TESIS DOCTORAL

Study of Stochastic and Machine Learning Techniques for Anomaly-based Web Attack Detection

Autora:

Carmen Torrano Giménez

Director:

Dr. Gonzalo Álvarez Marañón

Tutor:

Dr. Javier Carbó Rubiera

ESCUELA POLITÉCNICA SUPERIOR – INGENIERÍA INFORMÁTICA

INSTITUTO DE TECNOLOGÍAS FÍSICAS Y DE LA INFORMACIÓN

Leganés, septiembre de 2015

Universidad Carlos III de Madrid
www.uc3m.es

CSIC
CONSEJO SUPERIOR DE INVESTIGACIONES CIENTÍFICAS

TESIS DOCTORAL

Study of Stochastic and Machine Learning Techniques for Anomaly-based Web Attack Detection

Autora: *Carmen Torrano Giménez*

Director: Dr. Gonzalo Álvarez Marañón

Firma del Tribunal Calificador:

Firma

Presidente:

Vocal:

Secretario:

Calificación:

Leganés, de de

Este trabajo ha sido realizado en el marco de las becas predoctorales de la Junta de Amplicación de Estudios (JAE) de la Agencia Estatal Consejo Superior de Investigaciones Científicas (CSIC).

A mis queridos padres y hermana

To my beloved parents and sister

“Es preciso entender la tesis como una
ocasión única para hacer algunos ejercicios
que nos servirán mientras vivamos.”
— Umberto Eco

“It is necessary to understand the PhD. thesis
as a unique occasion to do some exercises that
will be useful while we live.”
— Umberto Eco

Agradecimientos

Este periodo doctoral ha sido un camino de gran aprendizaje. Un camino con sus picos, valles, aventuras, obstáculos... todos ellos merecen ser vividos. Y es que una tesis no es sólo un gran proyecto, sino una parte importante de nuestra vida. En esta etapa he tenido el privilegio de compartir camino con grandes personas, a las que me gustaría expresar todo mi agradecimiento.

Comenzaré por el director de esta tesis, Gonzalo. Gracias por haberme dado la oportunidad de hacer este doctorado y por todo lo que he podido aprender al tenerte como director. Javier, en su papel como tutor, ha sido un puente de unión excelente entre el CSIC y la universidad. Gracias por tu ayuda en las las labores burocráticas y por haber sido un excelente anfitrión. También me gustaría agradecer al programa JAE por la financiación recibida para realizar esta tesis doctoral.

Mis compañeros del CSIC han tenido un papel fundamental para mí. No puedo dejar de agradeceros por tantos momentos vividos y por haber crecido juntos. Me siento afortunada de haber formado parte de un equipo que no sólo destaca por su calidad profesional, sino por su calidad humana. Gracias a Luis, cabeza del Instituto, por su ayuda y su gestión en el centro. A Nacho, por arrancarnos siempre una carcajada. A Juanca, por su amabilidad y cuidado. A Alfonso, por su autenticidad e interés. A Jesús, por tenerlo siempre todo a punto y por su cordura. A David, por ilustrarnos con su vasto conocimiento y las charlas de horas. A Arancha, por su eterna sonrisa. A Alejandro, compañero de batallas; eso une. A Mari Jose, por su compañerismo y naturalidad. A Verónica, por su amistad sin igual y por inspirarme cada día. A Fausto, por enseñarnos con su experiencia y sabiduría. A Amalia por su ayuda y dulzura. A Amparo, por tener siempre palabras de aliento. A Mari Carmen, por su interés y afecto. A Ignacio, por su experiencia y cariño. A Jose, por mostrarnos su lado más humano. A Oscar por la huella que deja su valía y por contar siempre conmigo. A Alberto Carrasco, por los interesantes debates. A Alberto Guerra, por ser un gran compañero. Agradezco la espontaneidad de Natalia, la alegría de Sara, la simpatía de Marta, la compañía de Ricardo, Noemí, Víctor Gayoso, Raúl, Luis, Víctor Fernández, Gerardo, Jaime. Y también, a Agus,

maestro en lo laboral y en lo personal. Gracias por tu inestimable ayuda y por tantas horas compartidas juntos.

Gracias a todo el resto de personal que de alguna manera ha colaborado en que mi trabajo fuera posible: Mari Jose, Luci, Gema, Marisa y un largo etcétera. Gracias también a los vigilantes por abrirme la puerta a horas intempestivas. No puedo olvidarme de las personas de los otros departamentos que me han ayudado a ponerle sal a la vida y a hacer más cálidos los pasillos.

El CSIC ha sido como un segundo hogar para mi, lleno de imborrables recuerdos que me llevo en el corazón. Y es que no sólo he aprendido de ciencia en los cafés, sino también de las relaciones humanas y sobre todo, de la vida.

También me gustaría agradecer a mis compañeros de la universidad: José Antonio, Raúl, Juan por hacerme reír hasta que me doliera el estómago. También a Paz, Germán, Tomás, Bea, Jorge Blasco, Jorge Muñoz, Chema, Jose María Sierra. Aprender a vuestro lado siempre es más fácil. Gracias a Juan por haber aparecido en el momento preciso y por no escatimar en ánimos ni ayuda. A Sergio y Agustín por haber sido unos excelentes compañeros de trabajo, además de grandes personas. Agus, te llevo en el corazón. Gracias a Hai, por haberme hecho sentir tan cómoda trabajando a su lado. Y a Slobodan y Katrin por su colaboraración desde la distancia.

Durante mi periodo doctoral he tenido la oportunidad de realizar varias estancias en el extranjero, en las que he tenido el placer de trabajar y convivir con gente maravillosa. En mi estancia en la Universidad de Nottingham tuve el gusto de trabajar con Uwe Aickelin, Julie Greensmith, Jan, Gianni y Rafael Lahoz, entre otros.

Gracias por cada minuto de mi aventura en la Universidad de Munich. Involvidables momentos que compartí con Gehard, Lothard, Andreas, Dirk, Pia, Ralph, Tobi, Ali, Marc Fouquet, Nils, Heiko, Holgar, Andreas Müller, Corinna, Carlas, Mayur, Vineeth. Gracias a Prof. Carle por acogerme siempre tan bien y por su ayuda. Mis infinitas gracias a Marc (U), por lo especial de haberte conocido.

En la visita a Eurecom tuve el placer de trabajar con personas de la altura de Engin Kirda, Davide Balzarotti, Leila Bilge, Marco Balduzzi y Andrea Lanci. Gracias por las horas compartidas y el enriquecimiento multicultural de Miriam, Sara, Lluis–"ciccio", Mateo, Tony. Extraordinaria experiencia a vuestro lado allí donde el paisaje es inmejorable.

Me gustaría además recordar a mis hackers favoritos: Pablo, Red, Jhonattan, Juanito, Juanan, Chema, Román, Alfonso, Dani, Raúl y tantos otros, por compartir sus conocimientos y acompañarme a adentrarme en las profundidades del mundo de la seguridad informática.

También me gustaría dar las gracias a algunas personas que, fuera del ámbito laboral, tienen un papel fundamental en mi vida. Gracias a Paula y Anaida por haber estado a mi lado desde hace tantos años. A Emilio y a Natalia por enseñarme tanto con su sabiduría y por estar cerca. A Miguel y Eva, por ayudarme a entender mejor la vida. Gracias a Blanca, por su incansable escucha. A Edu por su humanismo y a Iser por su capacidad de improvisación. Agradezco a Santi por su cariño y por ser un chico genial. Gracias a Álvaro por su paciencia y apoyo, y por tanto compartido. A Adri, por enseñarme otra visión del mundo y por los extraordinarios momentos vividos. Y a tantas otras personas... Gracias a todos por formar parte de mi vida.

