Universidad Zaragoza

Trabajo Fin de Grado

Generación automática de reglas de seguridad en base a registros de sistema

Automatic generation of security rules based on system logs

Asier Salueña Sediles

Grado en Ingeniería Informática

Director:
Ricardo J. Rodríguez Fernández
Codirector:
Víctor Pérez Roche
Ponente:
Raquel Trillo Lado

Noviembre de 2018
Curso 2017/2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

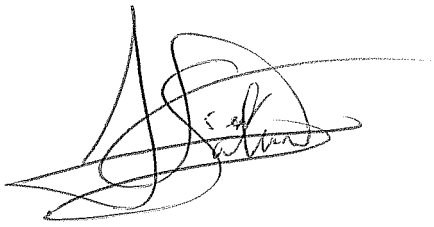
D./Dª. Asier Salueña Sediles,

con nº de DNI 17759656-E en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado en Ingenieria Informatica, (Título del Trabajo) Generación automática de reglas de seguridad en base a registros de sistema

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 01 de Septiembre de 2018



Fdo: Asier Salueña Sediles

Agradecimientos

Me gustaría dedicar este trabajo de fin de grado,
a mis padres, por animarme a comenzar este camino,
y ser la fuente de financiación de mis errores,
a Nadia, por aguantarme cuando este camino me puso a prueba,
a Ricardo, por no perder la paciencia y ayudarme cuantas veces lo he necesitado,
a todos aquellos miembros del equipo docente que estuvieron dispuestos a ayudar y aconsejar cuando fue necesario.

Generación automática de reglas de seguridad en base a registros de sistema

RESUMEN

Durante el año 2017, el Centro Criptológico Nacional (CCN-CERT), un organismo adscrito al Centro Nacional de Inteligencia, gestionó un total de 26.472 incidentes. Este número de incidentes supuso un 27.22 % más que en el año anterior. Las motivaciones de estos incidentes son frecuentemente ataques con el objetivo de la disrupción de sistemas, el ciberespionaje, la obtención de beneficios económicos, influenciación sobre la opinión pública o incluso la ciberguerra.

Este proyecto se enfoca en el desarrollo de un sistema que, mediante técnicas de aprendizaje automático, sea capaz de reconocer ataques a un sistema objetivo y los bloquee, minimizando el daño provocado en los mismos. Concretamente, los tipos de sistema que pretende defender son sistemas web. Para ello, el sistema monitoriza los logs generados por el sistema web bajo control tras recibir una petición web, apoyándose en una solución de almacenamiento y gestión de registros de sistemas que favorece la labor de parametrización de las peticiones a analizar (en concreto, mediante ELK).

Por tanto, el sistema desarrollado en este proyecto consta de tres sub-sistemas. El primer sistema se encarga de monitorizar el directorio donde se almacenan los logs del sistema a proteger. El segundo sistema analiza las peticiones recibidas y las evalúa mediante dos elementos: un clasificador automático y un sistema de reglas. El sistema de reglas usa los resultados del clasificador en conjunto con unas reglas pre-establecidas para determinar si la petición analizada es un ataque o no. Por último, el sistema de actuación valora la detección realizada por el sistema de análisis y aplica las medidas correctivas necesarias.

Como caso de estudio, se usa la plataforma Moodle para comprobar el funcionamiento del sistema desarrollado y valorar la efectividad de las defensas generadas. La plataforma Moodle es la herramienta más utilizada en el ámbito educativo como herramienta de apoyo docente, tanto en la Universidad de Zaragoza como en otras universidades.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

"A computer is secure if you can depend on it and its software to behave as you expect".- Simson Garfinkel and Gene Spafford, 1991

La proliferación de los llamados ciberataques está causando un daño cada vez mayor a las empresas, los gobiernos y las personas físicas. Entre 2013 y 2015 el coste por los delitos cibernéticos se cuadruplicó llegando a 400.000 millones de dólares y hasta 500.000 millones durante ese mismo periodo. En un informe de *Cybersecurity Ventures* sobre delitos cibernéticos [6], se prevé que el coste de las ciberamenazas aumente a 6 billones anuales en 2021 (véase la Figura 1.1), incluyendo diversos costes como daños y destrucción de datos, dinero robado, pérdida de productividad, robo de propiedad intelectual, robo de datos personales y financieros, malversación de fondos, fraude, interrupción de servicios, investigación forense, restauración y eliminación de datos y sistemas dañados, entre otros. Además de la cantidad de ciberataques, también está aumentando el daño producido por los mismos. Según un informe de PwC, multinacional del sector, los ataques se están volviendo progresivamente más destructivos y tienen como objetivo un abanico más amplio de sistemas [9].

La Encuesta Global de Seguridad del Estado de PwC 2017 afirma que el 59 % de los encuestados dice que la digitalización de sus ecosistemas empresariales ha incrementado sus presupuestos de seguridad. Como muestran estas estadísticas, la ciberseguridad es un área muy importante digna de compromiso, y las empresas están respondiendo en consecuencia.

Sin embargo, la lucha entre el aumento de las necesidades y la financiación limitada es característica de la industria de ciberseguridad. Afortunadamente, cada vez más empresas, departamentos gubernamentales y organizaciones están reconociendo la importancia de la ciberseguridad y están asignando fondos en consecuencia.

En la actualidad, los servicios y las empresas dependen casi en su totalidad de servicios web, por tanto la seguridad web se ha convertido en un elemento crítico para su operativa. El objetivo de la seguridad web es evitar cualquier tipo de ataque a un sistema web. Más formalmente, la seguridad web consiste en proteger los sitios web del acceso no autorizado (confidencialidad), modificación o destrucción de los datos no autorizados (integridad) o de la interrupción del servicio prestado (disponibilidad). La seguridad web efectiva

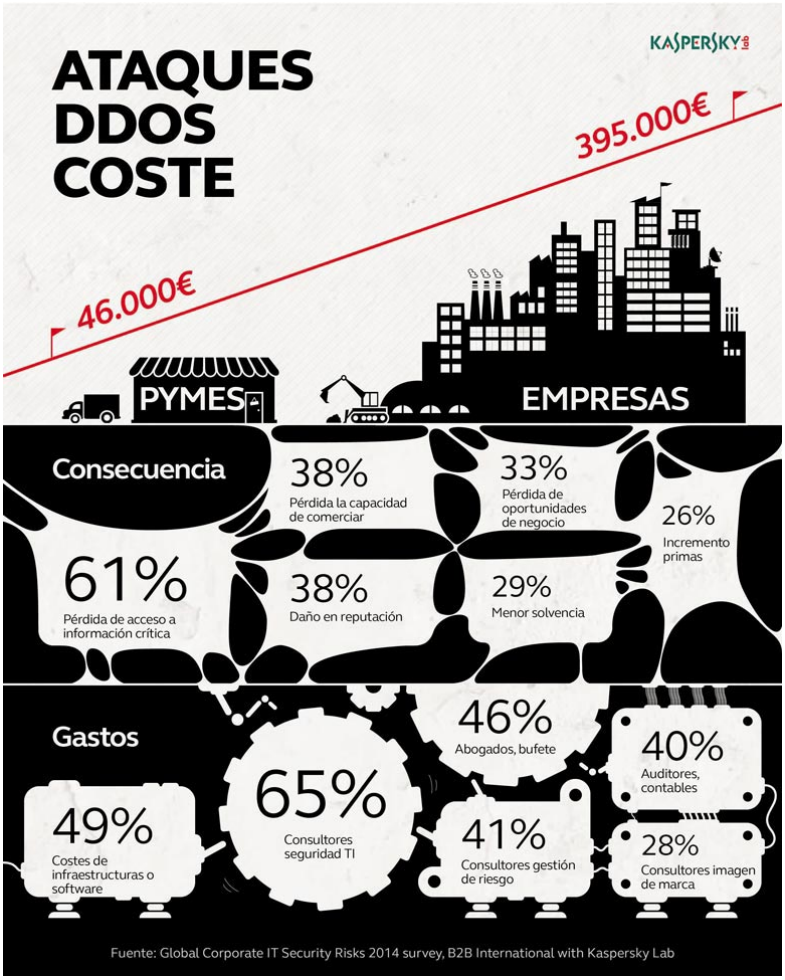


Figura 1.1: Coste de los ataques DDoS [8]

requiere por tanto un esfuerzo de diseño en todo el sitio web, desde su aplicación hasta la configuración del servidor o en el código del lado del cliente. Años atrás, el gasto en seguridad web era en la mayoría de los casos insuficiente. En la actualidad, ya no se valora la seguridad como un gasto, sino como una inversión que evita gastos futuros.

Según reporta la empresa especializada del sector de seguridad Akamai, desde noviembre de 2017 se han incrementado un 16 % los ataques de denegación de servicio, (consistentes en la inundación de una red o de un servicio web objetivo para sobrecargar sus recursos), otro 16 % de aumento en los ataques a la capa de infraestructura y un 4 % los ataques basados en reflexión en servicios web (consistentes en utilizar un componente externo potencialmente legítimo para enviar el tráfico de ataque a la víctima, ocultando así la propia identidad de los atacantes) [2]. Concretamente, el informe destaca un total de 400 millones de ataques a aplicaciones web, siendo los principales vectores de ataque

la inyección SQL (SQLi) con el 51 % del total, la inclusión de archivos locales (LFI[1]) con el 34 % del total, los filtros de script de sitios (XSS[2]) con un 8 % del total y un 7 % repartido entre diferentes vectores de ataque alternativos.

En el transcurso de una comunicación cliente-servidor, el servidor deja un rastro de todo lo que sucede con el cliente en un fichero de texto, conocido como registro del sistema (llamado también log, del inglés *system log*). Estos logs aportan información de los eventos que suceden en el servidor, como peticiones web y el resultado de las mismas. En el caso de los servidores `Apache` (que es el servidor web más utilizado, con un 42 % de cuota de mercado según W3Techs), existen dos tipos de logs diferentes. De ellos, el más importante es el registro de errores del servidor, donde Apache envía cualquier información de diagnóstico y registra cualquier error encontrado al procesar peticiones web. Por otro lado, el servidor almacena en el registro de acceso información sobre todas las peticiones que procesa.

Además de servidores web, es común en la infraestructura de red de una empresa disponer la presencia de distintos elementos de seguridad, cuyo objetivo es garantizar el correcto funcionamiento y la protección de la infraestructura en su totalidad (como por ejemplo *firewalls* o sistemas de detección de intrusiones). Un *firewall* es un elemento de seguridad de red diseñado para prevenir accesos no autorizados entre redes. Los sistemas de detección de intrusiones son elementos cuyo objetivo es la monitorización de la red buscando actividades sospechosas y alertando sobre ellas.

El objetivo de este proyecto es facilitar el proceso de securización de un servidor web. Así, se plantea el desarrollo de un sistema que genere reglas de manera automática en base a los eventos registrados en los logs del servidor web. Mediante un método de aprendizaje supervisado y usando criterios objetivos en base a datos de ataques actuales a sistemas web, el sistema desarrollado decidirá si una determinada petición web se considera ataque. Conviene destacar que este sistema no es un sustituto de los actuales sistemas de defensa, sino un complemento a los mismos. Como caso de estudio propuesto en este proyecto se usa la plataforma Moodle de apoyo docente para la evaluación del sistema. Esta plataforma está desplegada sobre las máquinas del Servicio de Informática y Comunicaciones (SICUZ) de la Universidad de Zaragoza y es ampliamente usada por la comunidad universitaria. Por tanto, puede ser un potencial objetivo de ataques web.

El documento esta organizado de la siguiente manera. En el Capítulo 2 se tratan los conocimientos previos necesarios para comprender este proyecto. En el Capítulo 3, se desarrollan los ataques que detecta el sistema de seguridad, describiendo su alcance y la metodología. A continuación, en el Capítulo 4 se habla sobre el diseño e implementación del sistema desarrollado y de su funcionamiento. En el Capítulo 5 se expone el caso de estudio contemplado acompañado de las pruebas realizadas. En el Capítulo 6 se habla del trabajo relacionado con este proyecto. Y, por último, en el Capítulo 7 se exponen las conclusiones del proyecto, así como los problemas encontrados y posibles líneas de trabajo futuro.

