Securing AIML Systems in the Age of Information Warfare, Susanna Cox (2022). Un estudio sobre el artículo.

Este documento ha sido analizado desde mis limitaciones técnicas, es una interpretación personal sujeta a posibles errores. Recomiendo que cada cual aborde el estudio de la fuente original y del documento al que hace referencia la autora, Measuring Cybersecurity and Cyber Resiliency, RAND (2020)

Aitor Saiz Lasheras, 25/05/2026

Resumen

Este documento sintetiza el análisis de Susanna Cox (2022) en *Securing AIML Systems in the Age of Information Warfare*. Shoshana (también llamada Disesdi) aborda la necesidad de proteger sistemas de inteligencia artificial y de aprendizaje automático (AIML) en entornos de guerra de información (Information Warfare, IW), un dominio con creciente regulación (ej. ley de reporte de incidentes cibernéticos de EE.UU. de marzo de 2022) y riesgos operacionales demostrados.

El texto identifica vulnerabilidades estructurales de los sistemas AIML derivadas de su dependencia de datos, bucles de retroalimentación algorítmica, reentrenamiento continuo sin validación robusta, y susceptibilidad a envenenamiento de datos y sesgo malicioso. Estas vulnerabilidades son explotables por actores que no requieren el acceso a los modelos, solo a los conjuntos de datos (datasets) —públicos en muchos casos—, pudiendo co-crear los datos de entrenamiento y apretar el bucle OODA (observar, orientarse, decidir, actuar) en favor del atacante. (del verdadero Bucle OODA hablaré en otro trabajo).

Frente a ello, Cox propone un proceso de mitigación operacionalizable en pipelines MLOps existentes, que incluye: análisis de modos de fallo (FMEA), pruebas adversariales, documentación obligatoria (*datasheets* y *model cards*), generación de un portafolio de activos de seguridad (vectores de ataque, triggers, umbrales), integración de dichos triggers y umbrales en monitorización continua y entrenamiento continuo, automatización de alertas con revisión humana solo al superar umbrales, y un índice de madurez dual con preguntas específicas para AIML.

1. Importancia del texto y por qué debe estudiarse

El texto de Cox es importante porque aborda la necesidad de asegurar sistemas AIML en entornos de guerra de información (IW), un área con creciente atención regulatoria y riesgos operacionales concretos. En marzo de 2022, Estados Unidos aprobó legislación bipartidista que exige el reporte de incidentes cibernéticos para organizaciones de infraestructura crítica, y el documento proporciona un marco práctico para ayudar al cumplimiento temprano de estos requisitos.

Cox identifica vulnerabilidades específicas de los sistemas AIML. Son sistemas de procesamiento de información cuya propagación en casi todos los aspectos de la sociedad (militar, transporte, banca, finanzas) genera consecuencias potencialmente graves ante fallos. La manipulación masiva de

algoritmos de redes sociales es posible y conveniente para actores no estatales; el uso de bots y *surrogates* (a veces llamados "actores naranja") ha demostrado efectividad para interrumpir redes a escala. La proliferación de sistemas AIML tanto ofensivos como defensivos crea un entorno de bajo riesgo y alta recompensa para posibles atacantes.

El documento señala que la amplificación algorítmica, junto con la falta de transparencia de las plataformas con respecto a decisiones algorítmicas en el diseño y el uso, es explotable por actores maléficos con potencial para disruptir normas democráticas. Existen crecientes llamamientos desde diversos sectores a legislar nuevos marcos para las plataformas AIML y a responsabilizar legalmente a las organizaciones propietarias de dichas plataformas por los mensajes que se propagan a través de las mismas.

Sin embargo, Cox advierte que los marcos de ética y auditoría existentes son difíciles de implementar. O son demasiado generales (y por tanto son difíciles de llevar a la práctica) o demasiado específicos (no son generalizables). Los profesionales indican dificultad con la operacionalización del desarrollo consciente de sesgos debido al gran número de marcos de alto nivel y la falta de guías de implementación. La implementación nunca es gratuita, y el costo de la experimentación debe ser factorizado. Además existe el riesgo del *ethics-washing* si los marcos se implementan mal. El Pew Research Center advirtió que una implementación industrial robusta de reducción de daños podría tomar años.

Disesdi propone un enfoque basado en MLOps (*machine learning operations*) porque muchos profesionales son lentos en adoptar estas prácticas y continúan con experimentación *ad hoc* difícil de escalar. Un informe de Deloitte (Ronanki & Davenport, 2017) listó problemas de integración como cuellos de botella principales, y un estudio de McKinsey de 2020 (Balakrishnan et al.) encontró que los procesos de desarrollo estandarizados y escalables eran un diferenciador clave en organizaciones AIML de alto rendimiento. La escala y velocidad de despliegue de sistemas AIML modernos, combinada con su ubicuidad en casi todas las capas de la pila sociotécnica, indica la urgencia en desarrollar métricas de evaluación de preparación cibernética escalables e inteligentes en MLOps.

El marco presentado por Cox adapta un informe de RAND Project AIR FORCE (PAF), originalmente creado para sistemas de armas de la Fuerza Aérea de EE.UU. Cox argumenta que los sistemas de IA, incluso en aplicaciones no militares, están tan incrustados en sistemas sociotécnicos críticos que evaluar su resiliencia a vectores de ataque de guerra de información es crucial para la seguridad nacional. La guerra de información puede entenderse como una guerra de datos; por tanto, los profesionales de la "IA" deben estar cada vez más alertas en los entornos de amenaza IW elevada.

Finalmente, Shoshana señala que, a pesar del interés organizacional, social y gubernamental en la reforma ética de AIML, la implementación industrial robusta podría tardar años. Por ello, el texto proporciona un *blueprint* práctico y accionable para que profesionales y organizaciones evalúen y aumenten la resiliencia de sus sistemas frente a vectores de ataque IW, integrando mejores prácticas MLOps en entornos de despliegue rápido.

Por estas razones, el texto debe ser estudiado por profesionales de AIML, equipos de seguridad, gestores de riesgos, responsables de políticas regulatorias y cualquier organización que opere sistemas AIML en entornos donde la integridad de los datos y la resiliencia a la manipulación sean críticas.