Y por supuesto, a mi familia, uno de los pilares básicos de mi vida. Gracias a mis padres, los mejores que podía tener, por todo lo que me dáis y por vuestro ejemplo. Gracias a mi madre, por su amor incondicional y por dar sin conocer límite. Y a mi padre, por ilustrarme con su ejemplo y por su ayuda. Y qué decir de mi hermana, a la que quiero con locura incluso desde antes de que naciera. También al resto de mi familia por rebosar salero y sentido común. Gracias por apoyarme en todo momento, es un privilegio teneros cerca.

En definitiva, gracias a todos los que habéis colaborado, de una forma u otra, en este proyecto. Porque en él, hay un pedacito de cada uno de vosotros. Gracias por formar parte de él, y sobre todo, por formar parte de mí.

Con todo mi agradecimiento,

Carmen

Acknowledgements

This doctoral period has been a path of great learning for me. A path with peaks, valleys, adventures, obstacles... that are all worth to be lived. Because a Ph.D. thesis is not only a large project, but also an important part of our lives. During this time I have had the privilege of sharing this path with extraordinary people, to whom I would like to express all my gratitude.

I will start with my supervisor, Gonzalo. Thanks for giving me the opportunity of doing this Ph.D. and for all I have learned from having you as a supervisor. Javier, my tutor, has been an excellent bridge between CSIC and University. I would also like to thank the JAE program for the funding received to carry out this Ph.D. thesis.

My colleagues at CSIC have been very important for me. I would like to express my gratitude for sharing so many wonderful moments. I feel lucky for being part of a team that not only stands out for its professional quality, but also for its human value. Thanks to Luis, head of the Institute, for his help and management labour. To Nacho, for making us laugh so much. To Juanca, for his kindness and care. To Alfonso, for his authenticity and attention. To Jesús, for having everything ready and for his sanity. To David, for sharing with us some of his huge knowledge and for our long conversations. To Arancha, for her eternal smile. To Alejandro, battle-mate, that brought us close. To Mari Jose, for being a great mate and being so natural. To Verónica, for her unique friendship and for inspiring me every day. To Fausto, for teaching us with his experience and wisdom. To Amalia, for her help and sweetness. To Amparo, for always having words of encouragement. To Mari Carmen, for her interest and affection. To Ignacio, for his experience and care. To Jose, for showing us his most human side. To Oscar for his immense value and for counting on me. To Alberto Carrasco for the interesting debates. To Alberto Guerra, for being an excellent mate. I thank Natalia's spontaneity, Sara's happiness, Marta's friendliness, and the company of Ricardo, Noemí, Víctor Gayoso, Raúl, Luis, Víctor Fernández, Gerardo, Jaime. Also to Agus, a teacher both in professional and personal areas. Thank you for your inestimable help and for so many hours shared together.

Thanks to the remaining staff members who somehow have collaborated in making my work possible, like Mari Jose, Luci, Gema, Marisa and a long etcetera. Thanks also to the guards for always opening the door. I cannot forget people from other departments who helped me to make life nicer and corridors warmer.

CSIC has been like a second home for me, full of unforgettable memories that are within my heart. I have not only learned about science in coffee times, but also about human relationships and mostly, about life.

I would also like to thank my university colleagues: José Antonio, Raúl, Juan for making me laugh until my stomach hurt. Also to Paz, Germán, Tomás, Bea, Jorge Blasco, Jorge Muñoz, Chema, Jose María Sierra. Learning with you is easier. Thanks to Juan for appearing in the right moment and not skimping on motivation or help. To Sergio and Agustín for being extraordinary workmates, besides great people. Agus, you are in my heart. Thanks to Hai for making me feel so comfortable working with him. And to Slobodan and Katrin for their collaboration through distance.

During my doctoral period I have had the opportunity to do several research stays abroad, where I had the pleasure of working and living with wonderful people. In my stay in Nottingham I had the chance to work with Uwe Aickelin, Julie Greensmith, Jan, Gianni and Rafael Lahoz, among others.

Thanks for every minute of my adventure in Technische Universität München. Unforgettable moments that I shared with Gehard, Lothard, Andreas, Dirk, Pia, Ralph, Tobi, Ali, Marc Fouquet, Nils, Heiko, Holgar, Andreas Müller, Corinna, Carlas, Mayur, Vineeth. Thanks to Prof. Carle for hosting me always so nicely and for his help. My infinite thanks to Marc (U), for the special fact of getting to know you.

In my visit to Eurecom I was honored to work with awesome experts like Engin Kirda, Davide Balzarotti, Leila Bilge, Marco Balduzzi and Andrea Lanci. Thanks for the hours shared and the multicultural enrichment of Miriam, Sara, Lluis–"ciccio", Mateo, Tony. It was an extraordinary experience with a superb landscape.

I would also like to remember my favorite hackers: Pablo, Red, Jhonattan, Juanito, Juanan, Chema, Román, Alfonso, Dani, Raúl and many more, for

sharing their knowledge and for walking with me to the depths of the computer security world.

I want to express my gratitude also to some people that, besides the professional area, have an essential role in my life. Thanks to Paula and Anaida for being with me since so many years ago. Thanks to Emilio and Natalia for teaching me so much with their wisdom and for staying close to me. To Miguel and Eva, for helping me understand life. Thanks to Blanca, for her constant listening. To Edu for his humanity and to Iser for his improvisation ability.
I thank Santi for his care and for being a great boy. Thanks to Alvaro for his patience and support, and for so much shared. To Adri, for showing me other vision of the world and for the extraordinary moments shared. And to so many others...Thanks to all of you for being part of my life.

And of course, to my family, one of the basic supports in my life. Thanks to my parents, the best I could have, for all you give me and for your good example. Thanks to my mother for her unconditional love and for giving endlessly. Thanks to my father for being an example and for his help. And what can I say about my sister, whom I have loved since even before she was born. Also to the rest of my family, for their charm and common sense. Thanks for always supporting me, it is a privilege to have you close.

In summary, thanks to all of you who have participated, in a way or another, in this project. Because in it, there is a part of all of you. Thanks for being part of it, and mainly, for being part of me.

With all my gratitude,

Carmen

Abstract

Web applications are exposed to different threats and it is necessary to protect them. Intrusion Detection Systems (IDSs) are a solution external to the web application that do not require the modification of the application's code in order to protect it. These systems are located in the network, monitoring events and searching for signs of anomalies or threats that can compromise the security of the information systems.

IDSs have been applied to traffic analysis of different protocols, such as TCP, FTP or HTTP. Web Application Firewalls (WAFs) are special cases of IDSs that are specialized in analyzing HTTP traffic with the aim of safeguarding web applications.

The increase in the amount of data traveling through the Internet and the growing sophistication of the attacks, make necessary protection mechanisms that are both effective and efficient.

This thesis proposes three anomaly-based WAFs with the characteristics of being high-speed, reaching high detection results and having a simple design. The anomaly-based approach defines the normal behavior of web application. Actions that deviate from it are considered anomalous. The proposed WAFs work at the application layer analyzing the payload of HTTP requests. These systems are designed with different detection algorithms in order to compare their results and performance.

Two of the systems proposed are based on stochastic techniques: one of the them is based on statistical techniques and the other one in Markov chains. The third WAF presented in this thesis is ML-based. Machine Learning (ML) deals with constructing computer programs that automatically learn with experience and can be very helpful in dealing with big amounts of data. Concretely, this third WAF is based on decision trees given their proved effectiveness in intrusion detection. In particular, four algorithms are employed: C4.5, CART, Random Tree and Random Forest.

