Impacto de la evolución temporal de datasets reales en el rendimiento de un IDS basados en anomalías: estudio experimental sobre HTTP

J. Diaz-Verdejo*, R. Estepa Alonso†, A. Estepa Alonso†, F.J. Muñoz-Calle†
* Dpt. Teoría de Señal, Telemática y Comunicaciones, CITIC, Univ. de Granada
C/ Periodista Daniel Saucedo Aranda, s/n, 18071 Granada (Spain)
E-mail: jedv@ugr.es
† Dpt. Ingeniería Telemática, Escuela Superior de Ingenieros, Univ. de Sevilla
C/ Camino de los Descubrimientos s/n, 41092 Sevilla (Spain)
E-mail: {rafa,aestepa}@trajano.us.es

***Resumen*—El desarrollo y evaluación de sistemas de detección de intrusiones basados en anomalías es de vital importancia en el contexto de la ciberseguridad, especialmente en relación a los ataques de día cero. La naturaleza altamente dinámica tanto de los sistemas a proteger como de los ataques hace que la detección de anomalías resulte una tarea compleja, ya que esta evolución temporal puede afectar a las capacidades de los modelos estimados en un escenario y periodo determinados. A pesar de su importancia, este efecto ha sido explorado de forma limitada en la literatura, especialmente por la práctica inexistencia de datos reales convenientemente etiquetados con la suficiente extensión temporal. En el presente trabajo evaluamos experimentalmente el impacto de la evolución temporal en un sistema para la detección de ataques basados en URL utilizando datos reales capturados en un escenario real durante un periodo de tiempo relativamente extenso. Nuestros análisis demuestran una degradación de creciente con la distancia temporal entre el entrenamiento y la evaluación. Esta degradación es debida a la combinación de la pérdida de calidad del modelo con el tiempo así como a la propia variación del comportamiento del servicio y/o ataques a lo largo del tiempo.**

***Index Terms*—sistemas de detección de intrusos, *data shift*, detección de anomalías, seguridad de servicios web**

Tipo de contribución: *Investigación original*

I. Introducción

Los sistemas de detección de intrusiones basados en anomalías (A-IDS), desempeñan un papel relevante en el campo de la ciberseguridad [1]. Este tipo de detectores permiten complementar a los detectores de intrusiones basados en reglas, facilitando la detección de ataques no conocidos [2]. Por ello existe abundante literatura sobre A-IDS (>290k publicaciones en Google Scholar), especialmente en los últimos años (17k desde 2020). Gran parte de los sistemas propuestos reportan resultados excelentes, con rendimientos próximos al 100 % según diversas métricas.

Sin embargo, pese a la abundancia de propuestas, no se observan despliegues reales de las mismas [3]. Por un lado, esto es debido a que los dataset empleados en las propuestas no son representativos de la realidad, utilizando datos de entrenamiento obsoletos, datasets reducidos en tamaño, datos sintéticos, etc. [4]. Por otra parte, los resultados presentados no son suficientemente relevantes y los sistemas no se comportan realmente como se predice fuera del laboratorio/escenario considerado. En este sentido, uno de los aspectos a tener en cuenta es la deriva temporal de los sistemas (*data-shift* o *concept drift* en la bibliografía) [5]. Entre los factores que motivan la deriva temporal podemos resaltar la evolución tecnológica y la variabilidad derivada de los usuarios y los propios servicios. Así, la propia naturaleza del servicio proporcionado puede originar modificaciones en los datos y/o comportamientos, lo que, evidentemente, puede afectar a la representatividad de los modelos entrenados. A modo de ejemplo, en una biblioteca universitaria, que es el caso de estudio considerado, se irán incorporando nuevos títulos y autores constantemente (variaciones en los datos). Posiblemente, se desplegarán nuevos servicios (p.e. préstamo on-line) y el patrón de peticiones será diferente en periodo de exámenes del correspondiente al periodo de vacaciones. Consecuentemente, un modelo de peticiones HTTP recibidas en este servicio establecido durante un periodo tiempo determinado puede no ser válido o suficientemente eficaz durante periodos posteriores alejados en el tiempo.

La gran mayoría de las propuestas existentes para el desarrollo de A-IDS obvian esta dependencia temporal de los sistemas subyacentes y, consecuentemente, de los modelos que se estimen, o, en el mejor caso, se limitan a mencionarlo y explicitan la necesidad de reentrenar los modelos a lo largo del tiempo según diversas estrategias [6].

Resulta difícil encontrar trabajos que evalúen el impacto de la evolución temporal en el rendimiento de los A-IDS, especialmente por la carencia de datasets reales etiquetados suficientemente representativos y con una duración temporal que permita el análisis. El motivo de la ausencia de este tipo de dataset es doble: los posibles problemas de privacidad y el excesivo consumo de tiempo que conllevaría el etiquetado del dataset completo. Por ello, la mayoría de las propuestas utilizan dataset con registros legítimos generados sintéticamente [7]. Sin embargo, es importante comprender que los dataset sintéticos no capturan la complejidad y diversidad de los sistemas actuales [8], fallando al emular el comportamiento a veces errático de los usuarios o las derivas temporales observables en entornos reales. Estas deficiencias se traducen en tasas de falsas alarmas significativamente inferiores a las registradas en casos reales.

ISBN:978-84-09-62140-8

Recientemente se ha puesto a disposición pública un dataset, denominado Biblio-US17 [9], con peticiones HTTP que reúne las características adecuadas para un estudio de este tipo. Este dataset corresponde a tráfico real capturado en un escenario en explotación durante un periodo relativamente extenso (véase la Sec. III-A para una descripción detallada).

En este trabajo nos proponemos evaluar la evolución del rendimiento de un modelo A-IDS entrenado durante un periodo de tiempo determinado a medida que aumenta la distancia temporal entre el periodo de entrenamiento y el de evaluación. Para ello consideraremos un esquema de particionado temporal propuesto en el dataset y un A-IDS basado en el modelado de Markov para la detección de ataques basados en URI [10]. Cuantificar la degradación de un A-IDS en un caso real debido a la deriva en el tiempo podría ser de utilidad para dirigir las investigaciones sobre cómo mitigar dicha degradación, por ejemplo, aplicando técnicas de reentrenamiento. Asimismo, mediante el análisis de los falsos positivos generados, nos planteamos la usabilidad de los modelos y su validez a lo largo del tiempo.