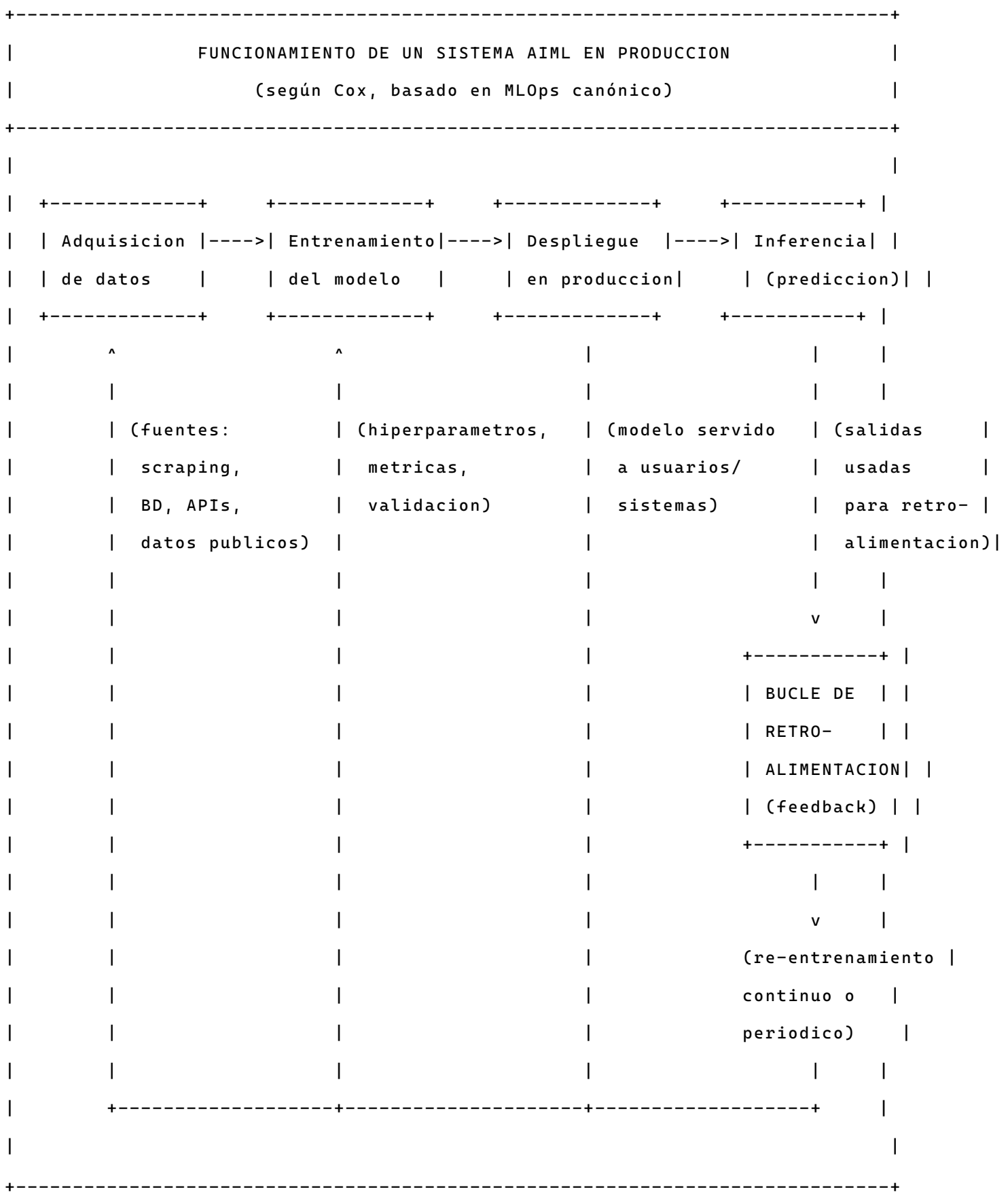
2. Funcionamiento de los sistemas AIML

Para comprender las vulnerabilidades y el proceso de mitigación propuesto por Cox, es necesario describir primero cómo opera un sistema AIML (*artificial intelligence/machine learning*) en un entorno de producción típico. Cox define los sistemas AIML como sistemas de procesamiento de información. Su "inteligencia" no es más que el grado en que han "aprendido" (en realidad ni son inteligentes, ni aprenden, ese lengüaje antropizado es erróneo y obedece a intereses corporativistas) —es decir, han sido entrenados— a partir de un subconjunto de datos. Su "inteligencia" también depende del grado en que ese subconjunto de datos refleje la realidad del mundo. Sin datos de calidad, no existe posibilidad de un AIML óptimo.

El ciclo de vida canónico de un sistema AIML en producción consta de las siguientes etapas:

1. Adquisición de datos: Los datos se obtienen de diversas fuentes: *scraping web*, bases de datos internas, APIs, conjuntos de datos públicos, o datos generados por usuarios. Estos datos pueden ser estáticos o dinámicos (de actualización continua).
2. *Entrenamiento* del modelo: Sobre los datos adquiridos, se *entrena* un modelo utilizando algoritmos de ML, ajustando hiperparámetros, validando con conjuntos de retención, y registrando métricas de rendimiento.
3. Despliegue en producción: El modelo entrenado se sirve a través de una API, incrustado en una aplicación, o integrado en un sistema automatizado.
4. Inferencia (predicción): El modelo recibe entradas (nuevos datos o consultas) y produce salidas (clasificaciones, recomendaciones, detecciones, etc.).
5. Bucle de retroalimentación (*feedback loop*): Las salidas del modelo, las interacciones de los usuarios, o los resultados observados se reintroducen como datos de *entrenamiento* para futuras iteraciones del modelo. Esto cierra el ciclo y permite el *reentrenamiento* continuo o periódico.

El diagrama ASCII de la página siguiente muestra esta estructura funcional


FUNCIONAMIENTO DE UN SISTEMA AIML EN PRODUCCION
(según Cox, basado en MLOps canónico)
Adquisicion de datos
Entrenamiento del modelo
Despliegue en produccion
Inferencia (prediccion)
(fuentes: scraping, BD, APIs, datos publicos)
(hiperparametros, metricas, validacion)
(modelo servido a usuarios/ sistemas)
(salidas usadas para retro- alimentacion)
BUCLE DE RETRO- ALIMENTACION (feedback)
(re-entrenamiento continuo o periodico)

Aspectos adicionales señalados por Cox sobre este funcionamiento:

> Los sistemas AIML no son solo modelos; son *sistemas sociotécnicos* que incluyen procesos de recolección, almacenamiento, preprocesado, versionado, monitorización y gobernanza.
>
> El bucle de retroalimentación no es opcional en muchos sistemas prácticos (ej. motores de recomendación, sistemas de seguridad adaptativa). Sin embargo, es precisamente este bucle el que introduce las vulnerabilidades más críticas, como se verá en la Sección 3.
>
> La escalabilidad y la automatización, promovidas por las prácticas MLOps, amplifican tanto los beneficios como los riesgos: un error o un ataque puede propagarse rápidamente a través de múltiples iteraciones de reentrenamiento sin intervención humana.

Con esta base funcional, la siguiente sección desglosará las vulnerabilidades estructurales que surgen en cada una de estas etapas, según el análisis de Disesdi.

3. Vulnerabilidades estructurales por etapa del proceso AIML

Cox identifica vulnerabilidades específicas en cada etapa del ciclo de vida de un sistema AIML. Estas vulnerabilidades no son accidentales, sino estructurales: derivan de la dependencia de datos, de los bucles de retroalimentación, y de la falta de documentación y procesos formalizados. A continuación se enumeran por etapa, tal como aparecen en el texto original.

<u>Etapa 1: Adquisición de datos</u>

Los sistemas AIML son tan "inteligentes" como los datos con los que se entrenan. Si los datasets se crean a partir de fuentes públicas no curadas, especialmente mediante *scraping* masivo, el acceso de *Red* (atacante) está de facto y definitivamente garantizado. Esto aplica tanto a sistemas que usan *scraping* único como a aquellos que usan *scraping* continuo. En el *scraping* continuo, la naturaleza dinámica de los datos amplifica los vectores de ataque. Además, si el sistema interactúa con los datos que *scrapea*, el potencial de amplificación por bucles de retroalimentación aumenta.

Vulnerabilidades concretas:

> *Red* puede co-crear los datos de entrenamiento sin necesidad de acceder al modelo.
>
> La falta de *datasheets* (hojas de datos) impide documentar la motivación, composición, preprocesado, usos, distribución y mantenimiento de los conjuntos de datos, lo que oculta los posibles vectores de ataque.

<u>Etapa 2: Entrenamiento del modelo</u>

En esta etapa, *Red* puede ejecutar ataques de envenenamiento de datos (*data poisoning*), insertando muestras maliciosas en el conjunto de entrenamiento. Esto es especialmente viable en modelos de lenguaje grandes y sistemas de NLP, como se ha demostrado repetidamente en la literatura. El sesgo malicioso no es solo una cuestión ética, sino un riesgo de seguridad crítico: un atacante puede

entrenar gradualmente un sistema de detección de anomalías para ignorar actividad maliciosa. Los modelos pre-entrenados en datos públicos heredan las vulnerabilidades de sus conjuntos de origen.

Vulnerabilidades concretas:

- Sin un *Análisis de Modos de Falla y Efectos (FMEA)*, los posibles puntos de fallo no se documentan sistemáticamente.
- Sin pruebas adversariales en la fase de entrenamiento, no se identifican las rutas de ataque antes del despliegue.

Etapa 3: Despliegue en producción

Una vez desplegado, *Red* puede sondear el sistema para inferir conocimiento sobre el modelo (*property inference*) sin necesidad de acceder al código propietario. La manipulación de algoritmos sociales (por ejemplo, influir en el ranking de feeds) permite a *Red* afectar las salidas sin acceso directo al modelo, solo interactuando con el sistema como un usuario más. La falta de *model cards* impide documentar el comportamiento esperado del modelo para diferentes grupos poblacionales, las asunciones de desarrollo, y las evaluaciones de rendimiento. Sin monitorización continua con *triggers* (avisos, disparadores, alarmas.. para entendernos) específicos, *Red* puede operar sin ser detectado.

Vulnerabilidades concretas:

- *Red* puede realizar ataques de *inference* y *model stealing* mediante consultas repetidas.
- La falta de transparencia algorítmica facilita la explotación de la amplificación algorítmica en plataformas sociales.