Typically, two phases are distinguished in IDSs: preprocessing and processing. In the case of stochastic systems, preprocessing includes feature extraction. The processing phase consists in training the system in order to learn the normal behavior and later testing how well it classifies the incoming requests as either normal or anomalous. The detection models of the systems are implemented either with statistical techniques or with Markov chains, depending on the system considered.

For the system based on decision trees, the preprocessing phase comprises feature extraction as well as feature selection. These two phases are optimized. On the one hand, new feature extraction methods are proposed. They combine features extracted by means of expert knowledge and *n*-grams, and have the capacity of improving the detection results of both techniques separately. For feature selection, the Generic Feature Selection *GeFS* measure has been used, which has been proven to be very effective in reducing the number of redundant and irrelevant features.

Additionally, for the three systems, a study for establishing the minimum number of requests required to train them in order to achieve a certain detection result has been performed. Reducing the number of training requests can greatly help in the optimization of the resource consumption of WAFs as well as on the data gathering process.

Besides designing and implementing the systems, evaluating them is an essential step. For that purpose, a dataset is necessary. Unfortunately, finding labeled and adequate datasets is not an easy task. In fact, the study of the most popular datasets in the intrusion detection field reveals that most of them do not satisfy the requirements for evaluating WAFs. In order to tackle this situation, this thesis proposes the new *CSIC* dataset, that satisfies the necessary conditions to satisfactorily evaluate WAFs.

The proposed systems have been experimentally evaluated. For that, the proposed *CSIC* dataset and the existing *ECML/PKDD* dataset have been used. The three presented systems have been compared in terms of their detection results, processing time and number of training requests used. For this comparison, the *CSIC* dataset has been used.

In summary, this thesis proposes three WAFs based on stochastic and ML techniques. Additionally, the systems are compared, what allows to determine which system is the most appropriate for each scenario.

Resumen

Las aplicaciones web están expuestas a diferentes amenazas y es necesario protegerlas. Los sistemas de detección de intrusiones (IDSs del inglés *Intrusion Detection Systems*) son una solución externa a la aplicación web que no requiere la modificación del código de la aplicación para protegerla. Estos sistemas se sitúan en la red, monitorizando los eventos y buscando señales de anomalías o amenazas que puedan comprometer la seguridad de los sistemas de información.

Los IDSs se han aplicado al análisis de tráfico de varios protocolos, tales como TCP, FTP o HTTP. Los Cortafuegos de Aplicaciones Web (WAFs del inglés *Web Application Firewall*) son un caso especial de los IDSs que están especializados en analizar tráfico HTTP con el objetivo de salvaguardar las aplicaciones web.

El incremento en la cantidad de datos circulando por Internet y la creciente sofisticación de los ataques hace necesario contar con mecanismos de protección que sean efectivos y eficientes.

Esta tesis propone tres WAFs basados en anomalías que tienen las características de ser de alta velocidad, alcanzar altos resultados de detección y contar con un diseño sencillo. El enfoque basado en anomalías define el comportamiento normal de la aplicación, de modo que las acciones que se desvían del mismo se consideran anómalas. Los WAFs diseñados trabajan en la capa de aplicación y analizan el contenido de las peticiones HTTP. Estos sistemas están diseñados con diferentes algoritmos de detección para comparar sus resultados y rendimiento.

Dos de los sistemas propuestos están basados en técnicas estocásticas: una de ellas está basada en técnicas estadísticas y la otra en cadenas de Markov. El tercer WAF presentado en esta tesis está basado en aprendizaje automático. El aprendizaje automático (ML del inglés *Machine Learning*) se ocupa de cómo construir programas informáticos que aprenden automáticamente con la experiencia y puede ser muy útil cuando se trabaja con grandes cantidades de datos. En concreto, este tercer WAF está basado en árboles de decisión, dada su probada efectividad en la detección de intrusiones. En particular, se han empleado cuatro algoritmos: C4.5, CART, Random Tree y Random Forest.

Típicamente se distinguen dos fases en los IDSs: preprocesamiento y procesamiento. En el caso de los sistemas estocásticos, en la fase de preprocesamiento se realiza la extracción de características. El procesamiento consiste en el entrenamiento del sistema para que aprenda el comportamiento normal y más tarde se comprueba cuán bien el sistema es capaz de clasificar las peticiones entrantes como normales o anómalas. Los modelos de detección de los sistemas están implementados bien con técnicas estadísticas o bien con cadenas de Markov, dependiendo del sistema considerado.

Para el sistema basado en árboles de decisión la fase de preprocesamiento comprende tanto la extracción de características como la selección de características. Estas dos fases se han optimizado. Por un lado, se proponen nuevos métodos de extracción de características. Éstos combinan características extraídas por medio de conocimiento experto y n-gramas y tienen la capacidad de mejorar los resultados de detección de ambas técnicas por separado. Para la selección de características, se ha utilizado la medida *GeFS* (del inglés *Generic Feature Selection*), la cual ha probado ser muy efectiva en la reducción del número de características redundantes e irrelevantes.

Además, para los tres sistemas, se ha realizado un estudio para establecer el mínimo número de peticiones necesarias para entrenarlos y obtener un cierto resultado. Reducir el número de peticiones de entrenamiento puede ayudar en gran medida a la optimización del consumo de recursos de los WAFs así como en el proceso de adquisición de datos.

Además de diseñar e implementar los sistemas, la tarea de evaluarlos es esencial. Para este propósito es necesario un conjunto de datos. Desafortunadamente, encontrar conjuntos de datos etiquetados y adecuados no

es una tarea fácil. De hecho, el estudio de los conjuntos de datos más utilizados en el campo de la detección de intrusiones revela que la mayoría de ellos no cumple los requisitos para evaluar WAFs. Para enfrentar esta situación, esta tesis presenta un nuevo conjunto de datos llamado *CSIC*, que satisface las condiciones necesarias para evaluar WAFs satisfactoriamente.

Los sistemas propuestos se han evaluado experimentalmente. Para ello, se ha utilizado el conjunto de datos propuesto (*CSIC*) y otro existente llamado *ECML/PKDD*. Los tres sistemas presentados se han comparado con respecto a sus resultados de detección, tiempo de procesamiento y número de peticiones de entrenamiento utilizadas. Para esta comparación se ha utilizado el conjunto de datos *CSIC*.

En resumen, esta tesis propone tres WAFs basados en técnicas estocásticas y de ML. Además, se han comparado estos sistemas entre sí, lo que permite determinar qué sistema es el más adecuado para cada escenario.

Contents

List of figures

List of tables

Chapter 1

Introduction

"Security, like correctness, is not an additional feature."
— Andrew S. Tanenbaum

This chapter shows the importance of web applications nowadays and points out the security risks that they are exposed to. Web Application Firewalls (WAFs) are able to protect web applications without the need of modifying their code. WAFs analyze web traffic and look for anomalies that could threaten the web server. On the one hand, modern WAFs deal with an increasing amount of data and on the other hand, attacks are becoming more complex. This leads to the necessity of both effective and efficient WAFs. This thesis presents three anomaly-based WAFs that satisfy these two conditions. The proposed systems apply stochastic and machine learning algorithms, in particular statistical techniques, Markov chains and decision trees. Additionally, adequate datasets are essential for evaluating WAFs. However, existing datasets present various disadvantages. In order to fill this gap, this thesis presents a new dataset for adequately evaluating WAFs. In this chapter, the motivation, objectives and structure of this dissertation are presented.

1.1 Background

The Internet has changed the way people interact with technology. It is considered one of the inventions that has changed the world the most in the last 40 years [Burkeman, 2009], [Wharton, 2009]. Web technologies have modified our daily life and transformed our customs in a large variety of sectors, such as economy, education, telecommunications, entertainment, healthcare and so on.

