**Ciberseguridad con Inteligencia Artificial**



Dr. Vitali Herrera Semenets – CENATAV, La Habana, Cuba ([vherrera@cenatav.co.cu](mailto:vherrera@cenatav.co.cu))

MSc. Felipe Antonio Trujillo Fernández – IBERO, Ciudad de México, México (<felipe.trujillo@ibero.mx>)

Dr. Lázaro Bustio Martínez – IBERO, Ciudad de México, México (<lazaro.bustio@ibero.mx>)

Coordinación de Ciencia de Datos - Departamento de Estudios en Ingeniería para la Innovación – Ibero

Primavera 2024

Sesión 4

# **Introducción**

En el contexto de la seguridad cibernética, la detección de aplicaciones maliciosas (APKs) representa un desafío crítico. Con el crecimiento exponencial de las aplicaciones móviles y su uso generalizado, es fundamental identificar y mitigar las amenazas potenciales que pueden comprometer la seguridad y privacidad de los usuarios. En esta práctica, se introducirá a los participantes en la detección de APKs maliciosas explorando técnicas de Aprendizaje Automático y los conocimientos adquiridos durante el taller.

# **Objetivo**

Aplicar técnicas y algoritmos de Aprendizaje Automatizado y clasificación de datos para la detección de APKs maliciosas.

# **Indicaciones**

1. Analiza el documento “*Android manifest file permissions.pdf*” para conocer los permisos que Android pone a disposición de los desarrolladores. También revisar el documento “*A Comprehensive Analysis of the Android Permissions System.pdf*” para entender la arquitectura de permisos del sistema Android.
2. Obtención de datos.
   * Descargue el dataset “*apk\_train.csv*” del sitio web del taller.

Este dataset está compuesto por 400 muestras de APKs, de las cuales 200 son maliciosas (type=1) y 200 benignas (type=0). Cada una de estas muestras se representa mediante cada uno de los permisos que Android pone a disposición de los desarrolladores. Cada una de las muestras de APK, si requiere de un permiso, se marca con 1 la celda correspondiente; y 0 en caso contrario.

1. Realiza el Análisis Exploratorio de Datos para entender la naturaleza de los datos.
   * Codifica las etiquetas 0 y 1 de la columna “type” como “*benigno*” y “*maligno*” respectivamente.
   * Analiza el balance de instancias por clase para determinar la distribución de los datos.
   * Determine la APK que menos y más permisos requieren las APKs, y cuáles son estos. Determine en promedio la cantidad de permisos que las aplicaciones requieren, y cuáles son estos.
   * Obtenga la matriz de correlación entre las columnas de los datos. Represente la matriz de correlación mediante un mapa de calor. ¿Qué se puede concluir?
2. Aplica un algoritmo de agrupamiento (por ejemplo, KMeans) para agrupar las APKs de los datos.
   * Visualiza los grupos obtenidos.
   * Entender la naturaleza de los grupos:
     1. Analiza las características de los grupos obtenidos.
     2. Identifica patrones comunes en cada grupo.
   * El dataset “*conn250k\_anomaly.csv*” contiene las etiquetas reales para cada transacción en “*conn250k.csv*”. Cargue “*conn250k\_anomaly.csv*” y compare las etiquetas asignadas (grupos) por el algoritmo de agrupamiento y las etiquetas reales. ¿Qué se puede concluir al respecto?
3. Selección de los algoritmos de clasificación a emplear.
   * Investiga al menos 5 algoritmos de clasificación adecuados para la detección de malware (por ejemplo, SVM, Random Forest, Regresión Logística, etc.). Elige los algoritmos que consideres más apropiados y justifica tu elección.
4. Entrenamiento y Evaluación de Modelos.

Divide el conjunto de entrenamiento en subconjuntos para validación cruzada[[1]](#footnote-1). Entrena los algoritmos seleccionados mediante un pipeline[[2]](#footnote-2) de Python utilizando validación cruzada y evalúa su rendimiento utilizando métricas como Precision, Recall y F-Score.

1. Selección del Mejor Clasificador.

Compara los resultados de los modelos de clasificación entrenados y determina cuál ofrece el mejor rendimiento. Explica las razones detrás de la elección del mejor clasificador.

1. Implementación del Modelo de Clasificación.

Utiliza el mejor clasificador para crear un modelo de detección de malware. Demuestra cómo se puede emplear este modelo para determinar si una APK desconocida es maliciosa o benigna.

1. Luego de realizar el ejercicio, ¿a qué conclusión se puede llegar?

1. La validación cruzada es una técnica utilizada en el aprendizaje automático para evaluar el rendimiento de un modelo predictivo. Consiste en dividir el conjunto de datos en subconjuntos de entrenamiento y prueba de manera iterativa. En cada iteración, se entrena el modelo en una parte de los datos (conjunto de entrenamiento) y se evalúa en otra parte (conjunto de prueba). Esto se repite varias veces, utilizando diferentes combinaciones de datos de entrenamiento y prueba, lo que permite obtener una estimación más robusta del rendimiento del modelo en datos no vistos. La validación cruzada ayuda a evitar el sobreajuste y proporciona una evaluación más confiable del modelo. [↑](#footnote-ref-1)
2. <https://www.turing.com/kb/building-ml-pipeline-in-python-with-scikit-learn> [↑](#footnote-ref-2)