Predicción de Infección por Malware
Universidad Del Valle de Guatemala
CC3094 Security Data Science

Gerardo Méndez[1], María José Castro[2]
[1]18239, [2]181202

Abstract

Telemetry is a service implemented commonly in computers, cars and medical equipment to provide remote and periodical data regarding the system status and performance. This allows providers to offer a better product, whilst bettering the user experience based on the information they anonymously receive. On the other hand, malware is on an all time high, and with the attacks becoming more and more diverse, security analysts need to develop new ways to combat the increasing danger of malicious software. The Microsoft Malware Prediction pretends to collect, via the Windows Telemetry Service, samples of Windows OS computers' insights, building a dataset that serves as a source of information for the community, in an effort to detect existing and predict potential infected Windows systems. In this paper, we utilize said dataset together with supervised machine learning algorithms to develop, test and evaluate models capable of accurately and consistently classify Windows systems as infected or not infected, given a piece of malicious software has been identified within the system or not. Two models were built and evaluated, implementing the algorithms for: XGBoost and Random Forest. The evaluation process was done obtaining some of the common classification metrics, such as accuracy, recall, precision and the AUC-ROC curves.

Keywords: security, data science, telemetry, windows, intrusion, classification, boosting, machine learning, ensemble, random forest.

1 Introducción

Este proyecto estudia y analiza un set de datos que contiene datos de telemetría de Windows, que describen las propiedades de una computadora, sus características y el estado en el que se encuentra. Se combinan métricas de Windows Defender con reportes de heartbeat del equipo, para generar entradas en el set de datos. El dataset provee una gran cantidad de registros: el nombre y version del equipo, sistema y aplicación, así como información de hardware tal como la capacidad de memoria principal y secundaria, el número de cores y el fabricante. Además, se contó con información de localización como el país y la ciudad. La presente entrega comprende la fase de preparación y elección de modelos y algoritmos para lograr el objetivo principal: proponer un modelo eficiente para el reconocimiento y prevención de intrusiones maliciosas en equipos de cómputo. Para ello, fueron seleccionados los siguientes algoritmos para poder realizar una comparación entre los mismos: XGBoost y Random Forest.

2 Marco Teórico

Los servicios telemetría se utilizan para recolectar, de manera remota y periódica, información relevante de un sistema, con el objetivo de ofrecer un mejor servicio. Esto se logra identificando potenciales fallos o malfuncionamientos del equipo, reconciendo sus patrones, y ofreciendo una trazabilidad que permite determinar su causa y su más viable solución. Es utilizado comunmente en computadores, servidores, equipo médico y automóviles.

El enfoque principal de estas herramientas se encuentra en enviar transmisiones de datos desde un gran número de dispositivos hacia un punto central, el cual tiene control de la información y, a cierto modo, del dispositivo. La información recolectada puede ser utilizada luego para fines de análisis, diagnóstico y predicción, por la entidad proveedora del equipo. [4]

2.1 Windows Compatibility Telemetry Service

El servicio de telemetría de Windows fue introducido por primera vez en la versión de Windows 10. Los reportes que genera contienen métricas y datos técnicos acerca de cómo el dispositivo y su software relacionado están funcionando. El servicio envía datos a Microsoft periódicamente para la mejora continua de sus sistemas y la experiencia de sus usuarios. Se puede dividir la información recolectada en 5 categorías que engloban los servicios, componentes, características y funcionalidades del equipo:

1. **Información de Hardware**
   Esta información incluye la cantidad de memoria disponible, las especificaciones del procesador, el estado y capacidad de la batería, y datos de la memoria secundaria, entre otros.

2. **Información de red**
   Estos datos incluyen la velocidad del adaptador, información acerca de la tarjeta de red y el número IMEI o MAC del dispositivo.

3. **Detalles de cambios de estado**
   Métricas relaciondas a el tiempo, memoria y recursos utilizados en la ejecución de procesos y aplicaciones, así como errores o crashes de programas.

4. **Accesorios**
   Información acerca de los dispositivos que se conectan al equipo (impresora, teclado, cámara web, etc.).

5. **Logs de Microsoft Store**
   Resumen de transacciones realizadas con aplicaciones de la tienda de Microsoft (descargas, instalaciones, actualizaciones).