Etapa 4: Bucle de retroalimentación (*feedback loop*)

Cox identifica el bucle de retroalimentación como la vulnerabilidad más crítica. Cuando el sistema se *reentrena* con datos generados por sus propias salidas o por interacciones de usuarios (incluyendo a *Red*), *Red* se convierte en co-creador de los datos de entrenamiento. Esto le otorga acceso casi directo a los modelos de *Blue*. El bucle aprieta el OODA loop de *Red* (observar, orientarse, decidir, actuar), permitiéndole operar más rápido que *Blue*. Las campañas de desinformación se auto-refuerzan: los participantes involuntarios amplifican el mensaje, haciendo imposible la atribución. El sesgo se amplifica con el tiempo (*bias amplification*).

Vulnerabilidades concretas:

- *Red* puede inyectar datos maliciosos que se retroalimentan y se vuelven parte del entrenamiento futuro.
- La velocidad del bucle puede hacer que *Blue* pierda el control narrativo antes de poder responder.

Reentrenamiento continuo (transversal)

Si no hay validación robusta de datos en cada ciclo de *reentrenamiento*, la deriva de datos o deriva de concepto puede introducir sesgo malicioso lentamente (*slow-shift attack*). Modelos *reentrenados* sobre datos contaminados propagan la vulnerabilidad a través de versiones sucesivas. Sin análisis de linaje (*lineage analysis*), no es posible rastrear un modelo problemático hasta su dataset original ni hasta los metadatos de entrenamiento, lo que dificulta gravemente la respuesta a incidentes.

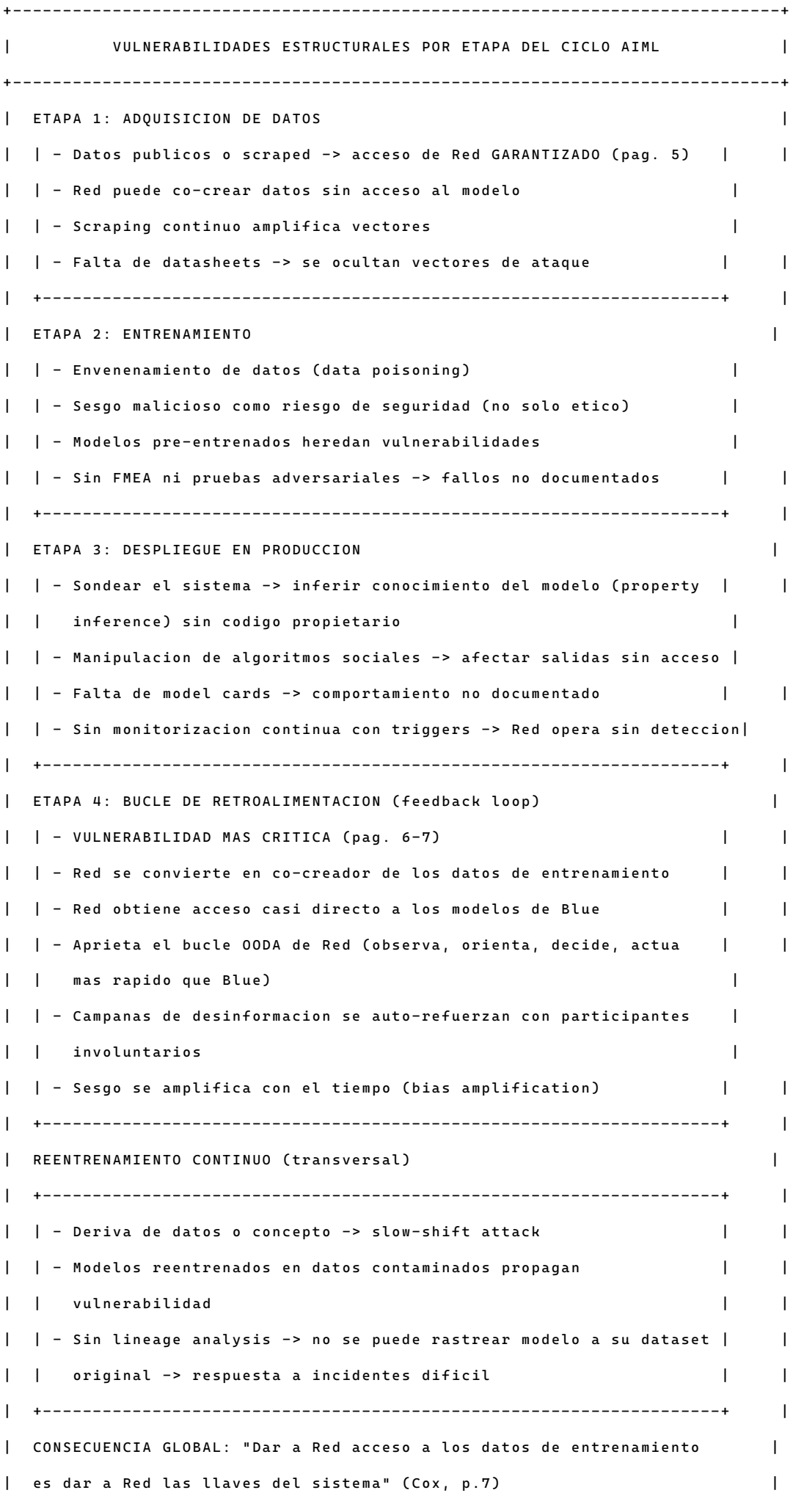
Consecuencia estructural global

Cox resume en la página 7:

> *"Dar a Red acceso a los datos de entrenamiento es dar a* Red *las llaves del sistema."*

El diagrama ASCII de la página siguiente integra estas vulnerabilidades en el esquema funcional de la Sección 2.

Esta tabla de vulnerabilidades es la base sobre la cual Cox construye su proceso de mitigación. La Sección 4 describirá dicho proceso en detalle, mostrando cómo cada vulnerabilidad es abordada por componentes específicos del marco (FMEA, pruebas adversariales, datasheets, model cards, portafolio de seguridad, triggers, umbrales, revisión humana, etc.).


VULNERABILIDADES ESTRUCTURALES POR ETAPA DEL CICLO AIML
ETAPA 1: ADQUISICION DE DATOS
- Datos publicos o scraped -> acceso de Red GARANTIZADO (pag. 5)
- Red puede co-crear datos sin acceso al modelo
- Scraping continuo amplifica vectores
- Falta de datasheets -> se ocultan vectores de ataque
ETAPA 2: ENTRENAMIENTO
- Envenenamiento de datos (data poisoning)
- Sesgo malicioso como riesgo de seguridad (no solo etico)
- Modelos pre-entrenados heredan vulnerabilidades
- Sin FMEA ni pruebas adversariales -> fallos no documentados
ETAPA 3: DESPLIEGUE EN PRODUCCION
- Sondear el sistema -> inferir conocimiento del modelo (property inference) sin codigo propietario
- Manipulacion de algoritmos sociales -> afectar salidas sin acceso
- Falta de model cards -> comportamiento no documentado
- Sin monitorizacion continua con triggers -> Red opera sin deteccion
ETAPA 4: BUCLE DE RETROALIMENTACION (feedback loop)
- VULNERABILIDAD MAS CRITICA (pag. 6-7)
- Red se convierte en co-creador de los datos de entrenamiento
- Red obtiene acceso casi directo a los modelos de Blue
- Aprieta el bucle OODA de Red (observa, orienta, decide, actua mas rapido que Blue)
- Campanas de desinformacion se auto-refuerzan con participantes involuntarios
- Sesgo se amplifica con el tiempo (bias amplification)
REENTRENAMIENTO CONTINUO (transversal)
- Deriva de datos o concepto -> slow-shift attack
- Modelos reentrenados en datos contaminados propagan vulnerabilidad
- Sin lineage analysis -> no se puede rastrear modelo a su dataset original -> respuesta a incidentes dificil
CONSECUENCIA GLOBAL: "Dar a Red acceso a los datos de entrenamiento es dar a Red las llaves del sistema" (Cox, p.7)

4. El proceso de mitigación de riesgos propuesto por Cox

Cox propone un proceso de mitigación estructurado en torno a seis componentes principales, diseñados para ser operacionalizados dentro de *pipelines* MLOps existentes. El objetivo es interrumpir los vectores de ataque de *Red* (acceso, conocimiento, capacidad, impacto) mediante la documentación obligatoria, las pruebas sistemáticas, la automatización de alertas y la revisión humana solo cuando se superen umbrales predefinidos. A continuación se describen cada uno de estos componentes en el orden lógico de implementación, según el texto original.

