Manual de Implementación

OctoMatrix como Middleware de Validación de Eventos

---

1. Propósito del Documento

Este manual describe los pasos técnicos y conceptuales necesarios para implementar **OctoMatrix** como un middleware de validación de eventos dentro de una arquitectura basada en productores y consumidores. El documento está diseñado para entornos académicos, de laboratorio y prototipos controlados, priorizando claridad arquitectónica sobre optimizaciones de producción.

OctoMatrix **no reemplaza** mecanismos de seguridad existentes; actúa como una **capa adicional de correlación y observabilidad** entre comportamiento de red e intención de aplicación.

---

2. Alcance de la Implementación

La implementación descrita cubre:

* Validación de eventos a nivel de aplicación usando modelos entrenados (archivo `.pkl`).
* Correlación básica entre identidad de red (IP/MAC) y acciones observadas.
* Integración lógica con aplicaciones existentes (ej. Parchate Pereira).

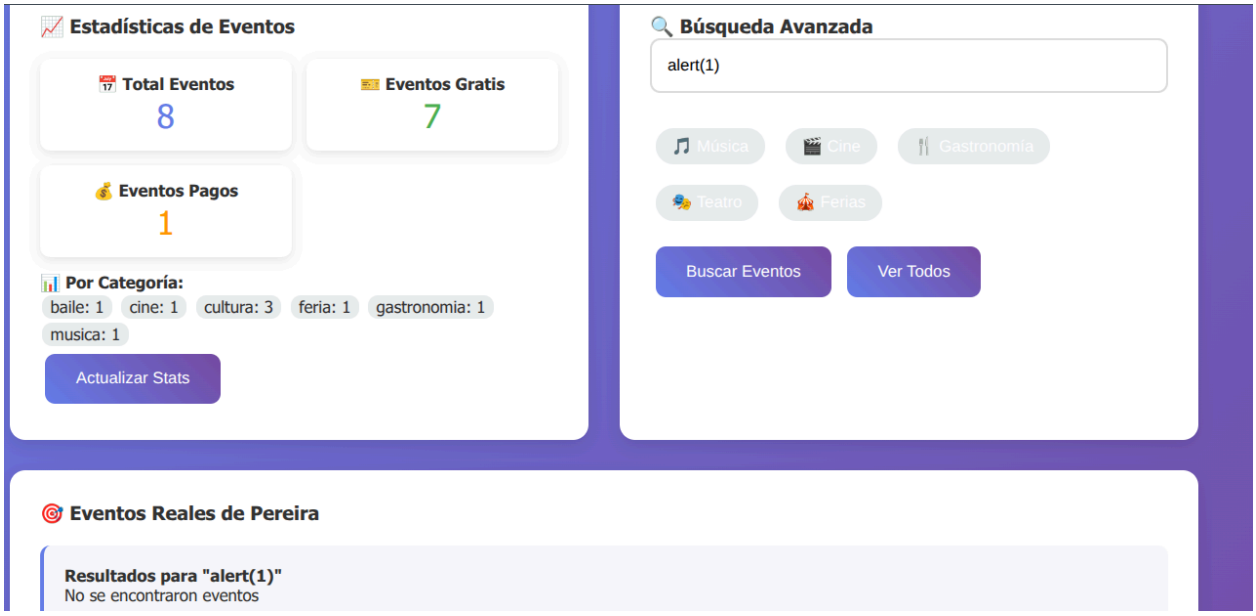
Fuera de alcance:

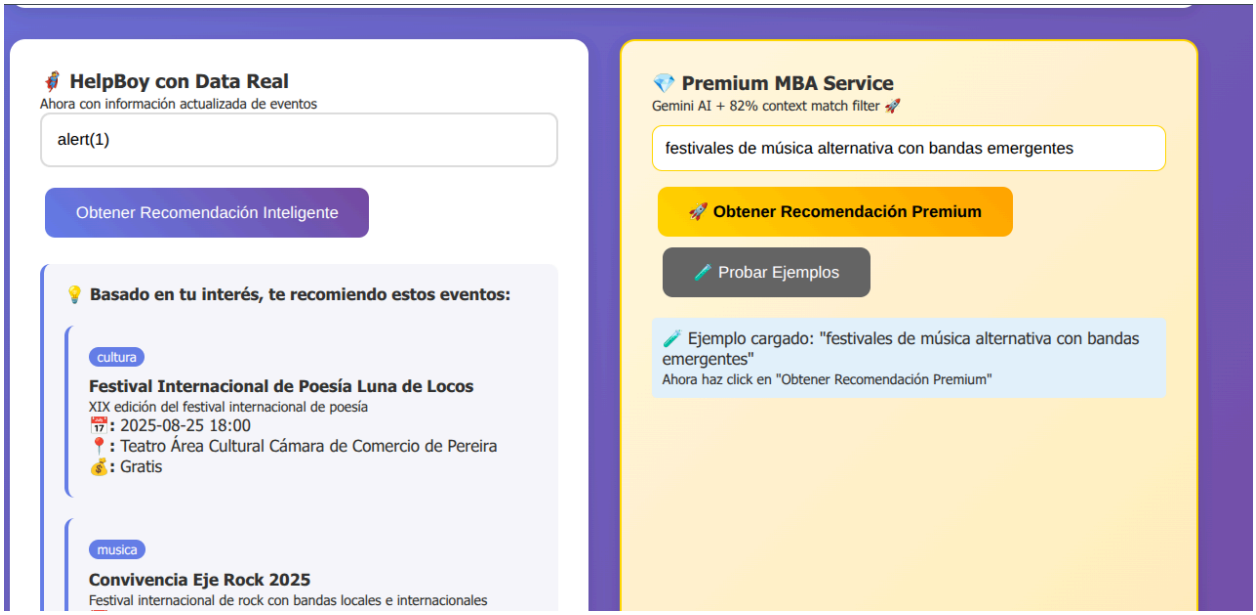
* Inspección profunda de paquetes (DPI).
* Reemplazo de sistemas IAM, ACLs o firewalls.
* Ambientes productivos de alta criticidad.

3. Componentes del Sistema

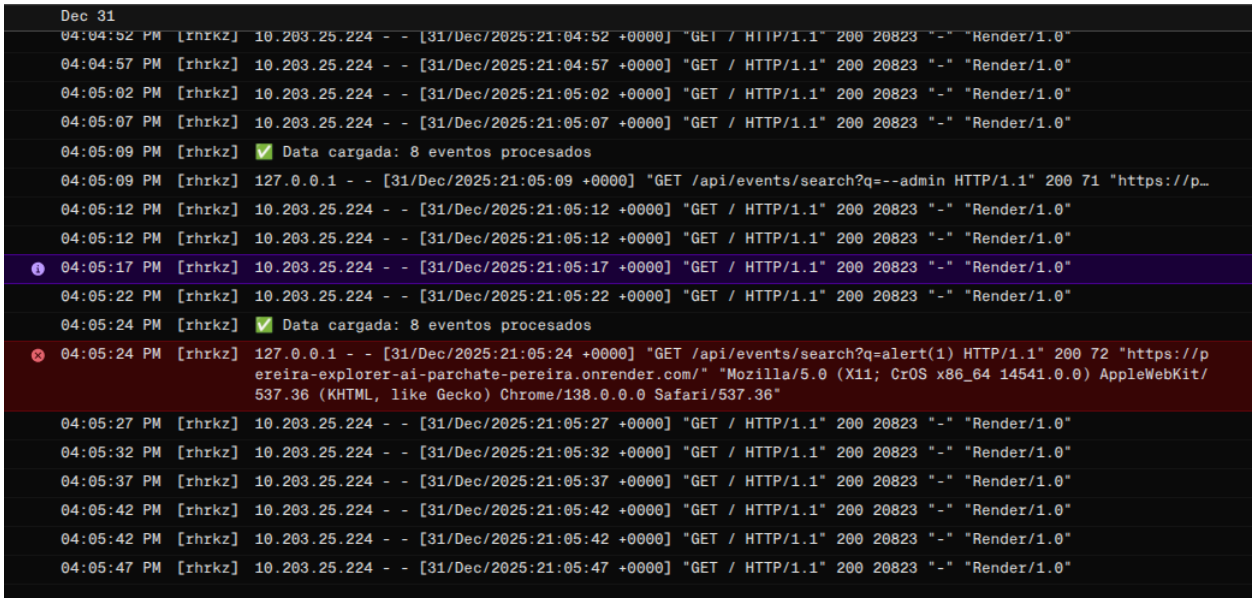
3.1 Aplicación (Parchate Pereira)

