UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de la Rioja (UNIR)

Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Análisis y Visualización de Datos Masivos

Caracterización de equipos informáticos mediante clustering en una red empresarial

Trabajo Fin de Máster

Presentado por: Javier Artiga Garijo

Director: Luis Miguel Garay Gallastegui

Ciudad: Logroño
Fecha: 15 de julio de 2020

Índice general

Resumen

En la monitorización de redes informáticas a gran escala, resulta de alto interés conocer el comportamiento de sus equipos finales y detectar aquellos que puedan ser sospechosos. Sin embargo, clasificar a cada equipo según su actividad supone un importante ejercicio de síntesis. Además, es difícil obtener unas categorías útiles, sobre todo en el caso de las anomalías, ya que son desconocidas a priori. El presente trabajo aborda este reto haciendo uso de técnicas de *clustering* a partir de logs extraídos de firewalls. Con un muestreo del 5 %, que supone un millón de sesiones al día, se han distinguido 5 clases de comportamientos. Los comportamientos anómalos se han conseguido reunir en un solo cluster con menos de diez casos al día.

Palabras clave: Monitorización de redes empresariales, clasificación de tráfico, detección de anomalías, seguridad informática, clustering, aprendizaje automático no supervisado

Abstract

In network monitoring at a large scale, knowing how the hosts behave and detecting those who might be suspicious is of high interest. Nevertheless, classifying each host according to its activity requires a major summary exercise. Besides, it is hard to get useful categories, especially in the case of anomalies, since they are a priori unknown. The present work addresses this challenge using clustering techniques on firewall logs. With a sampling of 5 %, which means a million of sessions each day, 5 classes have been identified. The anomalous behaviours have been gathered in a single cluster with less than ten cases each day.

Keywords: Corporate network monitoring, traffic classification, anomaly detection, computer security, clustering, unsupervised machine learning

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

El primer capítulo resume de forma esquemática y clara las ideas que componen este trabajo y sobre las que se fundamenta. En él se presentan elementos como la identificación del problema a tratar, la justificación de su importancia, cómo se vertebra el proyecto y de qué manera contribuye a la resolución de los retos planteados.

1.1. Motivación

La monitorización de la red informática en una gran empresa es un problema complejo por muchos factores. El más inmediato podría ser el alto volumen de conexiones que se producen, superior a varios millones diarios. Pero se suman otras muchas dificultades a la hora de procesar el tráfico de forma que se obtenga información útil para el analista y, en última instancia, para el cliente final: la gran variabilidad de comportamientos, la complejidad de sintetizar lo importante sin perder exactitud, el compromiso entre rapidez en la respuesta y certeza en su fiabilidad, el desconocimiento a priori de cómo se caracteriza un comportamiento anómalo, etc.

Interesa buscar una solución a estos obstáculos porque las empresas quieren garantizar que los recursos de sus redes se gestionan de la manera más óptima posible. Además, es especialmente importante la seguridad de la red corporativa, esto es, protegerla de acciones no autorizadas u otras amenazas que puedan comprometer su disponibilidad o la integridad de los equipos que la componen.

Equipos de seguridad como puedan ser los firewalls contribuyen de manera decisiva a esta protección, pero su funcionamiento basado en firmas no cubre todos los casos ante una intrusión. Sin embargo, sí generan una enorme cantidad de datos que, si se tratan adecuadamente, sirven para ampliar el alcance de las técnicas empleadas en materia de

seguridad.

Por ello, un sistema que modele el comportamiento normal de una red y detecte anomalías usando métodos estadísticos y de inteligencia artificial permitirá conocer mejor el contexto de dicha red e identificar comportamientos sospechosos que no se considerarían de otro modo.

1.2. Planteamiento del trabajo

Las cuestiones planteadas para guiar esta investigación han sido:

- Si podemos clasificar las direcciones IP de una gran red empresarial en categorías relevantes según su comportamiento de red
- Cuáles serían esas categorías
- Si sereremos capaces de identificar comportamientos sospechosos en base a esta clasificación

A través de estas cuestiones se inicia el estudio del problema, para cuya solución se propone obtener un modelo clasificador que distinga patrones de comportamiento normales y desviaciones respecto de la actividad normal. Este clasificador se realizará mediante una técnica de aprendizaje automático no supervisado como es el *clustering* (técnica que lleva a cabo una agrupación en categorías de manera natural, buscando características en común sin haber definido las clases previamente).

El objetivo principal es localizar en la red interna orígenes de tráfico catalogable como extraño, lo que puede indicar un equipo infectado o mal configurado. La determinación de qué se sale de lo habitual dependerá de las particularidades de la red, algo difícil de concretar a priori y más aún de generalizar, razón por la cual el *clustering* (como etapa final tras un análisis y preprocesado de los datos adaptado al caso) se ha considerado una técnica idónea en esta tarea.

1.3. Estructura del documento

Una vez expuesto en este capítulo 1 el problema que se va a abordar durante este trabajo y cómo se enfoca su estudio, se dedica el capítulo 2 a definir los objetivos, tanto generales como específicos, hacia los que se dirigirá el proyecto. También se incluye en él la metodología seguida. Esta explica qué pasos se dan en la aplicación del *clustering* a este caso, el por qué de cada paso, qué instrumentos se van a utilizar, cómo se analizan los resultados, etc.

El capítulo 3 dota de contexto científico a este trabajo: se revisa el estado del arte. Mediante la relación de diversos artículos y publicaciones académicas sobre aprendizaje automático aplicada a la clasificación de tráfico, en esta parte de la memoria se sintetizan los conceptos fundamentales y los hallazgos más relevantes para este campo. Se diferencian distintos enfoques en la detección de anomalías, valorando los aspectos positivos y las desventajas de cada uno. Para cerrar, se justifican las decisiones tomadas con las que, sobre el conocimiento disponible en torno al tema, sentar las bases del desarrollo de este piloto.

En el capítulo 4 se entra en los detalles del desarrollo. Paso a paso, se profundiza en todos los puntos que se han recorrido hasta alcanzar unos resultados satisfactorios. Esto incluye: presentación del escenario, trabajo con estándares y *scripts* para extraer de los datos en bruto la información útil para el propósito actual, análisis estadístico de las variables, selección de estas y parametrización del algoritmo K-Means para obtener una clasificación. Todo ello se ha ido conseguiendo y mejorando de forma incremental, volviendo atrás e incorporando avances. En el capítulo se presenta ordenadamente, siguiendo la línea a través de la cual los datos se transforman por etapas hasta tener finalmente la clasificación de los equipos.

Es entonces cuando se pasa al capítulo 5: resultados. Ahí se describe más ampliamente y se analiza la salida del algoritmo, evaluando cómo se han compuesto los *clusters* y qué indicadores reflejan la calidad del resultado. Se interpreta el valor del mismo y qué significa este resultado de cara a los objetivos marcados.

En último lugar, el capítulo 6 recoge las conclusiones derivadas del trabajo realizado. Se comentan también las líneas futuras que este piloto podría seguir para evolucionar.

Capítulo 2

Objetivos y metodología

Se procede a explicar en este capítulo los objetivos concretos y la metodología de trabajo que van a marcar cómo se materializará la contribución que se va a realizar.

2.1. Objetivo general

El objetivo general del presente trabajo es extender la capacidad de monitorización que se tiene sobre una red empresarial. Se enfocará para ello en la detección de anomalías, haciendo uso de técnicas basadas en los equipos que la componen (a diferencia de las basadas en flujos de tráfico). De este modo, se desarrollará un sistema que categorice mediante *clustering* los equipos informáticos finales (es decir, todos menos la infraestructura de la red) y que pueda revelar cuándo la actividad de un equipo se desvía de su comportamiento habitual.

Con ello se espera seguir mejorando las prestaciones ofrecidas, especialmente en seguridad, pero también ayudará en la identificación de problemas de configuración, cambios de tendencia en la red y en definitiva cualquier evento inesperado que pudiera repercutir negativamente en la operativa normal de la empresa monitorizada.

2.2. Objetivos específicos

La consecución de dicho objetivo general vendrá pautada por las siguientes metas más específicas, que se abordarán en este orden:

- Determinar qué características son las más relevantes a la hora de *clusterizar* la red y extraerlas en tiempo real.
- Establecer unas categorías básicas mediante clasificación no supervisada, que sean verificables con la documentación que se tiene de la red.

- Comprobar que el sistema detecta anomalías.
- Refinar el algoritmo de *clustering* empleado, buscando categorías más específicas que no se estuvieran teniendo en cuenta antes.
- Alcanzar un modo de funcionamiento en tiempo real, en coordinación con los demás mecanismos de monitorización presentes y asegurando una precisión razonable.

2.3. Metodología de trabajo

Con los anteriores objetivos específicos marcados, se hace necesario establecer un marco de trabajo para llevarlos a cabo. En esta sección se estructurará una metodología de trabajo en base a cuatro fases, que tienen como fin último alcanzar el objetivo general. Sus enunciados vendrán acompañados de una descripción, explicando con un poco más de detalle cómo se planea ejecutar cada etapa.

- Selección de las características más relevantes para la clasificación

  A partir de la heterogénea colección de logs de varios firewalls de la que se dispone, se identificarán los datos que puedan ser interesantes para el *clustering*. Se tendrán en cuenta distintos aspectos que respondan a las actividades que realiza un equipo informático en una red empresarial, como por ejemplo la cantidad de conexiones, sus duraciones, horarios de actividad, interacción con ciertos servidores, etc. A continuación, se analizarán estas características a través de técnicas estadísticas, gracias a las cuales se adquirirá un entendimiento preliminar de la importancia que cada una supone para la posterior clasificación de instancias. Es posible que se logre la reducción de su dimensionalidad, la eliminación de alguna característica redundante o en definitiva alguna simplificación que haga el *clustering* más sencillo de ejecutar.

- Obtención de categorías mediante *clustering*

  Se pasará entonces a aplicar por primera vez algunos métodos de *clustering* sobre estos datos. Lo que se pretende es categorizar los equipos en su comportamiento ‘‘normal’’, de forma que se tengan clasificados en varios tipos según su actividad. Inicialmente se elegirá un algoritmo simple y ampliamente conocido como es K-Means, con variaciones sobre él. Se experimentará con diferentes valores para sus parámetros, ya que así podrán compararse resultados. De cara a determinar la efectividad de los algoritmos, se considerarán medidas objetivas que representen su rendimiento. Los métodos típicos

para ello valoran características intrínsecas o derivadas como su complejidad, estabilidad y tiempo de computación, así como la suma de cuadrados de las desviaciones. Además, se emplearán los indicadores comunes con los que se suele definir la bondad de una técnica de clasificación en aprendizaje automático.

- Detección de anomalías con análisis del *clustering*

  Una vez se cuente con una clasificación satisfactoria, se procederá a la siguiente fase, en la cual se aspira a detectar anomalías en los datos. En caso de contar con instancias que se hayan identificado como anomalías mediante otros métodos, se usarán para probar la capacidad de detección del sistema. Dado que también se desea detectar otras clases de anomalías más genéricas (no solo relacionadas con seguridad), se incluirán casos como cambios de configuración o de equipos que hayan cambiado de rol en la red.

- Evaluación en escenario real

  Finalmente, se desplegará el sistema desarrollado en un entorno de producción. Se ofrecerá una representación visual adecuada que permita al analista aprovechar el valor que el sistema aportará a la hora de revisar casos alertados. Además, se automatizará la extracción de características y se prepararán el resto de componentes del sistema para que funcione lo más cercano a tiempo real posible.

2.4. Metodología del desarrollo

Tras haber completado las fases del desarrollo requeridas para poner en funcionamiento este piloto y haber analizado y probado lo que mejor funcionaba en cada una, se ha plasmado la metodología seguida en esta serie de pasos, que ofrecen una visión global del desarrollo que se ampliará en el capítulo 4:

- A partir de logs en bruto de firewall, extracción de los vectores de características de las sesiones.
- Preprocesado antes del *clustering*:
  - Muestreo (necesario para el desarrollo, pero prescindible cuando se lleve a producción).
  - Transformación de los datos individuales en características para la entrada del *clustering*, formada básicamente, por un lado, por la agrupación diaria y por IP origen, y por otro, por el cálculo de las métricas con las que se ha decidido trabajar.

  - Normalización de estos datos.

- Aplicación de KMeans con $k = 5$
- Análisis:
  - Revisión manual en el entorno de la empresa de las IP clasificadas como anomalías.
  - Evaluación de la estabilidad del resto de *clusters* normales, para vigilar si hay cambios de tendencia.

El siguiente esquema de la figura 1 resume todo el procesado:

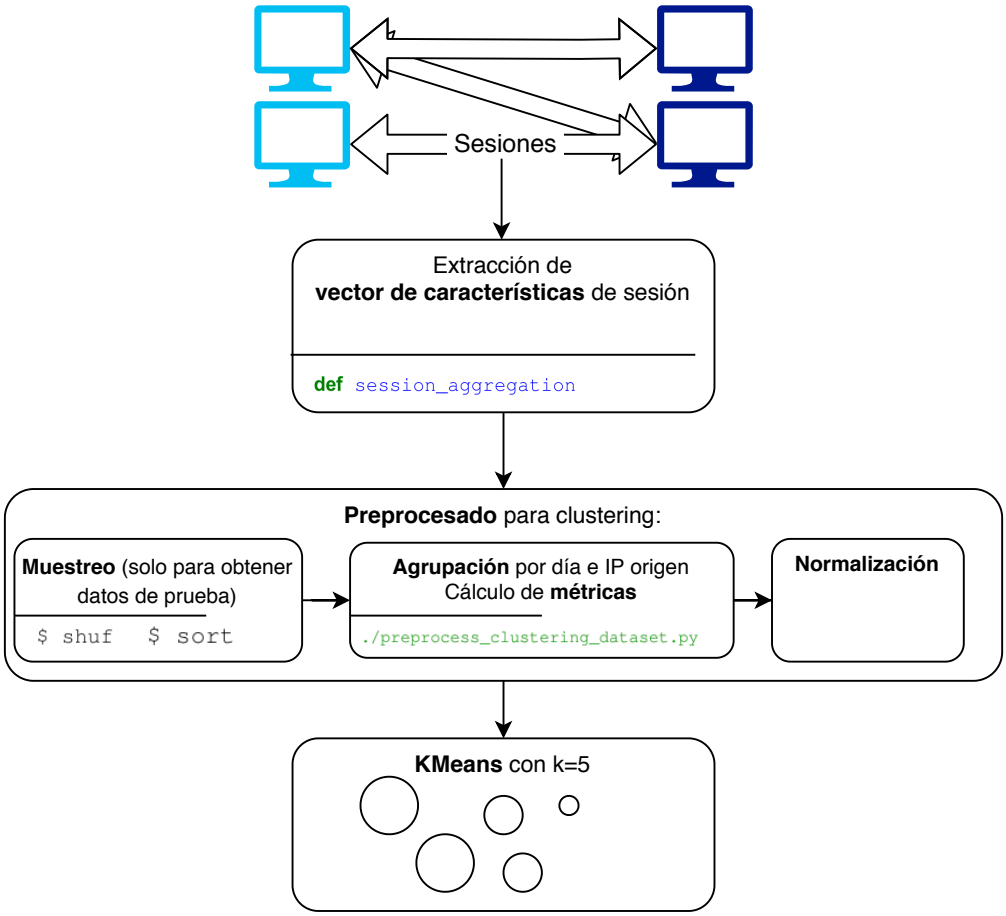

Figura 1: Diagrama de la metodología seguida en el procesado

Capítulo 3

Estado del Arte

En este capítulo se proporcionará un contexto adecuado para el trabajo, repasando el estado del arte que concierne al aprendizaje automático aplicado a clasificación de tráfico de red. Se distinguirá entre enfoques basados en flujos y basados en hosts. Después, se ahondará en la relevancia de la detección de anomalías dentro de la seguridad informática. Ahí se contrapondrán las aproximaciones basadas en firmas y las basadas en comportamientos anómalos. Entre las técnicas que se usan para estas últimas, se va a destacar el *clustering*, en torno al cual gira este desarrollo. Se relacionarán en la última sección algunos trabajos que aplican dicha técnica a la clasificación de tráfico.

3.1. Aprendizaje automático en la clasificación de tráfico

El aprendizaje automático puede resumirse como una colección de técnicas de gran potencial para la minería de datos y el descubrimiento de conocimiento (Nguyen y col., 2008). En concreto, a la hora de extraer este conocimiento, estas técnicas son especialmente eficaces buscando y describiendo patrones estructurales útiles en los datos. Además, la ventaja intrínseca de esta disciplina frente a la exploración que pueda efectuar un especialista es que, al poder definirse algorítmicamente el procedimiento para su aplicación, se hace posible implementar sistemas automatizados sobre equipos informáticos.

Un sistema de *machine learning* aprende automáticamente de la experiencia y perfecciona su base de conocimiento, entendiendo ‘‘aprender’’ como la acción de mejorar su rendimiento en el desempeño de una tarea determinada (Herbert, 1983). Esta mejora debe ser cuantificable objetivamente en base a lo que se denomina como medida de rendimiento. El sistema (como cualquier otro agente computacional en la rama de la inteligencia artificial) recibirá estímulos externos (en este caso, conjuntos de datos) y, en base a ellos y al conocimiento que recogen

el algoritmo y los resultados de sus ejecuciones previas, producirá una salida que maximizará la medida de rendimiento establecida.

En terminología de *machine learning*, el conjunto de datos que se toma como entrada se compone de instancias. Una *instancia* simboliza a un individuo específico de la población sobre la que se trabaja. Cada instancia se representa por sus valores en una serie de *características* o atributos, que no son más que medidas sobre aspectos de interés para el escenario en cuestión. Un conjunto de instancias con ciertas características comunes pertenece a una *clase* o concepto. De este modo, se aprende un concepto cuando, dada una instancia, se logra identificar correctamente con qué clase se corresponde. Este aprendizaje implica también que se es capaz tanto de generalizar la aplicación del nombre de la clase a todos los miembros de la misma como de discriminar a los miembros que pertenecen a otra clase.

Atendiendo a la naturaleza de las clases resultantes, los tipos de aprendizaje se dividen en aprendizaje supervisado y no supervisado. El primer tipo es capaz de clasificar nuevas instancias en clases predefinidas. Por el contrario, el segundo clasifica las instancias en clases no definidas con anterioridad. La técnica de aprendizaje no supervisado principal es el *clustering* o agrupamiento, que se explicará con detalle más adelante.

En los últimos tiempos, el aprendizaje automático se ha venido usando cada vez más en la clasificación de tráfico IP (Dainotti y col., 2012). Las técnicas de clasificación de tráfico siguen mejorando en acierto y eficiencia, pero la proliferación constante de aplicaciones en Internet con comportamientos muy variados, sumado a los incentivos que tienen ciertos agentes para enmascarar algunas aplicaciones y así evitar el filtrado o bloqueo en firewalls, son algunas de las razones por las que la clasificación de tráfico permanece como uno de los muchos problemas abiertos de Internet. Han quedado obsoletos métodos clásicos como la identificación de aplicaciones en base a sus puertos conocidos de nivel de transporte TCP/UDP (aquellos registrados por la IANA), que resulta muy simple y rápida pero también poco fiable. En el otro extremo, las técnicas de ‘‘Deep Packet Inspection’’(Finamore y col., 2011), que analizan en profundidad el funcionamiento de las aplicaciones desde la perspectiva de su uso de los protocolos o buscan datos específicos en paquetes IP para inferir a qué aplicación pertenecen, suponen una alta carga computacional y habitualmente requieren hardware específico. Además, el buen funcionamiento de un clasificador DPI está supeditado a dos condiciones: que pueda inspeccionar el contenido de los paquetes IP y que sepa cómo interpretar la sintaxis de cada aplicación. La primera condición queda comprometida por la estandarización de las conexiones cifradas, mientras que la viabilidad de la segunda se vería restringida por la complejidad de contar con un repositorio completo y constantemente

actualizado del formato de los paquetes que puede generar cada aplicación. Ante estas técnicas también se presentan dificultades legales y relacionadas con la privacidad.

