CAPÍTULO 4. MÉTODO DE COMBINACIÓN DE INFORMACIÓN DE COMPORTAMIENTOS

Hasta el momento se ha presentado un flujo de trabajo que permite incorporar técnicas de análisis de comportamientos dentro de los flujos de autenticación y autorización de los estándares de gestión de identidades federados. Este flujo de trabajo tiene como objetivo utilizar las técnicas de análisis de comportamientos para aumentar los niveles de seguridad proporcionados. Sin embargo, los modelos de análisis de comportamientos de la literatura poseen ciertas limitaciones.

En este capítulo se detalla el método propuesto para mejorar los modelos de análisis de comportamientos actuales. Este método se basa en combinar información de comportamientos a nivel de características. Tal y como se ha visto reflejado en el estado del arte (ver Capítulo 2.2.2), la combinación de información está posicionándose como una técnica novedosa que permite mejorar la eficacia de los mismos. Sin embargo, hoy en día, son pocos los trabajos que combinan la información de comportamiento, y muchos menos los que lo hacen a nivel de características.

El método propuesto se presenta como un conjunto de tareas novedosas que permiten combinar la información de comportamientos a nivel de características y mejorar, por tanto, la eficacia de los sistemas de autenticación basados en el análisis de comportamientos [130]. Ha sido específicamente diseñado para combinar información temporal, recogida de fuentes de información heterogéneas. Además, no asume que los datos temporales siguen una distribución específica. Esto es, no asume que los datos temporales se recogen con una frecuencia o patrón específico.

El método se presenta esquematizado en la Figura 4.1. Está compuesto de cuatro tareas principales. La primera tarea se basa en representar la información recopilada de fuentes heterogéneas de datos en el mismo espacio. Para ello, se transforma en secuencias de n-gramas. Esta tarea se describe en la Sección 4.1 y se basa en la implementación de una técnica de *Symbolic Aggregate approximation* (SAX) multivariante [131] novedosa utilizando *Random Trees Embedding*s (RTEs) [132], [133]. La siguiente tarea se basa en construir una matriz de distancias comparando los n-gramas extraídos anteriormente utilizando técnicas de alineamiento de secuencias de ADN. Esta tarea se detalla en la Sección 4.2. Posteriormente, se describe el proceso de entrenamiento de un algoritmo de *clustering* basado en densidades con el objetivo de poder extraer los núcleos de comportamiento que caracterizan a cada usuario utilizando la matriz de distancias. Este proceso se contempla

Figura 4.1. Método propuesto para combinar información de comportamientos

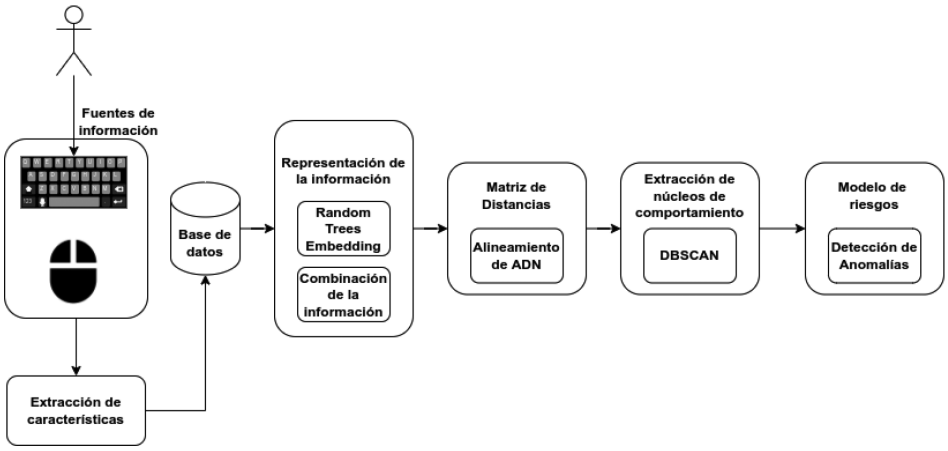
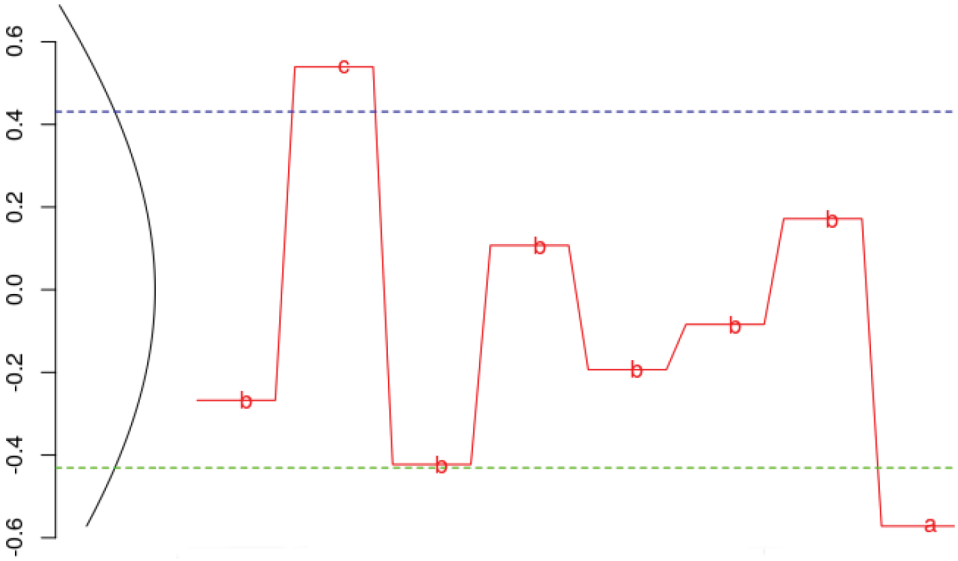


