

Recomendación de CWE y CVSS mediante técnicas de interpretación textual: Caso AuditForge

Ricardo Olalquiaga

ricardo.olalquiaga@usm.cl

Profesor guía: Carlos Valle

Resumen

La generación de reportes en auditorías de ciberseguridad consume una proporción significativa del tiempo de los auditores, afectando la eficiencia del proceso de hacking ético. Este trabajo propone una solución basada en técnicas de procesamiento de lenguaje natural y modelos de Deep Learning para automatizar la recomendación de clasificaciones CWE (*Common Weakness Enumeration*) y CVSS (*Common Vulnerability Scoring System*) a partir de descripciones de vulnerabilidades. El objetivo es reducir el tiempo invertido en la clasificación manual y mejorar la consistencia en la evaluación de riesgos. Para ello, se diseñaron dos arquitecturas: una jerárquica para CWE, basada en cross-encoders y clasificación jerárquica, y otra multi-output para CVSS, que primero determina la severidad y luego predice los componentes del vector correspondiente. La validación se realizó sobre conjuntos de datos obtenidos de la National Vulnerability Database, empleando métricas como accuracy, precision, recall y F1 score. Los modelos alcanzaron un 79,66 % de Top-1 Accuracy en CWE y un 91,85 % de accuracy promedio en la predicción de componentes CVSS. El impacto esperado incluye una mejora significativa en la eficiencia de los procesos de auditoría, estandarización en la clasificación de vulnerabilidades y soporte a la toma de decisiones en ciberseguridad.

Palabras Clave: cybersecurity, cwe, cvss, hierarchical classification, nlp

1 Glosario

CWE: Common Weakness Enumeration

CVSS: Common Vulnerability Scoring System

AV: Attack Vector (N: Network, L: Local)

AC: Attack Complexity (L: Low, H: High)

PR: Privileges Required (N: None, L: Low)

UI: User Interaction (R: Required, N: None)

C/I/A: Confidentiality/Integrity/Availability Impact (H: High, L: Low, N: None)

S: Scope (U: Unchanged)

ANN: Artificial Neural Network

2 Introducción

2.1 Contexto y motivación

El contexto de esta investigación se da en el marco del proyecto AuditForge, realizado para la 32ava Feria de Software organizado por el Departamento de Informática de la Universidad Técnica Federico Santa María.



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: Recomendación de CWE y CVSS mediante técnicas de interpretación textual: Caso AuditForge

Nombre del candidato(a): Ricardo Olalquiaga

Carrera / Grado: Ingeniería Civil Informática

Campus: Casa Central ; Departamento: Departamento de Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Carlos Valle , en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 01/08/2025

; Firma:

Estudiante o Candidato(a):

Fecha: 01/08/2025

; Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

AuditForge es un software open-source diseñado para automatizar los reportes de ciberseguridad de las organizaciones y personas interesadas en el análisis de seguridad de sistemas informáticos, o bien, *hacking* ético. Esta automatización se realiza mediante el uso de *templates* definidos para la generación customizada de los reportes, asimismo con la asistencia de algoritmos de Inteligencia Artificial para la clasificación de las vulnerabilidades halladas dado estándares internacionales.

Así, la recomendación de clasificadores de vulnerabilidades permite al usuario acelerar el proceso de reportería, buscando reducir el tiempo destinado a la generación de reportes en la búsquedas de vulnerabilidades.

2.2 Definición del problema

El *hacking* ético es la práctica de identificar y explotar vulnerabilidades en sistemas informáticos de manera autorizada. A diferencia de los hackers maliciosos, los hackers éticos actúan de manera legal y ética. Bajo este contexto, hay empresas especializadas que realizan pruebas de seguridad mediante auditorías, con el fin de informar sobre debilidades que comprometen los sistemas del cliente. Así, los auditores de seguridad son los encargados de realizar las pruebas de vulnerabilidad y culminan el proceso de identificación de vulnerabilidades con un reporte que resume todos sus hallazgos.

El tiempo utilizado en la generación de informes de los auditores de seguridad resulta en un problema, ya que puede ocupar hasta el 40 % del tiempo total dedicado al proyecto [1]. Esto hace que el proceso de *hacking* ético no sea óptimo, ya que resta tiempo útil a la realización de pruebas de seguridad adicionales en función del plazo establecido por el cliente. Este problema impacta negativamente al auditor y a la empresa de ciberseguridad, dado que reduce el tiempo que el auditor dedica a la búsqueda de vulnerabilidades y aumenta el tiempo destinado a la generación de reportes.

Dentro de las auditorías, se indican las vulnerabilidades dentro del alcance acordado entre el auditor y el cliente. Cada vulnerabilidad, o bien cada CVE (Common Vulnerabilities and Exposures), tiene varios campos que detallan a esta misma, las cuales suelen ser su descripción, observaciones detectadas, medidas de mitigación, calificación CVSS que es un estándar utilizado para medir la gravedad dentro del rango entre 0 (muy leve) a 10 (crítico), y la calificación CWE que es un clasificador estandarizado único en debilidades de software.

La clasificación de CWE no es una tarea trivial. Existen actualmente 1426 clasificaciones distintas detalladas en las áreas de desarrollo de software, diseño de hardware y conceptos de investigación en las cuales los auditores de seguridad deben conocer para hacer la clasificación de las vulnerabilidades [2]. Realizar una correcta clasificación se suma a los pobres datos existentes respecto a la metadata de los CWE, donde un 31 % de los registros de vulnerabilidades comunes (CVS) no poseen una clasificación de CWE (e.g. NVD-CWE-noinfo) [3].

Por otro lado, la clasificación de CVSS también requiere atención. La calificación de los CVSS es subjetiva [4] y es importante indicar el nivel de riesgo que representa una vulnerabilidad para poder priorizar su mitigación.

En esta investigación se propone en el contexto de la feria de software del año 2024, un sistema recomendador de CWE y CVSS, ambos mediante el uso de la descripción de la vulnerabilidad, para así automatizar el proceso de reportería en las auditorías de ciberseguridad con datos recabados de la National Vulnerability Database (NVD) para realizar los modelos, donde cada entrada recibida de la NVD corresponde a una vulnerabilidad conocida.

2.3 Metodología para validación

La validación de los modelos se hará en base a conjuntos de testeo respectivos para cada uno de estos, así se tiene un conjunto de datos que el modelo no ha visto para comprobar su efectividad.

2.4 Organización de la tesina

El informe contiene la Sección 1, se presenta un glosario con términos que se repiten durante todo el trabajo. En la Sección 2 contiene la introducción, donde se presenta el contexto y motivación del trabajo, la definición del problema, y la metodología para la validación. En la sección 3, se visita la literatura, con el marco teórico y el estado del arte. En la sección 4, se presentan los objetivos. En la Sección 5, se presenta la propuesta de solución. En la Sección 6 se especifica la solución propuesta, se diseñan los instrumentos de validación, se comenta la recolección y el tratamiento de los datos, y se analizan los datos y resultados. En la sección 7 se hacen las conclusiones, comentando el significado de los hallazgos y el trabajo futuro. Por último, hay un anexo que contiene los gráficos referenciados a lo largo del trabajo.

3 Revisión de la literatura

Esta sección permite introducir al marco teórico que se evidencia a lo largo de la tesina, además del estado del arte de las soluciones para clasificación de CWE y CVSS.

3.1 Marco teórico

1. CWE

Es un sistema de categorización estandarizado para debilidades y vulnerabilidades en software y hardware, desarrollado y mantenido por la MITRE Corporation con el apoyo de agencias gubernamentales como el Departamento de Seguridad Nacional de EE.UU. (DHS) y US-CERT Security Operations Center [4]. Su objetivo principal es proporcionar un lenguaje común para identificar, analizar y mitigar fallos de seguridad en el desarrollo y despliegue de sistemas tecnológicos. [5].

Actualmente, existen 943 CWE únicas en categorías jerárquicas, y algunas mucho más comunes que otras. Se relaciona con la Vista-1003 [6], que es un subconjunto de todas las clasificaciones CWE con dos niveles jerárquicos entre los CWE presentes. Cada vulnerabilidad en este árbol está relacionado a un padre, o al CWE-1003, el cual es la raíz del árbol jerárquico. La Vista-1003 posee 130 CWE asociados, los cuales permite relacionar entre 37 CWE padres, y 93 CWE hijos. Dentro de esta investigación, la clasificación de CWE se hará sobre estas clasificaciones.

Estos CWE se vinculan directamente con los CVE (Common Vulnerabilities and Exposures), o simplemente vulnerabilidades, para relacionar vulnerabilidades específicas con sus causas subyacentes o CWE.

2. CVSS

El CVSS es un estándar internacional desarrollado para evaluar cuantitativamente la gravedad de vulnerabilidades en sistemas informáticos. Proporciona una puntuación numérica (de 0 a 10) y una representación cualitativa (Baja, Media, Alta, Crítica) que permite priorizar la remediación de vulnerabilidades en función de su impacto potencial [7].

Es mantenido por el Forum of Incident Response and Security Teams (FIRST) desde el 2005, con colaboración de agencias como NIST y MITRE. [8] Es usado en bases de datos como el National Vulnerability Database (NVD) para enriquecer entradas de CVE con métricas CVSS.

Ha tenido una evolución en base a sus versiones, siendo la versión 1 del 2005, pasando por la versión 3.1 (ocupada en el contexto de esta investigación) del año 2015, al 4.0 que se introdujo el año 2023 [9]. El CVSS evalúa vulnerabilidades mediante tres grupos de métricas (en versiones anteriores a v4.0) o cuatro (en CVSS v4.0). Posee métricas base, que son las características intrínsecas de la vulnerabilidad, invariables en el tiempo y entornos; métricas temporales (para CVSS v3.1) o amenaza (para CVSS v4.0) que son factores dinámicos como madurez del exploit y disponibilidad de parches; métricas ambientales que adaptan la puntuación al contexto específico de una organización, y métricas complementarias asociadas a la versión v4.0 [10].

3. Fundamentos Técnicos para Clasificación de Vulnerabilidades

a) Procesamiento de Texto en NLP

El procesamiento de texto en NLP es un conjunto de técnicas para transformar lenguaje natural en representaciones estructuradas que puedan ser analizadas por modelos computacionales. Su objetivo es capturar el significado semántico, sintáctico y contextual del texto, eliminando ruido y normalizando la información para su procesamiento automático [11].