El presente trabajo se estructura como sigue. En la Sec. II se describe brevemente la motivación del presente trabajo y trabajos relacionados con la problemática abordada. En la Sec. III describiremos el escenario experimental utilizado para el estudio así como las métricas que se usarán en la evaluación. En la Sec. IV se presentan los resultados experimentales obtenidos. Finalmente, la Sec. V presenta las conclusiones del presente trabajo así como las líneas de trabajo futuras.

II. Antecedentes

El desarrollo y posterior despliegue de A-IDS requiere de una fase de entrenamiento/test/validación durante la que se ajustan/comprueban los parámetros del detector para conseguir maximizar el rendimiento. La naturaleza altamente dinámica de los servicios de red y, especialmente, de los servicios web, plantea problemas en lo que respecta a la estimación y ajuste de los modelos en relación a su deriva temporal [4] [11]. Lamentablemente, una proporción importante de los trabajos existentes obvia este problema y, simplemente, estima y evalúa los modelos usando todo el periodo de observación, que, por otra parte, suele ser reducido. En algunos casos, se identifica la aparición de la deriva en el tiempo y se aborda simplemente reorganizando los datos para suavizar el efecto (p.e. [12]). En otros casos, se considera el reentrenamiento del modelo de forma continua (*on-line retraininig*) o en bloques (*batches*). A modo de ejemplo, [13] aborda el problema mediante el uso de una ventana deslizante para el conjunto de representantes usados como modelo, mientras que [14] usa reentrenamiento diario sobre un dataset real. En cualquier caso, son muy pocos los trabajos que presentan resultados sobre la evolución temporal del rendimiento, limitándose a mostrar los resultados finales tras el periodo analizado. En este contexto resulta interesante el trabajo de [15], que compara el rendimiento con reentenamiento continuo frente a reentrenamiento por batches en un sistema basado en SVM. Los resultados en ambos casos muestran ligeros desplazamientos en la tasa de error con el tiempo.

Sin embargo, no hemos encontrado trabajos que muestren la evolución de las métricas de detección en función de la separación temporal entre los procesos de entrenamiento y evaluación. Probablemente esto se deba a la ausencia de bases de datos reales etiquetadas con la suficiente extensión temporal [16]. En este sentido, nos encontramos con algunas deficiencias en muchos de los trabajos en lo que respecta a los datasets utilizados, especialmente en relación a su naturaleza (real/sintética) y a la proporción normal/ataque [17]. A modo de ejemplo, [14] usa ataques sintéticos y no respeta el fuerte desbalanceo entre tráfico normal y de ataque. Por su parte, [15] usa una proporción irreal entre tráfico normal y de ataque (un factor 2). Entendemos que el estudio de la evolución temporal de las métricas de detección en función del tiempo resulta clave para la toma de decisiones como cuándo debe lanzarse el reentrenamiento del modelo aprendido.

III. Escenario experimental

El objetivo de este trabajo es evaluar la deriva temporal de los modelos en escenarios reales en base a un caso de estudio. Para ello se considerará un A-IDS sencillo basado en el modelado de Markov [10] aplicado a la detección de URIs de ataque sobre un dataset real [9] de gran tamaño. El rendimiento del A-IDS (en el punto de operación considerado óptimo) será evaluado con observaciones correspondientes a periodos posteriores al de entrenamiento. Para facilitar la interpretación de los resultados, se comparará el rendimiento encontrado con el que se obtendría sin tener en cuenta la deriva temporal. Por último se ofrecerán también los resultados cuando se utiliza el reentrenamiento para mitigar la degradación en el rendimiento provocada por la deriva temporal de los datos.

La elección del escenario está directamente relacionada con dos factores relevantes: una proporción significativa de los ataques actuales se basa en el uso de HTTP y, específicamente, de URIs maliciosas [17]; y, por otro lado, disponemos del mencionado dataset de URIs etiquetadas de gran tamaño adquirido en condiciones reales de tráfico [9].

III-A. El dataset Biblio-US17

Biblio-US17 es un dataset público[1] [9] con campos seleccionados de peticiones HTTP recopilado a partir de las trazas del servidor web de la biblioteca de la Universidad de Sevilla (España), entre el 1/1/2017 y el 17/7/2017. El servidor proporciona un servicio altamente dinámico (amplio uso de *queries* con abundantes parámetros) centrado en los recursos bibliográficos disponibles. Fue desarrollado en *Drupal CMS* (v7.96) y opera sobre *Apache web server* (v2.2), por lo que es representativo de las tecnologías actuales.

El dataset incorpora etiquetas que establecen una *ground truth*, identificándose peticiones legítimas y ataques basados en URI (Tabla I). Las peticiones han sido etiquetadas a partir de un proceso semiautomático [18], tomando como punto de partida la clasificación de las URI realizada por varios SIDS.

En la Tabla I se muestran las principales características del dataset utilizado y sus particiones de entrenamiento, test y validación. Hemos de destacar que el dataset ha sido adquirido a partir de la operación real del servidor, incorpora un gran volumen de datos y corresponde a un lapso de tiempo considerable. Adicionalmente, incorpora ataques reales que, obviamente, aparecen en la proporción real, lo que resulta muy relevante de cara a obtener tasas de detección y falsos

[1]Disponible en HTTPs://idus.us.es/handle/11441/148254

Tabla I
REGISTROS Y PARTICIONES EN *Biblio-US17*.