Figura 4.2. Discretización de una serie temporal usando SAX



en la Sección 4.3. A continuación, se detalla la implementación de un modelo de riesgos, el cual utiliza los núcleos de comportamiento para categorizar las muestras, en la Sección 4.4 . Finalmente, en la Sección 4.5 se detalla cómo se pueden fijar los diferentes parámetros e hiperparámetros que utiliza el modelo con el objetivo de ser reproducible.

4.1. Representación de la información

En primer lugar, se presenta una técnica para transformar datos temporales procedentes de fuentes heterogéneas de datos (p. ej. el teclado y el ratón) en una representación tabular adecuada, que sirve para poder alimentar los modelos tradicionales de aprendizaje máquina. Este enfoque se basa en implementar una técnica novedosa de SAX multivariante [131]. Específicamente, esta técnica se implementa haciendo uso de los RTEs [132], [133].

La técnica de SAX tradicional se define como una técnica para discretizar series temporales y reducir la dimensionalidad de las mismas. Para lograr esto, se hace uso de *Piecewise Aggregate Approximation* (PAA) para normalizar y dividir las series temporales en secciones con un tamaño equidistante. Posteriormente, se calculan los valores medios de cada sección y se les asigna un nivel de probabilidad de la función gaussiana. Este proceso permite discretizar la serie. Finalmente, a cada uno de estos niveles se les asigna un símbolo concreto, logrando así transformar una serie temporal en una secuencia de símbolos (ver Figura 4.2).

Tomando como punto de partida y objetivos los mismos que la técnica de SAX tradicional, se propone extender esta implementación para considerar series temporales multivariantes. Para ello se propone utilizar el algoritmo de RTE, ya que las técnicas basadas en árboles han demostrado ser más eficaces a la hora de representar y agrupar series temporales multivariantes [134].

El algoritmo de RTE se basa en generar una secuencia de árboles aleatorios de decisión, de tal manera que al clasificar una muestra se genera un vector con los valores de los nodos hojas que le han sido asignados. La longitud de esta secuencia de árboles viene definida por el parámetro *Número de árboles*, mientras que también se tiene que definir la profundidad de cada uno de estos árboles con el parámetro *Máxima profundidad*. Dicho de otra forma, los RTEs se basan en obtener un vector que indica, para cada árbol generado, el número de hoja a la que una muestra concreta pertenece. Esto es, la posición *n* del vector embedding resultante, indica el número de hoja (ordenando los árboles y, por consiguiente, las hojas de izquierda a derecha) donde la observación ha sido clasificada en el árbol de decisión *n*. Alternativamente, este vector embedding se suele representar de forma binaria, utilizando un vector cuya longitud es el número de hojas en el RTE y cuyo valor es 1 en caso de que la muestra haya caído en dicha hoja y 0 para el caso contrario.

Los embeddings obtenidos de los RTEs se mapean entonces a símbolos con el objetivo de habilitar la comparación entre ellos y poder procesarlos. El número de símbolos necesarios para poder mapear todos los embeddings crece de forma exponencial a medida que aumenta el *Número de árboles* y

el número de hojas (en función del parámetro *Máxima profundidad*) del RTE. A pesar de esto, la mayoría de los símbolos aparecerán en muy rara ocasión, es decir, si el número de árboles generados y hojas es extremadamente alto, la mayoría de las muestras caerán en hojas distintas y, por consiguiente, se les asignara un símbolo diferente. Esta limitación se soluciona asignando los símbolos a una tabla de frecuencias de aparición. De esta manera, a las muestras mayoritarias que compartan las mismas hojas se les asigna un símbolo representativo, mientras que las muestras minoritarias se les asigna un símbolo arbitrario (algo parecido en el SAX tradicional al utilizar la función gaussiana). Por ejemplo, el embedding con más ocurrencias se le asigna el carácter *A*, al siguiente el carácter *B*, y así sucesivamente. De este modo, si dos muestras caen en las mismas hojas, son asignadas con el mismo símbolo. Cuando todos los símbolos son utilizados, los embeddings que quedan y que agrupan a las muestras menos representadas en el conjunto de datos se les asigna un símbolo arbitrario.

El proceso explicado anteriormente se ejecuta para cada fuente de información disponible. Esto es, cada fuente de información posee su propio RTE y su único conjunto de símbolos.