[1]Por sus siglas en inglés *Local File Inclusion*
[2]Por sus siglas en inglés *Cross Site Scripting*

Capítulo 2

Conocimientos previos

En este capítulo se van a desarrollar los conocimientos previos necesarios para comprender este proyecto. En primer lugar, se habla del `ELK Stack`, el software dedicado a almacenamiento normalizado de logs que sirve como base del resto del proyecto. En segundo lugar, se habla sobre el funcionamiento de un sistema de detección de intrusiones (IDS por sus siglas en inglés) y sus capacidades, y de manera más concreta de Snort, el sistema IDS utilizado en este proyecto. Por último, se explica el lenguaje UML (*Unified Model Language*)para modelado de sistemas usado en este proyecto durante la fase de desarrollo y de experimentación.

2.1. ELK

ELK es un conjunto de herramientas de código abierto que se combinan para crear una única plataforma de administración de logs permitiendo la monitorización, consolidación y análisis de logs generados en múltiples servidores. ELK en concreto hace uso de tres herramientas, que aunque son usables de manera independiente, unidas crean una combinación perfecta para la gestión de logs. En concreto, las funciones de cada una de estas herramientas son:

- **ElasticSearch:** Motor de búsquedas y análisis donde se almacenan los datos ya optimizados para la indexación. Se caracteriza por ser un sistema distribuido, tolerante a fallos y de alta disponibilidad.
- **Logstash:** Se encarga de recoger los datos de diferentes fuentes simultáneamente, tratarlos en la medida que sea necesario y luego finalmente poder almacenarlos (en este caso en ElasticSearch).
- **Kibana:** Permite visualizar los datos que se encuentran almacenados en ElasticSearch, personalizando además la vista de los datos.

En la Figura 2.1 se puede observar una vista de alto nivel del flujo de trabajo de ELK. El proceso comienza cuando Logstash recibe un log, dándole el formato requerido y enviándolo a ElasticSearch para su almacenamiento. Aquí es donde actúa Kibana para ofrecer

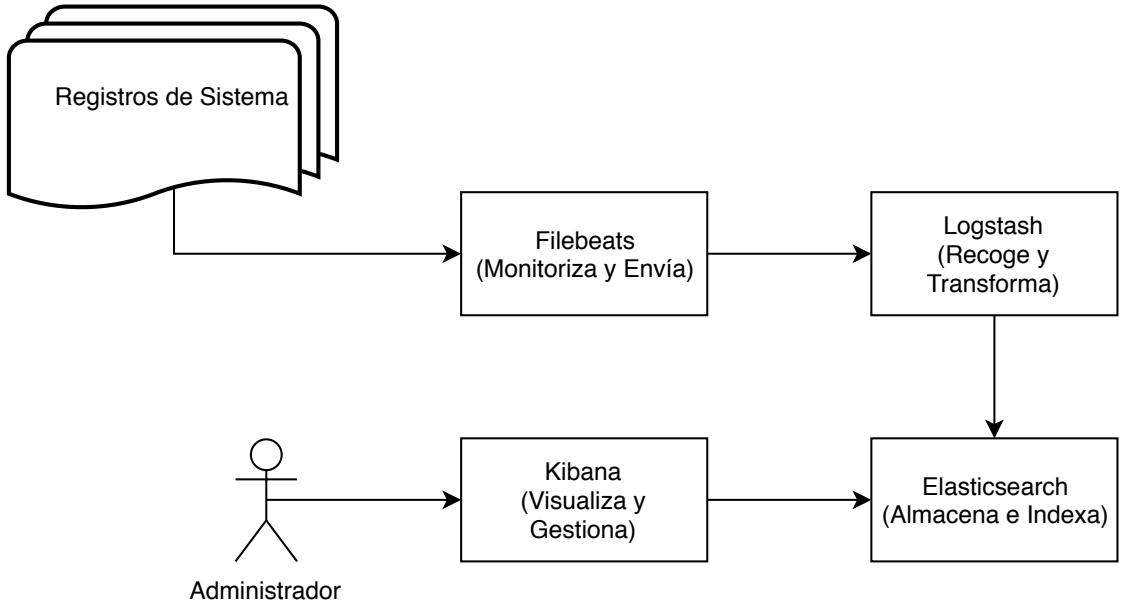


Figura 2.1: Componentes de ELK: ElasticSearch, Logstash y Kibana (adaptada de [11])

la visualización de los datos almacenados. Además, a estos tres pilares (ElasticSearch, Logstash, Kibana) también se le suman dos componentes adicionales:

- **Beats:** Componente para la recopilación de datos entre plataformas y el envío de estos datos a ElasticSearch directamente o a Logstash.
- **X-Pack:** Componente más reciente incorporado en la pila ELK, ofrece algunos plugins como de seguridad, alertas y monitorización de todo el sistema.

Estos dos componentes sumados a ELK forman el sistema completo, conocido como `Elastic Stack`.

2.2. Sistema IDS Snort

Un sistema de detección de intrusiones (IDS, por sus siglas en inglés *Intrusion Detection System*) es un sistema que monitoriza el tráfico de la red en busca de actividad sospechosa, emitiendo alertas cuando se descubre. Si bien la detección y notificación de anomalías es su función principal, algunos IDS son capaces de tomar medidas reactivas cuando se detecta actividad maliciosa o tráfico anómalo, como por ejemplo el bloqueo del tráfico enviado desde direcciones IP sospechosas.

Aunque los sistemas IDS supervisan las redes en busca de actividad potencialmente maliciosa, también son propensos a generar falsas alarmas (es decir, falsos positivos). Esto implica una configuración previa y adecuada de los sistemas IDS para reconocer cómo es el tráfico normal en la red en comparación con una actividad potencialmente maliciosa.

Los IDS tienen distintas funcionalidades y detectan actividades sospechosas utilizando diferentes aproximaciones. Por ejemplo, los *Network IDS* (NIDS) se implementan en uno o más puntos estratégicos dentro de la red, donde pueden monitorizar el tráfico entrante y saliente hacia/desde todos los dispositivos en la red. Del mismo modo, los sistemas *Host IDS* (HIDS) se ejecutan en todas las computadoras o dispositivos de la red con acceso directo tanto a Internet como a la red interna de la empresa, detectando paquetes de red anómalos que se originan dentro de la organización.

Atendiendo a su modo de funcionamiento, se distinguen también dos tipos de IDS. Por un lado, los *IDS basados en firmas* supervisan todos los paquetes que atraviesan la red y los compara con una base de datos de firmas o atributos de amenazas maliciosas conocidas. Por otro lado, los *IDS basados en anomalías* monitorizan el tráfico de la red y lo comparan con una línea base establecida, para determinar qué se considera normal o anómalo [15].

Los IDS también se categorizan como pasivos o activos. Un *IDS pasivo* detecta actividad maliciosa y genera entradas de alerta o de registro, pero no toma ninguna medida. Un *IDS activo* (a veces llamado *sistema de detección y prevención de intrusiones*) genera alertas y entradas de registro, pero también es capaz de tomar medidas adicionales, como bloquear direcciones IP o cerrar el acceso a recursos restringidos.

En este trabajo se hace uso del sistema IDS Snort[1]. Snort es un NIDS de código abierto que se usa para detectar amenazas emergentes y es compatible con los sistemas operativos basados en UNIX/Linux y Windows. Esta herramienta de detección de intrusiones puede ser desplegada para monitorizar redes TCP/IP y detectar una gran variedad de tráfico de red sospechoso, así como ataques directos [14, 16].

Snort es una herramienta que puede ser muy útil como parte de una infraestructura de seguridad de red integrada. Además, supone una buena alternativa cuando no es rentable económicamente desplegar otros sensores NIDS comerciales. Snort está disponible bajo licencia GNU, y es de uso gratuito en cualquier entorno.

2.3. Diagramas UML

El lenguaje unificado de modelado (UML, por sus siglas en inglés) define un lenguaje gráfico que sirve para visualizar, especificar, construir y documentar los elementos de un sistema. Desde el año 2004, es un estándar aprobado por la Organización Internacional de Normalización (más conocida como ISO -*International Standard Organization*-, por sus siglas en inglés). UML sirve como soporte para todo el proceso de diseño, desarrollo y mantenimiento del software independientemente de la plataforma sobre la que se desarrolle. Actualmente, se encuentra en su versión 2.5. UML define trece tipos diferentes de diagramas que sirven para describir las vistas que un modelo puede necesitar para ser representado, enfocados desde el paradigma Orientado a Objetos (OO). Los trece tipos se agrupan en tres bloques: los diagramas estructurales, los diagramas de comportamiento y los diagramas de interacción.

[1] `https://www.snort.org/`

En primer lugar, los diagramas estructurales son aquellos que muestran la estructura estática de lo objetos de un sistema. Estos diagramas corresponden a los diagramas de clases, los diagramas de componentes, los diagramas de despliegue, los diagramas de objetos, los diagramas de paquetes, los diagramas de perfiles y por último, los diagramas de estructura compuesta. En segundo lugar, los diagramas de comportamiento son aquellos que muestran el comportamiento dinámico de los objetos en el sistema. Estos diagramas son los diagramas de actividades, los diagramas de casos de uso y los diagramas de máquina de estados. Por último, los diagramas de interacción muestran las interacciones entre objetos, actores y otros elementos del sistema. En este último grupo se incluyen los diagramas de secuencia, los diagramas de comunicación, los diagramas de tiempo y los diagramas globales de interacciones.

En este proyecto, se han usado los diagramas de clases y los diagramas de secuencia. A continuación, se explican en detalle estos diagramas.

2.3.1. Diagrama de casos de uso

Un diagrama de clases de uso da una vista general de los casos de uso a tener en cuenta en un sistema. Se confunde con frecuencia con los propios casos de uso. A diferencia de estos, el diagrama sólo proporciona una vista gráfica y general de los casos de uso, siendo la definición de caso de uso la que aporta información clara y concisa sobre cómo se ha de comportar el sistema ante una tarea. Los elementos principales de un diagrama de clases de uso son los actores, la entidad, las relaciones y los casos de uso.

Los actores se entienden como un rol, ya sea de un usuario humano u otra entidad externa al sistema, y se representan simplificadamente con un símbolo de persona. La entidad es el propio sistema del cual se representan sus casos de uso, representado como un rectángulo que engloba los casos de uso, con el nombre en la parte superior del mismo. Las relaciones se representan con líneas rectas y unen tanto actores con casos de uso, como casos de uso con otros casos de uso. En el caso de las relaciones entre casos de uso, UML define tres relaciones principales: inclusión, extensión y generalización. Los casos de uso se representan con una elipse con su nombre en el interior.

Las relaciones de inclusión (usualmente en inglés en los diagramas, *include*) representan aquellas situaciones en que un caso de uso dado "incluye" otro caso de uso (es decir, hay una relación de dependencia entre lo casos de uso). Se representan con una recta discontinua entre los casos de uso con una punta de flecha abierta en el extremo del caso de uso incluido. Las relaciones de extensión (usualmente en inglés en los diagramas, *extend*) son útiles para casos especiales o para incluir nuevos requisitos del sistema que aparecen durante su mantenimiento o desarrollo posterior. Como antes, se representan con una recta discontinua entre los casos de uso con una punta de flecha abierta en el extremo del caso de uso extendido. Las relaciones de generalización se usan para definir las relaciones entre una superclases o un concepto general y una subclase o concepto especializado. Su representación es una línea continua terminada en un triángulo dibujado desde el caso de uso especializado al caso de uso general.

La Figura 2.2 muestra un diagrama de casos de uso desarrollado en la asignatura "Ingeniería del Software" de la carrera de Graduado en Informática de la UZ. En ella se

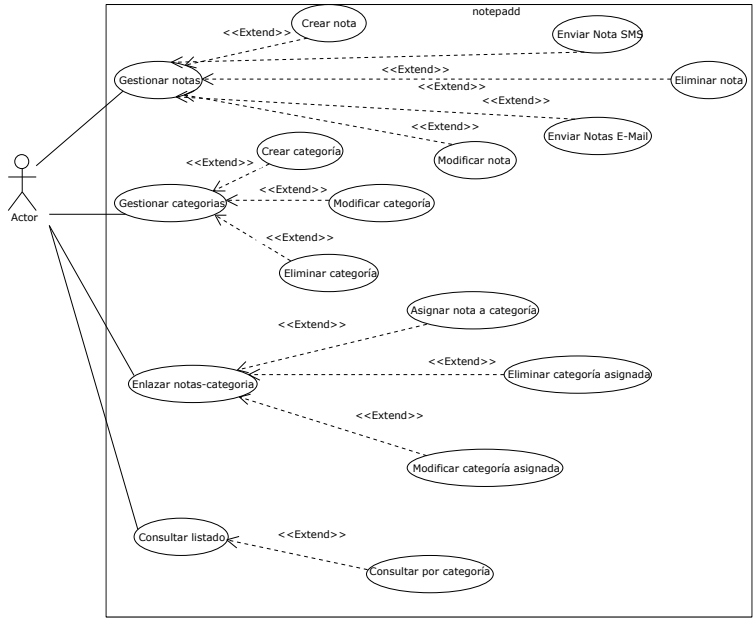


Figura 2.2: Ejemplo de Diagrama de Caso de Uso

representa un actor humano llamado Usuario que interactúa con la entidad *notepadapp*, una aplicación de gestión de notas. En esta aplicación se representan los distintos casos de uso del sistema con relaciones de extensión entre los casos de uso básicos y los casos de uso específicos que representan acciones concretas.

