Generación De Nombres De Dominios Mediante Redes Neuronales Híbridas CNN-LSTM

Reynier Leyva La O[1,2], Rodrigo Gonzalez[1], and Carlos A. Catania[2]

[1] GridTICs, Facultad Regional Mendoza, Universidad Tecnológica Nacional, Mendoza, Argentina
rleyvalao@mendoza-conicet.gob.ar
[2] LABSIN, Facultad de Ingeniería. Universidad Nacional de Cuyo, Mendoza, Argentina
harpomaxx@gmail.com

Resumen El uso creciente de algoritmos de generación de dominios (DGA) para la comunicación entre servidores de comando y control (C&C) plantea serios desafíos en el proceso de detección de botnets. Los especialistas en seguridad deben estar en permanente actualización de las nuevas técnicas a fin de poder realizar una detección eficiente. En este trabajo, se propone el uso de una red neuronal que combina las técnicas de Convolutional Neural Network (CNN) y Long Short-Term Memory (LSTM) para generar nombres de dominios de manera algoritmica. El objetivo de contar con un generador de nombres de dominio es la de poder utilizar la información provista por el mismo en el proceso de detección de los mismos. Por lo que para su validación se realizaron pruebas a los dominios generados utilizando un detector de DGA y se encontró que los nombres generados no fueron fácilmente identificados como generados por DGA. Dicho detector mostró una exactitud de un 10 %. Estos resultados sugieren que el generador de nombres propuesto puede ser una herramienta efectiva para detectar limitaciones en las técnicas de detección de DGA.

Palabras Claves DGA, seguridad informática, CNN, LSTM, nombres de dominio.

1. Introducción

Un DGA (Domain Generation Algorithm) es un algoritmo utilizado por los ciberdelincuentes para generar nombres de dominio de forma automática y aleatoria. Su principal objetivo es establecer una comunicación segura y encubierta entre los bots y el servidor de comando y control (C&C). A diferencia de los dominios estáticos y predefinidos, que son más fáciles de detectar y bloquear, los dominios generados por DGA cambian constantemente, lo que dificulta su detección y bloqueo por parte de los sistemas de seguridad. Los DGA se utilizan comúnmente en las botnets para evitar ser detectados y para garantizar una mayor resiliencia y persistencia en las comunicaciones [1].

El funcionamiento de un DGA se basa en una semilla inicial o clave, a partir de la cual se generan secuencias de caracteres que formarán los nombres de dominio. Estas secuencias pueden variar en longitud y estructura, y se suelen utilizar técnicas de cifrado y ocultamiento para dificultar aún más su detección.

Al generar constantemente nuevos nombres de dominio, los ciberdelincuentes pueden mantener su infraestructura activa y operativa, lo que les permite controlar los bots y llevar a cabo sus actividades maliciosas sin ser detectados fácilmente.

Los DGA representan un desafío significativo para la seguridad cibernética debido a su capacidad para evadir las medidas de detección y bloqueo tradicionales.

En particular, el presente trabajo fue motivado por dos aspectos fundamentales: fortalecer la capacidad de los detectores de Dominios Generados Algoritmicamente (AGD) y explorar el potencial del aprendizaje profundo (deep learning) en la generación de nombres de dominios. Para tal fin se entrenó una red neuronal utilizando técnicas de aprendizaje supervisado. La red consta de una capa convolucional CNN que se encarga de extraer características relevantes de los nombres de dominio. Luego una capa LSTM ayuda a capturar patrones secuenciales y contextuales en la generación de los dominios. Posteriormente, se evalúan los dominios generados utilizando varios detectores de AGD existentes para ver si el modelo creado genera nombres de dominios difíciles de distinguir por los detectores. Entonces estos dominios pueden ser utilizados como casos de prueba para mejorar y fortalecer los detectores de AGD, ayudando a identificar y corregir posibles vulnerabilidades en su funcionamiento.

