2024

Análisis de vectores de ataque a dispositivos médicos

PRACTICA MACHINE LEARNING Y CIBERSEGURIDAD

JOSE BENITO IBARRIA TOPETE

VII BOOTCAMP DE CIBERSEGURIDAD FULLSTACK | Keepcoding.io

1.- Descripción del caso de uso

El problema central radica en la vulnerabilidad de los dispositivos del Internet de las Cosas Médicas (IoMT) ante ciberataques, lo cual es crítico dado el carácter sensible de los datos de salud, el apalancamiento que los actores maliciosos podrían tener para negociar los “rescates” y principalmente la vida humana. La conexión de dispositivos médicos críticos a redes amplias sin las salvaguardas adecuadas crea oportunidades únicas para estos actores maliciosos. Esto se agrava por el hecho de que muchos protocolos de comunicación existentes no están diseñados específicamente para dispositivos médicos conectados, lo que requiere una evaluación y clasificación rigurosas de las tecnologías de comunicación IoT en términos de seguridad.

Fundamentalmente necesitamos mejorar la detección de comportamientos maliciosos en dispositivos IoMT, al utilizar técnicas avanzadas de Machine Learning podríamos mejorar nuestra capacidad para asegurar, en principio la vida humana, la protección de datos sensibles de salud y por supuesto evitar que una empresa tenga que “negociar” su supervivencia con actores maliciosos, todo lo anterior resulta muy delicado y es una situación que merece una atención preponderante. La motivación viene dada por la creciente demanda de estos dispositivos en el mercado médico, con un aumento en la distribución y uso de todos estos lógicamente se incrementa de forma exponencial la amenaza de ciberataques y el impacto potencial a la vida del paciente, la reputación de la empresa, en la privacidad y en la seguridad de los datos de los pacientes. Los beneficios de que un modelo de Machine Learning aplicados a estos dispositivos son, en general una mayor precisión en la detección de ataques, la minimización de falsos positivos y la consecuente conservación de todo lo que estos ataques amenazan.

Actualmente, todo lo relacionado con estos la seguridad en estos dispositivos se basa en métodos tradicionales y con las mismas fallas puntuales que, cualquier otro dispositivo de TI, sin embargo, dado el contexto de lo que se “está jugando en este rubro” es obligatoria la seriedad y buscar mejorar significativamente con modelos que, en esencia, sean más robustos, como pueden ser los modelos de Machine Learning o todos aquellos modelos predictivos automatizados.

Una de las aristas por las cuales se podría abordar esta sensible situación es mediante el análisis de comportamientos maliciosos en el tráfico de red en estos dispositivos de IoMT que podrían detectarse con una mayor precisión mediante técnicas de Machine Learning, ya se ha estado trabajando en ello, existe la propuesta del Instituto Canadiense para la ciberseguridad (CIC) cuyo trabajo “Attack Vectors in Healthcare Devices – A MultiProtocol Dataset for Assessing IoMT Device Security” que, básicamente y a manera de resumen se enfocó en el análisis de

Comportamiento de Dispositivos IoMT, donde la investigación se centró en analizar el comportamiento de los dispositivos IoMT desde el momento en que se unen a la red. También realizaron una evaluación Multidimensional ya que fueron más allá de la creación de datasets para evaluar la eficacia de varios algoritmos de ML en la detección y clasificación de ciberataques IoMT. Esencialmente evaluaron técnicas como Regresión Logística, Adaboost, Random Forest y Redes Neuronales Profundas, el estudio no solo estableció un punto de referencia para el estado actual del ML en la seguridad de IoMT, sino que también abre caminos para futuras exploraciones en optimización de algoritmos e ingeniería de características, como lo estamos abordando hoy nosotros. La evaluación de ML en la Identificación de Diferentes Ataques fue otro de los puntos que tuvieron en cuante los experimentos realizados evaluaron el rendimiento de cuatro técnicas de ML: Regresión Logística (LR), Random Forest (RF), Adaboost (AD) y Redes Neuronales Profundas (DNN), en escenarios de clasificación binaria, clasificación categórica y clasificación multiclase. Los resultados generales mostraron que todas las técnicas de ML tuvieron un buen desempeño en varios escenarios, aunque se observó una disminución del rendimiento en algunos casos. Si es cierto que existe un trabajo previo, buscamos implementar y explorar nuevos algoritmos de ML, específicamente Redes Neuronales Convolucionales (CNN), para analizar patrones complejos en el tráfico de red IoMT y detectar anomalías indicativas de ciberataques. Este enfoque permitirá una detección más precisa y adaptativa de amenazas en tiempo real.