2.3.2. Diagrama de secuencia

Un diagrama de secuencia muestra los módulos o clases que forman parte del programa, así como las llamadas que se hace en cada uno de ellos para realizar una tarea determinada. Los diagramas de secuencia se realizan para definir acciones que se pueden realizar en la aplicación en cuestión. Los objetos se representan con rectángulos, los mensajes se representan con líneas continuas con una punta de flecha en el extremo del receptor, y el tiempo se representa en forma de una progresión vertical; es decir, lo que está más arriba en el diagrama es algo que sucede antes que lo que está representado más abajo. Los mensajes pueden ir de la línea de vida de un objeto a otro, o al mismo objeto que los crea. Los mensajes pueden ser simples, síncronos o asíncronos. Un mensaje simple es la transferencia del control de un objeto a otro. Un mensaje síncrono es aquel en que el objeto emisor espera la respuesta antes de continuar su trabajo. Por último, un mensaje asíncrono es aquel en que el objeto emisor no espera a la respuesta para continuar.

La Figura 2.3 muestra el diagrama de secuencia de un borrado de un grupo en una aplicación de colecciones de vinos, desde que el usuario abre la pantalla de borrado de grupos en su aplicación móvil, hasta que se le muestra la lista actualizada. En este caso se trata de una secuencia de mensajes que inician mensajes a los distintos subsistemas y

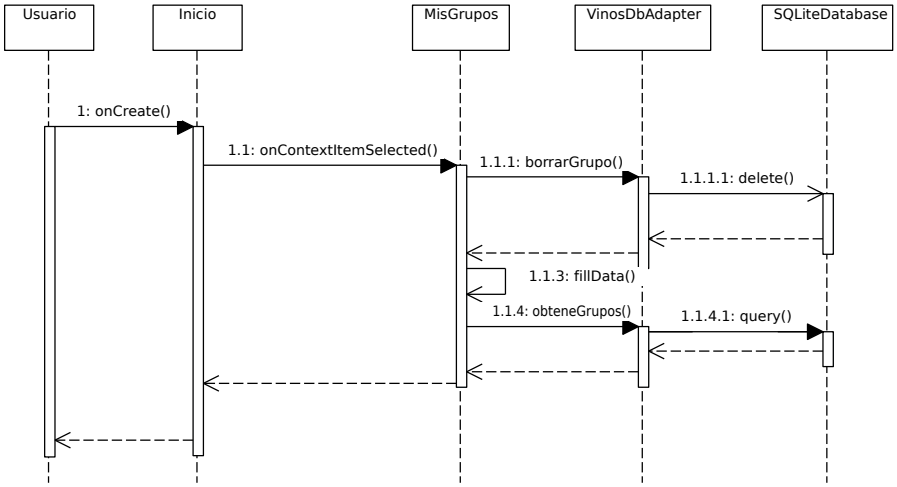


Figura 2.3: Ejemplo de Diagrama de Secuencia

quedan a la espera de la respuesta para continuar la ejecución. En el caso de la acción 1.1.3: *fillData()* se ve un mensaje que parte de su línea de vida para llegar a su misma línea de vida.

Capítulo 3

Tipos de ataque contemplados y herramientas

Por limitaciones de alcance y de tiempo, en el sistema desarrollado se han limitado los ataques contemplados. En concreto, sólo se trata la detección de los conocidos como ataques de denegación de servicio (más conocidos por sus siglas en inglés, *DoS*) y los ataques de inyección SQL (en inglés, *SQL Injection*). En este capítulo, en primer lugar se describen estos ataques. Después, se detallan las herramientas utilizadas en este trabajo para la generación de logs sintéticos que recojan este tipo de ataques de cara a la experimentación del sistema desarrollado.

3.1. Ataques

3.1.1. Ataques de denegación de servicio

Un ataque de denegación de servicio es un ataque a un sistema de computadoras o red que causa que un servicio o recurso sea inaccesible a los usuarios legítimos del mismo. Normalmente, se provoca la pérdida de conectividad de red por el consumo del ancho de banda de la red de la víctima debido a la sobrecarga de los recursos computacionales del sistema atacado. Una ampliación del ataque DoS es el llamado ataque de denegación de servicio distribuido (*DDoS* por sus siglas en inglés), que se lleva a cabo generando un gran flujo de información desde diferentes puntos de conexión hacia un mismo punto de destino [5].

Los métodos más comunes para la realización de estos ataque son el envío de paquetes malformados a la víctima para confundir al protocolo o aplicación que está corriendo (ataques que se aprovechan de una vulnerabilidad conocida), o el envío masivo de paquetes en nivel de red/transporte o en nivel de aplicación.

Los ataques de denegación de servicio se pueden clasificar según la capa de red en la que funcionen, teniendo en primer lugar los que trabajan en capa de red o transporte y por otro lado los que trabajan en la capa de aplicación.

Los ataques que funcionan en capa de red o transporte usan los protocolos TCP,

UDP, ICMP y DNS. Estos ataques se pueden sub-clasificar a su vez en cuatro subtipos [19]: *Packet flooding attacks*, centrados en en interrumpir la conectividad del usuario mediante agotamiento del ancho de banda de la red; *Procotol exploitation flooding attacks*, centrados en explotar diferentes características o fallos de implementación de algunos de los protocolos de la víctima para consumir grandes cantidades de sus recursos (TCP SYN flood, TCP SYN-ACK flood, ACK & PUSH ACK flood, RST/FIN flood, etc.); los *Reflection-Based flooding attacks*, consistentes en el envío de peticiones falsificadas (e.g., ICMP echo request) a otros sistemas que envían sus respuestas a la víctima y consumen así sus recursos; y por último, los *Amplification-based flooding attacks*, donde los atacantes explotan servicios para generar mensajes largos o múltiples mensajes por cada mensaje que reciben para amplificar el tráfico hacia la víctima.

Por otro lado, los ataques que trabajan en la capa de aplicación son ataques que buscan la interrupción del servicio legítimo mediante el agotamiento de los recursos del servidor. Estos también están organizados según su metodología en: los *Reflection/amplification based flooding attacks*, similares a *Reflection-Based flooding attacks* y a los *Amplification-based flooding attacks*, aunque en este caso el ataque se realiza en la capa de aplicación; y los *HTTP flooding attacks*, que se dividen según el tipo de "inundación" que realizan: los *Session flooding attacks*, consistentes en aumentar la tasa de peticiones de conexión de sesión; los *Request flooding attacks*, en los que los atacantes envían sesiones que contienen más peticiones de lo habitual y provocan un *DDoS flooding attack* en el servidor; los *Asymmetric attacks*, donde atacantes envían peticiones que generan una alta demanda de recursos; y por último, los *Slow request/response attacks*, en los que los atacantes envían sesiones que contienen peticiones con alta demanda de trabajo.

3.1.2. Inyección SQL

Un ataque de inyección SQL (en inglés, *SQL Injection* y sus siglas SQLi) es un ataque a un sistema web que provoca que datos contenidos en una base de datos sean vistos o modificados por usuarios no autorizados a ello. Esto se consigue mediante la inserción de sentencias SQL en campos de un formulario. Con estos ataques se puede conseguir eludir los mecanismos de autenticación y autorización de una aplicación web y recuperar el contenido de una base de datos completa. La inyección SQL también se puede usar para agregar, modificar y eliminar registros en una base de datos, lo que afecta la integridad de los datos [1].

Las vulnerabilidades de inyección SQL se producen debido a una validación insuficiente de la entrada del usuario. Para solventar este problema, se han propuesto diversas buenas prácticas de desarrollo como la codificación defensiva o a la codificación de entrada y validación de datos provenientes del usuario.

Los diferentes tipos de ataques SQLi no se realizan de forma aislada, si no que muchos de ellos se usan juntos o secuencialmente, dependiendo en los objetivos específicos del atacante. Hay ataques basados en tautologías, donde el objetivo general es inyectar código en una o más sentencias condicionales para que siempre sean evaluadas como verdadero (de esta manera son capaces de superar el proceso de autenticación y extraer los datos). Hay también, ataques basados en consultas lógicamente incorrectas pero que

Herramienta	Tipo de Ataque	Anónima
LOIC	*Packet Flooding Attack, Request flooding attack*	NO
HOIC	*Request flooding attack*	SI
hping	*Reflection-Based flooding attack*	SI
Slowloris	*Slow request/response attack*	SI
R u Dead Yet? (R.U.D.Y.)	*Slow request/response attack*	NO
#Refref	*Session flooding attack*	SI
HULK	*Asymmetric attack, Slow request/response attack*	SI
DDOSIM-Layer7	*Protocol exploitation flooding attack*	SI

Tabla 3.1: Herramientas de ataque DDoS

permiten recolectar información importante sobre el tipo y la estructura de la base de datos de una web. Los ataques basados en unión de consultas permiten cambiar el conjunto de datos que se devuelven en una consulta. Con esta técnica, un atacante puede engañar a la aplicación para devolver los datos de una tabla de manera diferente a como estaba diseñado. También existen los ataques basados en consultas *Piggy-Backed* donde el atacante intenta inyectar consultas en la consulta original. Estos ataques se distinguen de otros explicados en este trabajo porque el atacante no está tratando de modificar el resultado de la consulta original, sino que trata de anidar consultas en una consulta legítima. Existen también los ataques basados en procedimientos almacenados en la base de datos para realizar una escalada de privilegios, una denegación de servicio, o la ejecución de comandos remotos. Hay también ataques de inferencia, donde el atacante modifica una consulta para transformarla en una acción que se ejecuta basada en una respuesta a una pregunta de verdadero o falso. Por último, se encuentran los ataques de codificaciones alternativas, en las que el texto inyectado se modifica para evitar la detección [7].

Un tipo diferenciado de ataque SQLi es la técnica conocida como inyección SQL ciega (más conocida en inglés como *blind SQLi*). Se diferencia de las anteriores en que no se muestran mensajes de error al no producirse resultados correctos ante una consulta a la base de datos, mostrándose siempre el mismo contenido.

3.2. Análisis de herramientas para la generación de ataques

Para la creación de los datos para entrenar y evaluar el sistema desarrollado en este trabajo se ha recabado información sobre varias herramientas.

Para los ataques DDoS, se han investigado varias herramientas que automatizan dichos ataques. Hay multitud de herramientas en este ámbito dado que la metodología de este vector de ataque es muy simple, ya que consiste en el envío de multitud de paquetes al objetivo. Se considera importante a la hora de determinar las herramientas a utilizar en el proyecto el tipo de tráfico que generan en el ataque, si anonimizan el origen del ataque y el tipo de ataque que realizan según la clasificación explicada. En la Tabla 3.1 se analizan las 7 herramientas pre-seleccionadas para ser utilizadas. Otro método de ataque usual en un ataque DDoS es la utilización de una red controlada de ordenadores de terceros (*botnet*) como herramienta de ataque. Independientemente de las herramientas

de ataque DDoS utilizadas, la capacidad de lanzar un ataque desde cientos, miles o millones de computadoras amplifica significativamente el potencial del ataque para causar la denegación de servicio, razón por la cual las *botnets* son usadas comúnmente para un ataque DDoS [13].

Elección de herramientas. Para la generación de ataques de muestra se han elegido dos de las anteriores:

- **LOIC:** Se ha escogido esta herramienta por ser una de las usadas en los ataques de denegación de servicio más famosos ocurridos últimamente y servir como base de otras herramientas similares. La versión utilizada ha sido la 1.0.8.
- **HULK:** Se ha escogido esta herramienta debido a su simplicidad de uso y su potencia a la hora de generar el ataque. Se ha utilizado la versión 1.0.

