# Agente autónomo de ciberseguridad basado en aprendizaje por refuerzo

Javier Rivero Iglesias

# Índice general

# Capítulo 1

# Introducción

En este Trabajo Fin de Grado se explora el uso de técnicas de aprendizaje por refuerzo (*Reinforcement Learning*, RL) para el diseño de agentes defensivos en ciberseguridad. El objetivo global es desarrollar un agente capaz de tomar decisiones de defensa de forma autónoma y adaptativa frente a tráfico malicioso, partiendo inicialmente de datos etiquetados y, en fases posteriores, de entornos de red más dinámicos.

En esta primera versión de la memoria se recoge, sobre todo, la fase experimental centrada en la clasificación de ataques utilizando el conjunto de datos NSL-KDD, comparando un agente RL con modelos supervisados clásicos.

# Capítulo 2

# Definición del problema

El problema que se aborda se puede dividir en dos fases complementarias:

**F1. Fase 1: clasificación de tráfico malicioso**.
A partir de un conjunto de datos etiquetado de tráfico de red (NSL-KDD), se plantea la detección binaria *normal* / *ataque* formulada como un problema de decisión para un agente RL. En esta fase se estudia hasta qué punto un agente entrenado por refuerzo puede aproximarse (o no) al rendimiento de modelos supervisados como Random Forest.

**F2. Fase 2: defensa en un entorno de red simulado**.
Una vez validada la viabilidad del enfoque y de la infraestructura de entrenamiento, el objetivo es extender el agente al control de mecanismos de defensa (por ejemplo, reglas de cortafuegos, políticas de bloqueo o limitación de conexiones) en un entorno de red emulado, donde las decisiones tienen un efecto secuencial en el tiempo.

En este documento se desarrolla principalmente la Fase 1, dejando la Fase 2 como línea de trabajo en curso.

# Capítulo 3

# Trabajos relacionados

En esta sección se revisarán trabajos previos sobre:

- Sistemas de detección de intrusos (IDS) basados en aprendizaje supervisado.
- Aplicación de Deep Reinforcement Learning a la detección de intrusos y a la defensa autónoma en redes.
- Conjuntos de datos de referencia en ciberseguridad (NSL-KDD, UNSW-NB15, CIC-IDS2017, etc.).

# Capítulo 4

# Clasificación de ataques con aprendizaje por refuerzo

## 4.1. Descripción del conjunto de datos NSL-KDD

Para la fase de clasificación se ha utilizado el conjunto de datos NSL-KDD, que es una revisión del dataset KDD'99 clásico, eliminando redundancias y problemas de desbalance extremo presentes en la versión original. NSL-KDD incluye registros de conexiones de red etiquetados como *normal* o distintos tipos de ataque (DoS, probe, R2L, U2R, etc.), junto con características estadísticas de cada flujo.

En este trabajo se considera una versión binaria del problema, en la que se agrupan todas las variantes de ataque en una única clase *malicious*, mientras que las conexiones etiquetadas como normales se mantienen en la clase *benign*. La motivación es simplificar inicialmente la tarea a una decisión de tipo *permitir/bloquear* que resulte natural para un agente de defensa.

### 4.1.1. Partición de entrenamiento y prueba

El preprocesado se realiza en dos etapas:

1) Se combinan los ficheros de entrenamiento (`KDDTrain+.TXT`) y prueba (`KDDTest+.TXT`) originales de NSL-KDD.

2) Se mantiene la partición que propone el propio dataset:

   - Conjunto de entrenamiento: aproximadamente 125 973 instancias.
   - Conjunto de prueba: aproximadamente 22 544 instancias.

Durante la fase de iteración rápida se utiliza también la variante reducida `KDDTrain+_20Percent.TXT` como conjunto de entrenamiento, manteniendo el conjunto de prueba completo. Esta configuración reduce el coste de cómputo permitiendo explorar distintas funciones de recompensa y arquitecturas del agente RL.

