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 según su comportamiento en una red empresarial

Trabajo Fin de Máster

Presentado por: Javier Artiga Garijo

Director: Luis Miguel Garay Gallastegui

Ciudad: Logroño
Fecha: [Fecha de Entrega]

Índice general

Resumen

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Palabras clave: Plabra clave 1, Palabra clave 2, Palaba Clave N

Abstract

Proin tincidunt enim nec fringilla euismod. Quisque id efficitur sapien. Sed hendrerit, nisl id efficitur elementum, arcu ex efficitur ipsum, sed auctor metus nibh id leo. Maecenas eu sem tortor. Etiam accumsan bibendum ante vitae auctor. Donec eget dolor gravida, tincidunt diam non, ornare orci. Pellentesque ornare blandit eros, sed maximus neque varius non.

Keywords: Plabra clave 1, Palabra clave 2, Palabra Clave N

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

[Los fragmentos entre corchetes son ideas que ampliaré próximamente]

[Párrafo introductorio del capítulo. El primer capítulo es siempre una introducción. En ella debes resumir de forma esquemática pero suficientemente clara lo esencial de cada una de las partes del trabajo. La lectura de este primer capítulo ha de dar una primera idea clara de lo que se pretendía, las conclusiones a las que se ha llegado y del procedimiento seguido.]

Como tal, **es uno de los capítulos más importantes de la memoria**. Las ideas principales a transmitir son la identificación del problema a tratar, la justificación de su importancia, los objetivos generales (a grandes rasgos) y un adelanto de la contribución que esperas hacer. Típicamente una introducción tiene la siguiente estructura:

1. Motivación: ¿Cuál es el problema que quieres tratar? ¿Cuáles crees que son las causas? ¿Por qué es relevante el problema?
2. Planteamiento del trabajo (sección 1.2): ¿Cómo se podría solucionar el problema? ¿Qué es lo que se propone? Aquí describes tus objetivos en términos generales (‘‘mejorar el aprendizaje de idiomas’’)
3. Estructura del trabajo (sección 1.3): Aquí describes brevemente lo que vas a contar en cada uno de los capítulos siguientes.

1.1. Motivación

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec tincidunt libero viverra mi elementum pellentesque. Mauris sollicitudin elementum turpis. Nunc dictum lectus nec ligula fringilla auctor. Proin ut sem felis. Sed suscipit maximus lorem, nec vulputate nisi. Donec eu

interdum lectus 1. Etiam viverra nec nulla vel facilisis. Vivamus purus lacus, laoreet id justo eu, euismod commodo justo.

1.2. Objetivos

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec tincidunt libero viverra mi elementum pellentesque. Mauris sollicitudin elementum turpis. Nunc dictum lectus nec ligula fringilla auctor. Proin ut sem felis. Sed suscipit maximus lorem, nec vulputate nisi. Donec eu interdum lectus 1. Etiam viverra nec nulla vel facilisis. Vivamus purus lacus, laoreet id justo eu, euismod commodo justo.

Vestibulum risus eros, fringilla quis tristique quis, tincidunt et dui. Sed et magna blandit, sagittis nibh dictum, eleifend magna. Suspendisse a nulla a augue ultrices molestie sit amet et magna. Vestibulum molestie metus id lorem bibendum rhoncus. Maecenas sit amet massa pretium, commodo nunc id, viverra augue. Curabitur nec ultricies sem. Donec congue lectus lorem, vel laoreet arcu mollis volutpat. Duis sem metus, consectetur ac diam et, pretium congue tortor. Quisque elementum mollis enim, nec blandit mauris feugiat quis.

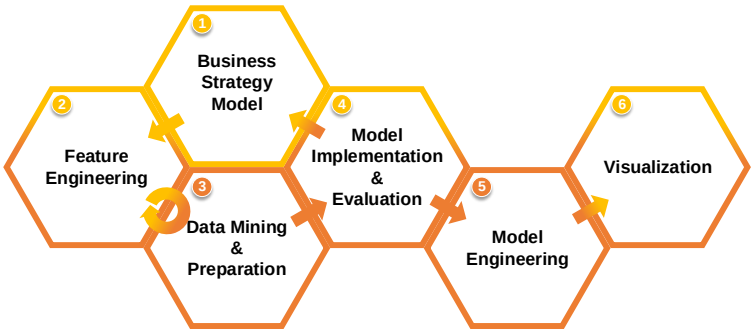

Figura 1: Duis tristique velit velit. Curabitur auctor, nibh.