2.1 Descripción de Datos

Los datos para nuestro proyecto provienen del conjunto de datos CICIoMT2024, que incluye una variedad de ataques ejecutados contra dispositivos IoMT reales y simulados, utilizando protocolos como Wi-Fi, MQTT y Bluetooth. Este rico conjunto de datos abarca diferentes categorías de ataques (DDoS, DoS, Recon, MQTT y Spoofing), así como el tráfico benigno, ofreciendo una diversidad de patrones de tráfico de red para entrenar y evaluar nuestro modelo CNN.

2.2 Procesamiento de los Datos

Para adaptar los datos al modelo CNN, se realizarán los siguientes pasos de procesamiento:

- Conversión de Formato: Transformar los datos capturados en formato PCAP a un formato estructurado (por ejemplo, CSV) que facilite la manipulación y análisis de los datos. Esta conversión se centrará en extraer características relevantes del tráfico de red que sean significativas para la detección de ataques, como tamaño del paquete, banderas TCP, protocolos utilizados y patrones temporales.

- Limpieza de Datos: Se limpiarán los datos para eliminar cualquier registro incompleto o irrelevante, asegurando que el modelo CNN trabaje con información precisa y completa. Esto incluye la normalización de las características para que tengan una escala común, crucial para el rendimiento de las CNN.
- Etiquetado de Datos: Cada instancia de datos será etiquetada como "benigna" o según el tipo específico de ataque que representa, basándose en la clasificación proporcionada en el conjunto de datos CICIoMT2024.

2.3 Transformación de los Datos

Dado que las CNN son ampliamente utilizadas para el análisis de imágenes, un desafío único es transformar los datos de tráfico de red en un formato que sea interpretable para una CNN. Aquí hay algunas estrategias que podemos emplear:

- Representación Visual de Datos: Transformar las características numéricas de los datos de tráfico de red en imágenes o representaciones visuales. Por ejemplo, se pueden crear imágenes de matrices donde cada fila representa una característica diferente y cada columna representa un momento en el tiempo, o se pueden utilizar técnicas de embedding para mapear los datos a un espacio que pueda ser visualmente representado.
- Aumento de Datos: Para mejorar la robustez y generalización del modelo CNN, se aplicarán técnicas de aumento de datos a las representaciones visuales de los datos de tráfico de red. Esto puede incluir la rotación, el escalamiento y la traslación de las imágenes de datos.
- División de Datos: Los datos serán divididos en conjuntos de entrenamiento, validación y prueba. Este paso es crucial para evaluar la capacidad del modelo CNN de generalizar a datos no vistos anteriormente.

Este enfoque no solo nos permitirá aplicar modelos de CNN a un problema de seguridad de redes tradicionalmente numérico, sino que también abre nuevas vías para la exploración de patrones complejos y la detección de anomalías en el tráfico de red IoMT. Al emplear CNN, esperamos capturar relaciones espaciales entre las características de los datos que métodos más tradicionales podrían pasar por alto, ofreciendo una mejora significativa en la detección de ciberataques en dispositivos IoMT.

Algoritmos

En la esencia de nuestro proyecto se encuentra la adopción y adaptación de Redes Neuronales Convolucionales (CNN) para el análisis y clasificación del tráfico de red de dispositivos IoMT, con el objetivo de identificar comportamientos maliciosos y prevenir ciberataques. Este enfoque nos permite explorar la capacidad de las CNN para capturar y aprender patrones complejos dentro de los datos que los métodos tradicionales podrían no detectar eficientemente.