During their evolution, web applications have increased in complexity. Applications belonging to Web 1.0 were basically document repositories, where pages were static and the user was limited to view their content [Stuttard and Pinto, 2007]. The evolution to the next and current generation (2.0), introduced interactive and collaborative web applications [Shelly and Frydenberg, 2011]. Examples of Web 2.0 applications are social networks, blogs, wikis and search engines. Differently to first web applications, web servers of the second generation usually connect to supporting back-end systems such as databases, mainframes and financial or logistical systems [Stuttard and Pinto, 2007]. Evolution of web applications continues: researchers are currently working on Web 3.0. This next generation is characterized by the semantic web. This technology consists in making machines understand and interpret the content and semantics of web applications [Rollett et al., 2007]. As can be seen, the evolution of web applications make them grow in complexity, introducing new functionalities and mixing a vast amount of technologies.

Web applications are an open window to the world that provides to millions of users the access to huge amounts of information and services in a wide range of different areas:

Economy. Web applications have changed the economy sector in different aspects. For example, nowadays users can interact online with their banks, performing actions such as checking the balance of their bank accounts, making money transfers or contracting deposits.

Another economy sector revolutionized by web applications is commerce. On the one hand, thousands of online shopping applications have been created regarding business to consumer. On the other hand, many interesting web applications help companies in their commerce transactions in the field of business to business.

Within organizations, web applications help in managing company resources and give access to human resource services [Stuttard and Pinto, 2007]. Furthermore, they play a critical role in advertisement.

Education and Science. E-learning consists in leaning through internet. This paradigm has transformed education and science. Nowadays many web applications offer access to online courses. Furthermore, wikis have changed the generation of contents, by allowing the participation of users in their creation in a collaborative manner. Regarding science, specialized websites and online journals offer scientific contents.

Entertainment. Web applications have helped in the evolution of the entertainment field with the introduction of blogs, multimedia streaming and news feeds. In the gaming world, distant users can play together connected through the network. Additionally, the industry of porn moves big amounts of traffic and money every day. It is estimated that the yearly revenue of this business ranges between 1 and 97 billion USD [Wondracek et al., 2010].

Communications. Videoconferences are an example of how web applications have impacted communications. Other example is webmail, that brings people into contact. It makes possible to handle emails in a remote server as if the traditional mail clients were installed. Furthermore, social interactions are conceived differently since social networks appeared. These kind of webs are used in diverse environments, ranging from labor to personal relationships.

Other areas. Web search engines are an example of other uses of web applications. These engines are massively used nowadays to retrieve online information. Other example are applications dedicated to file downloading and sharing, that have impacted sectors such as cinema or music. The Internet of Things is the network of physical objects embedded with hardware and software that provides connectivity with the manufacturer, operator or other connected devices [Atzori et al., 2010]. Web applications are widely used as configuration interfaces for network-capable embedded devices [Bojinov et al., 2009].

According to the International Telecommunication Union (ITU) [ITU, 2015] and Internet Live Stats [Real Time Statistics Project, 2014], the number of Internet users constitutes more than 40% of the world population. Figure 1.1 shows the growth in the number of internet world users over the last years.

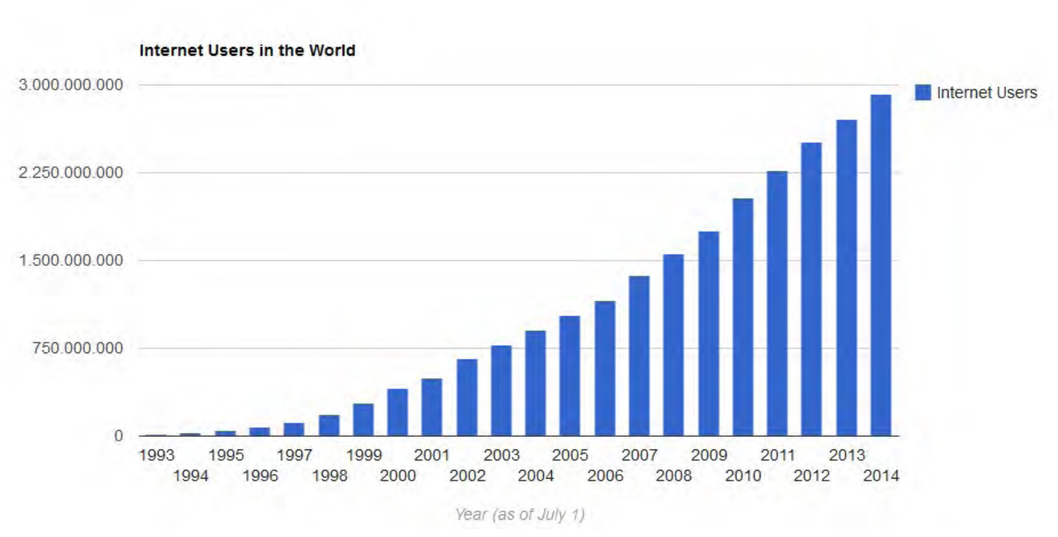


Figure 1.1 – Number of Internet users in the world over the last years. Internet Live Stats 2014.

This incessant growth in the number of internet users implies an increase in the amount of data available in the Internet. The research fields big data [Cárdenas et al., 2013], [Camacho et al., 2014], [Curry et al., 2013] and data science [Jifa and Lingling, 2014] study how to deal, analyze, store and extract knowledge from the high amount of data transferred over the Internet today. According to the Cloud Security Alliance [Cárdenas et al., 2013], human beings create 2.5 10^{18} bytes of data per day. Besides operating with big volumes of data, web applications move large amounts of money every day. Moreover, new computing and communication paradigms such as cloud computing, wireless networks and smartphone platforms allow web applications be accessed from many different places and devices.

These characteristics of web applications make them attractive targets for attackers. The fact of having millions of users, being accessed from different platforms, moving large amounts of money and handling private information make them tempting for cybercriminals. Attacker's motivations are classified by Barber into curiosity, vandalism, hacktivism, industrial espionage, extortion and fraud, information warfare and ethical hacking [Barber, 2001].

Web applications present vulnerabilities very often. As long as web applications turn more complex and include diverse technologies, they become more fragile [Stuttard and Pinto, 2007]. Moreover, the immature awareness of developers about the importance of security might produce source code with security vulnerabilities that can be exploited by an attacker. According to a recent Symantec Report [Wood, 2014], the number of total breaches experimented an increase of 62% from 2012 to 2013 and the number of new vulnerabilities for web-based attacks increased by 28%. The number of web attacks is increasingly growing. Furthermore, attacks are rapidly evolving and their sophistication increasing. For example, botnets are recent threats that aim to manage and control illegitimately a network of computers. They differentiate from traditional malware in their capability to combine several ideas implemented separately by traditional malware [Brezo et al., 2013]. Advanced Persistent Threats (APTs) are a breed of threats that use multiple attack techniques to avoid being detected and maintain the control over a target system for long periods of time [Tankard, 2011]. They are complex attacks that usually consist of several stages. APTs are performed by highly-skilled and motivated attackers targeting sensitive information from specific organizations that might even employ social engineering methods [Cárdenas et al., 2013].

The consequences of web attacks can be drastic. They can imply huge losses of money or reputation. Thus, it is necessary to saveguard web applications and their users. To that aim, reliable and effective security protection mechanisms are necessary. Furthermore, given the amount of data these systems have to deal with, they should be efficient in the request processing and consume low resources.