Es importante destacar que la generación de dominios poco distinguibles ofrece beneficios significativos en términos de fortalecimiento de los detectores de AGD y mejora de la seguridad cibernética. Por un lado, permite evaluar la eficacia de los detectores existentes y descubrir posibles vulnerabilidades, lo que impulsa su mejora continua. Además, la capacidad de generar dominios que se asemejan a los dominios legítimos aumenta la capacidad de detección y bloqueo de comunicaciones maliciosas. Sin embargo, es importante tener en cuenta las limitaciones, como el posible mal uso de esta tecnología, la necesidad de adaptar constantemente los detectores de AGD ante nuevas técnicas de generación de dominios y la posibilidad de falsos positivos y negativos que podrían afectar la precisión de los sistemas de detección.

En resumen, las contribuciones del presente estudio son por un lado explorar la factibilidad de utilizar redes neuronales en la generación de dominios y su evaluación por parte de detectores.

2. Metodología

En esta sección, se aborda el enfoque metodológico utilizado para generar dominios similares a los dominios normales. El objetivo perseguido consistió en concebir una red neuronal híbrida que combinara capas convolucionales (CNN) y memoria a largo plazo (LSTM) para generar nombres de dominio difíciles de distinguir de los dominios legítimos. Para lograrlo, se procedió a procesar los datos de dominios normales y a vectorizarlos para entrenar la red neuronal, para esto se utilizo Keras, un framework the aprendizaje profundo. Esta combinación permitió la generación de nombres de dominio que se asemejaban a los normales. A través de este proceso, se buscó fortalecer la detección de AGD y mejorar la seguridad cibernética en general. Se utilizó una amplia cantidad de dominios normales para un entrenamiento óptimo. Este enfoque proporciona una base sólida para mejorar la detección de AGD y fortalecer la seguridad contra posibles ataques de botmasters.

2.1. Descripción del Conjunto de Datos Utilizado

Para este trabajo se utilizó un dataset que contenía tanto AGD como dominios normales. Los dominios normales fueron extraídos de los 1 millón de dominios más populares según Alexa [24], y se incluyeron adicionalmente 3.161 dominios normales proporcionados por el feed de Bambenek Consulting [23]. Este último grupo de dominios es particularmente interesante, ya que consiste en dominios sospechosos que no fueron generados por DGA. En total, la base de datos contenía 1.003.161 dominios normales [19].

Por otro lado, los dominios generados por DGA se obtuvieron de los repositorios de dominios AGD de Andrey Abakumov [22] y John Bambenek. En total, había 1.915.335 dominios DGA que correspondían a 51 familias diferentes de malware. Cabe destacar que aproximadamente el

55 % de la porción DGA del dataset estaba compuesta por muestras de las familias de malware `Banjori`, `Post`, `Timba`, `Cryptolocker`, `Ramdo` y `Conficker`.

En el proceso de generación de dominios, se optó por utilizar únicamente los dominios normales de este dataset, ya que el objetivo era entrenar la red neuronal para generar nombres de dominio similares a los dominios normales.

Con base en el conjunto de datos descrito anteriormente, en la subsección 2.2, se aborda detalladamente el proceso de preprocesamiento de los datos. Este paso es fundamental para preparar los dominios normales como entrada para la red neuronal y asegurar que la generación de nombres de dominio se realice de manera eficiente y efectiva.

2.2. Preprocesamiento de los Datos

Es esencial resaltar la importancia del preprocesamiento de los dominios como paso fundamental para prepararlos adecuadamente antes de introducirlos en la red neuronal. Este proceso de preparación de los datos juega un papel crucial en asegurar que la red pueda comprender y procesar la información de manera efectiva. Al realizar el preprocesamiento, los dominios son transformados en formatos adecuados y estructurados, lo que facilita su manejo por parte de la red neuronal. Además, durante esta etapa, se llevan a cabo transformaciones y codificaciones necesarias para representar los dominios en forma de secuencias vectorizadas, permitiendo que la red neuronal aprenda las relaciones entre los caracteres y sus posiciones dentro de la secuencia de entrada. Mediante un preprocesamiento adecuado, se logra maximizar la calidad del entrenamiento y la capacidad de la red para generar nombres de dominio similares a los dominios normales. Esto resulta esencial para el cumplimiento del objetivo principal del estudio.