Clase	N registros	Entr.	Test	Val
Particionado TI (*Time Independent*)				
Raw	47 402 907	-	-	-
Legítimos	42 473 128	25 483 092	12 741 546	4 248 490
Ataques	327 906	-	245 689	82 217
Particionado TD (*Time Dependent*)				
P1	23 242 663	12 437 152	8 270 214	2 535 297
P2	24 227 443	13 638 751	7 000 209	3 588 483
P3	23 547 843	14 612 283	6 123 780	2 811 780
P4	25 011 619	13 849 043	6 400 263	4 762 313
P5	25 422 533	14 393 994	7 574 093	3 454 446
P6	24 285 502	13 400 472	8 216 759	2 668 271
P7	21 765 762	13 697 873	6 122 717	1 945 172
Particionado de ataques (esquema *Time Dependent*)				
A1	6 258	2 102	2 749	1 407
A2	7 436	2 697	2 806	1 933
A3	8 010	3 634	3 340	1 036
A4	47 735	4 513	2 969	40 253
A5	193 184	6 089	41 289	145 806
A6	246 754	5 775	186 059	54 920
A7	323 055	44 629	200 726	77 700

positivos representativas de la operación real del detector [4] [17].

El dataset se encuentra particionado de acuerdo a dos esquemas. El primer esquema (denominado *Time Independent* o TI) establece particiones que incluyen registros recolectados durante todo el periodo de adquisición, respetando la distribución temporal. Es decir, se han asignado los registros a las particiones de entrenamiento/test/validación en base a la proporción establecida (60/30/10) a partir de los bloques de registros diarios. Para los ataques no se establece en el dataset ningún particionado. Para poder realizar la experimentación adecuadamente, hemos aplicado el mismo esquema incorporando únicamente los ataques etiquetados como indubitados (LVL1, máximo nivel de certidumbre). Dado que no se usan para entrenamiento, se establece la proporción 3/1 para test y validación y se pondera su número en un factor 0.4 en la obtención de las métricas.

El dataset también incluye otro esquema de particionado denominado *Time Dependent* (Tabla I), que establece varios intervalos temporales consecutivos tanto para datos de entrenamiento como de test y validación. En particular, para cada partición se consideran 2 meses para el tráfico de entrenamiento, el mes siguiente para test y la quincena siguiente a esta para validación, siguiendo un esquema de ventana deslizante (Figura 1). Se establecen así 7 particiones. Análogamente al caso TI, el dataset no establece un particionado de ataques, por lo que, de nuevo, hemos tomado únicamente los ataques considerados indubitados (LVL1, máximo nivel de certidumbre) y hemos procedido a particionarlos siguiendo el mismo esquema que el usado para el tráfico legítimo (Tabla I).

III-B. Detección mediante un A-IDS basado en SSM

Aunque existen multitud de A-IDSs para tráfico HTTP [17], en este estudio se ha optado por el uso de un sistema relativamente simple que permite ilustrar el efecto de la deriva temporal en el rendimiento: el detector SSM [10], basado en un autómata de estados finitos, que resulta muy adecuado para modelar URIs y que ha mostrado resultados satisfactorios para la detección de anomalías en este escenario. Desde

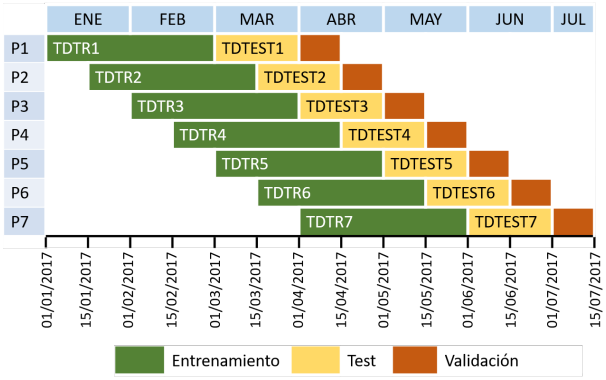


Figura 1. Esquema de ventana deslizante usado para el particionado TD en Biblio-US17.

el punto de vista matemático, SSM utiliza un modelo de Markov de primer orden para construir URIs conformes con la especificación correspondiente a partir de un conjunto de observaciones (palabras), cada una con una probabilidad de ocurrencia asociada.

El modelo, $\lambda = \{\Sigma, D, A, B, \pi\}$ se compone de:

- $\Sigma = \{S_i, 1 \leq i \leq N\}$: Un conjunto de N estados,
- $D = \{o_j, 1 \leq j \leq M\}$: Un conjunto de M simbolos observables,
- A: Una matriz NxN de probabilidades de transición entre estados,
- B: Una matriz NxM de probabilidades de observación de cada símbolo en cada estado, y
- π: Un vector de probabilidades de estado inicial.

En el caso de modelado de URIs, cada uno de los segmentos en un URI (inicial, path, atributo, valor, final) se asocia a un estado. Las observaciones son las palabras individuales que componen el URI y las probabilidades de transición se derivan de la definición del estándar. Por su parte, las probabilidades de observación corresponden a la probabilidad de cada palabra en cada uno de los estados, que deben ser estimadas durante el entrenamiento, mientras que las probabilidades de estado inicial son nulas excepto para el estado inicial.

De esta forma, dado un URI, U_k, y un modelo entrenado, λ, el detector SSM evalúa un índice a anomalía, $AS(U_k)$, a partir de la probabilidad de que el URI haya sido generado por el modelo, esto es,

$$AS(U_k) = -log\left(P(U_k|\lambda)\right) \tag{1}$$

A partir del índice de anomalía, un URI se clasificará como normal o anómalo (ataque) utilizando un umbral de detección, Θ, según

$$class(U_k) = \begin{cases} normal & si\ AS(U_k) \leq \Theta \\ anómalo & si\ AS(U_k) > \Theta \end{cases} \tag{2}$$

Una vez entrenado el modelo, se puede ajustar el punto de operación del detector sin más que modificar el umbral de detección Θ. Este ajuste busca habitualmente maximizar el rendimiento del sistema sujeto a ciertas restricciones, motivo por el cual se describirán a continuación las principales métricas aplicables.

III-C. Métricas para la evaluación del rendimiento y selección del punto de operación

La detección de ataques usando un A-IDS es un caso particular de detección binaria, por lo que se pueden presentar los siguientes 4 casos [19]:

- TP, *true positives*: Ataques correctamente identificados
- FN, *false negatives*: Ataques que no son identificados como tales
- TN, *true negatives*: Observaciones legítimas que no son consideradas ataques, y
- FP, *false positives*: Observaciones legítimas que son identificadas como ataques.