El análisis forense posterior a la recolección de los datos es de suma importancia, y una de las razones por la que se recogen estos datos. La información de Windows Telemetry permite, a través de una investigación forense digital, construir una línea de tiempo con los eventos recolectados. Dado que la información recolectada engloba a un sistema completo, los resultados son más confiables que si solamente analizaramos cada categoría por separado. De igual forma, provee datos que no pueden obtenerse de forma convencional. Los identificadores de piezas de hardware como la memoria secundaria o la tarjeta RAM, solamente pueden ser extraídas por el WCT. [3]

3 Antecedentes

El aumento en cantidad y complejidad de los nuevos ataques de malware presenta un desafío para la seguridad de los sistemas de cómputo. Siguiendo las tasas actuales, es normal ver cientos de miles de potenciales archivos maliciosos en la red. Analizar este volumen de archivos sería costoso en tiempo y recursos. Es por ello que surgen nuevas soluciones, con el objetivo de proveer una herramienta de análisis e identificacioón de malware confiable. Una de ellas es Valkyrie, un sistema de detección de malware que utiliza datos de eventos recolectados por telemetría de distintos dispositivos y enviados a la nube. La información recolectada permite la construcción de un perfil para archivos PEM (Windows Portable Executable), que consecuentemente pueden ser clasificados utilizando técnicas de machine learning. [5]

La información recolectada se almacena en la nube, y cada entrada nueva se identifica con un PID único, independiente de la máquina de la que proviene. Los datos utilizados para Valkyrie consisten principalmente en:

1. **Datos de procesos**
   El evento iniciado, nivel de privilegio otorgado, y ubicación del PID.

2. **Datos de red**
   Direcciones IP de origen y destino, puertos y métricas de protocolos.

3. **Datos de DNS**
   Registros consultados y/o enviados.

4. **Archivos modificados**
   Nombre y tipo de los archivos.

Valkyrie es capaz de detectar actividad maliciosa sin tener acceso a los archivos binarios de malware. Esto permite la utilización información de telemetría recolectada a bajo costo.

4 Análisis exploratorio

Para la exploración de datos nos apoyamos en Pandas y en otras librerías de visualización y optimización, como Seaborn y Dask. La intención de esta exploración fue realizar la limpieza de datos inicial, y conocer acerca de la forma en la que la información se presenta, para lograr encontrar insights que sirvieran de respaldo para la selección de features.

Durante la selección de características, se removieron del set de datos aquellas columnas que tuvieran un mismo valor con una frecuencia mayor al 97%. También se eliminaron aquellas columnas con altas cantidades de valores nulos que no pudieran ser reemplazados con técnicas de manejo de datos. Por último, se eliminaron las columnas con una cardinalidad alta, que no tuviesen relevancia con relación al resto de datos.

5 Selección de modelos

Para la seleccion de modelos se tomó en cuenta la naturaleza del problema, y se determinaron necesarios algoritmos de clasificación que fuesen útiles en la identificación de si una computadora haya sufrido o no alguna intrusión de malware.

5.1 XGBoost

El nombre de este modelo se deriva de eXtreme Gradient Boosting. El algoritmo de XGBoost ha cobrado relevancia recientemente en la comunidad de Data Science. El XGBoost es simplemente una implementación de árboles de decisión de gradient boosting, y busca principalmente rapidez y un buen performance. [2]

Este algoritmo pertenece a la familia algoritmos, que utilizan la técnica de Boosting, donde nuevos modelos son utilizados para corregir errores de modelos anteriores. Así, se utilizan muchos 'weak learners' para crear un solo 'strong learner', dejándonos un modelo robusto y preciso. Sin embargo, aunque este algoritmo pueda ser clasificado dentro de la familia de las boosting machines (GBM, SVM, etc.), una característica que lo diferencia es la velocidad a la que opera, ya que es mucho más veloz que cualquiera de los algoritmos que implementan el gradient boosting.

5.2 Random Forest

Este modelo es de tipo supervisado, y se utiliza en su mayoría para problemas de clasificación y regresión. El algoritmo utiliza árboles de decisión, de los cuales toma muestras y lleva a cabo un voto por mayoría, el cual sirve para la clasificación y, en el caso de la regresión, la media. Una de sus características más importantes es que puede manejar features continuas y categóricas en caso la regresión o clasificación, respectivamente, lo necesite. Generalmente, tiene un mejor desempeño en problemas de clasificación.