Posee una serie de pasos comunes, como *tokenización* que es la división del texto en unidades mínimas (tokens), como palabras o subpalabras; *lematización* que es la reducción de palabras a sus formas canónicas (lemas) o raíces morfológicas (ej: "corriendo" → "correr"); *eliminación de stop-words* que es el filtrado de palabras frecuentes pero poco informativas (ej: artículos, preposiciones) para enfocarse en términos relevantes; y los *embeddings* que son representaciones vectoriales que capturan relaciones semánticas entre palabras.

b) Modelos Cross-Encoder

Los modelos Cross-Encoder son arquitecturas de Transformers que procesan pares de textos simultáneamente (ej: una descripción de CVE y la definición de un CWE) para calcular su relevancia semántica mediante atención bidireccional. A diferencia de los Bi-Encoder, que generan embeddings independientes, los Cross-Encoder evalúan la interacción directa entre los textos, lo que permite una mayor precisión en tareas de clasificación y ranking [12].

c) Modelos Multi-Output

Los modelos Multi-Output (o multi-tarea) son redes neuronales que predicen múltiples variables interdependientes simultáneamente. En el contexto de CVSS, permiten clasificar varios componentes del vector (ej: Attack Vector, User Interaction) en paralelo, aprovechando características compartidas en las capas ocultas [13].

d) DistilBERT

Este es una versión optimizada de BERT, creada mediante *distillation* (destilación de conocimiento). Conserva el 97 % del rendimiento del modelo original pero con un 40 % menos de parámetros y un 60 % más de velocidad, ideal para entornos con recursos limitados. Es un modelo multilingüe, y se puede usar para generar embeddings y clasificar texto [14].

3.2 Estado del arte

3.2.1 CWE

En la clasificación de CWE, los enfoques modernos se basan principalmente en procesamiento de lenguaje natural y modelos jerárquicos. Por ejemplo, Das et al. presentaron V2W-BERT, un marco de clasificación jerárquica multiclase de CVEs a CWEs, obteniendo una *accuracy* precisa de 87 % de exactitud en 124 CWE encontrados por la Vista-1003 [15]. Otros enfoques utilizan similitud semántica entre la descripción del CVE y las definiciones de CWE como Text2Weak que hizo un emparejamiento de textos de vulnerabilidad a CWEs usando embeddings de lenguaje (GPT), pero encontró un bajo F1 promedio de 22 % debido a la diversidad de clases [16]. Hay trabajos especializados como CySecBERT-ARD, que fine-tunea un BERT adaptado a ciberseguridad con mecanismos de atención adicionales, mejorando la comprensión contextual, enfocándose en las 50 CWE más frecuentes para garantizar datos suficientes con 91 % de exactitud y F1 en la clasificación de las categorías CWE elegidas [17]. Se emplean modelos basados en transformers (BERT, DistilBERT, RoBERTa, etc.) para capturar el contexto de la descripción; asimismo se han explorado arquitecturas de codificadores cruzados (cross-encoders) que procesan juntas la descripción del CVE y la descripción de cada CWE candidato como el trabajo de Kota et al. [18].

Los principales retos incluyen el desequilibrio de clases (muchas CWE raras con pocos CVEs) y la naturaleza jerárquica de CWE. Esto provoca que los modelos tengan dificultades para distinguir subclases finas, confundan un CWE con sus padres/hijos frecuentes y obtengan bajos F1 en clases minoritarias. Además, muchos CVEs pueden corresponder a múltiples CWE (tarea multilabel), aunque frecuentemente solo se etiqueta el más representativo.

Esta investigación se basa principalmente en el trabajo realizado por Kota et al. el año 2024. Su propuesta consiste en capturar la relación jerárquica de CWE obtenido por la Vista-1003 mediante una solución con relación semántica entre las descripciones de CVE mediante el uso de cross-encoders para cada nivel jerárquico, y un modelo clasificador como punto de partida para identificar el nivel correspondiente del CVE. Sin embargo, esta investigación falla en combatir el evidente imbalance entre las clases de CWE.

3.2.2 CVSS

En la predicción de componentes CVSS, la tendencia reciente es tratarlo como un problema de clasificación múltiple basado en texto descriptivo del CVE. Los principales enfoques utilizan desde transformers como BERT y variantes, hasta redes neuronales y métodos basados en NLP clásico como MLP. Por ejemplo, varios estudios emplean BERT o DistilBERT fine-tuneados para predecir la severidad directamente a partir de la descripción [19]. Shi et al [20] logran entrenar XLNet fine-tuneado sobre un corpus específico de CVEs para lograr un rendimiento de 88 % en F1 score. Manjunatha et al [21] predice la severidad CVSS utilizando GPT-2 con un aumento de datos utilizando BERT, y logrando un F1 score de 0.81.

Predecir CVSS desde descripciones textuales presenta dificultades inherentes. El desequilibrio de clases (menos CVEs con severidad “Low”) disminuye el rendimiento en esas categorías más raras. Además, las puntuaciones CVSS dependen de factores contextuales (configuración del sistema, vectores de ataque) que a veces no se expresan completamente en el texto, lo que limita la precisión teórica del modelo. Los modelos de aprendizaje profundo suelen ser opacos: la falta de interpretabilidad hace que los analistas desconfíen de las predicciones automáticas [22].

4 Objetivos

El objetivo general es proponer modelos de Deep Learning que permitan realizar recomendaciones de CWE y CVSS para optimizar el proceso de reportería en auditorías de ciberseguridad.

Los objetivos específicos son:

1. Recopilar y preparar datos públicos de la National Vulnerability Database (NVD) para generar un conjunto de datos adecuado para el entrenamiento y validación de los modelos.
2. Diseñar y entrenar modelos de clasificación enfocados en la recomendación de CWE y CVSS, asegurando su alineación con las necesidades del proceso de reportería en auditorías.
3. Evaluar el desempeño de los modelos propuestos mediante métricas de clasificación para comprobar su efectividad y utilidad en escenarios reales.

5 Propuesta de solución

5.0.1 Recomendación de CWE

En el contexto para la identificación de CWE, se ocupará la arquitectura propuesta por Kota et al. [18], que consiste en un clasificador jerárquico que es un modelo MLP que su función es clasificar a la descripción preprocesada hacia el clasificador jerárquico superior o inferior adecuado, en base a la Vista-1003.

El clasificador jerárquico superior e inferior son modelos cross-encoder, que en base a la relación semántica textual, relaciona el contenido del CVE preprocesado y el contenido de los CWE apropiados a su nivel jerárquico.

Una vez predicho, se obtiene una lista ordenada de los CWE más relacionados con el CVE.

La figura 1 representa el flujo de la recomendación de CWE desde la etapa de recolección, limpieza de datos, clasificación y salida de CWE predichos.

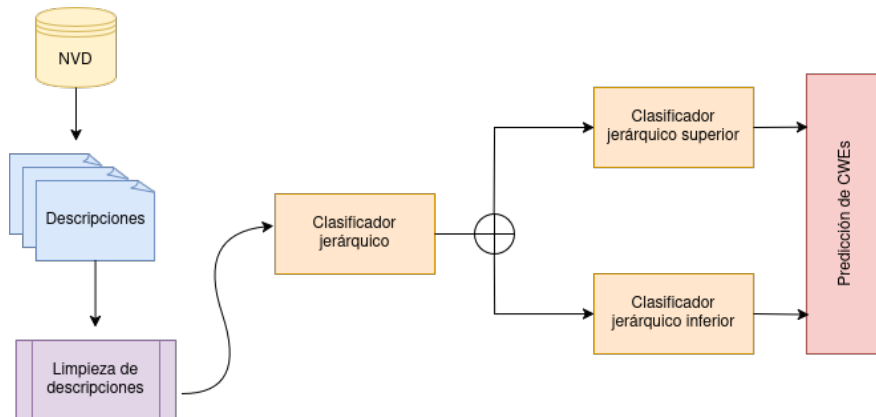


Figura 1: Flujo de predicción de CWE

5.0.2 Recomendación de CVSS

Para lograr la predicción de CVSS, cabe destacar que importa la predicción de los componentes de estos más que su severidad en el contexto de AuditForge. Así, las descripciones de CVE preprocesadas son primero

usadas por el clasificador de severidad, el cual se encarga de dirigir hacia el clasificador de componentes de CVSS necesario para determinar cada uno de los vectores.

La figura 2 representa el flujo de la predicción de CVSS. Primero se considera la recolección de datos y la limpieza de estos, para luego permitir la clasificación de la severidad, y los componentes asociados a esta severidad en su propio clasificador. Este último clasificador entrega la predicción de componentes CVSS.

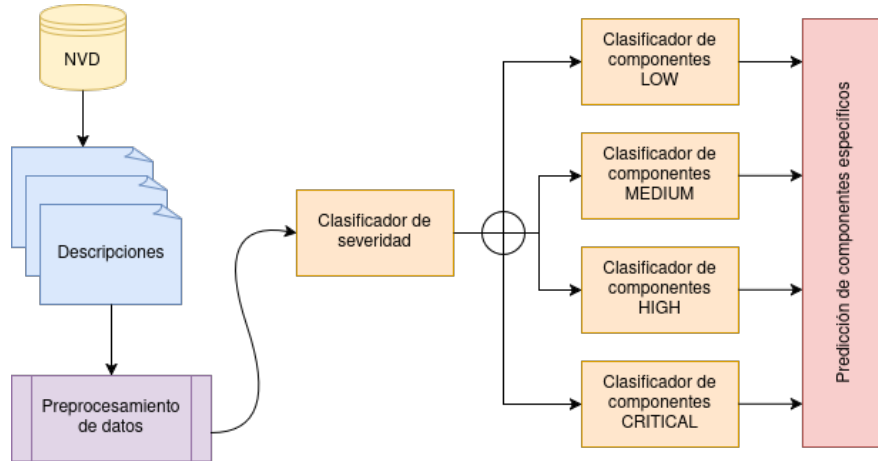


Figura 2: Flujo de predicción de componentes CVSS

6 Desarrollo

Esta sección abarca toda la especificación de la solución propuesta en el capítulo anterior, donde se indica su construcción respectiva separado en el modelo respectivo de CWE y de CVSS, diseño de las métricas para evaluación de los modelos, recolección y tratamiento de los datos, y el análisis de los datos y resultados.

6.1 Especificación de la solución propuesta

Esta subsección detalla las partes y los componentes de los modelos de CWE y de CVSS.

