UNIVERSIDAD TECNOLÓGICA DE PEREIRA

IMPLEMENTACION OCTOMATRiX

LABORATORIO DE CYBERSEGURIDAD

SEMILLERO BLACKBYTE

PARCHATE PEREIRA + OCTOMATRIX

Informe No 1.1

Docente: Biviana Yulieth Aguirre Arias

Integrantes:
Santiago Potes Giraldo

Resumen

El laboratorio de OctoMatrix, identificado como el **"Pilar 1: ML + Regex OWASP top 10 library for the new network"**, constituye un **Pipeline Completo de Seguridad** basado en Python que utiliza Machine Learning para la detección de ataques de inyección,.

Su objetivo es entrenar un clasificador, específicamente un **RandomForestClassifier**, para diferenciar el tráfico web legítimo de diversas amenazas de inyección,. La base de este laboratorio es la colección de **datos estilo Kaggle y datos sintéticos**,. Esto incluye la generación de muestras realistas de ataques del OWASP Top 10, tales como **SQL Injection** (ej. `' OR '1'='1'--`), **XSS**, **Path Traversal** (ej. `../../../etc/passwd`), y **Command Injection** (ej. `| cat /etc/passwd`), además de patrones simulados de **Parameter Pollution** y **Buffer Overflow**.

Para optimizar la precisión, el pipeline emplea una sofisticada **Ingeniería de Características**. Combina la vectorización estándar **TF-IDF** con la **Extracción de Características Avanzadas** basadas en Regex,. Estas características avanzadas analizan la longitud del texto, el conteo de **caracteres especiales** (`<>;='\"&|%`), y la detección de **palabras clave SQL(** (`SELECT`, `DROP`) y **patrones XSS** (`<script>`). También calcula la **Entropía de Shannon** del texto.

Tras combinar todas las características, el modelo se entrena,. El resultado del laboratorio es la exportación del modelo entrenado y el vectorizador a un archivo `.pkl`,. Una prueba rápida verifica su capacidad para clasificar correctamente *payloads* maliciosos como `ATAQUE`, . Este proceso asegura la **Protección de Tráfico en Tiempo Real** dentro de la arquitectura del sistema

1. MARCO TEÓRICO

El marco teórico del proyecto Parchate Pereira y su núcleo de seguridad, OctoMatrix, se basa en una combinación de filosofía de desarrollo rigurosa, arquitectura de sistemas escalable y la aplicación de Machine Learning para la seguridad en tiempo real.

A continuación, se presenta el marco teórico, dividido en principios arquitectónicos, pilares técnicos de seguridad y manejo de datos

Arquitectura Segura y Escalable

I. Filosofía Arquitectónica y Diseño (Safe by Design)

El proyecto se fundamenta en un conjunto de convicciones que priorizan la eficiencia y la seguridad desde la concepción del sistema:

a1. **Seguridad como Fundamento:** El principio rector es que **"Safe by Design isn't a feature - it's the foundation"**. Esto implica que la seguridad no es un añadido, sino la base estructural del diseño.

a2. **Arquitectura Limpia y Escalabilidad:** Se utiliza un *backend* construido con **Python/Flask** basado en **Arquitecturas Limpias**. El objetivo de diseño es crear una **"Arquitectura que escala a planetas"**, logrando un **"PLANET-SCALE IMPACT"**.

a3. **Eficiencia por Necesidad:** El entorno de desarrollo se rige por la restricción **"If it doesn't run on a chromebook, it won't scale to a planet"**. Esta limitación garantiza **código "puro, eficiente"** y evita el **"cloud bloat"** (hinchazón de la nube).

a4. **Desarrollo Mínimo Viable (MVP):** La metodología se enfoca en la construcción rápida y limpia, demostrada por el enfoque de **"MVP in 6hr"**.