1.2 Concepts about web security

Despite the effort on security awareness of several institutions such as MITRE [MITRE, 2014a], SANS Institute [SANS Institute, 2015] or Open Web Application Security Project (OWASP) [OWASP, 2013], web application developers are not completely aware of web vulnerabilities, or they are not implementing correctly the corresponding countermeasures [Scholte et al., 2011]. Given the increasing need of protecting web applications, the security research community has developed several tools and techniques to improve the security of web applications.

Some of these techniques try to guarantee a secure coding of web applications. This is the case of static [Jovanovic et al., 2006], [Wassermann and Su, 2008] and dynamic [Schwartz et al., 2010], [Newsome and Song, 2005], [Nguyen-Tuong et al., 2005] code analysis. These solutions require a thorough examination of the application source code, either with (dynamic) or without (static) the compilation of the application. Considering that some web applications contain thousands of lines, this can be a tedious task.

Additionally, other solutions have been developed, like Intrusion Detection Systems (IDSs). These systems monitor the events of a computer or network in order to detect malicious actions and behaviors that can compromise their security. One of the advantages of IDSs is that they can safeguard web applications without modifying the code even if it is wrongly programmed.

This technology has been extensively used for protecting computer systems. They have been applied to analyze traffic corresponding to different network protocols, such as File Transfer Protocol (FTP), Transmission Control Protocol (TCP) or Hypertext Transfer Protocol (HTTP).

In these systems two stages are usually distinguished: preprocessing and processing. Preprocessing includes those tasks carried out before the formal processing of data, such as dataset creation, feature extraction and feature selection [Kotsiantis et al., 2006]. Processing makes reference to the detection itself.

Traditionally, IDSs have been classified as either signature detection systems or anomaly detection systems [García-Teodoro et al., 2009]. On the one hand, signature detection looks for signatures (patterns) of known attacks. For that, it uses pattern matching techniques against a database with attack signatures. This approach is unable to detect new attacks and databases must be frequently updated. Additionally, signature matching usually requires a high computational effort. This detection method is also called negative approach since it defines what is not allowed. On the other hand, anomaly detection overcomes these problems. However, it is prone to more false positives. It looks for anomalous system activity by defining the normal behavior of the web application. Then, actions deviating more than a threshold from the defined normal behavior are tagged as intrusive. The base idea is "denying everything unless explicitly allowed". A disadvantage is that in complex environments, obtaining an accurate and updated description of the normal behavior might not be an easy task. Anomaly detection is also referred to as positive approach since it defines the allowed application behavior.

Anomaly IDSs are usually trained first and tested later [García-Teodoro et al., 2009]. During the training the system learns several models that describe the normal behavior of the web application. After the training phase, the system is operative. During the test phase the system classifies incoming traffic. Later, the performance of the system can be evaluated. It is done by contrasting the classification information of the system, obtained in the test phase, with the corresponding label in the traffic.

Web Application Firewalls (WAFs) are considered a particular case of IDSs. They are specialized in working with web traffic. IDSs and WAFs are explained in more detail in Chapter 2.

1.3 Motivation

The global motivation of this thesis is improving the security of web applications. It means reducing the gap between attacks and defense. This would help millions of people to enjoy a safe Internet experience. In order to do that, this thesis basically focuses on five aspects:

- Designing several WAFs.
- Creating a new dataset for evaluating WAFs.

- Analyzing how the number of training requests influences the detection results.
- Searching for new feature extraction methods.
- Comparing different WAFs.

These aspects are further explained in the following sections.

The motivation of these five points is presented next:

- Regarding the design of WAFs, stochastic and Machine Learning (ML) techniques are applied in order to **improve both the effectiveness and efficiency of this type of systems**. Effectiveness measures how well a system detects. Effective systems are those that detect attacks at the same time that they do not raise false alarms. Whereas, efficiency considers how to perform the detection in an optimal way. It considers the amount of resources that the system needs to reach such detection results. Additionally, it is desired that WAFs are designed following the **simplicity principle**.

- Detection results of WAFs are directly influenced by the dataset selected. Choosing an adequate dataset to evaluate these systems is critical. However, acquiring labeled and quality datasets is an open problem in web intrusion detection [Sommer and Paxson, 2010]. Many of the existing datasets, such as *LBNL*, *DARPA* and *UNB ISCX* count with a number of drawbacks. In some cases, these drawbacks can even turn the datasets unusable. To be considered appropriate, a dataset should fulfill a number of requirements, like containing both HTTP normal traffic and modern attacks. Additionally, they should be labeled, publicly available and not anonymized. Given the **importance of adequately evaluating IDSs** and considering the drawbacks of most existing datasets, creating adequate datasets for evaluating WAFs is a main motivation of this thesis.

- When evaluating IDSs, generally the whole dataset is used, regardless of its number of requests. Besides affecting the training phase duration, the number of training requests influences the detection results of the WAFs (i.e., it determines the number of attacks detected and the number of false alarms). Studying the influence of the number of training requests in the detection results can reveal useful information for **reducing the number of training requests needed to achieve certain results**. Although this is an important issue for the optimization of the WAF resources, it has been scarcely studied.

- Features extracted from requests affect the detection results of WAFs. Depending on how well these features can distinguish between normal traffic and attacks, the detection results of the system vary. Thus, studying feature extraction methods is relevant for optimizing WAFs. The motivation in this point is to **improve the results of current extraction techniques, without using a big amount of features**.
- For the **comparison of different WAFs**, a labeled and adequate dataset is necessary. However, existing WAFs are frequently evaluated independently, i.e., without the same conditions or the same data. This impedes the comparison of different WAFs. Filling this gap is the motivation of this thesis.

1.4 Objectives

The main goal of this thesis is optimizing the security capabilities of WAFs based on stochastic and ML detection techniques, as well as creating an adequate dataset that allows the evaluation and comparison of these systems.

Concretely, the objectives of this work are the following:

- **Design and implement anomaly-based WAFs that use stochastic and machine learning detection algorithms. The challenge is that they work at high-speed, reach high detection results, consume low resources and count with a simple design.**

  In particular, stochastic algorithms refer to statistical techniques and Markov chains. Regarding ML, the algorithms applied are decision trees (DT).

  The systems should analyze HTTP traffic and detect attacks that involve a single request, being able to detect multiple types of web attacks.

Regarding the detection results, the goal is that the systems achieve a high detection rate (DR) while their false positive rate (FPR) is low. Detection rate measures the number of attacks that are detected and false positive rate measures false alarms.

The resource consumption, measured by the number of features, should be low.

These objectives should be reached by using a simple design, that does not increment the processing time of the system (time consumed to process a single request). This simplicity criteria makes reference to Ockham's razor. This principle states that among competing hypotheses that predict equally well, the one with the fewest assumptions should be selected. Therefore, the simplest, the best (provided that it is equally effective).

- **Study how the number of requests used in the training phase influences the detection results of the system**.

  The objective is to study which is the minimum number of requests necessary to train the system in order to achieve a certain detection result of the system. Also to know if using higher number of training requests implies obtaining better results.

- **Propose new feature extraction methods for machine learning that improve the detection results of existing strategies and consume low resources**.

  The resource consumption is measured in terms of the number of features used.

- **Build an adequate dataset to train and test WAFs**.

  In order to be considered adequate and overcome the disadvantages of the existing datasets, the objective is to build a labeled and public dataset. Additionally, it should contain modern attacks and its requests should not be anonymized.

  Working with a common dataset allows researchers to compare their systems. This aspect is related to the next objective.

- **Compare the behavior of the proposed detection techniques (statistical methods, Markov chains and decision trees) to determine which of them is the most appropriate for a given scenario**.