6.1.1 Recomendación de CWE

Para hacer la recomendación de CWE, se ocupará la arquitectura propuesta por Kota et al. [18], con un enfoque jerárquico basado en la relación semántica entre las descripciones de CVE y las de CWE. Este enfoque se compone de 3 modelos:

1. Modelo jerárquico superior (o bien *top layer model*): Modelo cross-encoder encargado de relacionar los CWE superiores o padres de la Vista-1003. Este modelo recibe de input una tupla de dos oraciones de texto en formato string, y entrega de output un valor numérico entre 0 y 1 indicando la similaridad entre estos dos strings.
2. Modelo jerárquico inferior (o bien *bottom layer model*): Modelo cross-encoder encargado de relacionar los CWE inferiores o hijos de la Vista-1003. Este modelo recibe de input una tupla de dos oraciones de texto en formato string, y entrega de output un valor numérico entre 0 y 1 indicando la similaridad entre estos dos strings.

3. Clasificador jerárquico (o bien *layer classifier*): Transformer basado en BERT encargado de clasificar inicialmente el CVE hacia alguno de los modelos jerárquicos. Este modelo recibe de input un string y retorna un valor binario que mapea directamente hacia los modelos jerárquicos.

La arquitectura consiste en usar el *layer classifier* para poder, posteriormente, asignarlo correspondientemente al *top layer model* o al *bottom layer model* para realizar la recomendación de los CWE. Se retorna una lista con los CWE predecidos ordenados de mayor a menor con el valor de relación semántica que exista entre las descripciones de CVE y de CWE.

Para cada uno de los modelos jerárquicos, al ser modelos cross-encoder, requieren de input un par de strings para poder generar la relación semántica textual entre estos. Para generar estos input, se generan 3 tipos de pares de input, en conjunto con el target deseado:

1. Lista de pares 1: Se genera una lista con tuplas del estilo (s_1, s_2, t_1) donde s_1 corresponde a la descripción postprocesada de CVE, s_2 corresponde a la descripción postprocesada de CWE, y $t_1 \in \{0, 1\}$ corresponde a un valor que indica con un 1 si el CVE está relacionado al CWE correcto, si no es 0. Para el *top layer model*, se consideran todos los CVE con su CWE correcto para los pares positivos, y en los casos negativos, solo se seleccionaron 20 CWE aleatorios para no hacer un aumento explosivo de los datos de entrenamiento. Así, se crearon 2.365.062 pares. Para el *bottom layer model*, se generaron 6.058.542 datos de entrenamiento.
2. Lista de pares 2: Se genera una lista con tuplas del estilo (s_1, s_2, t_1) del producto cartesiano entre los CVE, donde s_1, s_2 corresponden a descripciones postprocesadas de los CVE, y $t_1 \in \{0, 1\}$ es su producto cartesiano. Para el *top layer model*, se generaron 1.369 pares. Para el *bottom layer model*, se generaron 8.649 datos de entrenamiento.
3. Lista de pares 3: Se genera una lista con tuplas del estilo (s_1, s_2, t_1) , donde s_1, s_2 corresponden a descripciones postprocesadas de CVE, y $t_1 \in \{0, 1\}$ corresponde a un valor que indica 1 si ambos CVE pertenecen al mismo CWE o, en caso contrario, es 0. Se limita a un máximo de 300 pares por CWE para evitar el aumento explosivo de los datos de entrenamiento. Para el *top layer model*, se generaron 22.200 datos de entrenamiento. Para el *bottom layer model*, se generaron 54.720 datos de entrenamiento.

En total, se tienen 2.388.631 pares para el modelo *top layer* y 6.121.911 pares para el modelo *bottom layer*. Estos pares son separados en un 80 % para el conjunto de entrenamiento, un 10 % para el conjunto de validación y un 10 % para el conjunto de testeo. Esta separación fue realizada con estratificación para asegurar la igualdad de clases en cada conjunto.

Parameter	Value
Model Architecture	CrossEncoder
Base Model	distilbert-base-uncased
Train Batch Size	32
Number of Epochs	3

Cuadro 1: Parámetros - Top cross-encoder model

Parameter	Value
Model Architecture	CrossEncoder
Base Model	distilbert-base-uncased
Train Batch Size	32
Number of Epochs	4

Cuadro 2: Parámetros - Bottom cross-encoder model

Para el modelo de clasificación de capa jerárquica (*layer classifier*), se fusionaron los conjuntos de entrenamiento, validación y testeo de los modelos jerárquicos para ser congruente con los datos utilizados para entrenar y validar. En este caso, todos los datos provenientes del modelo jerárquico superior fueron asignados con un target 1, y los del inferior fueron asignados con un target 0.

El formato para el *layer classifier* es una tupla de la forma (s_1, t_1) , donde s_1 corresponde al string postprocesado de la descripción del CVE, y t_1 es el target asociado. Los datos fusionados fueron mezclados, se eliminó el segundo string (s_2 de las tuplas de pares), y eliminar las filas duplicadas de los dataset combinados. El conjunto de entrenamiento quedó con 396.958 CVE, el de validación con 354.453 CVE, y el de testeo con 354.807 datos.

El modelo fue adecuado con varios métodos para su regularización, como el uso de scheduler y early stopping. Así, se usó un `WeightedRandomSampler` como otro método para manejar el desbalanceo de clases en el entrenamiento.

Parámetro	Valor
Modelo base	distilbert-base-multilingual-cased
Dropout	0.3
Batch size	32
Optimizador	AdamW
Tasa de aprendizaje (lr)	2e-5
Weight decay	0.01
Scheduler	ReduceLROnPlateau
Norma máxima de gradiente (max_grad_norm)	1.0
Épocas	12

Cuadro 3: Parámetros - Layer classifier

6.1.2 Recomendación de CVSS

Para hacer la recomendación de CVSS, se ocupará la arquitectura propuesta en 2. Este enfoque se compone de 3 modelos:

- Modelo clasificador de severidad: Modelo MLP encargado de clasificar la severidad de las descripciones preprocesadas de CVE. Este modelo recibe de input un string correspondiente a una descripción CVE y retorna una categoría entre 4 posibles correspondientes a las severidades CVSS.
- Modelos clasificadores de componentes CVSS: Modelo MLP multi-output encargado de clasificar los componentes del vector CVSS correspondiente para cada severidad. Este modelo recibe de input un string correspondiente a la descripción CVE, y retorna una lista con 8 elementos, donde cada elemento representa un valor categórico correspondiente al componente CVSS que representa.

Estos modelos siguen un flujo tal que las descripciones de CVSS preprocesadas son primero digeridas por el modelo clasificador de severidad. Este modelo tiene 4 output posibles, de acuerdo a las cuatro severidades en las cuales se pueden clasificar los CVSS. Al obtener la clasificación más probable, se elige el modelo clasificador de componentes CVSS que está asociada a esta clasificación.

Cada modelo clasificador de componentes CVSS 3.1 predice los siguientes elementos correspondiente al vector CVSS y en el siguiente orden:

1. AC: Vector de ataque (Attack Vector)
2. AC: Complejidad del ataque (Attack Complexity)
3. PR: Privilegios requeridos (Privileges Required)

4. UI: Interacción del usuario (User Interaction)
5. S: Alcance (Scope)
6. C: Impacto en la confidencialidad (Confidentiality Impact)
7. I: Impacto en la integridad (Integrity Impact)
8. A: Impacto en la disponibilidad (Availability Impact)

Como estos modelos son entrenados únicamente con los datos propios de cada severidad, no se realiza aumentos de datos. Se aplica un Label Encoder a cada componente.

La arquitectura de estos modelos se basa en un clasificador multi-tarea utilizando el modelo **distilbert-base-multilingual-cased**. Para representar el objetivo multi-output, se implementa un ModuleDict (diccionario de módulos) que mapea cada componente CVSS a una capa lineal que contiene 768 neuronas intermedias y una cantidad de neuronas de output en función de la cantidad de clases para cada componente. Los hiperparámetros se visualizan en la tabla 4.

Parámetro	Valor
Épocas	5
Early Stopping Patience	3
Optimizer	AdamW
Learning Rate	5e-5
Weight Decay	0.01
Scheduler	ReduceLROnPlateau
Patience	1

Cuadro 4: Hiperparámetros de Entrenamiento para modelos de componentes CVSS

Así, el modelo para la severidad **high** ocupó 92.900 datos de entrenamiento, y 11.613 de validación; **medium** ocupó 104.657 datos de entrenamiento y 13.083 de validación; **critical** ocupó 33.795 datos de entrenamiento y 4.225 datos de validación; y **low** ocupó 5.000 datos de entrenamiento y 625 datos de validación.

El modelo para la clasificación de severidad ocupa todos los datos, ya sea sintéticos, duplicados o reales. Esto implica que se ocuparon en total 228.519 datos para entrenamiento y 48.968 datos para validación, con una estrategia de estratificación. La cantidad de datos por severidad corresponde a 130.822 datos totales **medium**, 116.125 para **high**, 87.694 para **low**, y 81.614 para **critical**. Las clases se codificaron utilizando un Label Encoder.

Este modelo ocupa el modelo **distilbert-base-multilingual-cased**. La arquitectura del modelo es el uso del transformer con una capa adicional de dropout, y la capa lineal clasificadora. Los hiperparámetros están indicados en la tabla 5.

6.2 Diseño de los instrumentos para validar la solución

Esta subsección indica de qué forma fueron evaluados los modelos y bajo qué métricas.

6.2.1 Recomendación de CWE

El conjunto de testeo considerado para validar la arquitectura de CWE es la combinación entre el conjunto de testeo para el modelo jerárquico superior y el inferior. Se genera el *ground truth* para comparar las

Parámetro	Valor
batch_size	32
patience	3
epochs	3
Tipo	AdamW
Learning rate (lr)	2e-5
Weight decay	0.01
Scheduler	ReduceLROnPlateau
Patience	2

Cuadro 5: Configuración de Entrenamiento del Modelo clasificador de severidad

predicciones con el valor real. Para representar un escenario real, solo se considera de input las descripciones preprocesadas de los CVE como input para hacer el flujo de la arquitectura.

Para la generación del *ground truth*, se establece un mapeo de las descripciones de CVE a su correcta descripción CWE, y otro mapeo entre la descripción de CVE y la capa jerárquica a la cual corresponde (siendo 1 la superior, y 0 la inferior).