Asi pues, las métricas más habituales en la literatura [20] se definen a partir de proporciones entre TP, TN, FP y FN. Una de las métricas más empleadas para la evaluación del rendimiento de un detector es la curva ROC, que representa la tasa de TP frente a la tasa de FP obtenidas para todos los posibles valores del umbral de detección. Aunque el área bajo la curva ROC es habitualmente utilizada para evaluar el rendimiento general de un detector (un detector ideal tendía un área de 1), las curvas ROC permiten evaluar visualmente la dependencia entre las tasas de TP y de FP, facilitando la elección de un punto de operación teniendo en cuenta ambos factores.

Otra métrica de rendimiento global de un detector ampliamente utilizada en la literatura que combina la tasa de TP y FP es *F-score*, definida como

$$F - score = \frac{2TP}{2TP + FP + FN} \tag{3}$$

La métrica de rendimiento *F-score* resulta especialmente adecuada para dataset desbalanceados, lo cual es una característica intrínseca de la inmensa mayoría de los dataset reales, donde los ataques suelen representar un pequeño porcentaje del tráfico total analizado (habitualmente inferior al 0.1 %). Sería posible, por tanto, ajustar el punto de operación de nuestro detector para maximizar el rendimiento del sistema utilizando esta métrica.

Por otra parte, cabe señalar que el ajuste del punto de operación del detector no debería intentar maximizar una única métrica de rendimiento (p.e. *F-score*) sin tener en cuenta el número de FP generados. El motivo radica en que las alarmas generadas por los detectores deben siempre ser supervisadas por el SOC, lo cual constituye un coste importante. No resulta extraño, pues, que los detectores comúnmente utilizados en entornos de producción trabajen en punto de operación con tasas de FP mínimas (en el anexo V se puede encontrar resultados experimentales que muestran tasas de FP por debajo del 0,1 %). En este trabajo, por lo tanto, vamos a ajustar el punto de operación del AIDS tal que maximice el *F-score* sujeto a que la tasa de FP esté por debajo de un cierto umbral considerado admisible en escenarios reales. Con fines comparativos, también realizaremos un ajuste del punto de operación que simplemente maximize el *F-score* sin tener en cuenta la tasa de FP.

En el estudio experimental realizado utilizaremos tanto la curva ROC como el *F-score* para la evaluación del rendimiento.

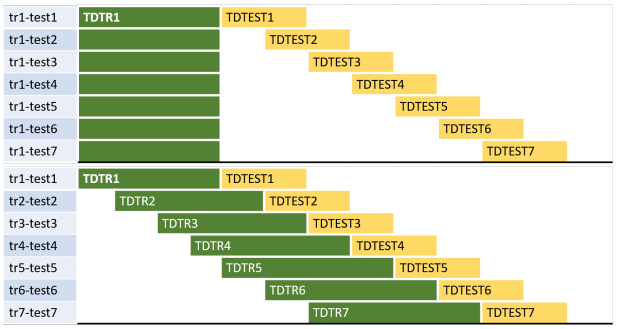


Figura 2. Particiones usadas en los experimentos TD: SR *tr1-testn* (arriba) y CR *trn-testn* (abajo).

III-D. Metodología y baterías de experimentos

Los dos esquemas de particionado definidos en BiblioUS17 posibilitan de forma natural varias configuraciones para el estudio de la evolución temporal del rendimiento. Obviamos, en primera aproximación, las particiones de validación, ya que nuestro objetivo es observar el posible decaimiento del rendimiento, para lo cual utilizaremos las particiones de entrenamiento para entrenar el modelo, y las de test para ajustar el punto de operación que maximice el rendimiento en dicha partición (con y sin tener en cuenta requisitos sobre la tasa máxima admisible de FP).

En primer lugar se realizará una evaluación del rendimiento del dataset completo, sin considerar su evolución temporal, utilizando para ello el particionado independiente del tiempo (TI) tal y como se recoge en la Tabla I. Este resultado nos servirá de referencia con la que comparar los resultados cuando se considere la deriva temporal del dataset.

A continuación se utilizará el esquema de particionado dependiente del tiempo (TD) para evaluar la evolución del servicio y la adecuación de los modelos a lo largo del tiempo. Para facilitar el análisis, consideraremos dos tipos de escenarios. En el primero, que denominaremos sin reentrenamiento (SR), analizaremos la evolución temporal a partir del modelo entrenado para una partición, usando todas las particiones posteriores de test para el ajuste del punto de operación y la evaluación del rendimiento. Denominaremos tr(i)-test(n) al experimento en el que se usa la partición TDTR(i) para entrenar y las particiones TDTEST(n) y ATTEST(n) (n-ésima partición de test de ataques) para el ajuste del punto de operación y evaluación del rendimiento. Así pues, los experimentos realizados para este escenario sin reentrenamiento puede expresarse como:

$$TD(SR) = \{tr(i) - test(n) / 1 \leq i \leq 7,\ i \leq n \leq 7\} \tag{4}$$

Nuestra hipótesis es que el rendimiento en este escenario irá decayendo a medida que aumente la separación temporal entre las particiones usadas para entrenamiento y test.

En el segundo escenario considerado, denominado con reentrenamiento (CR), se irá actualizando el modelo a lo largo del tiempo y se ajustará el punto de operación y evaluará el rendimiento con la partición de test correspondiente a la de entrenamiento:

$$TD(CR) = \{tr(i) - test(i) / 1 \leq i \leq 7\} \tag{5}$$

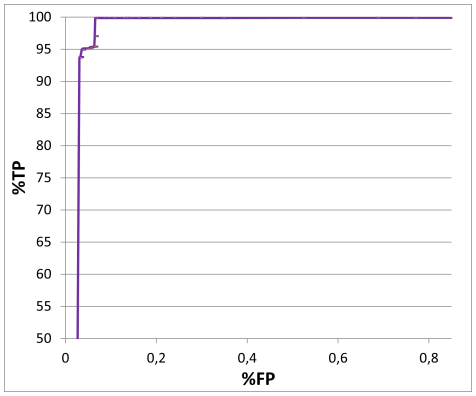


Figura 3. Curva ROC para el dataset completo -particionado TI-

De esta forma, los dos grupos de experimentos más significativos de cada escenario (plasmados en la Figura 2) serán los compuestos por *tr1-test(n)* con $1 \leq n \leq 7$ (entrenado con el primer bloque y evaluado con todos), y *tr(i)-test(i)*, con $1 \leq i \leq 7$. El primer grupo permite evaluar la evolución en las prestaciones de un modelo con el tiempo en el caso de la mayor extensión temporal permitida por el particionado. El segundo correspondería al caso de entrenar el sistema y evaluarlo a continuación.