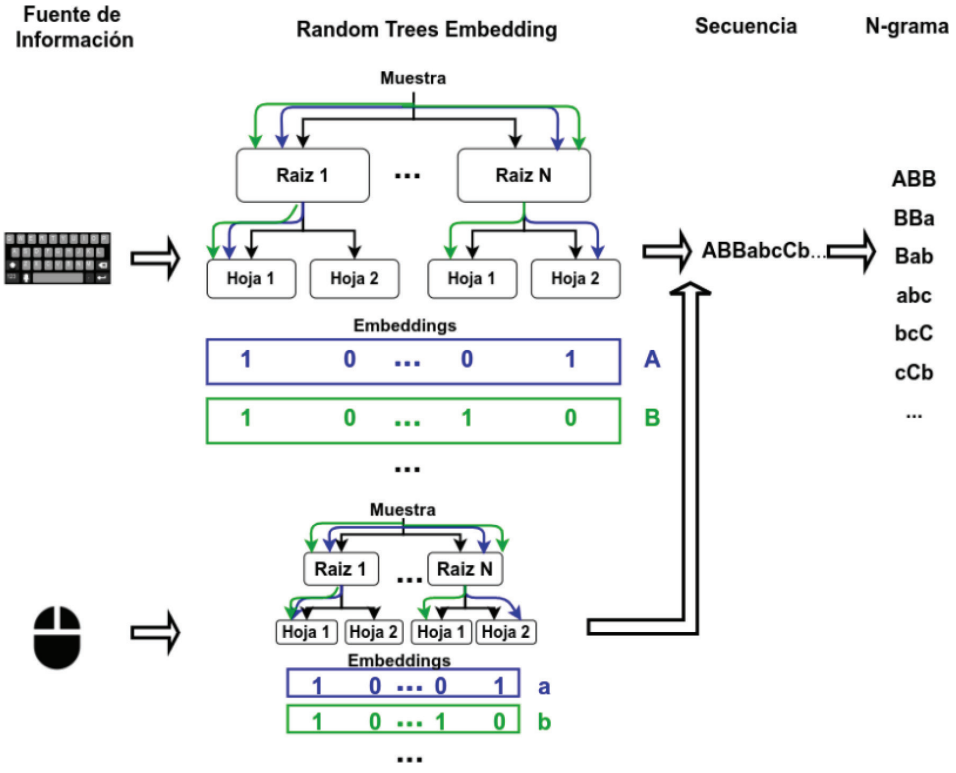
En este caso, la información extraída del teclado se representa utilizando símbolos de letras mayúsculas, mientras que para la información extraída del ratón se va a representar utilizando símbolos de letras minúsculas. Una vez que cada muestra de cada fuente de información se ha convertido a un símbolo, se utiliza la información temporal para poder ordenarlos. De este modo, estos símbolos se agrupan formando una secuencia ordenada que representa el comportamiento de un usuario específico para ambas fuentes de información. Finalmente, estas secuencias se dividen en n-gramas definidos por el parámetro *N-Gram Length (NGL)*. Estos n-gramas representan el comportamiento de un usuario durante un tiempo limitado de interacciones. Por ejemplo, el n-grama *AbB*, cuyo *NGL* viene definido por el valor 3, podría representar una pulsación lenta de teclado, seguido por un movimiento rápido de ratón y finalmente una pulsación rápida de teclado. Cuanto mayor sea el conjunto de símbolos, mayor precisión se obtendrá a la hora de representar el comportamiento de un usuario, sin embargo, si el número de símbolos es muy elevado, se obtiene un sobreajuste, pues cada símbolo representará una interacción muy específica y, por lo tanto, cada n-grama será diferente al resto de n-gramas haciendo así que la comparación entre ellos sea ineficaz. El proceso de combinación de la información de fuentes de datos heterogéneas se puede ver ilustrado en la Figura 4.3. Este proceso logra transformar la información de comportamientos recogida de fuentes de datos heterogéneas en el mismo espacio de representación (símbolos). De este modo, la representación de la información elegida permite combinar información a nivel de características.

4.2. Generación de la matriz de distancias

Una vez obtenida la representación de la información en n-gramas, la siguiente tarea es establecer un método para poder comparar estas secuencias entre sí. Existen multitud de métricas de distancias entre secuencias de símbolos en la literatura, siendo algunos ejemplos de las más conocidas, la distancia de Hamming, la distancia de Levenshtein, la distancia coseno y el coeficiente de Jaccard [135]. En el método propuesto, la distancia entre dos n-gramas se calcula utilizando técnicas de alineamiento de secuencias de ADN [136].

En el ámbito de la bioinformática, las secuencias de ADN se representan como secuencias de símbolos. Analizar la similitud o disimilitud entre las diferentes regiones del ADN permite detectar, entre otras cosas, multitud de enfermedades o variaciones genéticas que pueden ser de interés para su análisis. Las técnicas de alineamiento de secuencias de ADN se posicionan

Figura 4.3. Proceso de SAX multivariante utilizando RTEs para multiples fuentes de información



como el método más utilizado para extraer patrones y comparar dichas secuencias [137]. Es por esta razón, por la que se ha decidido incorporar este tipo de técnicas en el método propuesto.