Posterior a la inferencia de todos los datos de testeo, se presenta un conjunto de métricas para establecer su efectividad:

- Top-1 Accuracy: Precisión de acuerdo al primer elemento inferido, o bien, si el CWE predecido corresponde al del *ground truth*.
- Top-3 Accuracy: Precisión de acuerdo a los 3 primeros elementos inferidos, o bien, si el CWE real se encuentra dentro de los primeros 3 elementos predecidos.
- Top-5 Accuracy: Precisión de acuerdo a los 5 elementos inferidos.
- Precision: Mide la exactitud en los casos clasificados como positivos.
- Recall: Mide la capacidad de detectar todos los casos relevantes considerando a los falsos negativos y los casos positivos reales.
- F1 Score: Balance entre precisión y recall al ser su media armónica.

6.2.2 Recomendación de CVSS

El conjunto de testeo considerado para validar la arquitectura de CVSS es la combinación de los conjuntos de testeo para los modelos clasificadores de componentes CVSS. Se separa para cada modelo el conjunto de testeo respectivo, para luego fusionarlo y mezclarlo. Así, el modelo predice y se obtienen las métricas de rendimiento finales.

Así, se obtienen 12.904 datos para la severidad **medium**, 3.969 para **critical**, 751 para **low**, y 11.919 para **high**.

Las métricas en las cuales se testea la efectividad está separado en una métrica general, y otra específica para cada severidad. Cada severidad se evalúa con:

- Precision: Mide la exactitud en los casos clasificados como positivos.

- Recall: Mide la capacidad de detectar todos los casos relevantes considerando a los falsos negativos y los casos positivos reales.
- F1 Score: Balance entre precisión y recall al ser su media armónica.
- Accuracy: Mide la precisión en relación de los elementos predecidos y los casos reales sin distinguir clases.

6.3 Recolección y tratamiento de datos

La NVD posee una API que se usa para recolectar cada CVE que se haya guardado históricamente en la base de datos. La NVD posee en total 288.741 vulnerabilidades únicas al momento de la recolección. Esta recolección se hace mediante el uso de scripts con Python, almacenando toda la información en un DataFrame de la librería pandas y haciendo la distinción entre si estos CVE poseen el estándar de CVSS V1 o CVSS V2. El Dataframe finalmente contiene una cantidad de 532.490 filas, ya que se separó las descripciones considerando que pueden venir tanto en español como en inglés.

El preprocesamiento de los datos es igual tanto para la arquitectura para la recomendación de CWE como de CVSS. Este se hace posterior al tratamiento de los datos.

1. Convertir a minúscula: Transformar todo el texto a letras minúsculas para normalización.
2. Eliminar caracteres especiales, conservando caracteres propios del español: Eliminar símbolos y caracteres no alfabéticos (como #, \$), conservando letras acentuadas y signos de puntuación básicos.
3. Tokenizar: Dividir el texto en unidades lingüísticas individuales (palabras, símbolos o elementos significativos).
4. Eliminar *stopwords*: Eliminar palabras frecuentes pero poco significativas (artículos, preposiciones como 'el', 'de', 'y').
5. Lemmatizar: Reducir palabras a su forma canónica o de diccionario (ej: 'corriendo' → 'correr').
6. Stemmizar: Cortar prefijos y sufijos para obtener la raíz de la palabra (ej: 'corredor' → 'corr'), sin garantizar que sea una palabra válida.

6.3.1 Recomendación de CWE

Para el tratamiento de los datos para la recomendación de CWE, se comenzó eliminando los CVE que posean de valor de CWE de NVD-CWE-noinfo o NVD-CWE-Other, al no poseer información útil para el entrenamiento de los modelos. Asimismo, se elimina las filas con CWE nulo. Se hace la separación de múltiples CWE en el mismo campo para dejarlos en filas separadas para facilitar el manejo de las clases dentro del dataset, aunque generalmente entre los CVE con múltiples CWE, estos poseían relación jerárquica. Luego de esta separación, quedó en 405.051 elementos.

Se puede visualizar en la Figura 4, que el dataset es altamente desbalanceado con 307 clases. El CWE más común es CWE-79 que alcanza 65.107 vulnerabilidades asociadas. Así, dado que la investigación se basa en los CWE dentro de la Vista-1003, se consideran solo estos, los cuales abarcan el 88.82% del total de los CVE con 359.771 filas, donde el nivel jerárquico superior considera 37 CWE y el inferior considera 93.

Para solucionar el desbalance de clases, se estableció una cantidad objetivo de 4000 CVE para cada CWE. Se undersampleó los CVE con una cantidad mayor a este valor, y las clases con menos CVE se oversampleó

mediante la generación de datos sintéticos con el uso de la LLM GPT-4o-mini al ser el modelo más económico de OpenAI.

La cantidad de elementos luego del balanceo de clases se puede visualizar por nivel jerárquico en la Figura 6. En el programa para generar los datos sintéticos, se obtuvieron varias fallas respecto a la generación de datos y la posterior verificación de que se hubieran generado los 4000 elementos deseados. Se decidió no generar más datos debido a las limitaciones económicas.

Así, entre los dos niveles jerárquicos se obtuvieron un total de 401.124 CVEs, con 112.622 CVEs asociados al nivel jerárquico superior, y 288.502 al inferior.

6.3.2 Recomendación de CVSS

Para el tratamiento de los datos para la recomendación de CVSS, se basó desde el dataset original obtenido de la NVD. Para el contexto de este trabajo, se ocupó la versión 3.1 del CVSS, existiendo ya la versión 4.0, aunque esta última es reciente y no posee una cantidad de datos disponibles para el entrenamiento (131 datos en Mayo del 2025 para la versión 4.0 versus alrededor de 170.000 datos agrupados para la versión 3.1). Se filtraron aquellos campos nulos tanto en el campo de la severidad como en en la descripción.

Luego del tratamiento de datos, se obtuvieron 295.435 datos para el entrenamiento. De estos datos, se puede apreciar que existe en la figura 7 una gran desigual en los datos presentes en las severidades. Así, la severidad **medium** posee 130.819 datos, **high** posee 116.123 datos, **critical** posee 42.243, y **low** posee 6.250. Estos mismos datos fueron usados para el entrenamiento de los modelos de componentes de CVSS, ya que no es confiable generar sintéticamente estos datos usando LLM.

Para combatir el desbalance de clases, se generarán datos sintéticos para las descripciones de las clases con menos datos (**critical** y **low**). Se ocupó el modelo GPT-4o-mini, donde se generaban datos con un objetivo de tener 100.000 datos para la cantidad objetivo total en las clases a generar. Adicionalmente, los CVE no sintéticos de estas clases fueron duplicados. Después de fallos en la cantidad de elementos obtenidos en la generación, se llegó a una cantidad final de 81.614 datos únicos para la clase **critical**, y 81.444 datos únicos para **low**. Así, para la clasificación de severidad de CVSS se obtuvieron 416.255 datos como visualizado en la figura 3.

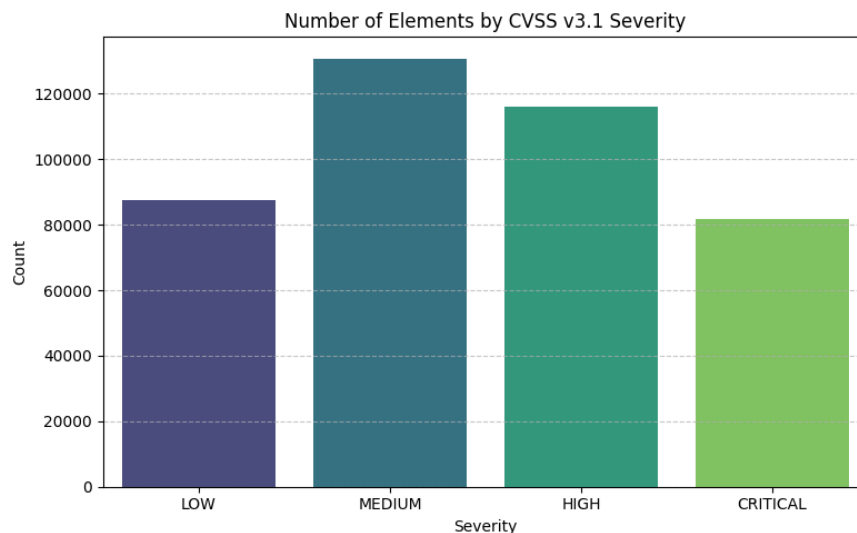


Figura 3: Distribución de elementos para entrenamiento de modelo de severidad de CVSS

6.4 Análisis de datos y resultados

En esta subsección se exponen los resultados obtenidos por los modelos de CWE y CVSS de forma separada y con las métricas anteriormente expuestas. Estos datos fueron obtenidos en base a los conjuntos de test.

6.4.1 Evaluación del Modelo de predicción de CWE

La evaluación del modelo de recomendación de CWE se realizó sobre un conjunto de 39,808 vulnerabilidades. Como muestra el Cuadro 8, el clasificador jerárquico (*layer classification*) obtuvo excelentes resultados globales con un *accuracy* de 0.9647, *precision* de 0.9671, *recall* de 0.9647 y *F1 score* de 0.9651. Los modelos específicos presentaron desempeños diferenciados: el *bottom cross-encoder* (Cuadro 6) logró una *precision* de 0.99, *recall* de 0.96 y *F1 score* de 0.97, mientras que el *top cross-encoder* (Cuadro 7) alcanzó 0.90 de *precision*, 0.98 de *recall* y 0.94 de *F1 score*. Estos resultados se complementan con la matriz de confusión mostrada en la Figura 8, que evidencia la robustez del enfoque jerárquico.

Para el clasificador final de CWE (Cuadro 9), los resultados demostraron una efectividad notable con *Top-1 Accuracy* de 0.7966 (79.66 %), indicando que en casi 8 de cada 10 casos la primera recomendación fue correcta. La capacidad predictiva mejoró al considerar más opciones, alcanzando *Top-3 Accuracy* de 0.9083 (90.83 %) y *Top-5 Accuracy* de 0.9316 (93.16 %). Las métricas adicionales confirmaron un balance adecuado entre precisión y cobertura: *precision* de 0.8005, *recall* de 0.7966 y *F1 score* de 0.7930, lo que valida la utilidad del modelo para la clasificación automática de vulnerabilidades. De modelo de baseline se ocupó una ANN con 3 capas densas y técnicas de regularización, esta ocupa vectorización TF-IDF.

