Las técnicas de aprendizaje automático para tareas de ingeniería de software relacionadas con a ciberseguridad son cada ves mas populares.
En este survey se estudia los diferentes enfoques basados en el aprendizaje automatico, existentes y se muestra que tipo de representaciones se utilizaron en tareas para ciberseguridad y lenguajes de programación.
- Se ha descubierto que las representaciones de grafos son la categoria mas popular, y que las tokenizadores y los arboles de sintaxis abstracta (AST) son las dos representaciones mas populares en general.
- La tarea de deteccion de vulnerabilidades es la mas popular , y que el lengueje mas usado es C para la mayoria de tecnicas.
- Los modelos basados en secuencias son la categoría de modelos mas popular, y que las maquinas de vectores de soporte son el modelo mas popular en general.

1. Introduccion:
   Las vulnerabilidades de software son defectos que afectan a las propiedades de seguridad previstas por un sistema de software y que permiten a los atacantes hacer acciones maliciosas.
   A medida que los humanos tienden a volver su vida mas dependiente de la tecnologia los proveedores de software se ven forzados a diseñar software cada vez mas seguros.
   Existen varias practicas para abordar este tipo de problemas, y se pueden hacer en cada fase del ciclo de vida del software, aunque estas practicas ayudan de alguna manera u otra para evaluar y mejorar estos problema de seguridad, toman mucho tiempo, llegan a ser costosas y pueden ser propensas a errores. Entonces es aquí donde entra la IA y el ML que se aplican a tareas de ciberseguridad, como la detección de vulnerabilidades, deteccion de malware, etc. Entonces con esto nos damos cuenta que estas tecnicas son valiosas, ya que ayudan a mejorar la calidad del software y ahorramos tiempo , que de otro modo serian propensas a errores y llevarian mucho tiempo.
   Un ejemplo claro seria: un modelo que detecte vulnerabilidades antes de la implementacion, esto ahorraria tiempo y dinero , y a la par aumentaria la seguridad del sistema en conjunto.
   Los modelos de ML no son capaces procesar el código fuente como tal, por ende se busca la manera de pasarlo a datos numericos para ajustar las ponderaciones necesarias para el aprendizaje, esto quiere decir que debemos buscar la manera que el codigo fuente sea entendible de manera eficiente tanto en su informacion estructural como semanticamente, esto hace que el modelo aprenda y comprenda de los patrones subyacentes en el codigo. Por esto el codigo fuente es una parte crucial durante el desarrollo de tecnicas de ML, un codigo se puede representar como: AST (ARBOL DE SINTAXIS ABSTRACTA), CFG(grafico de flujo de control), o tokenizado.
   En este articulo veremos un analisis sistematico de la literatura, para comprender el estado actual de tecnica de representacion del codigo fuente para tecnicas basadas en ML para tareas de ciberseguridad, Investigaremos los lenguajes con los cuales se hacen

estas tecnicas, la popularidad de ciertas representaciones y los tipicos modelos de ML. El objetivo del estudio es permitir a los investigadores comprender las lagunas en este ámbito, en particular si hay ciertas tareas de ciberseguridad o lenguajes que son descuidados por las técnicas existentes.