En cuanto a los ataques SQLi, se han considerado tres herramientas. En primer lugar, `sqlmap`, que es la herramienta más famosa entre las herramientas de ataque SQLi. En segundo lugar, `BSQL Hacker`, especializada en los ataques *blind SQLi*. Y por último, `SQLNinja`, cuya particularidad es la posibilidad de integración con `Metasploit` (se trata de un proyecto de código abierto usada en labores de auditoría de sistemas para intentar explotar vulnerabilidades de un equipo).

Elección de herramientas. En este caso se ha elegido sólo la herramienta `sqlmap`, dado que se trata de una de las herramientas más famosas y usadas en el campo de la seguridad de redes. En el caso de la inyección SQL, la metodología que usa la herramienta es similar a la utilizada por un atacante: realiza el intento continuado de inserción de instrucciones SQL en los campos de un formulario del servicio web, usando además ataques de fuerza bruta con múltiples instrucciones SQL diferentes.

Capítulo 4

Diseño e implementación del sistema

En este capítulo se detalla toda la información del sistema desarrollado en este trabajo. Tras una breve descripción del sistema completo, se detalla cada subsistema en particular.

El sistema completo se divide en tres partes con funciones diferenciadas. La primera parte es el sistema que almacena los logs de sistema. La segunda parte es el sistema desarrollado durante el proyecto que abarca tanto la monitorización de los sistemas objetivo como el sistema de detección de ataques en los logs y el sistema que determina la actuación en base a la predicción. Por último, el sistema IDS que recibe las instrucciones del sistema de detección.

En la Figura 4.1 se muestra un diagrama de alto nivel del sistema descrito. En primer lugar, el conjunto de servidores que envían sus logs al sistema ELK. A continuación, el sistema generador de reglas que extrae logs de ELK y los analiza para actuar según la política establecida. Por último, el sistema IDS que se encarga del control del tráfico entrante desde la red externa al sistema.

En la Figura 4.2 se observa el comportamiento del sistema mediante un diagrama de secuencia UML. Dado que el sistema usa técnicas de aprendizaje supervisado, antes de ponerlo a funcionar será necesario entrenarlo. En primer lugar, se lanza el proceso que observa el fichero que almacena los logs de la aplicación web ante cambios. Mientras tanto, el sistema queda a la espera de recibir un nuevo fichero log para analizar. Cuando el observador detecta cambios en el fichero se avisa al sistema que extrae los nuevos logs del sistema ELK. Una vez extraídos se procede a realizar la clasificación y la actuación en consecuencia del resultado.

4.1. Tecnologías utilizadas

Desde un primer momento se diseñó un sistema que fuera lo más independiente de la tecnología posible, aunque dependiendo en todo caso de un sistema ELK. Por tanto, el sistema de monitorización y el sistema de detección son independientes de la tecnología a la hora de generar las reglas, ya que este último apartado del sistema está desarrollado de manera modular dejando abierta la posibilidad de ampliar su funcionamiento para otras tecnologías distintas a las expuestas en el Capítulo 5. Todo el sistema desarrollado

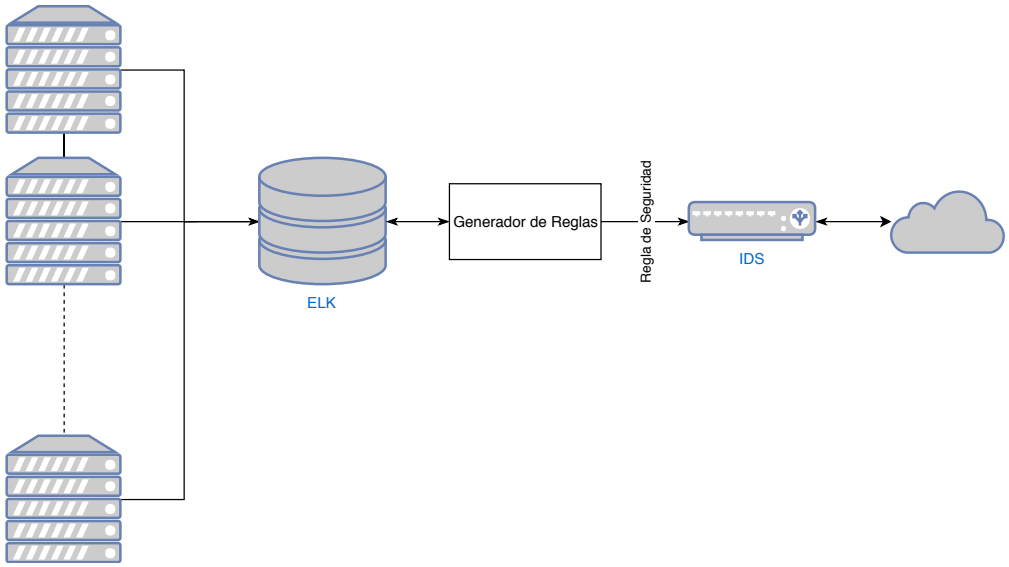


Figura 4.1: Vista general del sistema

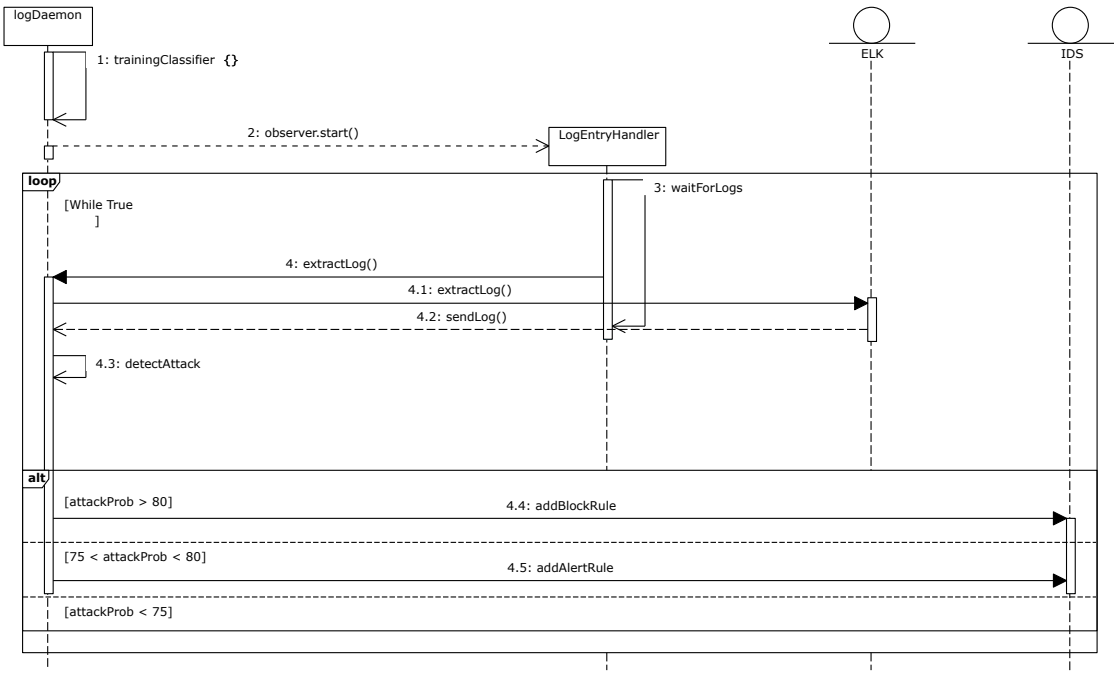


Figura 4.2: Diagrama de Secuencia del Sistema.

se encuentra implementado con Python en su versión 2.7, haciendo uso de distintos módulos ya implementados para facilitar el desarrollo del sistema.

4.2. Sistema de almacenamiento de logs

El sistema hace uso de una pila ELK (véase la Sección 2.1) para el almacenamiento de logs. Esta herramienta como ya se ha explicado permite una fácil gestión de los logs de sistema generados por distintos servicios. Esta herramienta dispone de una API de Python que facilita la gestión de los logs almacenados en ella. Dicha herramienta no permite de forma nativa conocer si hay nuevos logs para analizar. Por tanto, como se explica en la sección siguiente es necesario el uso de módulos propios de Python para dar solución a este problema.

4.3. Sistema de monitorización de logs y detección de logs

Este sistema se apoya en el módulo `watchdog` de Python [12], cuya labor es monitorizar un archivo o directorio en busca de cambios, dado que la API de ELK no aporta una solución a la monitorización de índices en busca de nuevas entradas. Este módulo provee al desarrollador de herramientas de monitorización para uno o varios directorios, permitiendo configurar qué ficheros se han de monitorizar, así como los eventos que se tienen que desencadenar.

En el caso de este proyecto, se ha determinado que los eventos a controlar son tanto la creación de nuevos ficheros como la modificación de ficheros existentes, atendiendo sólo a los ficheros log de `Apache`. Para ello, monitorizan los ficheros con extensión *".log"*. La acción a realizar ante la creación y la modificación del fichero monitorizado es la misma. Concretamente, la acción desencadenada ante estos eventos consiste en extraer de los logs los registros de los últimos cinco minutos, crear una estructura interna para facilitar el trabajo posterior, almacenando en ella los campos considerados necesarios, y encolarla en una cola con orden FIFO a la espera de que el sistema de detección los analice.

4.4. Sistema de detección de ataques

Para detectar los ataques se evalúan las distintas entradas de los logs almacenados en el servidor `Apache` por cada petición web. Cada vez que un usuario accede a una aplicación web, se realiza una petición, usualmente de tipo HTTP GET y POST, que queda registrada en el servidor. Las peticiones POST por defecto no se registran en los servidores `Apache` por motivos de rendimiento, ya que esto implicaría un agotamiento de los recursos de almacenamiento. Las peticiones GET, al contrario, dejan su contenido completo en el log del servidor. Una petición GET incluyen información del usuario que realiza la petición como la dirección IP de origen de la petición, día y hora, recurso sobre el que se ha realizado, datos sobre desde que software se realiza la petición (llamado *user-Agent* o *Agent*), información sobre las cookies de sesión, etc.

Como ejemplo de una cabecera HTTP GET, se ha conectado a la página web de la Universidad de Zaragoza (`www.unizar.es`) para observar las cabeceras. En la Figura 4.3 se muestra el acceso a la web, mientras que en la Tabla 4.1 se muestran algunos campos de esta cabecera.