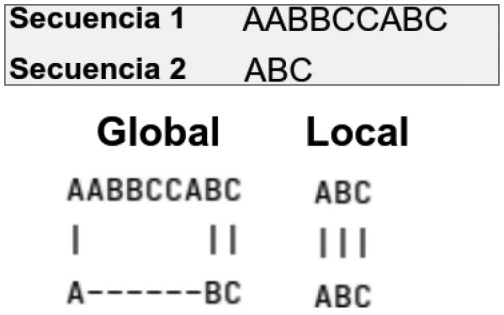
A nivel general, existen dos técnicas de alineamiento de ADN [138]: la global y la local (ver Figura 4.4).

El algoritmo de alineamiento global se basa en encontrar la mejor sincronía entre dos secuencias haciendo coincidir todos los caracteres de ambas secuencias. Es adecuado cuando las dos secuencias a comparar tienen una longitud igual o similar y se espera que compartan similitudes a lo largo de toda la secuencia. Por otro lado, el algoritmo de alineamiento local se basa en la obtención de la subcadena de la primera secuencia que más se acerca a otra subcadena de la segunda secuencia. Es adecuado para comparar secuencias de diferentes longitudes y secuencias menos similares o relacionadas.

Cualquiera de estos algoritmos devuelve dos valores. En primer lugar, el alineamiento como tal, es decir, la cadena o subcadena de mayor coincidencia entre dos secuencias. En el caso de la presente investigación, este alineamiento puede ser utilizado para tratar de explicar los diferentes patrones de comportamiento que caracterizan a cada uno de los usuarios. El segundo resultado, es el valor de similitud entre ambas secuencias. Para obtener este resultado, normalmente se debe fijar el valor por el que se va a ponderar obtener una coincidencia, obtener un error y la penalización por extender el error a lo largo de una ventana. Cabe destacar que, la ponderación de coincidencia debe ser positiva para aumentar la puntuación de similitud final, mientras que las puntuaciones de error y ventana de error deben ser negativas para disminuir la similitud final entre las secuencias.

Atendiendo a lo expuesto con anterioridad, en el método propuesto se utiliza la técnica de alineamiento global [139]. De este modo, todos los n-gramas

Figura 4.4. Alineamiento global y local de secuencias de ADN



obtenidos para cada usuario se comparan en pares entre sí. El alineamiento devuelve el valor de similitud entre ambos n- gramas, considerando la longitud total de la secuencia objetivo, y los valores fijados para los parámetros de coincidencia, error y ventana de error. Posteriormente, este valor de similitud se transforma a un valor de distancia. Para ello, en primer lugar, se normaliza en el rango [0,1], siendo 1 el valor máximo de similitud (es decir, un n-grama consigo mismo) y 0 el mínimo. A continuación, para obtener el valor de distancias se aplica uno menos el valor de similitud normalizado. De esta forma, el valor obtenido está también en el rango [0,1], siendo 0 el valor de mínima distancia (un n-grama consigo mismo) y 1 el valor de máxima distancia entre n- gramas. Cabe destacar que, no pueden existir valores de distancia negativos. Estos valores de distancias se apilan en una una matriz simétrica de tamaño NxN, donde N es el número de n-gramas para un usuario específico.

El resultado de este proceso es, por tanto, una matriz de distancias que representa la disimilitud entre los comportamientos generados por un mismo usuario. Por tanto, cada usuario obtendrá una matriz de distancias específica e independiente.

4.3. Extracción de los núcleos de comportamiento

La siguiente tarea se corresponde con utilizar la matriz de distancias, obtenida para cada usuario, para calcular los núcleos de comportamiento de los mismos. Identificar y definir correctamente estos núcleos de comportamiento es una de las tareas más críticas para poder generar un modelo de análisis de comportamiento preciso, pues es la fuente de conocimiento que se utilizará para comparar las nuevas muestras y, por lo tanto, la base que utiliza el modelo para detectar comportamientos anómalos que puedan suponer una brecha de seguridad. La mayoría de los sistemas de control de accesos del estado del arte utilizan toda la información disponible de los comportamientos para cada usuario. Sin embargo, en el propio comportamiento de los usuarios pueden existir valores atípicos que pueden afectar negativamente al rendimiento y precisión del clasificador. Además, utilizar todos los datos en lugar de un subconjunto de los mismos puede generar latencias añadidas, tanto a la hora de entrenar el clasificador, como a la hora de predecir y evaluar nuevas muestras. Esto se debe a que cada secuencia (n-grama) se debe comparar contra un número mayor de información. De este modo, lo ideal sería obtener núcleos de comportamiento tan pequeños como sea posible, siempre y cuando estos núcleos representen correctamente el comportamiento del usuario de tal manera que el clasificador pueda generalizar correctamente.

En la presente propuesta, se utiliza el algoritmo de *clustering* basado en densidades llamado DBSCAN [123] para obtener los núcleos de comportamiento. La idea bajo este algoritmo es dividir el espacio de decisión en áreas

de densidad. Para entrenar el DBSCAN, hay que fijar dos hiperparámetros. En primer lugar, la distancia máxima entre dos muestras que se consideran vecinas. Este hiperparámetro se denomina *EPS* y es el parámetro más crítico. En segundo lugar, el número mínimo de puntos que debe tener una vecindad para ser considerada como un *cluster*.