II. Pilares Técnicos del Sistema

La funcionalidad del sistema se divide en pilares modulares:

b1. **Pilar 1: OctoMatrix (Seguridad Inteligente)**

- **Definición:** OctoMatrix es una **"ML + Regex OWASP top 10 library for the new network"**.
- **Función:** Proporcionar **"Real-time Traffic Protection"** (Protección de Tráfico en Tiempo Real).
- **Mecanismo:** Actúa como un *middleware* robusto que utiliza un **Pipeline ML de Seguridad**.

b2. **Pilar 2: Event Crawler (Recopilación Ética de Datos)**

- **Definición:** Componente encargado del **"Ethical Scraping in social media"**.
- **Alcance:** Provee entre el **"60-70% data"** necesaria para el sistema, con la restricción de **no utilizar información OSINT** (Open Source Intelligence). Los datos recopilados se integran a través de un **Kafka Real-time Stream**.

b3. **Inteligencia Artificial (IA):**

- El proyecto utiliza **OctoMatrix + Gemini** para su funcionalidad principal.
- El uso de la **Gemini API** se rige por el modelo **"Paga Por Preguntar"** (Pay to Ask), con una licencia que aplica a chatbots y traductores.

III. Marco Teórico de Detección de Ataques (OctoMatrix ML Pipeline)

El núcleo de OctoMatrix es el `SecurityMLPipeline`, que opera bajo un marco de detección de anomalías y patrones de ataque:

c1. **Modelo de Clasificación:** Se utiliza el **RandomForestClassifier** para clasificar las entradas de tráfico como "Normal" o "Ataque".

c2. **Fuentes de Entrenamiento:** El modelo se entrena con **"Kaggle-style datasets"** y **"Synthetic data"**. Esto incluye la generación de patrones de vulnerabilidades críticas como **SQL Injection** (`admin' OR '1'='1'--`), **XSS**, **Path Traversal** (`../../../etc/passwd`), e **Inyección de Comandos** .

c3. **Ingeniería de Características:** Para maximizar la precisión, el pipeline combina dos tipos de características:

- **TF-IDF:** Utilizando **TfidfVectorizer** para el análisis de frecuencia de términos.
- **Características Avanzadas (Regex):** Extracción de métricas clave como el conteo de **palabras clave SQL** (`SELECT`, `DROP`), **patrones XSS** (`<script>`), detección de **caracteres especiales** (`<>;='\"&|%`) y el cálculo de la **Entropía de Shannon**.

c4. **Entrega:** El resultado final del laboratorio es un **paquete completo** (`.pkl`) que contiene el modelo y el vectorizador entrenados, listo para ser implementado para la predicción en tiempo real.

2. MATERIALES

El proyecto Parchate Pereira utiliza un *stack* tecnológico diseñado bajo los principios de **Arquitectura Limpia** y **Safe by Design**,.

A continuación, se indican los materiales y componentes clave utilizados en la implementación y el *pipeline* de seguridad de OctoMatrix, con la confirmación de que el proyecto está desplegado en **Render**.

I. Stack de Convicciones (Materiales del MVP)

Categoria	Componentes Utilizados	Descripción / Rol
BACK END	Python , FLASK	Utilizado para la construcción del backend siguiendo principios de Arquitecturas Limpias,.
FRONT END	HTML/CSS/JS	Componentes estándar usados "cuando es necesario".
INTELIGENCIA ARTIFICIAL	GEMINI API	Usada junto a OctoMatrix, con una licencia aplicada para chatbots y traductores bajo el modelo "Paga Por Preguntar".
BLOCKCHAIN	Web3 Educativo	Implementado

	Tokens Eticos	para la capa de Blockchain
BASES DE DATOS	SQL Server / PostgreSQL	Conocimiento y experiencia en la integración y optimización de estas bases de datos
GESTION DE DATOS	kafka Real-time Stream	Utilizado en la arquitectura para gestionar el flujo de datos en tiempo real proveniente del Pilar 2 (Event Crawler).

