Detección de Malware
Universidad Del Valle de Guatemala
CC3094 Security Data Science

Gerardo Méndez[1], Luis Pedro Cuéllar[2], Christopher Barrios[3]
[1]18239, [2]18220, [3]18207

Abstract

Malware is a commonly utilized technique for stealing, damaging and/or destroying private data on a computer system. Unauthorized access provides the attacker with the means to damage a system, its information and the service it provides. Many countermeasures can be implemented, and detection and prevention continue to be important yet sometimes quite difficult to achieve. The Secure Intelligent Methods for Advanced Recognition of malware and stegomalware project gives access to useful metrics that can provide certain insights into understanding and consequently identifying these attacks better. In this paper, we utilize supervised machine learning algorithms and the SIMARGL dataset to build, implement and evaluate models capable of accurately and consistently classifying network and communication flows as belonging to one of SYN Scan, DDoS R-U-Dead-Yet or DDos Slowloris attacks. Three models were built and evaluated, implementing the algorithms for: Random Forest, K-Nearest Neighbors (KNN), and Gradient Boosting Machine (GBM). The evaluation process was done obtaining some of the common classification metrics as accuracy, recall, precision and AUC-ROC.

Keywords: security, data science, cyberattacks, denial-of-service, classification, random forest, knn, gradient boosting machine, malware, machine learning, cybercrime

1 Introducción

Este proyecto estudia y analiza un set de datos que contiene información recolectada en la implementación del proyecto Secure Intelligent Methods for Advanced RecoGnition of malware and stegomalware (SIMARGL), en el cual, por medio de una red diseñada para capturar ataques maliciosos, se pretende proveer métodos efectivos que sirvan para contrarrestar actos criminales de tipo cibernético, y proponer e implementar métodos y modelos de Machine Learning para la detección de malware. La iniciativa surge debido a que anteriores datasets utilizados para la detección de intrusos registran atributos que ya no se usan en la actualidad, o tienen poca calidad en los datos haciendo que su uso sea inefectivo para la implementación de modelos de predicción y prevención. El dataset provee una gran cantidad de registros flujos normales y ataques, capturando varias métricas, especialmente de red, como las direcciones IP, la dirección del flujo de datos, la cantidad de datos y paquetes intercambiados y el tiempo que toma realizar este intercambio. La presente entrega comprende la fase de elección de modelos y algoritmos para poder lograr nuestro objetivo principal: proponer un modelo eficiente para el reconocimiento y prevención de ataques cibernéticos maliciosos. Para ello, fueron delimitados los siguientes algoritmos iniciales para poder realizar una comparación entre los mismos: Random Forest, Gradient Boosting Machine y K-Nearest Neighbors.

2 Marco Teórico

El malware es, en inglés, el término corto para referirse al 'malicious software', y se refiere a cualquier software intrusivo, diseñado para robar, dañar o destruir información privada dentro de equipos o sistemas de computación. [2] Dentro de la categoría de malware, existen varias técnicas y tipos comúnes como lo son los viruses, el spyware, troyanos, ransomware, etc. Los tipos de malware que se recolectaron dentro del dataset son: SYN Scan Attacks, DDoS (R-U-Dead-Yet) y DDoS (Slowloris).

2.1 SYN Scan

El ataque de tipo SYN Scan es una técnica maliciosa utilizada por atacantes, que les permite determinar el estado de un puerto de comunicación de un sistema, sin realmente establecer una conexión total y autenticada con él. [3] Este tipo de ataques puede servir como preámbulo o preparación para un ataque de mayor escala como los ataques Denial of Service (DDoS). Esta técnica es conocida también como un 'half-open scanning'.

2.1.1 Funcionamiento

Para llevar a cabo este ataque, el ente malicioso intenta establecer una conexión de tipo TCP (Transmission Control Protocol) en cada puerto del servidor. Esto se hace enviando un paquete de sincronización (SYN packet), dando la apariencia de un intento de conexión al servidor. Si el servidor responde con un paquete de tipo ACK (acknowledge), significa que el puerto del que se recibió la respuesta se encuentra abierto al público. Además de obtener información acerca del diseño de red del servidor, un flujo grande de paquetes SYN entrantes también consume una cantidad alta de recursos, y puede provocar un mal funcionamiento.