Si se trata de clasificadores de tráfico, lo más común es encontrar planteamientos basados en flujos de tráfico. En ocasiones, la granularidad de la clasificación se afina hasta el uso de flujos bidireccionales (asumiendo que se tiene visibilidad de ambas direcciones), pero operar a este nivel entraña una complejidad bastante mayor. Un flujo se suele definir como una tupla de 5 elementos: protocolo de transporte (frecuentemente, TCP o UDP), direcciones IP de origen y destino, y puertos de origen y destino. Con este concepto como *objeto* fundamental, tradicionalmente se han buscado patrones estadísticos en los atributos de los flujos que son observables desde una perspectiva externa (es decir, sin considerar el contenido o *payload* de los paquetes). Ejemplos de estos atributos serían: tamaño de paquetes, tiempo entre llegadas, número de paquetes en cada dirección, duración total, etc. resumido cada uno con el estadístico muestral que se considere adecuado. Con la popularización del aprendizaje automático, se ha podido llevar la búsqueda de patrones entre dichos atributos a nuevos grados de profundidad.

Se trabaja también con otras variantes en cuanto a cómo agrupar los paquetes que se hayan intercambiado dos máquinas. Entre ellas, podrían destacarse las conexiones TCP o los servicios, definidos estos como el tráfico generado entre una pareja de IPs-puertos. En cualquiera de los casos anteriores, se pone el foco sobre flujos individuales, para después clasificarlos bajo categorías que comparten características. Este tipo de planteamientos no tienen tan en cuenta el conjunto de acciones que lleva a cabo un mismo equipo. Así, se corre el riesgo de perder información útil de cara a entender de forma completa qué es realmente lo que están haciendo los equipos de la red.

Por otro lado, se encuentran los clasificadores de tráfico basados en el comportamiento del host. Sirva de referente el trabajo de (Karagiannis y col., 2005), donde se propuso un novedoso método que identificaba patrones en el comportamiento de los hosts a la altura de la capa de transporte. Se trataba de una aproximación multinivel que descartaba incluir datos sobre el payload, los puertos bien conocidos (aspectos que conllevan las problemáticas anteriormente comentadas) o cualquier otra información separada de la que ofrecían los colectores de flujos. Consistía por tanto en un clasificador ‘‘a ciegas’’ (*‘‘BLINd Classification’’, abreviado como ‘‘BLINC’’*) que analizaba cada hosts desde tres perspectivas: social, funcional y aplicativa. La perspectiva social capturaba las interacciones del host con otros hosts, en términos de cuántos hosts se conectaban con qué hosts. La funcional los separaba según actuaran como proveedores de un servicio, consumidores o ambos. Se tenían en cuenta, por tanto, los roles

del modelo cliente-servidor. Por último, en la perspectiva aplicativa se utilizaba la información por encima de la capa de transporte con la intención de distinguir la aplicación en cuestión.

Mediante la premisa de no tratar cada flujo como una entidad distinta, se conseguiría acumular la información necesaria para reconocer el verdadero comportamiento de cada equipo informático final. Además de cumplir con la identificación de aplicaciones específicas, este método sería resistente a circunstancias de la red como congestión o cambios de rutas. Esto es así porque, a diferencia de otros métodos (véanse los mencionados sobre flujos), una aproximación centrada en el comportamiento de los hosts suele ser insensible a las variaciones que puedan presentar parámetros como los tiempos de llegada entre paquetes. En cuanto a resultados, sobre sistemas de detección de anomalías se suelen aplicar enfoques de aprendizaje automático basados en patrones de comunicación entre los equipos informáticos. Estos alcanzan resultados comparables a los de técnicas de DPI sobre firewalls, siendo notablemente más asequibles y menos invasivos con la privacidad.

Es por todo lo anterior que en los sistemas de detección de anomalías, que se van a desarrollar en la siguiente sección, priman los enfoques sobre el equipo informático final en vez de sobre el flujo.

3.2. Detección de anomalías sobre actividad de red

En la gestión y monitorización de una red empresarial cobra especial relevancia la seguridad. En este ámbito, la seguridad informática se centra en proteger la red corporativa de ataques que puedan comprometer su disponibilidad o la integridad de los equipos que la componen, así como bloquear acciones no autorizadas y evitar el uso indebido de los recursos que quedan expuestos al exterior.

Las organizaciones toman numerosas medidas de seguridad frente a estas amenazas, tanto *software* como *hardware*. Dos ejemplos claros serían los antivirus y los firewalls, que podríamos englobar dentro de las aproximaciones a la seguridad basadas en firmas (D'Alconzo y col., 2019). Sin embargo, estos métodos dependen de que el fabricante del producto de seguridad haya detectado el ataque previamente, haya generado una firma que lo identifique y haya distribuido la misma hasta el cliente final. Es decir, solo pueden ofrecer protección ante ataques conocidos y requieren que todos los pasos anteriores se hayan completado.

En contraposición, existen los sistemas de seguridad basados en detección de anomalías. Este tipo de métodos asumen que el impacto de un ataque modificará el comportamiento de la red, así que construyen un modelo que represente el comportamiento normal de la red, especificado por ciertas métricas. A continuación, monitorizan el tráfico y fijan alarmas que se

dispararán cuando el valor recogido en alguna de esas métricas de referencia se desvíe del rango considerado normal (Boutaba y col., 2018).

Habitualmente, este tipo de defensas basadas en detección de anomalías son complementarias a las basadas en firmas. Se sitúan en una segunda línea con el objetivo de detectar a tiempo síntomas tempranos de ciberataques de tipo *zero-day*, para así poder actuar antes de que causen daños. Ambos enfoques pueden encontrarse integrados en soluciones conocidas como IDS/IPS (*Intrusion Detecion/Prevention System*).

Hablando en términos generales, pueden distinguirse tres fases básicas que cumplen todos los NIDS (*Network* IDS) basados en anomalías (García-Teodoro y col., 2009):

- *Parametrización*: las instancias del sistema objetivo se representan de forma adecuada para su tratamiento.
- *Entrenamiento*: se caracteriza el comportamiento normal del sistema, mediante un modelo que puede construirse con técnicas basadas en alguna de las categorías descritas después.
- *Detección*: se compara el modelo con el tráfico disponible, de forma que se dispara una alarma si una instancia se desvía.

Según cómo se modele el comportamiento normal del sistema (Lazarevic y col., 2005), las técnicas pueden categorizarse en: basadas en estadística, en conocimiento o en aprendizaje automático. Las primeras no requieren un conocimiento previo sobre la actividad normal del sistema, pero la presunción que asumen de cuasi-estacionalidad es poco realista. Las segundas son robustas, pero el mantenimiento de datos de calidad resulta difícil y costoso. En cuanto a las técnicas basadas en aprendizaje automático, son flexibles y adaptables. También pueden capturar interdependencias entre las variables que no son fáciles de encontrar de otra forma. No obstante, estas técnicas tienen una dependencia importante de lo que se acepte como comportamiento normal.

Para este trabajo, se ha elegido desarrollar un sistema de detección de anomalías basado en una técnica de aprendizaje automático como es el *clustering*. En (D'Alconzo y col., 2019) se resaltan acertadamente las bondades y debilidades de los métodos de detección de anomalías, al decir que ‘‘son atractivos porque permiten la pronta detección de amenazas desconocidas (por ejemplo, *zero-days*). Estos métodos, sin embargo, puede que no detecten ataques sigilosos, insuficientemente amplios para perturbar la red. A veces también adolecen de un alto número de falsos positivos.’’

Continúa señalando cómo beneficia el aprendizaje automático a este tipo de sistemas: ‘‘el

machine learning ha recibido una significativa atención en la detección de anomalías, debido a la autonomía y robustez que ofrece en el aprendizaje y también a la hora de adaptar el perfil de la normalidad según va cambiando. Con *machine learning*, el sistema puede aprender patrones de comportamientos normales dentro de entornos, aplicaciones, grupos de usuarios y a lo largo del tiempo. Además, ofrece la capacidad de encontrar correlaciones complejas en los datos que no pueden deducirse de la mera observación''.

Se concluye por tanto que la obtención de una representación completa de la normalidad, requisito no trivial en estos sistemas basados en detección de anomalías, puede tomarse como un problema de clasificación en instancias normales y no normales. Dicho problema puede abordarse mediante la técnica de aprendizaje no supervisado descrita a continuación.

3.3. Clustering

El *clustering* se define en (Nguyen y col., 2008) como la agrupación de instancias que tienen características *cercanas* en forma de *clusters*, sin aplicar ninguna orientación previa. Esta técnica de aprendizaje automático no supervisado asocia a las instancias con propiedades similares bajo el mismo grupo, determinando dicha similaridad en un modelo que posibilite la medición de distancias específicas, como pueda ser el espacio euclídeo. Los grupos pueden ser exclusivos, si cada instancia pertenece a un único grupo; solapados, si una instancia puede pertenecer a varios grupos; o probabilísticos, si la pertenencia de una instancia a un grupo se expresa mediante una cierta probabilidad.

El primer uso de *clustering* para detección de intrusiones se vio en (Portnoy, 2000). La hipótesis en base a la cual los autores aplicaron *clustering* para esta tarea es que las conexiones entre datos normales crearán *clusters* más grandes y más densos. Si se lleva el análisis un paso más allá, para incrementar la precisión de la técnica, también debe tenerse en cuenta la distancia entre *clusters*, como se demuestra en (Jiang y col., 2006).

Los tipos de algoritmos de *clustering* usados en clasificación de tráfico son variados. Encontramos en (McGregor y col., 2004) la primera aplicación de un algoritmo de *clustering* probabilístico como es *Expectation Maximization* sobre flujos de tráfico. Considerando varias estadísticas sobre longitud de paquetes, tiempos entre llegadas, cantidad de bytes, duración de la conexión y número de permutaciones entre conexiones de tipo ''transaccional'' y de tipo ''por lotes'', se consigue una clasificación de aplicaciones con baja granularidad. En (Zander y col., 2005), se usan estadísticas similares y un método de selección de características para encontrar el conjunto de características óptimo que aplicar a la entrada del algoritmo de *clustering* ''AutoClass''. Así lograron diferenciar con alta granularidad entre ocho aplicaciones

previamente estudiadas, demostrando una precisión media del 86 %.

Otros estudios valoraban muchas menos variables. En (Bernaille y col., 2006b), tan solo se miraban la longitud y dirección de los primeros cinco paquetes de cada flujo (excluyendo el *handshake* y los ACKs sin *payload*) para decidir mediante K-Means a cuál de las diez aplicaciones tipo pertenecía cada flujo. Permitía una temprana identificación de la aplicación con una precisión cercana al 80 %, extensible hasta el 85 % con mejoras posteriores (Bernaille y col., 2006a) (Bernaille y col., 2007).

Para la detección de anomalías también se usan habitualmente algoritmos de *clustering*. Se advierte en (Leung y col., 2005) del alto coste de tener datos etiquetados a priori, por lo que se aborda la detección de anomalías mediante técnicas de aprendizaje automático no supervisado. Esta decisión se fundamenta en dos presunciones sobre los datos. La primera, que la mayoría de las conexiones de red serán tráfico normal, y solo un pequeño porcentaje del tráfico será malicioso (Portnoy, 2000). La segunda, que el tráfico de un ataque será estadísticamente distinto al tráfico normal (Javitz y col., 1993). Estos trabajos citados, junto con otros muchos como los que se recogen y comparan entre sí en (Bhuyan y col., 2014) a través de varias medidas de validez de los *clusters*, demuestran la utilidad de la técnica del *clustering* en esta tarea.

Comprobada la idoneidad del *clustering* para detectar anomalías, en (Syarif y col., 2012) se prueban varios algoritmos de *clustering* frente a varios algoritmos clasificadores. Se resalta que, como se ha apuntado anteriormente, los algoritmos clasificadores (como aprendizaje automático supervisado que son) son incapaces de detectar formas de intrusión o ataques no identificados previamente. Se elaboraron experimentos sobre un conocido *dataset* (el *dataset* de detección de intrusiones de la KDD Cup ’99). Sus resultados revelaron que las técnicas de *machine learning* supervisado testeadas (Naive Bayes, reglas de inducción, árboles de decisión y KNN) no alcanzaban el 64 % de precisión ante intrusiones desconocidas. Por el contrario, las técnicas de *machine learning* no-supervisado lograron tasas de detección del 80 %, aunque los falsos positivos superaban el 20 %. Estas técnicas eran los siguientes algoritmos de *clustering*: K-Means, *k-medoids* (Velmurugan y col., 2010), *Expectation Maximization* (Lu y col., 2009) y detección de *outliers* basada en distancia (Orair y col., 2010).

Resumiendo, se ha constatado que el aprendizaje automático es adecuado para la clasificación de tráfico. Basarlo en el comportamiento del host tiene en cuenta las acciones con el mismo origen, además de ser resistente a cambios de red. Por último, en el ámbito de la seguridad, la detección de anomalías con aprendizaje automático es flexible y adaptable, puede capturar interdependencias entre variables y permite la pronta detección de *zero-days*.

Capítulo 4

Desarrollo

Este capítulo se centra en explicar cómo se ha llevado a cabo el desarrollo específico de la contribución. Se empezará por una presentación del escenario real que ha servido de fuente de datos, detallando el proceso de extracción y filtrado de los datos. Una vez vista la manera de reducir el volumen y obtener la información de interés, se expondrán los pasos de análisis y preprocesado previos al *clustering*, así como su parametrización. Se hará hincapié también en cómo se seleccionan las características más importantes para aplicar algoritmos.

4.1. Presentación del escenario

En el escenario del proyecto (la red de un banco), el tráfico atraviesa distintos equipos de seguridad según el entorno al que corresponda. Si se trata de tráfico que procede de los usuarios externos conectándose a servicios públicos del banco (expuestos a Internet), se cursa a través de una batería de equipos formada por: un IPS PaloAlto, un WAF[1] de marca Fortinet, balanceadores F5, otros firewalls Checkpoint y, finalmente, los servidores. En caso de ser tráfico desde la red corporativa hacia Internet, los equipos en este camino son el IPS PaloAlto mencionado (pero en esta ocasión está funcionando solo como IDS) y un firewall Fortinet, en este orden. Si son conexiones internas entre diferentes organizaciones, se procesa con otro firewall Fortinet independiente, etc.

El objetivo de este trabajo se centra en la clasificación de los equipos de la red empresarial, así que se pondrá la atención sobre el segundo itinerario descrito, de color amarillo en la figura 2: el tráfico saliente, desde la red hacia Internet. En especial, se quiere exprimir la información que proporciona el firewall Fortinet de ‘‘Internet Corporativo’’, ya que es principalmente quien reacciona con mecanismos de prevención o bloqueos ante las acciones iniciadas en la red

[1] *Web Application Firewall*

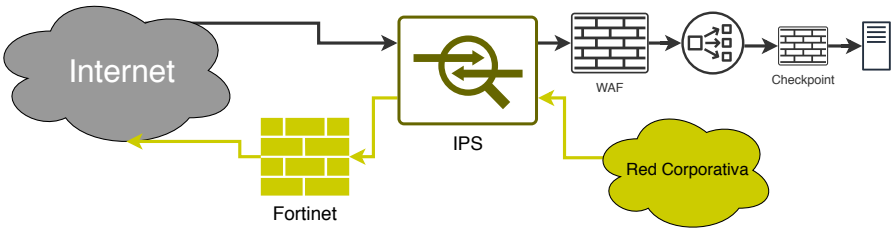

Figura 2: Diagrama del tratamiento de conexiones entrantes y salientes

corporativa. Como se ha explicado, al estar el IPS actuando como IDS en este ámbito y además procesando tráfico de otras redes y en otras direcciones, su información nos interesa menos. Sin embargo, si en un momento dado del desarrollo de este método de *clustering* se cree beneficioso incorporar los eventos que arroja el IPS sobre las sesiones de los hosts de la red, será necesario que se hayan analizado correctamente y puedan extraerse con facilidad. Por eso, en la siguiente sección se detalla cómo se han estudiado los logs de ambos.

4.2. Extracción y filtrado

El punto de interés para la captura se concentra por tanto en dos equipos por los que pasa el grueso del tráfico total: un firewall Fortinet y un IPS PaloAlto. Como es habitual, estos equipos reportan todas sus acciones a través de logs, con varios niveles de granularidad e importancia. Incluyen también multitud de información aportada por los propios sistemas que enriquecen el valor de cada evento. Esto es razón para preferir los logs de firewall como fuente de información frente a una captura de tráfico en crudo, ya que lo que procesan los firewalls casi siempre es más relevante que el tráfico completo pero sin procesar. Además, el volumen de una captura de tráfico sería notablemente mayor y más difícil de manejar.

La cantidad de datos recibida en los logs sigue siendo alta, pero además necesita ser procesada para obtener datos útiles. En los párrafos siguientes se explica cómo se han extraído los datos de sesión que se usarán como materia prima para tener finalmente unos datos de entrada al algoritmo de *clustering*. Se ha tomado una semana de muestra para trabajar con un periodo acotado. Nótese en la figura 3 la reducción en esta primera etapa del volumen de los datos, que seguirán transformándose en sucesivas fases hasta mantener únicamente la información valiosa para la tarea que nos ocupa.