6 Metodología

Se tomó una muestra significativa de los datos, del 25% equivalente a $\doteq$ 2,000,000 de registros, la cual fue utilizada para las fases de entrenamiento y evaluación de los modelos. Se realizó una división de 70% y 30% respectivamente en cada uno de los nuevos sets de datos. Luego de esto, para los modelos evaluados, se decidió hacer un entrenamiento inicial con los parámetros default, a modo de obtener un caso base con el cual comparar el desempeño de los modelos con distintas combinaciones de hiperparámetros. Luego de esto, se realizaron otras iteraciones con distintos parámetros, para validar cuál de todas las combinaciones era la que mejor resultados generaba en el entrenamiento. Finalmente, se generaron las métricas de evaluación para cada modelo, con lo cual se realizó la comparación final, en la que se determinó el rendimiento general de cada modelo.

7 Métricas de evaluación

Para la evaluación de los modelos, se decidió utilizar las métricas importantes para algoritmos de clasificación: accuracy, recall, precision y las curvas AUC-ROC. Estas métricas son las que determinaron cuál de los modelos mostró un mejor desempeño en comparación al resto. La gráfica AUC-ROC agrega valor a la comparación entre modelos, ya que nos permite visualizar el desempeño de nuestro modelo de una forma gráfica, lo que deriva en un mejor entendimiento.

8 Resultados

8.1 Implementación de XGBoost

Accuracy	Precision	Recall	F1-Score
0.61	0.63	0.64	0.63

Tabla 1: Métricas para XGBoost

Durante la implementación y validación de este modelo, se pudo observar un rendimiento ligeramente sobre el promedio por parte del algoritmo. Los valores de precision y recall, indican que el modelo clasifica más de la mitad de los registros correctamente, y que cuando el modelo clasifica un registro, la clase real es la clase clasificada en 6 de cada 10 iteraciones. Finalmente, el accuracy del modelo también se aproxima al 0.6, lo que termina de confirmar el comportamiento mediano de la construcción. Estos resultados se acercan a ser buenos, aunque no satisfacen el rendimiento esperado por un modelo de clasificación confiable y eficaz, lo cual se buscaba en esta implementación. El cálculo de los valores AUC y ROC se puede visualizar en el gráfico debajo 1, y se muestra, como se menciona, un rendimiento regular por parte del algoritmo y su implementación.

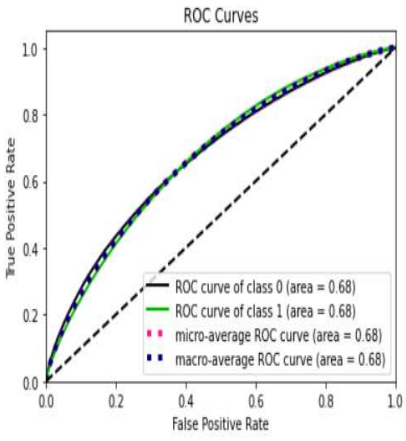

Figura 1: Curva ROC - XGBoost

8.2 Implementación de Random Forest

Accuracy	Precision	Recall	F1-Score
0.54	0.55	0.54	0.52

Tabla 2: Métricas para Random Forest

El modelo de Random Forest, al igual que el XGBoost, presenta métricas y un rendimiento regular en su implementación. Las métricas son ligeramente más bajas, acercándose más a los valores de 0.55, lo que indica un rendimiento aún menor al discutido para el modelo de Boosting. Los valores obtenidos para la precision y el recall indican una clasificación correcta de los valores en únicamente la mitad de las iteraciones. Esto demuestra un pobre rendimiento, quedándose corto para ser un modelo con capacidad de clasificación alta.

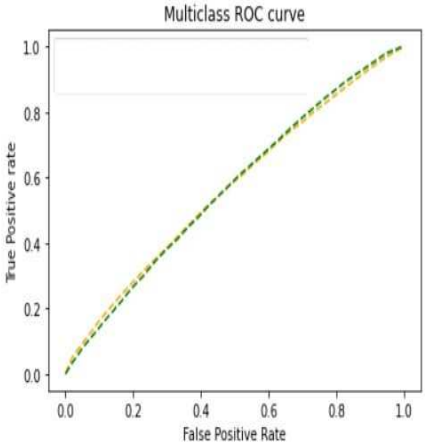

Figura 2: Curva ROC - Random Forest

Podemos observar que, las curvas AUC-ROC dibujan prácticamente una recta con pendiente 1, demostrando una vez más la eficiencia y precisión en únicamente la mitad de sus predicciones.