Para llevar a cabo el preprocesamiento de los dominios, se completó cada dominio para que todos tuvieran una longitud específica. Esto se logró mediante la adición de caracteres especiales (espacios) al final de cada dominio, siempre y cuando la longitud del dominio fuera menor que la deseada. De esta manera, todos los dominios alcanzaron la misma longitud deseada. Luego, los dominios completados se unieron en una sola cadena de texto, utilizando el carácter espacio como separador entre cada dominio. Este proceso de preprocesamiento garantizó que los dominios tuvieran una estructura uniforme.

Una vez obtenida la cadena de texto completa, se procedió a dividirla en secuencias más pequeñas con un desplazamiento de 3 caracteres en cada iteración. De esta manera, se generaron las secuencias de entrada (`sec_X`) junto con sus correspondientes caracteres siguientes (`car_y`). Un ejemplo básico para entender cómo estaban compuestas las secuencias de entrada (`sec_X`) y el carácter de salida (`car_y`) se puede visualizar en la tabla 1.

Tabla 1. Secuencia de entrada y carácter de salida

No.	Secuencia Entrada	Carácter Salida
1	www.google.com	“ ”
2	www.google.	“c”
3	www.goog	“l”

Una vez obtenidas las secuencias de entrada y el carácter de salida, se procedió a aplicar tokenización a dichos datos. El proceso de tokenización es un componente fundamental en el preprocesamiento de datos para la generación de dominios. Al convertir cada carácter único en un token, se logra una representación más compacta y procesable para la red neuronal. Esto

reduce significativamente la complejidad computacional del modelo y acelera el proceso de entrenamiento, ya que la red solo necesita aprender los patrones de los tokens en lugar de lidiar con cada carácter individual. Además, la tokenización permite una mejor generalización del modelo, ya que la red puede aprender a reconocer y manipular patrones comunes de caracteres, en lugar de memorizar cada carácter específico presente en los datos de entrenamiento.

En el contexto de este trabajo, una vez obtenidos los datos de entrada y salida, se llevó a cabo la identificación de los caracteres únicos presentes en las secuencias, formando así un alfabeto. A cada carácter se le asignó un índice correspondiente en dicho alfabeto. Para representar los caracteres de manera procesable por la red neuronal, se implementó la codificación one-hot [11], [8]. Mediante esta codificación, cada carácter se transformó en un vector disperso donde todas las posiciones eran 0, excepto una que contenía el valor 1 en la posición correspondiente al índice del carácter en el alfabeto. De esta forma, cada carácter se convirtió en un vector único y procesable por la red, lo que permitió que la red neuronal aprendiera eficientemente las relaciones entre los caracteres en las secuencias de entrada.

Este enfoque de codificación one-hot posibilitó la creación de un conjunto de datos adecuado para el entrenamiento y ajuste de la red neuronal, proporcionando la información necesaria para predecir los caracteres siguientes en función de las secuencias de entrada. Asimismo, la codificación one-hot permitió que la red comprendiera y procesara los caracteres de manera efectiva. Este proceso de tokenización y codificación fue fundamental para la eficiencia del modelo en la generación de dominios.

A continuación, se detallará la red neuronal utilizada en la arquitectura para la generación de dominios similares, en la subsección 2.3.

2.3. Descripción de la Red Neuronal

Como se mencionó en la subsección 2.2, en esta subsección se describirá en detalle la red neuronal utilizada para la generación de dominios similares a los normales.

En el proceso de selección de las redes neuronales para la arquitectura, se tuvo en cuenta la naturaleza secuencial de los datos y la necesidad de capturar características importantes de los dominios normales. Para lograr este objetivo, se optó por emplear capas de redes neuronales convolucionales (CNN) y redes de memoria de corto-largo plazo (LSTM).

Las redes neuronales convolucionales son ampliamente conocidas por su capacidad para extraer características relevantes de los datos de entrada, especialmente en imágenes y secuencias [11], [7]. En el contexto de generación de dominios, estas capas fueron fundamentales para capturar patrones y características clave presentes en los dominios normales. Permitieron que la red aprendiera las estructuras esenciales necesarias para generar dominios similares y coherentes.