Aunque el fin para el que sirven ambos equipos (análisis y protección frente a amenazas informáticas) sea similar, la estructura usada por cada uno en los logs que produce es completamente distinta. Los logs de Fortinet siguen un formato clave-valor con ciertas particularidades, mientras que los de PaloAlto tienen una serie de campos fijos que están delimitados

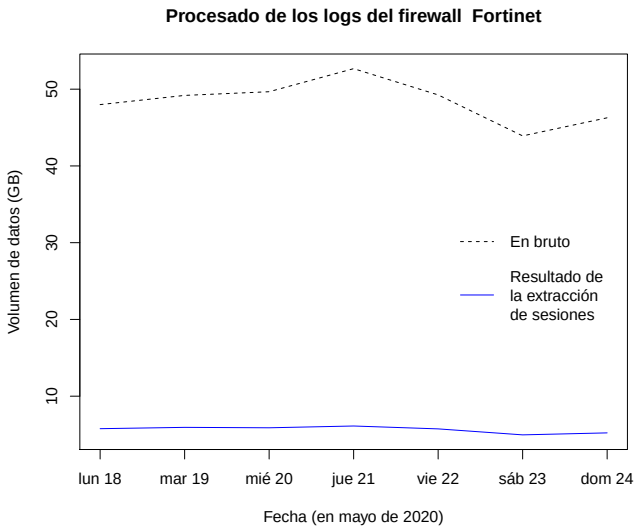

Figura 3: Volumen de logs en bruto y tras ser procesados, en una semana

por comas. A modo de ejemplo, las dos líneas adjuntadas a continuación corresponden a un evento del firewall Fortinet y otro del IPS PaloAlto, respectivamente (viene un evento por línea):

```
1585572524|1585572524|2020-03-30T06:48:44.202297|10.2.0.11|6|local7|
date=2020-03-30 time=06:48:44 devname="FW1_ICORP" devid="FG1800000"
logid="1059028704" type="utm" subtype="app-ctrl" eventtype="app-ctrl-all"
level="information" vd="root" eventtime=1585572524 appid=41470 user="NOM"
group="GrupoOffice365" authserver="SV1" srcip=172.2.9.6 dstip=23.203.51.72
srcport=54697 dstport=443 srcintf="p18" srcintfrole="undef" dstintf="p20"
dstintfrole="wan" proto=6 service="HTTPS" direction="outgoing" policyid=124
sessionid=325186437 applist="AC_CORREO" appcat="Collab" app="Microsoft.CDN"
action="pass" hostname="img-prod-cms-rt-microsoft-com.akamaized.net"
incidentserialno=1500000000 url="/" msg="Collaboration: Microsoft.CDN,"
apprisk="elevated" scertcname="a248.e.akamai.net"

1585659863|1585659863|2020-03-31T07:04:23.027791|10.2.0.73|6|local0|
1,2020/03/31 07:04:23,001801000000,TRAFFIC,end,2049,2020/03/31 07:04:03,
10.138.4.7,186.151.236.155,0.0.0.0,0.0.0.0,OUTBOUND,,,incomplete,vsys1,
trust,untrust,ethernet1/10,ethernet1/9,Log-Panorama,2020/03/31 07:04:03,
41602,1,55074,80,0,0,0x19,tcp,allow,132,132,0,2,2020/03/31 07:03:55,3,any,
0,1307298109,0x80000,10.0.0.0-10.255.255.255,America,0,2,0,aged-out
```

Otro hecho reseñable que afecta al formato es que se emplea *syslog* (Gerhards, 2009) (el estándar de facto) como protocolo para trasladar los datos desde cada equipo hasta el punto de recolección, de forma que se cuenta con ciertos campos adicionales a los enviados por los equipos. Para el tema que nos ocupa, los únicos campos que se extraen de esta cabecera son: la marca de tiempo en la que ha llegado cada evento, conocida en el vocabulario informático como *timestamp*, y lo que conoceremos como la prioridad del evento (que en la especificación de *syslog* se denomina severidad, pero se ha creído que el término ‘‘prioridad’’ es más acertado en este entorno). En cualquier caso, esta sección adicional dentro de los logs tiene también su propio formato, por lo cual también se deberá tratar de forma específica. En nuestra configuración (que aplica a la herramienta *rsyslog*[2]), la siguiente directiva establece cómo se vuelcan a fichero estos campos de syslog:

```
template(name="FORMATO_LOGS" type="string"
string="%timereported:::date-unixtimestamp%
    |%timegenerated:::date-unixtimestamp%
    |%timegenerated:::date-rfc3339%|%fromhost-ip%
    |%syslogseverity%|%syslogfacility-text%| %syslogtag%%msg%\n")
```

Así que, en los *scripts* que procesan los ficheros donde se han volcado los datos traídos mediante *syslog*, se obtiene la fecha de cada evento a partir de este primer campo ‘‘timereported:::date-unixtimestamp’’ y la prioridad a partir del quinto campo. Esta primera parte del procesado (que está programado en Python) se hace de la siguiente manera:

```
for syslogline in sys.stdin:
    try:
        splitted_syslogline = syslogline.rstrip().split("|")
        ↪  # .rstrip() removes last "\n" character
        tstamp_line = int(splitted_syslogline[0])
        prio = splitted_syslogline[4]
```

Seguidamente, el resto de la línea actual (sin la cabecera de *syslog*) se convierte en una estructura de diccionario. Como se apreciaba en las líneas de ejemplo que se han incluido antes, la relación entre claves y valores depende de cada caso.

Para el equipo Fortinet, la relación está definida en el propio evento como `clave="valor"` o `clave=valor` para valores no considerados como cadenas de texto. Cabe destacar que

[2] https://www.rsyslog.com/

el símbolo ‘‘=’’ puede estar contenido en el valor. El nombre de la clave, sin embargo, nunca llevará comillas. Establecidas las anteriores reglas, en Fortinet se convierten los campos con una expresión regular y una *dict comprehension*[3] (es decir, una forma concisa de crear diccionarios a través de la iteración sobre una lista con la posibilidad de incluir condicionales):

```
line = "".join(splitted_syslogline[6:])
↪  # removes "1581410810|1581410810|2020-02-11T02:46:51.421302|10.25.0.6|5|local7|"

fields = re.split("([^ \"]+=[^ \"]+)|([^ \"]+=\"[^\"]+\)", line)

dict_line = {k:v.strip('"')
             for k,v in [f.split("=", 1)
             for f in fields if (f and "=" in f)]}
```

Para el IPS PaloAlto, la extracción de los campos es más sencilla. Como cumplen con el formato CSV estandarizado (Shafranovich, 2005), los valores se tienen en una lista con solo leer la línea a través de una función de la librería `csv`. En cuanto a las claves, en la documentación[4] de PaloAlto se explica que depende del tipo del evento. Por tanto, se asignan unas claves u otras consultando primero de qué tipo se trata. Finalmente, se construye el diccionario con otra *dict comprehension*:

```
line = "".join(splitted_syslogline[6:])
↪  # removes "1581410810|1581410810|2020-02-11T02:46:51.421302|10.25.0.6|5|local7|"

values = list( csv.reader([line]) )[0]

this_type_keys = []

if values[3]=="TRAFFIC": # type is on 4th field
    this_type_keys.extend(common_trafficthreat_fields)
    this_type_keys.extend(traffic_fields)
elif values[3]=="THREAT":
    ...

if this_type_keys!=[]:
    dict_line = {k:v for k,v in zip(this_type_keys,values)}
```

Posteriormente se lleva a cabo otra serie de operaciones necesarias para la transformación de los datos de entrada en información útil para la monitorización. Sin embargo, desde la perspectiva de este trabajo, el único apartado de interés es la agregación de sesiones, que se desarrolla a continuación.

Una parte de este procesado consiste en guardar cierta información asociada a cada sesión. El concepto de ‘‘sesión’’ sería equivalente al de ‘‘flujo’’ presentado en el capítulo anterior: una serie de eventos asociados que se corresponden con la misma tupla de {IP

[3] https://www.python.org/dev/peps/pep-0274/

[4] https://docs.paloaltonetworks.com/pan-os/8-1/pan-os-admin/monitoring/use-syslog-for-monitoring/syslog-field-descriptions.html

origen, IP destino, protocolo, puerto origen, puerto destino}. Los dos equipos mantienen un campo ‘‘Identificador de Sesión’’ interno que se añade a la tupla de la sesión. Este campo permite distinguir los eventos de sesiones que coinciden en origen y destino pero se producen en intervalos temporales diferentes. Se incluye en el procesado con esta finalidad de no confundir sesiones distintas en etapas posteriores, pero para nuestro propósito de clasificación de equipos puede ignorarse.

La información de cada sesión se compone de:

- Tupla que define la sesión:
  {ID de sesión, IP origen, IP destino, protocolo, puerto origen, puerto destino}
- Timestamps del primer y último evento pertenecientes a esta sesión
- Máxima prioridad de evento vista en esta sesión
- Bytes recibidos y enviados (solo en el IPS)
- Nivel de anomalía
- Nivel de amenaza
- Contador y lista de eventos

Los niveles de anomalía y amenaza son unos índices simples que se han diseñado para resumir cualidades de interés acerca de la sesión, como son cuánto se aleja de la normalidad la cantidad de eventos prioritarios que se han visto y cómo son de peligrosas las amenazas recibidas. Se entiende como ‘‘evento’’ una entrada de log, que está relacionada con una sesión concreta. Puede comprobarse cómo es un evento en las líneas de logs del firewall Fortinet y del IPS PaloAlto que aparecían como ejemplos al principio de esta misma sección.

Para cada sesión, se calculan sus niveles sumando los n eventos que le corresponden, según las siguientes fórmulas:

$$N_{\text{anomalía}} = \sum_{i=1}^{n} \frac{1}{\text{prioridad}_{\text{evento}_i}}$$
$$\text{para eventos de prioridad } \leq 4 \text{ o eventos de tráfico que no son de inicio ni fin}$$

$$N_{\text{amenaza}} = \sum_{i=1}^{n} \frac{1}{\text{prioridad}_{\text{evento}_i}}$$
$$\text{para eventos de amenaza con prioridad } \leq 4 \text{ que no son bloqueados}$$

Tanto estas como el resto de características cuantificables pueden resultar de interés a la hora de aplicar *clustering* sobre un *dataset* derivado de este procesado.

La recolección de esta información se realiza a través de la función adjunta. Como puede verse, cuando se llama a esta función (lo cual ocurre ante todos los eventos de protocolo TCP o UDP) se actualizan los parámetros relativos a la sesión actual, que están almacenados en un diccionario. A su vez, este diccionario se encuentra dentro del diccionario `sessions`. Con él se mantienen en memoria todas las sesiones que todavía no se han cerrado. Cuando transcurre un *bucket* de tiempo determinado (por defecto, 60 segundos), se comprueban todas las sesiones vigentes. Aquellas que tienen un evento de finalización se imprimen en un fichero y se retiran del diccionario `sessions`. De este modo, cada 60 segundos se tienen los datos de nuevas sesiones completadas (en la práctica se alcanzan incluso más de 100 000 sesiones finalizadas cada minuto).

Esta serie de datos por sesión que produce la función `session_aggregation` se denominará "vector de características de la sesión", y el conjunto de estos vectores será el contenido en bruto a partir del cual se formará la entrada para el *clustering*.

```
def session_aggregation(dict_line, event_descript, event_tstamp):
    """
    It groups info related to actual session on the sessions dictionary, that is:
    - the sessionid and the session tuple (srcip-dstip-proto-srcport-dstport)
    - tstamp of first and last event observed for actual session
    - update max. priority of events observed for actual session
    - sent and received bytes for actual session
    - counter and list of events observed for actual session
    - recalculate anomaly level for actual session
    - recalculate threat level for actual session
    """

    session_tuple = "-".join([
            dict_line['SESSION ID'], dict_line['SRC_IP'], dict_line['DST_IP'],
            dict_line['PROTO'], dict_line['SRC_PORT'], dict_line['DST_PORT']
        ])

    priority = int(dict_line['priority'])

    if session_tuple not in sessions:
        sessions[session_tuple] = {}
        # store the session tuple values related to this new session_tuple:

        sessions[session_tuple]['events'] = []
        sessions[session_tuple]['anomaly_level'] = 0
        sessions[session_tuple]['threat_level'] = 0
        sessions[session_tuple]["counter"] = 0
        sessions[session_tuple]['max_prio'] = priority
        sessions[session_tuple]["bytes_sent"] = 0
        sessions[session_tuple]["bytes_rcvd"] = 0
        sessions[session_tuple]['first_event_tstamp'] = int(event_tstamp)

    sessions[session_tuple]['last_event_tstamp'] = int(event_tstamp)
    sessions[session_tuple]["counter"] += 1
```

```
    if dict_line['type']=="TRAFFIC":
        sessions[session_tuple]["bytes_sent"] += int(dict_line['BYTES_SENT'])
        sessions[session_tuple]["bytes_rcvd"] += int(dict_line['BYTES_RECEIVED'])

    if priority < sessions[session_tuple]['max_prio']:
    # lower prio value means more important (i.e., the most important priority is 1, or even 0 if priority=0 exists)
        sessions[session_tuple]['max_prio'] = priority
    else:
        sessions[session_tuple]['max_prio']

    if priority<=4 or (dict_line['type']=="TRAFFIC" and "end" not in event_descript
    ↪  and "start" not in event_descript):
        sessions[session_tuple]['anomaly_level'] += 1/priority
    if priority<=4 and dict_line['type']=="THREAT" and dict_line['ACTION']!="alert":
        sessions[session_tuple]['threat_level'] += 1/priority

    sessions[session_tuple]['events'].append("{}, {}".format(event_descript,
    ↪  dict_line['ACTION']))
```

4.3. Preprocesado para el clustering

A partir de los vectores de características de sesión descritos anteriormente, se debe obtener un *dataset* en el formato adecuado para poder aplicar algoritmos de *clustering* sobre él. Esta adecuación consistirá básicamente en resumir la información aplicando agregaciones y calculando métricas.

Ya se ha señalado que el volumen de datos es muy alto, así que como primera aproximación se extraerá una muestra suficientemente grande pero operable a priori. También se empezará por los datos de sesiones vistas en el Fortinet, ya que, al ser el equipo que vigila las conexiones desde la red corporativa a Internet, resulta de más interés y presumiblemente tendrá una actividad más relevante de cara a clasificarla.

La cantidad de sesiones diaria en el firewall Fortinet suele estar en torno a los 20 millones. Se practica un muestreo aleatorio simple que reduce los datos a un 5 %, de forma que el tamaño de la muestra (1 millón de sesiones) sea significativo para obtener unas primeras conclusiones pero su tratamiento no sea excesivamente costoso.

Los datos se guardan en un fichero por día, rotándose a diario y conservándose así (en texto plano, dispuestos para ser importados a una base de datos) durante 14 días antes de eliminarse. Mediante el siguiente comando, se copian 1 millón de vectores de características de sesión aleatorios, que están formados por:

{tstamp inicio, tstamp fin, IP origen, IP destino, protocolo, puerto origen, puerto destino, nivel de anomalía, nivel de amenaza, máxima prioridad, cantidad de eventos}

El comando se repite para los 7 ficheros de una semana:

```
$ shuf -n 1000000 FORTINET_INTERNET_CORP_10.251.0.101.1 | \
```

```
awk -F"···" '{print $1,$2,$5,$6,$7,$8,$9,$10,$11,$12,$13,$1-$2}' \
> datasets_clustering/FORTINET_INTERNET_CORP_10.251.0.101.23may
```

Y los vectores de características de sesión se vuelcan ordenados por *timestamp* de inicio en un mismo fichero, que es como se leerán para contar cuántas sesiones hay en cada fracción del día:

```
$ sort -n datasets_clustering/* > datasets_clustering/rawdata_7days
```

En las pruebas iniciales se descartaron las marcas de tiempo (campos `$1,$2`) por simplicidad, sabiendo que así se empezaría a trabajar con características más acotadas. Cuando ya se había validado el flujo de trabajo para aplicar los algoritmos con una cantidad limitada de características y se había adquirido práctica, se incorporaron las variables temporales. El campo `$1` es el *timestamp* de inicio de sesión, `$2` es el *timestamp* de fin de sesión, y con `$2-$1` se halla la duración de la misma (se incluye por comodidad para cálculos posteriores). La manera en la que se expresarían como características para el *clustering* requiere ser explicada con más profundidad en los párrafos sucesivos, donde se desarrollará cómo se han calculado los valores, cómo se ha realizado la agregación y por qué se ha escogido usar estas métricas.

El siguiente paso consiste en agrupar los vectores de características de sesión por IP de origen, que será la característica identificativa de cada *datapoint*. Para cada valor de la variable ‘‘IP de origen’’, se contabilizan cuántos valores distintos se observan en el resto de variables. Hay ciertas variables que no tiene sentido resumir mediante un recuento, como son: los protocolos usados, los niveles de amenaza y anomalía, la prioridad máxima y las características temporales.

A continuación se detallará lo que significa cada una de las 13 variables finales o características que se han usado. En la tabla 1, después de los siguientes párrafos de detalles, se sintetiza toda esta información.

Las cuatro primeras características cuentan el número de IPs destino, protocolos, puertos origen y puertos destino únicos que corresponden a una IP origen para cada uno de los 7 días (muestreados el 5 %) que conforman la colección de datos. Para distinguir si, en caso de usarse un solo protocolo de nivel de transporte, este es TCP o UDP, el número de protocolos se codifica como se indica: el valor de ‘‘protocolos usados’’ será ‘‘2’’ si la IP origen tiene sesiones tanto TCP como UDP, ‘‘1’’ si solo emplea UDP, y ‘‘0’’ si todas las sesiones son TCP.

Las dos características siguientes, que son los niveles medios de anomalía y amenaza, se hallan calculando la media de cada nivel en todo el periodo de tiempo bajo análisis (un día).

La característica ‘‘prioridad máxima’’ corresponde al valor de prioridad más bajo de un evento visto para esta IP origen. Nótese que los niveles de proridad en *syslog* (más

precisamente, lo que *syslog* llama ‘‘severidad’’) son: 0 el nivel de emergencia, 1 el nivel de alerta, 2 el nivel crítico, 3 error, 4 peligro, 5 aviso, 6 información y por último, como menos importante, 7 depuración. Por tanto, respetando la definición original de estos niveles de prioridad de los eventos, se debe tomar como prioridad de máxima importancia aquella prioridad de evento mínima de todas las vistas con esta IP origen.

La característica `count_events` es la suma de eventos que el firewall ha asociado a una misma IP origen.

Las últimas cinco características son las relativas al tiempo.

Primero se tienen dos métricas derivadas de la duración de las sesiones: ‘‘media de la duración de sesión’’ recoge cuántos segundos han durado de media las sesiones que ha mantenido una IP origen en el día en cuestión, mientras que ‘‘desviación estándar de la duración de sesión’’ cuantifica la variación de la duración de las sesiones respecto de su media. Con ellas se quiere caracterizar si un equipo (que corresponde a una IP origen) acostumbra a tener sesiones largas o cortas, y si esas duraciones son habituales o suelen variar mucho.

Las tres variables que cierran el conjunto de características son ‘‘número de sesiones activas en horas nocturnas / horas de trabajo / horas después del trabajo’’. En base a la hipótesis de que será relevante para la clasificación saber en qué momento del día se concentran las sesiones de un equipo, se ha definido una manera de contabilizar cuántas sesiones mantuvo activas una IP origen en cada fracción del día (se trabaja con 3 fracciones: horas nocturas de 00:00 a 08:00, horas laborales de 08:01 a 16:00 y horas después del trabajo de 16:01 a 23:59). Partiendo de todos los vectores de características de sesión ordenados cronológicamente por hora de inicio, para cada IP origen se incrementa el contador de cada fracción en la que una sesión haya permanecido activa (la colección de datos completa, de 7 días, se divide en 3 fracciones/día $\times$ 7 días $=$ 21 fracciones).

Las líneas de código que con las que se realiza este conteo son las siguientes:

```
week_tstamps = [i for i in range(1589752800,          1590357601, 60*60*8)]
                                #^^ L18may00:00; so L25may00:00 is included ^^; 8h step^^

slots = ['night', 'work', 'afterwork']
        # 00:00 - 08:00 - 16:00 - 00:00

t=0  # counter of actual element on week_tstamps

for line in raw_data:

    r = line.split()
    initial_tstamp = int(r[0])
    end_tstamp = int(r[1])
    src_ip = r[2]

    if initial_tstamp > week_tstamps[-1]:
```

```
        # so all week_tstamps have been covered:
            break

        if initial_tstamp > week_tstamps[t]:
        # so we move forward to the next week fraction:
            t+=1

        data[src_ip]['slots'][slots[ t%len(slots)-1 ]] += 1
        #example:
        # src_ip:10.212.138.53,from 1589807186(15:06:26) to 1589807191(15:06:31)
        #   -> {'night': 0, 'work': 1, 'afterwork': 0}
        # src_ip:10.212.138.53,from 1589920495(22:34:55) to 1589920504(22:35:04)
        #   -> {'night': 0, 'work': 1, 'afterwork': 1}

        i=t
        while end_tstamp > week_tstamps[i]:
        # long sessions count on every slots they are:
            data[src_ip]['slots'][slots[ i%len(slots)-1 ]] += 1
            i+=1
            if i==len(week_tstamps):
            # so end of the observed week has been reached:
                break
```

Como resultado de esta transformación (el código completo se adjunta en el apéndice E), se tiene una matriz con 13 columnas y *n* filas, siendo *n* el número de IPs de origen distintas presentes en la muestra de ese día. Cada valor de esta matriz resume las sesiones que un host (identificado por su IP de origen) ha mantenido, a través de las siguientes características o métricas calculadas sobre periodos de un día:

Característica	Explicación
Número de IPs destino únicas	IPs distintas a las que se ha conectado una IP origen
Protocolos usados	2 si IP origen ha usado TCP y UDP, 1 si UDP, 0 si TCP
Número de puertos origen únicos	Puertos de nivel transporte usados por una IP origen
Número de puertos destino únicos	Puertos a los que se ha conectado una IP origen
Nivel de anomalía medio	Media de $N_{\text{anomalía}}$ en las sesiones de una IP origen
Nivel de amenaza medio	Media de N_{amenaza} en las sesiones de una IP origen
Prioridad máxima	Prioridad más crítica vista en eventos de una IP origen
Número de eventos	Suma total de los eventos de una IP origen
Media de la duración de sesión	Duración media de las sesiones de una IP origen
Desv. estándar de la duración de sesión	Desv. est. de la duración de sesiones de una IP origen
Nº sesiones activas en horas nocturnas	Sesiones activas de 00:00 a 08:00
Nº sesiones activas en horas de trabajo	Sesiones activas de 08:01 a 16:00
Nº ses. activas en horas después del trabajo	Sesiones activas de 16:01 a 23:59

Tabla 1: Características con las que se resumen las sesiones en matrices diarias

Procediendo análogamente sobre los datos de cada día, obtenidos entre el lunes 25 de mayo de 2020 y el domingo 31, se tienen 7 matrices de 13 variables por unas 4800 filas (equivalentes al total de IPs origen distintas cada día), apreciándose un descenso de la cantidad de orígenes únicos en torno al fin de semana (4700 filas distintas el viernes y alrededor de 3000 filas tanto sábado como domingo). En total, se dispone de prácticamente 30 000 muestras. Con este material se puede abordar el prototipado del método de *clustering* sobre el que se fundamenta este proyecto.