  For this aim, the same dataset is used to evaluate the different detection techniques. Their behavior is compared in regard to their detection results, processing time and number of training requests needed.

1.5 Structure of this thesis

The remaining of this dissertation is structured as follows:

- **Intrusion Detection: Concepts and Related Work**.

  Chapter 2 firstly introduces concepts about intrusion detection and explains different criteria used to classify IDSs. Next, it describes related detection systems in the literature, including a review of different methods used for the preprocessing phases of the IDSs (feature extraction and feature selection) and for the processing step. This chapter also highlights the importance of studying the number of training requests.

- **Data Acquisition for Web Intrusion Detection**.

  Chapter 3 discusses problems on existing datasets for evaluating WAFs and proposes a solution: creating the new *CSIC* dataset. Details about its generation process and application are explained.

- **Stochastic detection techniques for web intrusion detection**.

  Chapter 4 presents two WAFs based on stochastic detection techniques. The chapter covers the theoretical design and the empirical experimentation of these WAFs. One of the systems is based on statistical techniques and the other one on Markov chains. Details about different aspects related to the design of the systems are supplied, including how the system acquires knowledge, which detection models are used and how the detection process is designed. Later, the chapter covers aspects related to the empirical stage. The experimental settings and the measures used to evaluate the systems are presented. At last, experimental results, as well as a study about the influence of the training requests in the detection results, are shown.

- **Machine learning techniques for web intrusion detection**.

  Chapter 5 presents a detection system within the machine learning approach. The decision trees used in this work are explained and it is argued why they are chosen. Several new feature extraction methods are proposed. They combine both expert knowledge and n-grams and they are able to improve the performance of these two approaches separately. Furthermore, feature selection is applied in order to reduce the number of irrelevant and redundant features. The Generic Feature Selection ($GeFS$) measure chosen for feature selection is explained, as well as the criteria used to select its instances. The setup and development of the experiments are shown, as well as the results obtained. A detailed discussion of the results is given. Moreover, a comparison of the stochastic and ML systems is presented.

- **Conclusions, contributions and future work**.

  The conclusions extracted from the elaboration of this dissertation are explained in Chapter 6. Additionally, the contributions are analyzed and the resulting publications are summarized. Finally, future research lines are presented.

Chapter 2

Intrusion Detection: Concepts and Related Work

"What characterizes a science man is not
owning knowledge or irrefutable truth,
but the altruistic and incessant search
of the truth."

— Karl Popper

This chapter introduces concepts related to intrusion detection. Intrusion detection systems are in charge of monitoring the traffic in order to detect malicious actions. Web application firewalls are a particular case of IDSs that analyze HTTP traffic. The different components of these systems are explained, as well as diverse criteria used to classify them. Two phases are usually distinguished in IDSs: preprocessing and processing. Preprocessing comprises feature extraction and feature selection. This chapter explains what these phases consist in and reviews related papers in the literature. Regarding the processing step, existing IDSs and WAFs that apply statistical techniques, Markov chains and decision trees are analyzed. Additionally, the importance of studying the number of training requests is discussed. Finally, the conclusions section shows some questions that are still open in the state of the art and that find answer in the following chapters.

2.1 Initial concepts

In this section several concepts related to intrusion detection and web security are introduced.

2.1.1 Web applications

A web application typically consists of a finite number of web pages. The number of pages is usually small and most of them are dynamic. Dynamic pages are those that retrieve their content from a database on demand [Grove, 2010].

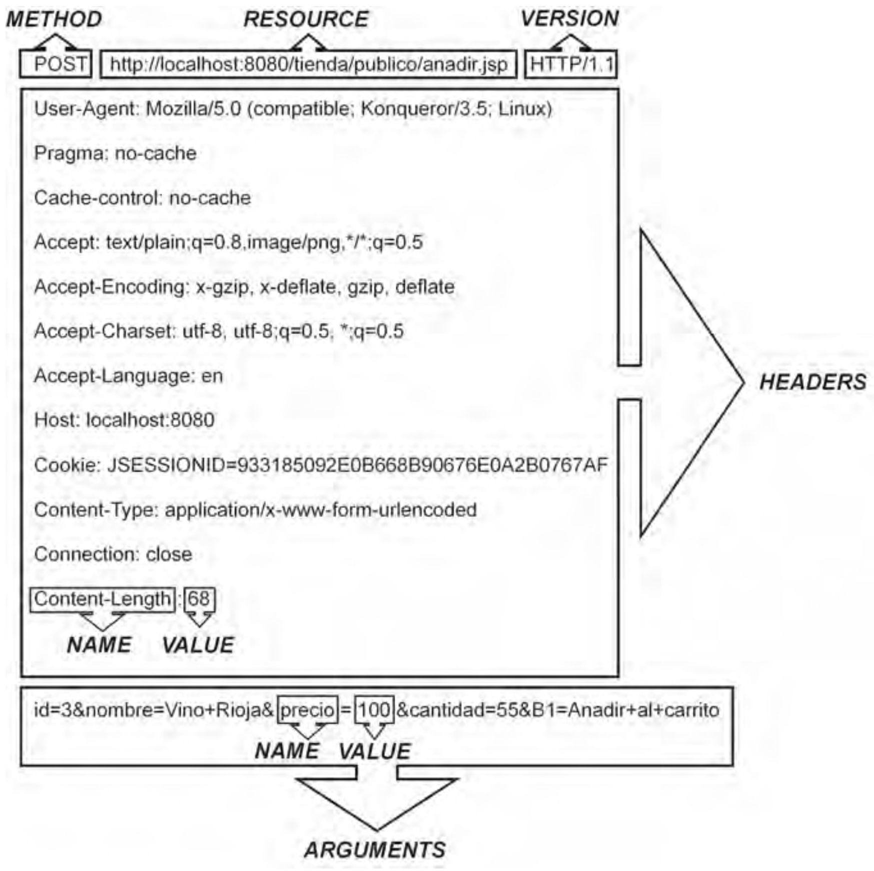
Web applications can be considered a particular case of the client-server architecture, where the application is hosted on a web server and a client requests pages. The communication between both is done through the HTTP protocol.

The structure of web applications is generally organized in three tiers: presentation, application and storage [Grove, 2010]. The presentation tier is responsible for the user interface, that usually consists of a web browser. The second tier hosts the web server, which handles all application processes. Typically, the web server is an engine using some dynamic web content technology, such as ASP, ASP.NET, CGI, JSP/Java, PHP, Perl, Python, Ruby on Rails or Struts2. The storage tier is in charge of handling data the application requires or provides, generally represented by a database.

2.1.2 HTTP protocol

Hypertext Transfer Protocol is an application layer protocol for distributed, collaborative, hypermedia information systems [Fielding et al., 1999]. The protocol enables the communication between the client and the web server. It governs the way documents are exchanged and establishes the format of the requests and responses. Although there are other drafts, the current stable version of this protocol is HTTP/1.1, released in 1999. HTTP is a client-server protocol. It works as follows: the client requests documents or services to the server and this one sends back a response to the client. In this protocol, each transaction is independent of the preceding and following ones. That is, it is a stateless protocol. It means that it does not record information about user's sessions. The protocol offers additional methods to store this information, like the usage of cookies. They are small pieces of data sent by a website that are stored in a user's web browser [Grove, 2010].

HTTP requests are composed of: HTTP method, resource, version of the HTTP protocol, headers and arguments. An example of an HTTP request and its components is shown in Fig. 2.1.


```
POST http://localhost:8080/tienda/publico/anadir.jsp HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux)
Pragma: no-cache
Cache-control: no-cache
Accept: text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=933185092E0B668B90676E0A2B0767AF
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 68

id=3&nombre=Vino+Rioja&precio=100&cantidad=55&B1=Anadir+al+carrito
```

Figure 2.1 – Example of an HTTP request and its components.