2. Background: En esta seccion se aborda la terminologia para que el publico entienda de manera correcta el survey.
   2.1) ML para ing. de software segura
   Los ing de software se encargan de generar codigo nuevo de calidad, y son responsables de ver que errores hay, que vulnerabilidades existen y cualquier otro problema del codigo fuente, estas tareas puede llegar a ser tediosas, propensa a errores, y lentas. Una tarea de ciberseguridad es aquella que tiene como objetivo frustar a los posibles intrusos, por lo tanto las tareas que hacen los ing de soft. son aquellas que se encargan en resolver e identificar vulnerabilidades en el codigo que podria permitir al atacante aprovecharse del sistema. Es ahi donde entra el ML, la ia podria ayudar a mejorar en tiempo y eficiencia estos trabajos y asi mitigar los problemas que suceden en estos casos y mejorar la calidad del codigo fuente que se pone a disposicion del publico.
   2.2) Representación del código fuente e incrustaciones de código.
   Una representacion de codigo fuente captura sintaxis y la semantica del codigo fuente, de modo que el modelo de ML puede aprender caracterisiticas clave, para poder representar un codigo fuente hay muchas formas, algunas son: representacion de grafos, tecnicas de procesamiento de lenguaje natural, etc. Estas representacions ofrecen informacion diferente sobre el codigo y de esta manera hacen que el modeo de ML aprenda mas cosas sobre el. Por ejemplo las tecnicas de NLP se basan mas en la informacion semantica del codigo, por lo que el modelo aprendera relaciones semanticas, mas no relaciones estructuradas del codigo.
   Los modelos aprenden a base de incrustaciones vectoriales, estas son una forma de baja dimension para represnetar datos de alta dimension, para este caso las incrustaciones se crean a partir de la representacion del codigo fuente. La representacion del codigo fuente el el primer nivel de abstraccion del codigo fuente original, a esto se le llama fase extraccion de caracteristicas, los datos originales se transforman en datos de menor dimension, esto tiene como objetivo preservar la mayor cantidad de informacion de los posibles datos originales. El segundo nivel de asbtraccion son las incrustaciones vectoriales y son las que permiten aplicar tecnicas de ML a las representaciones transformandolas en representacion de forma numerica, que es la forma en la cual la maquina es capaz de entender. Nos centraremos en el primer nivel de asbtraccion, aunque hay varias tecnicas para generar incrustraciones como word2vec , doc2vec y GraphCode2vec
3. Related Work: Segun el conocimiento del creador de este survey, menciona que existen mchos mas como este survey pero la diferencia radica en que los otros articulos se centran principalmente en los modelos de ML utilizados, pero pocos mencionan o describen detalladamente las representaciones utilizadas en estos trabajos. El articulo se centra en principalmente en el tema de las representaciones utilizadas para el ML, en

tareas relacionadas a la seguridad. Se hace una comparativa del survey de los autores con otros surveys que ya existen, el enfoque de este survey es algo netamente referente a las representaciones de codigo fuente que se usan para tecnicas de ML para tareas de cybersguridad.

- Comparacion con Surveys Generales sobre ML en la ing de soft
- Comparación con estudios especificos en deteccion de vulnerabilidades.
- Comparación con revisiones sobre tecnicas de PLN.
- Comparacion con un estudio terciario.
- Comparacion con articulos que crean taxonomía
- Comparacion con surveys sobre ciberseguridad en general

4. Metodología: Para llevar a cabo nuestra revisión sistemática, que implica 3 actividades:
    1. Planificar: Definimos las preguntas de investigacion de este estudio y la consulta de busqueda utlizada para los artículos.
    2. Realizar: Durante esta fase buscamos 3 fuentes bibliograficas y descargamos todos los articulos en formato CSV. Luego de esto, se aplican criterios de inclusion y exclusion en tres fases para eliminar articulos hasta llegar al grupo final de articulos que incluyen en este estudio. Dos revisores leyeron cada articulo de forma independiente e hicieron un analisis extrayendo info. relevante para las preguntas de investigacion que se desarrollaron en la fase de planificacion. Se revisa y se resuelven las discrepancias para obtener el analisis final, se calcula el kappa de cohen para evaluar la fiabilidad de nuesyra evaluacion. La puntuacion que se obtiene es de 0.97 y esto indica que obtuvimos una concordancia casi perfecta en nuestro analisis.
    3. Presentar: En esta fase, analizamos los datos y los organizamos para poder responder a a las preguntas de investigacion que planteamos.

    4.1) Research Questions: A traves de la revision sistematica, pretendemos responder a las sgtes preguntas:

    - ¿Cuáles son las representaciones de codigo mas utilizadas?
    - ¿Ciertas tareas de ciberseguridad utilizan solo o principalmente un tipo de representacion del codigo fuente?
    - ¿Qué tareas de ciberseguridad cubren las técnicas que se han creado?
    - ¿Qué lenguajes de programación son los principales objetivos de las técnicas basadas en el aprendizaje automático para las tareas de ciberseguridad?
    - ¿Qué modelos se utilizan habitualmente con diferentes representaciones?