2.a tabla del stack de Convicciones define los componentes esenciales utilizados en la construcción del Producto Mínimo Viable (MVP)

II. Materiales del Laboratorio de Seguridad (OctoMatrix)

OctoMatrix es implementado usando el siguiente conjunto de herramientas y librerías de Machine Learning:

Tipo de Herramienta	Librerías Específicas	Propósito dentro de OctoMatrix
Lenguajes	Python	Lenguaje de programación principal del pipeline.
Estructuras de datos	Pandas, NumPy	Utilizados para el manejo y procesamiento de datos y la combinación de matrices de características,.
machine learning	Scikit-learn	Provee los algoritmos clave, incluyendo RandomForestClassifier (el modelo de clasificación) y TfidfVectorizer (para la extracción de características de texto)
Procesamiento de texto	Regex (re)	Utilizado para la Extracción de Características Avanzadas, como el conteo de palabras clave SQL y patrones XSS
Serializacion	Pickle	Empleado para exportar el modelo entrenado y el vectorizador a archivos .pkl para su implementación,.
Fuentes de datos	Datasets estilo Kaggle, Datos Sintéticos	Utilizados para entrenar el modelo en la detección de ataques OWASP Top 10,.

2.b tabla del stack del pipeline de octomatrix

III. Entorno e Infraestructura

Parchate Pereira ya está desplegado en Render.

El proyecto está diseñado para funcionar en infraestructura en la nube, y su costo de infraestructura se estima en **$500/mes** en servidores de **Render o AWS**.

El proyecto de prueba de concepto (el "proof") tiene un despliegue activo en la plataforma **Render**, con la URL `https://pereira-explorer-ai-parchate-pereira.orender.com/`.

Además, la filosofía del sistema está basada en el uso de restricciones de hardware, con el lema: **"If it doesn't run on a chromebook, it won't scale to a planet"**, lo que implica que la máquina de desarrollo debe ser lo más ligera posible.

3. METODOLOGÍA EXPERIMENTAL

La metodología experimental se centra en la construcción y validación del **Pilar 1: OctoMatrix**, que funge como el *Security Machine Learning Pipeline* (`SecurityMLPipeline`). Este proceso garantiza la **Protección de Tráfico en Tiempo Real** mediante la aplicación de técnicas avanzadas de Machine Learning y Regex, cumpliendo con la filosofía de **"Safe by Design"**.

3.1. Recolección y Preparación de Datos

La fase inicial se realiza mediante la función `collect_kaggle_style_data`, que combina diversas fuentes para lograr una base de entrenamiento robusta:

- **Generación de Ataques Sintéticos:** El modelo se entrena con datos sintéticos que replican ataques críticos del **OWASP Top 10**. Esto incluye la creación de muestras realistas de:
- **Inyección SQL (SQLi):** Patrones como `admin OR 1'='1'--` o sentencias `DROP TABLE users`.
- **Cross-Site Scripting (XSS):** Patrones inyectados en *scripts* o llamadas `javascript:alert('XSS')`.
- **Path Traversal:** Intentos de acceso a archivos sensibles, como `../../../etc/passwd`.
- **Inyección de Comandos:** Patrones que ejecutan comandos del sistema operativo, como `| cat /etc/passwd`.
- **Otros Patrones:** Se incluyen patrones de vulnerabilidades como *Parameter Pollution* y *Buffer Overflow* (estilo CSIC 2010).
- **Tráfico Normal:** se generan muestras de tráfico legítimo (peticiones API, navegación web, rutas de archivos) para enseñar al modelo a distinguir los patrones seguros.
- **Ensamblaje:** Los textos de ataque (etiquetados con 1) y los textos normales (etiquetados con 0) se combinan en un conjunto de datos único.

3.2. Ingeniería de Características y Procesamiento

Para optimizar la capacidad predictiva del modelo, se implementa una combinación de características:

3.2.1. **Características TF-IDF:** Se aplica **TfidfVectorizer** a los textos para vectorizar y medir la importancia de las palabras y frases (utilizando *n-grams* de 1 a 3).

3.2.2. **Características Avanzadas (Regex):** Mediante la función `extract_advanced_features`, se utilizan expresiones regulares (`re`) para cuantificar indicadores de ataque que no son capturados por TF-IDF. Estas incluyen:

- Conteo de **palabras clave SQL** (`SELECT`, `DROP`, `EXEC`).
- Conteo de **patrones XSS** y detección de *path traversal* .
- Conteo de **caracteres especiales** peligrosos.
- Cálculo de la **Entropía de Shannon** del texto.