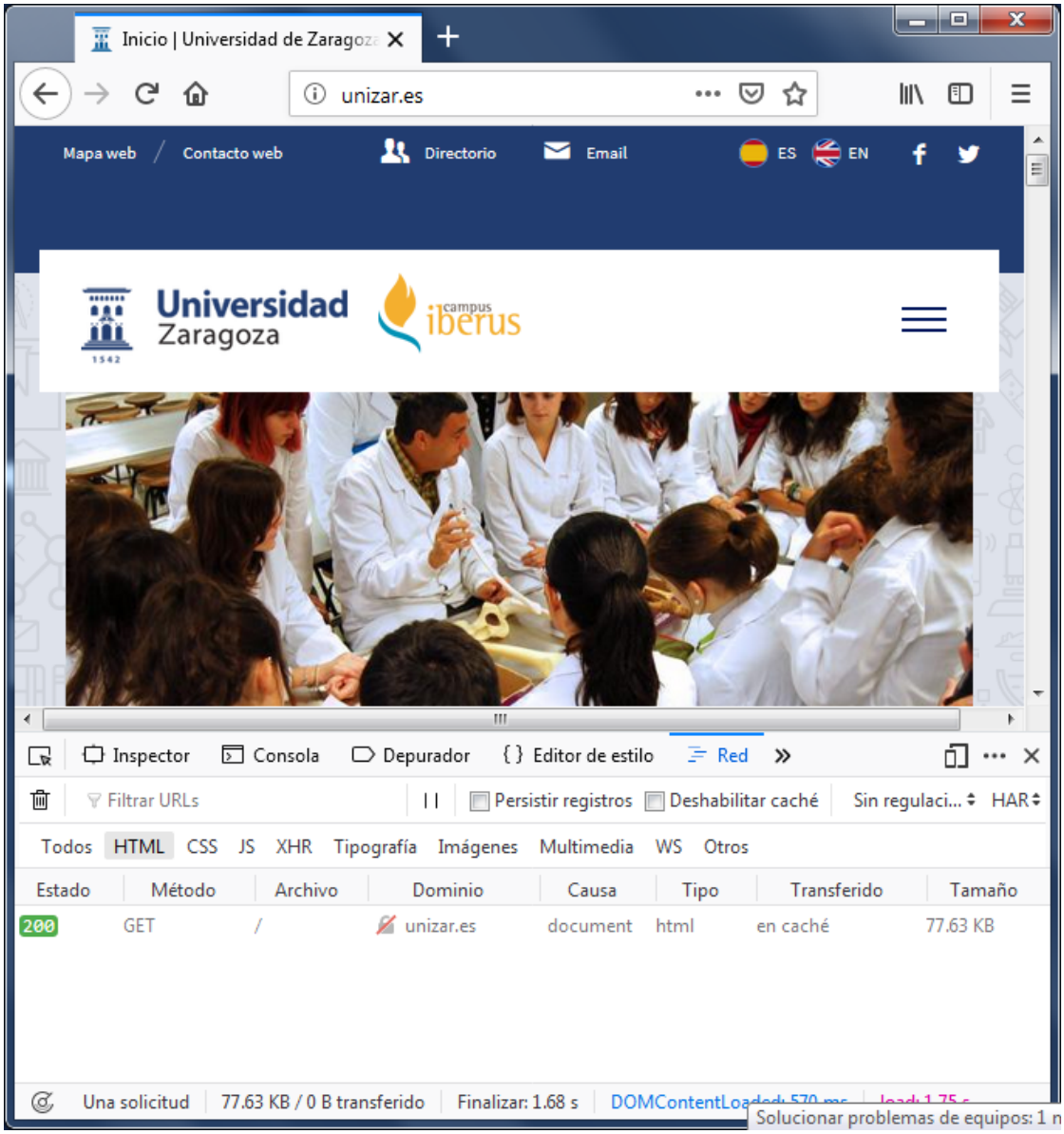


Figura 4.3: Petición sobre la web institucional

Para el sistema detector de ataques se han contemplado sólo varios campos concretos de las peticiones: el campo *Agent*, que aporta información del software utilizado para realizar la petición (Firefox, Chrome,...); el campo *Referrer*, que indica desde que URL

Cabecera	Valor
URL solicitada	htttp://www.unizar.es
Método de la petición	GET
Dirección remota	155.210.11.37:80
Código de estado	200 OK
Versión	HTTP/1.1

Tabla 4.1: Cabeceras principales de la petición anterior

Cabecera	Valor
Accept	text/html.application/xhtml+xm.../xml:q-0.9,*/*;q=0.8
Accept-Encoding	gzip, deflate
Accept-Language	es-ES,es;q-0.8,en-US;q-0.5,en:q-0.3
Connection	keep-alive
Cookie	https%3a%2f%2frrhh.unizar.es%2...dashboard; has_js-1
Host	www.unizar.es
If-None-Match	"1537345731-1"
Referer	"http://www.unizar.es/"
Upgrade-Insecure-Requests	1
User-Agent	Mozilla/5.0(X11, Linux x86_64...) ...Firefox/60.0

Tabla 4.2: Cabeceras en detalle de la petición anterior

llega la petición; los campos *Continent_Code* y *Country_Name*, extraídos del *plugin* `geoip` de Logstash que aportan información del continente y país desde donde se realiza la petición, respectivamente; y por último, el campo *response*, que indica la respuesta del servidor a dicha petición.

El sistema de detección de ataques consta de dos partes diferenciadas, la primera consiste en el uso de un clasificador para los campos *Referrer* y *Agent* de las peticiones HTTP y la segunda en el uso de reglas para el resto de campos contemplados. Cada una de estas partes se explican a continuación.

4.4.1. Clasificación

El problema de clasificación abordado durante el transcurso de la titulación corresponde a la clasificación de emails de spam. Este problema es similar a la clasificación de logs en peticiones de ataque o peticiones legítimas, dado que en ambos casos se tiene en cuenta las apariciones de palabras para determinar si el elemento analizado es benigno o no. En el caso de la clasificación de emails, se evalúa todo el texto del mensaje para valorar la posibilidad de que sea spam o no, y en el caso de peticiones, se evalúa cada campo por separado para realizar determinar si se trata de peticiones legítimas o no. Para ello, se ha utilizado el paquete de Python `Scikit-learn`. En primer lugar, se realizó una batería de ataques en un entorno controlado para obtener datos de entrenamiento. Con estos datos, se usó el 80 % como datos de entrenamiento para el clasificador, el 10 % para

la fase de validación y el 10 % restante se guardó como datos de prueba a usar tras el entrenamiento y la validación.

Con esta división de datos se procede a entrenar el clasificador con los datos de entrenamiento. El clasificador hará uso del algoritmo de validación cruzada (explicado a continuación) para determinar el hiper-parámetro de suavizado de Laplace. El hiper-parámetro de suavizado de Laplace sirve al clasificador para que los resultados con valor cero no provoquen problemas en los cálculos de la clasificación. Una vez calculado este hiper-parámetro, se implementan cuatro variaciones del clasificador *Naive Bayes* utilizado en este sistema. Un clasificador Bayesiano ingenuo (más conocido como *Naive Bayes*) es un clasificador probabilista basado en el teorema de Bayes que supone existencia de independencia entre los predictores. La primera variación supone una distribución Multinomial y el uso de bolsa de palabras. La segunda variación supone una distribución Multinomial y el uso de bigramas. La tercera variación supone una distribución Bernoulli y el uso de bolsa de palabras. La última variación supone una distribución Multinomial y el uso de bigramas. Nótese que la bolsa de palabras guarda el número de apariciones de una palabra. En cambio, en el uso de bigramas se guarda el número de apariciones de cada bigrama. Un bigrama es un grupo de dos letras, dos sílabas, o dos palabras. Los bigramas son utilizados comúnmente como base para el simple análisis estadístico de texto.

Algoritmo de validación cruzada. Este algoritmo funciona de la siguiente manera. En primer lugar, se establece un valor de k y se divide el conjunto de datos en K subconjuntos aproximadamente del mismo tamaño. A continuación, el primer subconjunto se trata como un conjunto de validación, y el método se ajusta a los $k-1$ restantes. En segundo lugar, se calcula el error cuadrático medio (MSE) con los datos del subconjunto usado como validación. Este proceso se repite K veces. En cada iteración, se elige un subconjunto diferente como conjunto de validación. Este proceso da como resultado k estimaciones del error de prueba $MSE_1, MSE_2, ..., MSE_k$. Por último, la estimación del algoritmo se calcula realizando un promedio de estos valores:

$$CV(k) = \frac{1}{k} \sum_{i=1}^{k} MSE_i \tag{4.1}$$

Mediante este proceso se calcula el λ para la primera variación y se aplica al resto. Todo el procedimiento se repite 10 veces con los distintos valores de λ que se quieren probar, con el objetivo de afinar la exactitud del clasificador para dicho valor de λ [18].

El entrenamiento funciona de la siguiente manera. En primer lugar, con los logs de entrenamiento escogidos se crea una bolsa de palabras con el contenido del campo *Agent* y otra bolsa con el contenido del campo *Referrer*. Una vez definidas las bolsas de palabras, se procede con el entrenamiento del clasificador, el cual aprende a calcular si se trata de una petición legítima o de un ataque. Para ello, se almacena para cada palabra su probabilidad de aparición en un log de ataque o legítimo. Una vez que se han recorrido

todas las palabras, se obtiene una probabilidad final de que se trate de un ataque o una petición legítima, pudiendo entonces clasificarlo en la categoría de mayor probabilidad.

Para evaluar la exactitud del clasificador entrenado, se usan los logs de tests tras finalizar el entrenamiento. En este momento, se crean bolsas de palabras con los contenidos de los campos en los log de test y se realiza una predicción. Posteriormente, con la predicción obtenida y el etiquetado previo se realizan las comprobaciones oportunas para estimar la exactitud del clasificador. Para ello, se comprueba tanto el porcentaje de acierto como la cantidad de falsos positivos y negativos obtenidos en el proceso. Con estos datos a la vista se selecciona la variación con mejor resultado.

Con este procedimiento de entrenamiento y elegida la variación que se usará finalmente para el clasificador se obtiene el sistema de clasificación del sistema, cuya labor será determinar la clasificación de ataque para los campos *Agent* y *Referrer* para en colaboración con el sistema de reglas realizar la clasificación final de la petición.

4.4.2. Sistema de reglas

El sistema de reglas toma en consideración los campos *Continent_ Code*, el campo *Country_ Name*, el campo *response*, y por último, un conteo de las peticiones web realizadas desde una misma dirección IP.

Para el campo *Continent_ Code*, se evalúa si la petición proviene de fuera de la UE, estableciendo así que el uso habitual de los servicios web contemplados en el sistema bajo estudio es desde dentro de la UE y, por tanto, una petición desde fuera de la UE se considera como un factor de riesgo. Para el campo *country_ name*, se evalúa si la petición proviene de una lista de países determinados, dado que según el portal Statista, países como Estados Unidos, Holanda, China, Brasil y Rusia son los países origen de la mayoría del tráfico de ataques del mundo [17]. Para el campo *response*, se valora si la petición devuelve el código 200 (petición correcta) o no.

Este sistema de reglas se entrelaza con el clasificador de la siguiente manera: primero, se realiza una clasificación de los campos *Agent* y *Referrer*. Cada una de estas predicciones se pondera con un 25 % en la clasificación final. El sub-sistema de reglas pondera la evaluación de los otros campos en otro 25 %. Por último, se cuentan las apariciones de peticiones desde una misma dirección IP en el lapso de tiempo de una hora. Si se superan las 50 peticiones en menos de una hora, se cataloga como un posible ataque y se pondera con otro 25 %. En caso contrario, se pondera con un 0 %.

4.5. Sistema de actuación

Con los resultados obtenidos por el sistema de detección, se establece una política de actuación según sea el porcentaje asignado a la petición. Para ser considerado una tentativa de ataque, el resultado del detector debe ser superior al 75 % . Si está comprendido entre el 75 % y el 80 %, el sistema generará una nueva regla de alerta en el sistema IDS, de modo que dicha petición será registrada y monitorizada en el IDS. Si el resultado es superior al 80 %, entonces la petición se cataloga como ataque y se genera una regla de

Sistema de reglas	
Clasificación positiva del campo *Agent*	25 %
Clasificación positiva del campo *Referrer*	25 %
Resto de campos:	25 %
Continent_ Code != UE	8.33 %
Country_ name == EEUU || Holanda || China || Brasil || Rusia	8.33 %
Response != 200	8.33 %
+50 peticiones en 60 minutos	25 %

Tabla 4.3: Ponderaciones del sistema de reglas

bloqueo. En caso de que no se llegue al 75 %, el sistema no actúa y prosigue la normal ejecución.

En la fórmula siguiente se define matemáticamente la acción a llevar a cabo según el porcentaje obtenido, donde r viene dado por la suma de las ponderaciones establecidas según la Tabla 4.3.

$$Acción = \begin{cases} Bloqueo, & si \quad r \geq 80\,\% \\ Alerta, & si \quad 75\,\% \leq r < 80\,\% \\ Normal, & si \quad r < 75 \end{cases}$$

Capítulo 5

Caso de estudio: Moodle Unizar

En este capítulo se explica el caso de estudio usado para evaluar el sistema desarrollado. En concreto, este sistema se ha desplegado sobre una copia (parcial) de la plataforma Moodle de apoyo docente usada por la Universidad de Zaragoza. Es conveniente destacar que la plataforma Moodle es el sistema de gestión de aprendizaje de distribución libre usado como apoyo docente en la Universidad de Zaragoza, que además utiliza una base de datos SQL para almacenar los datos de la aplicación. Esta plataforma se despliega sobre un servidor Apache, cuyos logs servirán de entrada al sistema desarrollado.

Debido a la criticidad del servicio que proporciona Moodle se decidió que no era ni óptimo ni aconsejable desplegar el sistema sobre la plataforma real, aunque hubiera aportado una mayor cantidad de información para análisis. Además, debido a la infraestructura de red de la Universidad de Zaragoza, la generación de reglas en un IDS real en producción no es viable por la posible interferencia con la oferta de servicios de la universidad. Por este motivo, se ha puesto en marcha una versión reducida del Moodle oficial en un servidor que simula el servidor real, con una base de datos con únicamente los datos de un usuario debido al limitado almacenamiento disponible. De esta manera, se crea un Moodle cuyo funcionamiento es equivalente al Moodle oficial pero con capacidades limitadas.