Este hiperparámetro se denomina *MINS*. Todos los vecinos dentro del radio *EPS* de un punto central se consideran parte del mismo *cluster*. Si alguno de estos vecinos es de nuevo un punto central, sus vecinos se incluyen de nuevo para su análisis. Los puntos no centrales de este conjunto se denominan puntos fronterizos. Los puntos que no son alcanzables desde ningún punto central se consideran ruido y no pertenecen a ningún cluster. DBSCAN no necesita información sobre el número de *clusters* deseados a priori para su entrenamiento. Esto es coherente con el hecho de que, es imposible saber cuántos núcleos de comportamiento (*clusters*) se obtendrán para un usuario específico.

El funcionamiento de DBSCAN se puede ver ilustrado en la Figura 4.5. En este ejemplo, el parámetro MIN es 4 y el radio EPS está representado por los círculos. A es un punto central. Los puntos B y C son puntos fronterizos. Las flechas indican la densidad. Los puntos B y C están conectados pues son alcanzables desde A tanto de forma directa, como por medio de otro punto central. R es un punto de ruido puesto que no es alcanzable desde A.

De este modo, la matriz de distancias (que solo contiene comportamiento generado por el propio usuario) alimenta el algoritmo DBSCAN. Así, las regiones de alta densidad obtenidas representan los núcleos de comportamiento, mientras que las regiones de baja densidad representan comportamientos atípicos del usuario. Estos comportamientos atípicos se descartan para que las muestras que se van a evaluar no puedan ser comparadas con ellos.

Figura 4.5. Ejemplo del funcionamiento del algoritmo de DBSCAN

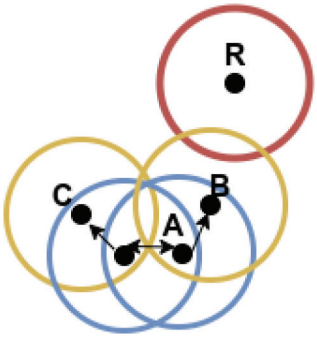
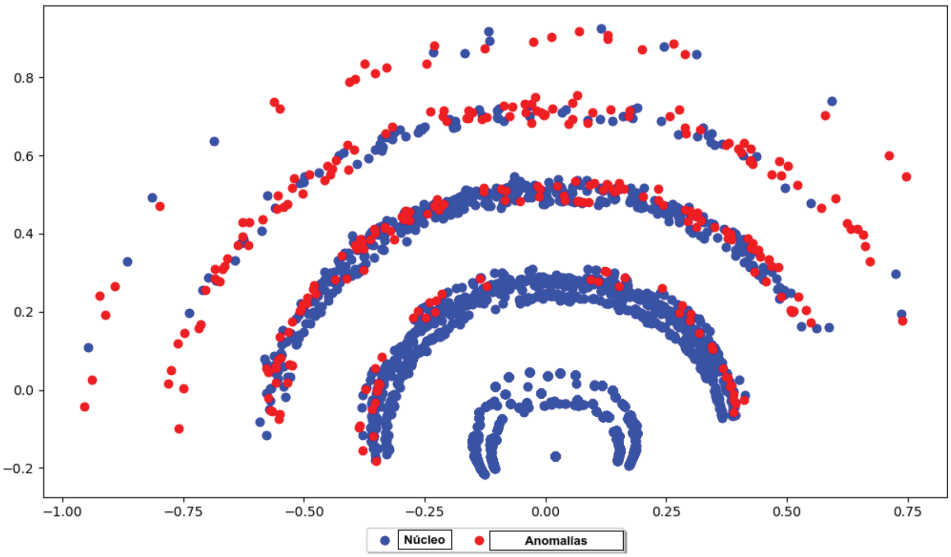


Figura 4.6 Representación de los núcleos de comportamiento de un usuario específico utilizando escalado multidimensional



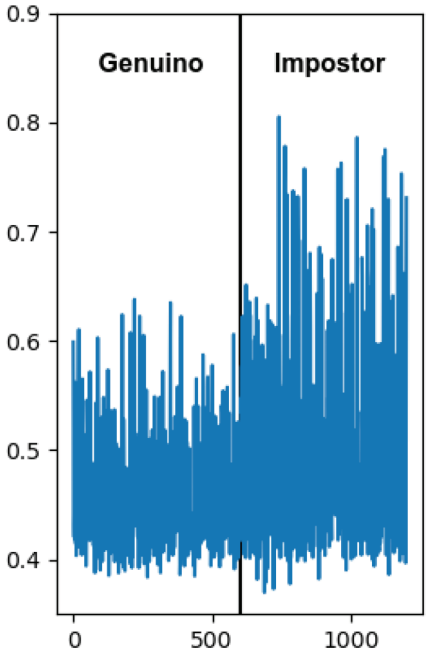
En la Figura 4.6, se puede observar un ejemplo, para un usuario específico, de los núcleos extraídos (puntos azules) y de los comportamientos atípicos (puntos rojos). Estos puntos se han representado utilizando las dos primeras componentes principales calculadas utilizando el algoritmo de escalado multidimensional [140] (solo la información más representativa está siendo visualizada en la figura). Como se puede observar, a medida que los puntos se alejan del núcleo principal (mostrado en la parte inferior de la figura), la distribución se va convirtiendo en más heterogénea, es decir, contiene un mayor número de comportamientos atípicos. Esto corrobora que DBSCAN está descartando correctamente los comportamientos que se desvían de la distribución de comportamientos esperada para este usuario concreto.