    4.2) Search Method: Se utilizó la siguiente cadena de busqueda para encontrar estudios primarios relacionados con la representacion o representaciones del codigo fuente para tareas de ciberseguridad basadas en ML.

    - (“machine learning” OR “deep learning” OR “artificial intelligence”) AND (“security” OR “vulnerability”) AND (“code”)

    Aunque estas cadena de busqueda suele ser muy general , arrojaró un resultado de 67 512 articulos , lo que se decidio es , en lugar de utilizar una cadena especifica que pudiera omitir una categoria de tareas o representaciones de seguridad de software, crearíamos una cadena general y eliminariamos

manualemente cualquier articulo que no cumplieran nuestros criterios de evaluacion (inclusión o exclusión)
Se busco en 3 base de datos:
- ACM library
- IEEE xplore
- Springer Link
- Tambien se busco en nueve conferencias de ML y NPL
4.3) Criterio de Inclusion y Exclusion: La tabla de abajo enumera los criterios de inclusion y exclusion aplicados a los articulos en multiples etapas para eliminar aquellos que no eran relevantes para el estudio.

Table 1. Inclusion and Exclusion Criteria

Inclusion Criteria	Exclusion Criteria
I1 Written between 2012 and May 2023 **I2** A full paper **I3** Focused on ML for cybersecurity tasks **I4** Contains information regarding the source code's representation	**E1** Duplicated studies **E2** Books, reference work entries, reference works **E3** Position papers, short papers, tool demo papers, keynotes, reviews, tutorials, and panel discussions **E4** Studies not in English **E5** Survey/comparative studies.

4.4) Paper Selection:
- Comenzamos con un totla de 67512 artículos
- Excluimos los duplicados y los que no se encontraban dentro del rango de años comprendido entre el 2012 y mayo del 2023, asi como los articulos que no eran completos. Esto dio un resultado de 52 836 articulos
- Posteriormente revisamos titulo, palabras clave, y el resumen de cada articulo para incluirlos o excluirlos en funcion de si se ajustaban a nuestros criterios, tras esta busqueda nos quedamos con 520 articulos.
- A continuación aplicamos los mismo criterios esta vez leyendo el articulo completo y fruto de este resultado no quedo un total de 141 articulos que se incluyen en este survey.

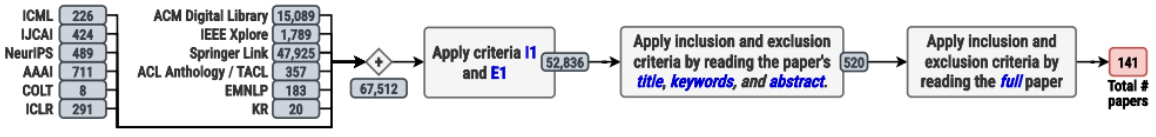

Fig. 1. Overview of the Three Stages of our Search Process

4.5) Extraccion de Data: El objetivo principal era responder a las 5 preguntas de investigacion, para ello, de cada uno de los articulos, extrajeron sistematicamente 4 datos fundamentales:
- La representacion utilizada
- La tarea de ciberseguridad
- Los lenguajes de programacion
- El tipo de modelo de IA
¿Como manejaron los detalles de la "representacion"?
Aqui los investigadores establecieron una regla muy importante para ser consistentes:

- Regla de "Un paper, multiples representaciones"
- Regla de la "representacion final"

¿Que hicieron ademas de responder las preguntas?