5.1. Entorno de experimentación

La herramienta Moodle permite a docentes y alumnos realizar interacciones entre ellos, tales como entrega de tareas, la puesta a disposición de los alumnos de las diapositivas de una asignatura, envío de mensajes, etcétera. Dichas interacciones son susceptibles de ser sustituidas por acciones ilegítimas, bien antes del proceso de autenticación o después del mismo.

En la Figura 5.1 muestra un diagrama de casos de uso de un usuario con la herramienta Moodle. En concreto, se permiten realizar las siguientes interacciones: para los usuarios estudiantes, está permitido subir sus tareas, visualizar cursos y visualizar ficheros; para los usuarios profesores, está permitido visualizar cursos y ficheros, y además, gestionar los ficheros de de la plataforma (subir y eliminar), añadir mensajes al foro del curso y

modificar un curso; por último, el/los administrador/es del sistema, que genera los cursos, los elimina, gestiona los permisos sobre el sistema, y comprueba las configuraciones. Es necesario aclarar que hay más casos de uso no reflejados en el diagrama, o incluso casos de uso que pueden sufrir modificaciones según la implementación propia de cada centro educativo que use la plataforma. En el entorno de este proyecto, se considera suficiente la especificación descrita.

Para explicar el funcionamiento de dicha herramienta se incluyen los siguientes diagramas de secuencia, en los que se puede observar las tareas habituales de un usuario cuando lleva a cabo los casos de uso especificados en el diagrama anterior. La Figura 5.2(a) muestra el proceso normal de entrega de una tarea. En la Figura 5.2(b) se observa el proceso normal de visualización de un curso. En la Figura 5.3(a) se puede ver el proceso normal de visualización de un fichero. En la Figura 5.3(b) se ve el proceso normal de gestión de ficheros. En la Figura 5.4(a) refleja el proceso normal de añadir mensajes al foro de un curso. Por último, en la Figura 5.4(b) se puede ver el proceso normal de modificación de un curso. Por último, en la Figura 5.5 se puede ver el proceso normal de eliminar un curso.

5.2. Escenario 1: Moodle sin el sistema desarrollado

Sobre este sistema se lanzan tres baterías de ataques automatizados, una con cada herramienta de ataque propuesta. En primer lugar se realizaron ataques DDoS con la herramienta `LOIC`, mientras en otra máquina se hacía un uso normal del sistema. Segundo, se realizaron ataques DDoS con la herramienta `HULK`, haciendo también uso normal del sistema en otra máquina. En tercer y último lugar se realiza un ataque SQLi con la herramienta `sqlmap`. Dicho ataque realiza un escaneo sobre la URL destino para detectar posibles vulnerabilidades de SQLi. Una vez que la herramienta detecta alguna inyección de SQL, el atacante puede elegir realizar una extensa toma de características del sistema de administración de la base de datos, recuperar la base de datos, enumerar usuarios, contraseñas, volcado completo de tablas, etc. En este caso, se ha realizado un ataque automatizado de manera completa, lo cual detecta los parámetros de un formulario y realiza ataques automáticos para descubrir el contenido correcto de la petición. Como en los casos anteriores, mientras se realizaba el ataque se estaba haciendo un uso normal del sistema en otra máquina. En este caso, se ha considerado únicamente interesante probar cómo afecta este ataque a la disponibilidad y no a la confidencialidad o integridad de los datos almacenados en la base de datos.

La Figura 5.6 muestra los resultados de las distintas baterías de ataque. Como se puede ver, el uso de `HULK` es el que genera un tiempo de respuesta mayor, siguiendo una tendencia exponencial. En el resto de casos, la tendencia es más lineal. El tiempo de respuesta en el caso de la herramienta `LOIC` es cercano a los 2 segundos de media, mientras que el uso de `sqlmap` no incrementa de manera significativa el tiempo de respuesta (normalmente, de media 0,15 ms).

En la Figura 5.7 se muestra el tiempo medio de respuesta ante cada ataque. Como se comentaba en la figura anterior, resulta evidente que los ataques de DoS incrementan el

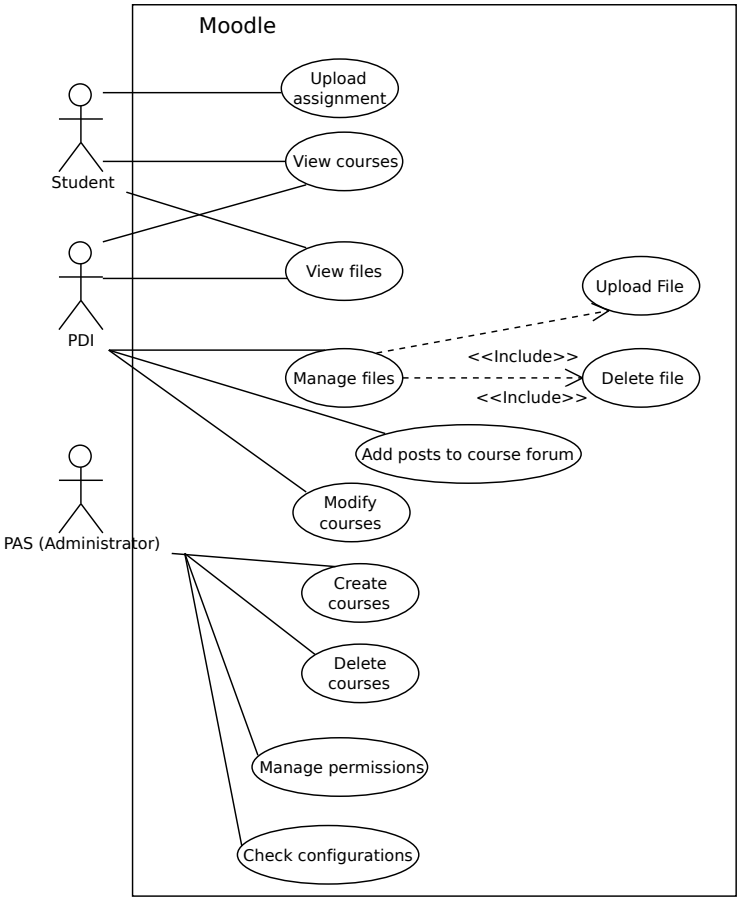


Figura 5.1: Diagrama de Casos de Uso de Moodle

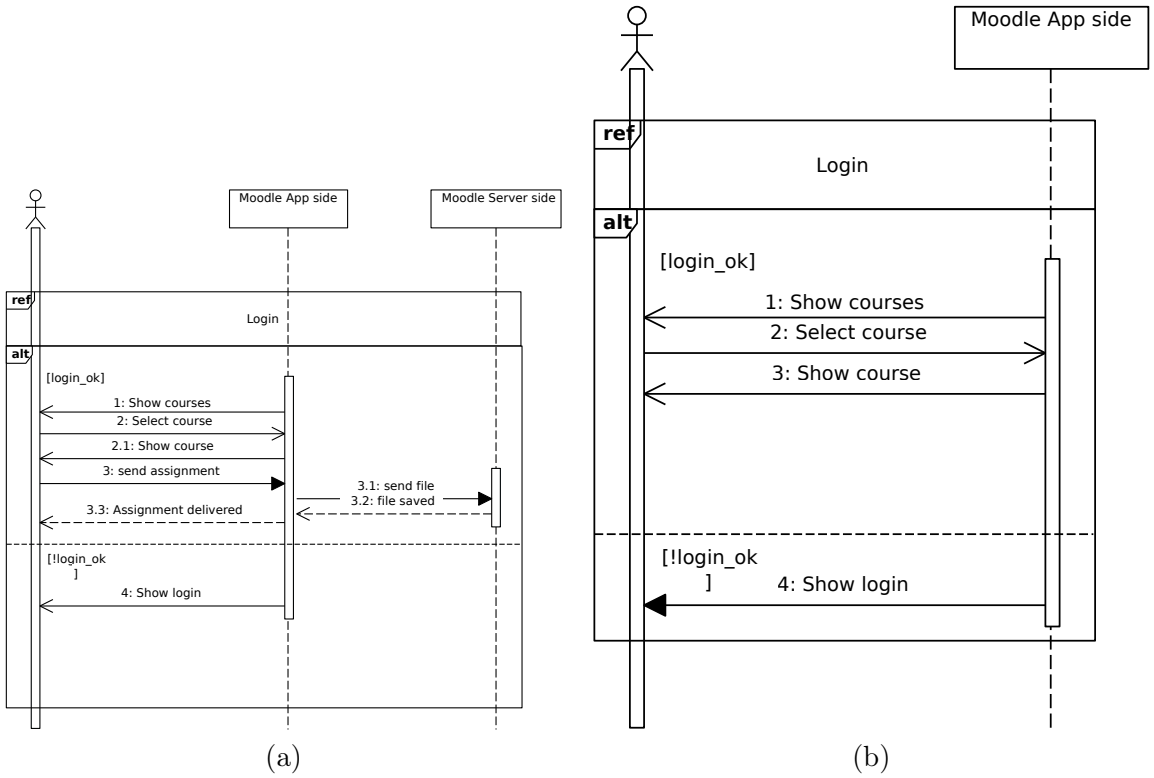


Figura 5.2: Diagramas de secuencia de (a) entrega de tarea y (b) visualización de un curso

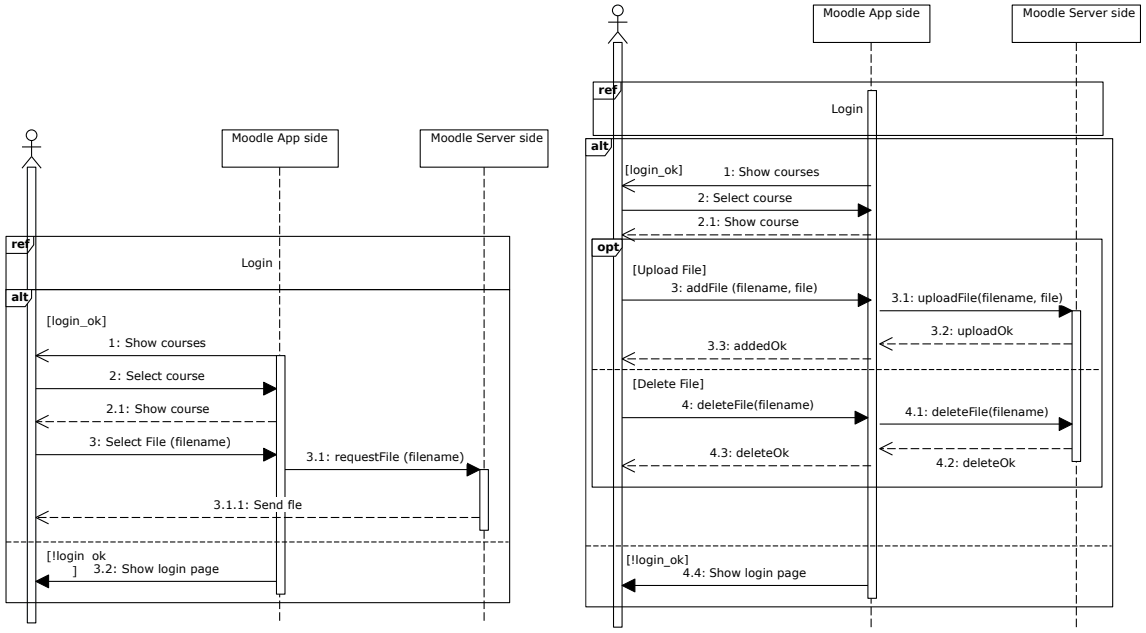


Figura 5.3: Diagramas de secuencia de (a) visualización de un fichero y (b) gestión de ficheros

tiempo de respuesta, siendo este incremento más marcado con la herramienta `HULK`. En el caso de `sqlmap`, el incremento en el tiempo de respuesta es menos significativo. Estos resultados parecen indicar que la monitorización del tiempo medio de respuesta puede ayudar para la detección de ataques DoS, no así con la detección de ataques SQLi, donde quizás se podrían emplear técnicas de análisis de estados transitorios del servidor (por ejemplo, detectar una sucesión de peticiones en muy poco tiempo donde se producen códigos de errores en la devolución de la página web, seguido de peticiones correctas).