Métrica	Valor
F1 Score	0.97
Precision	0.99
Recall	0.96

Cuadro 6: Métricas de modelo bottom cross-encoder

Métrica	Valor
F1 Score	0.8833
Precision	0.8948
Recall	0.8722

Cuadro 7: Métricas de modelo top cross-encoder

Métrica	Valor
Accuracy	0.9647
Precision	0.9671
Recall	0.9647
F1 Score	0.9651

Cuadro 8: Métricas del modelo Layer Classifier

Métrica	Valor	Baseline
Top-1 Accuracy	0.7966	0.6525
Top-3 Accuracy	0.9083	0.7987
Top-5 Accuracy	0.9316	0.8547
Precision	0.8005	—
Recall	0.7966	—
F1 Score	0.7930	0.6531

Cuadro 9: Rendimiento de la arquitectura propuesta

6.4.2 Análisis Cualitativo de Predicción de Componentes CVSS dado resultados

■ Caso 1: Predicción Correcta (CWE-330)

Descripción: The web service in VersionControlSystem 3.5.2 has an insecure RNG for session ID generation. Attackers can predict session IDs and hijack user sessions, compromising data integrity. This issue was addressed in version 3.5.3.

Valores predichos: CWE-330 (0.9995), CWE-407 (0.0001), CWE-662 (0.0000)

Análisis: El modelo identificó perfectamente el patrón de generación aleatoria insegura ("insecure rng session id generat"), mostrando alta precisión y clara diferenciación con CWEs menos relevantes.

■ Caso 2: Predicción Correcta (CWE-345)

Descripción: A vulnerability in the firmware update functionality of SmartCamera version 1.4.0 allows an attacker to perform a man-in-the-middle attack and inject malicious firmware, as the device does not properly validate the authenticity of the firmware source.

Valores predichos: CWE-345 (0.9992), CWE-682 (0.0006), CWE-754 (0.0006)

Análisis: Reconoció adecuadamente el problema de autenticidad en firmware ("proper valid authent firmwar sourc"), con alta confianza y alternativas conceptualmente distintas.

■ Caso 3: Error en Clasificación (CWE-665)

Descripción: In a popular payment processing system, improper validation of payment request parameters allows attackers to manipulate transaction amounts. This could lead to unauthorized transactions and financial losses.

Valores predichos: Capa Bottom (0.648) cuando era Top

Análisis: El modelo se equivocó, ya que se consideró como una falla técnica de bajo nivel cuando en realidad esta vulnerabilidad es un problema de validación de entrada y control de acceso a nivel lógico de la aplicación.

6.4.3 Evaluación del Modelo de Predicción de Componentes CVSS

La evaluación del modelo de clasificación de severidad CVSS se realizó sobre 29,543 vulnerabilidades. Como se detalla en los resultados globales, el sistema alcanzó un *accuracy* de 0.8580, con *precision* de 0.8590, *recall* de 0.8580 y *F1 score* de 0.8582. Un análisis por categorías de severidad reveló variaciones significativas en el desempeño:

Para vulnerabilidades críticas (Cuadro 10), el modelo mostró *precision* de 0.84, *recall* de 0.79 y *F1 score* de 0.81. En el caso de severidad alta (Cuadro 11), se registraron valores de 0.84 (*precision*), 0.79 (*recall*) y 0.85 (*F1 score*). Las vulnerabilidades de baja severidad (Cuadro 12) presentaron *precision* de 0.70, *recall* de 0.84 y *F1 score* de 0.76, mientras que para severidad media (Cuadro 13) los resultados fueron notablemente superiores con 0.89 de *precision*, 0.88 de *recall* y 0.89 de *F1 Score*. La Figura 9 muestra visualmente estas relaciones mediante la matriz de confusión, confirmando la capacidad del modelo para diferenciar efectivamente entre los distintos niveles de severidad.

Ahora para los modelos específicos de severidad para componentes CVSS, se obtuvo la *accuracy*, la *precision*, el *recall* y el *F1 Score* para cada componente.

En el Cuadro 14, se visualiza de forma general las métricas para cada componente en comparativa del modelo con el baseline. El modelo baseline es un ANN simple con 3 capas densas y regularización, en la cual se vectoriza el texto con embeddings TF-IDF de 5000 dimensiones.

En el Cuadro 15 se visualiza de forma específica de cada componente CVSS considerando su severidad.

Por último, considerando el flujo general considerando la arquitectura, se obtuvo un rendimiento promedio en la *accuracy* de 0.9185, *precision* de 0.9176, *recall* de 0.9185, y un *F1 score* de 0.9175. En el Cuadro 16 se puede visualizar estos resultados promedios en conjunto de los resultados finales con cada severidad. Estos resultados se visualizan asimismo en las Figuras 10 y 11.

Métrica	Valor
Precision	0.84
Recall	0.79
F1 Score	0.81

Cuadro 10: Métricas para severidad **critical**

Métrica	Valor
Precision	0.84
Recall	0.86
F1 Score	0.85

Cuadro 11: Métricas para severidad **high**

Métrica	Valor
Precision	0.70
Recall	0.84
F1 Score	0.76

Cuadro 12: Métricas para severidad **low**

Métrica	Valor
Precision	0.89
Recall	0.88
F1 Score	0.89

Cuadro 13: Métricas para severidad **medium**

Componente	Accuracy	Precision	Recall	F1 Score	Baseline Acc.	Baseline F1
Attack Vector	0.9359	0.9348	0.9359	0.9345	0.7354	0.6253
Attack Complexity	0.9628	0.9592	0.9628	0.9604	0.9457	0.9220
Privileges Required	0.8662	0.8651	0.8662	0.8648	0.5625	0.4071
User Interaction	0.9484	0.9483	0.9484	0.9484	0.6719	0.5413
Scope	0.9607	0.9606	0.9607	0.9606	0.7945	0.7064
Confidentiality Impact	0.8905	0.8907	0.8905	0.8900	0.5340	0.3724
Integrity Impact	0.8907	0.8912	0.8907	0.8905	0.3110	0.1608
Availability Impact	0.8929	0.8908	0.8929	0.8907	0.5351	0.3738

Cuadro 14: Métricas Generales por Componente CVSS v3.1 (Comparación con Baseline)

6.4.4 Análisis Cualitativo de Predicción de Componentes CVSS dado resultados

■ CVE-2022-47665

Descripción: Libde265 1.0.9 has a heap buffer overflow vulnerability in de265_image::set_SliceAddrRS (int, int, int)

Componentes (Predicho → Real): Attack Vector: N → L, Attack Complexity: L → L (correcto), Privileges Required: N → N (correcto), User Interaction: R → R (correcto), Confidentiality: H → H (correcto), Integrity: H → H (correcto), Availability: H → H (correcto), Scope: U → U (correcto).

Análisis: El principal error fue en Attack Vector (Network → Local). El modelo asumió incorrectamente un escenario remoto para esta vulnerabilidad en Libde265, cuando en realidad requiere acceso local. Todos los demás componentes fueron predichos correctamente, mostrando que el modelo identificó adecuadamente la gravedad del buffer overflow.

■ CVE-2023-31484

Descripción: CPAN.pm before 2.35 does not verify TLS certificates when downloading distributions over HTTPS.

Componentes (Predicho → Real): Attack Vector: N → N (correcto), Attack Complexity: L → H, Privileges Required: N → N (correcto), User Interaction: N → N (correcto), Confidentiality: H → H (correcto), Integrity: N → H, Availability: N → H, Scope: U → U (correcto).

Componente	Severidad	Accuracy	F1
Attack Vector	CRITICAL	0.9657	0.9800
	HIGH	0.9380	0.9375
	LOW	0.4768	0.3622
	MEDIUM	0.9463	0.9450
Attack Complexity	CRITICAL	0.9884	0.9918
	HIGH	0.9584	0.9574
	LOW	0.7808	0.7000
	MEDIUM	0.9671	0.9645
Privileges Required	CRITICAL	0.9060	0.9293
	HIGH	0.8844	0.8834
	LOW	0.4240	0.3318
	MEDIUM	0.8583	0.8573
User Interaction	CRITICAL	0.9635	0.9717
	HIGH	0.9517	0.9519
	LOW	0.7328	0.6880
	MEDIUM	0.9508	0.9508
Scope	CRITICAL	0.9640	0.9619
	HIGH	0.9630	0.9617
	LOW	0.7904	0.8550
	MEDIUM	0.9658	0.9658
Confidentiality	CRITICAL	0.9411	0.9588
	HIGH	0.9159	0.9185
	LOW	0.0896	0.1490
	MEDIUM	0.8900	0.8925
Integrity	CRITICAL	0.9313	0.9501
	HIGH	0.9084	0.9110
	LOW	0.1728	0.2857
	MEDIUM	0.8961	0.9020
Availability	CRITICAL	0.9271	0.9390
	HIGH	0.9110	0.9116
	LOW	0.1808	0.2974
	MEDIUM	0.8997	0.8988

Cuadro 15: Métricas por Severidad para cada componente CVSS

Cuadro 16: Métricas por Nivel de Severidad con Promedios

Severidad	Accuracy	Precision	Recall	F1 Score
CRITICAL	0.9484	0.9779	0.9484	0.9603
HIGH	0.9288	0.9303	0.9288	0.9291
MEDIUM	0.9218	0.9259	0.9218	0.9221
LOW	0.4560	0.7726	0.4560	0.4587
Promedio	0.9185	0.9176	0.9185	0.9175

Análisis: El modelo subestimó significativamente esta vulnerabilidad en CPAN.pm. Predijo Attack Complexity como Low cuando en realidad es High, ya que explotar la falta de verificación TLS requiere configurar un ataque MITM. Además, no reconoció los impactos en Integridad y Disponibilidad, considerándolos como None cuando en realidad son High.

- CVE-2023-30682

Descripción: Control de acceso inadecuado en Telecom en versiones anteriores a SMR Aug-2023 Release 1 que permite a los atacantes locales llamar a la API "silenceRinger" sin permiso.

Componentes (Predicho → Real): Attack Vector: N → L, Attack Complexity: L → L (correcto), Privileges Required: N → L, User Interaction: N → N (correcto), Confidentiality: H → N, Integrity: H → L, Availability: H → N, Scope: U → U (correcto).

Análisis: Este caso muestra los errores más graves. El modelo confundió completamente un ataque local en Telecom con uno remoto (Network → Local). Además, sobrestimó todos los impactos (de None/Low a High) y no consideró que se necesitan bajos privilegios. Esto sugiere un fuerte sesgo hacia vulnerabilidades críticas en componentes del sistema.