3.2.3. **Combinación:** Las características TF-IDF y las avanzadas se unen en una única matriz mediante `np.hstack` para el entrenamiento.

3.3. Entrenamiento y Validación del Modelo

- **Modelo y División:** Se utiliza un **RandomForestClassifier**. El conjunto de datos combinado se divide en conjuntos de entrenamiento y prueba (por defecto, 80/20) mediante `train_test_split`.
- **Evaluación:** Una vez entrenado, el modelo se valida calculando la **precisión** (`accuracy_score`) y generando un **Reporte de Clasificación** para las clases "Normal" y "Ataque".
- **Prueba de Integridad:** Se realiza una prueba rápida (`quick_test`) con casos de ataque conocidos para asegurar que el modelo clasifique el *payload* correctamente como **"🚨 ATAQUE"**.

3.4. Implementación y Serialización

Una vez validado, el modelo se prepara para su uso en la aplicación Parchate Pereira, que está desplegada en **Render**:

- **Exportación:** El modelo entrenado (`RandomForestClassifier`) junto con el `TfidfVectorizer` se serializa en un archivo `.pkl`.
- **Metadata:** El paquete de exportación incluye metadatos como la precisión final, la versión y la información sobre las fuentes de datos utilizadas.

4. RESULTADOS EXPERIMENTALES

La siguiente sección presenta los datos en crudo (sin procesar) utilizados para entrenar el OctoMatrix Security Machine Learning Pipeline (SecurityMLPipeline). Estos datos son la base sintética y simulada de ataques y tráfico normal, utilizados para definir el ground truth del modelo antes de la fase de Ingeniería de Características.

Nota sobre Magnitudes y Unidades: Dado que el laboratorio se centra en la detección de patrones textuales de ataques web (payloads), las "magnitudes" son cadenas de texto (strings) y no poseen unidades físicas convencionales. La incertidumbre se considera nula en la recolección, ya que las etiquetas de clasificación (Label) son el ground truth definido por el investigador (0 = Normal, 1 = Ataque) para el entrenamiento.

Caso de Estudio	Payload (Cadena de Texto en Crudo)	Clasificación / Etiqueta	Fuente de Generación
SQL Injection (SQLi)	admin' OR '1'='1'--	1	OWASP
SQL Injection (SQLi)	' UNION SELECT username, password FROM users–`	1	OWASP
SQL Injection (SQLi)	'; DROP TABLE users;--	1	OWASP
SQL Injection (SQLi)	OR 1=1--	1	OWASP
SQL Injection (SQLi)	; EXEC xp_cmdshell('format c:');--	1	OWASP
XSS	javascript:alert('XSS')	1	OWASP
Path Traversal	../../../etc/passwd	1	OWASP
Path Traversal	..\\..\\windows\\system32\\drivers\\etc\\hosts	1	OWASP
Command Injection	`; cat /etc/passwd`	1	OWASP
Command Injection	; rm -rf /	1	OWASP
XXE Injection (XXE)	%init;]>	1	OWASP
EXTRA PATTERNS	text	1	OWASP

Tabla 1. Datos Sintéticos de Ataques OWASP Top 10

Caso de Estudio	Payload (Cadena de Texto en Crudo)	Clasificación / Etiqueta	Fuente de Generación
Parameter Pollution	user=admin&user=guest	1	CSIC
Parameter Pollution	id=1&id=2&id=3	1	CSIC
Buffer Overflow	A (repetido 1000 veces)	1	CSIC
Buffer Overflow	password=+ y (repetido 1000 veces)	1	CSIC
Integer Overflow	limit=2147483648	1	CSIC
Integer Overflow	offset=-1	1	CSIC
Format String	user=%s%s%s%s%s	1	CSIC

TABLA 2: Datos Sintéticos de Patrones CSIC 2010

Estos payloads representan otros tipos de ataques de inyección y sobrecarga, también etiquetados como Ataque (Label = 1), simulando patrones conocidos como los del conjunto CSIC.