## 4.2. Preprocesado y representación de las observaciones

Cada muestra del dataset incluye características numéricas y categóricas (`protocol_type`, `service`, `flag`, etc.). El preprocesado implementado realiza:

- Codificación *one-hot* de las variables categóricas.
- Conversión de la etiqueta textual original (`normal` frente a tipos de ataque) a una etiqueta binaria:

$$y = \begin{cases} 0, & \text{si la conexión es normal (benigna),} \\ 1, & \text{si la conexión corresponde a cualquier tipo de ataque.} \end{cases}$$

- Normalización de las características numéricas mediante escalado estándar cuando se utilizan modelos supervisados clásicos.

Tras este preprocesado se obtiene una matriz de características $\mathbf{X} \in \mathbb{R}^{N \times d}$ y un vector de etiquetas $\mathbf{y} \in \{0,1\}^N$, donde $N$ es el número de muestras y $d$ el número de características tras la codificación.

## 4.3. Formulación RL del problema de clasificación

En lugar de entrenar un modelo supervisado que aproxime directamente la probabilidad $\mathbb{P}(y \mid \mathbf{x})$, se plantea el problema como una tarea de decisión para un agente de aprendizaje por refuerzo que interactúa con un entorno de tipo Gym.

### 4.3.1. Definición del entorno `RLDatasetDefenderEnv`

Se define un entorno `RLDatasetDefenderEnv`, implementado en Python sobre Gymnasium, donde cada episodio corresponde a un recorrido secuencial por una permutación de las muestras del conjunto de entrenamiento.

En este entorno:

- **Estado (observación)**: el vector de características $\mathbf{x}_i \in \mathbb{R}^d$ de la muestra $i$-ésima.
- **Acciones**: el agente dispone de dos acciones discretas:

$$\begin{aligned} a = 0 &: \text{ PERMIT (permitir el tráfico)} \\ a = 1 &: \text{ BLOCK (bloquear el tráfico).} \end{aligned}$$

- **Etiqueta real**: la etiqueta binaria asociada a la muestra se denota $y_i \in \{0,1\}$, donde 0 representa tráfico benigno y 1 tráfico malicioso.

### 4.3.2. Función de recompensa sensible al coste

La clave del enfoque RL es definir una función de recompensa que refleje el coste operativo de las decisiones del agente. En este trabajo se emplea una función de recompensa binaria sensible al coste de los falsos negativos y falsos positivos, configurada mediante un diccionario de parámetros:

| Parámetro | Significado |
|---|---|
| `tp` | Recompensa por bloquear correctamente un ataque (TP). |
| `fp` | Penalización por bloquear tráfico benigno (FP). |
| `fn` | Penalización por permitir un ataque (FN). |
| `omission` | Recompensa parcial por permitir tráfico benigno. |

En la versión actual se ha trabajado, entre otras, con la siguiente configuración:

$$\text{reward\_config} = \{\texttt{tp} = 1.0,\ \texttt{fp} = -1.0,\ \texttt{fn} = -5.0,\ \texttt{omission} = 0.5\}. \tag{4.1}$$

Bajo esta configuración, la recompensa instantánea $r_i$ para la muestra $i$ se define como:

$$r_i = \begin{cases} \texttt{tp}, & \text{si } y_i = 1 \text{ y } a_i = 1 \quad \text{(ataque bloqueado)} \\ \texttt{fn}, & \text{si } y_i = 1 \text{ y } a_i = 0 \quad \text{(ataque permitido)} \\ \texttt{omission}, & \text{si } y_i = 0 \text{ y } a_i = 0 \quad \text{(benigno permitido)} \\ \texttt{fp}, & \text{si } y_i = 0 \text{ y } a_i = 1 \quad \text{(benigno bloqueado).} \end{cases}$$

Es decir, bloquear un ataque se recompensa con $+1.0$, permitirlo se castiga severamente con $-5.0$, bloquear tráfico benigno con $-1.0$ y permitir tráfico benigno se considera una *omisión acertada* con una recompensa parcial de $+0.5$.

Esta formulación permite explorar explícitamente el compromiso entre la reducción de falsos negativos (seguridad) y la minimización de falsos positivos (disponibilidad del servicio).

## 4.4. Agente DQN y modelo Random Forest de referencia

### 4.4.1. Agente DQN

El agente de aprendizaje por refuerzo se implementa mediante un algoritmo *Deep Q-Network* (DQN), usando la librería Stable-Baselines3. Se emplea una política de tipo `MlpPolicy`, basada en una red neuronal totalmente conectada con varias capas ocultas y función de activación ReLU.