7 Conclusiones

Este trabajo abordó el problema de la automatización en la clasificación de vulnerabilidades mediante técnicas de interpretación textual, proponiendo modelos de Deep Learning para recomendar CWE y CVSS a partir de descripciones de vulnerabilidades preprocesadas. Los resultados demuestran que es posible optimizar significativamente el proceso de reportería en auditorías de ciberseguridad, reduciendo el tiempo que los auditores dedican a la clasificación manual y mejorando la consistencia en la evaluación de riesgos.

Para el modelo de clasificación de CWE, se implementó un enfoque jerárquico basado en cross-encoders y transformers (DistilBERT), logrando una Top-1 Accuracy del 79.66 % y un Top-5 Accuracy del 93.16 %. La estructura en dos niveles (Vista-1003) permitió manejar la complejidad de las clasificaciones CWE, aunque hay desafíos en el desbalance de clases.

En el modelo de predicción de CVSS, se desarrolló un sistema multi-output que primero clasifica la severidad y luego predice los componentes del vector CVSS llegando a tener una accuracy promedio del 91,85 %. Aún así, la severidad *low* presentó el menor rendimiento, dada la baja cantidad de datos presentes para esta clase.

Este trabajo resuelve las principales implicancias prácticas a las cuales se tenía de objetivo, las cuales era la automatización de las clasificaciones de CWE y CVSS en las auditorías, lo que permite que se tengan estándares más consistentes en estas evaluaciones, y adicionalmente, facilitar la vida a los auditores de ciberseguridad al clasificar las vulnerabilidades al ser más sencillo evaluar las sugerencias de los modelos que hacer el etiquetado manual.

Para trabajo futuro, se considerará el uso de modelos de LM como BERT para generar el *oversampling* de las descripciones de vulnerabilidades en vez del uso de modelos de pago usados para esta investigación, al ser modelos más económicos. Adicionalmente, se podría seguir entrenando los modelos para mejorar sus métricas respecto al conjunto de validación y testeo, para así que presente un mejor uso generalizado en escenarios reales. Por último, es necesario extender la clasificación a CVSS 4.0 cuando se disponga suficientes datos para hacer el entrenamiento.

Referencias

- [1] Cyver. *Pentest Reporting Tips: Spend Less Time on Reports*. 25 de oct. de 2022. URL: <https://core.cyver.io/pentest-reporting-tips-spend-less/> (visitado 25-10-2022).
- [2] MITRE. *CWE List Version 4.15*. 2024. URL: <https://cwe.mitre.org/data/downloads.html> (visitado 01-01-2024).
- [3] A. A. Anwar et al. "Cleaning the NVD: Comprehensive quality assessment, improvements, and analyses". En: *IEEE Transactions on Dependable and Secure Computing* 19.6 (2021), págs. 4255-4269.
- [4] National Institute of Standards and Technology (NIST). *NVD CWE Slice*. National Vulnerability Database. 2025. URL: <https://nvd.nist.gov/vuln/categories>.
- [5] Parasoft. *Todo sobre CWE: enumeración de debilidades comunes*. 2025. URL: <https://es.parasoft.com/blog/what-is-cwe/>.
- [6] National Institute of Standards and Technology (NIST). *Vulnerability Categories*. <https://nvd.nist.gov/vuln/categories>. Último acceso: 18 de julio de 2025. 2023. URL: <https://nvd.nist.gov/vuln/categories>.
- [7] National Institute of Standards y Technology. *Common Vulnerability Scoring System (CVSS)*. 2025. URL: <https://nvd.nist.gov/vuln-metrics/cvss>.
- [8] Forum of Incident Response y Security Teams (FIRST). *Common Vulnerability Scoring System (CVSS) v1*. 2005. URL: <https://www.first.org/cvss/v1/>.
- [9] Hispasec. *CVSS 4.0: Nueva versión de evaluación de vulnerabilidades*. 2023. URL: <https://unaaldia.hispasec.com/2023/11/cvss-4-0-nueva-version-de-evaluacion-de-vulnerabilidades.html>.
- [10] Cyberseguridad.net. *Calificación de vulnerabilidades CVSS*. Accedido: 2025-06-08. 2022. URL: <https://www.cyberseguridad.net/calificacion-de-vulnerabilidades-cvss>.
- [11] Pablo Huet. *Técnicas clave para el procesamiento de texto en NLP*. 2024. URL: <https://openwebinars.net/blog/tecnicas-clave-para-procesamiento-texto-nlp/>.
- [12] UKP Lab. *Aplicaciones de Cross-Encoders en Sentence Transformers*. 2025. URL: https://sbert.net/examples/cross_encoder/applications/README.html.
- [13] Jason Brownlee. *Deep Learning Models for Multi-Output Regression*. 2020. URL: <https://machinelearningmastery.com/deep-learning-models-for-multi-output-regression/>.
- [14] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: 1910.01108 [cs.CL]. URL: <https://arxiv.org/abs/1910.01108>.
- [15] Siddhartha Das et al. "V2W-BERT: A Framework for Effective Hierarchical Multiclass Classification of Software Vulnerabilities". En: oct. de 2021, págs. 1-12. DOI: 10.1109/DSAA53316.2021.9564227.
- [16] Stefano Simonetto et al. "Text2Weak: mapping CVEs to CWEs using description embeddings analysis". En: *4th Workshop on Artificial Intelligence-Enabled Cybersecurity Analytics*. 2024.
- [17] Jingyi Su y Yan Wu. "Refining CVE-to-CWE mapping with enhanced attention in BERT-based models". En: *Applied and Computational Engineering* 71 (2024), págs. 107-112.

- [18] Kethan Kota, Manjunatha A y Sree Vivek S. “CWE Prediction Using CVE Description - The Semantic Similarity Approach”. En: *Procedia Computer Science* 235 (2024). International Conference on Machine Learning and Data Engineering (ICMLDE 2023), págs. 1167-1178. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2024.04.111>. URL: <https://www.sciencedirect.com/science/article/pii/S1877050924007877>.
- [19] Joana Cabral Costa et al. “Predicting cvss metric via description interpretation”. En: *IEEE Access* 10 (2022), págs. 59125-59134.
- [20] Fan Shi et al. “XLNet-Based Prediction Model for CVSS Metric Values”. En: *Applied Sciences* 12.18 (2022). ISSN: 2076-3417. DOI: 10.3390/app12188983. URL: <https://www.mdpi.com/2076-3417/12/18/8983>.
- [21] Manjunatha A et al. “CVE Severity Prediction From Vulnerability Description - A Deep Learning Approach”. En: *Procedia Computer Science* 235 (2024). International Conference on Machine Learning and Data Engineering (ICMLDE 2023), págs. 3105-3117. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2024.04.294>. URL: <https://www.sciencedirect.com/science/article/pii/S1877050924009748>.
- [22] Elyes Manai, Mohamed Mejri y Jaouhar Fattahi. “Helping CNAs Generate CVSS Scores Faster and More Confidently Using XAI”. En: *Applied Sciences* 14.20 (2024). ISSN: 2076-3417. DOI: 10.3390/app14209231. URL: <https://www.mdpi.com/2076-3417/14/20/9231>.

A Anexo

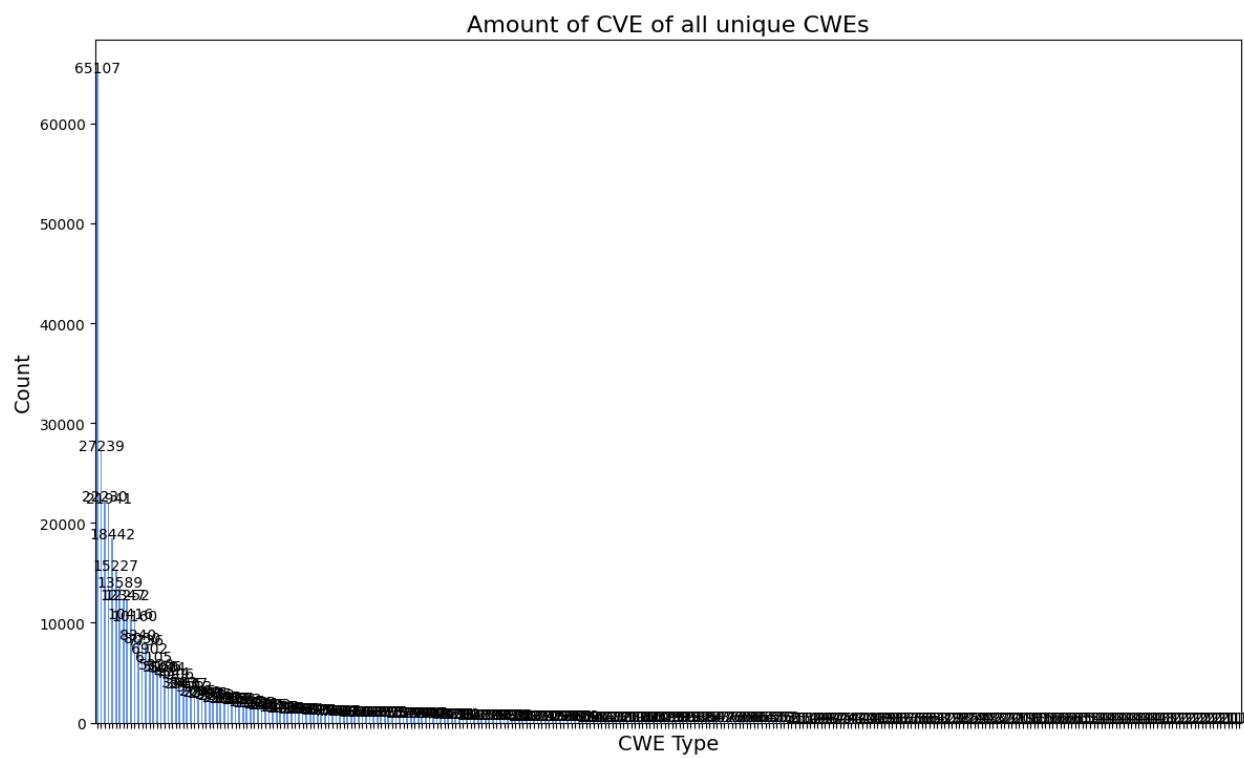


Figura 4: Gráfico que muestra la cantidad de CVE para todos los CWE.