IV. Resultados experimentales

IV-A. Resultados para el particionado independiente del tiempo

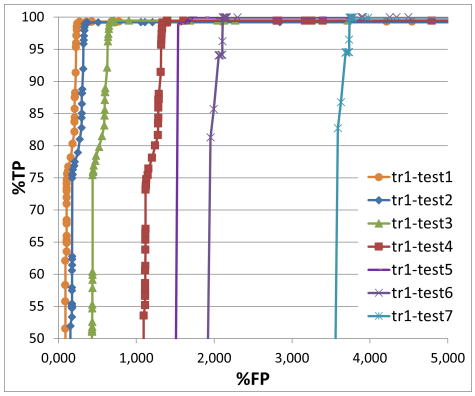
Para esta sección utilizaremos el particionado TI, donde el 60 % de los registros totales se dedican al entrenamiento y el 30 % al test. En la Figura 3 se puede apreciar la curva ROC que se obtiene para los diversos valores del punto de operación. Como se puede apreciar, el rendimiento del detector con el dataset completo resulta muy bueno, con un área bajo la ROC cercana a la unidad. El punto óptimo de funcionamiento del A-IDS alcanza un rendimiento *F-score* de 95,86 % con un porcentaje de FP=0,07 %. En el caso de buscar una restricción aun más severa en la tasa de FP (por ejemplo: FP<0.01 %), el punto de operación elegido conseguiría un *F-score* de 50,7 con una tasa de FP=0 %.

En la gran mayoría de los artículos que podemos encontrar en la literatura científica se ajusta el punto de operación y se evalúa el rendimiento de esta forma. A continuación veremos cómo evolucionan estos valores cuando consideramos la dependencia con el tiempo.

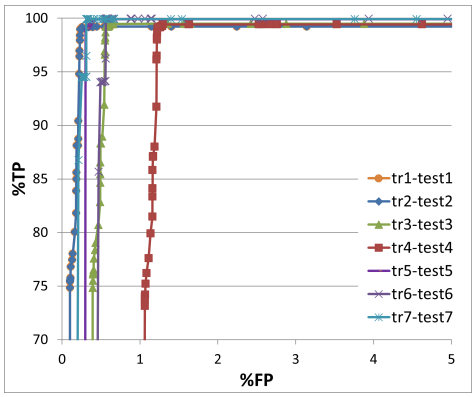
IV-B. Resultados dependientes del tiempo y efecto del reentrenamiento

En una primera inspección consideraremos las curvas ROC para analizar el comportamiento y rendimiento general del A-IDS. Posteriormente ofreceremos los resultados en base al rendimiento obtenido por la métrica *F-score* con y sin restricción en la tasa de FP.

IV-B1. Análisis basado en la ROC: La Figura 4 refleja las ROC obtenidas para los experimentos del escenario sin y con reentrenamiento. Para el primer caso, TD(SR), se puede observar claramente la degradación de las prestaciones del modelo a medida que aumenta la distancia temporal con la partición



a)



b)

Figura 4. Evolución temporal de la curva ROC: a) TD(SR) entrenando con partición 1, b) TD(CR).

de entrenamiento. En particular, si consideramos el modelo *tr1* —Figura 4.a)— se constata que la ROC va desplazándose hacia la derecha, lo que implica peor comportamiento.

Por otra parte, si analizamos la configuración TD(CR) —Figura 4.b)—, observamos un mejor rendimiento para cada conjunto de test cuando se entrena con su correspondiente partición de entrenamiento, si bien se aprecia una oscilación temporal en la partición 4. Esto confirma que el reentrenamiento permite cierta capacidad de adaptación a la deriva temporal, aunque no garantiza que las prestaciones del sistema cambien con la misma.

Podemos explicar estos resultados como la combinación de dos efectos: el de la pérdida de calidad del modelo a medida que se evalúa con mayor distancia temporal, tal y como se observa en el escenario SR, y un segundo efecto asociado a un cambio significativo en el comportamiento del servicio y/o de los ataques alrededor de la partición 4, observable en el caso de CR.

IV-C. Análisis basado en la métrica F-score

El análisis de las curvas ROC permite observar las posibilidades y el rendimiento en los distintos puntos de operación de un A-IDS, lo cual puede ilustrar bien la degradación del sistema. Sin embargo, durante la fase de operación, el A-IDS funciona en un punto de operación fijo, que suele ser establecido en base a alguna métrica. En este trabajo utilizaremos la métrica *F-score*, dada la naturaleza fuertemente

desbalanceada de las clases. A continuación analizamos el comportamiento del valor de *F-score*, mostrando en gráficas los resultados para los diferentes conjuntos de experimentos. A fin de facilitar la comparación con las curvas ROC de la sección anterior, en las gráficas de esta sección se puede observar en línea de barras el valor de *F-score* obtenido con el experimento sin reentrenamiento –SR–; esto es, entrenando con la primera partición y ajustando el punto de operación para maximizar el rendimiento con las demás particiones (*tr1-test(n)* con $1 \leq n \leq 7$) . En las mismas gráficas, como línea de color, también se muestra el valor de *F-score* obtenido en los experimentos con reentramiento –CR– (*tr(n)-test(n)* con $1 \leq n \leq 7$). Esto nos permitirá valorar el efecto del reentramiento en el rendimiento obtenido.

