” para escribir el nombre de la página web a la que se está ejecutando el análisis. Aquí se representa la variable de entrada “DOM” que se utilizó en la Figura 63.
- ***Paso 16:*** Se volvió a utilizar la clave especial ENTER para seguir con la ejecución del script.
- ***Paso 17:*** Se programó una pausa de 4 segundos hasta que se complete la ejecución del script, arrojando como resultado la tabla “Análisis” que contiene los resultados que posteriormente serán guardados en un archivo de Excel.
- ***Paso 18:*** Con el comando “Xtype” el bot escribe la instrucción para realizar la ejecución del script “python cookiesDelete.py”. Este último paso se ejecuta para preparar la base de datos para los nuevos registros que arroje el siguiente análisis que se realice sobre las siguientes páginas web almacenadas en el archivo CSV.

Figura 64

Escritura del comando para la ejecución de "cookiesDelete.py"



Nota. La figura representa el texto que el bot escribirá automáticamente al ejecutarse y llegar a este paso.

- ***Paso 19:*** Se utiliza la clave especial ENTER para ejecutar el script.
- ***Paso 20:*** Se programó una pausa de 2 segundos hasta que los registros de las cookies sean eliminados y que el estado de la base de datos se guarde.
- ***Paso 21:*** En este paso se utilizó la combinación de las claves especiales “${KEY_WIN + KEY_DOWN}” para realizar la acción de minimizar el CMD en donde se encuentran los resultados del análisis.
- ***Paso 22:*** Aquí se utilizó la expresión "return Number (${i}) +1;", la cual indica que al realizar todo el proceso del WHILE le aumenta una iteración accediendo de esta manera al siguiente elemento de la lista de páginas web del archivo CSV.
- ***Paso 23:*** Para finalizar se ejecutó el cierre de la instrucción WHILE por lo que se utilizó la instrucción “END”. Aquí el proceso se vuelve a repetir hasta que cumpla la condición del WHILE.

Análisis del historial de visitas de un usuario en Chrome

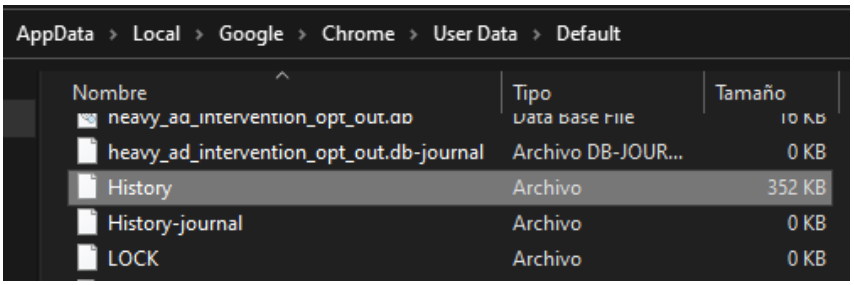
A medida que vamos navegando por el internet nuestra actividad en la red se va almacenando a lo largo del tiempo, este espacio se lo conoce como historial de navegación. El historial del navegador almacena todos los datos e información de las páginas web visitadas, generando un peligro debido a que puede revelar mucha información pública o privada sobre un usuario.

De acuerdo a nuestro estudio, existe una estrecha relación entre las páginas almacenadas con la generación de las cookies, que son otro factor a analizar ya que tener una gran cantidad de cookies en nuestro equipo aumenta las posibilidades de sufrir ataques XSS, en especial cuando se visitan páginas web sospechosas o que contienen publicidad.

Con esta afirmación, se estableció un método de análisis para todas las cookies que un usuario puede generar en un tiempo determinado y a medida que va visitando diferentes páginas web. Para implementar este método, el primer paso fue identificar la ruta en donde se almacena el archivo con el historial de visitas de un usuario de Chrome.

Figura 65

Ruta en donde se almacena el historial de Chrome

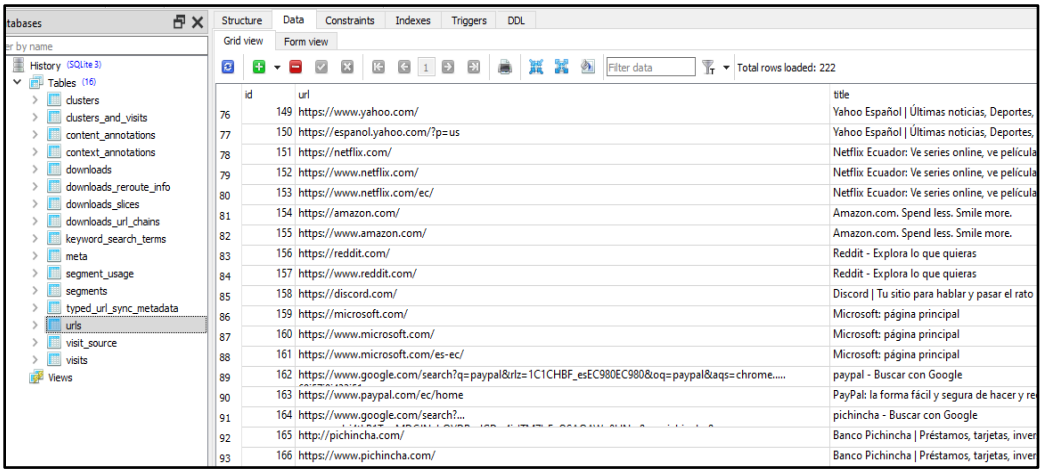


Nota. Para acceder a este directorio se debe ingresar a la ruta "*C:\Users\Usuario\AppData\Local\Google\Chrome\User Data\Default".*

Este archivo representa una base de datos sqlite3, para poder visualizar las tablas y registros que contiene se lo movió a un directorio en el escritorio. Esta acción fue necesaria ya que el archivo se bloquea cuando el navegador Chrome se encuentra abierto. Una vez trasladado a otro directorio se lo abrió con la herramienta SQLiteStudio, dando como resultado los registros mostrados en la Figura 66.

Figura 66

Visualización de las tablas y registros que contiene el archivo “History”



Nota. Para acceder a las URLs del historial se debe dar click en la tabla “urls”.

Analizada la estructura y registros que contiene la base de datos “History” se creó un script en python con el nombre “*exportarHistory.py*”. Su función es extraer los URLs de la tabla “url” para exportarlos a un archivo CSV. En la Figura 67 se muestra el código implementado para ejecutar la consulta de dicha acción.

Figura 67

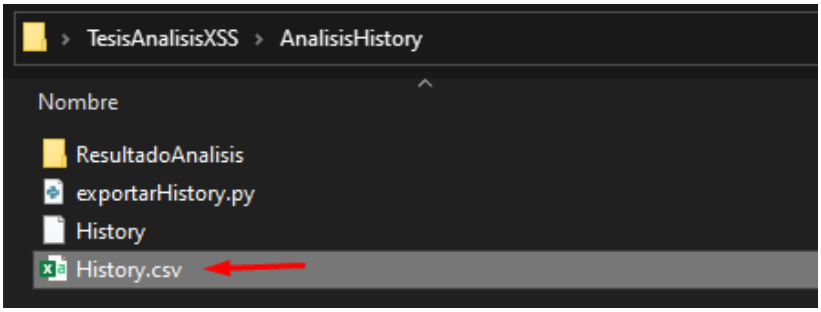
Script en Python para exportar las URLs del historial a un formato CSV

Nota. Se almacenaron los URLs en un formato CSV debido a la opción que nos ofrece UI.Vision para trabajar con este tipo de archivos.

El script mostrado en la figura anterior puede ser utilizado para cualquier archivo “History” del que se desee exportar las URLs. En la Figura 68 se observa que al ejecutar el script de python, se guarda el archivo CSV en la ruta asignada.

Figura 68

Archivo de URLs exportadas desde la base de datos “History” de Chrome

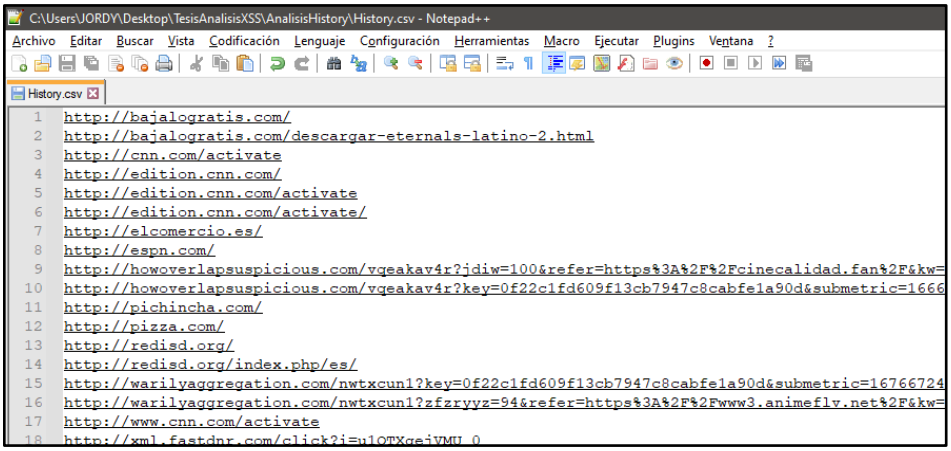


Nota. Visualización del resultado que se obtiene al ejecutar “exportarHistory.py”

Como se muestra en la Figura 69 al abrir el archivo *History.csv* se observan todas las URLs correspondientes al historial de navegación que registró un usuario dentro del navegador web. Con esto ya se pueden preparar los datos y elementos necesarios para realizar el análisis mediante el uso UI.Visión RPA.

Figura 69

Archivo de URLs exportadas desde la base de datos “History” de Chrome



Nota. El número de las URLs está dado por el tamaño del historial.

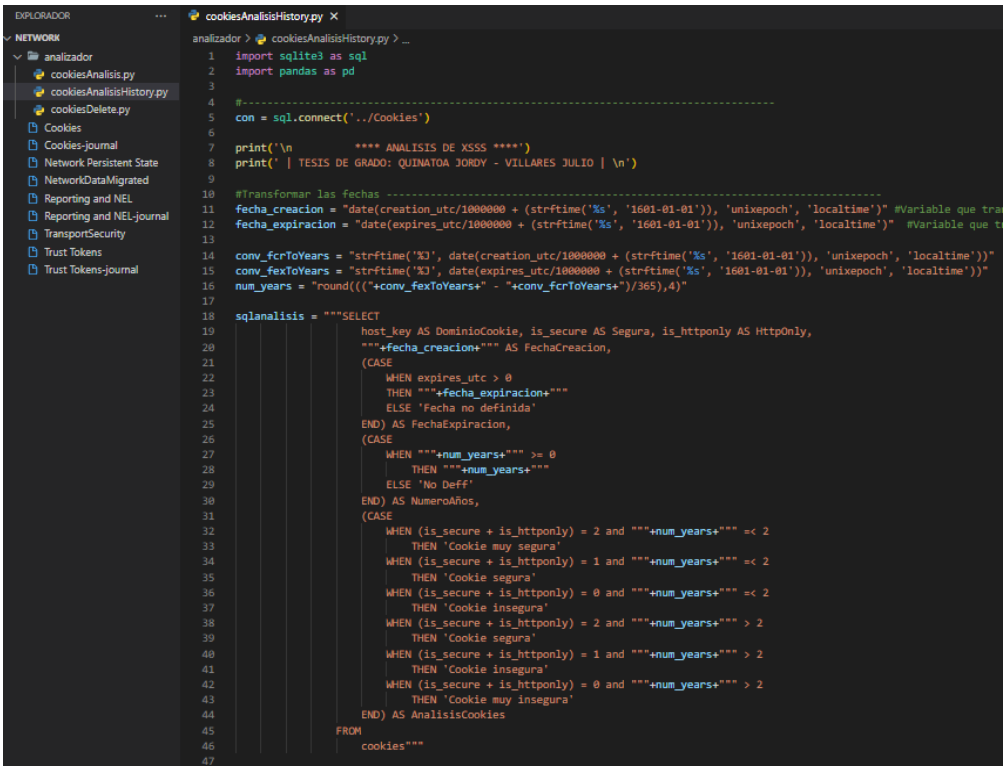
Análisis de las cookies del historial de Chrome de un usuario

Uno de los principales objetivos de obtener y acceder al historial de visitas web de un usuario, es analizar las cookies que se han creado y que se han almacenado localmente en su equipo, dado a que estas cookies contienen o acceden a información personal que puede ser vulnerada por ataques informáticos de tipo XSS. En este contexto, para analizar estas cookies se programó un nuevo script con el nombre “cookiesAnalisisHistory.py” aplicando la misma lógica presentada en la sección “*Modelo en Python para el análisis del archivo Cookies”*.

Este nuevo script establece una conexión con la base de datos que contiene las cookies por medio de la sentencia *sql.connect.* Seguido a esto se programó la lógica necesaria para el análisis de las cookies tomando como referencia los criterios planteados en la sección anterior.

Figura 70

Script para realizar el análisis de las cookies en base al historial del navegador



Nota. En la figura se muestra una consulta SQL con todas las variables y campos necesarios para analizar todas las cookies del historial.

De igual forma se utilizó la librería “pandas” para exportar los resultados de la consulta a un archivo Excel. Este archivo nos ayudará a establecer un análisis más detallado para calcular la probabilidad de que un usuario sea o no vulnerable a ataques XSS.

Figura 71

Exportación de los resultados “History” a un archivo de Excel

```
#Exportar los resultados a un excel
analisis = pd.read_sql(sql = sqlanalisis , con = con)
analisis.to_excel("C:\\Users\\JORDY\\Desktop\\TesisAnalisisXSS\\AnalisisHistory\\ResultadoAnalisis\\Analisis_History.xlsx", index=False)
#-----------------------------------------------------------------------------------------------------------------------------

con.close()

#Mostrar mensaje en cosola
print('||||| Resultados exportados correctamente |||||')
```

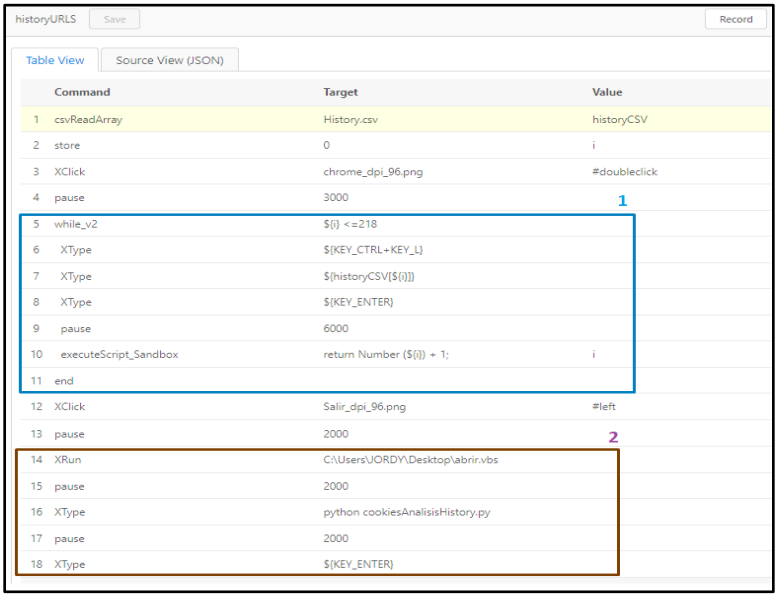
Nota. Los resultados se guardan en el directorio creado en el Escritorio.

Bot para el análisis de las cookies del historial de navegación

Para establecer este proceso, se creó una nueva macro dentro de la herramienta Ui.Vision, la misma que está implementada a partir de la secuencia mostrada en la Figura 72.

Figura 72

Exportación de los resultados “History” a un archivo de Excel



Nota. En la imagen se muestra la macro realizada en UI.Vision que se encuentra definido por dos secciones que se explican más adelante.