4.4. Análisis de datos

Una vez obtenidas las características, se pasa a analizar la distribución de valores que presenta cada una de ellas en este *dataset*. Dicha tarea permitirá entender mejor la importancia relativa de cada característica y cómo afectará a los algoritmos de *clustering*. En concreto, se prestará especial atención a la forma, varianza y modalidades de las distribuciones.

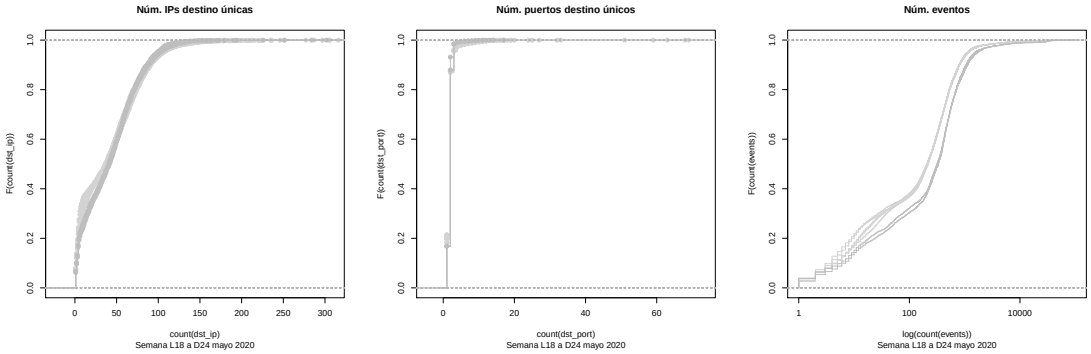

Figura 4: Funciones de distribución acumulada por número de IPs destino, puertos destino y eventos

La figura 4 muestra la función de distribución acumulada para varias características elegidas entre los datos disponibles. Se han superpuesto las siete líneas de los días de la semana extraída, de forma que se evidencia la estabilidad de estas distribuciones a lo largo de los días. Se representa cada variable sin normalizar, con lo que se obtiene una idea del rango de valores en el que se encuentra cada una. Cuando se pase a computar los *clusters*, se normalizarán para evitar que los atributos con escalas mayores dominen las distancias.

En el caso del número de eventos, se aprecia que es la distribución que más se acerca a una gaussiana, con una varianza un poco mayor en los días del fin de semana (son los que quedan por debajo porque tienen más varianza y una media algo superior, coloreados con un gris algo más oscuro). Pero, por lo general, las distribuciones que presenta el resto de características no son gaussianas y muestran asimetría positiva (los valores se concentran a la izquierda).

Por tanto, se espera que las colas de las distribuciones (es decir, los puntos con valores más altos) sean de más interés en el *clustering*. También se espera que los algoritmos sitúen a la gran mayoría de puntos dentro de unos pocos *clusters* bastante densos. Esto es una ventaja de cara al objetivo de detectar anomalías.

Como análisis preliminar al prototipado de métodos de *clustering* mediante la herramienta BigML[5], se visualizaron varias parejas de variables sobre un gráfico de dispersión, con la finalidad de adquirir una idea de sus dimensiones y cómo se relacionaban. Se aplicó también K-Means con varios valores de k, distinguiendo por colores los grupos formados, y se situaron los centroides en la gráfica (puntos rojos). Así se empezó a entender qué divisiones podían establecerse y dónde podrían encontrarse los umbrales, aunque de momento solo se estuvieran teniendo en cuenta subconjuntos reducidos de las características disponibles.

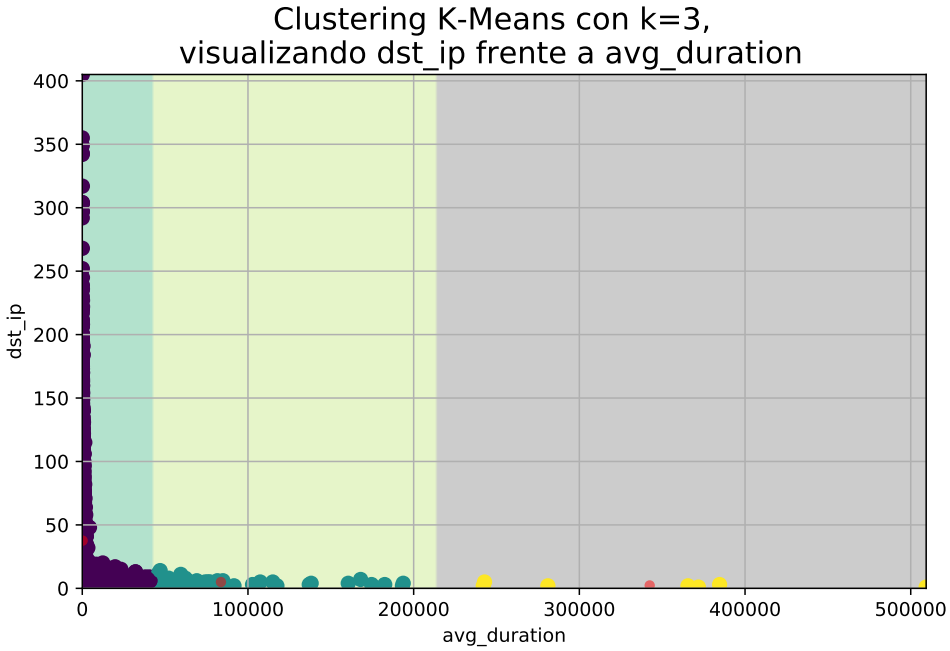

Figura 5: Gráfica de dispersión: número de IPs destino únicas frente a duración media, con K-Means ($k = 3$) y centroides (en rojo)

Las figuras 5 y 6 son una muestra de estas pruebas. En ellas, se aprecia cómo se reparten los puntos según los valores que toman en características como ‘‘número de IPs destino únicas’’, ‘‘duración media’’ o ‘‘número de eventos’’, así como los *clusters* formados en función del valor de k elegido. Se han elegido estas características porque presentan unos rangos muy distintos y los valores observados están distribuidos en varias zonas de dichos rangos, por lo que se diferencia una serie de grupos que pueden corresponder a algunos de los comportamientos de equipos informáticos en los que estamos interesados.

En la sección 4.6 se ampliará cómo se buscaron los valores de k con los que se trabajaría.

[5] https://bigml.com/

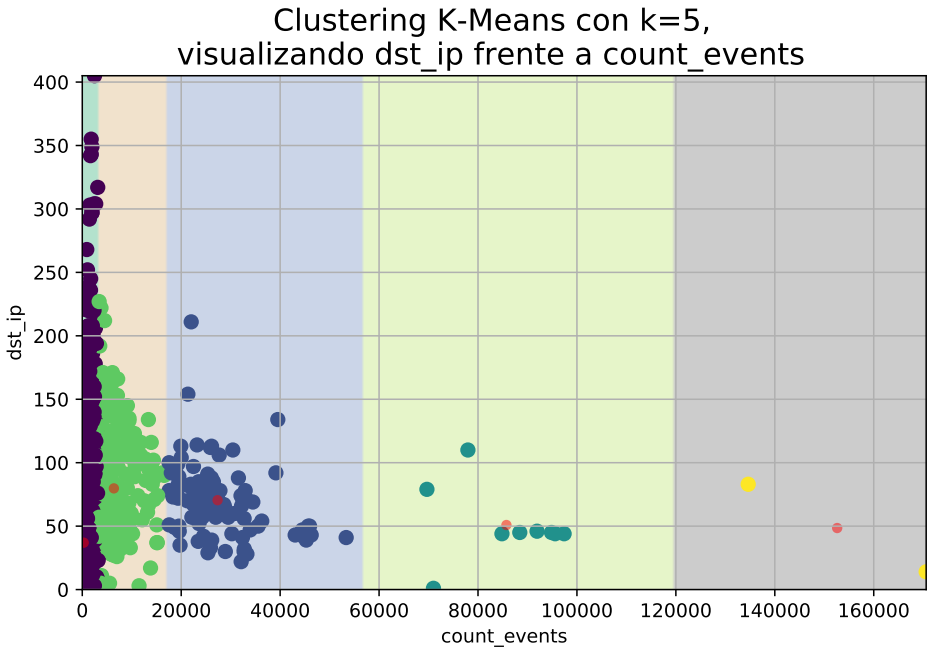

Figura 6: Gráfica de dispersión: número de IPs destino únicas frente a número de eventos, con K-Means ($k = 5$) y centroides (en rojo)

4.5. Selección de características

Como se apunta en ‘‘Análisis de las características en tráfico de red para detección de anomalías’’ (Iglesias y col., 2015), la meta que tiene la selección de características en la detección de anomalías es ‘‘eliminar características fuertemente correladas, redundantes e irrelevantes para mejorar la calidad de la detección’’. En este trabajo, los autores abordaron la selección de características de forma exhaustiva y rigurosa, mediante métodos multi-fase implementados con envolvedores (*wrappers*, que buscan el subconjunto de características con mejores resultados), combinando filtrado y técnicas de regresión gradual. Para nuestro caso, se ha optado por un procedimiento más sencillo, fundamentado en dos factores: la correlación entre características y la aportación de cada una al agrupamiento.

De acuerdo con la argumentación de (Bohara y col., 2016), si las características de un conjunto están altamente correladas, contendrán información redundante y provocarán un incremento innecesario de la complejidad del algoritmo. Para evitar esta redundancia, en dichos conjuntos fuertemente correlados se seleccionará una única característica. La correlación puede determinarse empleando el coeficiente de correlación de Pearson. Se decide que dos características se tomarán como fuertemente correladas si su coeficiente de correlación es mayor de 0,99. Ya que se tiene un *dataset* de tamaño bastante grande y el rango de valores de las características está acotado, se espera que el coeficiente de correlación de Pearson proporcione una estimación de la correlación suficientemente precisa.

El otro aspecto a considerar es la contribución de cada característica en la fase de *clustering*. Si, al analizar el vector de los pesos relativos que tiene cada característica sobre cada uno de los *clusters*, se aprecia que ciertas características tienen un papel poco activo en la clasificación, se simplificará el set de características eliminando aquellas que aportan débilmente al algoritmo.

Es importante distinguir la sutil diferencia entre selección de características y extracción de características. Ambos son métodos para reducir la dimensionalidad, pero el proceso que siguen es distinto. La selección de características descarta características irrelevantes o redundantes para los procesos posteriores de representación y clasificación de datos. En cambio, la extracción de características consiste en proyectar el *dataset* original en un nuevo espacio vectorial donde se minimice la dependencia lineal entre características, reduciendo por tanto el número de variables necesarias. Aunque estas técnicas (por ejemplo, el análisis de componentes principales o PCA) permiten ponderar la importancia de las características y por tanto seleccionarlas, debe destacarse que solo consideran relaciones lineales entre variables (ignorando cualquier otro tipo de interacción entre ellas). Por eso se puede afirmar que no es un método ideal para seleccionar características, si bien es cierto que pueden combinarse selección y extracción de características, eliminando características irrelevantes primero y proyectándolas en espacios optimizados después.

También se valora la observación de (Guyon y col., 2003) cuando dice que ‘‘el objetivo de la selección de variables es triple: mejorar el rendimiento de los predictores, posibilitar predictores más rápidos y eficientes en coste, y permitir una mejor comprensión del proceso subyacente que ha generado los datos’’. La selección de características aquí expuesta busca cumplir con estas tres finalidades.

Tras hacer tests de correlación entre todas las variables, no se ha encontrado ninguna pareja de variables con tanta correlación como para poder suprimir una de ellas. Sin embargo, sí se ha comprobado que puede reducirse el número de variables (y con ello el coste y complejidad del *clustering*) quedándose solo con las de mayor peso. Tras analizar la importancia relativa de cada una en los modelos generados tras el clustering (se adjuntan cifras concretas en los apéndices), se ha repetido el último ensayo (indicado en la subsección 4.7.4) manteniendo solo las 6 características más influyentes: ‘‘dst_ip’’, ‘‘proto’’, ‘‘src_port’’, ‘‘max_prio’’, ‘‘count_events’’ y ‘‘stdev_duration‘‘. El resultado ha sido que se lograban los mismos *clusters* y el mismo número de instancias en el *cluster* de anomalías, con un considerable decremento del tiempo de computación.

Contrariamente a lo que se esperaba, esto indica que las variables temporales no han sido

tan decisivas como se creía que serían.

4.6. Parametrización del algoritmo de clustering

Elegir un número k de *clusters* óptimo es crucial para obtener una clasificación útil. Si el número de *clusters* es insuficiente, las agrupaciones contendrán datos muy heterogéneos. Por el contrario, cuando el valor k es excesivo, se agruparán erróneamente datos que son muy similares unos a otros.

Aunque no existe un único criterio que seleccione objetivamente el valor k adecuado para cualquier caso, sí se conocen varios métodos que ayudan en esta tarea. Probablemente uno de los más extendidos sea el método del codo o *elbow method*.

Una de las medidas que se suelen utilizar para comparar resultados entre varios valores de k es la suma de cuadrados dentro de cada grupo (*within-cluster sums of squares*, abreviado como WCSS). Este concepto a veces también se conoce como inercia, y es una medida de variación o desviación con respecto a la media, en este caso representada por el centroide. La técnica de selección de k por el método del codo consiste en representar en una gráfica lineal la WCSS respecto del número de *clusters*. El punto para el cual se observe un cambio brusco en la evolución de la inercia indicará el valor óptimo de k.

La efectividad de este método está basada en que el algoritmo K-Means se fundamenta en el paradigma del análisis de la varianza (*analysis of variance* o ANOVA). Como la suma total de desviaciones al cuadrado respecto de los centroides es constante y está compuesta de la suma de cuadrados de las desviaciones inter-grupales más la de las desviaciones intra-grupales:

$$TSS = WCSS + BCSS$$

$$SS_{Total} = SS_{Within-cluster} + SS_{Between-clusters}$$

Si se minimiza la suma de cuadrados dentro de cada *cluster* (WCSS), se maximiza la distancia entre *clusters*. De esta forma, se logrará una separación en grupos adecuada, siempre que esta suma de cuadrados WCSS no sea tanta como para asociar puntos cercanos pero de distintas categorías.

En la figura 7 se muestra la gráfica lineal de WCSS respecto de k que se ha explicado, calculada para varios subconjuntos de características. Se puede observar que el valor de k que este método daría como óptimo es distinto según las variables consideradas, y no es inequívocamente claro.

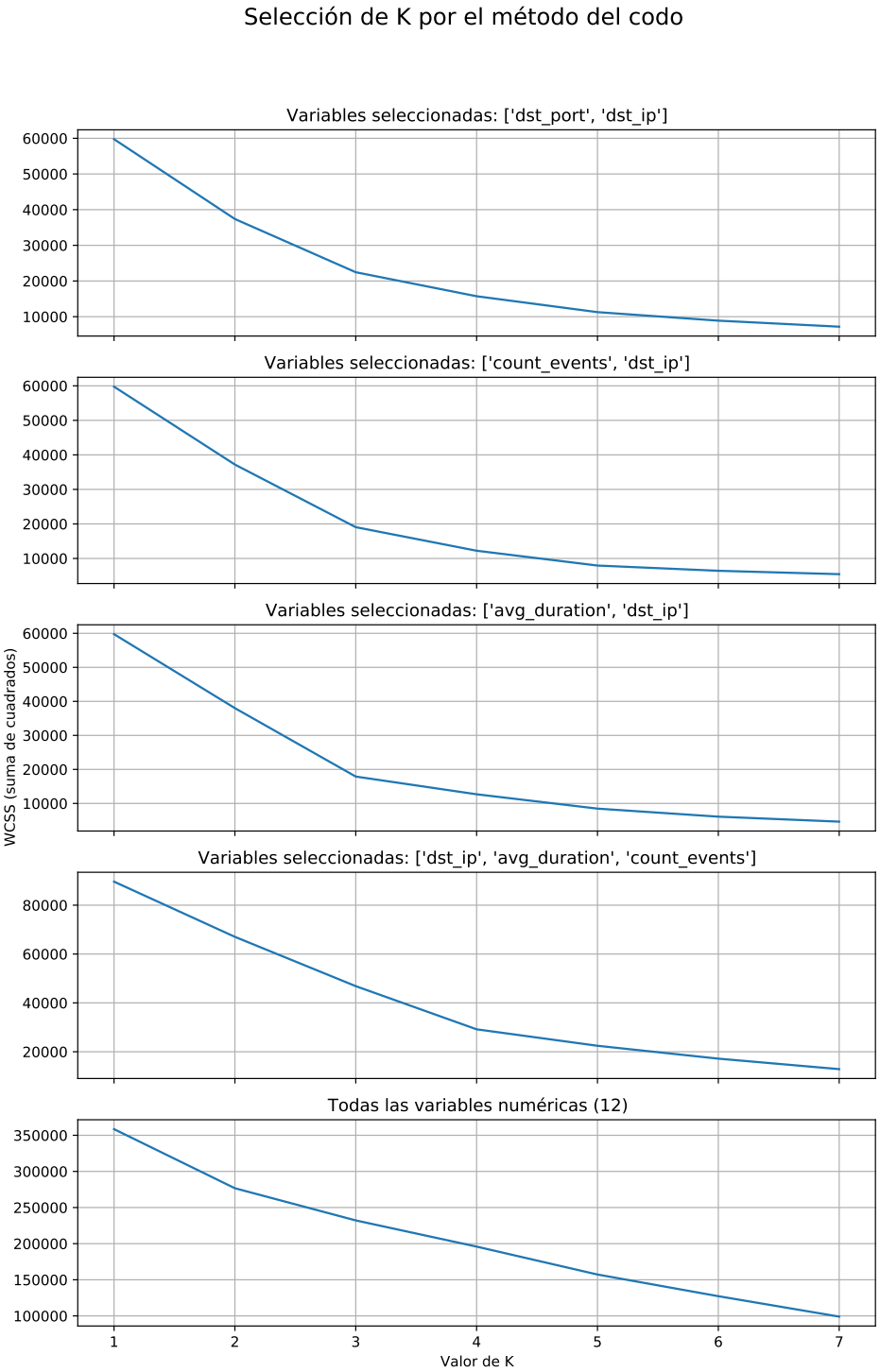

Figura 7: Gráficas de suma de cuadrados frente a valores de k en varias combinaciones seleccionadas de características

Tras las pruebas experimentales que se describen en la siguiente sección, se concluyó que el valor de k que arrojó mejores resultados empíricamente fue $k = 5$.

4.7. Evaluación experimental

Con la máxima de permitir un prototipado rápido y poder comparar los resultados de varias pruebas, se optó por emplear la herramienta online BigML para aplicar K-Means en los ensayos que se procede a explicar.