Por último, cabe comentar que el ataque DoS se ha lanzado desde una única máquina, con recursos limitados. Un atacante que disponga de más recursos será capaz de lanzar ataques más destructivos.

5.3. Escenario 2: Moodle con el sistema desarrollado

Tras las valoraciones realizadas sobre el sistema del escenario 1 se despliega el sistema desarrollado en este trabajo. En primer lugar, cuando este sistema se lanza se realiza el entrenamiento del clasificador con la batería de logs generados para ello.

Entrenamiento del clasificador. Se generan un total de 65.000 ataques entre ataques DDoS y SQLi, y se entrena el clasificador con ellos siguiendo la división propuesta en el Capítulo 4, teniendo de esta manera 52.000 logs de entrenamiento, 6.500 logs de validación

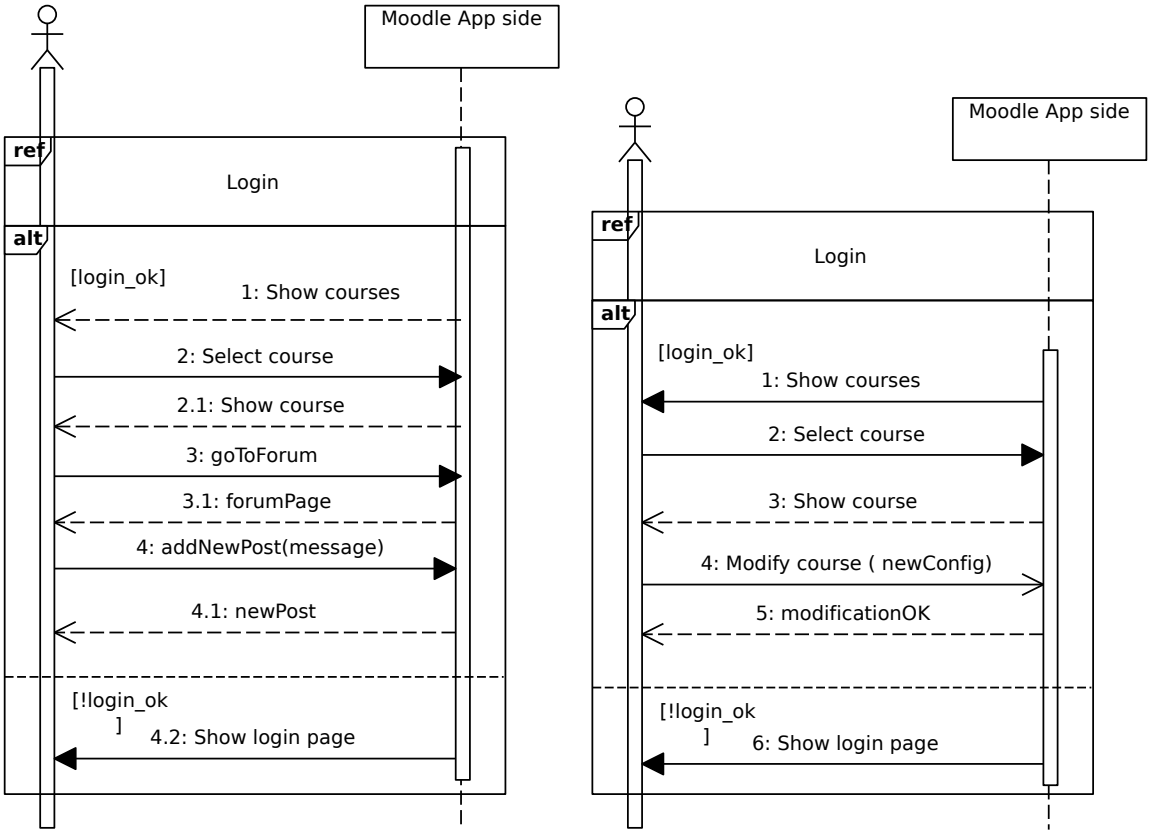


Figura 5.4: Diagramas de secuencia de (a) añadir posts al foro de un curso y (b) modificación de un curso

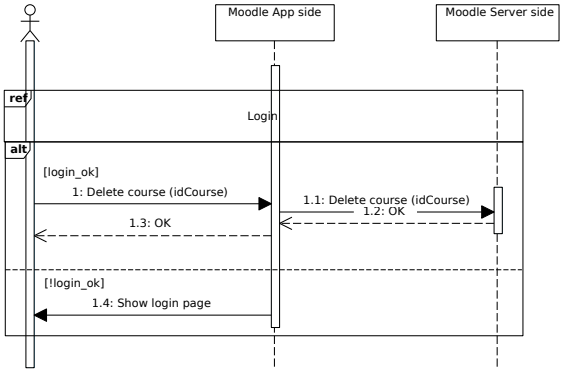


Figura 5.5: Diagrama de Secuencia de eliminar un curso.

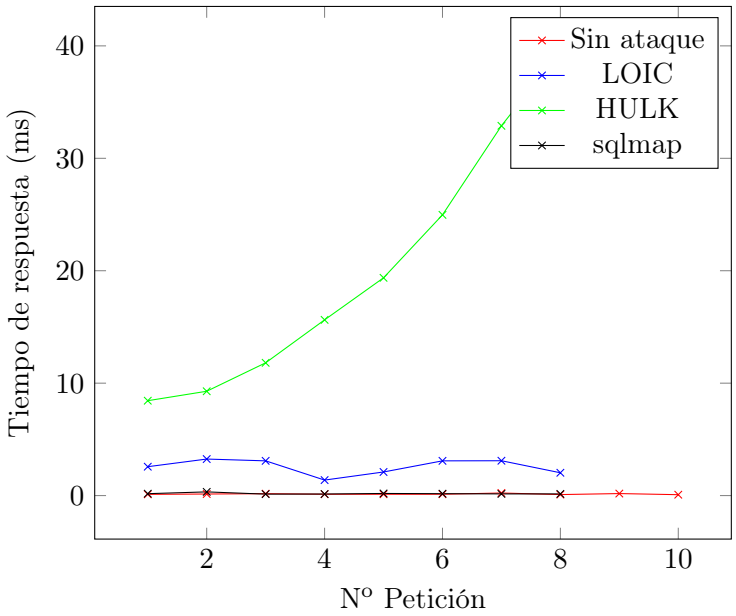


Figura 5.6: Tiempo de respuesta bajo ataque de diferentes herramientas

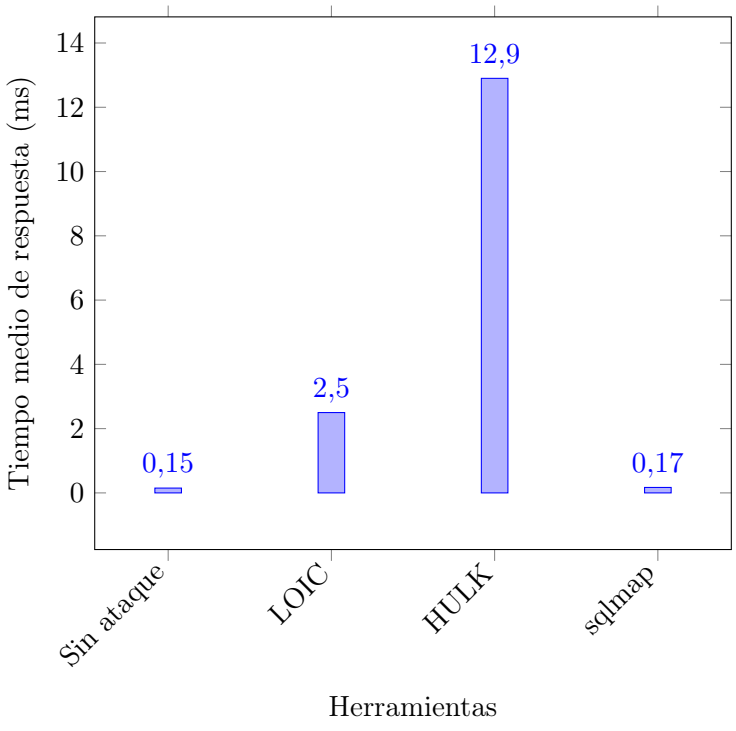


Figura 5.7: Tiempo de respuesta medio ante diferentes ataques

Variación	Falsos Positivos	Falsos Negativos
Bolsa palabras/Multinomial	4,49 %	6,28 %
Bolsa palabras/Bernoulli	4,51 %	2 %
Bigramas/Multinomial	4,49 %	6,28 %
Bigramas/Bernoulli	0 %	58,79 %

Tabla 5.1: Falsos positivos y falsos negativos para el campo *Agent*

Variación	Falsos Positivos	Falsos Negativos
Bolsa palabras/Multinomial	4,80 %	6,20 %
Bolsa palabras/Bernoulli	4,21 %	0,13 %
Bigramas/Multinomial	4,33 %	5,89 %
Bigramas/Bernoulli	0 %	66,49 %

Tabla 5.2: Falsos positivos y falsos negativos para el campo *Referrer*

y 6500 logs de test. Se establece el número de separaciones del algoritmo de validación cruzada en 10 y se consigue una media de 5362.0 con un hiper-parámetro de suavizado de Laplace de valor 1. La variación con mejor porcentaje de acierto en ambos campos a clasificar fue el uso de Bigramas y distribución Multinomial, superando en ambos casos el 95 % de acierto.

Es importante valorar también el número de falsos positivos y falsos negativos que se generan en la clasificación. En las Tablas 5.1 y 5.2 se muestran los falsos positivos y los falsos negativos de todas las variaciones tanto en el campo *Agent*, como en el campo *Referrer*. Se considera un falso positivo una petición legítima considerada como ataque. Tras los resultados obtenidos se establece la variación uso de Bigramas y distribución Multinomial para el sistema final a poner en marcha.

A continuación, tras la valoración del sistema se pone en marcha y se realizan pruebas en entorno "real". Estas pruebas se realizan desde distintas máquinas y localizaciones, ya sea mediante máquinas virtuales o máquinas físicas localizadas siempre fuera de la red universitaria, ya que la batería de ataques automáticos se realizaba desde máquinas ubicadas dentro de la propia red de la universidad. La motivación de realizar estos ataques desde fuera es aportar información diferente al clasificador para valorar su comportamiento.

Para evaluar el funcionamiento del sistema se ha utilizado una variación de la herramienta `HULK` que permite el uso de *proxies*[1] para la generación de ataques, de esta manera se permite lanzar ataques desde fuera del país y aportar mayor variedad a las pruebas.

Además, se ha desarrollado un programa en Python que lanza peticiones GET de manera continua al sistema. Este programa usa el mismo *proxy* que la herramienta `HULK`.

El sistema ante estos ataques es capaz de detectar que las peticiones cumplen varias reglas de ataque y actúa en consecuencia. En el primer caso, se cumplen todas las reglas a

[1]Programa o dispositivo que hace de intermediario en la comunicación. En el caso de estudio se utilizan varios *proxies* situados en Brasil.

excepción de la quinta, alcanzándose un porcentaje de 91.67 % en el sistema de detección (véase Tabla 4.3). Por tanto, se genera una regla de bloqueo en el IDS. Sin embargo, en el segundo caso, al tratarse de un programa muy sencillo que no intenta ocultarse ofuscando el campo *Agent* ni el campo *Referrer*, no considera la intención de ataque y sólo cumple las reglas tres, cuatro y seis, esto genera un 41.66 % de en el sistema de detección. Por tanto, el sistema de actuación no hace nada.

5.3.1. Problemas encontrados

Es difícil comprobar verdaderamente el funcionamiento del sistema, debido a que en el mundo de la ciberseguridad, tanto atacantes como defensores se hallan en constante evolución. Es cierto que el sistema desarrollado, es capaz de detectar los eventos considerados ataques y aplicar reglas en consecuencia, pero también es posible la aparición de vectores de ataque nuevos o cambios en los contemplados que hagan que el sistema no reaccione correctamente a ellos.

Durante la fase de entrenamiento del clasificador y desarrollo del sistema se encontró el mayor problema de este proyecto. Es sabido que la clave del éxito de un clasificador automático está en el catálogo de muestras de entrenamiento del que se dispone. En este proyecto se generó una batería de ataques para entrenar el clasificador de manera automática. Aunque en términos de cantidad esta batería hubiera sido suficiente, no lo era en variedad de los mismos, estrechando significativamente el alcance del proyecto. En un entorno ideal, estas muestras de entrenamiento para el clasificador se habrían recogido de diferentes fuentes, teniendo mayor variedad de ataques DDoS y SQLi, generadas por diferentes agentes y desde diferentes localizaciones.