Por otro lado, redes de memoria de corto-largo plazo (LSTM) son un tipo de red neuronal recurrente (RNN) diseñada para trabajar con secuencias de datos [12]. Estas capas resultaron esenciales para capturar dependencias a largo plazo en los datos de entrada, un aspecto crítico para la generación de secuencias de texto coherentes y realistas. La inclusión de LSTM en la arquitectura permitió que la red aprendiera las relaciones temporales y las dependencias complejas entre los caracteres en las secuencias de entrada.

La combinación de capas de redes neuronales convolucionales y capas LSTM en una sola red neuronal híbrida resultó en una arquitectura efectiva para la generación de dominios similares a los dominios normales. Cada capa desempeñó un papel importante en el proceso, permitiendo que la red aprendiera y generara secuencias de texto de manera coherente y realista. Estas características contribuyeron significativamente al logro del objetivo principal del estudio.

La arquitectura en detalle seleccionada fue la siguiente:

1. **Capa de entrada:** Se definió una capa de entrada con la forma correspondiente a la longitud máxima de las secuencias y el número de caracteres únicos en los datos.
2. **Capa convolucional 1D:** Se agregó una capa convolucional con 128 filtros y un tamaño de kernel de 3, con una función de activación 'relu'. Esta capa ayudó a extraer características relevantes de las secuencias de entrada.
3. **Capa de agrupación máxima 1D:** Se añadió una capa de agrupación máxima con un tamaño de agrupación de 2. Esta capa permitió reducir la dimensionalidad de las características extraídas por la capa convolucional.
4. **Capa LSTM:** Se incluyó una capa LSTM con 128 unidades. La capa LSTM permitió capturar las dependencias temporales en las secuencias de entrada.
5. **Capa densa:** Se agregó una capa densa con una función de activación 'softmax' para la clasificación de los caracteres. El número de unidades en esta capa corresponde al número de caracteres únicos en los datos.

Con los tipos de redes neuronales previamente definidos, el enfoque se dirige ahora hacia la implementación y entrenamiento de la arquitectura, haciendo uso de la biblioteca Keras. Esta herramienta proporciona las funcionalidades necesarias para aprovechar al máximo la capacidad de generación de texto. En la siguiente subsección (2.4), se aborda en detalle la utilización de Keras y cómo esta biblioteca facilita concretamente la generación de nombres de dominio. Este aspecto es fundamental para comprender exhaustivamente el proceso de generación de dominios y para obtener una comprensión completa de los resultados obtenidos en este estudio.

2.4. Algoritmo para generación de Dominios

En la presente subsección, se expone la implementación del código empleado en la generación de dominios mediante el empleo de Keras. En esta exposición, se abordarán minuciosamente los pormenores concernientes a la arquitectura de la red neuronal híbrida, así como las configuraciones y parámetros particulares que fueron empleados para el adiestramiento del modelo. El enfoque primordial se centra en la manera en que la biblioteca Keras fue utilizada con el propósito de materializar la generación de dominios que presentaran similitud con aquellos considerados como normales.

Para el entrenamiento del modelo, se utilizó el optimizador **Adam** [20] con una tasa de aprendizaje de 0.01. El modelo fue compilado utilizando la función de pérdida **categorical_crossentropy** [21]. Con esta estructura de modelo, se buscó que la red neuronal aprendiera a generar nombres de dominios similares a los dominios normales de manera efectiva, contribuyendo así al objetivo principal del estudio.