El bot implementado utiliza dos procesos bien definidos para establecer su funcionalidad, además permite ejecutar el análisis de las vulnerabilidades web existentes. Estos procesos fueron señalados en la Figura 72, pero se resumen a continuación:

1) En la primera sección de la macro se implementó un bucle de tipo While (Mientras), el cual contiene las instrucciones o comandos que cumplen la acción de abrir cada una de las páginas web o URLs del archivo CSV que se extrajeron desde la base de datos “History”. Al abrir cada una de estas páginas web, se generan las respectivas cookies necesarias para cumplir nuestro análisis.

2) Cumplida la condición del condicional WHILE que analiza todo el archivo CSV o un número determinado de páginas, se cumple la segunda sección que se conforma de instrucciones para ejecutar el script presentado en la Figura 68. Con esto obtenemos el archivo Excel con los resultados necesarios para establecer el análisis final.

Capítulo IV

Análisis e Interpretación de Resultados

Introducción del capítulo

El objetivo del siguiente capítulo es analizar e interpretar los resultados obtenidos al ejecutar los bots de búsqueda y análisis de vulnerabilidades en sitios web. Para esto, se han implementado las pruebas de rendimiento del bot desarrollado en UiPath, analizando el tiempo de ejecución que utiliza el bot para resolver el test de un sistema CAPTCHA. Luego, se ejecutó el bot en diferentes páginas web integradas con este sistema embebido de tipo hCatpcha, para determinar qué tan vulnerables son dichas páginas. Por otro lado, para las pruebas del bot que analiza las vulnerabilidades de tipo XSS, se ejecutó el script en diferentes páginas web, para determinar si estas son vulnerables o no, todo ello en base al estudio de las cookies que se almacenan en la carpeta raíz del navegador. Por último, se interpretan los resultados obtenidos del análisis comparativo de las herramientas RPA aplicadas previamente.

Resultados de la implementación del bot en los Sistemas CAPTCHA

Como se observa en la metodología de nuestra propuesta (Capítulo III), para el análisis de los sistemas CAPTCHA se implementó un bot desarrollado en UiPath Community Edition que utiliza tecnologías de visión artificial basado en la API de Google Cloud Vision para resolver este tipo de desafíos. Es importante mencionar que no es el único método para resolver los sistemas CAPTCHAs basados en imágenes. Otra forma de resolverlos es a través de extensiones para los navegadores web y mediante el uso de APIs de resolución de CAPTCHAS que prestan sus servicios por una suscripción mensual. En este contexto, se analizan dos de las alternativas más viables para la resolución de CAPTCHAs:

Resultados de la solución implementada con alternativas existentes

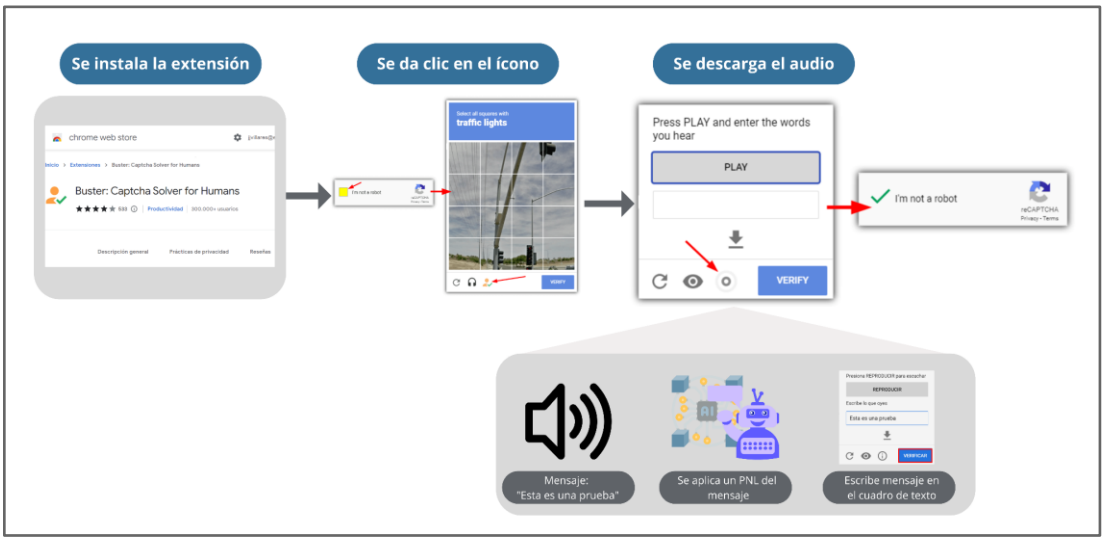
Buster: CAPTCHA Solver for Humans. Buster es una extensión para navegadores web (Chrome, Firefox, Mozilla, Opera) que ayuda a resolver los sistemas CAPTCHA que

aparecen en una página web. Los sistemas que pueden ser resueltos son los CAPTCHAs basados en texto y los reCAPTCHA V2. Sin embargo, a pesar de que es una extensión que facilita la resolución de CAPTCHA, resolver más de dos desafíos en un mismo día puede generar problemas de bloqueos temporales en la utilización de esta herramienta.

La forma de resolver los desafíos reCAPTCHA V2 es que no analiza directamente las imágenes proporcionadas, en vez de esto, descarga los audios que aparecen como segunda opción de resolución en el reCAPTCHA V2, luego los procesa por medio de la utilización de APIs de Procesamiento de Lenguaje Natural (PLN) y escribe lo que se menciona en el audio dentro de la caja de texto del CAPTCHA, logrando así pasar la prueba. En la Figura 73 se muestra una gráfica clara del funcionamiento que implementa la extensión Buster.

Figura 73

Funcionamiento de la extensión Buster CAPTCHA Solver for Humans



Nota. Para poder utilizar la herramienta solo basta con dar clic en el ícono de color amarillo que aparece en la parte inferior del reCAPTCHA.

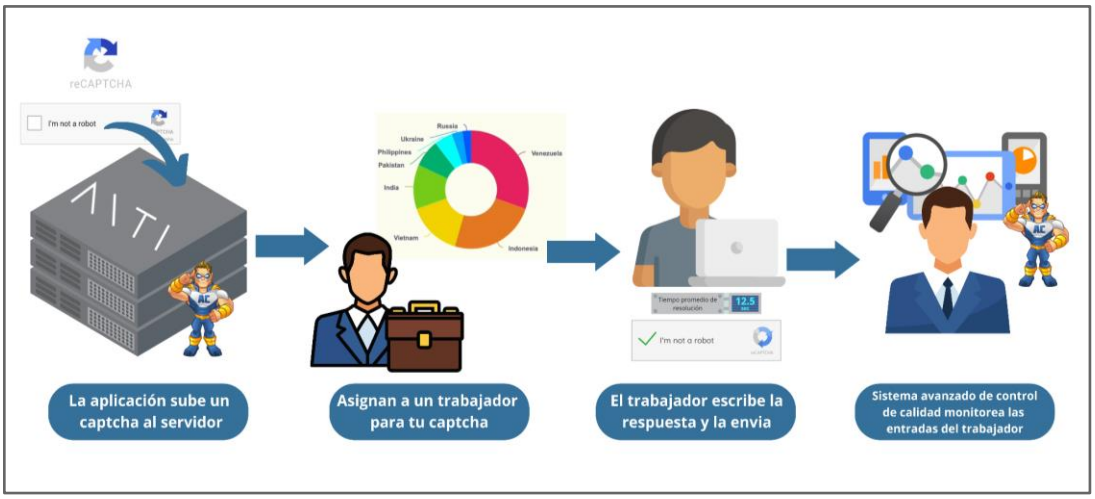
Anti-CAPTCHA Solving. Es un servicio online de pago para resolver cualquier CAPTCHA basado en imágenes, a través del uso de humanos reales para resolver los test, siendo uno de los servicios más económicos del mercado, con valores que comienzan desde

0.5 USD por cada 1000 CAPTCHAs analizados. Además, ofrece la ventaja de recargas que van desde 1 USD como mínimo, sin ningún cargo adicional. Tiene un valor de confiabilidad de 99% en la resolución de los retos y disponible las 24 horas del día y todos días del año.

Su funcionamiento es muy básico, la aplicación del usuario sube el CAPTCHA al servidor de Anti-CAPTCHA, se le asigna a un trabajador humano para que resuelva el CAPTCHA y se escribe la respuesta para enviarle nuevamente a la aplicación del cliente. El promedio de resolución del CAPTCHA oscila los 18 segundos dependiendo del tipo de CAPTCHA soportado y la complejidad del mismo. En la Figura 74 se muestra una gráfica más clara del funcionamiento de este servicio de resolución de CAPTCHAs (Anti, 2007).

Figura 74

Funcionamiento del servicio Anti-CAPTCHA Solving



Nota. Esta solución no ofrece una solución robotizada a un proceso, ya que utiliza humanos para resolver los desafíos. Es un servicio digital para la resolución de sistemas CAPTCHAs.

Análisis comparativo con la propuesta planteada. Estas dos soluciones (Buster y Anti-CAPTCHA) ofrecen otras alternativas de resolución para los sistemas CAPTCHAs. Por un lado, la alternativa gratis que implementa una extensión del navegador basada en un proyecto open source. Por otro lado, un servicio de resolución de sistemas CAPTCHAs con un pago

mensual. Para complementar, nuestra propuesta cuyo objetivo es demostrar las vulnerabilidades de los sistemas CAPTCHAs de tipo hCAPTCHA utilizando técnicas de visión artificial.

Para entender mejor las ventajas que ofrece cada solución, se realizó una comparación entre herramientas, mediante un análisis cualitativo de criterios en los que se exponen los puntos de vista de los autores de este trabajo, en base a la experiencia de usabilidad de las tres soluciones. En la Tabla 14 se muestran los resultados obtenidos.

Tabla 14

Análisis comparativo de las soluciones encontradas para resolver CAPTCHAs

Parámetros evaluados	Buster CAPTCHA Solver				Anti-CAPTCHA Solving				Bot con Visión Artificial			
	U1	U2	Prom	Cali. A	U1	U2	Prom	Cali. B	U1	U2	Prom	Cali. C
Tiempo de ejecución	6	7	6,5	**MEDIO**	8	7	7,5	**ALTO**	3	4	3,5	**BAJO**
Usabilidad	8	7	7,5	**ALTO**	9	8	8,5	**ALTO**	7	6	6,5	**MEDIO**
Efectividad	5	4	4,5	**MEDIO**	8	9	8,5	**ALTO**	7	5	6	**MEDIO**
Solución para automatización	6	5	5,5	**MEDIO**	1	2	1,5	**BAJO**	9	10	9,5	**ALTO**
Compatibilidad con navegadores	8	7	7,5	**ALTO**	6	7	6,5	**MEDIO**	7	9	8	**ALTO**
	TOTAL		**6,3**		**TOTAL**		**6,5**		**TOTAL**		**6,7**	

Nota. Cada una de los valores establecidos en la tabla serán detallados en los siguientes apartados.

Para determinar los resultados de la Tabla 14, es importante recalcar que el análisis cualitativo se basa en la experiencia de uso que tuvo cada autor de esta tesis, es así que la calificación final estará dada por los criterios ALTO MEDIO y BAJO, que fueron definidos en la escala de valores que se presenta en la Tabla 15:

Tabla 15

Escala de calificación del análisis cualitativo

Criterios	Valor	Descripción
7 AL 10	ALTO	Corresponde a que la solución ofrece características sólidas y sobresalientes en dicho parámetro en comparación con las demás soluciones.
4 AL 6.9	MEDIO	Corresponde a que la solución ofrece características que pueden ser mejorables y poco sobresalientes en dicho parámetro en comparación con las demás soluciones.
1 AL 3.9	BAJO	Corresponde a que la solución ofrece características ineficientes y débiles en dicho parámetro en comparación con las demás soluciones.

Nota. Para determinar cada uno de los valores se evaluó la experiencia de cada autor del proyecto frente a cada solución presentada.

A continuación, se determinan los valores y la descripción de los parámetros evaluados previamente:

- **Tiempo de ejecución**: El tiempo es uno de los parámetros más importante que se evalúa durante la ejecución de un proyecto de automatización, ya que la automatización de una tarea requiere de un proceso rápido, en comparación con las tareas que puede realizar un humano. En este contexto, se dará una calificación de ALTO a la solución que ejecute una tarea en menor tiempo, MEDIO a la solución que realice la tarea en un tiempo prudencial pero que no ofrece una respuesta inmediata y BAJO a la solución que realice la automatización de la tarea en mayor tiempo en comparación con las demás soluciones.
- **Usabilidad:** La usabilidad juega un papel importante en el desempeño correcto de una herramienta informática, por ende, a medida que una herramienta es funcional, también debe dar una buena experiencia de uso, por lo que el usuario debería sentirse cómodo

y adaptarse al funcionamiento. Para medir este criterio se dará una calificación de ALTO cuando la solución posea una interfaz amigable con el usuario y sea fácil de manejar, MEDIO cuando posea una interfaz poco amigable y que requiera más de un intento para poder entender su funcionamiento y BAJO cuando no sea entendible para el usuario y requiere muchos intentos para entender su funcionamiento.

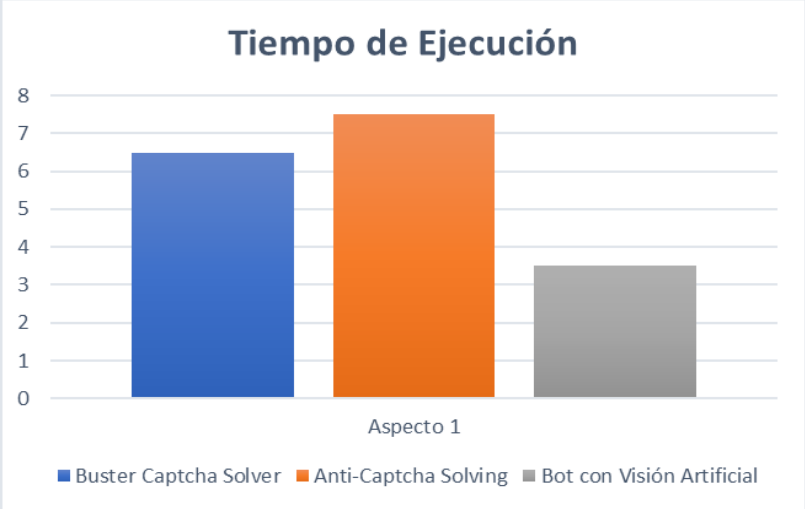
- **Efectividad:** Este criterio está asociado al criterio del tiempo, puesto que a medida que se ejecuta un proceso es importante que se ejecute con la menor cantidad posible de errores y mucho mejor si se ejecuta la tarea al primer intento. Por ello se dará una calificación de ALTO cuando la solución presentada ejecute la tarea en el primer intento y sin errores, MEDIO cuando la solución presentada requiera dos intentos para la ejecución de una tarea y arroje errores en su primer intento y BAJO cuando la solución requiera más de tres intentos de ejecución y posea múltiples errores al ejecutar la tarea.
- **Solución para la automatización:** Las soluciones presentadas tienen como objetivo la automatización de procesos repetitivos, por ende la herramienta seleccionada debe ofrecer una solución automatizada de software, para reducir el margen de error que pueden tener los humanos. Por ello se dará una calificación de ALTO a aquella herramienta que solucione efectivamente el sistema CAPTCHA de manera automatizada, MEDIO aquella herramienta que requiera de dos o más intentos para aprobar el reto del CAPTCHA y BAJO a aquella herramienta que no solucione el sistema CAPTCHA o que implemente otros medios no automatizados para resolver.
- **Compatibilidad con navegadores:** Como los sistemas CAPTCHAs son implementados en sitios web, la solución presentada debe ser compatible con la mayoría de los navegadores web existentes. Por ello, se proporciona una calificación de ALTO a aquella solución que es compatible con más de tres navegadores web, MEDIO aquella solución que es compatible con al menos dos navegadores web y BAJO aquella solución que sea aplicable en un solo navegador web.