4.7.1. Ensayo A: K-Means con k=5, sin características temporales

Como se adelantaba en la sección 4.3, en las primeras pruebas no se incluyeron las marcas de tiempo. Se usaron las características anotadas como ‘‘dst_ip’’, ‘‘proto’’, ‘‘src_port’’, ‘‘dst_port’’, ‘‘anom_level’’, ‘‘max_prio’’, ‘‘count_events’’ y ‘‘avg_duration’’, junto con ‘‘src_ip’’ como característica identificativa. Después se advirtió que usar únicamente la media para resumir la información sobre la duración de las sesiones de un equipo durante un día parecía impreciso, así que se incluyó la característica ‘‘stdev_duration’’.

Inicialmente se eligieron valores $k = 3$ y $k = 4$, pero las divisiones obtenidas resultaron de poca utilidad desde el punto de vista de la caracterización de comportamientos. Los *clusters* principales agrupaban demasiadas instancias sin un criterio fácil de deducir, y el *cluster* minoritario contenía únicamente algunas instancias con valores extremos.

Fijando el valor de $k = 5$ se mejoró sensiblemente la interpretabilidad de los *clusters* formados, porque sus centroides pudieron asociarse a patrones de comportamiento más claros.

Las categorías identificadas en los *clusters* formados durante este ensayo fueron:

1. Comportamiento normal, muchas conexiones (54,34 % de las instancias)
2. Comportamiento normal, pocas conexiones (30,96 %)
3. Sesiones UDP (14,12 %)
4. Nivel de anomalía alto (0,45 %)
5. Conexiones largas (0,14 %)

En el apéndice A se pueden consultar los detalles de la salida que produjo la aplicación de K-Means con $k = 5$ en estas condiciones. Se adjuntan datos de interés como los valores de las 10 características con los que se definía el centroide de cada *cluster*. Se incluyen también los modelos de cada *cluster* que ofrece BigML, definidos mediante algunas reglas de clasificación. Si se revisan las reglas generadas automáticamente para este ensayo y otros, se encontrará que a veces consideran características que no tienen mucho sentido. Por eso las

reglas pueden ser útiles para establecer ciertos umbrales, pero no deben seguirse a ciegas, independientemente de sus niveles de confianza y soporte.

Además, en el apéndice se listan las características que han contribuido a la formación de cada *cluster* ordenadas por influencia. Esta última información resulta muy interesante a la hora de comprender y optimizar la manera en la que se ha alcanzado esta clasificación no supervisada. Servirá también de realimentacion para mejorar la selección de características comentada en la sección 4.5.

En unas pruebas adicionales, se intentó incluir una dimensión temporal mediante la característica ‘‘active_hours_vector’’. Esta contenía un vector con 24 valores que representaban el número de sesiones que tenía una IP origen en cada hora del día. Se trató primero como una característica categórica (interpretando estos valores como items en vez de valores numéricos) y luego como 24 características numéricas independientes, pero no se tuvieron buenos resultados. Por eso se replanteó el preprocesado de estos datos temporales y finalmente se fijó el conjunto de características de la tabla 1.

4.7.2. Ensayo B: G-Means determina k=3

Con el *dataset* final, se decidió probar el algoritmo G-means (*Gaussian-means*) que ofrece BigML. Este algoritmo descubre el número k de *clusters* automáticamente, usando un test estadístico para decidir si dividir un centroide en dos. El algoritmo procede jerárquicamente, fijándose en un subconjunto de datos, y en cada iteración testea la hipótesis de que el subconjunto sigue una distribución gaussiana. Si no la sigue, divide el *cluster*.

Este algoritmo puede ser útil si no se tiene una estimación de cuál puede ser el valor de k adecuado, presuponiendo unos datos de distribución normal. Sin embargo, se comprobó que el algoritmo no era apropiado para este caso, ya que los 3 *clusters* resultantes no eran significativos para la aplicación en cuestión. Además, era de esperar que no funcionara bien con estos datos, porque en la sección 4.4 se había visto que la distribución de la mayoría de las variables no coincidía con una gaussiana.

Se adjuntan los resultados de esta prueba en el apéndice B, por constatar lo que se ha descrito aquí.

4.7.3. Ensayo C: K-Means con k=5, fracciones temporales con agrupación semanal

Se pasó entonces al *clustering* usando K-Means con $k = 5$ sobre el *dataset* final. En primer lugar se agrupó toda la información en un solo vector de características de la sesión,

sin separar por días. El informe completo del resultado está en el apéndice C, pero puede resumirse con que se lograron unos *clusters* ya bastante similares a los que se ampliarán en la próxima subsección y en el capítulo 5 de resultados. Dichos resultados son los que se tomaron como los mejores.

No obstante, el *cluster* de anomalías se componía de solo 3 instancias que, según revelaba el resumen del modelo de este *cluster*, se habían agrupado únicamente por sus valores altos en la característica ‘‘duración media’’. Se consideró que la condensación de información en el preprocesado (los datos de toda la semana se expresaban con un único punto) estaba siendo demasiada.

Además, el ratio de sumas de cuadrados ($SS_{Between-clusters}/SS_{Total}$) era de 0.3358, un indicador de que sería conveniente mejorar las propiedades de cohesión interna y separación externa entre *clusters*. En términos estadísticos, esto se consigue incrementando la suma de cuadrados inter-grupales y reduciendo la suma de cuadrados intra-grupales, hecho que se dio al hacer una agrupación diaria en el preprocesado.

4.7.4. Ensayo D: K-Means con k=5, fracciones temporales con agrupación diaria

En el último ensayo se aplicó K-Means con $k = 5$ por separado a los 7 *datasets* que se tenían cuando se hizo agrupación diaria sobre los datos de una semana. Los resultados (completos en el apéndice D) eran más claros y útiles que los de ensayos anteriores.

Las categorías finales en las que se clasificó la actividad de los equipos de la red en cada día fueron:

1. Comportamiento normal, muchas conexiones (de media 54,07 % de las instancias)
2. Comportamiento normal, pocas conexiones (de media 30,34 %)
3. Sesiones UDP (de media 8,05 %)
4. Conexiones largas o muchos puertos origen (de media 7,42 %)
5. Anomalías: muchos eventos y conexiones (0,30 %)

En el siguiente capítulo se entra de lleno a explicar los resultados obtenidos en este ensayo final.

Capítulo 5

Resultados

El penúltimo capítulo se dedica a analizar los resultados obtenidos en la iteración final de las adelantadas en el capítulo 4 ‘‘Desarrollo’’. Se ahonda en aspectos como las proporciones y composición de los *clusters*, sus centroides, métricas de bondad y en definitiva se explica qué significan estos resultados para este trabajo.

5.1. Composición de los clusters

Como se adelantaba en la página anterior, más del 80 % de los equipos de la red se clasifican como comportamientos totalmente normales (subdivididos en dos categorías: normal con muchas conexiones distintas y normal con pocas conexiones). Algo menos del 10 % se agrupan en una tercera categoría porque el tráfico de los equipos que la forman es mayoritariamente UDP (lo que implica un comportamiento de características distintas pero no anómalo). En la cuarta categoría por orden de frecuencia (que supone un 5-10 % de las instancias) se ha visto otro tipo de tráfico que destaca en unas ocasiones por la larga duración de sus conexiones y en otras ocasiones por el alto número de puertos origen (que indica muchas conexiones). Y, finalmente, se tiene un último grupo reducido (menos del 1 %) con casos notablemente anómalos, porque presentan un número de eventos y de puertos destino únicos muy superior al resto.

Esta proporción en la que se han separado los *clusters* es una ventaja, ya que ayuda a centrar la atención en unos pocos casos anómalos que pueden revisarse manualmente. En la escala del *dataset* obtenido para este trabajo, esa incidencia menor del 1 % para el grupo muy anómalo supuso siempre menos de 10 casos diarios que pudieron estudiarse individualmente y trasladarse al responsable de la administración de los mismos.

Los demás porcentajes se han mantenido bastante estables a lo largo de los días, así que

también será interesante vigilar variaciones respecto a esta estabilidad, como síntoma de un hipotético cambio de tendencia generalizado en el comportamiento de los equipos de la red.

Para permitir examinar los valores específicos que han tomado los centroides en las 8 características más relevantes durante los 7 días, se adjuntan las siguientes tablas de la 2 (primer día) a la 8 (séptimo día). Téngase en cuenta que la diferencia de los centroides del primer día respecto al del resto puede explicarse por cuestiones relacionadas con la extracción de los datos, que incluían sesiones iniciadas varios días atrás, y cómo se han tratado estos datos cortados. En un procesamiento continuo, este problema no sucedería.

Cluster	dst_ip	proto	src_port	dst_port	max_prio	count_events	avg_duration	stdev_duration
normal, muchas cnxs	6.02	0.00	11.60	1.66	4	37.34	2.81e+04	6.25e+04
normal, pocas cnxs	2.81	0.02	3.69	1.32	5	16.01	6.69e+04	3.46e+04
udp	14.37	0.91	30.72	2.09	4.19	93.90	8.60e+04	2.66e+05
cnxs largas	7.70	0.06	12.92	2.25	4.19	1595.12	7.2e+06	1.40e+07
anom(2)	24.65	0.49	2191	2	4.51	86121.10	8.51e+05	3.71e+06

Tabla 2: Valores de los centroides en el día 1

Cluster	dst_ip	proto	src_port	dst_port	max_prio	count_events	avg_duration	stdev_duration
normal, muchas cnxs	43.84	0.04	112.67	2.04	4	296.99	14157.92	54004.72
normal, pocas cnxs	5.11	0.03	9.84	1.44	5	20.30	19812.79	20215.99
udp	43.60	1.72	137.24	4.11	4.16	359.39	65984.58	187667.52
muchas dst_ip	97.34	0.16	492.50	2.23	4	1239.55	10575.59	66178.72
anom(16)	68.50	0.31	7324.59	2.18	4.05	38265.85	4469.58	99635.59

Tabla 3: Valores de los centroides en el día 2

Clusters	dst_ip	proto	src_port	dst_port	max_prio	count_events	avg_duration	stdev_duration
normal, muchas cnxs	58.61	0.07	257.97	2.06	4	697.03	34366.40	74.43
normal, pocas cnxs	4.80	0.04	9.98	1.41	5	18.16	7914.26	1.78
udp	52.93	1.64	171.56	3.87	4.11	400.45	65689.04	44.95
cnxs largas	33.25	0.21	86.52	2.13	4.38	2015.05	722651.46	23.28
anom(2)	46.50	0.50	15709.50	3	4.50	68211.50	1	21119.50

Tabla 4: Valores de los centroides en el día 3

5.2. Calidad del resultado

Existen diversas métricas de bondad para conocer de forma objetiva la calidad del *clustering*, aparte de lo útil que resulte para la aplicación en cuestión. Aquí se comentan algunas de ellas.

Se mencionaba antes que el ratio de sumas de cuadrados, indicador del equilibrio entre cohesión interna y separación entre *clusters*, era una de las métricas de bondad vigiladas durante los ensayos. En el resultado final, con agregación diaria, este ratio se incrementó hasta

Cluster	dst_ip	proto	src_port	dst_port	max_prio	count_events	avg_duration	stdev_duration
normal, muchas cnxs	57.97	0.28	253.07	2.13	4	679.49	10235.43	36196.79
normal, pocas cnxs	6.02	0.10	13.55	1.44	4.99	23.83	7345.21	9476.48
udp	7.14	1.39	130.93	10.94	4.19	264.87	20494.34	75272.26
cnxs largas	40.46	0.36	114.77	2.30	4.26	489.55	249823.38	617650.08
anom(1)	45	2	16152	4	4	94864	10	1164

Tabla 5: Valores de los centroides en el día 4

Cluster	dst_ip	proto	src_port	dst_port	max_prio	count_events	avg_duration	stdev_duration
normal, muchas cnxs	56.84	0.33	202.58	2.17	4	523.68	1.02e+04	32559.60
normal, pocas cnxs	6.21	0.08	12.88	1.50	5	21.53	8.96e+03	7923.57
udp	26.92	0.81	107.49	5.37	4.23	408.55	1.01e+05	365880.91
muchos src_ports	64.25	0.01	7209.30	2	4.13	26836.11	6.43e+02	7962.58
anom(5)	1.20	0	1.20	1.01	5	141.85	1.92e+06	338833.03

Tabla 6: Valores de los centroides en el día 5

Cluster	dst_ip	proto	src_port	dst_port	max_prio	count_events	avg_duration	stdev_duration
normal, muchas cnxs	55.42	0.01	227.65	2.10	3.99	610.90	22825.61	97734.84
normal, pocas cnxs	6.30	0.02	13.40	1.48	5	22.57	17373.38	15749.55
udp	53.53	1.36	169.91	2.48	4.02	497.80	18522.75	70955.22
cnxs largas	5.78	1.38	115.85	11.09	4.20	256.49	24985.20	92374.05
anom(19)	75.75	0.06	7933.31	2.18	4.06	30504.37	856	25463

Tabla 7: Valores de los centroides en el día 6

Cluster	dst_ip	proto	src_port	dst_port	max_prio	count_events	avg_duration	stdev_duration
normal, muchas cnxs	54.51	0.04	257.24	2.02	4	660.99	9426.71	33187.10
normal, pocas cnxs	5.54	0.01	17.40	1.38	4.99	34.45	8385.74	9746.59
udp	50.07	1.60	234.74	2.48	4.11	602.45	16391.64	57167.76
cnxs largas	6.15	0.66	65.48	5.41	4.48	367.28	503875.35	735594.61
anom(28)	68.48	0.07	8785.88	2.07	4.03	35278.18	771.92	6427.07

Tabla 8: Valores de los centroides en el día 7

alcanzar el 0.4133 en el peor día y 0.5064 en el mejor. Esto es señal de que se ha logrado una separación en grupos equilibrada.

Otro indicador estudiado ha sido el coeficiente de silueta, un valor diseñado en (Rousseeuw, 1987) y ampliamente usado para interpretar y validar la coherencia del *clustering*. En el artículo donde se especificó también se propuso una representación gráfica que simboliza lo bien que se ha clasificado cada instancia.

El coeficiente de silueta medio calculado para el resultado final es de 0.476, un índice perfectamente aceptable. En la figura 8 se observa cuánta similitud guarda una instancia con su *cluster* en comparación con el resto de *clusters*. Los posibles valores de este índice van de -1 a +1, por lo que se aprecia en la gráfica que la disposición de los *clusters* es apropiada. En especial, las instancias del *cluster* etiquetado con el 2 (el de las anomalías, más difícil de ver en la figura porque tiene muchas menos instancias) están claramente separadas del resto.

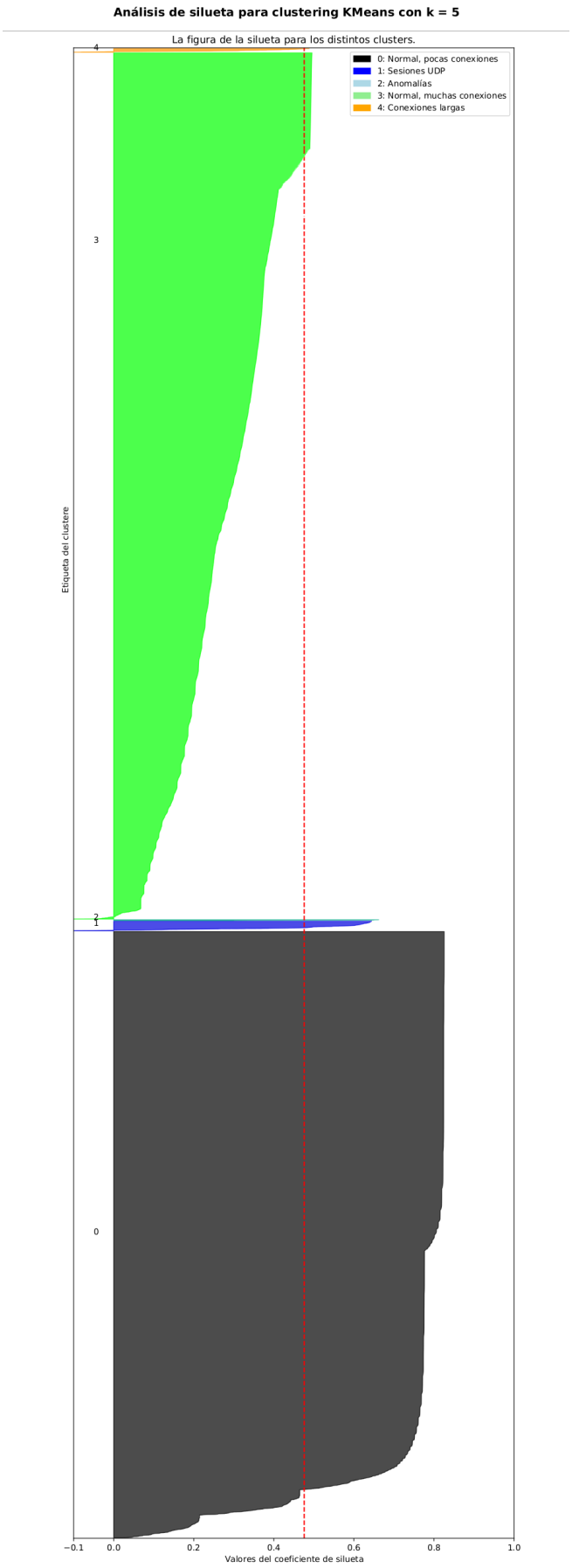

Figura 8: Representación gráfica del análisis de silueta para los *clusters* obtenidos con $k = 5$

Capítulo 6

Conclusiones y líneas futuras

Para acabar, en este capítulo se recoge el resumen final del problema tratado, cómo se ha abordado y qué respalda la validez de esta solución. Esta síntesis se enfoca en informar del alcance y relevancia de la aportación. Finalmente, se señalan las perspectivas de futuro que abre el trabajo desarrollado y cómo puede emplearse en el campo de la monitorización de redes empresariales.

6.1. Conclusiones

Se ha logrado modelar el comportamiento de los equipos informáticos en una red corporativa mediante métodos de *clustering*, descubriendo de forma no supervisada varias clases de equipos que cursan tráfico normal, y se han detectado de manera automática algunas IP origen anómalas por su cantidad de conexiones y eventos. Esto amplía el alcance de la monitorización que se mantenía hasta ahora, con nuevas capacidades.

Por tanto, puede decirse que esta aportación tendrá una aplicación práctica inmediata. A juzgar por el poco coste de computación y los satisfactorios indicadores de calidad medidos, el conjunto de procesados diseñados para este sistema podrá integrarse en los sistemas actuales de monitorización, programando su ejecución para que ofrezca las direcciones IP *clusterizadas* cada día (especialmente el de las anomalías más destacadas) y un analista pueda revisarlas.

Sin embargo, aunque los ensayos se han hecho sobre una cantidad suficiente de datos como para dar validez a las conclusiones extraídas, cabría esperar que su funcionamiento en producción requiera de más pruebas en tiempo real y sobre periodos de datos más prolongados, para perfeccionarlo y comprobar que los resultados alcanzan la calidad esperada.

6.2. Líneas futuras

En el entorno donde se ha desarrollado este proyecto se ha considerado oportuno dar continuidad a la investigación. Esta podría dirigirse en los siguientes sentidos:

- Incluir otros firewalls como fuente de datos, que enriquezcan la base de la que se parte (además, quizás podrían correlarse y con ello resultar mucho más valiosos).
- Trabajar sobre el seguimiento de los cambios de tendencia en los *clusters* normales e incluso procesarlos con un segundo método de *clustering*, que permita más granularidad a la hora de distinguir tipos de acciones habituales y descubra comportamientos particulares más sutiles.
- Aplicar preprocesados y técnicas de *clustering* similares a conexiones externas, para caracterizar el comportamiento de direcciones IP que se conecten a servicios públicos de la red de una empresa.
- Enfocarse en el diseño de una forma de visualización adecuada para que el analista pueda comprender y revisar los resultados más fácilmente.