1.3. Estructura del documento

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus eu risus convallis, cursus ante quis, condimentum nibh. Cras vulputate suscipit tortor eu pulvinar. Aenean aliquet felis lacinia aliquet interdum. In at diam sed lacus porta efficitur. Nunc hendrerit, felis id venenatis fringilla, augue justo egestas ipsum, vel pretium est lorem at arcu. Nullam finibus mauris et maximus viverra. Etiam sagittis ligula vel aliquam vehicula. Aenean consequat condimentum enim, eget posuere lectus dapibus sed. Suspendisse mollis eu orci eu tempor. Aliquam tempus enim eget egestas congue. Morbi nisi metus, suscipit in scelerisque quis, accumsan sit amet

nulla. Sed id imperdiet quam, eget commodo sem. Donec et molestie libero. Donec et quam mi. Vivamus consectetur est turpis, et ultricies leo dictum eu.

- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Vestibulum id tortor efficitur odio dictum suscipit ut blandit augue.
- Donec consectetur sem ac suscipit bibendum.
- Phasellus interdum metus sit amet lorem tempor, vel accumsan mi maximus.
- Suspendisse semper massa et porttitor gravida.

Capítulo 2

Objetivos y Metodología

Objetivos concretos y metodología de trabajo

[Párrafo introductorio del capítulo. Este bloque es el puente entre el estudio del dominio y la contribución a realizar. Según el tipo concreto de trabajo, el bloque se puede organizar de distintas formas, pero los siguientes elementos deberían estar presentes con mayor o menor detalle.]

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 si se le proporcionan datos modificados intencionadamente.
- Refinar el algoritmo de clustering empleado, buscando categorías más específicas que no se estuvieran teniendo en cuenta antes.
- Diseñar una forma de visualización adecuada para los resultados.
- Alcanzar un modo de funcionamiento en tiempo real, en coordinación con los demás mecanismos de monitorización presentes y asegurando una precisión razonable (detecciones correctas frente a falsos positivos).

2.3. Metodología de Trabajo

[Sección previsiblemente modificable y ampliable según avance en el desarrollo del 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 generales, 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 algoritmos 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 elegirán dos algoritmos simples y ampliamente conocidos como son K-Means y DB-SCAN. El primero es un algoritmo exclusivo basado en centroides, mientras que el segundo particiona según la densidad de los clusters formados. 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 métricas de validación internas y externas como la silueta de sus clusters o el índice de Rand. Estas medidas son útiles para evaluar la calidad de resultados del algoritmo, pero debe apreciarse que proporcionan un escaso conocimiento sobre lo que contienen los clusters. 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 intrusiones mediante otros métodos, se usarán para probar la capacidad de detección del sistema. Si, llegado el momento de hacer este testeo, no se dispone de este tipo de ejemplos, se elaborarán datos que representen distintas clases de anomalías con diversos grados de evidencia. 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 integrará con el framework de alerting empleado y 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 en tiempo real.

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 relevacia 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... 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, además de las consideraciones anteriores 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 contempladas 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).

[Conclusiones sobre los trabajos previos]

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 cómo se han extraído y filtrado 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 de qué manera 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*

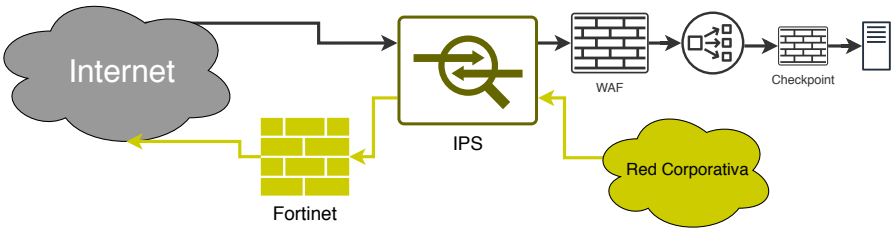

Figura 2: Diagrama del tratamiento de conexiones entrantes y salientes

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

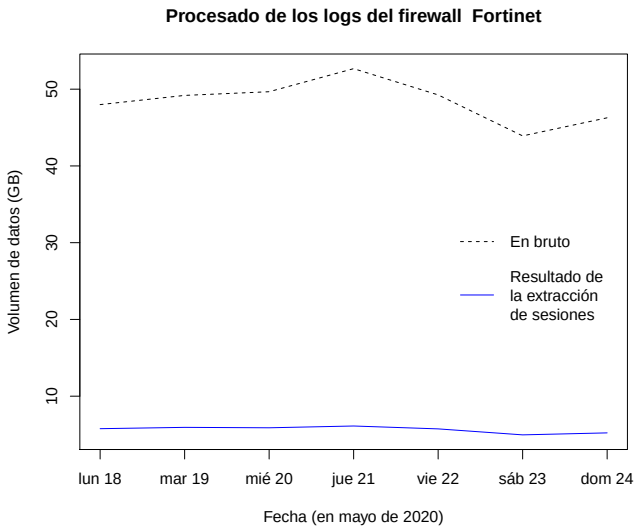

Figura 3: Volumen de logs en bruto y tras ser procesados, en una semana

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_INTERNETCORP" devid="FG1809999"
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=1513678724 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 07:04:23,001801037558,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 `$1-$2` 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 que corresponden a una IP origen. 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 descendentes, siendo 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 (esta muestra 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 A), 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.