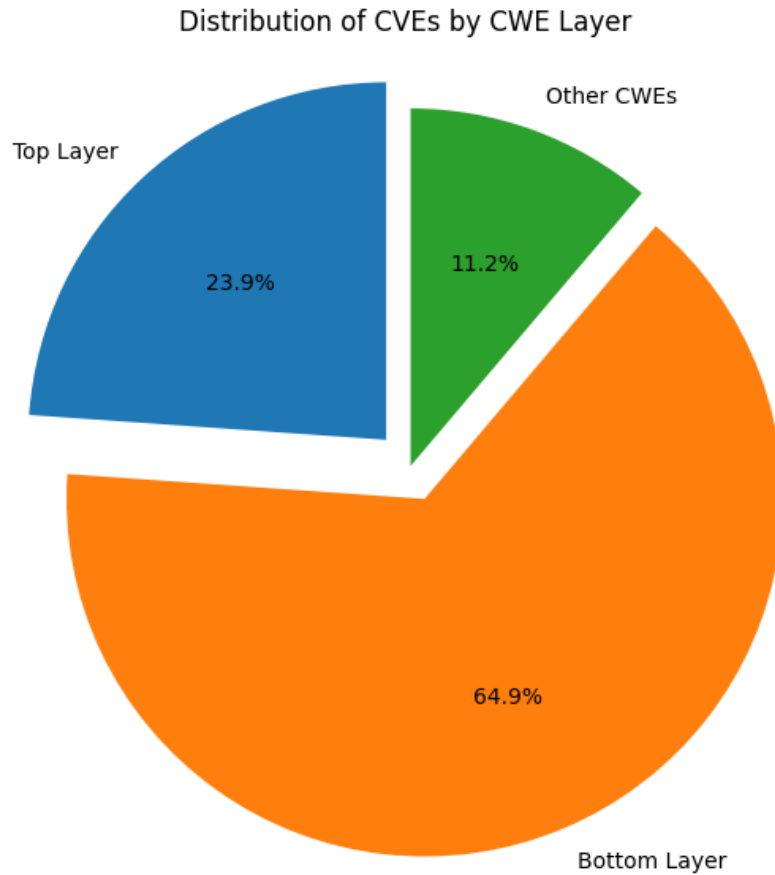


Figura 5: Proporción de CWE respecto a aquellos dentro de la Vista-1003 (considerando top y bottom layer).

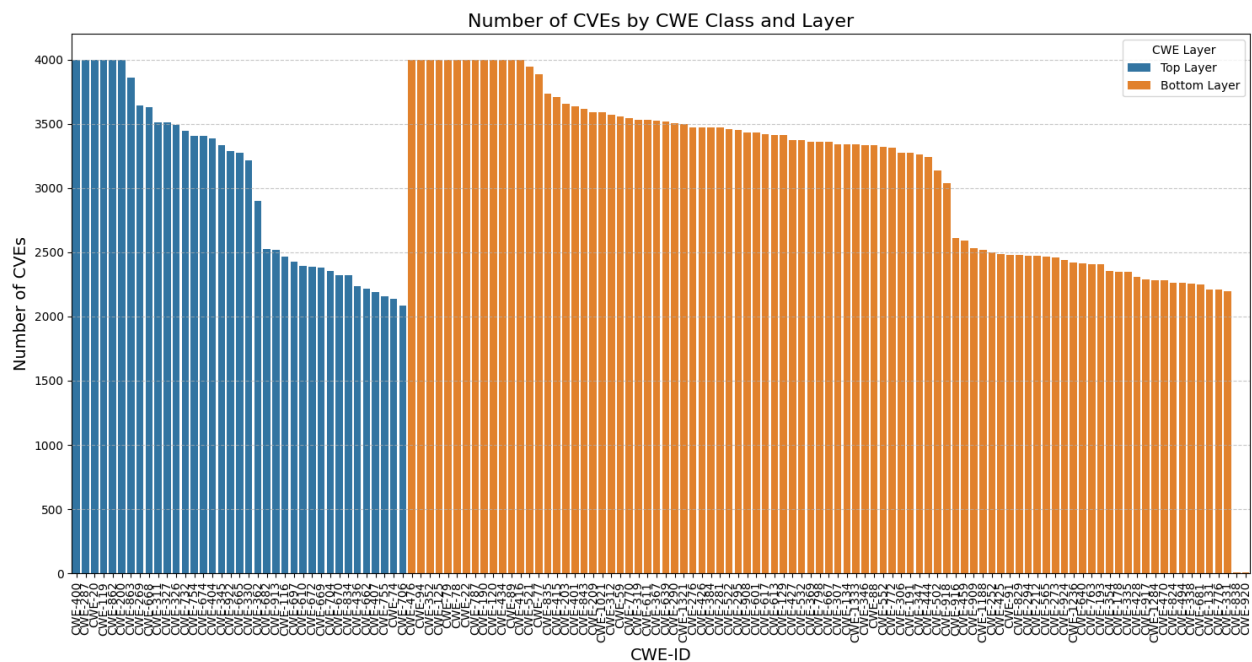


Figura 6: Cantidad de CVE para top y bottom layers luego de intento de balanceo de clases.