Componente 1: Descomposición de vectores de ataque mediante lógica booleana

Cox adopta el marco de *RAND Project AIR FORCE* (Snyder et al. 2020) que define cuatro atributos que *Red* debe cumplir para que un ataque tenga éxito: acceso, conocimiento, capacidad e impacto. Estos cuatro atributos están conectados por operadores lógicos *Y (AND)*: *Red* debe lograr todos ellos. Dentro de cada categoría, *Red* tiene múltiples opciones conectadas por operadores *O (OR)*. Por ejemplo, para lograr acceso, *Red* puede usar el acceso a la cadena de suministro, la amenaza interna, o la explotación de datos públicos. Para lograr conocimiento, *Red* puede usar la inteligencia de fuentes abiertas (OSINT) o la exfiltración mediante espionaje, pero con una restricción adicional: el conocimiento debe ser actual (*currency*), es decir, *Red* debe actuar antes de que *Blue* modifique el sistema.

Aplicado a AIML, Cox señala que:

> Para el acceso: *Red* no necesita acceder a los modelos, solo a los datasets. Si los datasets son públicos, el acceso está garantizado.
>
> Para el conocimiento: *Red* puede sondear sistemas públicos para inferir comportamiento del modelo sin código propietario.
>
> Para la capacidad: La IA disponible para *Red* (ej. GANs para generar perfiles falsos, sistemas de gestión de contenido masivo) amplifica sus recursos.
>
> Para el impacto: Los bucles de retroalimentación y la desinformación amplificada pueden tener efectos desproporcionados.

La mitigación consiste en que *Blue* debe aumentar la dificultad de que *Red* cumpla uno o más de estos cuatro requisitos. Por ejemplo, cambiando frecuentemente detalles del sistema para que el conocimiento de *Red* quede obsoleto (manteniéndose dentro del bucle OODA de *Red*), o limitando el acceso público a los datos de entrenamiento.

Componente 2: Producción de un portafolio de activos de seguridad durante el prototipado

En la fase de prototipado del modelo, Cox exige un proceso de desarrollo en paralelo que genera tres artefactos obligatorios, denominados colectivamente *portafolio de seguridad IW*:

1. Conjunto de vectores de ataque potenciales documentados a partir de un *Análisis de Modos de Falla y Efectos (FMEA)*. El FMEA es una técnica de ingeniería de seguridad que examina sistemáticamente un diseño o tecnología para identificar fallos previsibles. Incluye la revisión de literatura, las entrevistas con las partes interesadas y la recopilación de la documentación técnica. Para AIML, el FMEA debe cubrir los riesgos de envenenamiento de datos, los ataques de inferencia, los bucles de retroalimentación, la deriva de datos, y el sesgo malicioso.
2. Un conjunto formalizado de *triggers* (disparadores) para la revisión del rendimiento del modelo, derivados del FMEA y de las pruebas adversariales. Estos triggers son indicadores específicos de posibles ataques IW, por ejemplo: cambios inesperados en la precisión de subgrupos, aumento de la varianza en predicciones, o correlaciones anómalas entre características [estos ejemplos no son de Cox, son del autor de este estudio].
3. Un conjunto de puntos umbral de comparación operacionalizables para el proceso de monitorización automática. Estos umbrales actúan como un sistema redundante al anterior: definen los límites cuantitativos (ej. desviación máxima permitida en la distribución de una característica, la diferencia máxima en la tasa de aciertos por grupo demográfico). Límites que, si se superan, activan una revisión humana.

Además del portafolio, Cox considera indispensable la elaboración de dos documentos complementarios, incluso si no se realiza una auditoría completa:

> *Datasheets* (Gebru et al. 2018): Documentación para cada conjunto de datos que responde a preguntas sobre motivación, composición, preprocesado, etiquetado, usos previstos, distribución y plan de mantenimiento.
>
> *Model cards* (Mitchell et al. 2019): Documentación para cada modelo entrenado que describe cómo se construyó, qué asunciones se hicieron, qué comportamiento se espera de los diferentes grupos culturales o demográficos, y las evaluaciones de rendimiento por grupos.

Cox afirma:

> *"Los sistemas AIML no deben considerarse debidamente documentados para fines de seguridad IW sin al menos FMEA, pruebas adversariales, model cards y datasheets para los datos de entrenamiento."*

Componente 3: Integración de *triggers* y umbrales en *pipelines* de monitorización y entrenamiento continuo

Cox aprovecha las arquitecturas MLOps estándar para insertar los activos de seguridad sin necesidad de rediseñar los *pipelines*. En concreto:

En el *pipeline* de monitorización continua:

> El sistema carga los *logs* de inferencia, el portafolio de seguridad, las estadísticas de referencia del modelo y el esquema de referencia.

Un motor de monitorización compara las métricas de rendimiento del modelo en tiempo real (o por lotes) contra los *triggers* del portafolio.

Si se cumple cualquier condición de *trigger* (ej. caída repentina de precisión en un subgrupo), se declara un *incidente IW potencial* y se inicia una revisión humana automáticamente, con notificaciones a los responsables.

Si no se cumplen los *triggers*, el monitor continúa con comprobaciones normales (sesgos de esquema, deriva de distribución, deriva de concepto, decaimiento del modelo). Si estas comprobaciones normales indican necesidad de reentrenamiento, se inicia el *pipeline* de entrenamiento continuo.

En el *pipeline* de entrenamiento continuo:

Cuando se solicita un reentrenamiento (por programación, evento o umbral), el sistema extrae el modelo actual, el portafolio de seguridad, los datasets y los umbrales de comparación.

En la etapa de validación de datos (antes del entrenamiento), los datos nuevos se evalúan contra los umbrales (ej. distribución de características, rango de valores, frecuencias de categorías).

Si los datos superan algún umbral (es decir, se desvían más de lo permitido), el sistema alerta de un posible *incidente IW* y activa la revisión humana, sin proceder al entrenamiento.

Si los datos no superan los umbrales, el entrenamiento continúa normalmente, y se añade documentación del evento (fecha, métricas, versión del modelo) a los metadatos del modelo para su trazabilidad.

Componente 4: Flujo de trabajo de análisis de incidentes con revisión humana

Cuando se activa una revisión humana (desde la monitorización o desdeel entrenamiento), el proceso sigue estos pasos, según Cox:

1. El equipo de seguridad o los ingenieros responsables reúnen:

   Los *Datasheets* de los conjuntos de datos involucrados.

   El Análisis de linaje (*lineage analysis*) del modelo: si ha sido reentrenado, se rastrea hasta su dataset original y sus versiones intermedias, utilizando los metadatos almacenados.

   El Portafolio de seguridad IW (FMEA, resultados de pruebas adversariales, *triggers*, umbrales).

   La Documentación de auditoría SMACTR, si se ha realizado previamente.

2. Se analiza si el incidente corresponde a un ataque IW real o a una falsa alarma. En caso afirmativo, se identifica el vector de ataque, el alcance, y los modelos o datasets afectados.

3. Se actualiza el portafolio de seguridad con los hallazgos (nuevos vectores, *triggers* adicionales, ajuste de los umbrales).
4. Se documenta el incidente en un formato apto para el cumplimiento normativo (ej. reportes a agencias gubernamentales, según la legislación de incidentes cibernéticos). Cox señala que esta documentación es un artefacto producido por el proceso, no un añadido externo.
5. Dependiendo de la gravedad, se puede revertir el modelo a una versión anterior (*baseline trusted state*) si se ha definido, o parchear el dataset y reentrenar.

Componente 5: Pruebas adversariales continuas y actualización del FMEA

Las pruebas adversariales no son un evento único. Cox especifica que:

> Comienzan en la fase de prototipado, utilizando los riesgos documentados en el FMEA para diseñar entradas maliciosas (no solo manipulaciones de píxeles, sino cualquier entrada que pueda provocar salidas no deseadas, como cambios en el orden de palabras, inserciones de texto, etc.).
>
> Continúan durante todo el ciclo de vida del sistema, especialmente después de cada actualización del modelo o dataset.
>
> Los resultados de cada ronda de pruebas adversariales se utilizan para actualizar el FMEA, y los cambios en los riesgos se evalúan.
>
> La frecuencia de las pruebas depende del nivel de riesgo asignado en el FMEA: los sistemas con mayor superficie de ataque, los bucles de retroalimentación, o las consecuencias sociales amplificadas requieren pruebas más frecuentes.