9 Discusión de resultados

Luego de analizar los resultados, se obtuvo que el rendimiento de los modelos fue (de mejor a peor):

1. XGBoost
2. Random Forest

Se logró identificar la eficiencia de los distintos modelos por medio de las métricas de evaluación definidas. El algoritmo de XGBoost demostró tener una mejor eficiencia en cuestion de todas las métricas evauluadas (precision, accuracy y recall), sin embargo en cuestion de su tiempo de ejecución fue significativamente mucho más complejo y largo para la ejecución y predicción.

Al observar la Figura 1 podemos ver que el modelo inicia con la curva esperada, puesto que vemos que la tendencia inicia hacia la esquina superior izquierda. Esto nos da la premisa de que es posible que el modelo pueda comportarse mejor o presentar métricas más adecuadas si el dataset se manipulara de manera diferente, o si se refinaran más los hiperparámetros del modelo.

El modelo XGBoost, en este caso, no ha presentado un desempeño que justifique el tiempo que ha requerido en su entrenamiento y manejo de parámetros. Sin embargo, fue el modelo con mejor desempeño en la clasificación entre los dos que fueron evaluados.

Por otro lado el Random Forest presentó una menor precisión, y requirió de la eliminación de varias columnas categóricas que no permitían la ejecución del modelo. Aún así la diferencia entre este y el XGBoost es de $\tilde{0}.05$ en la mayoría de las métricas de ambos modelos. Esta diferencia no es lo suficientemente significativa para determinar que el algoritmo sí se vio afectado en cuestión de desempeño al elimiar las columnas categóricas.

Se puede determinar que los modelos tuvieron un desempeño promedio, pero el verdadero desafío se presentó en la lectura del set de datos, puesto que la manipulacion de la informacion es clave en problemas con un alto volumen de datos y una alta dimensionalidad en el set.

10 Trabajo futuro

En futuras implementaciones se recomienda una implementación con mayores recursos de cómputo. Aún y cuando son considerados de alta gama, dispositivos que cuentan con multi-core y cerca de 16 GB de memoria principal, sufren contratiempos con los recursos disponibles, lo cual limita y ralentiza el análisis y manipulación de datos, así como la implementación (especialmente en la fase de ajuste de hiperparámetros), fases de vital importancia en el desarrollo y construcción de modelos de aprendizaje .

11 Conclusiones

- Se determinó que el algoritmo con el mejor modelo fue XGBoots, ya que presentó la mejor accuracy con 0.61, además de ser superior en las demás métricas.
- Se determinó que la diferencia entre las métricas evaluadas entre ambos modelos no fue lo suficientemente significativa como para

descartar a la implementación de Random Forest como un modelo efectivo para el problema evaluado en el artículo.

- Se determinó que la correcta distribución, lectura, y procesamiento de la información cruda, es clave para permitir a los modelos alcanzar un mejor desempeño.
- Se concluyó que los modelos que utilizan técnicas de *Boosting* producen un mejor resultado que los algoritmos que utilizan técnicas de *Ensemble*, al construir modelos que utilicen en el set de datos del proyecto.

Referencias

[1] Altman, Naomi S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression" (PDF). The American Statistician. 46 (3): 175–185.

[2] Brownlee, J. (2021, 16 febrero). A Gentle Introduction to XGBoost for Applied Machine Learning. Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[3] Han, J., Park, J., Chung, H. & Lee, S. (2020). "Forensic analysis of the Windows telemetry for diagnostics" (PDF). arXiv, 2020, doi: 10.48550/ARXIV.2002.12506

[4] IBM. (30 de septiembre de 2019). "Telemetry concepts and scenarios for monitoring and control", URL https://www.ibm.com/support/knowledge center/en/SSFKSJ_7.5.0/com.ibm.mq.pro. doc/q002770_.htm

[5] S. Krasser, B. Meyer and P. Crenshaw, "Valkyrie: Behavioral malware detection using global kernel-level telemetry data," 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP), 2015, pp. 1-6, doi: 10.1109/MLSP.2015.7324334.

[6] R, S. E. (2021, 24 junio). Random Forest — Introduction to Random Forest Algorithm. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

[7] What is Malware? (2022, 24 febrero). Cisco. https://www.cisco.com/c/en/us/products/security/advanced-malware-protection/what-is-malware.html