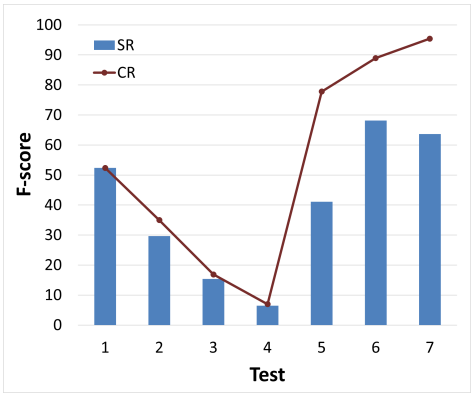
Dado que los sistemas utilizados en la realidad deben operan con una tasa de FP muy contenida, consideraremos dos posibilidades para el ajuste del punto de operación: a) utilizar aquél que maximiza el *F-score* sin restricción de FP, y b) utilizar aquél que maximiza el *F-score* sujeto a que la tasa de FP esté por debajo de un umbral máximo.

IV-C1. Ajuste sin restricción en FP: En la Figura 5 se muestran los resultados obtenidos para los dos escenarios, con y sin reentrenamiento, cuando no se considera ninguna restricción en la tasa de FP. Se puede observar que los resultados son relativamente pobres —Figura 5.a)—, obteniéndose para el caso sin reentrenamiento valores *F-score* que van desde el 50 % hasta el 10 %, para remontar hasta un máximo cercano al 70 % cuando nos alejamos hasta la sexta partición. Para el caso con reentrenamiento (CR), podemos observar un comportamiento muy similar en las 4 primeras particiones, recuperando algo más de *F-score* en las tres últimas particiones. Tan sólo la última partición CR alcanza un *F-score* cercano al 95 % similar al 95,86 % obtenido en el caso del particionado independiente del tiempo.

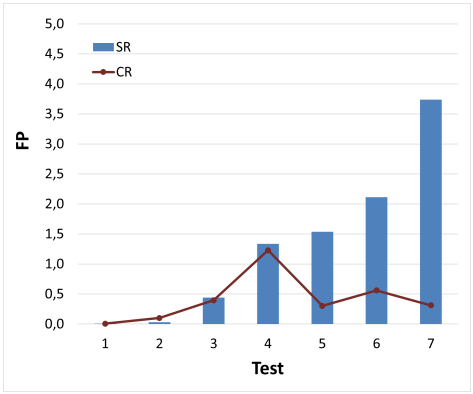
Por otra parte, es significativa la degradación hasta la partición 4 y el incremento posterior. Sin embargo, si analizamos la evolución de la tasa de FP —Figura 5.b)— vemos que estos aumentos se producen a costa de incrementar significativamente la tasa de FP hasta niveles muy superiores al 1 % (se alcanza casi el 3,5 %), lo que resulta inadmisible en sistemas reales. Tan sólo las tres últimas particiones con reentrenamiento consiguen mantener niveles de FP inferiores al 1 % (en torno a 0,5 %). Este comportamiento es claramente inadecuado en sistemas que deben operar en entornos reales.

En cualquier caso, vemos que el rendimiento resulta pobre incluso para la mayoría de los experimentos CR —p.e. TR4 evaluado con la partición de test 4 proporciona un F-score inferior al 10 %—, lo que nos puede llevar a pensar en un mal modelado.

A fin de tener un marco de referencia adicional para interpretar estos resultados, se ha evaluado el rendimiendo de varios IDS basados en firmas bien conocidos sobre las distintas particiones. El detalle de este análisis se muestra en el Apéndice A1. En los resultados obtenidos encontramos que el rendimiento de los IDS es también reducido, a excepción de *Nemesida* para las primeras particiones, donde alcanza un 97 %, aunque se deteriora para las últimas hasta el 7 %. Estos resultados nos muestran la gran variabilidad del rendimiento frente al tiempo en los datasets reales. En el caso del dataset



a)



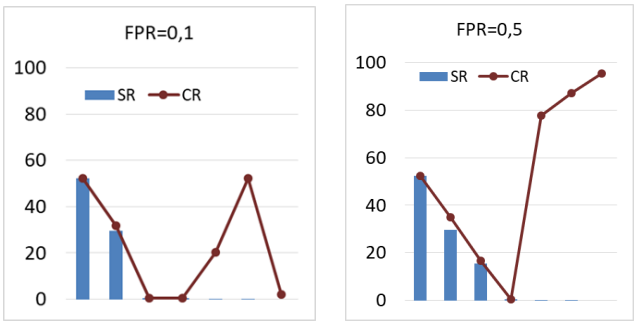
b)

Figura 5. Selección del punto de operación mediante F-score: a) Evolución de F-score, b) Evolución de FPR.

que nos ocupa, hemos podido verificar que las tres últimas particiones cuentan con miles de instancias de ataque de fuerza bruta con *percent encoding* que permanece inadvertido para los IDS basados en firmas, pero es identificado por nuestro A-IDS, lo que hace que mejore el rendimiento en dichas particiones, en especial cuando se utiliza el reentrenamiento.

IV-C2. Ajuste con restricción de FP: Como ya hemos mencionado, una tasa relativamente alta de FP como las obtenidas en el apartado anterior puede implicar una importante desproporción entre el número de detecciones correctas (TP) y el de FP que lleven al sistema a ser inutilizable en la práctica. Especialmente si, como es el caso, la proporción de ataques es muy baja. La cuestión es ¿qué tasa de FP máxima resulta apropiada en un sistema en explotación?. A fin de determinar dicho umbral, hemos obtenido las tasas de TP y de FP de tres IDS basados en firmas comúnmente utilizados en entornos de producción: Snort, Modsecurity y Nemesida. En todos los casos hemos encontrado que las tasas de FP están muy por debajo del 0,1 % (en concreto están sobre el 0,005 %). Si consideramos, por tanto, el ajuste del sistema al punto de operación que máximiza el *F-score*, pero que además debe cumplir con una tasa máxima de FP del, digamos de 0,1 % o de 0,5 %, se obtienen los resultados mostrados en la Figura 6.

En estas gráficas observamos que cuando el máximo valor de FP permitido es de 0,1 % (más laxo que el obtenido en los



a) b)

Figura 6. Selección del punto de operación mediante F-score con umbral de FP: a) FP≤0,1 %, b) FP≤0,5

IDS analizados en el Apéndice A1), el rendimiento siempre está por debajo de 50 %. Cuando no hay reentrenamiento, decae hasta el 0 % a partir de la tercera partición (esto es, no existe ningún punto de operación que cumpla la restricción del FP máximo permitido).