Con este entorno ideal, el sistema de reglas desarrollado no hubiera estado acotado por parámetros estimados por el desarrollador en base a la investigación realizada, sino que sería el propio sistema el que decidiría si se trata de un ataque o no en función del contenido, de manera similar a como funcionan los clasificadores de spam en los servicios de correo electrónico. De esta manera, se permitiría también una autoadaptación del sistema en función de los ataques recibidos a lo largo del tiempo.

Capítulo 6

Trabajo relacionado

A lo largo de este proyecto se desarrolla un sistema capaz de clasificar logs de un sistema determinando cual ha de ser el comportamiento considerado normal o legítimo y aportando ejemplos de comportamientos anómalos susceptibles de ser ataques. La detección de comportamientos anómalos como acción proactiva de defensa es objeto de múltiples investigaciones.

Entre las investigaciones en materia de análisis de comportamientos es frecuente el uso conjunto de la minería de procesos con la ingeniería dirigida por modelos. La ingeniería dirigida por modelos es un paradigma de ingeniería de software que se centra más en la creación de modelos de dominio en detrimento de los conceptos informáticos. Se entiende como modelos de dominio a las representaciones abstractas de los conocimientos y actividades que rigen un dominio de aplicación particular. Estas investigaciones proponen métodos de detección de patrones de ataque mediante identificación de las desviaciones del comportamiento del sistema [3, 4].

Otros investigadores consideran el uso de *botnets* como principal vector de ataque. A consecuencia de esto, proponen un nuevo enfoque para la detección de la actividad de estas *botnets* basándose en el análisis del comportamiento del tráfico de red. Los métodos de análisis del comportamiento del tráfico no dependen del contenido de los paquetes, con lo que pueden trabajar con protocolos de comunicación de red cifrados [20].

En otras investigaciones, se han propuesto sistemas de correlación de eventos. En particular, se propone un sistema que correlaciona los programas en el lado del servidor que son referenciados por consultas de clientes en los parámetros de estas. Las características específicas de la aplicación de los parámetros permiten que el sistema realice un análisis exhaustivo y produzca un número reducido de falsos positivos [10].

La aplicación usada en este proyecto para el almacenamiento de logs de sistemas, la pila ELK, ofrece un plugin de pago llamado `X-Pack` que evalúa logs y genera alertas en base a ellos (además de paquete de funcionalidades diferentes como estadísticos, generación automática de informes, etc). El objetivo de este plugin es similar al de este proyecto, aunque con diferencias. A diferencia de `X-Pack`, el sistema desarrollado en este proyecto crea una respuesta automática al evento detectado como anómalo durante el análisis.

Capítulo 7

Conclusiones

En este capítulo se explican en primer lugar, las conclusiones del proyecto. A continuación, otros problemas encontrados a lo largo del proyecto no detallados anteriormente, tanto en las primeras fases de investigación como en las fases de desarrollo. En último lugar, se establecen unas líneas de trabajo futuro para la mejora de este proyecto.

Como conclusión, la idea de crear un sistema que gestione de manera automática la respuesta a eventos de ataque a una aplicación web en base a aprendizaje automático es viable, siempre y cuando se disponga de un volumen elevado de datos para entrenar al sistema clasificador. El sistema desarrollado en este proyecto permite ver que es posible disponer de un sistema de defensa reactiva combinado con otros mecanismos de seguridad dentro de la empresa, como sistemas IDS.

7.1. Otros problemas encontrados

Fase de investigación En la fase de investigación del proyecto donde se buscaba obtener información sobre los ataques que afectan a las empresas en sus sistemas informáticos se encontraron problemas en la propia obtención de esta información. Por norma general, las empresas privadas no hacen públicos reportes de seguridad de sus servicios informáticos por la posible repercusión económica y social que esto podría tener. Por el contrario, las administraciones públicas hacen públicos informes anuales sobre el estado de la seguridad informática. En España, por ejemplo, el Centro Criptológico Nacional (CCN-CERT) hace público todos los años un informe sobre el estado de la seguridad en las administraciones públicas.

Aunque esta información es extrapolable al entorno privado, estas limitaciones han acotado la amplitud de miras de este proyecto.

7.2. Trabajo futuro

En este apartado se habla sobre las posibles líneas futuras de trabajo en torno a este proyecto.

7.2.1. Mejora de la automatización

Debido a las limitaciones tanto de tiempo como de datos de entrenamiento para el clasificador, el proceso de automatización de detección de ataques queda limitado al sistema de reglas desarrollado. La idea de trabajo futuro sería adquirir una base de información de mayor tamaño y variedad para con ello desarrollar clasificadores para cada campo de la petición HTTP, de tal manera que no se establezcan un conjunto de países como principales emisores de ataques informáticos sino que con el entrenamiento adecuado el sistema sea capaz de conocer de dónde provienen gran parte de los ataques, por ejemplo.

Esto podría llevarse a cabo en colaboración con entidades públicas como el CCN-CERT o el INCIBE (Instituto Nacional de Ciberseguridad de España).

7.2.2. Desarrollo de módulos

En este proyecto, como medida de seguridad propuesta ante una detección de ataque sólo se han considerado dos casos (alerta o bloqueo) con Snort. Como idea futura, sería interesante el desarrollo de módulos para generación de reglas en otros sistemas IDS o en otras soluciones de seguridad.

Bibliografía

[1] Acunetix, *What is SQL Injection (SQLi) and How to Fix It.* https://www.acunetix.com/websitesecurity/sql-injection/.

[2] Akamai, *[Estado de Internet]/Seguridad*, Abr 2018.

[3] S. Bernardi, R. P. Alastuey, and R. Trillo-Lado, *Using Process Mining and Model-Driven Engineering to Enhance Security of Web Information Systems*, in 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS PW), April 2017, pp. 160–166.

[4] F. Bezerra, J. Wainer, and W. M. P. van der Aalst, *Anomaly Detection Using Process Mining*, in Enterprise, Business-Process and Information Systems Modeling, T. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, and R. Soffer, Pninaand Ukor, eds., Berlin, Heidelberg, 2009, Springer Berlin Heidelberg, pp. 149–161.

[5] A. Buscher and T. Holz, *Tracking DDoS Attacks: Insights into the Business of Disrupting the Web.*, in LEET, 2012.

[6] Cybercrimemag, *Cybercrime Damages 6 Trillion by 2021*, Sep 2018.

[7] W. G. Halfond, J. Viegas, A. Orso, et al., *A classification of SQL-injection attacks and countermeasures*, in Proceedings of the IEEE International Symposium on Secure Software Engineering, vol. 1, IEEE, 2006, pp. 13–15.

[8] A. Kaspersky, *Global Corporate IT Security Risks 2014 Survey* . https://www.kaspersky.es/.

[9] A. Kay, *Why is Cybersecurity Important?* https://investingnews.com/daily/tech-investing/cybersecurity-investing/why-is-cybersecurity-important/, Sep 2018.

[10] C. Kruegel and G. Vigna, *Anomaly Detection of Web-based Attacks*, in Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS '03, New York, NY, USA, 2003, ACM, pp. 251–261.

[11] S. Losada, *¿Qué es ELK? ElasticSearch, Logstash y Kibana.* https://openwebinars.net/blog/que-es-elk-elasticsearch-logstash-y-kibana/, Jul 2018.

[12] Python, *API Python Watchdog*. `https://pythonhosted.org/watchdog/index.html`. [Online; accedido 15-Septiembre-2018].

[13] Radware, *DDoS Attack Tools: Seven Common DDoS Attack Tools Used By Hackers*. `https://security.radware.com/ddos-knowledge-center/ddos-attack-types/common-ddos-attack-tools/`.

[14] M. Roesch, *Snort - Lightweight intrusion detection for networks*, Proceedings of LISA '99: 13th Systems Administration Conference, (1999).

[15] M. Rouse, *Intrusion detection system (IDS)* . `https://searchsecurity.techtarget.com/definition/intrusion-detection-system`. [Online; accedido 31-Septiembre-2018].

[16] Snort, *Snort-Network Intrusion Detection and Prevention System*.

[17] Statista, *Web application attack traffic by country 2018 | Statistic*.

[18] R. C. Steorts, *K-Fold Cross-Validation*. Duke University, Disponible en `http://www2.stat.duke.edu/~rcs46/lectures_2017/05-resample/05-cv.pdf`.

[19] S. T. Zargar, J. Joshi, and D. Tipper, *A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks*, IEEE Communications Surveys & Tutorials, 15 (2013), p. 2046–2069.

[20] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, *Botnet detection based on traffic behavior analysis and flow intervals*, Computers and Security, 39 (2013), pp. 2 – 16. 27th IFIP International Information Security Conference.

Apéndice A

Anexo

En este Anexo se muestra la división en tareas del proyecto así como el tiempo dedicado en cada una de ellas. La cantidad de horas es una estimación basada en anotaciones durante el desarrollo del proyecto. En la Tabla A.1 se ve la distribución de horas dividida en cinco grandes grupos. Las horas de investigación corresponden a la lectura de artículos, webs de referencia, manuales de programación y otros materiales consultados durante el proyecto. Las horas de desarrollo corresponden tanto a desarrollo del sistema como a su puesta en marcha. Las horas de documentación corresponden en su gran mayoría a las horas dedicadas a la redacción de esta memoria. Las horas de test corresponden a las pruebas realizadas con el caso de estudio. Por último, en las horas de reuniones y otros se recogen las horas dedicadas a reuniones ya sean físicas o telemáticas con el director, así como horas no adjudicadas a los otros apartados ya sea por no ajustarse claramente a ellos o por no estar anotadas debidamente. De la misma manera, en la Figura A.1 se puede observar de una manera más gráfica el porcentaje de dedicación a cada apartado de trabajo.

Aunque sería conveniente incluir un apartado donde se establezcan las horas dedicadas al diseño del sistema, se decide incluir en el apartado desarrollo entendiendo como desarrollo todas las horas dedicadas a la creación del sistema generador de reglas, tanto en diseño como en desarrollo.

Tipo Tarea	**Tiempo dedicado**
Investigación	69
Reuniones y otros	23
Documentación	127
Desarrollo	93
Tests	10
TOTAL	322

Tabla A.1: Tiempo dedicado al proyecto

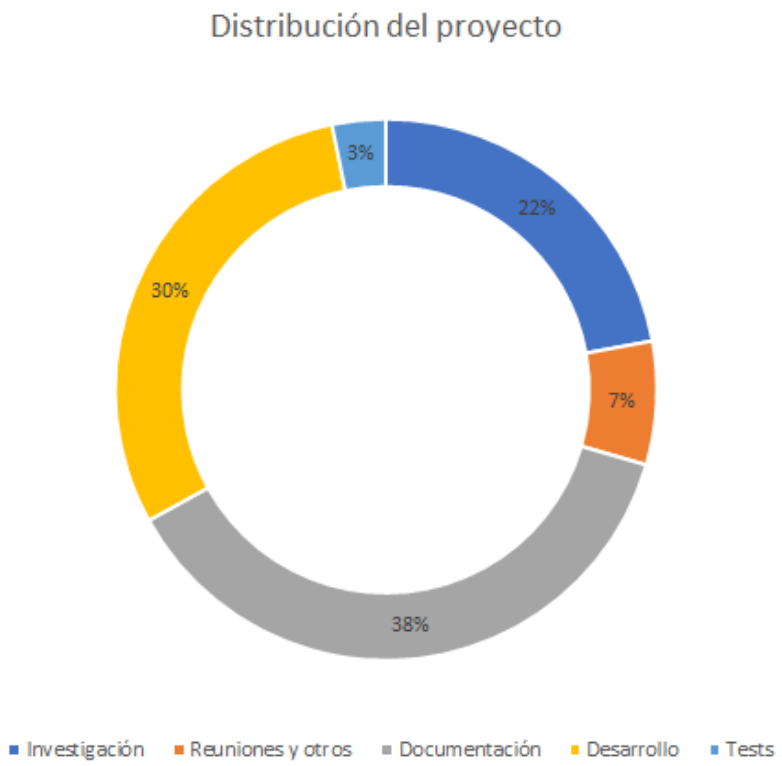


Figura A.1: Distribución del tiempo dedicado al proyecto