3.1 Selección de Algoritmos:

Hemos seleccionado las CNN como el núcleo de nuestra metodología debido a su probada eficacia en el análisis de datos estructurados visualmente. La arquitectura será diseñada y ajustada específicamente para manejar las características únicas del tráfico de red IoMT, teniendo en cuenta la diversidad de los ataques y el tráfico benigno presente en el conjunto de datos CICIoMT2024. Además, consideraremos comparar el rendimiento de nuestras CNN con otros modelos de aprendizaje automático como parte de nuestra evaluación para validar nuestra elección.

3.2 Preparación para la Prueba:

El conjunto de datos CICIoMT2024 ha sido dividido estratégicamente en subconjuntos de entrenamiento, validación y prueba. Esta división asegura que el modelo sea entrenado en un amplio rango de datos, mientras que el conjunto de validación "hold-out" nos permite ajustar los hiperparámetros y la arquitectura del modelo sin comprometer la objetividad de nuestra evaluación. El conjunto de prueba, utilizado solo al final del proceso, servirá para evaluar la capacidad de generalización del modelo.

3.3 Evaluación del Modelo:

La evaluación se centrará en métricas clave como la precisión, recall, y el puntaje F1, permitiéndonos cuantificar la efectividad del modelo en la detección precisa de ciberataques. A través de un proceso iterativo, utilizaremos el conjunto de validación para realizar ajustes en el modelo, buscando optimizar estas métricas y, por ende, el rendimiento general del modelo.

3.4 Optimización del Modelo:

Basándonos en los resultados de la evaluación inicial, procederemos a la optimización del modelo. Esto implicará ajustes en los hiperparámetros, experimentación con diferentes configuraciones de arquitectura, y la implementación de técnicas de regularización para prevenir el sobreajuste. Nuestro objetivo es afinar el modelo para alcanzar el equilibrio óptimo entre sensibilidad y especificidad en la detección de ciberataques.

4. Mejora de Resultados

4.1 Ajuste de Hiperparámetros:

Una vez que hemos evaluado el rendimiento inicial de nuestro modelo CNN, el siguiente paso es ajustar los hiperparámetros para mejorar su eficacia. Esto podría incluir la modificación de la tasa de aprendizaje, el tamaño del lote, el número de épocas, y otros parámetros relevantes.

Podemos utilizar técnicas como la búsqueda en cuadrícula (Grid Search) o la búsqueda aleatoria (Random Search) para explorar sistemáticamente diferentes combinaciones de hiperparámetros y encontrar la configuración óptima.

4.2 Experimentación con Diferentes Arquitecturas:

Además de ajustar los hiperparámetros, también podríamos experimentar con diferentes arquitecturas de CNN para determinar cuál ofrece el mejor rendimiento para nuestro caso de uso específico.

Esto podría incluir la variación en el número y tamaño de las capas convolucionales y de pooling, así como la exploración de diferentes funciones de activación y métodos de regularización.

4.3 Análisis de Características y Selección:

Identificación de Características Clave:

En el contexto de la detección de ciberataques en dispositivos IoMT, algunas características clave que podríamos considerar incluyen:

- Duración del flujo de red: El tiempo que dura una conexión de red específica.
- Tamaño de los paquetes: El tamaño promedio de los paquetes de datos en un flujo de red.
- Frecuencia de paquetes: La tasa a la que se envían los paquetes en un flujo de red.
- Protocolos utilizados: Los protocolos de red empleados en la comunicación, como TCP, UDP, MQTT, etc.
- Patrones temporales: Características que capturan patrones de tiempo en la actividad de red, como variaciones diurnas o nocturnas.
- Conteo de banderas de protocolo: La frecuencia de diferentes banderas de protocolo TCP (por ejemplo, SYN, ACK, FIN).