2.1.3 Web attacks

According to OWASP, attacks are techniques that attackers use to exploit vulnerabilities in applications [OWASP, 2013]. They pursue malicious intentions, such as getting illegal access, manipulating or disabling systems. In its report from 2009, the SANS Institute estimated that attacks against web applications constitute more than 60% of the total attack attempts observed on the Internet [Higgins, 2009]. Moreover, recent studies reveal that the number of web attacks is increasingly growing [Wood, 2014].

There are multiple types of web attacks, such as Structured Query Language (SQL) injection, content spoofing, cross-site scripting (XSS), buffer overflow, cross-site request forgery, denial of service, Operating System (OS) commanding, path traversal or Lightweight Directory Access Protocol (LDAP) injection.

Web attacks can imply drastic consequences, such as gaining access to databases with sensible user data, modification of web pages, non-availability of service, execution of the attacker's code, privilege escalation or access to unauthorized files.

One of the common causes of attacks is a poor validation of the user-supplied input. Such is the case of the well known SQL injection and XSS attacks. These attacks are two of the most frequent web attacks [Acunetix, 2014]. According to the Web Hacking Incident Database [Barnett et al., 2013], in 2011 SQL injection constituted 18.87% of attacks and XSS 12.58%. Acunetix [Acunetix, 2014] claims that the distribution of web attacks is the one represented in Fig. 2.2. Other attacks, like HTTP Parameter Pollution (HPP) [Carettoni and di Paola, 2009] have not received so much attention so far. Nevertheless, this attack constitutes a real threat for web applications, as is shown in our work [Balduzzi et al., 2011]. It presents the first automated tool for the discovery of HPP vulnerabilities. Our prototype PAPAS (PArameter Pollution Analysis System) was applied to more than 5 000 popular websites. The experimental results show that about 30% of the analyzed websites contain vulnerable parameters and that 46.8% of the discovered vulnerabilities (i.e., 14% of the total websites) can be exploited via HPP attacks.

Web attacks can be classified as either static or dynamic [Alvarez and Petrović, 2003a], [Alvarez and Petrović, 2003b].

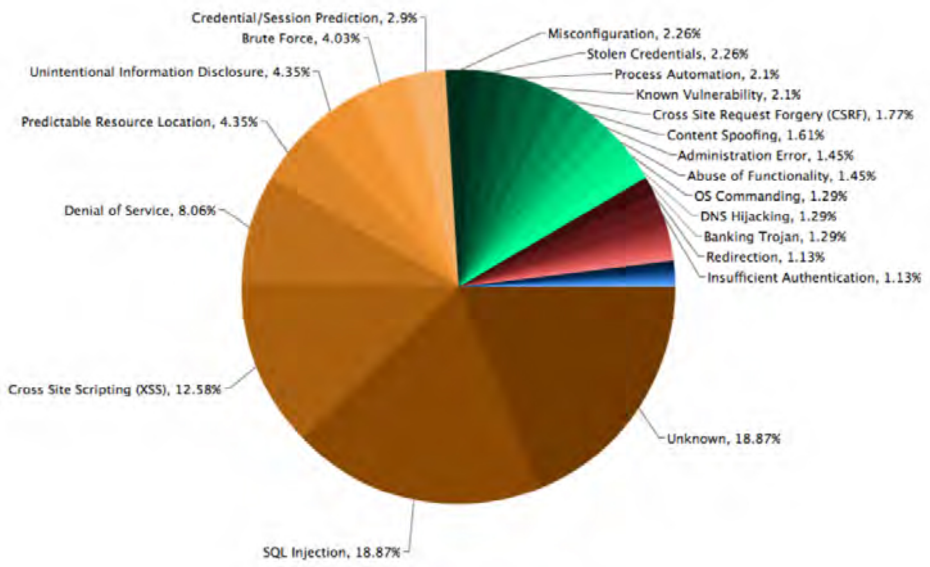


Figure 2.2 – Distribution of web attacks, according to Acunetix, 2014.

Static web attacks look for security vulnerabilities in a web application platform: web server, application server, database server, firewall, operating system and third-party components (such as cryptographic modules or payment gateways). These security threats comprise well-known vulnerabilities and erroneous configurations. A common characteristic of all these vulnerabilities is that they request pages, file extensions, or elements that do not form part of the web application as intended for the end user. Differently, dynamic web attacks only request legal pages of the application but they harm the expected parameters. Manipulation of input arguments can lead to different attacks of variable impact, such as errors in the application that disclose information about the platform, XSS attacks that steal information about other users or command execution by means of buffer overflows.

Several efforts have been made in creating a more detailed classification of web attacks. These efforts come, on the one hand, from organizations and on the other hand, from the scientific community. Some of the classifications from organizations are OWASP Top 10 [OWASP, 2013], Web Application Security Consortium (WASC) Threat Classification [WASC, 2010], Common Weakness Enumeration [MITRE, 2014b] and Common Attack Pattern Enumeration and Classification [MITRE, 2012]. Examples of scientific research are [Alvarez and Petrović, 2003a], [Lai et al., 2008], [Seo et al., 2004]. Although work is being done into this direction, there is not a clear consensus about a taxonomy for web attacks.

Web attacks can be performed by different actors, that can be divided into two types:

- **Outsiders**. They are external attackers, who do not belong to the organization.
- **Insiders**. They are inside the organization, therefore they can have easier access to certain systems. Ben Salem et al. distinguish between two types of insiders: traitors and masquerades [Salem et al., 2008]. These authors define a traitor as "a legitimate user within an organization who has been granted access to systems and information resources, but whose actions are counter to policy, and whose goal is to negatively affect confidentially, integrity, or availability of some information asset." Masqueraders are attackers who succeed in stealing a legitimate user's identity and impersonate another user for malicious purposes.

A zero-day attack is a computer attack exploiting a vulnerability that has not been disclosed publicly [Bilge and Dumitras, 2012]. It is difficult to defend against zero-day attacks because of their nature: as long as the vulnerability is not known, the affected software cannot be patched and signature-based defense products cannot detect the attack. Unpatched vulnerabilities can be very dangerous if exploited by attackers. It is estimated that the market value of a new vulnerability ranges between \$5 000 - \$250 000 [Bilge and Dumitras, 2012].

Acquiring a deep knowledge and understanding of web attacks is critical for developing adequate strategies and systems that protect against them.

2.2 Intrusion detection systems

The diversity of threats and the vulnerabilities of web applications make evident the necessity of protecting them. Web Application Security Statistics reported in 2008 an amount of 97 554 detected vulnerabilities of different risk levels in their study of 12 186 analyzed sites [WASC, 2008]. About 49% of the analyzed web applications contained vulnerabilities of high risk level (urgent and critical). In order to fulfill this need of protection, a range of solutions have been adopted. Some of these methods are focused on secure code development. This is the case of static and dynamic analysis techniques. The analysis of the code might not be an easy task when it is complex or it contains thousands of lines. Other solutions are external to the web application and do not require the modification of the application's code in order to protect it. As mentioned, this is the case of IDSs, that protect an application independently of whether it is correctly programmed or not.

In their guide to intrusion detection and prevention systems [Scarfone and Mell, 2007], the National Institute of Standards and Technology (NIST) defines intrusion detection as "the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices". Although many incidents are malicious in nature, many others are not, like user mistakes while typing. Therefore distinguishing benign from anomalous actions is the challenge of intrusion detection. In the same guide the problem of intrusion prevention is described as "the process of performing intrusion detection and attempting to stop detected possible incidents". NIST defines an IDS as "software that automates the intrusion detection process".