Para obtener más detalles sobre la implementación y creación del modelo, el código se encuentra disponible en el repositorio público de GitHub [18]. En la porción de código presentada en la Figura 1, se exhibe la sección pertinente del código que corresponde a la creación del modelo:

Durante el proceso de entrenamiento de la red neuronal, se realizaron varias pruebas con distintos valores de épocas, tamaño de batch, diversidad, cantidad de neuronas por capa, y métodos de optimización, entre otros hiperparámetros. Sin embargo, la configuración de modelo que se mencionó previamente demostró ser la más efectiva. Después de cada época, se generaba nuevo texto para evaluar el rendimiento del modelo en la generación de nombres de dominio. Esta estructura permitió que la red neuronal generara de manera consistente y realista nombres de dominios. Para obtener información más detallada acerca de la implementación del entrenamiento del modelo descrito en la Figura 1, se proporciona el código correspondiente al proceso de entrenamiento en la Figura 2:

En cada época se elegía aleatoriamente una secuencia inicial de la cadena de texto y se utilizaba un valor de diversidad de 0.5. A partir de esta secuencia inicial, la red neuronal previamente

Código fuente utilizado para la Creación Del Modelo

```
model = keras.Sequential([
   keras.Input(shape=(maxlen, len(chars))),
   layers.Conv1D(128, kernel_size=3, activation='relu'),
   layers.MaxPooling1D(pool_size=2),
   layers.LSTM(128),
   layers.Dense(len(chars), activation='softmax')
])

optimizer = Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer)
```

Figura 1. Código fuente del modelo utilizado para la creación del modelo.

entrenada generaba caracteres adicionales para completar la secuencia de texto. Se generaban específicamente 400 caracteres en los cuales se encontraban 9 nombres de dominios. Estos dominios generados se utilizaban para evaluar cómo el modelo aprendía a generar nombres de dominio similares a los dominios normales a lo largo del proceso de entrenamiento. Esta evaluación cualitativa permitía analizar el rendimiento del modelo y observar cómo mejoraba la coherencia y realismo de los dominios generados con el aumento de las épocas de entrenamiento.

Es relevante resaltar que, aunque el modelo recibe una semilla inicial para la generación de dominios, el proceso se basa en la combinación de caracteres para producir nombres de dominio. En consecuencia, los primeros dominios generados tendrán una fuerte relación con la semilla, pero a medida que el proceso avanza, los dominios subsiguientes se verán influenciados por los dominios generados previamente. Esta dinámica crea una relación continua y progresiva entre los dominios generados, lo que resulta en una mayor variedad y complejidad en la secuencia de caracteres utilizada.

Con el código y la arquitectura de la red neuronal mostrada anteriormente, se obtiene un modelo optimizado capaz de generar nombres de dominio similares a los dominios normales. Este modelo cumple con el objetivo del estudio, que es generar dominios que sean difíciles de distinguir de los dominios legítimos, lo que permite explorar y comprender mejor las posibles vulnerabilidades y riesgos asociados a la seguridad informática. La combinación de técnicas como las capas de redes neuronales convolucionales y LSTM, junto con la codificación one-hot y la generación iterativa de dominios, proporciona una potente herramienta para generar nombres de dominio realistas y coherentes, lo que contribuye significativamente al avance en el campo de la seguridad cibernética.

2.5. Pruebas de Detección a los Dominios Generados

Una vez que el modelo fue entrenado y los parámetros ajustados, se llevó a cabo la generación de dominios utilizando una semilla específica. Estos dominios generados fueron posteriormente evaluados mediante el uso de diversos detectores. Para llevar a cabo esta evaluación, se seleccionó un dominio normal de la base de datos original y se generaron un total de 4000 dominios utilizando dicha semilla. A continuación, se presentan 25 ejemplos representativos de los 4000 dominios generados:

Porción de código fuente utilizado en el generador de dominios.

```
epochs = 20
batch_size = 128

for epoch in range(epochs):
   model.fit(x, y, batch_size=batch_size, epochs=1)
   print()
   print("Generating text after epoch: %d" % epoch)

   start_index = random.randint(0, len(text) - maxlen - 1)
   for diversity in [0.5]:
      print("...Diversity:", diversity)
      generated = ""
      sentence = text[start_index : start_index + maxlen]
      print('...Generating with seed: "' + sentence + '"')

      for i in range(400):
         x_pred = np.zeros((1, maxlen, len(chars)))
         for t, char in enumerate(sentence):
            x_pred[0, t, char_indices[char]] = 1.0
         preds = model.predict(x_pred, verbose=0)[0]
         next_index = sample(preds, diversity)
         next_char = indices_char[next_index]
         sentence = sentence[1:] + next_char
         generated += next_char
      print("...Generated: ", generated)
      print()
```

Figura 2. Porción de código fuente utilizado en el generador de dominios.