Los investigadores no se detuvieron ahi, dieron un paso mas allá para ofrecer un analisis mas completo:

- Extraccion de Metricas de Rendimiento
- El proposito de extraccion extra: (agregar y comparar)

5. RQ1 Results: What Are the Commonly Used Source Code Representations?:
   Los autores identificaron 39 representaciones distintas, las cuales agruparon en 4 categorias principales:
   - Basadas en arboles:
     - Árbol de Sintaxis Abstracta (AST - Abstract Syntax Tree):
     - Árbol de Análisis Sintáctico (Parse Tree):
     - AST+
   - Basadas en grafos:
     - Grafo de Flujo de Control (CFG - Control Flow Graph):
     - Grafo de Dependencia de Programa (PDG - Program Dependence Gxraph)
       - Grafo de Flujo de Datos (DFG - Data Flow Graph)
     - Grafo de Llamadas (Call Graph)
     - Grafo de Propiedades de Código (CPG - Code Property Graph)
     - Otras representaciones de grafos
   - Representaciones Lexicas:
     - Tokenizador (Tokenizer):
     - iSeVC y sSyVC
     - Fragmento de Contrato (Contract Snippet)
   - Representaciones Miscelaneas
     - Imagen (Image)
     - Métricas de Código (Code Metrics)
     - Gadgets de Código (Code Gadgets)
     - Secuencias de Códigos de Operación (Opcode Sequences):
     - Expresión Regular (Regular Expression)
     - Información de la Aplicación (Application Information) y Llamadas a API (API Calls
6. RQ2: Do Certain Tasks Only Use or Mostly Use One Type of Representation? Este pregunta analiza la relacion entre las diferentes representaciones de codigo fuente y las tareas especfificas de ciberseguridad para las que se utilizan. La conclusión principal es que, si bien existen preferencias claras para ciertas tareas, la elección de una representación a menudo implica un equilibrio entre la robustez de la información que proporciona y los recursos computacionales necesarios para generarla y procesarla.
   - Deteccion de vulnerabilidades: La tareas mas común, dado que es la tarea mas investigada, es tambien la que utiliza la mayor variedad de representaciones.

        - Tokenizadores y Árboles de Sintaxis Abstracta (ASTs)
        - Representaciones Basadas en Grafos
        - Representaciones Específicas para Vulnerabilidades:
    - Deteccion de malware:
        - Grafo de Flujo de Control Interprocedimental (ICFG), Grafo de Llamadas y CADG (Grafo de Dependencia de API Contextual):
    - El equilibrio: Necesidades vs. Recursos
      - Un **Grafo de Propiedades de Código (CPG)** es extremadamente robusto, ya que combina AST, CFG y PDG, pero es muy complejo y costoso de generar para miles de muestras de código.
      - En contraste, si el objetivo es detectar un ataque de inyección, un **Grafo de Flujo de Datos (DFG)** podría ser suficiente, ya que proporciona la información necesaria sobre el flujo de datos sin ser excesivamente complejo o intensivo en recursos.
      Hallasgos clave de la RQ2:
      - La deteccion de vulnerabilidades es la tarea mas popular
      - Ciertas representaciones son altamente especializadas
      - Hay una clara preferencia para la deteccion de malware
7. RQ3: What Cybersecurity Tasks Are Covered by the ML-Based Techniques? Para organizar y comprender el alcance de las tareas de ciberseguridad abordadas por las tecnicas ML, los autores clasificaron las distintas tareas dentro del marco del ciclo de Proceso Unificado Racional, un proceso del desarrollo del software, de las nueve disciplinas del ciclo RUP, se observo que las tareas de ciberseguridad analizadas encajan en 5 categorias. La deteccion de vulnerabilidades es la tarea mas popular, siendo este con 75 articulos de los 141 en total
    - Analisis y Diseño: Tares que hay en esta fase.
        - Deteccion, prediccion y clasificacion de malware.
        - Analisis de seguridad, aplicado especialmente a contratos inteligentes
    - Implementacion
        - Reparacion de vulnerabilidades
        - Prediccion de vulnerabilidades
    - Pruebas (Testing)
        - Deteccion de vulnerabiliadades
        - Deteccion de mal uso de criptografia
        - Deteccion, clasificacion, filtrado, desofuscacion y localizacion de codigo malicioso
        - Deteccion de comportamiento malicioso
        - Deteccion de fugas de contraseña y ataques de inyeccion
        - Deteccion de ataques de reentrada
        - Analisis, clasificacion, localizacion, y extrapolacion de vulnerabilidades
        - Deteccion de clones de codigo vulnerable
        - Descubrimiento de vulnerabilidades API no protegidas