Con este análisis, se fundamentan los parámetros de evaluación. En el primer parámetro correspondiente al “Tiempo de Ejecución” se obtuvo que la solución “Anti-CAPTCHA Solving”, ejecutó la tarea de análisis del sistema CAPTCHA en un tiempo menor comparado con las soluciones de Buster CAPTCHA Solver y nuestra solución implementada.

Anti-CAPTCHA Solving es más rápido porque implementa humanos reales para resolver los sistemas CAPTCHAs. Por otro lado, la solución basada en Buster CAPTCHA Solver obtuvo una calificación de MEDIO, debido a que en ocasiones tiende a fallar, motivo por el cual puede demorar más. Por último, en nuestra solución que utiliza el “Bot con Visión Artificial” el tiempo de ejecución es lento, puesto que requiere de un tiempo prudencial para analizar las imágenes del CAPTCHA y compararlos con la API de Google Cloud Vision.

Figura 75

Análisis comparativo del tiempo de ejecución de las soluciones



Nota. En la imagen se muestra el análisis comparativo del tiempo de ejecución entre las tres soluciones que permiten resolver sistemas CAPTCHA.

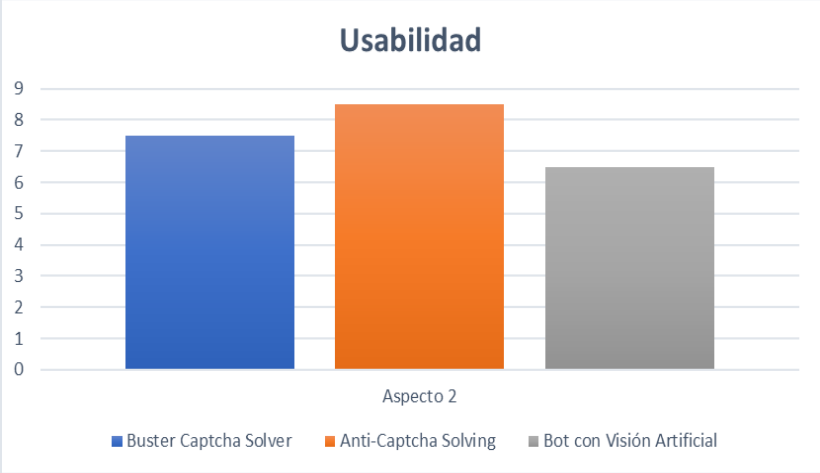
Como siguiente parámetro de evaluación se plantea a la “Usabilidad” en donde se obtuvo como resultado que la herramienta Anti-CAPTCHA Solving y Buster CAPTCHA Solver poseen una calificación de ALTO. Estos resultados se dan por el promedio de calificaciones de

los autores del proyecto, que evaluaron estas dos herramientas y han calificado como una solución sencilla y fácil de implementar. Por otro lado, la solución del Bot con Visión Artificial obtuvo una calificación de MEDIO puesto que requiere abrir el software UiPath para ejecutar la solución lo que implica un consumo extra de recursos de la máquina de donde se ejecuta.

En la Figura 76 se muestra el diagrama de barras con los resultados obtenidos del análisis de este parámetro. Se destaca que ninguna de las herramientas probadas posee una calificación de BAJO puesto que son intuitivas para el usuario y se requiere de una curva de aprendizaje corta.

Figura 76

Análisis comparativo de la usabilidad de las soluciones



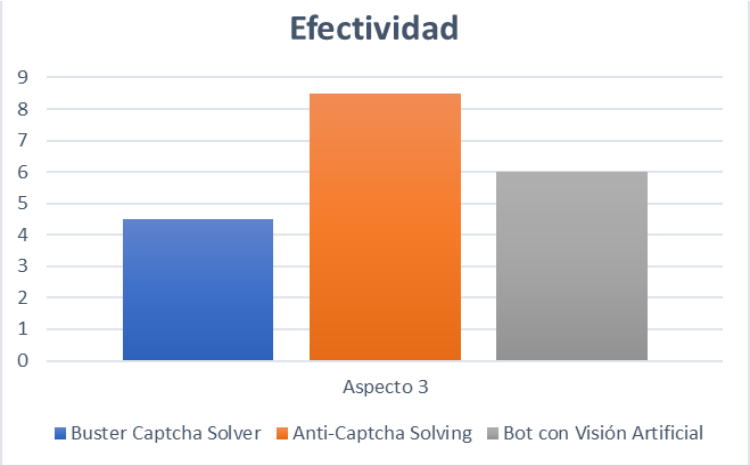
Nota. En la imagen se muestra el análisis comparativo de la usabilidad entre las tres soluciones para resolver sistemas CAPTCHA.

Otro parámetro medible es la efectividad, puesto que a medida que se ejecuta un proceso es importante que este se desempeñe de la mejor manera y con la menor cantidad de errores. Por ello, en la Figura 77 se puede observar que la herramienta que se ejecuta con la menor cantidad de errores es la solución del “Anti-CAPTCHA Solving” que obtuvo una calificación de ALTO, ya que la imagen CAPTCHA es resuelta por humanos. Por otro lado la solución de Buster CAPTCHA Solver obtuvo una calificación de MEDIO ya que en ocasiones

presenta errores de ejecución y no resuelve completamente el desafío del CAPTCHA, siendo necesario ejecutar la solución más de una sola vez. De igual forma nuestra solución “Bot con Visión Artificial” obtuvo una calificación de MEDIO puesto que para el desarrollo del bot se utilizaron selectores que corresponden a los elementos que componen la página web, por ende, en ocasiones estos selectores no son reconocidos por la herramienta RPA y ocasionan errores de ejecución.

Figura 77

Análisis comparativo de la efectividad que presentan las soluciones



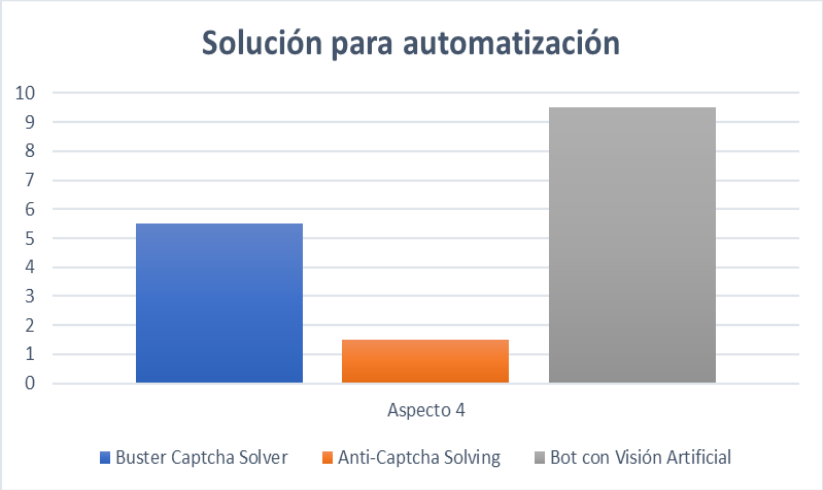
Nota. En la imagen se muestra el análisis comparativo de la efectividad que presentan las tres soluciones al ejecutarse.

Las herramientas evaluadas deben ser capaces de ejecutar el proceso de manera automática y sin intervención humana. En este contexto, en la Figura 78 se visualizan los resultados obtenidos en el análisis de cada solución presentada, calificando su capacidad de automatizar una tarea sin intervención humana. Nuestra solución “Bot con Visión Artificial” obtiene una calificación de ALTO en este criterio, puesto que analiza los sistemas CAPTCHA sin intervención humana y de manera automática, lo que sin duda cumple con la esencia de un bot. Por otro lado, la solución presentada con “Buster CAPTCHA Solver” obtuvo una calificación de MEDIO puesto que implementa una extensión de Google Chrome para resolver el desafío,

pero en ocasiones tiende a cometer errores por lo que requiere que el usuario repita el proceso. Por último, con el análisis de la solución “Anti CAPTCHA Solving” se obtuvo como resultado una calificación de BAJO ya que resuelve los desafíos CAPTCHA con 100% de intervención humana, por lo que no es considerado como una herramienta automatizada sino como un servicio.

Figura 78

Análisis comparativo de las herramientas como solución óptima para automatizar



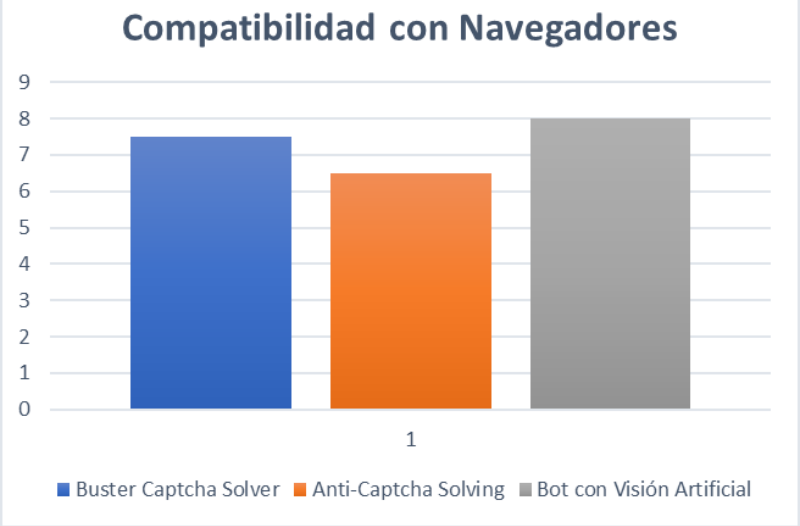
Nota. En la imagen se muestra el análisis comparativo de las herramientas tomando en consideración su manera de automatizar una tarea sin intervención humana.

Como la mayoría de los desafíos CAPTCHA son implementados en los sistemas web, las herramientas de solución deben ser compatibles con la mayoría de navegadores web. En este contexto, en la Figura 79 se muestra una gráfica de barras que expone los resultados del análisis de compatibilidad de las soluciones presentadas con diferentes navegadores web. Nuestro “Bot con Visión Artificial” y el “Buster CAPTCHA Solver” obtuvieron una calificación de ALTO en el análisis cualitativo. Nuestro Bot utiliza la herramienta RPA de UiPath que posee compatibilidad con cinco tipos de navegadores web (Internet Explorer, Edge, Chrome, Mozilla, Safari). Por otro lado, el Buster CAPTCHA Solver es utilizable en navegadores como: Edge,

Chrome y Mozilla. Sin embargo, el servicio de Anti-CAPTCHA Solving obtuvo una calificación de MEDIO, puesto que solo es compatible con Google Chrome y Edge.

Figura 79

Análisis comparativo de las herramientas con los navegadores web

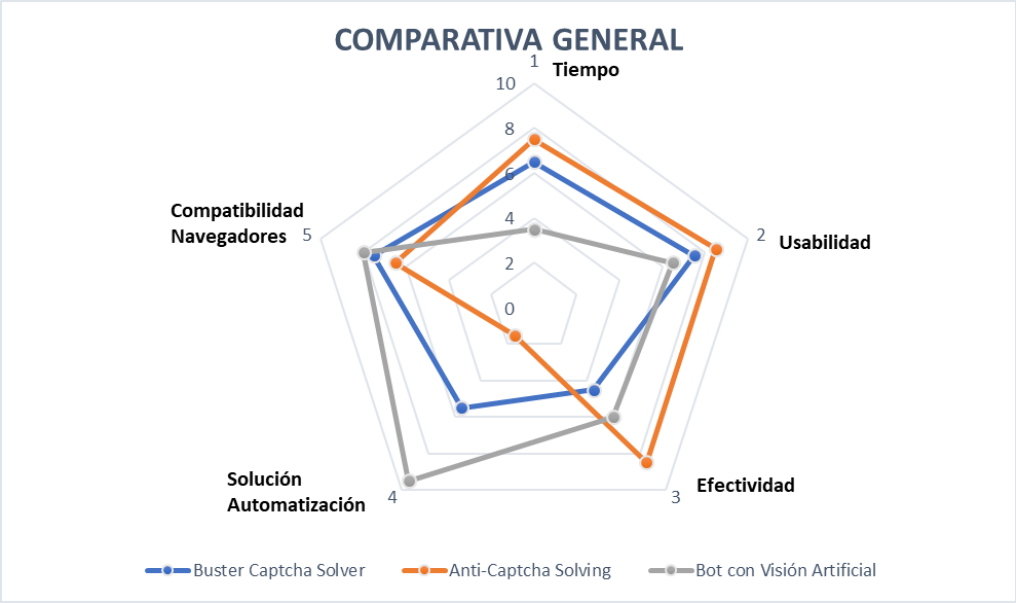


Nota. En la imagen se muestra el análisis comparativo de las herramientas tomando en consideración su compatibilidad con los navegadores web.

A continuación, se establece un análisis general de los resultados obtenidos. En la Figura 80 se observa que cada herramienta posee una o más características que las hacen fuertes en perspectiva con las demás. En la solución del “Buster CAPTCHA Solver” se evidencia que sus características más fuertes son la usabilidad y el tiempo que implementa para resolver el desafío CAPTCHA. La solución “Anti-CAPTCHA Solving” es fuerte en los parámetros de usabilidad, tiempo de ejecución y efectividad, pero posee una gran desventaja, ya que no es considerada como una solución óptima para trabajar como RPA debido a que no automatiza, sino que utiliza humanos para resolver los CAPTCHAs. Nuestra solución “Bot con Visión Artificial” es fuerte en los parámetros de compatibilidad con los navegadores web y se presenta como la solución más efectiva para automatizar la resolución de los desafíos CAPTCHA, ya que implementa tecnologías como la Inteligencia Artificial.

Figura 80

Análisis general de las herramientas para resolver los sistemas CAPTCHA



Nota. En la imagen se muestra el análisis general de las soluciones presentadas, cada una tiene fortalezas y debilidades en los parámetros analizados.

Pruebas de rendimiento del Bot

Para evaluar el correcto funcionamiento del bot, se han realizado pruebas de rendimiento que evalúan la efectividad de resolución del desafío CAPTCHA. Con esto, se puede comprobar si realmente los desafíos CAPTCHA de tipo hCAPTCHA son vulnerables a ataques realizados por bots y se confirma o no su afirmación de seguridad, verificando su funcionamiento en base al análisis utilizando visión artificial. Para esto, se midieron los tiempos de ejecución de nuestro bot durante 10 intentos para demostrar la efectividad que posee el sistema en la resolución del desafío. Se evaluó nuestro bot desarrollado en UiPath y probado en la página web “https://www.map24.com/contact/”, correspondiente a una revista digital. En la Tabla 16, se muestran los resultados obtenidos producto de cada ejecución, considerando el parámetro de tiempo (segundos) como la principal variable a medir.