4.4. Modelo de riesgos

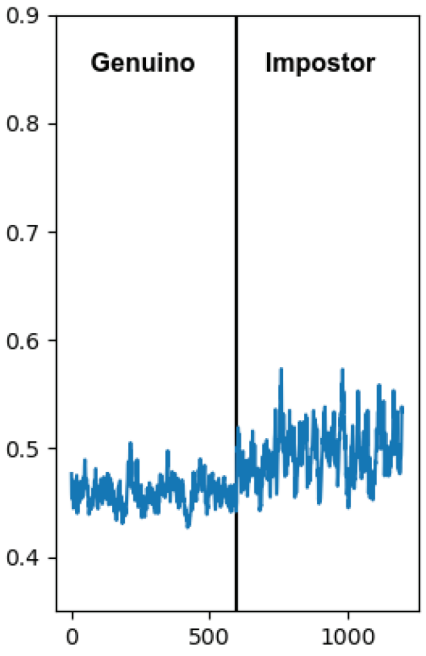
En esta sección, se va a explicar detalladamente la elaboración de un modelo de riesgos. Este modelo tiene el objetivo de clasificar las muestras en función de su similitud con los núcleos de comportamiento obtenidos previamente. Esto permite categorizar estas nuevas muestras en genuinas, o por el contrario detectar si son anomalías y por lo tanto pertenecen a un impostor.

En primer lugar, toda nueva muestra se procesa siguiendo los pasos explicados en las secciones anteriores. De este modo, se obtienen los n-gramas

Figura 4.7. Valores del *buffer* de riesgo para un usuario concreto



(a) Valores de riesgo en bruto.



(b) Valores de riesgo aplicando MME.

que representan el comportamiento de dichas muestras. Estos n-gramas se comparan utilizando las técnicas de alineamiento de ADN con todos los n-gramas de los núcleos de comportamiento. El resultado de esta operación es un vector de distancias para cada nueva muestra, cuya longitud es el número de puntos de los núcleos de comportamiento. Este vector, por lo tanto, determina la distancia entre esta nueva muestra y el conocimiento extraído previamente para un usuario concreto.

La premisa de partida es que un mismo usuario va a generar, por norma general, vectores de distancias con valores pequeños (semejanzas altas), mientras que las dinámicas de comportamiento generadas por un usuario impostor resultarán en vectores de distancias con valores altos (poco semejantes con los núcleos de comportamiento).

El riesgo asociado a esta nueva dinámica de comportamiento se puede calcular como la media de los valores del vector de distancias. De este modo, un valor bajo de riesgo determina que la nueva dinámica (en forma de n-grama) se encuentra cerca de los núcleos de comportamiento y, por lo tanto, es altamente probable que pertenezca al mismo usuario objetivo. Por el contrario,

un valor alto de riesgo representa que esta nueva dinámica se encuentra lejos de los núcleos de comportamiento y por lo tanto es probable que no pertenezca al usuario objetivo, sino que sea de un usuario impostor. Este proceso permite comparar cada nueva dinámica de comportamiento de forma individual. Sin embargo, en un entorno real se generan multitud de dinámicas de comportamiento y además lo hacen de forma secuencial. De este modo, estos valores de riesgo se ordenan a lo largo del tiempo, generando así, un *buffer* de riesgo (ver Figura 4.7a). Tal y como se puede observar, los valores de riesgo obtenidos son muy dispares, es decir, son muy cambiantes a lo largo del tiempo. Esto se debe a que cada dinámica de comportamiento, de forma independiente, puede ser muy parecida o distinta a los núcleos de comportamiento calculados. Este resultado se traduce en que el sistema final obtiene multitud de falsos positivos y falsos negativos.

Tomando como base la premisa de partida, se procede a suavizar el *buffer* de riesgo obtenido. Para ello se aplica la Media Móvil Exponencial (MME), con el objetivo de reducir los altos cambios que se producen en los valores de riesgos, de la siguiente manera:

$$MME_t = \begin{cases} Y_1, & \text{si } t = 1 \\ \alpha Y_1 + (1-\alpha) \cdot MME_{t-1}, & \text{si } t > 1 \end{cases}$$

donde MME_t es la media móvil exponencial en el instante t, Y_t es el valor de riesgo en un instante t y α es el coeficiente de suavizado en el rango [0,1]. Un valor de α pequeño pondera más alto las observaciones más antiguas, mientras que un valor alto repercute en un modelo más olvidadizo que pondera más las observaciones últimas y cercanas al instante t.