Si suavizamos la restricción permitiendo una tasa máxima de FP del 0,5 %, –Figura 6.b)–, observamos una leve mejoría en los resultados sin reentrenamiento y una mejoría notable en las tres últimas particiones el rendimiento se recupera cuando utilizamos reentrenamiento. Tal y como se obtuvo en el apartado anterior, los resultados sin reentrenamiento necesitan unas tasas de FP superiores al 1 % cuando la partición está suficientemente alejada. Esto resulta difícilmente aceptable en sistemas en explotación, lo que compromete la usabilidad del detector.

IV-D. Discusión de los resultados

De los resultados obtenidos podemos destacar en primer lugar la enorme diferencia entre los resultados de rendimiento obtenidos para el A-IDS con el dataset completo (TI), con un *F-score* de 95,86 % y un porcentaje de FP=0,07 %, frente al dataset considerando la deriva de tiempo (TD), cuyo *F-score* oscila entre el 10 y el 70 % para el caso sin reentrenamiento. Resulta muy llamativo cómo la tasa de FP va aumentando en función de la separación entre la partición de entrenamiento y la de test, alcanzando un pico del 4 %, lo que resulta inasumible en escenarios reales. Estos resultados confirman nuestra hipótesis de que el modelo entrenado va perdiendo validez a lo largo del tiempo debido a la deriva presente en los dataset reales. Si hubiéramos considerado una restricción en la tasa de FP máxima permitida (por ejemplo, un 0,5 %), el *F-score* se hubiera degradado desde el 50 % hasta el 0 % a partir de la tercera partición cuando consideramos la dependencia con el tiempo. Estos datos contrastan con el rendimiento cuando se considera el dataset completo (TI), donde se alcanza una tasa FP=0 % con un *F-score* de 50,7.

Podemos señalar como dato curioso del análisis realizado que las particiones donde peor comportamiento muestran los IDS basados en firmas (las tres últimas) son las que mejor funcionan para los IDS basados en anomalías, lo que confirma que ambos sistemas pueden funcionar de forma complementaria.

El uso de reentrenamiento, nos ofrece una mejora en los resultados sólo a partir de las 3 últimas particiones, alcanzando valores de *F-score* que oscilan entre el 75 y el 95 %. Para el resto de particiones los resultados son similares a los obtenidos sin reentrenamiento. Estos resultados confirman que la deriva temporal de los dataset reales provoca grandes oscilaciones en el rendimiento. Este efecto queda oculto cuando se utiliza el dataset completo sin tener en cuenta su distribución temporal.

V. Conclusiones

En este trabajo hemos realizado un estudio experimental de la degradación de las prestaciones de un A-IDS con el tiempo utilizando un dataset real. Nuestro estudio abarca tanto los casos en los que hay re-entrenamiento del modelo como los casos en los que no. Los datos han sido analizados en base a las curvas ROC obtenidas y también en base al punto de operación del detector (F-score y umbral de falsos positivos). Nuestros análisis demuestran consistentemente una degradación creciente con la distancia temporal entre el entrenamiento y la evaluación. Esta degradación es debida a la combinación de la pérdida de calidad del modelo con el tiempo así como a la propia variación del comportamiento del servicio y/o ataques a lo largo del tiempo.

En el futuro, pensamos evaluar el impacto de un re-entrenamiento continuo (online) mediante ventana deslizante con batch y re-estimación del modelo mediante interpolación. También deseamos evaluar un nuevos esquemas de particionado del dataset que ofrezcan mayores resoluciones temporales e incluso solapamiento entre particiones entrenamiento y test.

Agradecimientos

Esta publicación es parte del proyecto de I+D+i PID2020-115199RB-I00 financiado por MICIN/AEI/10.13039/501100011033.

Referencias

[1] I. Ghafir, V. Prenosil, J. Svoboda, M. Hammoudeh: "A Survey on Network Security Monitoring Systems", *Proc. 2016 IEEE 4th Int. Conf. on Future Internet of Things and Cloud Workshops (FiCloudW)* 77-82, 2016.

[2] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, E. Vázquez: " Anomaly-based Network Intrusion Detection: Techniques, Systems and Challenges", *Computers & Security*, 28, 18–28, 2009.

[3] M. Shashanka, M. Y. Shen, J. Wang: "User and Entity Behavior Analytics for Enterprise Security", *Proc. 2016 IEEE International Conference on Big Data (Big Data)*, 1867–1874, 2016.

[4] R. Sommer, V. Paxson: "Outside the Closed World: On using Machine Learning for Network Intrusion Detection", *Proc. 2010 IEEE symposium on security and privacy*, 305-–316, 2010.

[5] Moreno-Torres, J.G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N.V., Herrera, F.: "A unifying view on dataset shift in classification", *Pattern Recognit.*, 45 (1) 521-–530, 2012.

[6] Maggi, F., Robertson, W., Kruegel, C., Vigna, G.: "Protecting a moving target: addressing web application concept drift", *LNCS*, 5758, 21–40, 2009.

[7] Gao, Yang and Ma, Yan and Li, Dandan. "Anomaly detection of malicious users' behaviors for web applications based on web logs". 2017 IEEE 17th International Conference on Communication Technology (ICCT). 2017.

[8] Landauer, Max and Skopik, Florian and Höld, Georg and Wurzenberger, Markus. "A User and Entity Behavior Analytics Log Data Set for Anomaly Detection in Cloud Computing". 2022 IEEE International Conference on Big Data (Big Data). 2022

[9] J. Diaz-Verdejo J., R. Estepa Alonso, A. Estepa Alonso, F. Muñoz-Calle, G. Madinabeitia: "Labeled HTTP requests dataset: Dataset Biblio-US17", HTTPs://doi.org/10.12795/11441/148254, [Online; último acceso, 18/03/2024], 2023.