Tabla 16

Tiempo de ejecución de las pruebas realizadas a la página web "map24"

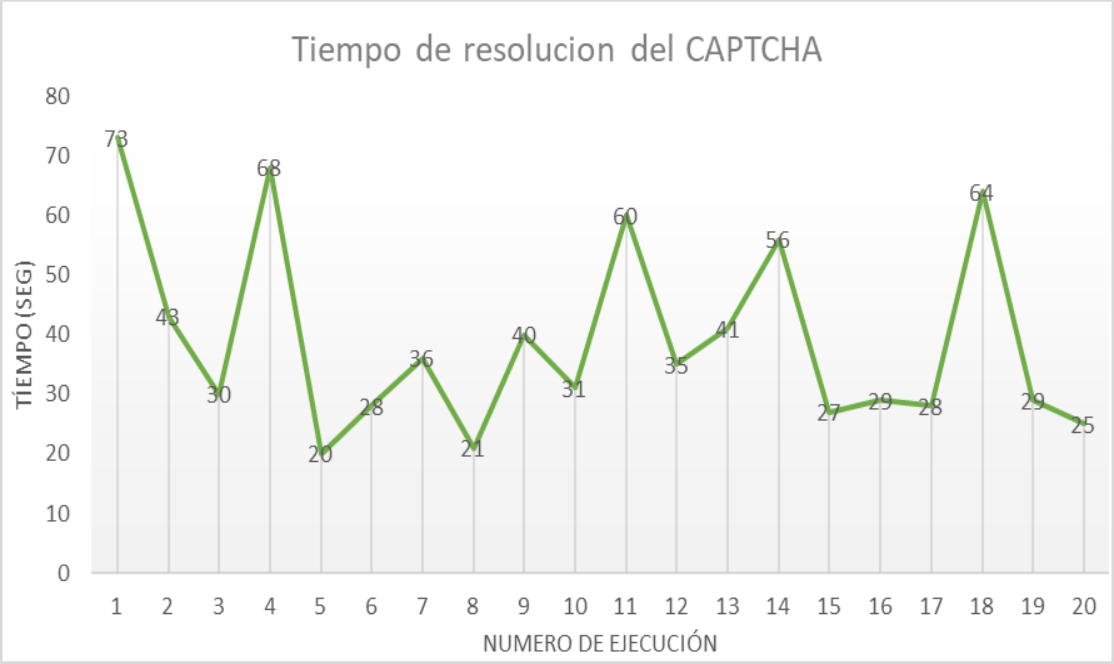
Número Intento	Tiempo de resolución	Tranformación tiempo (segundos)
1	0:01:13	73
2	0:00:43	43
3	0:00:30	30
4	0:01:08	68
5	0:00:20	20
6	0:00:28	28
7	0:00:36	36
8	0:00:21	21
9	0:00:40	40
10	0:00:31	31
11	0:01:00	60
12	0:00:35	35
13	0:00:41	41
14	0:00:56	56
15	0:00:27	27
16	0:00:29	29
17	0:00:28	28
18	0:01:04	64
19	0:00:29	29
20	0:00:25	25

Nota. La tabla muestra los tiempos obtenidos en cada ejecución para resolver el CAPTCHA dentro de la página web *https://www.map24.com/contact/*.

Los datos de la Tabla 16, muestra los resultados del tiempo en segundos que emplea el bot para resolver el desafío de hCAPTCHA en dicha página web, tomando en consideración la ejecución de este proceso por 20 veces. Durante esta ejecución se obtuvieron diferentes valores que variaron durante las pruebas de resolución del CAPTCHA. En la Figura 81 se muestra la gráfica de líneas que detalla los tiempos que tardó el bot en resolver cada desafío a lo largo de las 20 ejecuciones.

Figura 81

Gráfica de líneas del tiempo que emplea el bot para resolver el desafío CAPTCHA



Nota. Para la representación de los resultados que se muestran en la figura, se implementó la gráfica de líneas ya que muestra de mejor manera la tendencia de un dato a lo largo de un tiempo definido.

Estas variaciones de tiempo son resultado de diferentes aspectos que se analizaron durante la ejecución del proceso. A continuación se detallan en resumen:

- En las 3 primeras ejecuciones, se visualiza una tendencia a la baja con respecto al tiempo, esto se debe a que las imágenes presentadas al inicio dentro del CAPTCHA tenían las mismas características (las tres ejecuciones tenían imágenes de animales), lo que implica que el modelo de visión artificial de la API se entrenó correctamente, comprendiendo las imágenes con más rapidez.
- En la 4ta ejecución, el tiempo subió considerablemente, porque se presentaron nuevas imágenes en el CAPTCHA implicando un nuevo análisis de visión artificial. Además, la palabra que toma el bot para el análisis se dividía en dos partes, lo que implicaba que el bot solo obtenga la parte final y no analice correctamente.
- En la quinta, sexta y séptima ejecución, existe una tendencia alta con respecto al tiempo, esto se debe a que el desafío CAPTCHA presentó nuevas imágenes que debían ser analizadas por el modelo de visión artificial.
- En las ejecuciones del 8 al 11, existe una tendencia alta con respecto al tiempo, debido a que aparecieron imágenes que tenían la palabra que toma el bot para el análisis, dividida en dos partes, motivo por el cual el bot solo obtenía la parte final y no analizaba correctamente.
- En la ejecución 12, el tiempo de resolución del CAPTCHA bajó notablemente debido a que el bot logró identificar más rápido las imágenes presentadas.
- En las ejecuciones 13 y 14, existe una tendencia alta con respecto al tiempo, debido a que se presentaron imágenes similares (figuras de perros) con colores idénticos que dificultaron el correcto análisis de las figuras en la API de visión artificial.
- En las ejecuciones 15 a 17 el tiempo es considerablemente bajo, puesto que el modelo de visión artificial que posee la API se entrenó correctamente con las imágenes presentadas anteriormente y pasaba el desafío más rápido.
- En la ejecución 18 hubo un aumento considerable del tiempo de resolución del CAPTCHA, debido a que aparecieron imágenes con características similares y colores

idénticos, dificultando el correcto análisis del modelo de visión artificial que posee la API.

- Por último, en las ejecuciones del 19 y 20 hubo una tendencia a la baja con respecto al tiempo, debido que el CAPTCHA mostró imágenes que ya habían aparecido previamente, por lo que el modelo de visión artificial resolvió más rápido el desafío.

A pesar de que el análisis de tiempo presentó variaciones muy notorias, es importante recalcar que ninguna ejecución superó el tiempo que se obtuvo en la primera ejecución del bot, puesto que el modelo de visión artificial al analizar una imagen que ya apareció previamente, se entrenaba y analizaba mucho más rápido.

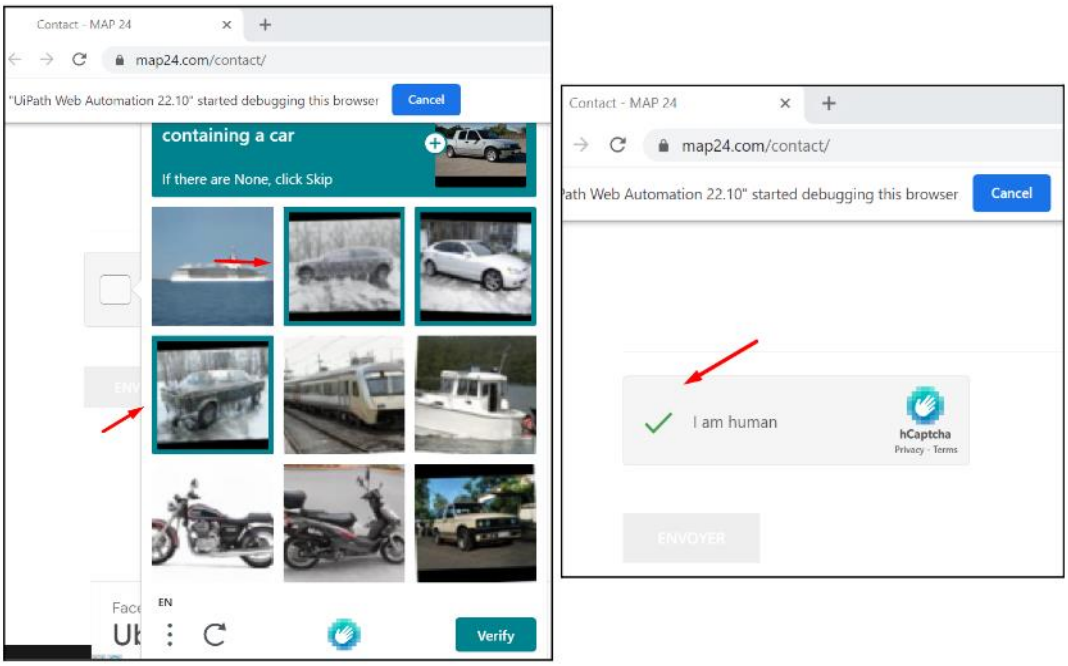
Pruebas y análisis de vulnerabilidades a desafíos CAPTCHA de páginas web

Luego de evaluar las pruebas de rendimiento de nuestro bot, es importante analizar su funcionamiento sobre diferentes sistemas CAPTCHA. Una de las ventajas que ofrece UiPath, es que permite el desarrollo de nuevos proyectos a partir del código de proyectos previamente creados, con la facilidad de exportarlos como plantilla. Con ello se puede reutilizar el código y aplicarlo a nuevos proyectos.

Se realizaron las pruebas de vulneración en los sistemas CAPTCHAs de la página "*https://www.map24.com/contact/*", en donde se obtuvo como resultado que las imágenes que muestra el hCAPTCHA que no poseen texturas sobre el objeto, son más propensas a ser vulneradas. Esto se debe a que la API de Google Cloud Vision analiza cada imagen estableciendo porcentajes de similitud y cada figura es comparada con la gran colección de imágenes indexadas desde la web, estableciendo etiquetas para determinar si concuerda con el texto e imagen requerida en el desafío CAPTCHA. Si la imagen se distorsiona o posee texturas que dificultan el análisis, el bot no automatiza la tarea. Cabe mencionar que el tiempo que demoró el bot para pasar este desafío CAPTCHA fue de 24 segundos.

Figura 82

Prueba de vulneración del hCAPTCHA en la página "map24.com"



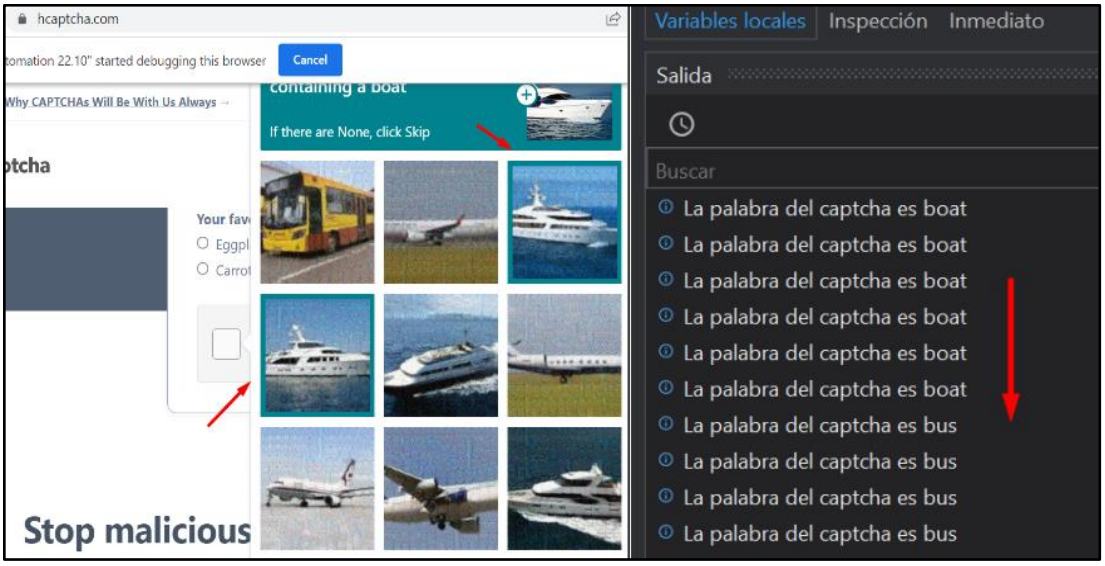
Nota. Esta imagen representa el proceso en donde el bot reconoce y selecciona las imágenes para resolver el hCAPTCHA.

Como segunda prueba se ejecutó el bot en la página web "https://www.hCAPTCHA.com/", correspondiente a la página principal de hCAPTCHA. Durante nuestra investigación se pudo comprobar que esta página web en sus inicios presentaba imágenes en los desafíos CAPTCHAs sin distorsiones y/o alteraciones, lo que facilitaba que sea vulnerada por nuestro bot. Con la última actualización implementada en la página el 23 de julio del 2022, los sistemas hCAPTCHAs presentan texturas y distorsiones en las imágenes que dificultan el análisis y la automatización del proceso.

Por otro lado, como se observa en la Figura 83, el tiempo de ejecución y resolución del sistema hCAPTCHA es indeterminado, puesto que el bot al analizar y no identificar correctamente las imágenes se encuentra en un bucle de tipo While (configurado en el código) el cual mientras no obtenga la imagen correcta no puede pasar el desafio.

Figura 83

Prueba de vulneración del hCAPTCHA en la página "hCAPTCHA.com"



Nota. Esta imagen representa el proceso de reconocimiento y selección de las imágenes por parte de nuestro bot, sin embargo, se observa que no es capaz de resolver el sistema hCAPTCHA.

Para entender mejor, en la Tabla 16 se muestra el estudio detallado de las dos páginas web analizadas, en donde se proporcionan los siguientes criterios:

- **Detección de las imágenes que proporciona el CAPTCHA:** Corresponde al análisis de funcionamiento de la API. En este criterio se determina si el bot analiza cualquier tipo de imágenes o tiene limitaciones con imágenes que poseen diferentes texturas.
- **Resolución del hCAPTCHA:** Corresponde al análisis del funcionamiento del bot para resolver el desafío CAPTCHA. En este criterio se determina si la página web es vulnerada o no, en base a las imágenes que proporciona el sistema hCAPTCHA.
- **Tipo de imágenes:** En este criterio se visualizan las imágenes que son analizadas por el Bot de resolución de CAPTCHA con visión artificial.

- **Tiempo promedio de ejecución:** Se visualiza el tiempo que se demora el bot para pasar el desafío de hCAPTCHA. En el caso de no pasar la prueba se coloca “No determinado”, puesto que el bot sigue en ejecución hasta que se satura la RAM del equipo (bucle infinito).

Tabla 16

Análisis comparativo de la resolución del sistema hCAPTCHA en las páginas web

Criterios de evaluación	map24.com	hCAPTCHA.com
Detección de las imágenes que proporciona el CAPTCHA	Si	Si
Resolución del hCAPTCHA	Si	No
Tipos de imágenes	Posee imágenes más nítidas que facilitan el reconocimiento del objeto.	Posee imágenes con texturas que dificultan el reconocimiento del objeto.
Promedio de tiempo de ejecución	24 segundos	No determinado

Nota. En la tabla se muestran los resultados obtenidos al analizar las dos páginas web mencionadas, el detalle de su análisis se explicará en los siguientes apartados.

Como se observa en la Tabla 16, los criterios de análisis son muy generales, sin embargo, se seleccionan los más importantes para evaluar el funcionamiento de nuestro bot. En el primer criterio correspondiente a la “Detección de las imágenes que proporciona el CAPTCHA”, se obtuvo que en las dos páginas web analizadas, el bot si logró detectar las imágenes del hCAPTCHA, con la particularidad que en la página web *hCAPTCHA.com,* las imagenes detectadas poseen colores cálidos y fondos claros con delimitación de contornos más nítidos, sin embargo, también se presentan imágenes con texturas onduladas como fondo.

En el segundo criterio enfocado a la “Resolución del hCAPTCHA”, el bot de la página web *map24.com*, si logro resolver el sistema CAPTCHA, obteniendo como resultado el visto de

comprobación. Por otro lado, el bot de la página web *hCAPTCHA.com*, no logró resolver el CAPTCHA en vista de las texturas que poseen las imágenes.

Por último, al analizar el criterio correspondiente al “Tiempo de ejecución” se obtuvo que nuestro bot resolvió el desafió CAPTCHA en 24 segundo sobre la página web *map24.com* (tiempo muy aceptable para ser un bot), por otro lado, para el bot que se ejecutó sobre la página web *hCAPTCHA.com* si logró identificar las palabras de las imágenes a pesar de las texturas (Figura 81), pero el porcentaje de reconocimiento no permitió que el desafío se resuelva, no se pudo medir el tiempo de resolución, por lo que nuestro bot solo seleccionaba las imágenes más nítidas pero no resolvió el desafío.

Resultados del bot de análisis de ataques de tipo Cross-Site Scripting (XSS)

Con la ejecución de las soluciones mostradas en la metodología, las cuales forman parte del bot de análisis de vulnerabilidades XSS en páginas web por medio del estudio de sus cookies, se obtuvieron los resultados esperados. Estos resultados se dividieron en dos partes tomando como referencia las macros o soluciones desarrolladas en el plan gratuito de UI.Visión.