El coeficiente α se suele calcular en función del número de observaciones pasadas a tener en cuenta [141]. En este caso, se calcula de acorde al parámetro *WinSize* como sigue:

$$\alpha = 2/(WinSize + 1)$$

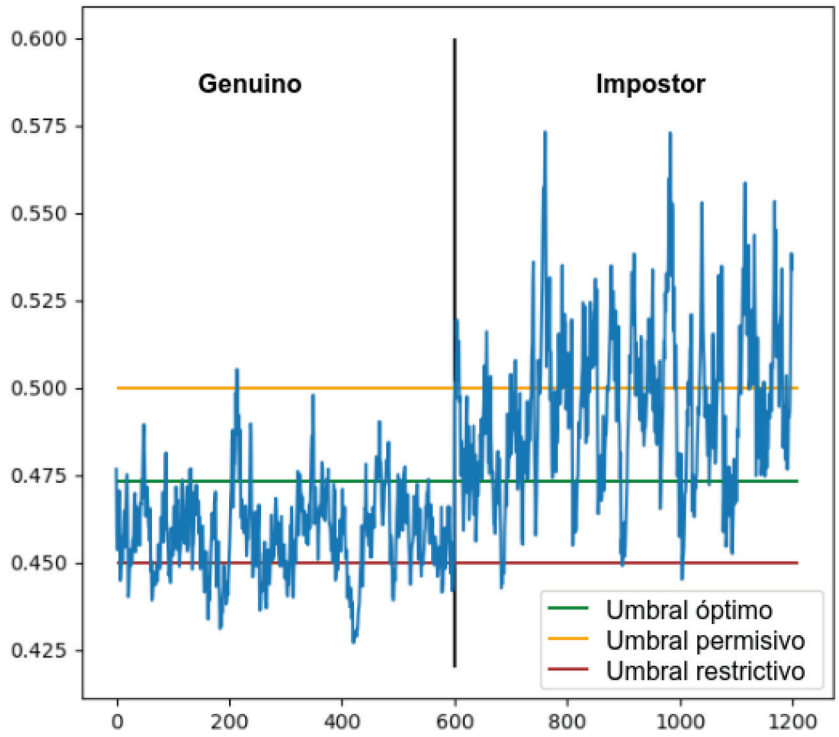
donde *WinSize* es el número de observaciones a tener en cuenta de la secuencia de riesgos. Una vez se ha aplicado la MME, los valores pasan de ser muy cambiantes de lo largo del tiempo, a ser mucho más estables (ver Figura 4.7b). Esto se debe a que, en esta ocasión, no es un único valor el que determina el riesgo asociado para un instante, sino el conjunto de *WinSize* observaciones el que lo hace, es decir, el histórico de los valores de riesgo. Es por esto que para establecer un nivel óptimo del parámetro *WinSize* debe existir un equilibrio entre un mejor desempeño a la hora de realizar predicciones y la usabilidad del método en un entorno real. Un valor alto del parámetro *WinSize*, considera más información y por lo tanto suaviza

mejor la curva obteniendo así más exactitud al categorizar las dinámicas de comportamiento. Sin embargo un valor bajo del parámetro *WinSize* genera un modelo de riesgos más usable, pues al considerar menos información se necesitan menos recursos computacionales para su funcionamiento y también se pueden empezar a realizar predicciones en un intervalo menor de tiempo.

Una vez calculado el *buffer* de riesgo, la siguiente tarea es determinar que comportamientos se consideran normales y cuales se consideran anómalos. Esto permite categorizar las observaciones en genuinas (pertenecen al mismo usuario objetivo) y en impostores (pertenecen a un usuario impostor). Para lograr esto se tiene que establecer una frontera de decisión en la curva de riesgo. Para ello se determina un umbral de tal manera que los valores de riesgo por debajo del umbral se consideran genuinos, y los valores por encima del umbral se consideran impostores (ver Figura 4.8).

La forma de determinar el umbral de forma óptima se fijándolo al valor que produce el menor EER. Tal y como se ha explicado en la Sección 3.3.4, este

Figura 4.8. Tipos de umbrales aplicados sobre el *buffer* de riesgo utilizando MME



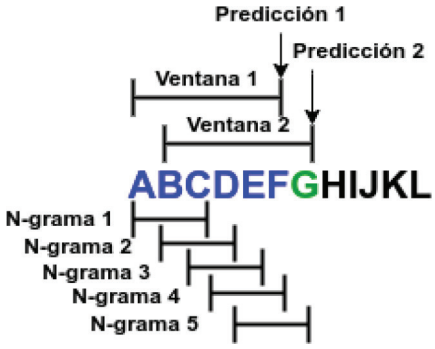
valor se calcula como la intersección entre la función de FAR y la función de FRR. A modo recordatorio, FAR se define como la probabilidad de que un usuario no autorizado (impostor) sea aceptado por el modelo. Por otro lado, FRR se define como la probabilidad de un usuario autorizado (genuino) sea rechazado injustamente por el modelo (sea considerado impostor). Tanto FAR como FRR son funciones, pues se pueden obtener múltiples valores para ellos dependiendo del umbral seleccionado. Fijar un valor extremo del umbral resulta en obtener un modelo restrictivo (menor FAR pero mayor FRR) o un modelo permisivo (menor FRR pero mayor FAR), pero nunca ambos valores menores simultáneamente. De esto modo, el EER se considera como el punto óptimo para fijar el umbral, pues es el valor que consigue obtener el mínimo valor para FAR y FRR simultáneamente.