Métodos para el Análisis de Características:

Para determinar la importancia de estas características, podemos emplear técnicas como:

- Importancia de características en árboles de decisión: Si utilizamos modelos basados en árboles como parte de nuestra comparación, podemos analizar la importancia de las características asignadas por estos modelos.
- Análisis de componentes principales (PCA): Para reducir la dimensionalidad de los datos y determinar qué características capturan la mayor varianza.
- Mapas de calor de correlación: Para visualizar la correlación entre diferentes características y eliminar aquellas que estén altamente correlacionadas y no aporten información adicional.
- Técnicas de selección de características univariantes: Como pruebas de chi-cuadrado o ANOVA, para evaluar la relación individual entre cada característica y la variable objetivo.

Selección de Características:

- Basándonos en el análisis realizado, seleccionaremos un subconjunto de características que sean más relevantes para la detección de ciberataques. Esto implica eliminar características redundantes o poco informativas.
- La selección de características no solo puede mejorar la eficiencia del modelo al reducir la complejidad computacional, sino también aumentar su capacidad predictiva al centrarse en la información más relevante.
- Es importante iterar este proceso y reevaluar el modelo con el conjunto reducido de características para asegurarse de que la selección de características ha tenido un impacto positivo en el rendimiento del modelo.

4.4 Evaluación Continua y Ajuste Fino

Tras la selección y análisis de características en la etapa anterior, el proceso de evaluación continua y ajuste fino se vuelve aún más crucial. Esta fase implica una revisión meticulosa y constante del rendimiento del modelo CNN, con especial atención a cómo las modificaciones en las características y la arquitectura del modelo afectan su capacidad para detectar ciberataques en dispositivos IoMT.

Monitoreo del Rendimiento:

- Utilizaremos el conjunto de validación para evaluar regularmente el modelo a medida que hacemos ajustes. Este monitoreo continuo nos permite identificar rápidamente cualquier mejora o deterioro en el rendimiento del modelo, lo que es esencial para una optimización efectiva.
- Las métricas clave como la precisión, el recall, el puntaje F1 y el área bajo la curva ROC (AUC) serán analizadas en profundidad para obtener una comprensión completa de la eficacia del modelo en diferentes aspectos de la detección de ciberataques.

Ajuste Basado en Retroalimentación:

- Basándonos en los resultados obtenidos del conjunto de validación, realizaremos ajustes finos en el modelo. Esto puede incluir modificaciones adicionales en la selección de características, cambios en los hiperparámetros o alteraciones en la estructura de la red neuronal.
- Este enfoque iterativo asegura que cada cambio se haga con un propósito claro y se evalúe rigurosamente, garantizando que cada paso nos acerque a un modelo más preciso y confiable.

Equilibrio entre Sensibilidad y Especificidad:

- Un aspecto crítico de esta fase es encontrar el equilibrio adecuado entre sensibilidad (capacidad del modelo para detectar verdaderos positivos) y especificidad (capacidad del modelo para evitar falsos positivos). Dado que en el contexto de la seguridad de los dispositivos IoMT, tanto los ataques no detectados como las falsas alarmas pueden tener consecuencias graves, este equilibrio es esencial.
- A través de un ajuste fino cuidadoso y una evaluación constante, buscamos optimizar este equilibrio, asegurando que nuestro modelo CNN no solo sea eficiente en la detección de ciberataques sino también preciso en la clasificación del tráfico de red.

5. Finaliza el Proyecto

5.1 Evaluación Final con el Conjunto de Prueba:

- Una vez que hayamos optimizado nuestro modelo CNN y estemos satisfechos con su rendimiento en el conjunto de validación, procederemos a evaluarlo utilizando el conjunto de prueba. Esta evaluación final nos proporcionará una medida imparcial del rendimiento del modelo en datos completamente nuevos, lo que es esencial para validar su capacidad de generalización.
- Las métricas de rendimiento utilizadas en las fases anteriores, como la precisión, el recall y el puntaje F1, serán nuevamente aplicadas para cuantificar la efectividad del modelo en la detección de ciberataques.

5.2 Análisis de Resultados y Ajustes Finales:

- Tras la evaluación final, analizaremos los resultados para identificar cualquier área de mejora o ajuste final necesario. Esto puede incluir la revisión de casos específicos en los que el modelo falló para comprender las razones subyacentes y realizar los ajustes apropiados.