- aatuaeic.eo
- aii..uo
- bstrcuc.oz
- aatkrst.to
- aaiaea.cem
- aa.cx
- bena.io
- aaam..no.u
- btiareraci..oi
- oloraoire.com
- cbroeis.ae
- abrrunec..omg
- baekeranr.uccc
- cec..ov
- wbece.com
- bask.lo
- a3ac..oz
- 3a-ic.ao
- baadpibra..oog
- aaitt.ge
- bgs.-o
- aepan.ec
- ccressn..oo.r
- ain.so
- bsleca.eo

Estos ejemplos muestran la diversidad y variedad de los dominios generados a partir de la semilla proporcionada, demostrando la capacidad del modelo para generar nombres de dominio de forma automática y aleatoria.

Una vez generados los dominios, se procedió a evaluar su detección mediante el uso de un Detector de Dominios Generados Algoritmicamente (AGD). Con el objetivo de medir la calidad de las predicciones del detector, se empleó la matriz de confusión y diversas métricas de evaluación. Para llevar a cabo esta evaluación, se utilizó el modelo predictor propuesto por Catania et al. [7]. Cabe resaltar que este detector ha demostrado resultados prometedores en investigaciones previas, lo que lo hace una elección adecuada para evaluar la efectividad del modelo de generación de dominios en cuestión.

La matriz de confusión se muestra en la figura 3, y refleja que de los 4000 dominios generados, solo 399 fueron correctamente detectados como AGD. Las métricas de evaluación obtenidas se muestran en la tabla 2.

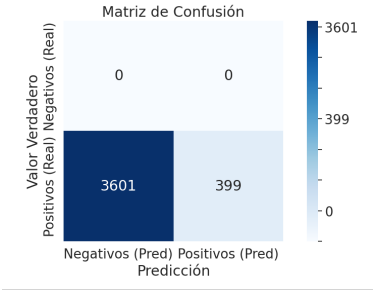

Figura 3. Matriz de Confusión

Tabla 2. Métricas de evaluación del Detector de Dominios Generados Algoritmicamente (AGD)

Métrica	**Valor**
Accuracy (exactitud)	0.10
Precision	1.00
Recall	0.10
F1-score	0.18
Soporte (cantidad de instancias)	4000

Estas métricas proporcionan una evaluación detallada del desempeño del detector de AGD utilizado. La precisión indica la proporción de dominios detectados como AGD que fueron clasificados correctamente. El recall representa la proporción de dominios reales que fueron detectados correctamente como AGD. El F1-score combina la precisión y el recall en una única métrica que pondera ambas medidas. El soporte indica la cantidad total de instancias evaluadas.

Resumiendo, si vemos estos resultados en conjunto, el detector tuvo un 100 % de precisión, lo que significa que cuando dijo que algo era un dominio generado algoritmicamente, generalmente tenía razón. Pero, desafortunadamente, solo logró encontrar el 10 % de los dominios que realmente eran generados de esta forma. Esto muestra que el detector necesita mejorar para ser más eficaz en la detección de este tipo de dominios.

2.6. Análisis Estadístico de los Dominios

El estudio de las características estadísticas de los dominios proporciona una visión esencial sobre las tendencias en los nombres de dominio. Al analizar la frecuencia de aparición de caracteres, se identificaron patrones significativos. El carácter más recurrente es 'o', presente en un total de 6.265 ocasiones, seguido de cerca por el punto '.' con 6.262 apariciones. Asimismo, la letra 'c' se presenta comúnmente con 5.596 apariciones, seguida de la letra 'a' con 4.746 ocurrencias. Por otro lado, se encontraron letras menos frecuentes, como la 'x', que solo aparece 5 veces, y el guión bajo '_' con tan solo 1 aparición.