Componente 6: Auditoría SMACTR como nivel de madurez superior

Cox recomienda, para organizaciones con alta madurez, la adopción del marco de auditoría SMACTR (Raji et al. 2020), diseñado específicamente para sistemas AIML en producción. SMACTR incluye pasos como *scoping*, mapeo de sistemas, recolección de evidencia, pruebas de sesgo, y generación de reportes. Sin embargo, Cox establece claramente que la ausencia de una auditoría SMACTR no excusa la falta de los componentes mínimos (FMEA, pruebas adversariales, *datasheets*, *model cards*, portafolio). La auditoría SMACTR se suma a estos, no los reemplaza.

Componente adicional: Índice de madurez dual

Para facilitar la auto-evaluación, Cox propone un índice de madurez de 5 niveles (1=mayor madurez, 5=menor madurez) con las características generales de la resiliencia cibernética y las características específicas para AIML. Las preguntas clave para evaluar la madurez incluyen (agrupadas en 8 grupos):

> Grupo 1 (baseline): ¿Existen *datasheets* y *model cards*? ¿Hay *pipelines* formalizados para *tracking* de metadatos que soporten el análisis de linaje?

Grupo 2 (monitorización): ¿Hay monitorización continua con *triggers* de seguridad y hay umbrales en la validación de datos?

Grupo 3 (resiliencia): ¿Se han identificado métodos resilientes para volver al estado base? ¿Hay un flujo de respuesta a incidentes formalizado?

Grupo 4 (OODA): ¿Qué consideraciones especiales de OODA existen? ¿Hay bucles de retroalimentación que puedan *apretar* el bucle de *Red*? ¿Cómo se han abordado?

Grupo 5 (recuperación): ¿Cuáles son las consecuencias de perder el control de los datos? ¿Se amplifican con el tiempo? ¿Es aceptable el tiempo de recuperación?

Grupo 6 (implementación): ¿Están implementados sistemas formalizados de prototipado, desarrollo, despliegue y monitorización?

Grupo 7 (ejercitación): ¿Se han probado y ejercitado los flujos de trabajo y el portafolio en el entorno de producción?

Grupo 8 (adecuación): ¿Se ha encontrado que son adecuados?

Cox concluye que la integración de estos componentes en un *pipeline* MLOps permite la automatización de la mayoría de las comprobaciones de seguridad IW, reutilizando artefactos creados para cumplimiento normativo y ético, y minimizando la necesidad de costosas revisiones humanas. Solo cuando los *triggers* o los umbrales se superen, se activará el recurso humano especializado.

5. En la páginas siguientes, un diagrama ASCII del proceso de mitigación

A continuación se muestra el diagrama completo del proceso de mitigación propuesto por Cox, que integra los componentes descritos en la Sección 4. El flujo comienza en la fase de prototipado (desarrollo en paralelo), continúa con la monitorización continua y el entrenamiento continuo, y finaliza en la revisión humana cuando se superan *triggers* o umbrales. En rojo, modificaciones o aportaciones con lo que el autor de este estudio considera lógico.

Notas sobre el diagrama:

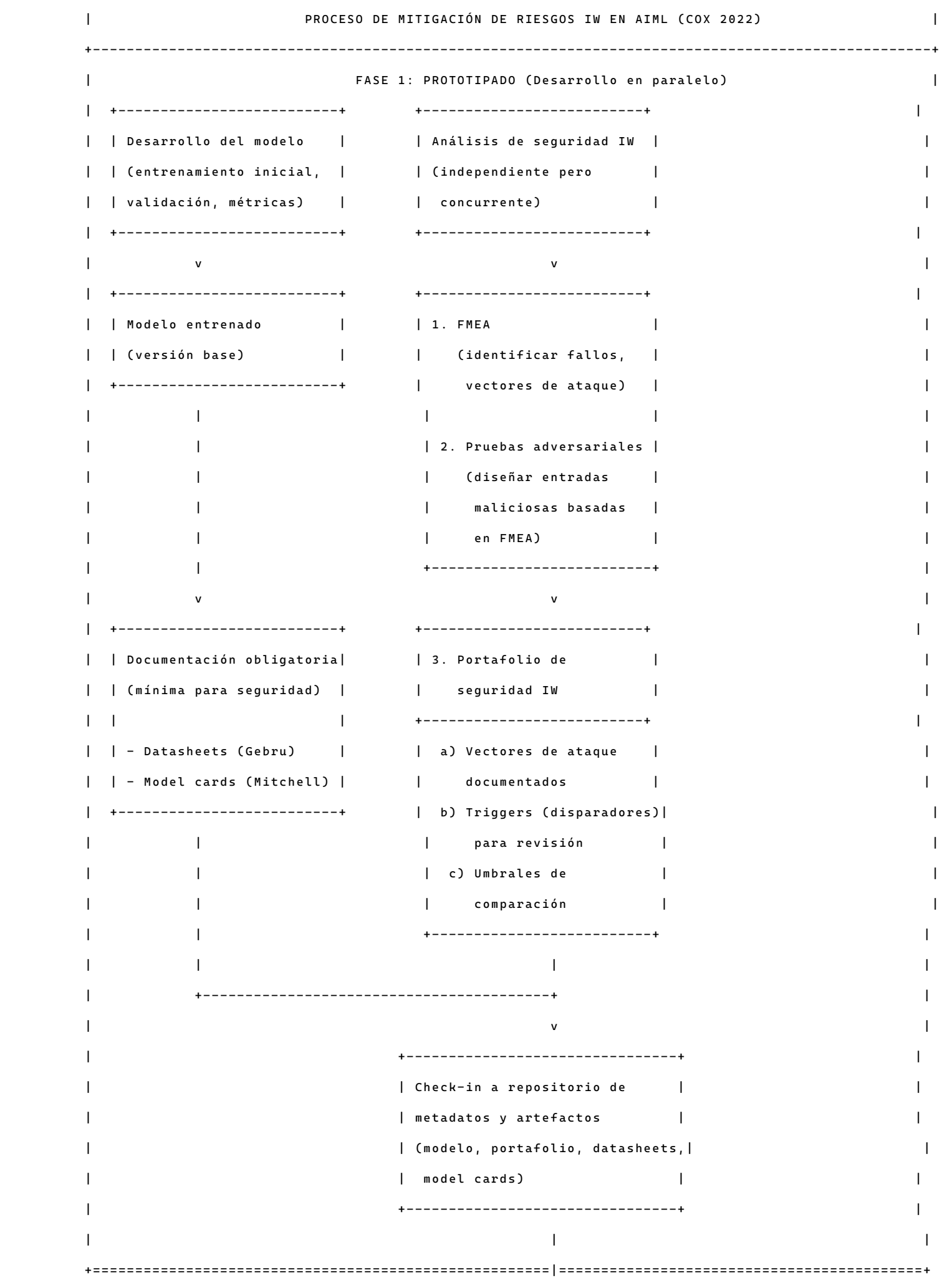
Las fases 2 y 3 se ejecutan de forma continua y autónoma. La intervención humana solo ocurre en la Fase 4.