https://github.com/SPotes22/Pereira-Explorer-AI-Parchate-Pereira





https://pereira-explorer-ai-parchate-pereira.onrender.com/



La aplicación cliente es responsable de:

* Generar eventos de negocio.
* Consumir decisiones de validación emitidas por OctoMatrix.
* No contiene lógica de seguridad predictiva.

Archivos de referencia:

* `app.py`
* `brain.py`

3.2 OctoMatrix Core

OctoMatrix es el componente encargado de:

* Cargar el modelo entrenado (`.pkl`).
* Recibir eventos normalizados.
* Evaluar patrones y emitir decisiones (válido / sospechoso).

Archivo de referencia:

* `octomatrix_poc_moe_owasp.py`
https://www.kaggle.com/code/santiagopotes/octomatrix-poc-moe-owasp/log

---

3.3 Modelo de Machine Learning

El modelo entrenado:

* Fue generado offline.
* Presenta una precisión superior al 82%.
* No se entrena en tiempo real.

El modelo se carga en memoria al iniciar OctoMatrix.

---

4. Flujo General de Implementación

1. La aplicación genera un evento.
2. El evento se normaliza (estructura JSON o dict).
3. El evento se envía a OctoMatrix.
4. OctoMatrix evalúa el evento usando el modelo.
5. Se retorna una decisión.
6. La aplicación actúa según la decisión.

---

5. Requisitos del Entorno

5.1 Software

* Python 3.9+
* Librerías estándar de ML (según entrenamiento previo)
* Entorno virtual recomendado

5.2 Hardware

* CPU estándar (no GPU requerida)
* Mínimo 4 GB RAM

---

6. Preparación del Entorno

1. Crear entorno virtual.
2. Instalar dependencias.
3. Ubicar el archivo `.pkl` en la ruta definida.
4. Verificar permisos de lectura.

---

7. Inicialización de OctoMatrix

Al iniciar OctoMatrix:

* Se carga el modelo en memoria.
* Se validan estructuras internas.
* El sistema queda en modo escucha.

Recomendación: inicializar una sola vez por proceso.

---

8. Integración con la Aplicación

La aplicación debe:

* Invocar a OctoMatrix como servicio o módulo.
* Enviar eventos estructurados.
* Interpretar la respuesta sin lógica adicional.

OctoMatrix **no decide acciones finales**, solo emite evaluaciones.

---

9. Manejo de Resultados

Las respuestas posibles incluyen:

* Evento válido.
* Evento sospechoso.

La aplicación decide:

* Registrar.
* Bloquear.
* Alertar.

---

10. Buenas Prácticas

* Mantener el modelo desacoplado del código.
* No modificar el `.pkl` en caliente.
* Registrar decisiones para análisis posterior.
* No asumir falsos positivos como errores.

---

11. Limitaciones Conocidas

* Dependencia de calidad del entrenamiento.
* No detección de ataques completamente nuevos.
* No reemplaza validaciones determinísticas.

---

12. Extensiones Futuras

* Correlación con identidad de red (MAC/IP).
* Integración con Kafka.
* Uso de métricas SNMP.
* Soporte multi-modelo.

---

13. Consideraciones Académicas

Este manual está diseñado para facilitar:

* Reproducción del experimento.
* Evaluación técnica en laboratorio.
* Discusión arquitectónica.

No pretende ser una guía de despliegue comercial.

---

14. Cierre

OctoMatrix debe entenderse como una **capa experimental de validación**, útil para explorar nuevas formas de correlacionar comportamiento, identidad y eventos sin comprometer la arquitectura existente.