[10] J. M. Estévez-Tapiador, P. García-Teodoro, J. E. Díaz-Verdejo: "Detection of web-based attacks through Markovian protocol parsing", *Proceedings of 2005 IEEE Symposium on Computers and Communications*, 457-–462, 2005.
[11] N. Agarwal, S. Hussain: "A closer look at intrusion detection system for web applications". *Security and Communication Networks*, 2018.
[12] Y. Dong, Y. Zhang, Y., H. Ma, Q. Wu, Q. Liu, K. Wang, W. Wang: "An adaptive system for detecting malicious queries in web attacks´´. *Science China Information Sciences*, 61, 1-16,2018.
[13] M .Kirchner: "A framework for detecting anomalies in HTTP traffic using instance-based learning and k-nearest neighbor classification", en *2010 2nd International Workshop on Security and Communication Networks (IWSCN)*, 1-8, 2010.
[14] A. Kim, M. Park, D. Lee: "AI-IDS: Application of deep learning to real-time Web intrusion detection", *IEEE Access*, 8, 70245-70261, 2020.
[15] J. Ma, L.K. Saul, S. Savage, G.M. Voelker: "Learning to detect malicious urls". *ACM Transactions on Intelligent Systems and Technology'*, 2(3), 1-24, 2011.
[16] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, A. Hotho: "A survey of network-based intrusion detection data sets". *Computers & Security*, 86, 147-167, 2019.
[17] J. E. Díaz-Verdejo, R. Estepa Alonso, A. Estepa Alonso, G. Madinabeitia. 2023. A Critical Review of the Techniques used for Anomaly Detection of HTTP-based Attacks: Taxonomy, Limitations and Open Challenges. *Computers & Security*, 124 (2023) 102997.
[18] J.E. Díaz-Verdejo, A. Estepa, R. Estepa, G. Madinabeitia, F.J. Muñoz-Calle: "A methodology for conducting efficient sanitization of HTTP training datasets". *Future Generation Computer Systems*, 109, 67-82, 2020.
[19] G. Naidu, T. Zuva, E.M. Sibanda: "A Review of Evaluation Metrics in Machine Learning Algorithms". *Proc. Artificial Intelligence Application in Networks and Systems. CSOC 2023*. Lecture Notes in Networks and Systems, 724, 2023.
[20] M .Bekkar, H.K. Djemaa, T.A. Alitouche: "Evaluation Measures for Models Assessment over Imbalanced Data Sets.", *J Inf Eng Appl*, 3(10), 27–38, 2013.
[21] InspectorLog, available at HTTPs://gitlab.com/neus_cslab/inspectorlog [Online; accessed 24-Jan-2024] (2024)

A1. Rendimiento de IDS basados en firmas

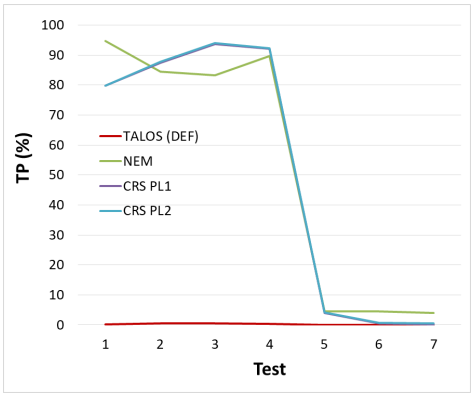
A fin de facilitar un contexto más amplio para la interpretación de los resultados, se ha evaluado el rendimiento de las distintas particiones usando varios detectores basados en firmas de amplio uso: *Snort*, *Nemesida* y *ModSecurity*. Para ello hemos utilizado la herramienta *Inspectorlog* [21], desarrollada por nuestro equipo, con los siguientes conjuntos de reglas (usados por separado):

- *Talos Community* (24/03/2022): Reglas utilizadas por Snort desarrolladas por el equipo Talos de Cisco.
- *Nemesida Community Edition*: reglas publicadas el 09/01/2022[2]
- *OWASP*: Se ha usado el conjunto OWASP CRS 3.2.0. En este caso se han considerado los niveles de paranoia (PL) 1 y 2.

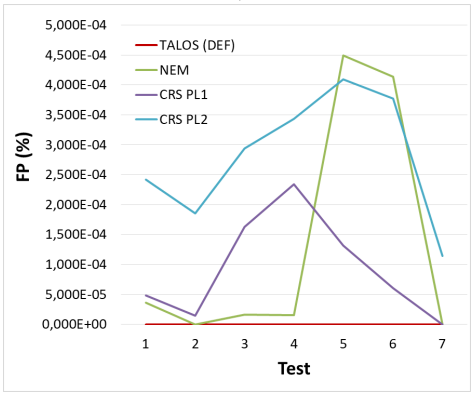
Los resultados se muestran en la Figura 7.

Llama la atención la baja tasa de detección que tiene Snort con las reglas por defecto. Tan sólo es capaz de reconocer un máximo de 20 ataques en cada una de las particiones, aunque la tasa de FP es 0. Los otros dos IDS tienen una tasa de detección muy superior en los ataques de las primeras 4 particiones, que mayoritariamente son de tipo *directory traversal* o ataques de codificación. A partir de la cuarta partición, la mayoría de las instancias de ataque se corresponden con un ataque de fuerza bruta con codificación que permanece indetectado por los sistemas. Este hecho hace que baje el ratio de ataques detectados. Los valores de F-score registrados para Snort no superan el 1 % en ninguna de las particiones, mientras que

[2]Disponibles en HTTPs://rlinfo.nemesida-security.com/.



a)



b)

Figura 7. Resultados de detección mediante SIDS: a) Detección (TP), b) Falsos positivos

oscilan entre el 7 y el 97 % para Nemesida (en función de la partición) y entre el 1 y el 88 % para ModSecurity. Estos resultados nos muestran la gran variabilidad en el rendimiento que se obtiene en los casos reales.

Resulta muy significativo, sin embargo, que las tasas de falsos positivos están por debajo del 0,00045 %. Este resultado confirma nuestra hipótesis de que los productos IDS utilizados habitualmente en escenarios reales tienen tasas de FP muy bajas a fin de evitar falsas alarmas que deban ser revisadas por el personal del SOC.