2.2 DDoS (R-U-Dead-Yet)

El ataque 'R U Dead Yet?' es un ataque de tipo denial-of-service, que busca mantener a un servidor ocupado, limitando y en algun punto interrumpiendo por completo el tráfico entrante de usuarios legítimos del sistema. Este ataque (conocido como RUDY) es un ataque de la capa de aplicación de DDoS y, a diferencia de la aplicación tradicional que trata de sobrecargar el servidor con una cantidad alta de conexiones en un período de tiempo corto, tiene una tasa de trabajo más lenta. Esta velocidad de ataque minimizada, se traduce en un ataque prolongado. El objetivo principal del ataque es abrir una baja cantidad de conexiones con el servidor en un período de tiempo determinado, pero mantenerlas abiertas el mayor tiempo posible para suspender los recursos del sistema, y que las operaciones funcionales de la aplicación no puedan ser completadas. [1] Eventualmente, las sesiones agotan los recursos, y el servicio es 'denegado' para los usuarios legítimos del sistema. El tráfico relativamente bajo y lento de este ataque, hace que a muchas de las herramientas tradicionales para detección y prevención de ataques DDoS se les dificulte detectar estos ataques correctamente.

2.2.1 Funcionamiento

1. El ataque RUDY examina la aplicación para encontrar una plantilla con la cual hacer requests.

2. El atacante envía un request de tipo HTTP POST, imitando una entrada legítima. El POST contiene un encabezado que alerta al sistema de que un bloque de información grande será enviado.

3. El proceso de enviar la información toma la forma de: enviar paquetes de 1 byte cada uno, en intervalos de tiempo aleatorios de unos 10 segundos cada uno.

4. Se continua enviando información de manera indefinida. El servidor por otra parte mantendrá la conección abierta aceptando los paquetes, mientras que su capacidad de manejo de paquetes disminuye.

2.3 DDoS (Slowloris)

El ataque slowloris es un ataque denial-of-service que intenta sobrecargar un servidor determinado manteniendo conexiones simultáneas entre el atacante y el objetivo. [4] Esta técnica consiste en enviar solicitud HTTP parcial, es decir una solicitud que no termina. Al igual que el ataque RUDY, el objetivo principal es mantener las conexiones abiertas el mayor tiempo posible.
Este tipo de ataque consume una baja cantidad de ancho de banda, y busca utilizar por completo los recursos del servidor al que se realiza el ataque. [4] Esto se logra con requests que imitan tráfico de red normal, que únicamente parecen más lento de lo normal, pero podría ser atribuido a una conexión de internet pobre. El servidor tiene una cantidad límite de hilos para manejar la concuerrencia, y cada una de las conexiones intentará mantenerse abierta para lograr completar la solicitud HTTP, la cual nunca se completará. Esta técnica cae dentro de la categoría de los ataques "low and slow".

3 Análisis exploratorio

Para la exploración de datos nos apoyamos en Pandas y en otras librerías de visualización, como Seaborn y Pandas Profiling. 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.

4 Selección de modelos

Para la seleccion de modelos se tomó en cuenta la naturaleza del problema, y lo que se necesitan son algoritmos de clasificación que puedan identificar a un flujo de comunicación como malicioso o no malicioso, y de ser malicioso a cuál de las tres categorías pertenece (SYN, RUDY, Slow).

4.1 Decision Tree

Este modelo de árboles de decisión toma distintos caminos en sus nodos, y establece cúal de ellos será mejor para la predicción. Se seleccionó este algoritmo porque es uno de los modelos más comunes para la clasificación, y su utilización e implementación es simple y eficiente, sin necesidad de manejar una cantidad muy alta de hiper-

parámetros. La característica principal es su simplicidad en la implementación.

4.2 KNN

El algoritmo de K-Nearest Neighbors es un algoritmo de aprendizaje supervisado, utilizado tanto en problemas de regresión como en problemas de clasificación. En pocas palabras, el algoritmo clasifica cada punto del dataset, y lo agrupa con sus K-vecinos más cercanos, formando así las categorías en las que se dividen los datos. La salida del modelo es precisamente eso: un valor de pertenencia a una de las clases, con las cuales cada punto de información será clasificado. Se seleccionó este modelo debido a su muy buen rendimiento en problemas de clasificación, y la cantidad baja de hiperparámetros que necesita su implementación.

4.3 Gradient Boosting Machine

Este modelo pertenece a la familia de 'boosting algorithms', su enfoque es entrenar y combinar varios weak learners para condensarlos y formar un solo stron learner. En su forma más simple, las GBMs son algoritmos de clasificación binarios, en donde el resultado de la comparación de nueva información con lo aprendido es una función positiva o negativa. La clasificación multiclase también puede ser implementada y es la que se utilizó en esta investigación. Se seleccionó este modelo debido a su naturaleza de algoritmo de clasificación, su alta utilización en problemas en distinas áreas de la ciencia de datos y su robustez, la cual provee una característica distinta a los otros dos modelos que se utilizaron.