Caso de Estudio	Payload (Cadena de Texto en Crudo)	Clasificación / Etiqueta	Fuente de Generación
API Requests	/api/users/list?page=1&limit=10	0	Normal Traffic
API Requests	/auth/login?user=john&pass=secure123	0	Normal Traffic
Web Traffic	/home	0	Normal Traffic
Web Traffic	/search?term=hello+world	0	Normal Traffic
File Paths	/static/css/m	0	Normal

	ain.css		Traffic
File Paths	/downloads/document.pdf	0	Normal Traffic
Data Inputs	user@company.com	0	Normal Traffic
Data Inputs	securePassword123!	0	Normal Traffic

TABLA 3: Datos Sintéticos de Tráfico Normal

Este conjunto de datos representa el tráfico legítimo y seguro, etiquetado como Normal (Label = 0), crucial para enseñar al modelo a no generar falsos positivos.

5. PROCESAMIENTO DE DATOS EXPERIMENTALES

En esta sección se detalla la transformación de los datos en crudo (payloads) en un modelo de seguridad funcional. El **procesamiento de datos** del proyecto Parchate Pereira se basa en un pipeline híbrido que combina el análisis estadístico de texto con la extracción de métricas heurísticas avanzadas.

A continuación, se describen las fases del procesamiento experimental realizado por el `SecurityMLPipeline`:

5.1. Vectorización de Texto (TF-IDF)

Los datos textuales se transforman en representaciones numéricas mediante la técnica **TF-IDF** (*Term Frequency-Inverse Document Frequency*).

- **Configuración:** Se utiliza un `TfidfVectorizer` con un rango de **n-gramas de (1, 3)**, lo que permite al sistema analizar no solo palabras individuales, sino secuencias de hasta tres caracteres o palabras,.
- **Objetivo:** Esta fase identifica la relevancia estadística de ciertos patrones de ataque (como palabras clave de SQL o etiquetas HTML) frente al tráfico normal.

5.2. Extracción de Características Avanzadas (Regex)

Simultáneamente a la vectorización, el sistema ejecuta la función `extract_advanced_features` para obtener métricas específicas que el análisis TF-IDF podría ignorar. Estas características incluyen:

- **Métricas de Estructura:** Longitud total del texto y ratio de espacios en blanco,.
- **Detección de Patrones Críticos:** Conteo de **caracteres especiales** (`<>;='\"&|%`) y detección de **palabras clave SQL** o **patrones XSS** mediante expresiones regulares.
- **Análisis de Complejidad:** Cálculo de la **Entropía de Shannon**, utilizada para detectar *payloads* codificados o aleatorios que intentan evadir filtros simples.
- **Codificación:** Detección de patrones de codificación URL (`%XX`)

5.3. Consolidación de la Matriz de Datos

Una vez procesadas ambas vertientes, los resultados se unifican:

- **Combinación Híbrida:** Se utiliza la función `np.hstack` para fusionar la matriz de características TF-IDF con el vector de características avanzadas.
- **Resultado:** Se genera una matriz única de entrenamiento (`X_combined`) que ofrece una visión multidimensional de cada muestra de datos.

5.4. Entrenamiento y Evaluación del Clasificador

Con la matriz consolidada, se procede a la fase de aprendizaje automático:

- **Algoritmo:** Se implementa un **RandomForestClassifier** con 100 estimadores, seleccionado por su robustez ante datos ruidosos y su capacidad para manejar múltiples características,.
- **Validación:** Los datos se dividen en un **80% para entrenamiento** y un **20% para pruebas** (`train_test_split`), asegurando una evaluación imparcial de la precisión del modelo,.
- **Métricas de Éxito:** El rendimiento se mide mediante un reporte de clasificación que incluye la **precisión (accuracy)**, el *recall* y el F1-score para las categorías de "Normal" y "Ataque".

5.5. Exportación para Despliegue (Render)

El paso final del procesamiento es la serialización para su uso en producción:

- **Empaquetado:** El modelo entrenado y el vectorizador se guardan en un archivo de formato **.pkl** utilizando la librería `pickle`,.
- **Implementación:** Este paquete permite que la instancia activa en **Render** realice predicciones en tiempo real sobre el tráfico entrante, garantizando la **"Protección de Tráfico en Tiempo Real"**,.