Por último, cabe destacar que cuando se generan los n-gramas, la secuencia adyacente obtenida es la misma que la posterior excepto por el primer y último símbolo. Es por esto que el método propuesto en esta tesis puede realizar predicciones para cada interacción del usuario (es decir, para cada pulsación de teclado o movimiento de ratón) una vez que se han obtenido al menos *WinSize* interacciones (ver Figura 4.9).

4.5. Selección de parámetros

El método de combinación de la información propuesto en esta investigación necesita definir un total de dos parámetros y cuatro hiperparametros (ver Tabla 4.1). Los resultados obtenidos están muy ligados a la selección correcta de estos parámetros. Es por esto que es necesario hacer especial atención en entender y definir correctamente cada uno de ellos. Los parámetros requeridos por el modelo son *NGL* y *WinSize*.

Figura 4.9. Secuencia de predicciones en función de los parámetros del método. *N-Gram Length (NGL)* = 3 y *WinSize*=4



El valor de *NGL* determina el número de interacciones (teclado, ratón o ambas) que se agrupan para cada instante con el objetivo de obtener una predicción. Un valor extremo de este parámetro puede resultar en subajuste (para valores pequeños) y sobreajuste (para valores altos). La búsqueda de este parámetro se ha limitado a los valores [5,10,20,30] basándose en la experiencia empírica.

El valor de *WinSize* define cuanta información histórica de comportamiento (en forma de secuencias de n-gramas) va a ser utilizada para realizar una predicción. El objetivo de este parámetro es suavizar el *buffer* de riesgo. La búsqueda de este parámetro de ha limitado a los valores [5,10,20,50,100] basándose en la experiencia empírica.

El algoritmo de RTE necesita fijar dos hiperparámetros: el *Número de árboles* y la *Máxima profundidad* de cada árbol. Ambos hiperparámetros determinan la longitud del abecedario en el que cada muestra puede ser categorizada. De esta forma, un valor alto puede resultar en sobreajuste, pues se obtendrá un alfabeto muy amplio, en el que cada símbolo representa algo muy específico, haciendo que las secuencias obtenidas no se parezcan entre sí. Por otro lado, un valor pequeño de ambos resultara en un abecedario muy limitado, obteniendo así secuencias muy parecidas entre sí y por lo tanto no discriminatorias (subajuste). Para cada fuente de información, la búsqueda de estos hiperparámetros se ha limitado al rango [2, 5] basándose en la experiencia empírica.

Para el algoritmo de DBSCAN se necesitan fijar también dos hiperparámetros: *EPS* y *MINS*. Estos hiperparámetros se han delimitado utilizando una búsqueda en cuadrículas (en inglés, *grid search*). Para hacer la búsqueda más eficiente, los valores posibles se han acotado. De esta forma, los valores de *EPS* están en el rango $[0 : f_{100}]$, donde f_{100} representa la distancia con posición cien de la matriz de distancias ordenada de menor a mayor y descartando los valores iguales a cero. El parámetro *MINS* se ha limitado a los valores [2,10,50,100] basándose en la experiencia empírica.

Cabe destacar que fijar de forma óptima estos parámetros dependerá ampliamente de los requisitos particulares de los datos en donde se quiera aplicar el método. Sin embargo, se debe tener en cuenta algunas consideraciones. Fijar los parámetros *NGL* y *WinSize* a valores altos repercute en obtener un modelo con más exactitud a la hora de clasificar dinámicas de comportamiento. Esto se debe a que, el aumento de cualquiera de ellos hace que se considere más información para realizar una predicción. Sin embargo, esto también repercute en la usabilidad.

Por otro lado, para fijar los hiperparametros de RTE y DBSCAN se recomienda utilizar una búsqueda en cuadrículas acotando los posibles valores para realizarla de forma eficiente. Hay que tener en cuenta las consideraciones explicadas anteriormente para evitar tanto el sobreajuste como el subajuste.

Tabla 4.1. Resumen de los parámetros e hiperparametros del método de combinación de la información de comportamientos. f_{100} representa la distancia con posición cien de la matriz de distancias ordenada de menor a mayor y descartando los valores iguales a cero

Nombre	Descripción	Valores
NGL	Longitud del N-grama	[5,10,20,30]
WinSize	Información histórica considerada en forma de secuencias	[5,10,20,50,100]
Número de árboles	Número de árboles generados en el RTE	[2,5]
Máxima profundidad	Máxima profundidad de los árboles del RTE	[2,5]
EPS	Máxima distancia entre dos muestras para ser consideradas vecinas en DBSCAN	$[0 : f_{100}]$
MINS	Mínimo número de puntos que tiene que tener una vecindad para considerarse *cluster* en DBSCAN	[2,10,50,100]