En primera instancia se obtuvieron los resultados a partir del análisis de las cookies de cada una de las páginas web por separado. La segunda parte consta de la presentación de los resultados obtenidos a partir del análisis de las cookies generadas a partir del historial de navegación de un usuario. En este contexto, a continuación, se detallan los resultados de las dos soluciones implementadas para el análisis de vulnerabilidades XSS.

Resultados obtenidos en el análisis de las cookies por página web

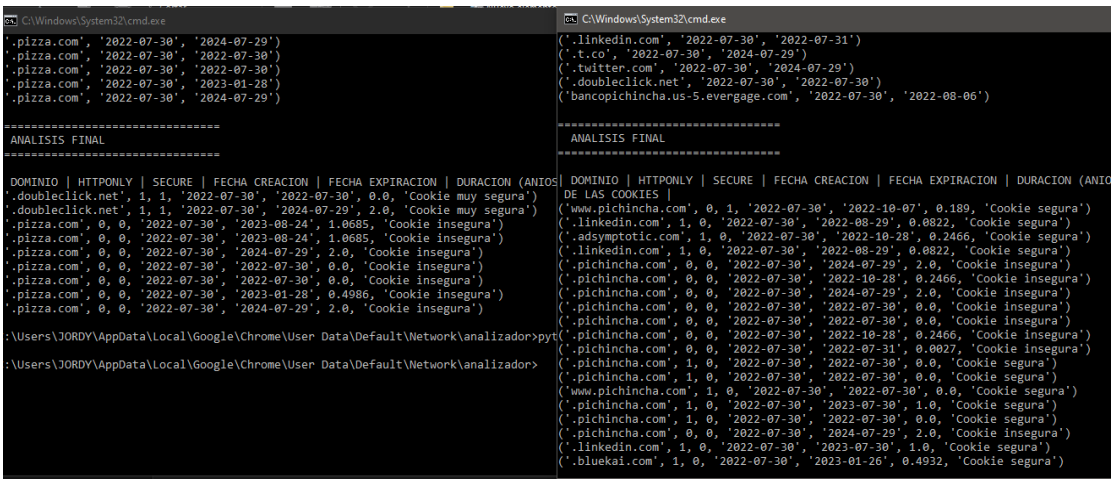
Como parte de la ejecución del bot se realizó la prueba de funcionamiento y análisis en dos páginas web. Estas páginas fueron seleccionadas en base a sus características y tipo de contenido, por lo cual se establecieron en el archivo CSV las siguientes páginas:

- **http://pizza.com**, esta página web representa a un dominio que no se encuentra protegido ya que utiliza el protocolo HTTP. Es por esto que se la puede considerar como una página vulnerable, considerando que sus cookies también pueden ser inseguras.
- **https://pichincha.com**, por otro lado, tenemos una página web que representa al sector bancario, debido a los procesos que maneja se la puede considerar segura ya que utiliza el protocolo HTTPs. En base a esto se puede esperar que la mayoría de sus cookies sean seguras.

Partiendo de esta definición de las páginas web objetivo, el siguiente paso fue ejecutar nuestro bot desarrollado en UI.Visión. El primer resultado que se visualiza son las ventanas del CMD, en donde se detallan las consultas SQL implementadas. Como se visualiza en la Figura 84, estas ventanas nos sirven para obtener más detalles de la información generada por las cookies.

Figura 84

Visualización de los resultados de análisis por cada página web

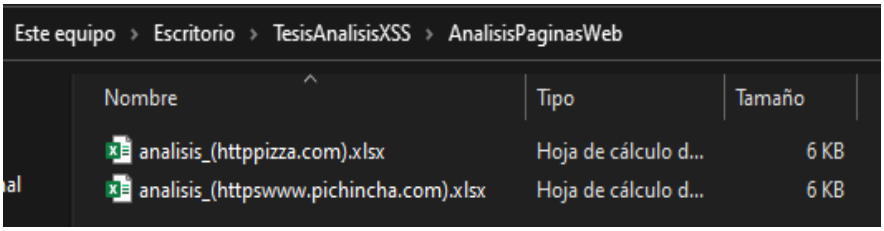


Nota. Estas pantallas pueden ser omitidas como parte de los resultados, ya que a mayor número de páginas analizadas, el número de estas ventanas aumenta.

La siguiente parte dentro de la ejecución del bot es la obtención de los resultados en formato .xlsx, en la Figura 85 se observan los archivos Excel que se generaron a partir del análisis de las páginas web seleccionadas.

Figura 85

Archivos Excel que contienen los resultados del análisis de las cookies generadas



Nota. Se genera un Excel por cada página web analizada.

El objetivo de exportar los resultados a estos archivos en formato Excel es que las personas que usen el bot puedan visualizar de mejor manera la información de cada una de las cookies. Otra ventaja que esto trae es que a partir de estos archivos se pueden implementar análisis adicionales por medio de las herramientas que nos ofrece Excel, logrando de esta manera generar reportes que nos ayuden a determinar la seguridad de una página web.

Análisis de las cookies del dominio http://pizza.com. En la Figura 86, se muestra la hoja de cálculo obtenida a partir del análisis de las cookies mediante la ejecución de los scripts en python y el uso del bot en UI.Visión. En este apartado se visualizan los datos extraídos y transformados a través de las consultas SQLite, en donde se encuentra la información sobre las etiquetas de seguridad (is_secure y is_httponly), fecha de creación, fecha de expiración, número de años de duración y el análisis final que nos ayuda a determinar si una cookie es segura o no.

Figura 86

Resultado del análisis de las cookies del dominio http://pizza.com

	A	B	C	D	E	F	G
1	DominioCookie	Segura	HttpOnly	FechaCreacion	FechaExpiracion	NumeroAños	AnalisisCookies
2	.doubleclick.net	1	1	2022-07-30	2022-07-30	0	Cookie muy segura
3	.doubleclick.net	1	1	2022-07-30	2024-07-29	2	Cookie muy segura
4	.pizza.com	0	0	2022-07-30	2023-08-24	1.0685	Cookie insegura
5	.pizza.com	0	0	2022-07-30	2023-08-24	1.0685	Cookie insegura
6	.pizza.com	0	0	2022-07-30	2024-07-29	2	Cookie insegura
7	.pizza.com	0	0	2022-07-30	2022-07-30	0	Cookie insegura
8	.pizza.com	0	0	2022-07-30	2022-07-30	0	Cookie insegura
9	.pizza.com	0	0	2022-07-30	2023-01-28	0.4986	Cookie insegura
10	.pizza.com	0	0	2022-07-30	2024-07-29	2	Cookie insegura

Nota. Esta figura representa la tabla de resultados obtenida a partir de la ejecución del script “cookiesAnalisis.py”.

De acuerdo a la tabla de resultados el dominio “http://pizza.com” generó 9 cookies en total. A partir de este total, se estableció el gráfico que se muestra en la Figura 87, en el cual se obtuvo que de las 9 cookies generadas: solo 2 son “Cookies muy seguras” lo que representa el 22%; y 7 son “Cookies inseguras” que representa el 78%. Estos datos nos indican que el dominio “http://pizza.com” y sus cookies son vulnerables a ataques XSS.

Figura 87

Análisis del total de las cookies generadas por http://pizza.com

Nota. La clasificación de las cookies se obtiene en base a las condiciones establecidas en el script “cookiesAnalisis.py”.

Análisis de las cookies del dominio https://pichincha.com. Como resultado de esta página web se obtuvo un archivo Excel con la misma estructura de la tabla de análisis de cookies. En la Figura 88 se muestra la hoja de cálculo correspondiente a la información de cookies del dominio “https://pichincha.com”, en donde se observa que se generaron un mayor número de cookies, esto está relacionado con la cantidad de servicios que utiliza y ofrece esta página web bancaria.

Figura 88

Resultado del análisis de las cookies del sitio web https://pichincha.com

	A	B	C	D	E	F	G
1	DominioCookie	Segura	HttpOnly	FechaCreacion	FechaExpiracion	NumeroAños	AnalisisCookies
2	www.pichincha.com	0	1	2022-07-30	2022-10-07	0.189	Cookie segura
3	.linkedin.com	1	0	2022-07-30	2022-08-29	0.0822	Cookie segura
4	.adsymptotic.com	1	0	2022-07-30	2022-10-28	0.2466	Cookie segura
5	.linkedin.com	1	0	2022-07-30	2022-08-29	0.0822	Cookie segura
6	.pichincha.com	0	0	2022-07-30	2024-07-29	2	Cookie insegura
7	.pichincha.com	0	0	2022-07-30	2022-10-28	0.2466	Cookie insegura
8	.pichincha.com	0	0	2022-07-30	2024-07-29	2	Cookie insegura
9	.pichincha.com	0	0	2022-07-30	2022-07-30	0	Cookie insegura
10	.pichincha.com	0	0	2022-07-30	2022-07-30	0	Cookie insegura
11	.pichincha.com	0	0	2022-07-30	2022-10-28	0.2466	Cookie insegura
12	.pichincha.com	0	0	2022-07-30	2022-07-31	0.0027	Cookie insegura
13	.pichincha.com	1	0	2022-07-30	2022-07-30	0	Cookie segura
14	.pichincha.com	1	0	2022-07-30	2022-07-30	0	Cookie segura
15	www.pichincha.com	1	0	2022-07-30	2022-07-30	0	Cookie segura
16	.pichincha.com	1	0	2022-07-30	2023-07-30	1	Cookie segura
17	.pichincha.com	1	0	2022-07-30	2022-07-30	0	Cookie segura
18	.pichincha.com	0	0	2022-07-30	2024-07-29	2	Cookie insegura
19	.linkedin.com	1	0	2022-07-30	2023-07-30	1	Cookie segura
20	.bluekai.com	1	0	2022-07-30	2023-01-26	0.4932	Cookie segura
21	.bluekai.com	1	0	2022-07-30	2023-01-26	0.4932	Cookie segura
22	.bluekai.com	1	0	2022-07-30	2023-01-26	0.4932	Cookie segura
23	.www.linkedin.com	1	1	2022-07-30	2023-07-30	1	Cookie muy segura
24	.facebook.com	1	1	2022-07-30	2022-10-28	0.2466	Cookie muy segura
25	.linkedin.com	1	0	2022-07-30	2022-10-28	0.2466	Cookie segura

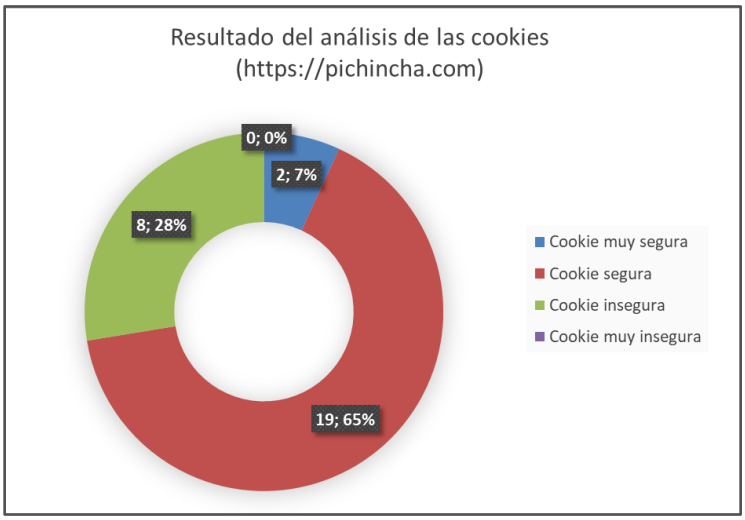
Nota. Esta figura representa la tabla de resultados obtenida a partir de la ejecución del script “cookiesAnalisis.py” para el dominio pichincha.com.

Para la tabla de resultados del sitio “https://pichincha.com” se obtuvo información de un total de 29 cookies. En la Figura 89, se observa que de las 29 cookies generadas: 2 son “Cookies muy seguras” lo que representa el 7%; 19 son “Cookies seguras” lo que representa el 65%; y 8 son “Cookies inseguras” que representa el 28%. Con el análisis de estos datos se

determina que el dominio “https://pichincha.com” y sus cookies, tanto propias como de terceros son en su mayoría cookies seguras ante ataques XSS.

Figura 89

Análisis del total de las cookies generadas por https://pichincha.com



Nota. Esta figura representa la tabla de resultados obtenida a partir de la ejecución del script “cookiesAnalisis.py”.

Tiempo de ejecución del bot por cada dominio analizado. Otro factor que se analizó fue el tiempo de ejecución que el bot tarda para analizar cada una de las páginas web. A continuación, se observa los tiempos aproximados obtenidos a partir de este análisis.

Tabla 17

Tiempos aproximados de ejecución por cada dominio analizado

Número de páginas web	Tiempo aproximado
1 página web	38 seg.
2 páginas web	76 seg.
10 páginas web	380 seg.
100 páginas web	3800 seg.

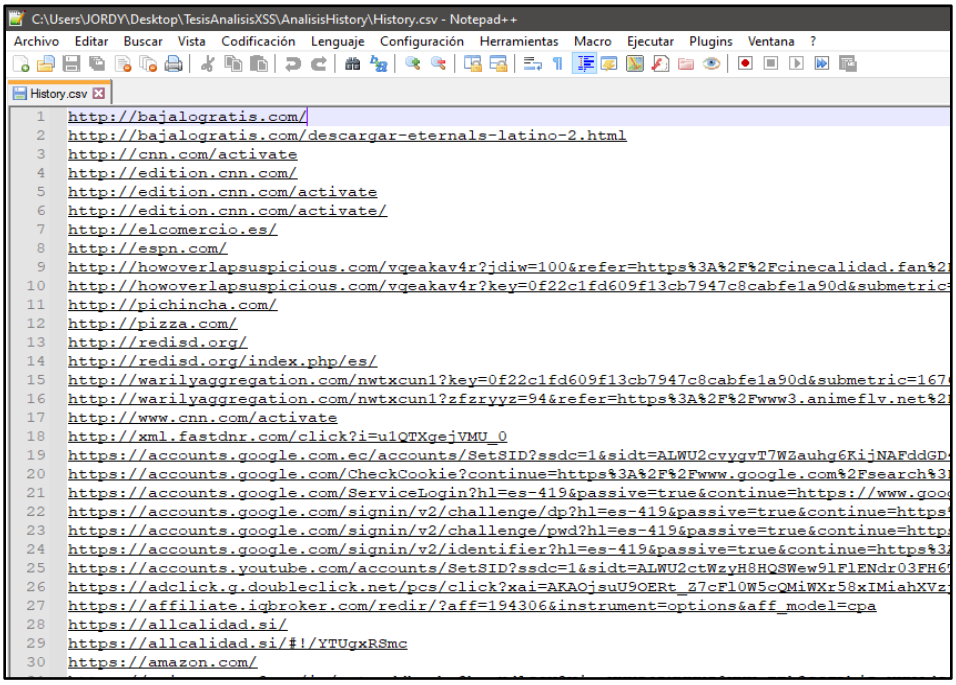
Nota. La proyección del tiempo se realizó en base al tiempo de ejecución de los dominios o páginas web.

Resultados de la solución para el análisis de las cookies del historial de navegación

En esta sección se presentan los resultados obtenidos a partir de la ejecución de la macro (bot) para el análisis de las cookies generadas a partir del historial de páginas web visitadas por un usuario. De acuerdo a la metodología propuesta, el primer paso que realizó el bot fue la ejecución del script “exportarHistory.py”, para extraer las URLs de las páginas web e importarlas al archivo CSV. En la Figura 90, se observa el contenido que consta de un total de 2019 URLs, las cuales corresponden a 85 dominios principales.