Analogía para entender el procesamiento: Imagine que OctoMatrix es un guardia de seguridad en una discoteca. El **TF-IDF** es como revisar si la persona lleva ropa prohibida (patrones conocidos), mientras que la **Extracción de Características Avanzadas** es como medir su nivel de nerviosismo o revisar si lleva objetos ocultos (comportamiento y estructura). Al final, el guardia combina ambas impresiones para decidir si permite el paso o da una señal de **ATAQUE**.

6. ANÁLISIS Y DISCUSIÓN DE RESULTADOS

El análisis de los resultados obtenidos a través del **SecurityMLPipeline** de OctoMatrix demuestra la viabilidad de integrar inteligencia artificial y heurísticas tradicionales para la protección de aplicaciones web en tiempo real. A continuación, se discuten los hallazgos basados en los datos y la arquitectura del sistema:

Eficacia de la Detección Híbrida

La principal fortaleza del sistema reside en su enfoque híbrido. Mientras que la vectorización **TF-IDF** permite identificar la importancia estadística de términos en secuencias de hasta tres caracteres (n-grams), la **Extracción de Características Avanzadas** mediante Regex compensa las limitaciones del análisis estadístico puro.

- **Identificación de Patrones Críticos:** El uso de expresiones regulares específicas para detectar **palabras clave SQL** (`SELECT`, `UNION`) y **patrones XSS** garantiza que intentos de inyección conocidos sean interceptados con alta prioridad.
- **Análisis de Evasión:** El cálculo de la **Entropía de Shannon** es un resultado técnico clave, ya que permite al modelo identificar *payloads* que intentan evadir filtros mediante codificación aleatoria u ofuscación, un comportamiento típico en ataques avanzados.

Rendimiento y Validación del Modelo

El modelo **RandomForestClassifier** fue seleccionado por su capacidad para procesar la matriz combinada de 1,500 características TF-IDF y métricas avanzadas.

- **Precisión:** El sistema reporta una alta precisión en la clasificación de las clases "Normal" y "Ataque". En la implementación práctica de Parchate Pereira, esto se traduce en una **"Búsqueda segura del 83%"**, lo que valida la efectividad del entrenamiento con datos sintéticos y de estilo Kaggle.
- **Fiabilidad en Tiempo Real:** Durante la **Prueba Rápida (`quick_test`)**, el modelo demostró ser capaz de asignar niveles de confianza específicos a sus predicciones (ej. identificando correctamente un ataque de *path traversal* con un mensaje de **ATAQUE**).

Discusión sobre la Arquitectura "Safe by Design"

Los resultados confirman que la seguridad no es un componente externo, sino el cimiento del sistema (**"Safe by Design isn't a feature - it's the foundation"**).

- **Eficiencia Operativa:** El hecho de que el sistema pueda operar en un entorno restringido (menos de 10GB de espacio, como un **Chromebook**) demuestra que es posible lograr un **"Impacto a Escala Planetaria"** sin recurrir al *cloud bloat* o desperdicio de ciclos de procesamiento.
- **Viabilidad en Producción:** La exportación exitosa a un archivo `.pkl` y su posterior despliegue en **Render** prueban que el laboratorio de OctoMatrix es una solución lista para la **Protección de Tráfico en Tiempo Real** en aplicaciones Web 2 y 3.

En conclusión, el éxito del pipeline radica en su capacidad para transformar datos crudos de ataques en un modelo ligero y altamente predictivo, permitiendo que la aplicación Parchate Pereira mantenga un entorno seguro mientras escala globalmente.

A continuación se presentan las conclusiones formales del laboratorio y los pasos futuros para el desarrollo del ecosistema **Parchate Pereira** y la librería **OctoMatrix**, basándose en la filosofía de diseño y los resultados técnicos obtenidos:

7. CONCLUSIONES FORMALES

7.1. **La Seguridad como Cimiento Estructural:** Se valida la tesis de que el principio **"Safe by Design"** no es una característica opcional, sino la base fundamental de cualquier arquitectura de software moderna. El laboratorio demuestra que es posible integrar una capa de seguridad robusta (Pillar 1) desde la fase de MVP sin comprometer la velocidad de desarrollo.