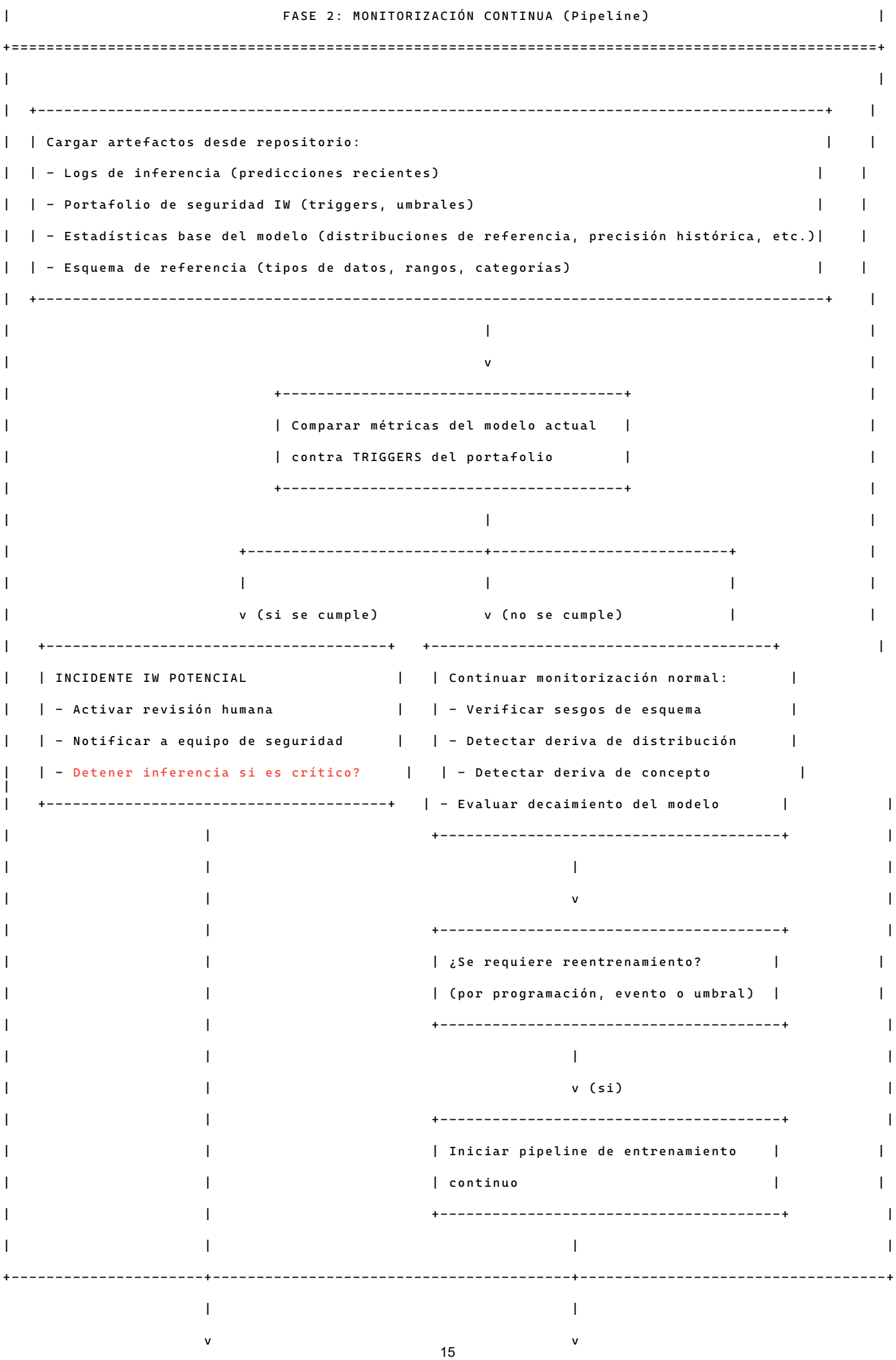
Los *triggers* (Fase 2) son condiciones cualitativas o cuantitativas derivadas del FMEA (ej. "caída de precisión en subgrupo X superior al 5% en 1 hora").

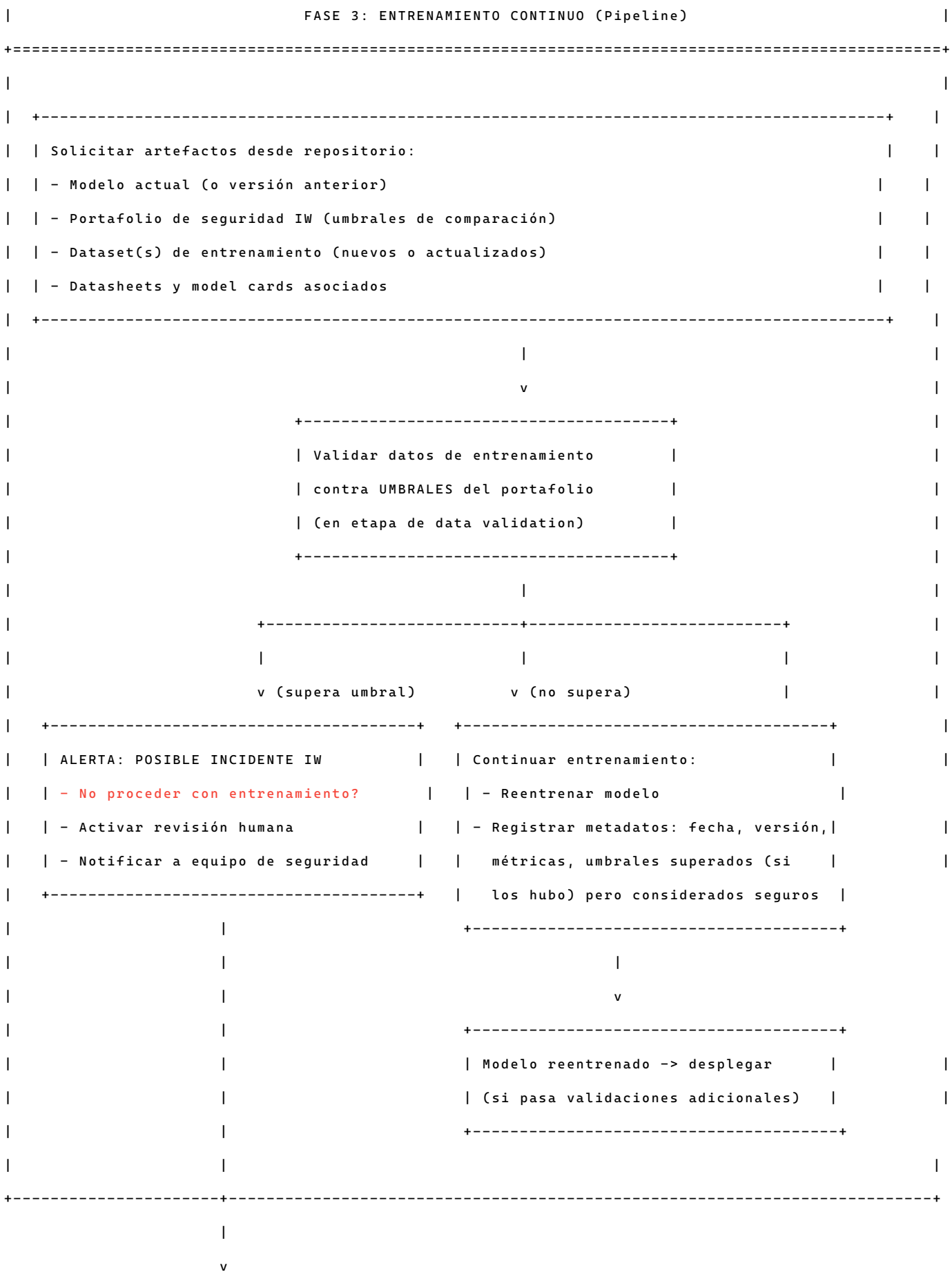
Los *umbrales* (Fase 3) son límites cuantitativos para la validación de datos (ej. "desviación máxima de la media de la característica Y: ±2 desviaciones estándar").

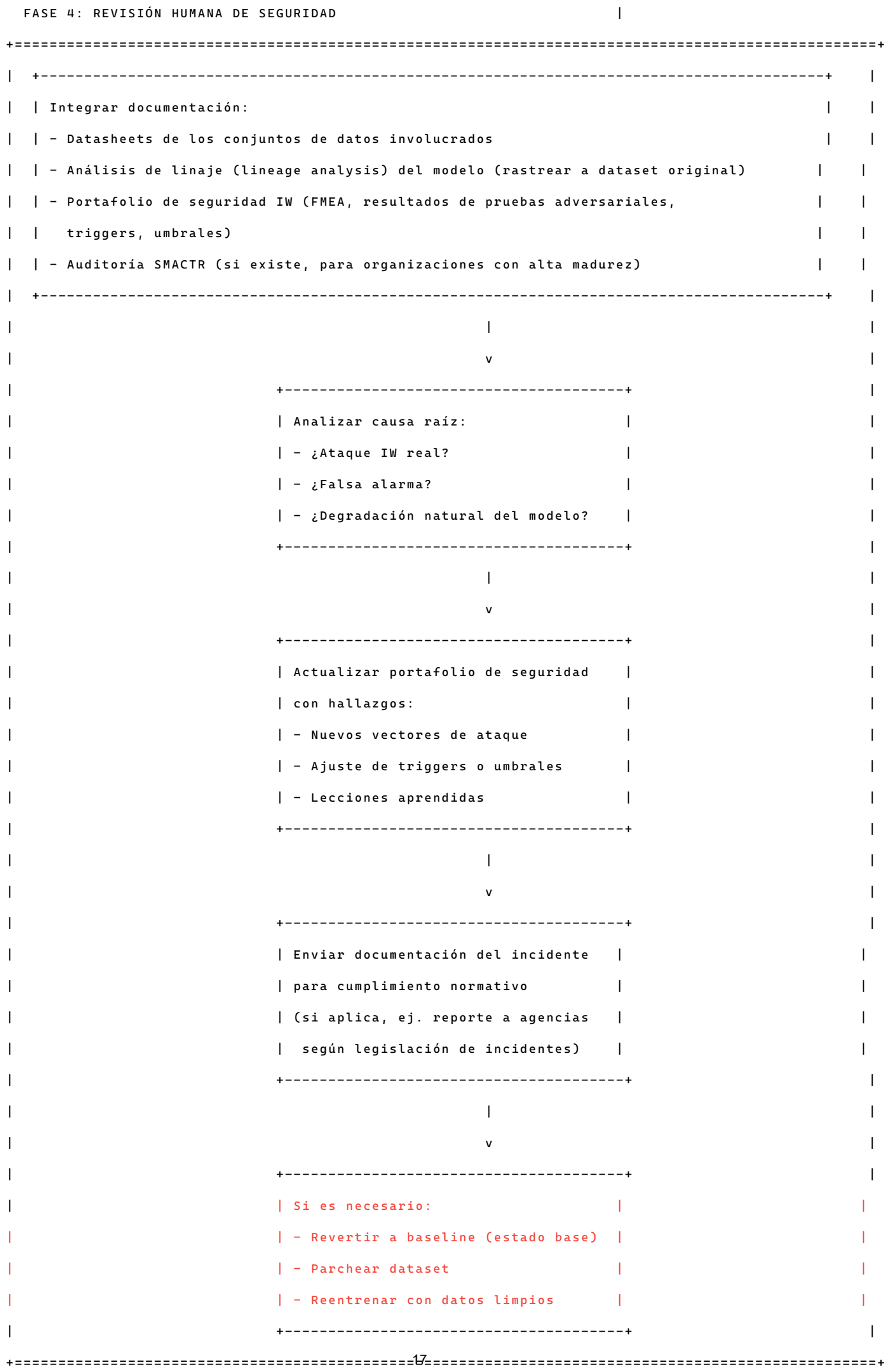
El *análisis de linaje* (Fase 4) requiere que el pipeline de entrenamiento haya registrado metadatos suficientes (versiones de dataset, hiperparámetros, transformaciones, etc.).