Figura 90

Archivo CSV con las URLs extraídas del historial de navegación de un usuario



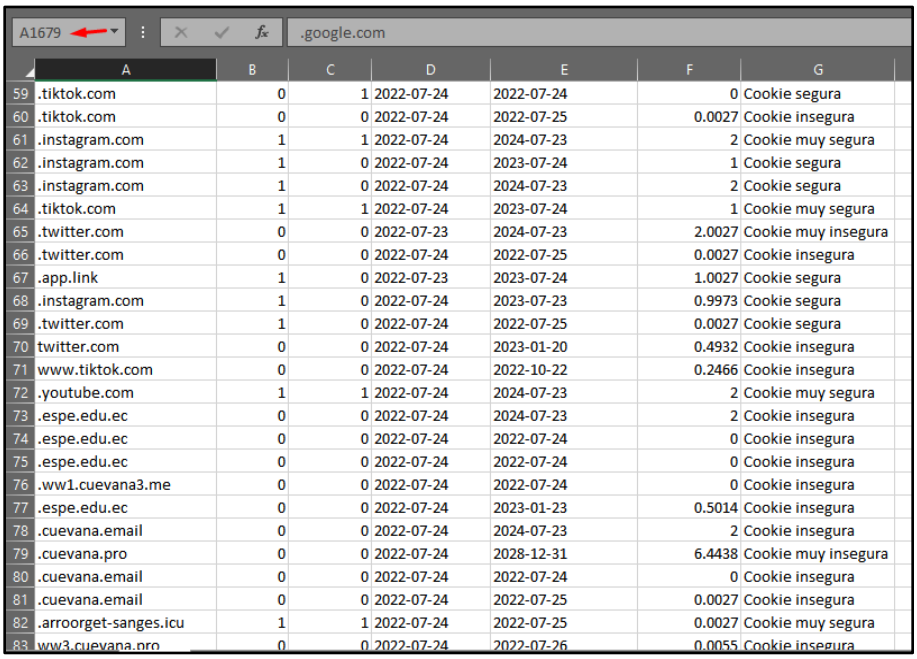
Nota. La figura representa todas las URLs a las que el usuario accedió desde el navegador Chrome, teniendo en cuenta que algunas comparten el mismo dominio.

La obtención de este archivo es esencial, ya que es la base para que el bot visite cada uno de estos enlaces, con el fin de generar las cookies relacionadas con estos dominios. Una vez abiertos estos enlaces en el navegador Chrome el bot ejecutó el script “cookiesAnalisisHistory.py que corresponde a la segunda parte de su funcionamiento.

La ejecución de este script arrojó como resultado un archivo Excel que se muestra en la Figura 91, que contiene los resultados del análisis de todas las cookies del historial de navegación.

Figura 91

Excel con el resultado del análisis del historial de navegación del usuario



A1679 | .google.com

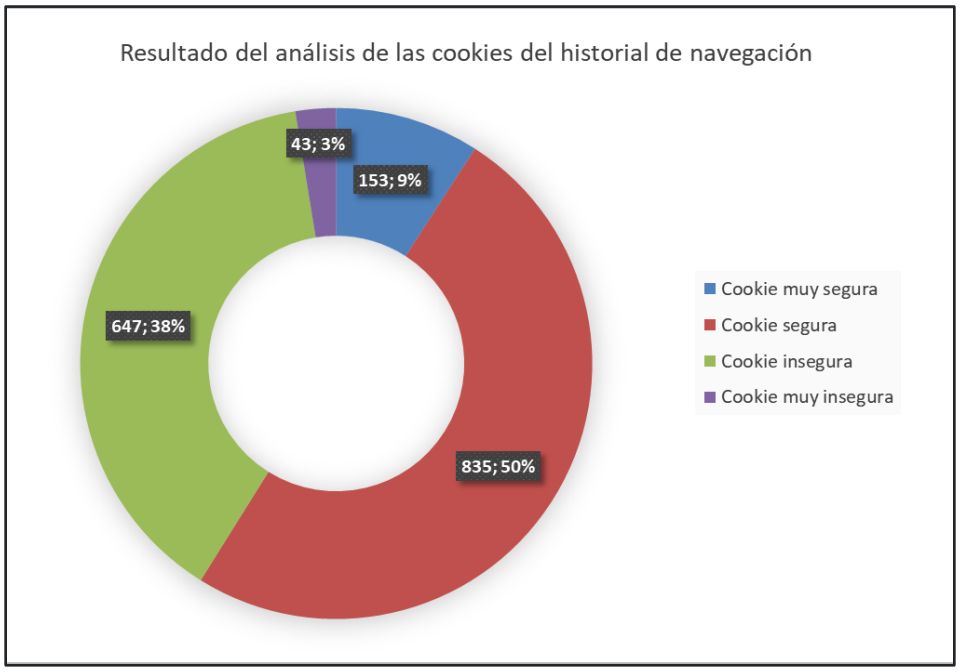
	A	B	C	D	E	F	G
59	.tiktok.com	0	1	2022-07-24	2022-07-24	0	Cookie segura
60	.tiktok.com	0	0	2022-07-24	2022-07-25	0.0027	Cookie insegura
61	.instagram.com	1	1	2022-07-24	2024-07-23	2	Cookie muy segura
62	.instagram.com	1	0	2022-07-24	2023-07-24	1	Cookie segura
63	.instagram.com	1	0	2022-07-24	2024-07-23	2	Cookie segura
64	.tiktok.com	1	1	2022-07-24	2023-07-24	1	Cookie muy segura
65	.twitter.com	0	0	2022-07-23	2024-07-23	2.0027	Cookie muy insegura
66	.twitter.com	0	0	2022-07-24	2022-07-25	0.0027	Cookie insegura
67	.app.link	1	0	2022-07-23	2023-07-24	1.0027	Cookie segura
68	.instagram.com	1	0	2022-07-24	2023-07-23	0.9973	Cookie segura
69	.twitter.com	1	0	2022-07-24	2022-07-25	0.0027	Cookie segura
70	twitter.com	0	0	2022-07-24	2023-01-20	0.4932	Cookie insegura
71	www.tiktok.com	0	0	2022-07-24	2022-10-22	0.2466	Cookie insegura
72	.youtube.com	1	1	2022-07-24	2024-07-23	2	Cookie muy segura
73	.espe.edu.ec	0	0	2022-07-24	2024-07-23	2	Cookie insegura
74	.espe.edu.ec	0	0	2022-07-24	2022-07-24	0	Cookie insegura
75	.espe.edu.ec	0	0	2022-07-24	2022-07-24	0	Cookie insegura
76	.ww1.cuevana3.me	0	0	2022-07-24	2022-07-24	0	Cookie insegura
77	.espe.edu.ec	0	0	2022-07-24	2023-01-23	0.5014	Cookie insegura
78	.cuevana.email	0	0	2022-07-24	2024-07-23	2	Cookie insegura
79	.cuevana.pro	0	0	2022-07-24	2028-12-31	6.4438	Cookie muy insegura
80	.cuevana.email	0	0	2022-07-24	2022-07-24	0	Cookie insegura
81	.cuevana.email	0	0	2022-07-24	2022-07-25	0.0027	Cookie insegura
82	.arroorget-sanges.icu	1	1	2022-07-24	2022-07-25	0.0027	Cookie muy segura
83	ww3.cuevana.pro	0	0	2022-07-24	2022-07-26	0.0055	Cookie insegura

Nota. Este archivo contiene la información de todas las cookies, en donde se evidencia que, del historial analizado para este caso de estudio, se generaron un total de 1678 cookies de los 87 dominios visitados.

Los campos presentes en la hoja de cálculo sirvieron como base de análisis para determinar qué cookies eran sospechosas y que tan vulnerable es un usuario a ataques XSS en base a las cookies almacenadas en su PC. El análisis general mostrado en la Figura 92 muestra que de las 1678 cookies generadas: 153 son “Cookies muy seguras” lo que representa el 9%; 835 son “Cookies seguras” lo que representa el 50%; 647 son “Cookies inseguras” lo que representa el 38%; y 43 son “Cookies muy inseguras” lo que representa del 3% del total de las cookies.

Figura 92

Análisis del total de las cookies generadas por el historial de navegación

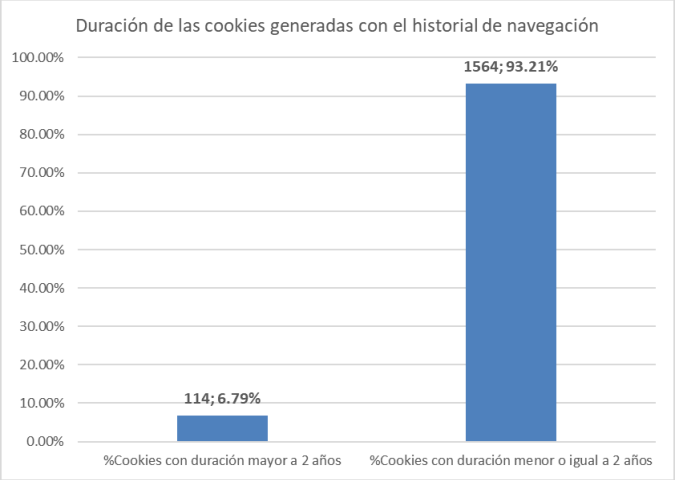


Nota. La figura representa la clasificación de las cookies a partir de su análisis, en donde se observa que existe una tendencia igualada en lo que respecta a cookies seguras e inseguras.

Una de las referencias para establecer si una cookie es sospechosa fue determinar la duración de estas, a partir de la fecha de creación y fecha de caducidad. El análisis de las cookies en la Figura 93 muestra que de los sitios web visitados un 6.79% de las cookies generadas tuvieron una duración mayor a 2 años, mientras que el 93.21% tiene una duración menor o igual al tiempo permitido. Esta información permitió establecer que el usuario posee cookies confiables en lo que respecta a los años de duración ya que la media se encuentra por debajo de los 2 años de duración.

Figura 93

Análisis de las cookies basadas en su tiempo de caducidad

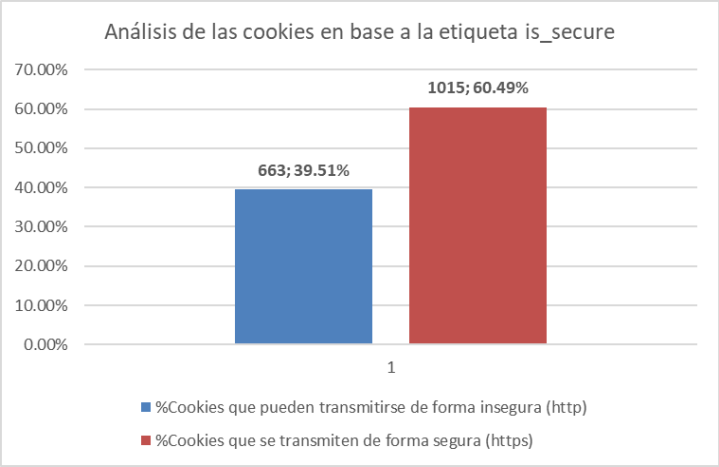


Nota. Esta figura representa el porcentaje de cookies clasificadas mediante su duración.

Otro campo que se analizó para la obtención de los resultados fue la etiqueta o bandera *is_secure* . De acuerdo a la Figura 94, se observa que, de las 1678 cookies, el 39.51% permiten la transmisión de los datos de manera no segura ya que se usa el protocolo HTTP, mientras que el 60.49% usa el protocolo HTTPs considerado como un protocolo seguro.

Figura 94

Análisis de las cookies en base a la bandera is_secure

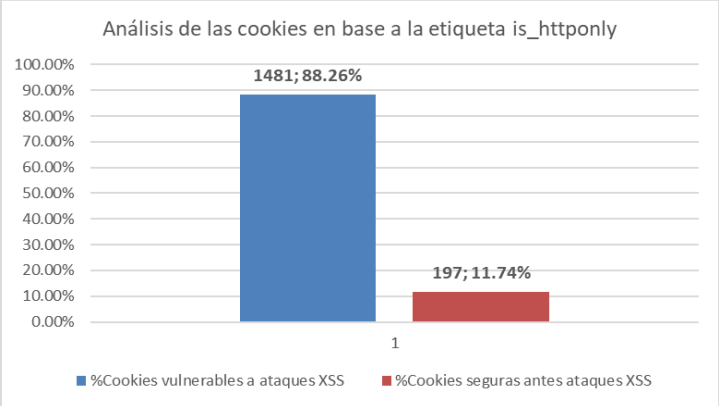


Nota. La barra azul representa *is_secure=0,* y la barra roja *is_secure=1.*

Por último, la bandera *is_httponly* también fue considerada para el análisis de las cookies. Los resultados obtenidos en la Figura 95 muestra que el 88.26 % de las cookies son vulnerables a ataques XSS, y solo el 11.74% de las cookies se crearon con la etiqueta httponly=1.

Figura 95

Análisis de las cookies en base a la bandera is_httponly



Nota. La barra azul representa *is_httponly=0,* y la barra roja *is_httponly=1.*

Tiempo de ejecución del bot para análisis del historial. Se evaluó el tiempo de ejecución que el bot tarda en analizar un determinado número de URLs en base al historial de navegación de un usuario. En la Tabla 18 se observa los tiempos aproximados obtenidos.

Tabla 18

Tiempos aproximados de ejecución del bot de análisis del historial

Número de páginas web	Tiempo aproximado
Historial con 1 URL	27 seg.
Historial con 10 URLs	108 seg. (1.8 min)
Historial con 100 URLs	918 seg. (15.3 min)
Historial con 1000 URLs	9018 seg. (150.3 min)

Nota. La proyección del tiempo se realizó en base al tiempo de ejecución que tarda en analizar cada uno de los URLs del historial de navegación.

Comparativa de las herramientas RPA UiPath y Ui.Vision

Una vez concluido con los procesos de automatización en las dos herramientas, correspondientes a la resolución de los sistemas CAPTCHA y el análisis de ataques XSS en páginas web, se plantea el siguiente análisis cualitativo. Este determinó las características que más destacan en cada una de las soluciones. De igual forma se analizaron los puntos fuertes que tiene cada herramienta y sus mejores habilidades en cada criterio de análisis presentes al momento de realizar una automatización.

En la Tabla 19, se muestran los resultados obtenidos del análisis cualitativo de las dos herramientas, con un rango de calificación del 1 al 10 en base a la utilización y experiencia que obtuvo cada autor del proyecto.

Tabla 19

Análisis comparativo de las herramientas RPA utilizadas en el proyecto

Aspecto a calificar	UiPath				UI.Vision			
	U1	U2	Prom.	Cali. A	U1	U2	Prom.	Cali. A
Tiempo de ejecución	7	6	6,5	**MEDIO**	9	9	9	**ALTO**
Usabilidad	7	6	6,5	**MEDIO**	9	8	8,5	**ALTO**
Efectividad	8	9	8,5	**ALTO**	7	6	6,5	**MEDIO**
Solución para automatización	9	9	9	**ALTO**	6	5	5,5	**MEDIO**
Compatibilidad con navegadores	8	8	8	**ALTO**	6	7	6,5	**MEDIO**
	TOTAL		**7,7**		**TOTAL**		**7,2**	

Nota. Cada uno de los valores establecidos en la tabla serán detallados en los siguientes apartados.

Para determinar el criterio de calificación cualitativo se utilizó un rango de calificación, con valores establecidos entre 1 al 3.99 corresponde a un criterio BAJO, de 4 a 6.99 corresponde a un criterio MEDIO, mientras que los valores establecidos entre 7 a 10 corresponde a un criterio de ALTO. La determinación del criterio cualitativo se define en base al promedio obtenido de la calificación de los dos autores en cada parámetro.

Uno de los parámetros que define una herramienta RPA al momento de implementar una solución de automatización es el “Tiempo de Ejecución”. Partiendo de este criterio se obtuvieron los resultados mostrados en la Figura 96, en donde se observa que UiPath obtuvo un promedio de 6.5 correspondiente a una calificación de MEDIO, ya que para ejecutar una solución necesita compilar el código y requiere de más recursos de la PC. En lo que respecta a UI.Vision, obtuvo una calificación ALTA ya que consume menos recursos de la máquina, debido a que utiliza una extensión del navegador y su funcionamiento se basa en macros implementadas en formato JSON.