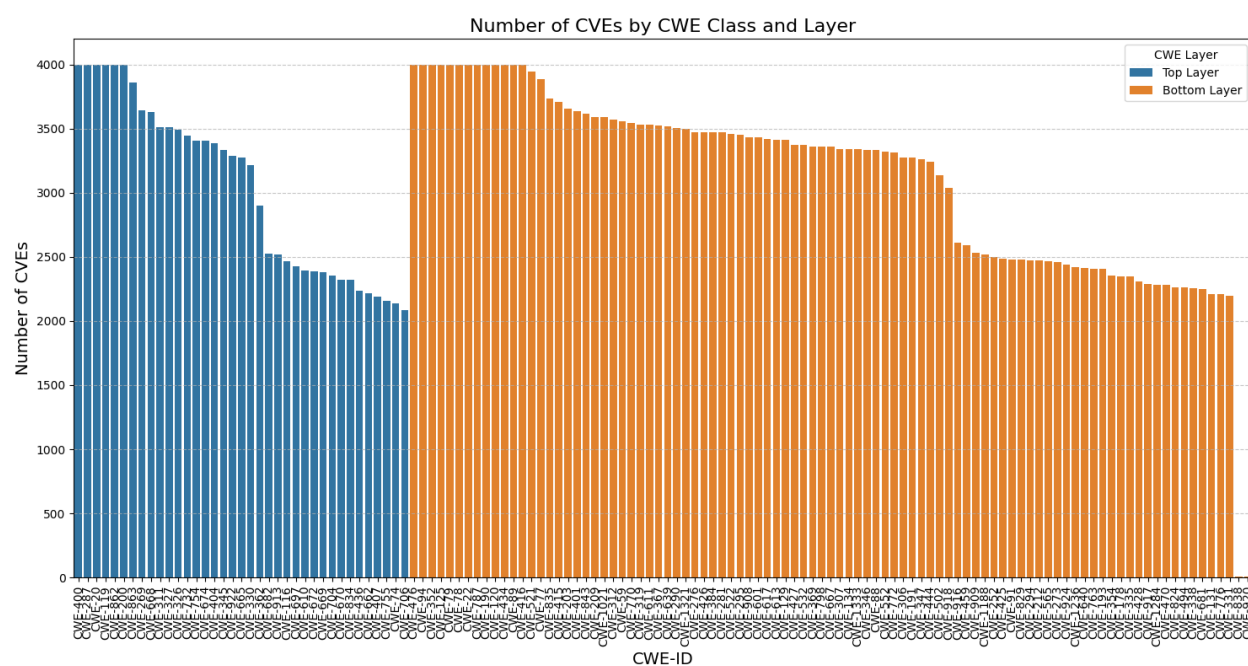


Figura 7: Distribución de componentes de CVSS para cada severidad

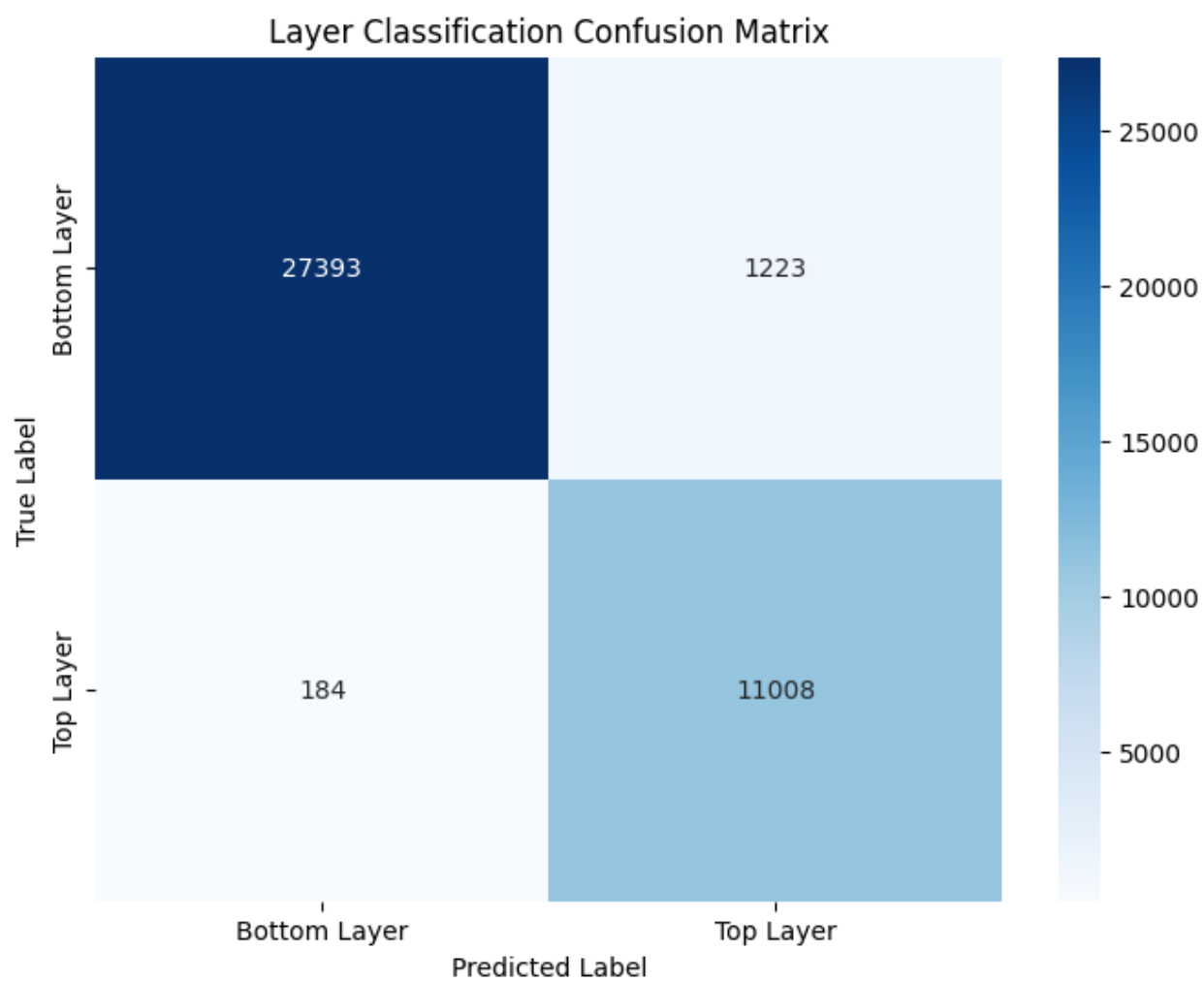


Figura 8: Matriz de confusión para clasificación jerárquico de CWE

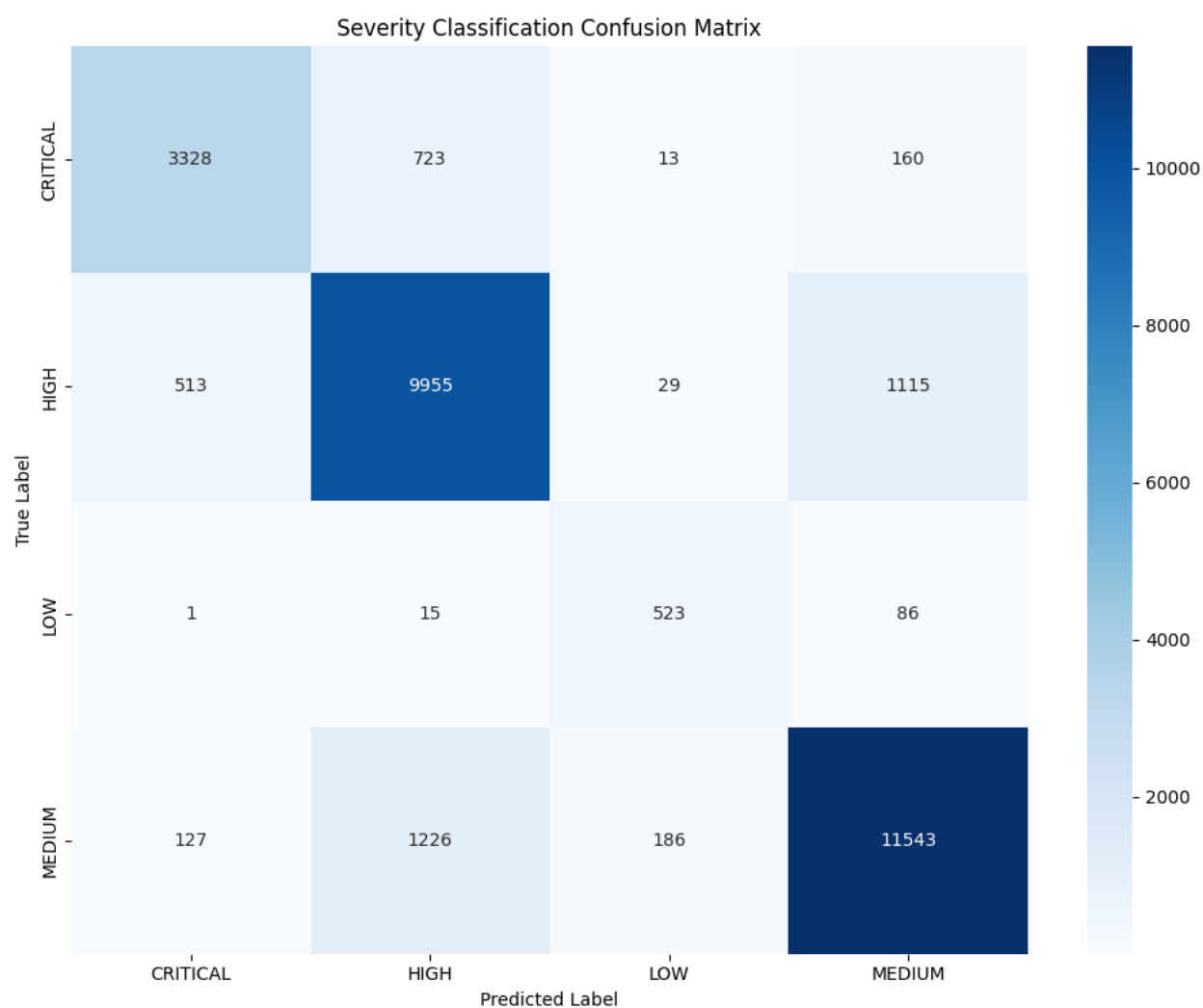


Figura 9: Matriz de confusión para clasificación de severidad CVSS

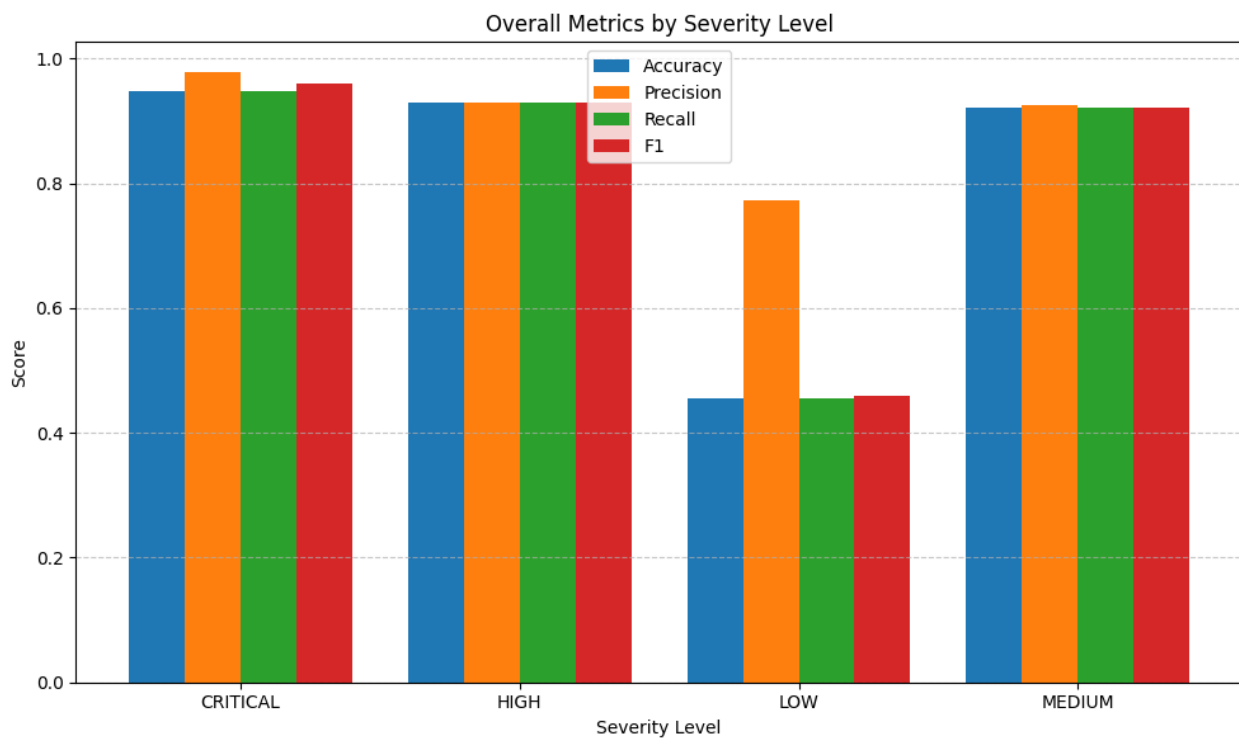


Figura 10: Métricas finales de modelos de CVSS respecto a severidad

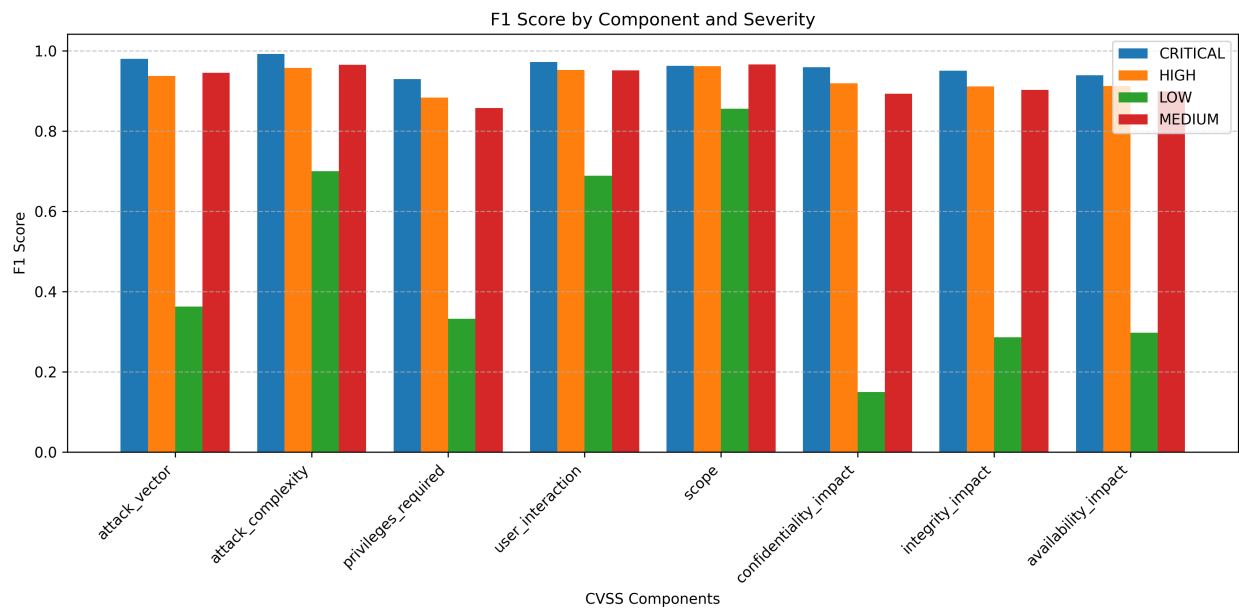


Figura 11: Métricas finales de modelos de CVSS respecto a severidad y componentes