    - Coincidencia de codigo fuente binario para tareas como deteccion de software.
  - Configuracion y Gestion del cambio
    - Clasificacion de commits de seguridad
    - Identificacion de commits que introducen o corrigen vulnerabilidades
    - Deteccion de parches de seguridad
  - Entorno
    - Detección de paquetes maliciosos
    Importante:
  - Detección vs. Localización: Las tareas de *detección* se centran en identificar de forma general que existe una característica (por ejemplo, un fragmento de código vulnerable), mientras que la *localización* se enfoca en señalar exactamente dónde ocurre esa característica (por ejemplo, la línea específica de la vulnerabilidad).
  - **Predicción vs. Detección:** Las técnicas de *predicción* buscan encontrar problemas antes de que se manifiesten en un sistema lanzado oficialmente. En cambio, las técnicas de detección_ se enfocan en identificar problemas que ya existen en un sistema en funcionamiento.
8. RQ4: What Programming Languages Are Predominantly Targeted by the ML-Based Techniques for Cybersecurity Tasks?

Table 3. Languages Supported by Existing Techniques

Lang.	#Papers	Lang.	#Papers	Lang.	#Papers	Lang.	#Papers	Lang.	#Papers	Lang.	#Papers	Lang.	#Papers
C	81 (57.4%)	JS	12 (8.5%)	Python	6 (4.3%)	C#	2 (1.4%)	Gecko	1 (0.7%)	Powershell	1 (0.7%)	SM	1 (0.7%)
C++	50 (35.5%)	Solidity	12 (8.5%)	CSS	3 (2.1%)	SQL	2 (1.4%)	Go	1 (0.7%)	Ruby	1 (0.7%)	XML	1 (0.7%)
Java	36 (25.5%)	PHP	8 (5.7%)	Rust	3 (2.1%)	TS	2 (1.4%)	HTML	1 (0.7%)	Smali	1 (0.7%)	XUL	1 (0.7%)

JS = JavaScript; TS = TypeScript; SM = SpiderMonkey.

Esta seccion analiza que LP son los mas estudiados por las tecnicas de ciberseguridad basadas en ML
- C y C++ son los mas populares: C es el lenguaje mas popular con un total de 81 articulos cubiertos (57.4%), seguido de C++ cubierto por 50 (35.5%). La popularidad de ambos lenguajes se debe a dos factores principales: la gran disponbilidad de conjunto de datos para tareas de ciberseguridad y el mayor riesgo inherente de vulnerabilidades relacionadas con la memoria en estos lenguajes.
- Enfoque en Java y Solidity: Se observa un notable enfoque en tareas de seguridad para aplicaciones de Android, donde los estudios se centraron exclusivamente en aplicaciones escritas en **Java**. Además, debido a la creciente popularidad de los contratos inteligentes, varias técnicas se desarrollaron para Solidity, el lenguaje utilizado para escribirlos.
- Una brecha sorprendente en Python y JS: A pesar de ser lenguajes extremadamente populares entre los desarrolladores, hay sorprendentemente pocas técnicas de seguridad para **Python** (solo 6) y **JavaScript** (solo 12). Los autores sugieren que esto podría deberse a la falta de conjuntos de datos disponibles para que los investigadores entrenen y prueben sus modelos. Esta es considerada una brecha que debería ser abordada en futuros trabajos
- Ausencia de herramientas agnósticas al lenguaje: Ninguna de las técnicas analizadas

es "agnóstica al lenguaje", lo que significa que cada herramienta fue creada para uno o, como mucho, unos pocos lenguajes específicos. Crear una herramienta que funcione para cualquier lenguaje es muy difícil debido a la gran variedad de paradigmas de programación existentes.

9. RQ5: What Models Are Commonly Used with Different Representations?
10. Threats to Validity
11. Final Considerations