Bibliografía

Herbert, S (1983). «Why Should Machines Learn?» En: *Machine Learning: An Artificial Intelligence Approach*. Springer Berlin Heidelberg, págs. 25-37. isbn: 978-3-662-12405-5. doi: 10.1007/978-3-662-12405-5_2.

Rousseeuw, Peter J. (1987). «Silhouettes: A graphical aid to the interpretation and validation of cluster analysis». En: *Journal of Computational and Applied Mathematics* 20, págs. 53-65. issn: 0377-0427. doi: https://doi.org/10.1016/0377-0427(87)90125-7.

Javitz, H. y col. (1993). «Next Generation Intrusion Detection Expert System (NIDES) Statistical Algorithms Rationale and Rationale for Proposed Resolver». En: doi: 10.13140/RG.2.1.1847.9521.

Portnoy, L. (2000). «Intrusion Detection with Unlabeled Data Using Clustering». En: doi: 10.7916/D8MP5904.

Guyon, I. y A. Elisseeff (2003). «An Introduction to Variable and Feature Selection». En: *J. Mach. Learn. Res.* 3, págs. 1157-1182. doi: 10.5555/944919.944968.

McGregor, A. y col. (2004). «Flow Clustering Using Machine Learning Techniques». En: *Passive and Active Network Measurement* 3015, págs. 205-214. doi: 10.1007/978-3-540-24668-8_21.

Karagiannis, T., K. Papagiannaki y M. Faloutsos (2005). «BLINC: Multilevel Traffic Classification in the Dark». En: *SIGCOMM Computer Communications Review 35.4*, págs. 229-240. doi: 10.1145/1090191.1080119.

Lazarevic, A., V. Kumar y J. Srivastava (2005). «Intrusion Detection: A Survey». En: vol. 5, págs. 19-78. doi: 10.1007/0-387-24230-9_2.

Leung, K. y C. Leckie (2005). «Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters». En: págs. 333-342. isbn: 1920682201.

Shafranovich, Y. (2005). *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. RFC 4180. http://www.rfc-editor.org/rfc/rfc4180.txt.

Zander, S., T. Nguyen y G. Armitage (2005). «Automated traffic classification and application identification using machine learning». En: págs. 250-257. isbn: 0-7695-2421-4. doi: 10.1109/LCN.2005.35.

Bernaille, L., R. Teixeira y K. Salamatian (2006a). «Early Application Identification». En: *Proceedings of the 2006 ACM CoNEXT Conference*, págs. 1-12. isbn: 1595934561. doi: 10.1145/1368436.1368445.

Bernaille, L. y col. (2006b). «Traffic Classification on the Fly». En: *SIGCOMM Comput. Commun. Rev.* 36.2, págs. 23-26. doi: 10.1145/1129582.1129589.

Jiang, S. y col. (2006). «A clustering-based method for unsupervised intrusion detections». En: *Pattern Recognition Letters* 27, págs. 802-810. doi: 10.1016/j.patrec.2005.11.007.

Bernaille, L. y R. Teixeira (2007). «Early recognition of encrypted applications». En: págs. 165-175. doi: 10.1007/978-3-540-71617-4_17.

Nguyen, H.T. y G. Armitage (2008). «A survey of techniques for internet traffic classification using machine learning». En: *IEEE Communications Surveys & Tutorials* 10.4, págs. 56-76. doi: 10.1109/SURV.2008.080406.

García-Teodoro, P. y col. (2009). «Anomaly-based network intrusion detection: Techniques, systems and challenges». En: *Computers & Security* 28, págs. 18-28. doi: 10.1016/j.cose.2008.08.003.

Gerhards, R. (2009). *The Syslog Protocol*. RFC 5424. http://www.rfc-editor.org/rfc/rfc5424.txt.

Lu, W. y T. Hengjian (2009). «Detecting Network Anomalies Using CUSUM and EM Clustering». En: págs. 297-308. doi: 10.1007/978-3-642-04843-2_32.

Orair, Gustavo H. y col. (2010). «Distance-Based Outlier Detection: Consolidation and Renewed Bearing». En: *Proc. VLDB Endow.* 3.1–2. doi: 10.14778/1920841.1921021.

Velmurugan, T. y T. Santhanam (2010). «Computational Complexity between K-Means and K-Medoids Clustering Algorithms for Normal and Uniform Distributions of Data Points». En: *Journal of Computer Science* 6, págs. 363-368. doi: 10.3844/jcssp.2010.363.368.

Finamore, A. y col. (2011). «Experiences of Internet traffic monitoring with tstat». En: *IEEE Network* 25.3, págs. 8-14. doi: 10.1109/MNET.2011.5772055.

Dainotti, A., A. Pescape y K. C. Claffy (2012). «Issues and future directions in traffic classification». En: *IEEE Network* 26.1, págs. 35-40. doi: 10.1109/MNET.2012.6135854.

Syarif, I., A. Prugel-Bennett y G. Wills (2012). «Unsupervised clustering approach for network anomaly detection». En: doi: 10.1007/978-3-642-30507-8_7.

Bhuyan, M. H., D. K. Bhattacharyya y J. K. Kalita (2014). «Network Anomaly Detection: Methods, Systems and Tools». En: *IEEE Communications Surveys & Tutorials* 16.1, págs. 303-336. doi: 10.1109/SURV.2013.052213.00046.

Iglesias, F. y T. Zseby (2015). «Analysis of network traffic features for anomaly detection». En: *Machine Learning*, págs. 59-84. doi: 10.1007/s10994-014-5473-9.

Bohara, A., U. Thakore y W. Sanders (2016). «Intrusion Detection in Enterprise Systems by Combining and Clustering Diverse Monitor Data». En: *Proceedings of the Symposium and Bootcamp on the Science of Security*, págs. 7-16. isbn: 9781450342773. doi: 10.1145/2898375.2898400.

Boutaba, R. y col. (2018). «A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities». En: *Journal of Internet Services and Applications* 9. doi: 10.1186/s13174-018-0087-2.

D'Alconzo, A. y col. (2019). «A Survey on Big Data for Network Traffic Monitoring and Analysis». En: *IEEE Transactions on Network and Service Management* 16.3, págs. 800-813. doi: 10.1109/TNSM.2019.2933358.

Apéndices

Apéndice A

Datos de salida en Ensayo A: K-Means con k=5, sin características temporales

Resultados de aplicar K-Means sobre el *dataset* con la variable categórica (no numérica) "active_hours_vector" y con $k = 5$.

```
K-means Cluster (k=5) with 5 centroids

Data distribution:
    Global: 100 % (29968 instances)
    normal, muchas cnxs: 54.34 % (16285 instances)
    normal, pocas cnxs: 30.96 % (9277 instances)
    udp: 14.12 % (4230 instances)
    muchos src_port, alto anom_level: 0.45 % (134 instances)
    cnxs largas: 0.14 % (42 instances)

Cluster metrics:
    total_ss (Total sum of squares): 6045.737600
    within_ss (Total within-cluster sum of the sum of squares): 3522.316780
    between_ss (Between sum of squares): 2523.420820
    ratio_ss (Ratio of sum of squares): 0.417390

Models:
    normal, muchas cnxs:
        Field importance:
            1. max_prio: 51.04 %
            2. proto: 27.28 %
            3. dst_ip: 11.07 %
            4. dst_port: 3.93 %
            5. src_port: 3.89 %
            6. avg_duration: 1.53 %
            7. active_hours_vector: 0.47 %
            8. stdev_duration: 0.41 %
            9. count_events: 0.26 %
            10. anom_level: 0.10 %
        Rules summary:
        false: (data 45.66 % / prediction 45.66 %)
            - 65.91 %: max_prio > 4 and src_port <= 290 [Confidence: 99.96 %]
            - 25.52 %: max_prio <= 4 and proto > 1 and dst_ip <= 107 and src_port <= 399 [Confidence: 99.89 %]
        true: (data 54.34 % / prediction 54.34 %)
```

```
        - 95.00 %: max_prio <= 4 and proto <= 1 and dst_ip > 9 and src_port <= 5766 and avg_duration <= 253327 and dst_port
   > 1 and stdev_duration <= 4566850 [Confidence: 99.98 %]

normal, pocas cnxs:
    Field importance:
        1. max_prio: 70.15 %
        2. dst_ip: 12.62 %
        3. dst_port: 7.09 %
        4. proto: 6.60 %
        5. count_events: 2.14 %
        6. active_hours_vector: 0.46 %
        7. avg_duration: 0.26 %
        8. stdev_duration: 0.23 %
        9. src_port: 0.22 %
        10. anom_level: 0.22 %
    Rules summary:
    false: (data 69.04 % / prediction 69.04 %)
        - 91.40 %: max_prio <= 4 and dst_ip > 12 [Confidence: 99.98 %]
        - 4.26 %: max_prio <= 4 and 5 < dst_ip <= 12  and dst_port > 1 [Confidence: 99.57 %]
    true: (data 30.96 % / prediction 30.96 %)
        - 92.93 %: max_prio > 4 and proto <= 1 and count_events <= 528 and dst_port <= 7 and avg_duration <= 751550 and
  dst_ip <= 81 [Confidence: 99.96 %]
        - 3.90 %: max_prio <= 4 and dst_ip <= 11 and dst_port <= 1 and proto <= 1 and count_events <= 246 and anom_level <=
   187 and src_port <= 23 [Confidence: 98.95 %]

udp:
    Field importance:
        1. proto: 76.42 %
        2. dst_ip: 7.38 %
        3. dst_port: 5.28 %
        4. max_prio: 3.35 %
        5. src_port: 2.07 %
        6. count_events: 1.82 %
        7. avg_duration: 1.68 %
        8. active_hours_vector: 1.33 %
        9. stdev_duration: 0.54 %
        10. anom_level: 0.12 %
    Rules summary:
    false: (data 85.88 % / prediction 85.88 %)
        - 98.19 %: proto <= 1 and dst_port <= 4 and avg_duration <= 148445 and stdev_duration <= 3692329 [Confidence:
  99.99 %]
    true: (data 14.12 % / prediction 14.12 %)
        - 82.51 %: proto > 1 and dst_ip <= 107 and max_prio <= 4 and src_port <= 399 and count_events <= 37170 and
  avg_duration <= 578095 [Confidence: 99.89 %]
        - 5.51 %: proto > 1 and 3 < dst_ip <= 107  and max_prio > 4 and dst_port > 1 and count_events > 18 and avg_duration
   <= 514800 [Confidence: 98.38 %]
        - 4.18 %: proto > 1 and dst_ip <= 98 and max_prio <= 4 and src_port > 399 and count_events <= 5277 and avg_duration
   > 468 [Confidence: 97.88 %]

muchos src_port, alto anom_level:
    Field importance:
        1. count_events: 97.32 %
        2. src_port: 2.05 %
        3. dst_port: 0.63 %
    Rules summary:
    false: (data 99.55 % / prediction 99.55 %)
        - 99.99 %: count_events <= 20137 [Confidence: 99.99 %]
    true: (data 0.45 % / prediction 0.45 %) count_events > 20137
        - 97.01 %: count_events > 20137 and src_port > 5838 [Confidence: 97.13 %]

cnxs largas:
    Field importance:
        1. avg_duration: 76.05 %
        2. stdev_duration: 21.91 %
        3. active_hours_vector: 2.04 %
    Rules summary:
```

```
        false: (data 99.86 % / prediction 99.86 %)
            - 99.80 %: avg_duration <= 279231 and stdev_duration <= 5386965 [Confidence: 99.99 %]
        true: (data 0.14 % / prediction 0.14 %)
            - 71.43 %: avg_duration > 279231 and stdev_duration > 3337227 and active_hours_vector does not contain 49 [
     Confidence: 88.65 %]
            - 19.05 %: avg_duration > 598343 and 407525 < stdev_duration <= 3337227  [Confidence: 67.56 %]

Centroids:
    Global:
        dst_ip: 41.19751,
        proto: 1.14648,
        src_port: 185.42285,
        dst_port: 2.04224,
        max_prio: 4.29858,
        anom_level: 10.35474,
        count_events: 578.37231,
        avg_duration: 12047.19678,
        stdev_duration: 52380.30981,
        active_hours_vector: ['0','1','13','2','4','7','8']

    normal, muchas cnxs:
        dst_ip: 61.71194,
        proto: 1.12069,
        src_port: 246.80572,
        dst_port: 2.08495,
        max_prio: 3.99832,
        anom_level: 9.69975,
        count_events: 627.94491,
        avg_duration: 10820.71615,
        stdev_duration: 47315.15517,
        active_hours_vector: ['13','16','17','18','19','21','22','23','24','26','28','29','31']

    normal, pocas cnxs:
        dst_ip: 5.77314,
        proto: 1.01767,
        src_port: 14.80636,
        dst_port: 1.46643,
        max_prio: 4.8742,
        anom_level: 2.06572,
        count_events: 30.2212,
        avg_duration: 5596.7576,
        stdev_duration: 16062.41484,
        active_hours_vector: ['0','1']

    udp:
        dst_ip: 35.90647,
        proto: 1.97482,
        src_port: 119.72302,
        dst_port: 4.32374,
        max_prio: 4.10072,
        anom_level: 9.54676,
        count_events: 343.65468,
        avg_duration: 37688.95324,
        stdev_duration: 106667.93165,
        active_hours_vector: ['0','1','12','2','3','4','5']

    muchos src_port, alto anom_level:
        dst_ip: 63.13333,
        proto: 1.13333,
        src_port: 8547.53333,
        dst_port: 2.2,
        max_prio: 4.13333,
        anom_level: 579.2,
        count_events: 40270.93333,
        avg_duration: 3987.66667,
        stdev_duration: 49257.33333,
```

```
        active_hours_vector: ['1823','2202','394','397','405','416','418','419','421','426','436','439','445','447','448',
                \ '456','457','459','462','466','468','469','470','487','488','512']

    cnxs largas:
        dst_ip: 19.6,
        proto: 1.1,
        src_port: 43.1,
        dst_port: 2.4,
        max_prio: 4,
        anom_level: 12.6,
        count_events: 7771.4,
        avg_duration: 1042485.2,
        stdev_duration: 4482850.6,
        active_hours_vector: ['0','1','128','2','24','42','45','47','48']

Distance distribution:
    Global:
        Minimum: 0.11962
        Mean: 0.30352
        Median: 0.27856
        Maximum: 18.68361
        Standard deviation: 0.33109
        Sum: 9095.8272
        Sum squares: 6045.7376
        Variance: 0.10962
    cnxs largas:
        Minimum: 0.46805
        Mean: 2.26579
        Median: 1.65227
        Maximum: 15.2934
        Standard deviation: 2.50826
        Sum: 95.16312
        Sum squares: 473.56613
        Variance: 6.29138
    muchos src_port, alto anom_level:
        Minimum: 1.19458
        Mean: 1.86711
        Median: 1.43723
        Maximum: 16.81587
        Standard deviation: 1.94614
        Sum: 250.19265
        Sum squares: 970.8706
        Variance: 3.78747
    normal, muchas cnxs:
        Minimum: 0.08605
        Mean: 0.21608
        Median: 0.18162
        Maximum: 1.98052
        Standard deviation: 0.13239
        Sum: 3518.82143
        Sum squares: 1045.72926
        Variance: 0.01753
    normal, pocas cnxs:
        Minimum: 0.06391
        Mean: 0.10462
        Median: 0.07351
        Maximum: 2.1971
        Standard deviation: 0.09638
        Sum: 970.56251
        Sum squares: 187.70076
        Variance: 0.00929
    udp:
        Minimum: 0.0539
        Mean: 0.35029
        Median: 0.29939
        Maximum: 6.72883
```

```
        Standard deviation: 0.2774
        Sum: 1481.72937
        Sum squares: 844.45003
        Variance: 0.07695

Intercentroid distance:
    To centroid cnxs largas
        Minimum: 3.40099850543
        Mean: 3.68915984306
        Maximum: 4.42598450268
    To centroid muchos src_port, alto anom_level
        Minimum: 2.98637710724
        Mean: 3.3806781248
        Maximum: 4.42598450268
    To centroid normal, muchas cnxs
        Minimum: 0.407874203112
        Mean: 1.82877973845
        Maximum: 3.4489041555
    To centroid normal, pocas cnxs
        Minimum: 0.407874203112
        Mean: 1.88797806684
        Maximum: 3.48075220863
    To centroid udp
        Minimum: 0.471963487963
        Mean: 1.87914761673
        Maximum: 3.40099850543
```

Apéndice B

Datos de salida en Ensayo B: G-Means determina k=3

Resultados de aplicar G-Means sobre el *dataset* final, que mediante sus tests de distribución gaussiana sobre los *clusters* fijó automáticamente el valor $k = 3$.

```
G-means Cluster (critical_value=1) with 3 centroids

Data distribution:
    Global: 100 % (5524 instances)
    Cluster 0: 98.39 % (5435 instances)
    Cluster 1: 0.74 % (41 instances)
    Cluster 2: 0.87 % (48 instances)

Cluster metrics:
    total_ss (Total sum of squares): 4574.828540
    within_ss (Total within-cluster sum of the sum of squares): 3670.198800
    between_ss (Between sum of squares): 904.629740
    ratio_ss (Ratio of sum of squares): 0.197740

Centroids:
    Global: max_prio: 4.3418, dst_port: 2.66602, anom_level: 7.86768, proto: 0.34912, src_port: 790.23291, dst_ip: 100.78101,
      count_events: 2785.25269, avg_duration: 20541.0061, afterwork_sessions: 483.48486, night_sessions: 268.74609,
      work_sessions: 273.10547, stdev_duration: 99277.93237
    Cluster 0: max_prio: 4.34556, dst_port: 2.74616, anom_level: 6.81929, proto: 0.35597, src_port: 682.54065, dst_ip:
      100.32375, count_events: 1868.74988, avg_duration: 17123.96728, afterwork_sessions: 376.14874, night_sessions:
      182.16981, work_sessions: 183.91249, stdev_duration: 87918.17501
    Cluster 1: max_prio: 4.32258, dst_port: 2.54839, anom_level: 10.54839, proto: 0.12903, src_port: 345.90323, dst_ip:
      73.93548, count_events: 3958.54839, avg_duration: 598458.70968, afterwork_sessions: 165.80645, night_sessions: 91.35484,
       work_sessions: 103.83871, stdev_duration: 2621627.41935
    Cluster 2: max_prio: 4.03448, dst_port: 2.44828, anom_level: 139.75862, proto: 0.24138, src_port: 13328.34483, dst_ip:
      213.75862, count_events: 106285, avg_duration: 2182.31034, afterwork_sessions: 11855.41379, night_sessions: 9539.96552,
      work_sessions: 10449.2069, stdev_duration: 63621.10345

Distance distribution:
    Global:
        Minimum: 0.23841
        Mean: 0.55967
        Median: 0.48402
        Maximum: 29.96057
        Standard deviation: 0.71766
        Sum: 3091.6051
        Sum squares: 4574.82854
        Variance: 0.51504
```

```
    Cluster 0:
        Minimum: 0.23723
        Mean: 0.50618
        Median: 0.47349
        Maximum: 15.76721
        Standard deviation: 0.31266
        Sum: 2751.09305
        Sum squares: 1923.76893
        Variance: 0.09776
    Cluster 1:
        Minimum: 0.99235
        Mean: 1.81514
        Median: 1.35956
        Maximum: 15.22225
        Standard deviation: 2.26154
        Sum: 74.42067
        Sum squares: 339.66706
        Variance: 5.11458
    Cluster 2:
        Minimum: 0.91762
        Mean: 2.74905
        Median: 1.45014
        Maximum: 27.82768
        Standard deviation: 4.71308
        Sum: 131.95417
        Sum squares: 1406.76281
        Variance: 22.21308

Intercentroid distance:
    To centroid Cluster 0
        Minimum: 2.66534899241
        Mean: 3.03563979581
        Maximum: 3.4059305992
    To centroid Cluster 1
        Minimum: 2.66534899241
        Mean: 3.51343337939
        Maximum: 4.36151776636
    To centroid Cluster 2
        Minimum: 3.4059305992
        Mean: 3.88372418278
        Maximum: 4.36151776636
```