Otra característica relevante es la longitud promedio de los dominios, la cual se calculó en aproximadamente 10,51 caracteres. Este dato es valioso, ya que nos brinda una comprensión general de la extensión típica de los nombres de dominio estudiados. Un promedio de 10,51 caracteres sugiere que los nombres de dominio tienden a ser relativamente cortos, lo que facilita su memorización y escritura.

El análisis de la frecuencia de aparición de n-gramas (n = 2) también aporta información valiosa sobre la presencia y repetición de ciertas combinaciones de caracteres en los dominios analizados. Algunos de los n-gramas más frecuentes incluyen **cc** con 1.746 apariciones, **..** y **.o** con 1.377 y 1.391 apariciones, respectivamente. Asimismo, se encontraron combinaciones como **oo** con 1.124 apariciones, **aa** con 769 apariciones y **co** con 765 apariciones. Estos n-gramas pueden revelar patrones comunes en los nombres de dominio, lo que podría facilitar el análisis y la clasificación de futuros dominios.

Por último, se examinó la distribución de caracteres en las posiciones iniciales y finales de los dominios. En la posición inicial, los caracteres más frecuentes son 'c', 'a' y 'b', mientras que los menos comunes son 'j', '6', 'y' y 'z'. En cuanto a la posición final, el carácter más recurrente es 'o', seguido por 'm' y 't'. Por otro lado, los caracteres 'w', '9', '3' y 'x' presentan la menor frecuencia al final de los dominios.

En conjunto, este análisis estadístico nos proporciona una visión más clara de las características de los nombres de dominio examinados, permitiéndonos comprender mejor las tendencias y particularidades que influyen en la elección de estos identificadores en la web.

3. Conclusiones

En conclusión, este estudio investigó la generación de dominios utilizando una red neuronal híbrida compuesta por capas de convolución neuronal (CNN) y memoria de corto-largo plazo (LSTM). Los resultados obtenidos demostraron que los dominios generados presentaron una notable similitud con los dominios legítimos, lo que dificultó su detección por parte del detector de Dominios Generados Algoritmicamente (AGD) utilizado en las pruebas.

El enfoque de generación de dominios utilizado en este estudio tiene el potencial de identificar vulnerabilidades en los sistemas de detección de AGD y mejorar las técnicas de defensa contra ataques de botmasters. La combinación de capas CNN y LSTM en la red neuronal proporcionó resultados prometedores, lo que sugiere que este enfoque puede ser explorado aún más en futuras investigaciones.

Además, los dominios generados en este estudio sirven para aumentar la base de datos de dominios maliciosos, lo cual puede ser de gran utilidad en la preparación y desarrollo de detectores más robustos en investigaciones futuras.

La estadística muestra la frecuencia de apariciones de diferentes dominios de nivel superior (TLDs) en un conjunto de datos. El dominio `".com"` encabeza la lista con 506 apariciones. Luego, los dominios `".oo"` y `".eo"` tienen 255 y 212 apariciones, respectivamente. Se puede observar que la mayoría de los TLDs tienen menos de 100 apariciones, lo que sugiere que hay una amplia variedad de TLDs utilizados en este conjunto de datos. Algunos TLDs menos comunes tienen solo una aparición. Es importante destacar que hay TLDs conocidos como `".org"` y `".net"` con solo 6 y 1 apariciones, respectivamente.

En resumen, este trabajo contribuye al campo de la detección de AGD al presentar un enfoque para la generación de dominios que no son fácilmente detectables. Se espera que estos hallazgos impulsen el desarrollo de nuevas estrategias de defensa y fortalezcan la seguridad informática en la lucha contra los ataques de botmasters. Se reconoce que aún existen desafíos en la detección de dominios generados algorítmicamente, y se alienta a futuras investigaciones a seguir avanzando en este campo para asegurar la efectiva protección de los sistemas informáticos.