5 Metodología

Se tomó una muestra significativa de los datos, del 25% equivalente a $\doteq$ 2,500,000 de registros, la cual fue utilizada para las fases de entrenamiento, validación y evaluación de los modelos. Se realizó una división de 55%, 15% y 30% respectivamente en cada uno de los nuevos sets de datos. Luego de esto, para los tres modelos, 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 inicializaciones 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, y con esto se realizó la comparación final en la que se determinó cuál de los tres tuvo el mejor rendimiento.

6 Métricas de evaluación

Para las métricas de evaluación, se decidió utilizar métricas importantes y comunes de modelos de clasificación: accuracy, recall, precision y la curva AUC-ROC. Estas métricas son las que determinaron cuál de los modelos fue el que mostró un mejor desempeño en comparación a sus rivales. La gráfica de AUC-ROC agrega valor al análisis y comparación entre modelos, ya que nos permite ver y mostrar el desempeño de nuestro modelo de una forma gráfica, lo cual deriva en un mejor entendimiento.

7 Resultados

7.1 Decision Tree

Los árboles de decisión son modelos que generan un árbol, que toma distintos caminos entre sus nodos, basándose en las características que le fueron proporcionadas en el set de datos daod. Para este modelo se obtuvieron los resultados descritos en la Tabla 1.

n	Accuracy	Precision
1	0.498	0.47
2	0.756	0.53
3	0.957	0.93
6	0.999	0.98
10	1.000	0.99

Tabla 1: Accuracy para Decision Trees

Durante la implementación y validación de este modelo, se pudo observar una tendencia de sobreajuste pasado el valor de 4 en el parámetro de n (profundidad del árbol) debido a esto, pese a obtener mejores resultados en el accuracy a mayor número n, se decidió utilizar el valor de n=3 como el mejor valor de n obtenido. El overfitting puede verse reflejado en el siguiente gráfico:

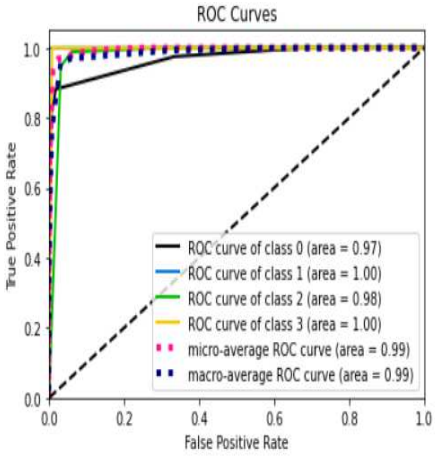

Figura 1: Curva ROC - Decision Trees

7.2 K-Nearest Neighbors

Este algoritmo genera un conjunto de clases en las cuales se clasifica la información de entrenamiento, y posteriormente, basado en lo aprendido, los datos de validación y evaluación. Para este modelo se obtuvieron los resultados descritos en la Tabla 2.

n_neighbors	Accuracy	Precision
50	0.964	1.00
200	0.951	0.96
400	0.933	0.94
500	0.918	0.93
700	0.942	0.94

Tabla 2: Accuracy para KNN

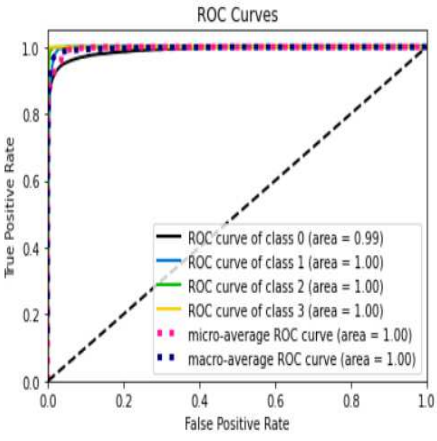

Figura 2: Curva ROC - KNN

Podemos observar que, a pesar de que la tendencia de overfitting se encuentra allí, el ajuste es ligeramente más óptimo que el del Decision Tree.

7.3 Gradient Boosting Machine

Los resultados de este modelo fueron sobreajustados desde un inicio, y aún con el cambio y optimización de parámetros los resultados continuaron siendo similares. Para este modelo se obtuvieron los resultados descritos en la Tabla 3.

kernel	Accuracy	Precision
linear	1.000	1.00

Tabla 3: Accuracy para GBM

Los resultados obtenidos con este modelo, a pesar de tener un accuracy con valor de 1, no son representativos de la eficiencia de este modelo, ya que presenta un sobreajuste prácticamente del 100% y presentará dificultades al ser expuesto a información desconocida.