Apéndice C

Datos de salida en Ensayo C: K-Means con k=5, fracciones temporales con agrupación semanal

Resultados de aplicar K-Means sobre el *dataset* con las variables de fracciones temporales "{night,work,afterwork}_sessions", agrupación semanal y $k = 5$.

```
K-means Cluster (k=5) with 5 centroids

Data distribution:
    Global: 100 % (5524 instances)
    Cluster 0: 64.48 % (3562 instances)
    Cluster 1: 20.17 % (1114 instances)
    Cluster 2: 13.38 % (739 instances)
    Cluster 3: 0.05 % (3 instances)
    Cluster 4: 1.92 % (106 instances)

Cluster metrics:
    total_ss (Total sum of squares): 3429.233470
    within_ss (Total within-cluster sum of the sum of squares): 2277.636800
    between_ss (Between sum of squares): 1151.596670
    ratio_ss (Ratio of sum of squares): 0.335820

Centroids:
    Global: max_prio: 4.3418, dst_port: 2.66602, anom_level: 7.86768, proto: 0.34912, src_port: 790.23291, dst_ip: 100.78101,
      count_events: 2785.25269, avg_duration: 20541.0061, stdev_duration: 99277.93237
    Cluster 0: max_prio: 4.45031, dst_port: 1.97083, anom_level: 6.03175, proto: 0.07254, src_port: 388.07659, dst_ip:
      74.96954, count_events: 1004.61423, avg_duration: 15438.01704, stdev_duration: 58771.78728
    Cluster 1: max_prio: 3.9908, dst_port: 2.79768, anom_level: 15.68699, proto: 0.80552, src_port: 2659.63215, dst_ip:
      230.19925, count_events: 11488.61274, avg_duration: 11275.87875, stdev_duration: 109375.0671
    Cluster 2: max_prio: 4.18354, dst_port: 8.22077, anom_level: 6.47877, proto: 1.70673, src_port: 360.7518, dst_ip:
      54.73612, count_events: 1019.58459, avg_duration: 19914.91378, stdev_duration: 99582.08556
    Cluster 3: max_prio: 4.66667, dst_port: 2, anom_level: 7, proto: 0, src_port: 12.33333, dst_ip: 7, count_events:
      904.66667, avg_duration: 3409650, stdev_duration: 6497667.33333
    Cluster 4: max_prio: 4.21724, dst_port: 2.64828, anom_level: 10.36552, proto: 0.23793, src_port: 569.22069, dst_ip:
      107.52069, count_events: 3105.40345, avg_duration: 198442.72414, stdev_duration: 1629567.21034

Distance distribution:
    Global:
        Minimum: 0.23773
        Mean: 0.5485
```

```
        Median: 0.4819
        Maximum: 17.7724
        Standard deviation: 0.56568
        Sum: 3029.88797
        Sum squares: 3429.23347
        Variance: 0.32
    Cluster 0:
        Minimum: 0.25048
        Mean: 0.36979
        Median: 0.35877
        Maximum: 2.42488
        Standard deviation: 0.09773
        Sum: 1317.20426
        Sum squares: 521.10642
        Variance: 0.00955
    Cluster 1:
        Minimum: 0.14427
        Mean: 0.58004
        Median: 0.43639
        Maximum: 17.63899
        Standard deviation: 0.81358
        Sum: 646.16115
        Sum squares: 1111.50423
        Variance: 0.66191
    Cluster 2:
        Minimum: 0.21881
        Mean: 0.52141
        Median: 0.44846
        Maximum: 15.48226
        Standard deviation: 0.59334
        Sum: 385.32542
        Sum squares: 460.72996
        Variance: 0.35205
    Cluster 3:
        Minimum: 3.21894
        Mean: 5.36183
        Median: 5.16783
        Maximum: 7.69873
        Standard deviation: 2.24619
        Sum: 16.0855
        Sum squares: 96.33849
        Variance: 5.04536
    Cluster 4:
        Minimum: 0.29329
        Mean: 0.80578
        Median: 0.70244
        Maximum: 2.51707
        Standard deviation: 0.42688
        Sum: 85.41292
        Sum squares: 87.9577
        Variance: 0.18222

Intercentroid distance:
    To centroid Cluster 0
        Minimum: 0.728139945617
        Mean: 3.24519323391
        Maximum: 10.0396344808
    To centroid Cluster 1
        Minimum: 0.728139945617
        Mean: 3.28459533292
        Maximum: 10.0630872788
    To centroid Cluster 2
        Minimum: 0.732942028295
        Mean: 3.28857417981
        Maximum: 10.039044428
    To centroid Cluster 3
```

```
        Minimum: 8.92628393067
        Mean: 9.76701252957
        Maximum: 10.0630872788
    To centroid Cluster 4
        Minimum: 1.48005648091
        Mean: 3.38096587797
        Maximum: 8.92628393067
```

Apéndice D

Datos de salida en Ensayo D: K-Means con k=5, fracciones temporales con agrupación diaria

Resultados de aplicar K-Means sobre el *dataset* con las variables de fracciones temporales "{night,work,afterwork}_sessions", agrupación diaria y $k = 5$.

D.1. Dataset del miércoles 20 de mayo

```
K-means Cluster (k=5) with 5 centroids

Data distribution:
    Global: 100 % (4836 instances)
    normal, muchas cnxs: 41.52 % (2008 instances)
    normal, pocas cnxs: 30.44 % (1472 instances)
    muchas dst_ip: 17.56 % (849 instances)
    udp: 10.15 % (491 instances)
    anom(16): muuchos eventos y cnxs: 0.33 % (16 instances)

Cluster metrics:
    total_ss (Total sum of squares): 2839.615230
    within_ss (Total within-cluster sum of the sum of squares): 1665.917530
    between_ss (Between sum of squares): 1173.697700
    ratio_ss (Ratio of sum of squares): 0.413330

Models:
    normal, muchas cnxs:
        Field importance:
            1. max_prio: 41.32 %
            2. dst_ip: 32.23 %
            3. proto: 15.78 %
            4. src_port: 6.86 %
            5. dst_port: 3.06 %
            6. stdev_duration: 0.41 %
            7. night_sessions: 0.12 %
            8. avg_duration: 0.07 %
            9. count_events: 0.07 %
            10. anom_level: 0.06 %
```

```
        11. work_sessions: 0.02 %
    Rules summary:
    false: (data 58.48 % / prediction 58.48 %)
        - 54.92 %: max_prio > 4 [Confidence: 99.75 %]
        - 25.92 %: max_prio <= 4 and dst_ip > 81 [Confidence: 99.48 %]
        - 7.67 %: max_prio <= 4 and dst_ip <= 76 and proto > 0 and dst_port > 2 [Confidence: 98.26 %]
    true: (data 41.52 % / prediction 41.52 %) max_prio <= 4
        - 88.00 %: max_prio <= 4 and dst_ip <= 74 and proto <= 0 and src_port <= 251 [Confidence: 99.78 %]
        - 4.63 %: max_prio <= 4 and dst_ip <= 68 and 0 < proto <= 1  and dst_port <= 2 and stdev_duration <= 341342 and
  avg_duration > 1332 [Confidence: 96.03 %]

normal, pocas cnxs:
    Field importance:
        1. max_prio: 86.73 %
        2. proto: 9.69 %
        3. dst_port: 2.84 %
        4. count_events: 0.74 %
    Rules summary:
    false: (data 69.56 % / prediction 69.56 %)
        - 97.59 %: max_prio <= 4 [Confidence: 99.88 %]
    true: (data 30.44 % / prediction 30.44 %) max_prio > 4 and proto <= 1
        - 99.52 %: max_prio > 4 and proto <= 1 and dst_port <= 3 and count_events <= 607 [Confidence: 99.74 %]

udp:
    Field importance:
        1. proto: 79.70 %
        2. dst_port: 10.69 %
        3. dst_ip: 4.14 %
        4. src_port: 1.96 %
        5. work_sessions: 1.33 %
        6. stdev_duration: 1.25 %
        7. anom_level: 0.54 %
        8. count_events: 0.26 %
        9. avg_duration: 0.13 %
    Rules summary:
    false: (data 89.85 % / prediction 89.85 %)
        - 93.44 %: proto <= 0 and stdev_duration <= 16159157 [Confidence: 99.90 %]
        - 5.34 %: 0 < proto <= 1  and dst_port <= 2 and stdev_duration <= 1089899 [Confidence: 98.37 %]
    true: (data 10.15 % / prediction 10.15 %)
        - 71.69 %: proto > 1 and work_sessions <= 380 and count_events <= 35902 [Confidence: 98.92 %]
        - 21.38 %: 0 < proto <= 1  and dst_port > 2 and dst_ip <= 74 and anom_level > 0 [Confidence: 96.47 %]

muchas dst_ip:
    Field importance:
        1. dst_ip: 61.67 %
        2. src_port: 17.25 %
        3. proto: 11.82 %
        4. count_events: 2.82 %
        5. dst_port: 2.16 %
        6. max_prio: 1.30 %
        7. work_sessions: 1.27 %
        8. stdev_duration: 1.10 %
        9. afterwork_sessions: 0.23 %
        10. night_sessions: 0.18 %
        11. avg_duration: 0.13 %
        12. anom_level: 0.06 %
    Rules summary:
    false: (data 82.44 % / prediction 82.44 %)
        - 84.47 %: dst_ip <= 65 and src_port <= 205 [Confidence: 99.89 %]
        - 5.32 %: 65 < dst_ip <= 75  and src_port <= 205 and proto <= 0 [Confidence: 98.22 %]
    true: (data 17.56 % / prediction 17.56 %)
        - 65.25 %: dst_ip > 78 and proto <= 0 and count_events <= 20998 and dst_port <= 4 and max_prio <= 4 and src_port >
  187 [Confidence: 99.31 %]
        - 8.24 %: dst_ip > 78 and proto <= 1 and count_events <= 20998 and dst_port <= 3 and max_prio <= 4 and 139 <
  src_port <= 187  [Confidence: 94.80 %]
```

```
            - 8.24 %: dst_ip > 78 and 0 < proto <= 1  and count_events <= 20998 and dst_port <= 4 and max_prio <= 4 and
      src_port > 261 [Confidence: 94.80 %]

    anom(16): muuchos eventos y cnxs:
        Field importance:
            1. count_events: 100.00 %
        Rules summary:
        false: (data 99.67 % / prediction 99.67 %) count_events <= 21052 [Confidence: 99.92 %]
        true: (data 0.33 % / prediction 0.33 %) count_events > 21052 [Confidence: 80.64 %]

Centroids:
    Global:
        dst_ip: 41.66992,
        proto: 0.23267,
        src_port: 176.81226,
        dst_port: 2.10693,
        max_prio: 4.31787,
        anom_level: 6.29883,
        count_events: 492.9646,
        avg_duration: 21632.82446,
        stdev_duration: 60528.82739,
        night_sessions: 57.94141,
        work_sessions: 57.36621,
        afterwork_sessions: 98.4292

    normal, muchas cnxs:
        max_prio: 4,
        dst_ip: 43.84316,
        proto: 0.04837,
        src_port: 112.67718,
        dst_port: 2.04181,
        stdev_duration: 54004.72215,
        night_sessions: 23.82766,
        avg_duration: 14157.9298,
        count_events: 296.99434,
        anom_level: 10.31259,
        work_sessions: 25.0738,
        afterwork_sessions: 64.37728

    normal, pocas cnxs:
        max_prio: 5,
        proto: 0.03475,
        dst_port: 1.4482,
        count_events: 20.30733,
        anom_level: 0,
        src_port: 9.847,
        dst_ip: 5.11645,
        avg_duration: 19812.79107,
        afterwork_sessions: 6.42015,
        night_sessions: 1.66241,
        work_sessions: 1.81784,
        stdev_duration: 20215.99608

    udp:
        proto: 1.72259,
        dst_port: 4.1145,
        dst_ip: 43.60057,
        src_port: 137.24005,
        work_sessions: 25.85965,
        stdev_duration: 187667.52753,
        count_events: 359.39521,
        avg_duration: 65984.58362,
        anom_level: 4.11826,
        max_prio: 4.16152,
        afterwork_sessions: 81.50734,
        night_sessions: 33.7245
```

```
    muchas dst_ip:
        dst_ip: 97.34976,
        src_port: 492.50022,
        proto: 0.16824,
        count_events: 1239.55768,
        dst_port: 2.23692,
        max_prio: 4.00119,
        work_sessions: 134.4331,
        stdev_duration: 66178.72904,
        afterwork_sessions: 232.31084,
        night_sessions: 142.14698,
        avg_duration: 10575.59195,
        anom_level: 8.4975

    anom(16): muuchos eventos y cnxs:
        count_events: 38265.85515,
        dst_ip: 68.50611,
        dst_port: 2.18674,
        src_port: 7324.59511,
        max_prio: 4.05759,
        anom_level: 591.00175,
        proto: 0.31239,
        avg_duration: 4469.58464,
        stdev_duration: 99635.59162
        night_sessions: 5052.32635,
        work_sessions: 5243.13962,
        afterwork_sessions: 5162.80628,

Distance distribution:
    Global:
        Minimum: 0.17795
        Mean: 0.42761
        Median: 0.3839
        Maximum: 24.3706
        Standard deviation: 0.63594
        Sum: 2067.90189
        Sum squares: 2839.61523
        Variance: 0.40442
    anom(16): muuchos eventos y cnxs:
        Minimum: 1.92792
        Mean: 4.73133
        Median: 2.35333
        Maximum: 19.64673
        Standard deviation: 5.14434
        Sum: 75.70126
        Sum squares: 755.1311
        Variance: 26.46424
    muchas dst_ip:
        Minimum: 0.08499
        Mean: 0.2853
        Median: 0.19042
        Maximum: 2.86214
        Standard deviation: 0.2966
        Sum: 242.22316
        Sum squares: 143.70551
        Variance: 0.08797
    normal, muchas cnxs:
        Minimum: 0.02243
        Mean: 0.14859
        Median: 0.12853
        Maximum: 2.31269
        Standard deviation: 0.12303
        Sum: 298.37279
        Sum squares: 74.71612
        Variance: 0.01514
```

```
    normal, pocas cnxs:
        Minimum: 0.05988
        Mean: 0.10034
        Median: 0.07307
        Maximum: 3.70515
        Standard deviation: 0.1399
        Sum: 147.70117
        Sum squares: 43.60945
        Variance: 0.01957
    udp:
        Minimum: 0.14194
        Mean: 0.557
        Median: 0.39236
        Maximum: 20.04429
        Standard deviation: 1.00653
        Sum: 273.48806
        Sum squares: 648.75535
        Variance: 1.01311

Intercentroid distance:
    To centroid anom(16): muuchos eventos y cnxs
        Minimum: 5.49925353941
        Mean: 5.65226803548
        Maximum: 5.74871982797
    To centroid muchas dst_ip
        Minimum: 0.372412734785
        Mean: 1.84826938648
        Maximum: 5.49925353941
    To centroid normal, muchas cnxs
        Minimum: 0.372412734785
        Mean: 1.82157121272
        Maximum: 5.6688626622
    To centroid normal, pocas cnxs
        Minimum: 0.535163040825
        Mean: 1.98257303533
        Maximum: 5.74871982797
    To centroid udp
        Minimum: 0.709846413056
        Mean: 2.0073695167
        Maximum: 5.69223611234
```