Los hiperparámetros principales utilizados en los experimentos son:

- Tasa de aprendizaje: $10^{-3}$.
- Tamaño del *replay buffer*: 100 000 transiciones.
- Tamaño de lote: 64.

- Factor de descuento $\gamma = 0.99$.
- Frecuencia de entrenamiento: cada 4 pasos de interacción.
- Actualización de la red objetivo cada 10 000 pasos.

El entrenamiento se realiza durante entre 200 000 y 1 000 000 pasos de interacción, dependiendo del experimento, y se registra tanto la evolución de la recompensa promedio como las métricas de clasificación sobre el conjunto de prueba.

### 4.4.2. Modelo Random Forest de referencia

Como referencia supervisada se entrena un clasificador Random Forest sobre el mismo preprocesado del conjunto NSL-KDD. El modelo se entrena con un número moderado de árboles (por ejemplo, 200 estimadores) y sin optimización exhaustiva de hiperparámetros, ya que el objetivo principal es disponer de un punto de comparación razonable frente al agente RL.

El Random Forest opera directamente en modo supervisado, minimizando el error de clasificación sobre el conjunto de entrenamiento, mientras que el agente DQN aprende a partir de señales de recompensa derivadas de las decisiones PERMIT/BLOCK.

## 4.5. Resultados experimentales

En esta sección se resumen los resultados obtenidos para diferentes configuraciones del agente DQN y del modelo Random Forest. Cada experimento se identifica con un código (`E01`, `E02`, ...) y se documenta en detalle en el fichero `experiments/nslkdd_experiments.md` del repositorio.

### 4.5.1. Métricas de evaluación

Las métricas que se emplean para evaluar el rendimiento son las habituales en problemas de detección de intrusos:

- **Accuracy** global.
- **Precision**, **recall** y **F1-score** para cada clase.
- Matriz de confusión, con especial atención a:
  - Falsos negativos (ataques permitidos).
  - Falsos positivos (tráfico benigno bloqueado).
- Tasa de falsos positivos (FP rate), definida como:

$$\text{FP rate} = \frac{\text{FP}}{\text{TN} + \text{FP}}.$$

### 4.5.2. Comparación DQN vs Random Forest (resumen cualitativo)

De forma resumida, los resultados obtenidos muestran que:

- El modelo Random Forest alcanza una *accuracy* en torno al 77 % y un *recall* de la clase de ataque cercano al 0.61, con una tasa de falsos positivos baja (del orden del 2–3 % del tráfico benigno).
- El agente DQN, incluso tras incrementar el castigo por falsos negativos y ajustar la recompensa por omisión, obtiene *accuracies* en el rango 71–76 %, con un *recall* de ataques inferior al del Random Forest y una tasa de falsos positivos ligeramente inferior.
- En otras palabras, el agente RL tiende a aprender políticas muy conservadoras respecto a los falsos positivos (bloquea muy poco tráfico benigno), pero a costa de permitir un porcentaje relativamente elevado de ataques, lo que limita su utilidad como IDS principal.

Estos resultados son coherentes con la naturaleza del problema en esta fase: se trata de una clasificación estática sobre un conjunto tabular etiquetado, donde los modelos supervisados clásicos están muy bien adaptados. El interés del enfoque RL aparece cuando se pasa a un entorno dinámico en el que las decisiones del agente tienen efectos a más largo plazo, como se plantea en la fase siguiente del TFG.

# Capítulo 5

# Conclusiones y líneas futuras

En esta primera versión de la memoria se han presentado los fundamentos del TFG y la fase inicial de experimentación basada en NSL-KDD. Como líneas futuras inmediatas destacan:

- Completar la comparación cuantitativa entre distintas funciones de recompensa para el agente DQN, documentando el compromiso entre falsos positivos y falsos negativos.
- Extender el enfoque a un entorno de red simulado (por ejemplo, basado en Mininet o máquinas virtuales en red) donde el agente pueda actuar sobre reglas de defensa y observar consecuencias secuenciales de sus decisiones.
- Evaluar estrategias de aprendizaje por refuerzo profundo más avanzadas (PPO, A2C, etc.) y su integración con arquitecturas específicas para tráfico de red.