El *baseline trusted state* (estado base de confianza) debe definirse para cada sistema, idealmente correspondiente a la primera versión documentada con datasheets y model cards.


PROCESO DE MITIGACIÓN DE RIESGOS IW EN AIML (COX 2022)
FASE 1: PROTOTIPADO (Desarrollo en paralelo)
Desarrollo del modelo (entrenamiento inicial, validación, métricas)
Modelo entrenado (versión base)
Documentación obligatoria (mínima para seguridad)
- Datasheets (Gebru)
- Model cards (Mitchell)
Análisis de seguridad IW (independiente pero concurrente)
1. FMEA (identificar fallos, vectores de ataque)
2. Pruebas adversariales (diseñar entradas maliciosas basadas en FMEA)
3. Portafolio de seguridad IW
a) Vectores de ataque documentados
b) Triggers (disparadores) para revisión
c) Umbrales de comparación
Check-in a repositorio de metadatos y artefactos (modelo, portafolio, datasheets, model cards)


FASE 2: MONITORIZACIÓN CONTINUA (Pipeline)
Cargar artefactos desde repositorio:
- Logs de inferencia (predicciones recientes)
- Portafolio de seguridad IW (triggers, umbrales)
- Estadísticas base del modelo (distribuciones de referencia, precisión histórica, etc.)
- Esquema de referencia (tipos de datos, rangos, categorías)
Comparar métricas del modelo actual
contra TRIGGERS del portafolio
v (si se cumple)
v (no se cumple)
INCIDENTE IW POTENCIAL
- Activar revisión humana
- Notificar a equipo de seguridad
- Detener inferencia si es crítico?
Continuar monitorización normal:
- Verificar sesgos de esquema
- Detectar deriva de distribución
- Detectar deriva de concepto
- Evaluar decaimiento del modelo
¿Se requiere reentrenamiento?
(por programación, evento o umbral)
v (si)
Iniciar pipeline de entrenamiento
continuo


FASE 3: ENTRENAMIENTO CONTINUO (Pipeline)
Solicitar artefactos desde repositorio:
- Modelo actual (o versión anterior)
- Portafolio de seguridad IW (umbrales de comparación)
- Dataset(s) de entrenamiento (nuevos o actualizados)
- Datasheets y model cards asociados
Validar datos de entrenamiento
contra UMBRALES del portafolio
(en etapa de data validation)
v (supera umbral)
v (no supera)
ALERTA: POSIBLE INCIDENTE IW
- No proceder con entrenamiento?
- Activar revisión humana
- Notificar a equipo de seguridad
Continuar entrenamiento:
- Reentrenar modelo
- Registrar metadatos: fecha, versión,
métricas, umbrales superados (si
los hubo) pero considerados seguros
Modelo reentrenado -> desplegar
(si pasa validaciones adicionales)


FASE 4: REVISIÓN HUMANA DE SEGURIDAD
Integrar documentación:
- Datasheets de los conjuntos de datos involucrados
- Análisis de linaje (lineage analysis) del modelo (rastrear a dataset original)
- Portafolio de seguridad IW (FMEA, resultados de pruebas adversariales,
triggers, umbrales)
- Auditoría SMACTR (si existe, para organizaciones con alta madurez)
Analizar causa raíz:
- ¿Ataque IW real?
- ¿Falsa alarma?
- ¿Degradación natural del modelo?
Actualizar portafolio de seguridad
con hallazgos:
- Nuevos vectores de ataque
- Ajuste de triggers o umbrales
- Lecciones aprendidas
Enviar documentación del incidente
para cumplimiento normativo
(si aplica, ej. reporte a agencias
según legislación de incidentes)
Si es necesario:
- Revertir a baseline (estado base)
- Parchear dataset
- Reentrenar con datos limpios

6. Esquema integrado

Esta sección integra el funcionamiento estructural de un sistema AIML (Sección 2) con las vulnerabilidades específicas de cada etapa (Sección 3), mostrando cómo el proceso de mitigación de Cox (Secciones 4 y 5) se aplica sobre cada vulnerabilidad. Se presenta una tabla de correspondencia entre la vulnerabilidad y el componente de mitigación; acompañada de un diagrama ASCII que coloca lado a lado el flujo funcional y las vulnerabilidades asociadas.

Correspondencia entre vulnerabilidades y componentes de mitigación. La siguiente tabla (basada en el texto de Cox) muestra cómo cada vulnerabilidad descrita en la Sección 3 es abordada directamente por uno o varios componentes del proceso de mitigación.

Vulnerabilidad	Componente de mitigación (Cox)
Acceso garantizado a datos públicos	- *Datasheets* (documentar fuente pública, motivación, riesgos). - *Triggers* de monitorización si se detecta acceso anómalo. - Definir el *baseline state* (si los datos son públicos, asumir el peligro potencial).
Red co-crea datos de entrenamiento	- FMEA (identificar los bucles de retroalimentación como vector). - Umbrales en validación de datos para detectar los cambios inducidos por *Red*. - Pruebas adversariales para simular la co-creación.
Falta de *datasheets*	- Obligatorio en nivel mínimo de madurez (Sección 4, Componente 2).
Envenenamiento de datos (*data poisoning)*	- FMEA (documentar como fallo previsible). - Pruebas adversariales específicas para *poisoning*. - Umbrales en validación de datos (detección de muestras anómalas).

Vulnerabilidad	Componente de mitigación (Cox)
Sesgo malicioso como riesgo de seguridad	- *Model cards* (documentar comportamiento por grupos). - *Triggers* en monitorización (caída de rendimiento en subgrupos). - Auditoría SMACTR (nivel maduro).
Falta de FMEA o pruebas adversariales	- Componente 1 y 2 del proceso (obligatorios).
Sondear el sistema para inferir el conocimiento del modelo	- Triggers de monitorización (detectar los patrones de consultas anómalas) *[Cox no menciona explícitamente la detección de patrones de consultas como trigger, pero es una aplicación razonable del concepto de monitoreo].* - Revisión humana al activarse el *trigger*.
Manipulación de los algoritmos sociales	- FMEA (incluir los ataques de manipulación de retroalimentación). - Pruebas adversariales con entradas de usuario simuladas.
Falta de *model cards*	- Obligatorio en nivel mínimo de madurez.
Sin monitorización continua con *triggers*	- Componente 3 (integración en el *pipeline* de monitorización).
Bucle de retroalimentación (lo más crítico)	- FMEA (identificar el bucle como vector prioritario). - Umbrales estrictos en la validación de datos. - *Triggers* sensibles en la monitorización. - Análisis de linaje para rastrear los cambios introducidos por el bucle. - Revisión humana inmediata al superar los umbrales.