D.2. Dataset del jueves 21 de mayo

```
K-means Cluster (k=5) with 5 centroids

Data distribution:
    Global: 100 % (4801 instances)
    normal, muchas cnxs: 56.26 % (2701 instances)
    normal, pocas cnxs: 29.41 % (1412 instances)
    udp: 12.37 % (594 instances)
    cnxs largas: 1.92 % (92 instances)
    anom(2): muuchos eventos y cnxs: 0.04 % (2 instances)

Cluster metrics:
    total_ss (Total sum of squares): 2799.699460
    within_ss (Total within-cluster sum of the sum of squares): 1381.713600
    between_ss (Between sum of squares): 1417.985860
    ratio_ss (Ratio of sum of squares): 0.506480

Models:
    normal, muchas cnxs:
        Field importance:
            1. max_prio: 55.41 %
            2. proto: 32.85 %
            3. dst_port: 4.68 %
```

```
        4. stdev_duration: 4.05 %
        5. src_port: 1.53 %
        6. avg_duration: 0.89 %
        7. work_sessions: 0.30 %
        8. dst_ip: 0.29 %
    Rules summary:
    false: (data 43.74 % / prediction 43.74 %)
        - 72.05 %: max_prio > 4 and work_sessions <= 1320 [Confidence: 99.75 %]
        - 15.38 %: max_prio <= 4 and proto > 1 [Confidence: 98.82 %]
        - 9.19 %: max_prio <= 4 and 0 < proto <= 1  and dst_port > 2 and dst_ip <= 268 [Confidence: 98.05 %]
    true: (data 56.26 % / prediction 56.26 %)
        - 93.04 %: max_prio <= 4 and proto <= 0 and stdev_duration <= 480091 [Confidence: 99.85 %]
        - 5.89 %: max_prio <= 4 and 0 < proto <= 1  and dst_port <= 2 and src_port > 50 and avg_duration <= 17903 and
 dst_ip > 41 [Confidence: 97.64 %]

normal, pocas cnxs:
    Field importance:
        1. max_prio: 79.08 %
        2. proto: 10.39 %
        3. avg_duration: 6.38 %
        4. dst_port: 2.51 %
        5. count_events: 0.75 %
        6. stdev_duration: 0.67 %
        7. work_sessions: 0.22 %
    Rules summary:
    false: (data 70.59 % / prediction 70.59 %)
        - 96.96 %: max_prio <= 4 [Confidence: 99.88 %]
    true: (data 29.41 % / prediction 29.41 %) max_prio > 4 and proto <= 1
        - 96.32 %: max_prio > 4 and proto <= 1 and avg_duration <= 249555 and dst_port <= 3 and count_events <= 109 [
 Confidence: 99.72 %]

udp:
    Field importance:
        1. proto: 76.85 %
        2. dst_port: 11.31 %
        3. stdev_duration: 3.00 %
        4. avg_duration: 2.82 %
        5. dst_ip: 1.92 %
        6. anom_level: 1.76 %
        7. max_prio: 1.11 %
        8. work_sessions: 0.57 %
        9. src_port: 0.53 %
        10. night_sessions: 0.12 %
    Rules summary:
    false: (data 87.63 % / prediction 87.63 %)
        - 93.77 %: proto <= 0 [Confidence: 99.90 %]
    true: (data 12.37 % / prediction 12.37 %) proto > 0
        - 61.62 %: proto > 1 and stdev_duration <= 467883 [Confidence: 98.96 %]
        - 31.48 %: 0 < proto <= 1  and dst_port > 2 and stdev_duration <= 560151 and max_prio <= 4 and work_sessions <= 348
  [Confidence: 97.99 %]

conexiones largas:
    Field importance:
        1. avg_duration: 70.23 %
        2. stdev_duration: 27.15 %
        3. count_events: 1.30 %
        4. night_sessions: 0.65 %
        5. dst_port: 0.42 %
        6. work_sessions: 0.25 %
    Rules summary:
    false: (data 98.08 % / prediction 98.08 %)
        - 96.37 %: avg_duration <= 39029 [Confidence: 99.91 %]
    true: (data 1.92 % / prediction 1.92 %) avg_duration > 39029
        - 61.96 %: avg_duration > 39029 and stdev_duration > 609055 and night_sessions <= 89 [Confidence: 93.69 %]
        - 19.57 %: avg_duration > 451875 and stdev_duration <= 488495 [Confidence: 82.41 %]
```

```
    anom(2): muuchos eventos y cnxs:
        Field importance:
            1. work_sessions: 100.00 %
        Rules summary:
        false: (data 99.96 % / prediction 99.96 %) work_sessions <= 10132 [Confidence: 99.92 %]
        true: (data 0.04 % / prediction 0.04 %) work_sessions > 10132 [Confidence: 34.24 %]

Centroids:
    Global:
        dst_ip: 41.43579,
        proto: 0.2583,
        src_port: 179.22656,
        dst_port: 2.0752,
        max_prio: 4.31445,
        anom_level: 8.08374,
        count_events: 523.83667,
        avg_duration: 17792.48413,
        stdev_duration: 45217.60181,
        night_sessions: 58.32178,
        work_sessions: 57.36987,
        afterwork_sessions: 99.26245

    normal, muchas cnxs:
        max_prio: 4.00087,
        proto: 0.07595,
        dst_port: 2.06424,
        stdev_duration: 34366.40104,
        src_port: 257.97569,
        avg_duration: 10231.86328,
        work_sessions: 74.32509,
        dst_ip: 58.61328,
        anom_level: 10.0638,
        count_events: 697.09332,
        afterwork_sessions: 126.63151,
        night_sessions: 74.43229

    normal, pocas cnxs:
        max_prio: 5,
        proto: 0.04395,
        avg_duration: 7331.02156,
        dst_port: 1.41045,
        count_events: 18.16667,
        stdev_duration: 7914.26119,
        work_sessions: 2.17413,
        anom_level: 0,
        src_port: 9.98093,
        dst_ip: 4.80514,
        afterwork_sessions: 6.09619,
        night_sessions: 1.78275

    udp:
        proto: 1.64372,
        dst_port: 3.87045,
        stdev_duration: 65689.04251,
        avg_duration: 15995.24291,
        dst_ip: 52.93117,
        anom_level: 4.61134,
        max_prio: 4.11538,
        work_sessions: 35.91498,
        src_port: 171.56478,
        night_sessions: 44.95344,
        count_events: 400.45142,
        afterwork_sessions: 94.30162

    conexiones largas:
        avg_duration: 358479.48889,
```

```
        stdev_duration: 722651.46667,
        count_events: 2015.05556,
        night_sessions: 23.28889,
        dst_port: 2.13333,
        work_sessions: 22.38889,
        max_prio: 4.38889,
        anom_level: 84.13333,
        proto: 0.21111,
        src_port: 86.52222,
        dst_ip: 33.25556,
        afterwork_sessions: 43.6

    anom(2): muuchos eventos y cnxs:
        work_sessions: 21090,
        count_events: 68211.5,
        dst_ip: 46.5,
        dst_port: 3,
        src_port: 15709.5,
        max_prio: 4.5,
        anom_level: 0,
        proto: 0.5,
        avg_duration: 0,
        afterwork_sessions: 25994.5,
        night_sessions: 21119.5,
        stdev_duration: 1

Distance distribution:
    Global:
        Minimum: 0.18145
        Mean: 0.45877
        Median: 0.39585
        Maximum: 23.72496
        Standard deviation: 0.61054
        Sum: 2202.56835
        Sum squares: 2799.69946
        Variance: 0.37275
    anom(2): muuchos eventos y cnxs:
        Minimum: 5.93656
        Mean: 5.93656
        Median: 5.93656
        Maximum: 5.93656
        Standard deviation: 0
        Sum: 11.87312
        Sum squares: 70.48549
        Variance: 0
    cnxs largas:
        Minimum: 0.45728
        Mean: 1.42678
        Median: 0.98558
        Maximum: 18.11576
        Standard deviation: 1.98942
        Sum: 131.26395
        Sum squares: 547.44472
        Variance: 3.9578
    normal, muchas cnxs:
        Minimum: 0.03385
        Mean: 0.25467
        Median: 0.18086
        Maximum: 5.97993
        Standard deviation: 0.3359
        Sum: 687.85791
        Sum squares: 479.80883
        Variance: 0.11283
    normal, pocas cnxs:
        Minimum: 0.06098
        Mean: 0.11059
```

```
        Median: 0.06908
        Maximum: 1.17357
        Standard deviation: 0.11127
        Sum: 156.1587
        Sum squares: 34.74097
        Variance: 0.01238
    udp:
        Minimum: 0.1558
        Mean: 0.5003
        Median: 0.37693
        Maximum: 7.88024
        Standard deviation: 0.41179
        Sum: 297.17786
        Sum squares: 249.23359
        Variance: 0.16957

Intercentroid distance:
    To centroid anom(2): muuchos eventos y cnxs
        Minimum: 18.0489403657
        Mean: 18.1100414314
        Maximum: 18.1538634236
    To centroid cnxs largas
        Minimum: 1.4937042126
        Mean: 5.68766296031
        Maximum: 18.1453578698
    To centroid normal, muchas cnxs
        Minimum: 0.593072338758
        Mean: 5.19572714444
        Maximum: 18.0489403657
    To centroid normal, pocas cnxs
        Minimum: 0.593072338758
        Mean: 5.29305100718
        Maximum: 18.1538634236
    To centroid udp
        Minimum: 0.647191660718
        Mean: 5.2901030157
        Maximum: 18.0920040667
```

D.3. Dataset del viernes 22 de mayo

```
K-means Cluster (k=5) with 5 centroids

Data distribution:
    Global: 100 % (4882 instances)
    normal, muchas cnxs: 64.42 % (3145 instances)
    normal, pocas cnxs: 31.18 % (1522 instances)
    cnxs largas: 2.77 % (135 instances)
    udp: 1.62 % (79 instances)
    anom(1): muuchos eventos y cnxs: 0.02 % (1 instance)

Cluster metrics:
    total_ss (Total sum of squares): 2712.877320
    within_ss (Total within-cluster sum of the sum of squares): 1339.019240
    between_ss (Between sum of squares): 1373.858080
    ratio_ss (Ratio of sum of squares): 0.506420

Models:
    normal, muchas cnxs:
        Field importance:
            1. max_prio: 73.33 %
            2. stdev_duration: 15.66 %
            3. dst_port: 8.15 %
            4. dst_ip: 1.41 %
            5. count_events: 0.63 %
            6. work_sessions: 0.56 %
```

```
            7. anom_level: 0.13 %
            8. night_sessions: 0.13 %
        Rules summary:
        false: (data 35.58 % / prediction 35.58 %)
            - 90.27 %: max_prio > 4 and work_sessions <= 1312 [Confidence: 99.76 %]
            - 5.70 %: max_prio <= 4 and stdev_duration > 428751 and night_sessions <= 210 [Confidence: 96.27 %]
        true: (data 64.42 % / prediction 64.42 %)
            - 95.99 %: max_prio <= 4 and stdev_duration <= 428751 and dst_port <= 5 and dst_ip > 6 and work_sessions <= 17935 [
      Confidence: 99.87 %]
            - 3.12 %: max_prio <= 4 and stdev_duration <= 428751 and dst_port <= 5 and dst_ip <= 6 and count_events <= 63 and
      anom_level > 2 [Confidence: 96.23 %]

    normal, pocas cnxs:
            1. max_prio: 85.82 %
            2. count_events: 6.90 %
            3. avg_duration: 3.53 %
            4. dst_port: 2.57 %
            5. anom_level: 0.48 %
            6. work_sessions: 0.31 %
            7. stdev_duration: 0.28 %
            8. dst_ip: 0.11 %
        Rules summary:
        false: (data 68.82 % / prediction 68.82 %)
            - 95.60 %: max_prio <= 4 and dst_port > 1 [Confidence: 99.88 %]
        true: (data 31.18 % / prediction 31.18 %)
            - 88.57 %: max_prio > 4 and count_events <= 35 and dst_port <= 6 [Confidence: 99.72 %]
            - 10.84 %: max_prio > 4 and count_events > 35 and avg_duration <= 118094 and dst_port <= 6 and work_sessions <=
      1312 [Confidence: 97.72 %]

    udp:
        Field importance:
            1. dst_port: 98.34 %
            2. afterwork_sessions: 1.13 %
            3. night_sessions: 0.53 %
        Rules summary:
        false: (data 98.38 % / prediction 98.38 %)
            - 99.94 %: dst_port <= 5 [Confidence: 99.92 %]
        true: (data 1.62 % / prediction 1.62 %) dst_port > 5
            - 94.94 %: dst_port > 6 [Confidence: 95.13 %]

    cnxs largas:
        Field importance:
            1. stdev_duration: 75.58 %
            2. avg_duration: 17.39 %
            3. count_events: 3.61 %
            4. dst_port: 2.05 %
            5. work_sessions: 0.71 %
            6. night_sessions: 0.67 %
        Rules summary:
        false: (data 97.23 % / prediction 97.23 %)
            - 99.37 %: stdev_duration <= 355718 and avg_duration <= 206046 [Confidence: 99.92 %]
        true: (data 2.77 % / prediction 2.77 %)
            - 82.96 %: stdev_duration > 431547 and dst_port <= 7 and night_sessions <= 210 [Confidence: 96.68 %]
            - 11.11 %: stdev_duration <= 431547 and avg_duration > 206046 and count_events > 63 [Confidence: 79.61 %]

    anom(1): muuchos eventos y cnxs:
        Field importance:
            1. work_sessions: 100.00 %
        Rules summary:
        false: (data 99.98 % / prediction 99.98 %) work_sessions <= 22641 [Confidence: 99.92 %]
        true: (data 0.02 % / prediction 0.02 %) work_sessions > 22641 [Confidence: 20.65 %]

Centroids:
    Global:
        dst_ip: 40.63965,
        proto: 0.24683,
```

```
    src_port: 178.65649,
    dst_port: 2.06372,
    max_prio: 4.32324,
    anom_level: 6.31201,
    count_events: 493.44165,
    avg_duration: 16031.04517,
    stdev_duration: 45076.73242,
    night_sessions: 58.35034,
    work_sessions: 58.30225,
    afterwork_sessions: 97.61182

normal, muchas cnxs:
    max_prio: 4.00061,
    stdev_duration: 36196.79903,
    dst_port: 2.13778,
    dst_ip: 57.97209,
    count_events: 679.49076,
    work_sessions: 74.2723,
    anom_level: 9.43639,
    night_sessions: 74.51753,
    proto: 0.28652,
    src_port: 253.07345,
    avg_duration: 10235.43411,
    afterwork_sessions: 127.20052

normal, pocas cnxs:
    max_prio: 4.99797,
    count_events: 23.83739,
    avg_duration: 7345.21212,
    dst_port: 1.44002,
    anom_level: 0.00125,
    work_sessions: 3.07279,
    stdev_duration: 9476.48047,
    dst_ip: 6.02046,
    proto: 0.10247,
    src_port: 13.55686,
    afterwork_sessions: 7.8502,
    night_sessions: 2.73789

udp:
    dst_port: 10.94048,
    afterwork_sessions: 87.0119,
    night_sessions: 40.22917,
    max_prio: 4.19345,
    anom_level: 1.27679,
    proto: 1.39583,
    src_port: 130.93452,
    dst_ip: 7.14881,
    count_events: 264.87798,
    avg_duration: 20494.34226,
    work_sessions: 6.0506,
    stdev_duration: 75272.2619,
    work_sessions: 6.0506

cnxs largas:
    stdev_duration: 617650.08703,
    avg_duration: 249823.38737,
    count_events: 489.55973,
    dst_port: 2.30034,
    work_sessions: 30.76621,
    night_sessions: 31.3413,
    max_prio: 4.26451,
    anom_level: 6.2116,
    proto: 0.36007,
    src_port: 114.77645,
    dst_ip: 40.46416,
```

```
        afterwork_sessions: 55.657

    anom(1): muuchos eventos y cnxs:
        work_sessions: 30444,
        max_prio: 4,
        dst_port: 4,
        anom_level: 0,
        proto: 2,
        src_port: 16152,
        dst_ip: 45,
        count_events: 94864,
        avg_duration: 10,
        afterwork_sessions: 35553,
        night_sessions: 28774,
        stdev_duration: 1164

Distance distribution:
    Global:
        Minimum: 0.17917
        Mean: 0.50098
        Median: 0.44513
        Maximum: 24.43108
        Standard deviation: 0.55206
        Sum: 2445.76741
        Sum squares: 2712.87732
        Variance: 0.30477
    anom(1): muuchos eventos y cnxs:
        Minimum: 0
        Mean: 0
        Median: 0
        Maximum: 0
        Standard deviation: 0
        Sum: 0
        Sum squares: 0
        Variance: 0
    cnxs largas:
        Minimum: 0.39578
        Mean: 0.99327
        Median: 0.73855
        Maximum: 6.23048
        Standard deviation: 0.80792
        Sum: 134.09164
        Sum squares: 220.65574
        Variance: 0.65273
    normal, muchas cnxs:
        Minimum: 0.11273
        Mean: 0.38009
        Median: 0.29724
        Maximum: 12.5356
        Standard deviation: 0.40402
        Sum: 1195.39787
        Sum squares: 967.56632
        Variance: 0.16323
    normal, pocas cnxs:
        Minimum: 0.07072
        Mean: 0.13533
        Median: 0.08138
        Maximum: 1.32384
        Standard deviation: 0.15132
        Sum: 205.97859
        Sum squares: 62.70337
        Variance: 0.0229
    udp:
        Minimum: 0.21299
        Mean: 0.61204
        Median: 0.48209
```

```
        Maximum: 7.83686
        Standard deviation: 0.86603
        Sum: 48.35141
        Sum squares: 88.09381
        Variance: 0.75001

Intercentroid distance:
    To centroid anom(1): muuchos eventos y cnxs
        Minimum: 24.4047458741
        Mean: 24.4693647495
        Maximum: 24.5114735664
    To centroid cnxs largas
        Minimum: 1.2702389869
        Mean: 7.215316254
        Maximum: 24.4909530155
    To centroid normal, muchas cnxs
        Minimum: 0.654412314709
        Mean: 6.91304421433
        Maximum: 24.4047458741
    To centroid normal, pocas cnxs
        Minimum: 0.654412314709
        Mean: 6.99209139976
        Maximum: 24.5114735664
    To centroid udp
        Minimum: 1.32277968156
        Mean: 7.23346660305
        Maximum: 24.4702865419
```

Apéndice E

Código del preprocesado

Este es el *script* con el que se transforman los vectores de características de sesiones (que ya se han extraído de los logs de firewall) en los datos agregados sobre los cuales se aplicará el *clustering*. Básicamente, recorre todas las líneas de entrada (cada una es un vector de características) para agruparlas por IP origen y calcular las siguientes métricas por día: número de IPs destino únicas, protocolos usados, número de puertos origen únicos, número de puertos destino únicos, nivel de anomalía medio, nivel de amenaza medio, prioridad máxima, total de eventos, duración de sesión media, desviación típica de duración de sesión, número de sesiones activas en horas nocturas, número de sesiones activas en horas de trabajo y número de sesiones activas en horas después del trabajo. La parte más compleja es la asignación de sesiones según correspondan a una franja horaria u otra, pero el proceso se ha explicado en los comentarios y se han incluido ejemplos.

```
#!/usr/bin/env python3
#
#usage: ./preprocess_clustering_dataset.py rawdata_7days
#exec time: 56s
#output:
#$ wc -l dataset_for_clustering.*
#   2884 dataset_for_clustering.19may
#   4837 dataset_for_clustering.20may
#   4802 dataset_for_clustering.21may
#   4883 dataset_for_clustering.22may
#   4876 dataset_for_clustering.23may
#   4671 dataset_for_clustering.24may
#   2931 dataset_for_clustering.25may
#  29884 total

import sys
from statistics import mean, stdev

raw_data = open(sys.argv[1])
data = {}

week_tstamps = [i for i in range(1589752800,                  1590357601,   60*60*8)]
#                               L18may00:00, so L25may00:00 is included^^, 8h step^^
days = ['18may', '19may', '20may', '21may', '22may', '23may', '24may', '25may']
slots = ['night', 'work', 'afterwork']
```

```
#     00:00 - 08:00 - 16:00   -   00:00

def print_data():
    global data
    with open("dataset_for_clustering.{}".format(days[int(t/3)]), 'w' ) as f:
        print("src_ip,dst_ip,proto,src_port,dst_port,anom_level,threat_level,max_prio,count_events,avg_duration,stdev_duration
     ,night_sessions,work_sessions,afterwork_sessions", file=f)
        for d in data:
            print( "{},{},{},{},{},{:.2f},{:.2f},{},{},{:.2f},{:.2f},{},{},{}".format( d,
                len(data[d]['dst_ip']),data[d]['proto'],len(data[d]['src_port']),len(data[d]['dst_port']),
                mean(data[d]['anom_level']),mean(data[d]['threat_level']),min(data[d]['max_prio']),sum(data[d]['count_events'
     ]),
                mean(data[d]['duration']),
                stdev(data[d]['duration']) if len(data[d]['duration'])>1 else 0,
                data[d]['slots']['night'],data[d]['slots']['work'],data[d]['slots']['afterwork']
            ), file=f)
            #                                                ^^ int() thows away the decimal part
    data = {}

t=0 # counter of actual element on week_tstamps
for line in raw_data:
    #example:
    # 1589212544 1589993516 172.28.0.83 52.177.165.30 tcp 57541 443 0.00 0.00 5 479 780972
    r = line.split()

    initial_tstamp = int(r[0])
    end_tstamp = int(r[1])
    src_ip = r[2]

    if initial_tstamp > week_tstamps[-1]:
    # so all week_tstamps have been covered:
        break

    if initial_tstamp > week_tstamps[t]:
    # so we move forward to the next week fraction:
        t+=1
        if t%3==0:
        # so 'night','work','afterwork' slots have passed and it's a new day:
            print_data()

    if src_ip not in data:
        # []: list. it's a mutable, or changeable, ordered sequence of elements. it preserves order, it admits duplicates
        # set(): it's an unordered collection of distinct hashable objects. it doesn't preserve order, it doesn't admit
     duplicates
        data[src_ip] = {
            'dst_ip': set(), 'proto': 0, 'src_port': set(), 'dst_port': set(),
            'anom_level': [], 'threat_level': [], 'max_prio': set(), 'count_events': [],
            'duration': [], 'slots': {s: 0 for s in slots}
        }

    ### count this session on the pertinent time slot:

    data[src_ip]['slots'][slots[ t%len(slots)-1 ]] += 1
    #example:
    # src_ip:10.212.138.53,from 1589807186(15:06:26) to 1589807191(15:06:31) -> {'night': 0, 'work': 1, 'afterwork': 0}
    # src_ip:10.212.138.53,from 1589920495(22:34:55) to 1589920504(22:35:04) -> {'night': 0, 'work': 1, 'afterwork': 1}

    i=t
    while end_tstamp > week_tstamps[i]:
    # long sessions count on every slots they are:
        data[src_ip]['slots'][slots[ i%len(slots)-1 ]] += 1
        i+=1
        if i==len(week_tstamps):
        # so end of the observed week has been reached:
            break
```

```
### count the rest of the features:

if r[3] not in data[src_ip]['dst_ip']: data[src_ip]['dst_ip'].add(         r[3] )
if r[4]=="udp": data[src_ip]['proto'] = 2 if ( data[src_ip]['proto']==1 ) else 1
if r[5] not in data[src_ip]['src_port']: data[src_ip]['src_port'].add(     r[5] )
if r[6] not in data[src_ip]['dst_port']: data[src_ip]['dst_port'].add(     r[6] )
data[src_ip]['anom_level'].append(                                    float(r[7]))
data[src_ip]['threat_level'].append(                                  float(r[8]))
if r[10] not in data[src_ip]['max_prio']: data[src_ip]['max_prio'].add(int(r[9]))
data[src_ip]['count_events'].append(                                   int(r[10]))
data[src_ip]['duration'].append(                                       int(r[11]))
```