4. Referencias Bibliográficas

Referencias

1. Plohmann, D., Yakdan, K., Klatt, M., Bader, J., Gerhards-Padilla, E.: A comprehensive measurement study of domain generating malware. In: 25th USENIX Security Symposium (USENIX Security 16), pp. 263–278. USENIX Association, Austin, TX (2016)
2. Wang, Z., Guo Y.: Neural networks based domain name generation. Journal of Information Security and Applications, vol. 61, pp. 102948 (2021)
3. Schüppen S, Teubert D, Herrmann P, Meyer U. FANCI: Feature-based automated nxdomain classification and intelligence. In: Proceedings of the 27th USENIX Security Symposium, pp. 1165–1181. USENIX Association (2018)
4. Vij P, Nikam S, Bhatia A. Detection of algorithmically generated domain names using LSTM. In: 2020 International Conference on Communication Systems and Networks, COMSNETS 2020, pp. 1–6 (2020)
5. Zhou S, Lin L, Yuan J, Wang F, Ling Z, Cui J. CNN-Based DGA detection with high coverage. In: 2019 IEEE International Conference on Intelligence and Security Informatics, ISI 2019, pp. 62–67 (2019)
6. Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM networks. Proc Int Joint Conf Neural Netw 2005;4(5–6):2047–2052
7. Catania, C., Garcia, S., Torres, P.: An analysis of convolutional neural networks for detecting DGA. In: Proceedings of XXIV Congreso Argentino de Ciencias de la Computación, La Plata, pp. 1–10 (2018)
8. Chollet, F.: Generate text from Nietzsche's writings with a character-level LSTM. Keras Examples (2015). Disponible en: `https://keras.io/examples/generative/lstm_character_level_text_generation/` (Fecha de acceso: 23/06/2023).
9. Catania, C.A.: Using CNN for a Domain name Generation Algorithm (1) (2023). Disponible en: `https://harpomaxx.github.io/post/dga-convnet/` (Fecha de acceso: 23/06/2023).
10. Catania, C.A.: Using CNN for a Domain name Generation Algorithm (2) (2023). Disponible en: `https://https://harpomaxx.github.io/post/dga-convnet-p2/` (Fecha de acceso: 23/06/2023).
11. Chollet, F.: Deep Learning con Python. En: Aprende Machine Learning con Python, pp. 325–360. Anaya Multimedia (2021). ISBN: 978-84-415-4351-3.
12. Klaus Greff, Rupesh K. Srivastava, and Jürgen Schmidhuber. "LSTM: A Search Space Odyssey.ÏEEE Transactions on Neural Networks and Learning Systems 28, no. 10 (2017): 2222-2232.
13. Banjori's DGA. 2021, johannesbader.ch/blog/the-dga-of-banjori/ (Accessed April 12, 2021).
14. Conficker's DGA. 2021, //www.honeynet.org/files/KYE-Conficker.pdf (Accessed April 12, 2021).
15. Symantec. (2015). Timba - Spanish speaking banker targets Chilean banks. Symantec Blogs. `https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/timba-spanish-speaking-banker-targets-chilean-banks`.
16. Symantec. (2013). CrytoLocker Ransomware. Symantec Blogs. `https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/crypto-ransomware-crime-unstoppable`.
17. Ramdo's DGA. 2021, www.securityweek.com/ramdo-click-fraud-malware- continues-evolve (Accessed April 12, 2021).
18. Leyva, Reynier; Catania, Carlos A. (2023). Generador de Dominios. GitHub. URL: https://github.com/reypapin/Generador-de-Dominios (Consultado el 2 de agosto de 2023).
19. Carlos A. Catania, "DGA Detection Dataset,"Hugging Face Datasets, 2023. URL: https://huggingface.co/datasets/harpomaxx/dga-detection
20. Diederik P. Kingma and Jimmy Lei Ba. .ªdam: A Method for Stochastic Optimization."Published as a conference paper at ICLR 2015.
21. TensorFlow. tf.keras.losses.CategoricalCrossentropy. `https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy`. Accessed August 3, 2023.
22. Aeva, A. (2017). DGA. GitHub. `https://github.com/andrewaeva/DGA`.
23. Bambenek Consulting. (n.d.). DGA Domain Feeds. `https://faf.bambenekconsulting.com/feeds/`.
24. Website ranking.(Accessed April 12, 2021.),`s3.amazonaws.com/alexa-static/top-1m.csv.zip`,