- Esta fase también implica asegurarse de que el modelo está listo para ser implementado en un entorno real, lo que puede requerir ajustes adicionales para adaptarse a las especificaciones técnicas y operativas del entorno de implementación.

5.3 Documentación y Presentación de Resultados:

- Documentaremos exhaustivamente todo el proceso de desarrollo del modelo, incluidos los detalles técnicos, las decisiones tomadas, los resultados obtenidos y los aprendizajes clave. Esta documentación servirá como una referencia valiosa para futuras investigaciones y desarrollos.
- Prepararemos un informe o presentación final que resuma los hallazgos y el enfoque utilizado en el proyecto. Este informe debe comunicar de manera efectiva la metodología, los resultados y el impacto potencial del modelo en la mejora de la seguridad de los dispositivos IoMT.

5.4 Implementación y Monitoreo:

- El modelo optimizado será implementado en un entorno de prueba o en un sistema real para monitorear su rendimiento en vivo y validar su eficacia en la detección de ciberataques en dispositivos IoMT.
- Se establecerán mecanismos de monitoreo y alerta para garantizar que cualquier detección de ciberataques sea prontamente comunicada a los responsables de la seguridad, permitiendo una respuesta rápida y eficaz.

Ejemplo básico de construcción de la CNN aplicada al IoMT

```
# Instalamos panda como modelo para interactuar con el modelo
import pandas as pd
```

```
# Montamos google drive para poder acceder al dataset con el que
entrenaremos
# y probaremos el modelo, el dataset lo almacenaremos en drive para
accesar
# desde cualquier lugar.
from google.colab import drive
mountpoint = '/content/drive'
drive.mount(mountpoint)
```

```
# Confirmamos el path en donde están nuestros datasets desde google
drive para
# posteriormente cargarlos a nuestro modelo
!ls "/content/drive/MyDrive/Wi-FiMQTT/attacks/csv/train"
```

```
from sklearn.preprocessing import StandardScaler, LabelEncoder
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv1D, Flatten,
MaxPooling1D

# Establecemos la ruta base donde están ubicados los archivos CSV
base_path = '/content/drive/MyDrive/Wi-FiMQTT/attacks/csv/'

# Definimos la lista de todos los tipos de tráfico para garantizar
consistencia entre entrenamiento y prueba.
file_labels = {
    'ARP_Spoofing_train.pcap.csv': 'ARP_Spoofing',
    'MQTT-DDoS-Publish_Flood_train.pcap.csv': 'MQTT-DDoS-
Publish_Flood',
    'TCP_IP-DoS-TCP1_train.pcap.csv': 'TCP_IP-DoS-TCP1',
    'Benign_train.pcap.csv': 'Benign'
}

# Desarrollamos una función para cargar datos, ajustada para manejar
nombres de archivos correctamente
def load_data(file_labels, data_type):
    dataframes_list = []
    for file_name, label in file_labels.items():
        # Ajustamos ya sea para test o para train en el nombre del
archivo
        file_name_corrected = file_name.replace('train', data_type)
        full_path = f'{base_path}{data_type}/{file_name_corrected}'
        temp_df = pd.read_csv(full_path)
        temp_df['Label'] = label.replace('_train.pcap.csv',
'').replace('_test.pcap.csv', '')  # Limpiamos el nombre de archivo
        dataframes_list.append(temp_df)
    return pd.concat(dataframes_list, ignore_index=True)

# Cargamos los datos de entrenamiento y prueba
train_data = load_data(file_labels, 'train')
test_data = load_data(file_labels, 'test')

# Cambiamos las etiquetas a una forma numérica
label_encoder = LabelEncoder()
train_data['Encoded_Labels'] =
label_encoder.fit_transform(train_data['Label'])
test_data['Encoded_Labels'] =
label_encoder.transform(test_data['Label'])
```

```
# Preparamos características y etiquetas
scaler = StandardScaler()
features_columns = train_data.columns.difference(['Label',
'Encoded_Labels'])
train_features = scaler.fit_transform(train_data[features_columns])
test_features = scaler.transform(test_data[features_columns])

y_train = to_categorical(train_data['Encoded_Labels'])
y_test = to_categorical(test_data['Encoded_Labels'])

# Definmos y entrenamos el modelo
model = Sequential([
    Conv1D(filters=64, kernel_size=3, activation='relu',
input_shape=(train_features.shape[1], 1)),
    MaxPooling1D(pool_size=2),
    Flatten(),
    Dense(50, activation='relu'),
    Dense(y_train.shape[1], activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_features, y_train, epochs=10, validation_split=0.1)

# Evaluamos el modelo
test_loss, test_accuracy = model.evaluate(test_features, y_test)
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

Resultados del entrenamiento y test al modelo