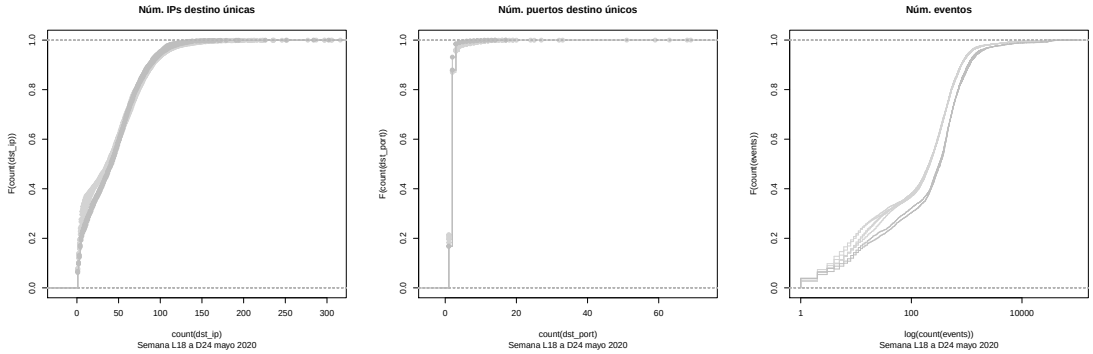

Figura 4: Funciones de distribución acumulada por número de IPs destino, puertos destino y eventos

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.

[Fig distrib. duración de sesiones (se tienen las vars. avg_duration y stdev_duration)]

[PCA? En (Bohara y col., 2016) explica por qué las componentes ppales, que representan la mayor parte de la varianza, pueden no capturar comportamientos anóms.]

[Normalización]

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.

[Explicar proceso de selección (filtrar feats. correladas y quedarse con las de más peso)]

4.6. Parametrización para los algoritmos de clustering

[Se puede hablar de: método del codo (cambios bruscos en la evolución de la inercia) para KMeans (y para más?), método de la silueta, elección de la distancia, método del gap, dendograma]

4.7. Análisis del clustering

4.8. Evaluación experimental

Capítulo 5

Resultados

Explicación de los resultados

[Párrafo introductorio del capítulo. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Suscipit maxime expedita possimus consequatur labore, id, dignissimos praesentium repellendus quisquam tempore natus eveniet magnam! Ad laborum quas, at tenetur eligendi officiis!]

Capítulo 6

Conclusiones

Conclusiones finales y trabajo futuro

[Párrafo introductorio del capítulo. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Suscipit maxime expedita possimus consequatur labore, id, dignissimos praesentium repellendus quisquam tempore natus eveniet magnam! Ad laborum quas, at tenetur eligendi officiis!]

Este último bloque (habitualmente un capítulo; en ocasiones, dos capítulos complementarios) es habitual en todos los tipos de trabajos y presenta el resumen final de tu trabajo y debe servir para informar del alcance y relevancia de tu aportación.

Suele estructurarse empezando con un resumen del problema tratado, de cómo se ha abordado y de por qué la solución sería válida. Es recomendable que incluya también un resumen de las contribuciones del trabajo, en el que relaciones las contribuciones y los resultados obtenidos con los objetivos que habías planteado para el trabajo, discutiendo hasta qué punto has conseguido resolver los objetivos planteados.

Finalmente, se suele dedicar una última sección a hablar de líneas de trabajo futuro que podrían aportar valor añadido al TFM realizado. La sección debería señalar las perspectivas de futuro que abre el trabajo desarrollado para el campo de estudio definido. En el fondo, debes justificar de qué modo puede emplearse la aportación que has desarrollado y en qué campos.

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.

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: 10.1145/1368436.1368445.

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

Código del preprocesado

Nam viverra, odio et vulputate ultricies, sem libero ornare nunc, nec suscipit urna sapien eu sapien. Nulla erat nulla, hendrerit non elit vitae, venenatis malesuada libero. Duis ornare sapien sed lacus condimentum rutrum. Donec feugiat erat id elit aliquam, a ultrices lectus mattis. In vitae nisl tortor. Proin pellentesque nec odio et posuere. Ut aliquet quam ac magna tincidunt ultrices. Nullam sit amet elementum leo. Pellentesque vitae mi dolor. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Suspendisse tincidunt mi arcu, vitae porttitor mauris elementum ut.

```
#!/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 = {}

###
# 1. aggregate:
###

t=0 # counter of actual element on week_tstamps

for line in raw_data:
    #examples:
    # 1589212544 1589993516 172.28.0.83 52.177.165.30 tcp 57541 443 0.00 0.00 5 479 780972
    # 1589212549 1589825851 172.28.8.92 52.177.165.30 tcp 58421 443 0.00 0.00 5 380 613302

    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
    #
    ###

    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]))

###
# 2. calculate:
###
```