7.2. **Efectividad del Modelo Híbrido:** La combinación de **Machine Learning (RandomForest)** y **heurísticas de Regex** en OctoMatrix resulta altamente efectiva para la **protección de tráfico en tiempo real**. El sistema logra identificar con precisión ataques del **OWASP Top 10**, como SQL Injection y XSS, alcanzando una métrica de **"Búsqueda segura del 83%"** en el entorno de Parchate Pereira.

7.3. **Eficiencia y Escalabilidad:** Bajo la restricción de diseño de que "si no corre en un Chromebook, no escalará a un planeta", se concluye que el código generado es **puro, eficiente y libre de "cloud bloat"**. El uso de menos de 10GB para construir los cimientos de "Nivel 1 de Civilización" demuestra que la arquitectura limpia permite un **impacto a escala planetaria** con recursos mínimos.

7.4. **Viabilidad del Modelo de Negocio:** El despliegue exitoso en **Render** y el uso de la **Gemini API** bajo el esquema de **"Paga por preguntar"** confirman la viabilidad técnica y comercial del proyecto, manteniendo costos de infraestructura controlados (aprox. $500/mes).

8. FUTUROS PASOS

- **Escalamiento a "Planet-Scale":** Evolucionar la arquitectura actual de microservicios para soportar una carga global, manteniendo la elegancia del código y la eficiencia en el consumo de recursos.
- **Optimización del Pilar 2 (Event Crawler):** Mejorar los algoritmos de **"Ethical Scraping"** en redes sociales para aumentar la tasa de actualización automática del sistema (actualmente en 17%) y alcanzar una cobertura de datos del 60-70% sin recurrir a información OSINT.
- **Integración Web 2 y Web 3:** Profundizar en la integración de OctoMatrix como una guía pública de PiTech para aplicaciones tanto en la red tradicional como en entornos de **Blockchain y Tokens Éticos**.
- **Liderazgo y Mentoría en PiTech:** Bajo el liderazgo de Santiago Potes como CEO y Arquitecto, se planea expandir el equipo de investigación de **PiTech** para seguir transformando "problemas humanos en código elegante".
- **Refinamiento del Pipeline de IA:** Mejorar el `SecurityMLPipeline` mediante la inclusión de datasets de ataques más complejos y el ajuste fino de la **Entropía de Shannon** para detectar amenazas persistentes avanzadas.

Metáfora final: Construir este sistema es como edificar un rascacielos en una zona sísmica: **OctoMatrix** no es solo la fachada o las cámaras de seguridad, sino los amortiguadores y cimientos profundos que permiten que la estructura crezca hacia las nubes sin temor a derrumbarse ante cualquier vibración externa.

9. REFERENCIAS BIBLIOGRÁFICAS

Referencias bibliográficas (formato página web / dataset / repositorio técnico)

[6] Impact Cyber Trust. Cybersecurity Datasets Repository: Dataset ID 940. Impact Cyber Trust, plataforma de datos abiertos para investigación en ciberseguridad. Disponible en: https://www.impactcybertrust.org/dataset_view?idDataset=940

(Consulta: diciembre de 2025).

[7] OWASP SpiderLabs. OWASP Core Rule Set (CRS) Regressions. Proyecto SpiderLabs, repositorio oficial de reglas de seguridad para detección de ataques web OWASP Top 10. Disponible en: https://github.com/SpiderLabs/OWASP-CRS-regressions

(Consulta: diciembre de 2025).

[8] Ravi Kumar. HackOWASP 2025 – LB1 v1. Kaggle Notebook orientado al análisis y clasificación de ataques OWASP mediante técnicas de Machine Learning. Plataforma Kaggle. Disponible en: https://www.kaggle.com/code/ravi20076/hackowasp2025-lb1-v1

(Consulta: diciembre de 2025).

[9] Canadian Institute for Cybersecurity (CIC), University of New Brunswick. CIC-IDS2018 Dataset. Conjunto de datos para detección de intrusiones en redes, ampliamente utilizado en investigación académica en ciberseguridad. Disponible en: https://www.unb.ca/cic/datasets/ids-2018.html

(Consulta: diciembre de 2025).