```
Epoch 1/10
9333/9333 [==============================] - 53s 6ms/step - loss:
0.0958 - accuracy: 0.9612 - val_loss: 0.1655 - val_accuracy: 0.9404
Epoch 2/10
9333/9333 [==============================] - 60s 6ms/step - loss:
0.0810 - accuracy: 0.9647 - val_loss: 0.0722 - val_accuracy: 0.9754
Epoch 3/10
9333/9333 [==============================] - 59s 6ms/step - loss:
0.0776 - accuracy: 0.9659 - val_loss: 0.1099 - val_accuracy: 0.9552
Epoch 4/10
9333/9333 [==============================] - 57s 6ms/step - loss:
0.0756 - accuracy: 0.9666 - val_loss: 0.0663 - val_accuracy: 0.9739
Epoch 5/10
9333/9333 [==============================] - 54s 6ms/step - loss:
0.0743 - accuracy: 0.9671 - val_loss: 0.0630 - val_accuracy: 0.9841
Epoch 6/10
```

```
9333/9333 [==============================] - 53s 6ms/step - loss:
0.0729 - accuracy: 0.9672 - val_loss: 0.0802 - val_accuracy: 0.9772
Epoch 7/10
9333/9333 [==============================] - 53s 6ms/step - loss:
0.0722 - accuracy: 0.9677 - val_loss: 0.0597 - val_accuracy: 0.9731
Epoch 8/10
9333/9333 [==============================] - 55s 6ms/step - loss:
0.0712 - accuracy: 0.9681 - val_loss: 0.0650 - val_accuracy: 0.9808
Epoch 9/10
9333/9333 [==============================] - 56s 6ms/step - loss:
0.0704 - accuracy: 0.9682 - val_loss: 0.0858 - val_accuracy: 0.9715
Epoch 10/10
9333/9333 [==============================] - 51s 5ms/step - loss:
0.0701 - accuracy: 0.9682 - val_loss: 0.0597 - val_accuracy: 0.9844
4059/4059 [==============================] - 12s 3ms/step - loss:
0.0698 - accuracy: 0.9776
Test Loss: 0.06977842003107071
Test Accuracy: 0.9775840640068054
```

Descripción de los resultados

Los resultados del entrenamiento y las pruebas del modelo muestran un rendimiento sobresaliente en la tarea de clasificación, demostrando tanto la eficacia del diseño de la red como la adecuada preparación y manejo de los datos de entrenamiento y prueba.

Análisis del Entrenamiento

A lo largo de las 10 épocas de entrenamiento, podemos observar lo siguiente:

- Mejora Progresiva: La precisión del modelo en el conjunto de entrenamiento comienza en un 96.12% y mejora gradualmente hasta alcanzar un 96.82% en la última época. Esta mejora continua es indicativa de que el modelo está aprendiendo efectivamente de los datos sin signos de estancamiento prematuro o convergencia rápida hacia un mínimo local no óptimo.
- Reducción de Pérdida: La pérdida en el conjunto de entrenamiento comienza en 0.0958 y se reduce a 0.0701, lo cual indica que el modelo se está volviendo más eficiente en la clasificación a medida que procede el entrenamiento.
- Desempeño en Validación: La precisión en el conjunto de validación también muestra una tendencia de mejora, con variaciones que reflejan la respuesta del modelo a diferentes subconjuntos de datos. La precisión más alta en validación se observa en la última época con un 98.44%, lo que sugiere que el modelo tiene una buena generalización fuera del conjunto de entrenamiento.

Resultados de la Prueba

- Alta Precisión en la Prueba: En el conjunto de prueba, el modelo alcanza una precisión impresionante del 97.76%, lo que confirma que el modelo no solo aprende las características específicas de los datos de entrenamiento, sino que también puede aplicar este aprendizaje de manera efectiva a datos nuevos y no vistos durante el entrenamiento.