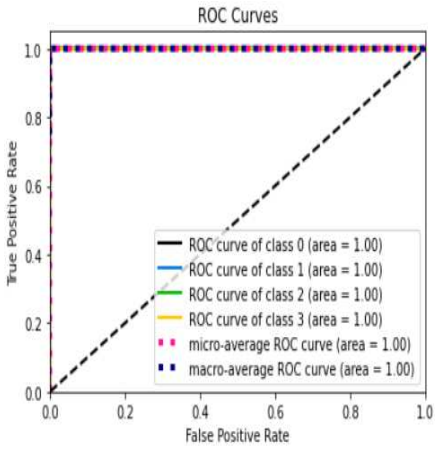

Figura 3: Curva ROC - Light GBM

8 Discusión de resultados

Luego de analizar los resultados, se obtuvo que la clasificación para los mismos fue (en orden descendente):

1. K-Nearest Neighbors
2. Decision Trees
3. GBM

Se logró identificar la eficiencia de los distintos modelos por medio de las métricas de evaluación definidas. El algoritmo de KNN fue el que mejor resultados dio, pese a presentar un accuracy más bajo que el algoritmo de Decision Trees. Se clasificaron los modelos de esta forma, ya que el accuracy más alto no necesariamente indica que el modelo sea el más eficiente y más confiable, y,

apoyándonos en las demás métricas de evaluación, podemos comprobar que los valores de precision y recall para estos dos modelos son similares, y el accuracy de KNN muestra un mejor entrenamiento más cercano a un ajuste óptimo a los datos. Por otra parte, el modelo de Gradient Boosting Machines presentó un sobreajuste del 100%, lo cual indica un mal entrenamiento y un muy probable mal rendimiento al encontrar información desconocida.

El rendimiento de los modelos en general fue bueno, sin embargo se pudo reconocer un patrón de sobreajuste en la mayoría de los modelos mientras más se trabajaba con los hiperparámetros. Esto puede atribuirse a la toma de muestras inicial, en la cual se redujo el tamaño del set de datos inicial en un 75%. De igual manera, las técnicas para combatir el desbalance como el 'undersampling' puede también provocar que el entrenamiento de los datos se vea desbalanceado, y caiga en este sobreajuste. Es recomendable realizar la implementación nuevamente con especificaciones de hardware capaces de soportar un mayor poder de cómputo.

9 Trabajo futuro

En futuras implementaciones se recomienda realizar la comparación y evaluación de los modelos con una cantidad mayor de recursos computacionales. De esta forma, se puede utilizar más información, lo cual se cree supondría una mejora en el rendimiento de los modelos. Además, también en la línea de mayor capacidad de cómputo, se recomienda considerar modificaciones en los modelos como la modificación de hiperparámetros. Se podría considerar también diferentes modelos de clasificación como XGBoost e incluso modelos de Neuronal Networks para comparar y validar el rendimiento de los modelos obtenidos en esta investigación.

10 Conclusiones

- Se determinó que el algoritmo con el mejor modelo fue el KNN, ya que presentó la mejor accuracy teniendo en cuenta el ajuste que tiene sobre los datos.
- Se concluyó que los modelos presenta un sobreajuste sobre el set de datos, lo cual fue evidenciado en la evaluación realizada con las métricas definidas.
- Se determinó que es necesaria una implementación en la que se pueda trabajar con el dataset en su completitud, para evaluar los resultados cuando no se trabaja con una porción del dataset.

Referencias

[1] Maryam M. Najafabadi; Taghi M. Khoshgoftaar; Amri Napolitano; Charles Wheelus. "RUDY Attack: Detection at the Network Level and Its Important Features". Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference.

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

[3] Hanna, K. T. (2021, 22 julio). SYN scanning. SearchNetworking. https://www.techtarget.com/search-networking/ definition/SYN -scanning

[4] Cloudflare. (2021). What is a Slowloris DDoS attack? Slowloris DDoS Attack. https://www.cloudflare.com/learning/ddos/ddos-attack -tools/slowloris/

[5] Lukaseder, T., Ghosh, S., & Kargl, F. (2018). Mitigation of Flooding and Slow DDoS Attacks in a Software-Defined Network. arxiv.org/abs/1808.05357v1

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

[7] Yu, H., & Kim, S. (2012). SVM Tutorial - Classification, Regression and Ranking. Handbook of Natural computing, 1, 479-506.