Figura 96

Análisis comparativo del tiempo de ejecución entre UiPath vs UI.Vision

Nota. En la imagen se muestra el análisis comparativo del tiempo de ejecución que requieren las herramientas para ejecutar una solución.

El siguiente parámetro evaluado fue la “Usabilidad” o conocida como experiencia de usuario, en base a las calificaciones se obtuvieron los promedios mostrados en la Figura 97. Se determina que UiPath cuenta con una calificación de MEDIO en comparación con UI.Vision que obtuvo una calificación de ALTO. El motivo de estos puntos de vista se debe a que la curva de aprendizaje de UiPath es más alta en comparación a UI.Vision y requiere de ciertos conocimientos en programación para utilizar todas las opciones que brinda esta herramienta, mientras que por otro lado UI.Visión posee una interfaz más sencilla y su funcionamiento se basa en pequeños pasos o comandos que se encuentran en su página oficial.

Figura 97

Análisis comparativo de la usabilidad entre UiPath vs UI.Vision

Nota. En la imagen se muestra el análisis comparativo de la usabilidad entre las dos herramientas utilizadas en este proyecto.

Otro parámetro analizado fue la “Efectividad”, puesto que en una automatización de procesos se requiere de un alto grado de eficacia al ejecutar una tarea repetitiva. Por ello, en la Figura 98 se observa que la herramienta que se ejecuta con la menor cantidad de errores es UiPath, que obtuvo una calificación de ALTO, esto se debe a la precisión y compilación que realiza en cada uno de sus proyectos. Para la herramienta UI.Vision, se obtuvo una calificación

de MEDIO, puesto que en ocasiones presenta problemas al trabajar con elementos visuales y suele generar inconvenientes en su ejecución.

Figura 98

Análisis comparativo de la efectividad entre UiPath vs UI.Vision

Nota. En la imagen se muestra el análisis comparativo de la efectividad que presentan las herramientas al ejecutar un bot o solución.

El siguiente parámetro evaluado fue la “Solución para la automatización”, que determina las facilidades y opciones que nos ofrece una herramienta para implementar una solución para determinados problemas de automatización. Se obtuvieron los resultados presentados en la Figura 99. UiPath presenta una calificación de ALTO, ya que esta herramienta ofrece varias opciones que se pueden utilizar para diseñar bots más complejos.

Por otro lado, la herramienta UI.Visión obtuvo una calificación de MEDIO, ya que su funcionamiento se basa en una cantidad determinada de comandos, que a pesar de que es una herramienta gratuita posee ciertos comandos que requieren de pagos extras para utilizarlos de manera ilimitada.

Figura 99

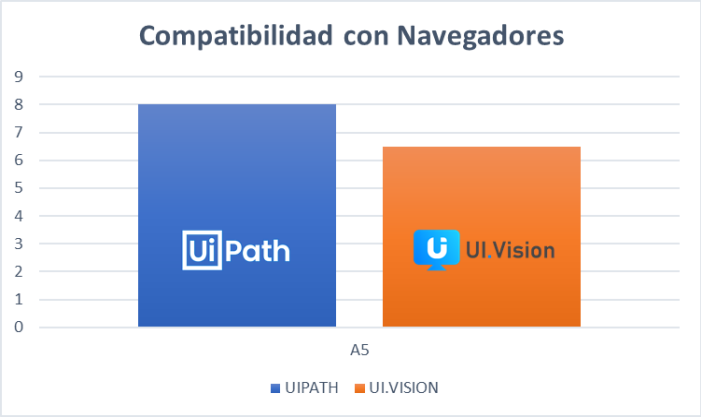
Análisis comparativo de la solución óptima para automatizar

Nota. En la imagen se muestra el análisis comparativo de las herramientas para determinar la mejor opción para automatizar procesos.

Como último parámetro se evaluó la “Compatibilidad con Navegadores”, cuyo resultado se presenta en la Figura 100, que muestra que UiPath obtuvo una calificación de ALTO en comparación con UI.Visión. El motivo de esta calificación se debió a que UiPath posee compatibilidad con el navegador Safari a excepción de UI.Visión.

Figura 100

Análisis comparativo de la compatibilidad con navegadores

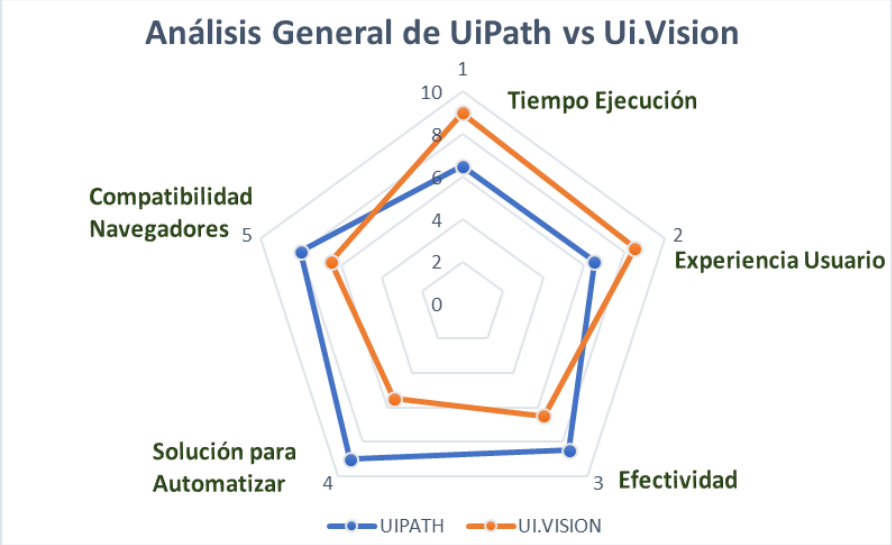


Nota. En la imagen se muestra el análisis comparativo entre las dos herramientas tomando en consideración su compatibilidad con los navegadores web.

Con esta evaluación de parámetros, se estableció un estudio comparativo de las dos herramientas. En la Figura 101, se muestran los resultados obtenidos a partir de la calificación final de cada una de las herramientas, obteniendo como resultado que UiPath posee mejores características en cuanto a efectividad, solución para automatizar una tarea y compatibilidad con navegadores. Por otro lado, la herramienta RPA UI.Vision, es fuerte en los criterios relacionados con el tiempo de ejecución y experiencia de usuario. Con estos resultados se ha determinado que el software UiPath sobresale como herramienta RPA, en el contexto de implementar soluciones de automatización eficientes y funcionales, que a pesar de ser un software con licenciamiento, su costo se compensa con las numerosas ventajas que ofrece en la automatización de tareas más complejas, en comparativa con la herramienta Ui.Vision.

Figura 101

Análisis general de las herramientas UiPath vs UI.Vision



Nota. En la imagen se muestra el análisis general de las herramientas RPA, donde se resaltan los criterios fuertes que posee cada solución.

Capítulo V

Conclusiones y Recomendaciones

Conclusiones

La importancia de la implementación de una medida de seguridad tipo CAPTCHA, radica en su utilización dentro de aplicaciones web que requieran el ingreso/envío de datos. El usuario se puede encontrar con diferentes tipos de desafíos para resolver, dependiendo de la configuración del sitio web. De acuerdo a la literatura analizada, los CAPTCHA más utilizados están basados en texto, en imágenes, en audio, en la resolución de problemas matemáticos o lógicos, así también hay CAPTCHA lúdicos y el llamado reCAPTCHA que asigna un token cada vez que el usuario resuelve un desafío.

Como sucede con cualquier software, los sistemas CAPTCHA no se encuentran exentos de ataques informáticos, más aún si son muy utilizados para prevenir el ingreso de bots maliciosos a sitios web. Se concluye que las técnicas más utilizadas para vulnerar este tipo de medidas de seguridad son los sistemas de reconocimiento de caracteres (OCR), los sistemas de reconocimiento de habla y los analizadores sintácticos, cuya función es intentar resolver los desafíos y simular la interacción de un humano.

En base al análisis comparativo de las herramientas RPA planteado en este estudio, se determinó que UiPath es una de las soluciones más completas y accesibles del mercado. Su versión Community Edition, a pesar de ser limitada, ofrece las opciones suficientes para implementar una automatización funcional. Sin embargo, con la versión licenciada, el costo se compensa con las numerosas ventajas que ofrece para automatizar tareas más complejas, en comparativa con la herramienta UI.Vision.

Se cumplió con el objetivo de desarrollar un bot para vulnerar los desafíos de los sistemas CAPTCHA, se utilizó como referencia el mostrado en la página web *www.map24.com* (revista online francesa) que corresponde a un tipo hCapcha, se ha integrado con la API de

Google Cloud Vision, proporcionando las técnicas de visión artificial para el análisis de imágenes, y se ha complementado el desarrollo utilizando código en el software UiPath. A pesar de que el bot puede llegar a fallar en ciertas ocasiones, cumple con su propósito de resolver el desafío presentado por el hCapctha.

Con esta propuesta, se demostró la integración de las herramientas RPA para automatizar tareas de búsqueda y análisis de vulnerabilidades en páginas web. Este proceso de evaluación servirá como una metodología complementaria a las herramientas ya existentes. Con esta implementación, se han automatizado algunas tareas (búsqueda y análisis de vulnerabilidades), cuyos resultados permitieron el desarrollo de bots que analizan los sistemas CAPTCHA y las vulnerabilidades de tipo XSS.

En base a los resultados obtenidos en el Capítulo IV, se comprueba que el bot desarrollado en UiPath resuelve el desafío CAPTCHA siempre y cuando, la imagen presentada por el hCapctha cumpla con ciertas condiciones (imagen clara, sin texturas, fondos claros). En otro escenario, con las pruebas realizadas en la web oficial hCapctha, el bot analizó las imágenes, pero no resolvió el desafío, esto se debe a que las imágenes actualizadas presentaban texturas que dificultaron el análisis.

En la ejecución de las pruebas de rendimiento, el tiempo que el bot requirió para resolver el desafío CAPTCHA fue variable (de mayor a menor), ya que las imágenes presentadas en forma secuencial eran similares (perros, animales, barcos, gatos, etc.). Con esto se ha demostrado que los sistemas CAPTCHA de tipo hCAPTCHA puede ser vulnerados con facilidad utilizando técnicas de visión artificial.

Cross-site scripting (XSS) se considera como una vulnerabilidad que aprovecha las fallas de seguridad para que los atacantes puedan inyectar scripts ejecutables y maliciosos en aplicaciones o sitios web. Existen tres tipos de ataques, almacenado, reflejado y basado en DOM. Se concluye entonces que el ataque más peligroso es el almacenado porque el atacante

inyecta el código malicioso en el servidor web una sola vez y luego afecta a todos los usuarios web que lo visiten.

La automatización de procesos se ha posicionado en las labores de las empresas u organizaciones actuales, permiten optimizar el tiempo en las tareas relacionadas con web scraping. Existen soluciones como Parsehub que permite extraer datos directamente desde la página web, soluciones basadas en la nube como Dexi.io y Octoparse, que extrae rápidamente datos de cualquier sitio web sin técnicas de codificación.

Empleando la herramienta UI.Vision se implementó un bot que analiza vulnerabilidades de tipo XSS de una manera rápida y sencilla. A través del estudio de los atributos (nombre, is_secure, is_httponly, fecha de creación y fecha de expiración) de las cookies generadas, se observó que los sitios web que utilizan el protocolo HTTP (Hypertext Transfer Protocol) tienden a generar cookies inseguras a comparación de las que usan HTTPs (Hypertext Transfer Protocol Secure).

El bot de análisis de vulnerabilidades XSS se ha estructurado en dos partes, la primera consta en un análisis individual de las páginas web, en donde el tiempo aproximado de ejecución es de 38 segundos por página web. La segunda parte se basa en el análisis del historial completo de un usuario, en donde se evidencia que el tiempo aproximado de ejecución es de 27 segundos por cada URL, incluyendo los 9 segundos que se requieren para abrir una página web. Se concluye entonces, que el tiempo empleado por el bot, es prudente a diferencia del que se obtiene si se realiza el mismo proceso de forma manual. Con este bot se concluye además que dependiendo del número de cookies que un usuario puede almacenar en su PC, aumenta la posibilidad de sufrir ataques de tipo XSS.

Recomendaciones

Con el avance de las tecnologías como la Visión Artificial, los sistemas CAPTCHA cada vez son más vulnerados, motivo por el que es necesario la implementación de nuevas metodologías para mejorar la seguridad en este tipo de sistemas. Una muy buena recomendación es la constante actualización del sistema de visión artificial con las nuevas imágenes que se proporcionan en los desafíos, ya que como se demostró en el presente proyecto, a un bot se le hace más difícil vulnerar imágenes con texturas o patrones incrustados.

Para obtener la ubicación de un componente (button, textfield, títulos, imágenes) la herramienta UiPath realiza un web scraping de la página basándose en su estructura HTML, por ello, para tener una automatización funcional y más precisa es importante detallar la ubicación exacta del componente utilizando el explorador de etiquetas que proporciona UiPath.

Como se demostró en el proyecto, el bot puede llegar a fallar en ciertas ocasiones debido a que las imágenes del CAPTCHA pueden presentar texturas, patrones y/o distorsiones que dificultan el análisis. Para que la detección logre un 100% de efectividad, es recomendable la incorporación de nuevas imágenes con estas características en un dataset propio y actualizado.

La capacidad de ejecución del bot de análisis de vulnerabilidades XSS realizado en UI.Vision es un punto que se puede mejorar, ya que presenta limitantes relacionadas con el uso de comandos (XClick, XType y XMove) en una misma ejecución. La solución más viable a esta limitante es con la obtención de una licencia para la Edición Personal de UI.Visión RPA XModules.

Para mejorar la protección de los sistemas de información dentro de la web, se recomienda complementar nuestra propuesta con un verificador de cookies de terceros, ya que estas podrían obtener información confidencial del usuario que visita dicha página. Es

importante mencionar que ciertas cookies pueden crearse con el fin de acceder a cuentas personales (sesiones) que están activas en un ordenador.

Con el análisis comparativo de las herramientas RPA es importante mencionar que cada una posee características y particularidades que las hacen únicas. Por ello, si se va a automatizar un proyecto complejo el cual requiere del uso de aplicaciones web y de escritorio, se recomienda la implementación del software UiPath, porque posee muchas herramientas de automatización y el respaldo de un software sólido líder en el mercado. Por otro lado, si se desea implementar una automatización sencilla y repetitiva, la utilización de UI.Vision es una muy buena opción, no solo por su facilidad para automatizar, sino también por su portabilidad.