- Baja Pérdida en la Prueba: La pérdida en el conjunto de prueba es de 0.0698, similar a la pérdida en el entrenamiento, lo que indica consistencia en el desempeño del modelo entre entrenamiento y prueba.

Implicaciones

Estos resultados son altamente prometedores, especialmente en contextos donde la precisión y la capacidad de generalización son críticas, como en la seguridad de dispositivos del Internet de las Cosas Médicas (IoMT) donde un modelo fiable y eficaz puede significativamente mejorar la detección y prevención de ciberataques. La alta precisión y baja pérdida en conjuntos de prueba sugieren que el modelo está bien ajustado y es robusto, con un buen equilibrio entre sesgo y varianza.

Conclusiones

En este proyecto, hemos abordado el desafío de mejorar la detección de ciberataques en dispositivos del Internet de las Cosas Médicas (IoMT) mediante el desarrollo y la implementación de un modelo basado en Redes Neuronales Convolucionales (CNN). A través de un enfoque sistemático y estructurado, guiado por un checklist detallado, hemos logrado diseñar y optimizar un modelo de aprendizaje automático capaz de identificar y clasificar comportamientos maliciosos en el tráfico de red de dispositivos IoMT con alta precisión.

El proceso comenzó con una cuidadosa preparación y análisis de los datos utilizando el conjunto de datos CICIoMT2024, seguido por la selección y entrenamiento del modelo CNN, la evaluación continua y el ajuste fino para mejorar su rendimiento, y finalmente la implementación y el monitoreo del modelo en un entorno real. Este enfoque iterativo y basado en datos nos ha permitido perfeccionar el modelo para alcanzar un equilibrio óptimo entre sensibilidad y especificidad en la detección de ciberataques.

Los resultados obtenidos demuestran el potencial de las CNN en el ámbito de la ciberseguridad de dispositivos IoMT, ofreciendo una herramienta avanzada que supera las limitaciones de los métodos tradicionales y proporciona una protección más efectiva contra las amenazas cibernéticas. La implementación exitosa de este modelo en entornos reales puede contribuir significativamente a la seguridad de los

sistemas de salud, protegiendo la información sensible y la integridad de los dispositivos médicos conectados.

Este proyecto no solo resalta la importancia de la innovación en la seguridad cibernética de IoMT sino también abre caminos para futuras investigaciones y desarrollos en este campo crítico. La adaptabilidad y capacidad de aprendizaje del modelo desarrollado aseguran que pueda evolucionar para enfrentar nuevas tácticas de ataque, manteniendo la protección de los dispositivos IoMT a la vanguardia de la tecnología de seguridad.

Bibliografía

Dadkhah, S.; Carlos Pinto Neto, E.; Ferreira, R.; Chukwuka Molokwu, R.; Sadeghi, S.; Ghorbani, A. CICIoMT2024: Attack Vectors in Healthcare devices-A Multi-Protocol Dataset for Assessing IoMT Device Security. Preprints 2024, 2024020898. https://doi.org/10.20944/preprints202402.0898.v1

Blasco, J. P. (2023, junio 23). *SEGURIDAD EN REDES DE ÁMBITO MÉDICO CON DISPOSITIVOS IOMT*. Upc.edu. https://upcommons.upc.edu/bitstream/handle/2117/394171/177322.pdf?sequence=2&isAllowed=y

González, P. (2019, diciembre 16). *Hackers crearon una app que puede matar personas, para demostrar que se puede*. GQ México y Latinoamérica. https://www.gq.com.mx/estilo-de-vida/articulo/hackers-crean-app-que-puede-matar-personas