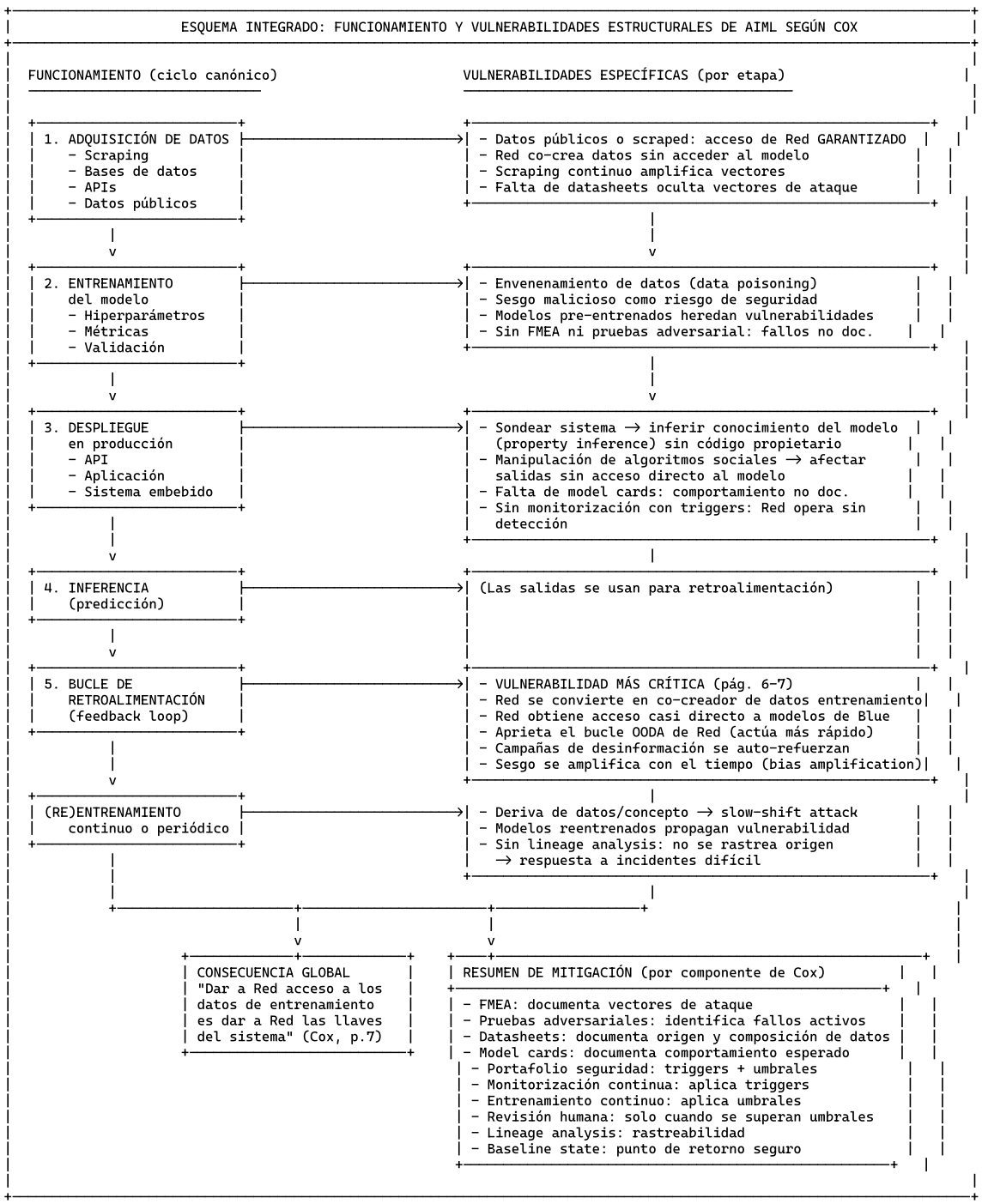
Vulnerabilidad	Componente de mitigación (Cox)
Apriete del bucle OODA de *Red*	- Acciones de *Blue* para mantenerse dentro del OODA de *Red* (cambios frecuentes de sistema). - *Triggers* de monitorización con tiempos de respuesta definidos.
Sesgo amplificado por un bucle	- *Model cards* con métricas por grupo. - Auditoría SMACTR para medir la amplificación.
Deriva de datos/concepto (*slow-shift attack*)	- Umbrales en la validación de datos (detección de deriva). - *Lineage analysis* para comparar las distribuciones históricas. - Revisión humana si la deriva supera el umbral.
Sin *lineage analysis*	- Requisito en los *pipelines* MLOps (Sección 4, Componente 3). - Necesario para cumplir el nivel mínimo de madurez (pregunta G1 adicional).
Falta de *baseline trusted state*	- Definir el estado base (primera versión documentada con *datasheets* y *model cards*). - Pregunta G1 del índice de madurez.

Conclusión del esquema integrado

El esquema muestra que cada vulnerabilidad estructural de los sistemas AIML tiene una correspondencia directa con al menos uno de los componentes del marco de Cox. El proceso no es una lista de buenas prácticas independientes, sino un sistema acoplado: el FMEA y las pruebas adversariales alimentan al portafolio; el portafolio alimenta a los *triggers* y los umbrales; los *triggers* y los umbrales se integran en los *pipelines* automatizados; la automatización reduce la carga de revisión humana; la revisión humana, cuando ocurre, actualiza el portafolio.

Este ciclo virtuoso de mejora continua es la contribución central de Cox para la resiliencia de AIML en los entornos de guerra de información o IW.

En la siguiente página, un diagrama ASCII integrado, incluye el funcionamiento del sistema y las vulnerabilidades estudiadas.


ESQUEMA INTEGRADO: FUNCIONAMIENTO Y VULNERABILIDADES ESTRUCTURALES DE AIML SEGÚN COX
FUNCIONAMIENTO (ciclo canónico)
VULNERABILIDADES ESPECÍFICAS (por etapa)
1. ADQUISICIÓN DE DATOS
- Scraping
- Bases de datos
- APIs
- Datos públicos
- Datos públicos o scraped: acceso de Red GARANTIZADO
- Red co-crea datos sin acceder al modelo
- Scraping continuo amplifica vectores
- Falta de datasheets oculta vectores de ataque
2. ENTRENAMIENTO del modelo
- Hiperparámetros
- Métricas
- Validación
- Envenenamiento de datos (data poisoning)
- Sesgo malicioso como riesgo de seguridad
- Modelos pre-entrenados heredan vulnerabilidades
- Sin FMEA ni pruebas adversarial: fallos no doc.
3. DESPLIEGUE en producción
- API
- Aplicación
- Sistema embebido
- Sondear sistema → inferir conocimiento del modelo (property inference) sin código propietario
- Manipulación de algoritmos sociales → afectar salidas sin acceso directo al modelo
- Falta de model cards: comportamiento no doc.
- Sin monitorización con triggers: Red opera sin detección
4. INFERENCIA (predicción)
(Las salidas se usan para retroalimentación)
5. BUCLE DE RETROALIMENTACIÓN (feedback loop)
- VULNERABILIDAD MÁS CRÍTICA (pág. 6-7)
- Red se convierte en co-creador de datos entrenamiento
- Red obtiene acceso casi directo a modelos de Blue
- Aprieta el bucle OODA de Red (actúa más rápido)
- Campañas de desinformación se auto-refuerzan
- Sesgo se amplifica con el tiempo (bias amplification)
(RE)ENTRENAMIENTO continuo o periódico
- Deriva de datos/concepto → slow-shift attack
- Modelos reentrenados propagan vulnerabilidad
- Sin lineage analysis: no se rastrea origen → respuesta a incidentes difícil
CONSECUENCIA GLOBAL
"Dar a Red acceso a los datos de entrenamiento es dar a Red las llaves del sistema" (Cox, p.7)
RESUMEN DE MITIGACIÓN (por componente de Cox)
- FMEA: documenta vectores de ataque
- Pruebas adversariales: identifica fallos activos
- Datasheets: documenta origen y composición de datos
- Model cards: documenta comportamiento esperado
- Portafolio seguridad: triggers + umbrales
- Monitorización continua: aplica triggers
- Entrenamiento continuo: aplica umbrales
- Revisión humana: solo cuando se superan umbrales
- Lineage analysis: rastreabilidad
- Baseline state: punto de retorno seguro