Referencias Bibliográficas

Adminx5. (2022, May 16). *Formularios reCAPTCHA falsos para engañar usuarios a través de sitios de WordPress comprometidos – enHacke*. enHacke. Retrieved May 19, 2022, from https://www.enhacke.com/2022/05/formularios-reCAPTCHA-falsos-para-enganar-usuarios-a-traves-de-sitios-de-wordpress-comprometidos/

Aguas, L., & Paredes, W. (2021, junio 25). Aplicación de Vega Vulnerability Scanner en Aplicaciones Web. *Nexos Científicos*, *5*(1), 1-8. https://nexoscientificos.vidanueva.edu.ec/index.php/ojs/article/view/39

Ahn et al., L. v. (2020). Telling Humans and Computers Apart (Automatically) or How Lazy Cryptographers do AI. *Carnegie Mellon's School of Computer Science*, (3-4), 1-11. http://www.cs.cmu.edu/~biglou/CAPTCHA.pdf

Anti. (2007). *CAPTCHA Solving Service*. Anti CAPTCHA: CAPTCHA Solving Service. Bypass reCAPTCHA, FunCAPTCHA Arkose Labs, image CAPTCHA, GeeTest, HCAPTCHA. Retrieved July 27, 2022, from https://anti-CAPTCHA.com/

Asquith, A., & Horsman, G. (2019). Let the robots do it! – Taking a look at Robotic Process Automation and its potential application in digital forensics. *ScienceDirect*, *1*. https://doi.org/10.1016/j.fsir.2019.100007

Baldassarre, M., Lenarduzzi, V., Romano, S., & Saarimäki, N. (2020). On the diffuseness of technical debt items and accuracy of remediation time when using SonarQube. *ScienceDirect*, *128*. https://doi.org/10.1016/j.infsof.2020.106377

Bhardwaj, M. (2017, febrero 14). *XSS Full Guide Tutorial: All About Cross-Site-Scripting*. iTech Hacks. https://itechhacks.com/xss-full-guide-tutorials/

Blanca, Á. (2021). *Estudio de la Tecnología RPA y su uso en la configuración de redes en Packet Tracer*. Universidad de Valladolid. https://uvadoc.uva.es/handle/10324/50090

Burnham, K. (2021, noviembre 18). *Web application attacks*. Mimecast. https://www.mimecast.com/blog/web-application-attacks/

Bursztein et. al, E. (n.d.). How Good are Humans at Solving CAPTCHAs? A Large Scale Evaluation. *Standford University*. http://web.stanford.edu/~jurafsky/burszstein_2010_CAPTCHA.pdf

Cady, H. (2022). *UI.Vision RPA*. Compsmag. https://www.compsmag.com/alternative/software/ui-vision-rpa/

Callisaya Uchani, Z. A. (2020). CAPTCHA para la Seguridad de la Información en las Aplicaciones Web. *INF-FCPN-PGI Revista PGI*, (7), 112–115. https://ojs.umsa.bo/ojs/index.php/inf_fcpn_pgi/article/view/123

Campos, P. (2017, noviembre 29). *Hacer testeo con Burp Suite*. OpenWebinars. https://openwebinars.net/blog/hacer-testeo-con-burp-suite/

Castro, A. (2020). *Tecnologías RPA en el sector logístico*. Universidad de Cantabria. http://hdl.handle.net/10902/20554

Chalabe, N. (2020, 01 10). *Hacking web (Análisis de ataques SQL Inyección, XSS).* Cartagena. https://repository.unad.edu.co/handle/10596/31471

Chen, H., Chen, J., Chen, J., Yin, S., Wu, Y., & Xu, J. (2020). An Automatic Vulnerability Scanner for Web Applications. *2020 IEEE 19th International Conference on Trust*. https://doi.org/10.1109/TrustCom50675.2020.00207

Cigoj, P., & Blazic, B. (2019). An Intelligent and Automated WCMS Vulnerability-Discovery Tool: The Current State of the Web. *IEEE Access*, *7*. https://doi.org/10.1109/ACCESS.2019.2957573

Condori, H. (2014). *Web Scraping para la obtención de información actualizada de Internet con push notifications para smartphone*. Universidad Mayor de San Andrés. https://repositorio.umsa.bo/handle/123456789/8405

da Silva, M., Soares, P., & Reis, M. (2019). *A Proposal To Use The Tagui Framework As An Rpa Tool* (5, 19th ed.). Institute of Technology Galileo of Amazon. https://dx.doi.org/10.5935/2447-0228.20190020

Deloitte, & AECI. (2020). *Estado Actual de la Ciberseguridad 2020*. IT ahora. Retrieved June 7, 2022, from https://www.itahora.com/wp-content/uploads/2020/06/ESTADO-ACTUAL-DE-CIBERSIGURIDAD-ECUADOR-2020-1.pdf

Dexi.io. (2022). *Dexi.io - Digital Commerce Intelligence, Retail, Brands & E-Commerce*. Dexi.io. https://www.dexi.io/

Escabias, P. R. (2013). *CAPTCHAS. DEBILIDADES Y FORTALEZAS*. Repositorio de la Universidad Carlos III de Madrid. http://hdl.handle.net/10016/18365

Everest Group. (2021). *Robotic Process Automation (RPA) – Technology Vendor Landscape with Products PEAK Matrix® Assessment 2020*. Everest Group.

Fernández, A. L. (2019, junio 19). *Reconocimiento de CAPTCHAs con redes neuronales*. Repositorio Institucional Universidad de la Laguna. https://riull.ull.es/xmlui/bitstream/handle/915/14736/Reconocimiento%20de%20CAPTCHAs%20con%20redes%20neuronales.pdf?sequence=1

Foro Económico Mundial. (2020). *Informe de Riesgos Globales 2022* (Edición 15 ed.). weforum. https://www3.weforum.org/docs/WEF_Global_Risk_Report_2020.pdf

Forsman, T. (2014). *Security in Web Applications and the Implementation of a Ticket Handling System*. Suecia. http://umu.diva-portal.org/smash/get/diva2:696516/FULLTEXT01.pdf

Gartner. (2021). *Magic Quadrant*. Gartner. Retrieved June 17, 2022, from https://www.gartner.es/es/metodologias/magic-quadrant

González, H., & Zúñiga, M. (2017). Estudio del impacto de las cookies en la seguridad de las aplicaciones web. *ResearchGate*. https://www.researchgate.net/publication/324485783_Estudio_del_impacto_de_las_cookies_en_la_seguridad_de_las_aplicaciones_web

Google Cloud. (2022). *Vision AI | Extrae estadísticas de imágenes mediante ML | API de Cloud Vision*. Google Cloud. Retrieved July 14, 2022, from https://cloud.google.com/vision?hl=es

Guixin Ye et. al. (2018, octubre). Otro texto de CAPTCHA Solver: un enfoque generativo basado en redes adversarias. *CCS '18: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pag. 332-248. https://doi.org/10.1145/3243734.3243754

Gupta, B. (2020). *Cross-Site Scripting Attacks - Classifc Countermeasuresation, Attack, and* (1st ed.). Taylor & Francis Group, LLC. https://edu.anarcho-copy.org/Against%20Security%20&%20%20Self%20Security/Cross-Site%20Scripting%20Attacks.pdf

hCAPTCHA. (2022). *Detener más bots. Empieza a proteger la privacidad de los usuarios.* hCAPTCHA - Stop more bots. Start protecting privacy. Retrieved July 25, 2022, from https://www.hCAPTCHA.com/

Hossen, I., & Hei, X. (2021). A Low-Cost Attack against the hCAPTCHA System. *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE Xplore. http://dx.doi.org/10.1109/SPW53761.2021.00061

IBM. (2020, octubre 28). *What is Three-Tier Architecture*. IBM. https://www.ibm.com/cloud/learn/three-tier-architecture

Kali. (2022). *Kali Tools*. Kali Linux. https://www.kali.org/tools/

Kaspersky. (2022). *Amenazas web*. Kaspersky. https://www.kaspersky.com.br/resource-center/threats/web

Kaspersky. (2022). *¿Qué son las cookies?* Kaspersky. https://latam.kaspersky.com/resource-center/definitions/cookies

López, J. (2021, septiembre 09). *Tipos de sistemas web*. ATURA. https://www.atura.mx/blog/tipos-de-sistemas-web

Maldonado, D. A. (2015). Desafíos sobre las nuevas tecnologías de resolución de CAPTCHA y características de evolución de CAPTCHA en el futuro próximo. *MASKANA. Revista Científica Multidisciplinaria.*, *Vol. 6*, 55 - 67. http://dspace.ucuenca.edu.ec/bitstream/123456789/23796/1/2015_TIC.EC_7.pdf

Maldonado, J. (2016). *Desarrollo e implementación de un sistema web de seguimiento y evaluación de las prácticas pre-profesionales para la Facultad de Ingeniería Escuela Civil de la PUCE*. Pontifica Universidad Católica del Ecuador. http://repositorio.puce.edu.ec/handle/22000/12562

Mansfield, M. (2017, January 3). *Cyber Security Statistics: Numbers Small Businesses Need to Know*. Small Business Trends. Retrieved June 7, 2022, from https://smallbiztrends.com/2017/01/cyber-security-statistics-small-business.html

Martinez, D., & Prieto, O. (2009). Servicios Accesibles de Acceso Exclusivamente Humano. *Universidad Carlos III de Madrid. Instituto de Documentación y Gestión de la Información Agustín Millares*, 3 - 4. http://hdl.handle.net/10016/10622

Mitchel, R. (2015). *Web Scraping with Python* (1st ed.). O'Reilly Media, Inc.,. https://coddyschool.com/upload/Web-Scraping-with-Python_proglib.pdf

Mittal, S., Kaushik, P., Hashmi, S., & Kumar, K. (2018). Robust Real Time Breaking of Image CAPTCHAs Using Inception v3 Model. *2018 Eleventh International Conference on Contemporary Computing (IC3)*. IEEE Xplore. https://doi.org/10.1109/IC3.2018.8530607

Navarrete, L. A. (2021). *Comparación entre herramientas que automatizan de procesos repetitivos para aumentar los índices de productividad de los departamentos de administración en las empresas*. Repositorio de la Universidad Ecotec. https://repositorio.ecotec.edu.ec/bitstream/123456789/231/1/NAVARRETE%20LUIS.pdf

Octoparse. (2022). *Octoparse: Herramienta de Web Scraping Gratis*. Octoparse. https://www.octoparse.es/

Ortegón, C. (2019, febrero 07). *Amenazas, vulnerabilidades, factores de riesgo y defensa en profundidad en aplicaciones web*. Universidad Piloto de Colombia. http://repository.unipiloto.edu.co/handle/20.500.12277/4913

OWASP Foundation. (2021). *OWASP Top Ten Web Application Security Risks*. OWASP Foundation. https://owasp.org/Top10/es/A00_2021_Introduction/

ParseHub. (2022). *ParseHub - Un raspador web gratuito que es fácil de usar*. ParseHub. https://www.parsehub.com/

Pianchiche, W. (2021, julio 28). *Software Para Web Scraping Desde Las Apis De Repositorios De Código*. Pontificia Universidad Católica del Ecuador. https://repositorio.pucese.edu.ec/handle/123456789/2605

Ramos, S. (2022, Febrero 23). *Ucrania denuncia un nuevo ciberataque "masivo" contra varias instituciones oficiales y bancos*. EITB. Retrieved May 30, 2022, from

https://www.eitb.eus/es/noticias/internacional/detalle/8673816/ucrania-denuncia-nuevo-ciberataque-masivo-contra-varias-instituciones-oficiales-y-bancos/

Reina R., M. J. (2021). *Automatización de procesos mediante la herramienta Uipath* [Trabajo Fin de Grado Inédito]. Escuela Técnica Superior de Ingeniería Universidad de Sevilla.

Ribeiro, J. (2021). Robotic Process Automation and Artificial Intelligence in Industry 4.0 – A Literature review. *ScienceDirect*, *181*, 51-58. https://doi.org/10.1016/j.procs.2021.01.104

Rodríguez, G., Benavides, D., Torres, J., & Fuertes, W. (2018, enero 05). Cookie Scout: An Analytic Model for Prevention of Cross-Site Scripting (XSS) Using a Cookie Classifier. *Springer Link*, 497–507. http://dx.doi.org/10.1007/978-3-319-73450-7_47

Rodríguez, G., Torres, J., Flores, P., & Benavides, D. (2019). Cross-site scripting (XSS) attacks and mitigation: A survey. *Computer Networks*. ResearchGate. http://dx.doi.org/10.1016/j.comnet.2019.106960

Ruiz, R. (2021, septiembre 28). *Soporte al microemprendimiento por medio de prototipo de dashboard de datos obtenidos mediante Web Scraping en medios digitales.* Universidad de Guayaquill. http://repositorio.ug.edu.ec/handle/redug/56200

Saquinaula, G. R. (2013). *Análisis del CAPTCHA características, problemática y aplicaciones*. Repositorio de la Universidad de Israel. http://repositorio.uisrael.edu.ec/handle/47000/351

School E. B. (2021). *Retos en la Supply Chain: Claves de la planaificacion para la cadena de suministro.* Madrid.

Sgobba, J. (2021, julio 13). *Vulnerabilidad XSS (Cross Site Scripting): Qué es y cómo solucionarla*. HackMetrix. https://blog.hackmetrix.com/xss-cross-site-scripting/

Shoshitaishvili et al., Y. (2018, marzo 30). Mechanical Phish: Resilient Autonomous Hacking. *IEEE Security & Privacy*, *Vol.16*(N. 2), pag. 12 - 22. 10.1109/MSP.2018.1870858

Simonite, T. (2017, febrero 13). *Este robot hacker encuentra y repara vulnerabilidades automáticamente*. Wikipedia, the free encyclopedia. Retrieved May 30, 2022, from https://www.technologyreview.es//s/6756/este-robot-hacker-encuentra-y-repara-vulnerabilidades-automaticamente

Sivanesan, A., Mathur, A., & Javaid, A. (2018). A Google Chromium Browser Extension for Detecting XSS Attack in HTML5 Based Websites. *2018 IEEE International Conference on Electro/Information Technology (EIT)*. IEEE Xplore. https://doi.org/10.1109/EIT.2018.8500284

Sotelo, A. (2018). *Soluciones basadas en automatización robótica de procesos (RPA) para la integración de sistemas empresariales y automatización de procesos de negocio en el sector seguros*. Universidad Politécnica de Madrid. https://oa.upm.es/54781/1/TFM_ANDY_MIGUEL_SOTELO_LEZAMA.pdf

Subía, E. (2018, enero 10). *Análisis de los ataques a aplicaciones web SQL Injection y Cross Site Scripting y sus medidas de precaución y defensa*. Repositorio Digital Universidad Técnica del Norte. http://repositorio.utn.edu.ec/handle/123456789/7803

Taulli, T. (2020). *The Robotic Process Automation Handbook: A Guide to Implementing RPA Systems*. https://doi.org/10.1007/978-1-4842-5729-6

Tutorials Point. (2019). *Robot Framework* (1st ed.). Tutorials Point (I) Pvt. Ltd. https://www.tutorialspoint.com/robot_framework/robot_framework_tutorial.pdf

UiPath Community Edition. (2022). *Descarga la Edición Community de UiPath*. UiPath. Retrieved June 27, 2022, from https://www.uipath.com/es/desarrollo/edicion-community

UI.Vision. (2022). *Open-Source RPA Software 2022 for macOS, Linux and Windows. Codeless UI Automation*. UI Vision. Retrieved July 10, 2022, from https://ui.vision/rpa

Vázquez, A. M. (2010). Diseño e Implementación de un Protocolo Seguro de Intercambio de Mensajes con un Dispostivo Seguro. In *PROYECTO FIN DE CARRERA*. Repositorio de la Universidad Carlos III de Madrid. https://e-archivo.uc3m.es/bitstream/handle/10016/9586/PFC_Ana_Vazquez_Pacheco.pdf?sequence=1&isAllowed=y

Vega, O., & Vinasco, A. (2014, julio 01). CAPTCHA. ¿Una solución para la seguridad informática o problema para la accesibilidad/usabilidad web? *e-Ciencias de la Información*, *Vol. 4*(No. 2), p. 1 -14. http://repositorio.ucr.ac.cr/bitstream/handle/10669/12571/15125-27618-1-PB.pdf?sequence=1&isAllowed=y

VentureBeat. (2022, febrero 21). *Report: 50% of all web applications were vulnerable to attacks in 2021*. VentureBeat. https://venturebeat.com/2022/02/21/report-50-of-all-web-applications-were-vulnerable-to-attacks-in-2021/

Verry, J. (2021, diciembre 17). *Web Application Attacks are Skyrocketing—Don't Get Caught in the Crossfire*. Pivot Point Security. https://www.pivotpointsecurity.com/blog/web-application-attacks-are-skyrocketing-dont-get-caught-in-the-crossfire/

Yashodhan, K. (2021, junio 19). Effectiveness of Robotic Process Automation for data mining using UiPath. *IEEE Xplore*. https://doi.org/10.1109/ICAIS50930.2021.9396024

Yatskiv, N., Yatskiv, S., & Vasylyk, A. (2020). Method of Robotic Process Automation in Software Testing Using Artificial Intelligence. *IEEE Xplore*. https://doi.org/10.1109/ACIT49673.2020.9208806