Solving the intrusion detection and prevention problem has some difficulties associated. Sommer and Paxson argue why anomaly intrusion detection is more difficult to solve than other problems in computer science. Some of the reasons the authors give are the high cost of errors and the variability in the input data [Sommer and Paxson, 2010]. In this thesis, various techniques for detecting intrusions will be presented.

2.2.1 Intrusion detection system classification

IDSs can be classified attending to different criteria, like the response mode, the location and the detection methodologies. These classification criteria can be seen in Fig. 2.3. In the following, they are further explained.

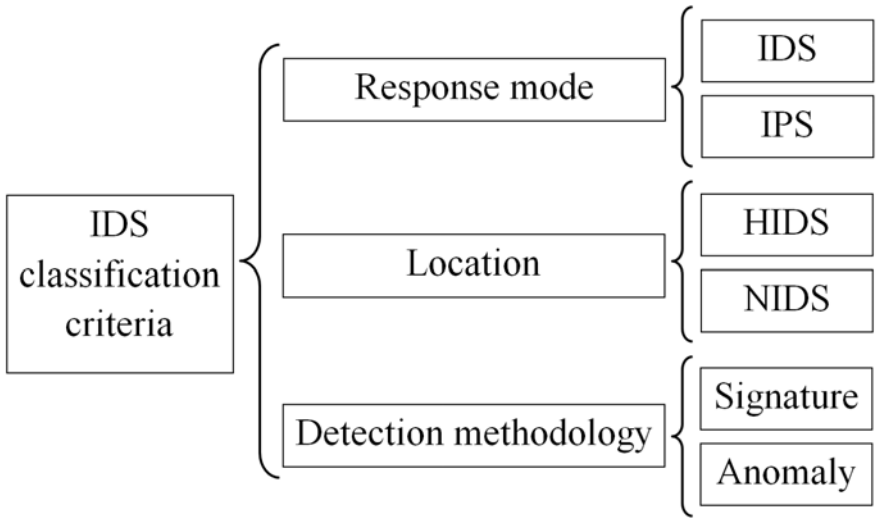


Figure 2.3 – Criteria to classify IDSs.

2.2.1.1 Classification according to the response mode

Depending on the action carried out by the detection system when an anomaly is detected, different types of systems can be distinguished:

- **Intrusion Detection System**: When an alert is detected, IDSs raise an alarm to acknowledge the event. They work in a passive mode.

- **Intrusion Prevention System** (IPS): They are active. NIST defines them as “software that has all the capabilities of an IDS and can also attempt to stop possible incidents” [Scarfone and Mell, 2007]. Hence, besides raising an alarm, they can block the anomaly or perform actions to neutralize the attack, in order to protect the server from being reached by malicious activity. An example of IPS in the literature is Masibty [Criscione and Zanero, 2009]. It detects XSS and SQL injection attacks. Another example is TokDoc [Krueger et al., 2010], that replaces the detected malicious content by near normal values.

There are systems that offer the possibility to work on both modes, letting the Information Technology (IT) professional configure the mode to be used.

2.2.1.2 Classification according to the location

Regarding the location of the system (or the source of the audit data), two categories of systems can be differentiated:

- **Host-based systems** (HIDSs) monitor the characteristics of a single host and the events occurring within it looking for suspicious activity. They can monitor different aspects, such as system logs, running processes, application activity, file access and modification, and configuration changes on the system or applications [Scarfone and Mell, 2007]. The analysis of system calls is usually performed for detection in such systems. This is the case of the seminal paper [Forrest et al., 1996]. Other examples are given in [Naiman, 2004] and [Lanzi et al., 2010].
- **Network-based systems** (NIDSs) monitor network traffic for particular network segments or devices in order to identify suspicious activity. They are most commonly deployed at a boundary between networks, such as in proximity to border firewalls or routers, Virtual Private Network (VPN) servers or remote access servers [Scarfone and Mell, 2007]. Examples of these systems can be found in [Kruegel et al., 2005b] and [Ariu et al., 2011].

When protecting an information system, usually a mixture of both NIDSs and HIDSs is necessary for a complete protection. Since HIDSs protect a specific host, they are commonly deployed on critical hosts, such as publicly accessible servers or servers containing sensitive information.

2.2.1.3 Classification according to the detection methodology

Several methodologies have been used to detect incidents. They are divided into three groups: signature-based, anomaly-based and hybrid. Although these concepts were introduced in Sec. 1.2, here they are described in more detail.

- **Signature-based detection**. It is also called misuse detection or negative approach. This approach looks for signatures (patterns) of known threats, that aim to exploit weaknesses in system and application software. It uses pattern matching techniques against a database of attack signatures. It is useful to detect already known attacks or their slight variations. However signature-based detection is not able to defend against new attacks, threats disguised by the use of evasion techniques or malicious variants of known threats that defeat the pattern recognition engine. Considering that new attacks appear constantly, this is not a desirable situation. The report by the European Union Agency for Network and Information Security (ENISA) claims that the volume of web-based attacks per day increased by 93% from 2009 to 2010, with 40 million attacks a day recorded in September 2010 [Ryck et al., 2011].

Additionally, attack pattern matching systems require large signature databases to be constantly updated. Moreover, the comparison of incoming traffic against every signature in the database requires a high computational effort and increases the time necessary to process requests.

Snort is a well-known signature-based IDS [Roesch, 1999]. It is free and open source, and it can act either as an IDS or an IPS. There is a big number of available signatures for Snort, described in a particular language that allows inspecting packets at different levels. The Malware Information Sharing Platform [MISP, 2015] is a software that helps to generate IDS rules.

Bro [Paxson, 2015] is another known open source IDS. It parses network traffic to extract its application level semantics. The network activity is compared with attack patterns or unusual activities. A specialized policy language is used to tailor its operation [Varadarajan, 2012].

In the academic field efforts have also been done in the direction of signature-based IDSs. For example, Anitha and Vaidehi used signatures for detecting application layer attacks. The signatures are formulated using regular expressions or finite state automata [Anitha and Vaidehi, 2006]. To deal with the constantly increasing number of rules that has to be compared to input event streams, the authors proposed a semantic classification tree that restructures the signature rules in order to reduce the redundant checks. Goyal et al. applied a genetic algorithm that uses a set of classification rules generated from a predefined intrusion behavior [Goyal and Aggarwal, 2012]. A genetic algorithm is a search heuristic that mimics the process of natural selection [Mitchell, 1998]. Meng et al. proposed hash-based contextual signatures in order to reduce the processing burden of signature matching and filter non-critical alarms [Meng and Kwok, 2014].

- **Anomaly-based detection.** This approach is also known as positive approach. Anomaly-based detection looks for behavior or use of computer resources deviating from "normal" or "common" behavior. The underlying principle of this approach is that attack behavior differs enough from normal user behavior to be detected by cataloging and identifying the differences involved. These systems have profiles that represent the normal behavior of applications, users, network connections, hosts and so on. Anomaly-based systems permit discerning normal traffic from suspicious activity without signature matching. The main idea of this approach is "denying everything unless explicitly allowed". The difficulty of such systems is obtaining an up-to-date, precise and feasible picture of the normal behavior. It is specially hard to draw when working in complex environments. In order to perform an accurate detection, it is important that the profiles do not include any malicious activity. Otherwise that activity would not be flagged as malicious in the detection phase.

  The major advantage of this method is its effectiveness in detecting previously unknown threats and their variations. As negative aspects, these systems are prone to more false positives and their profiles might need updating when the applications, users or network change. Additionally, determining why a particular alert was generated and validating that it is not a false positive might be a difficult task.

  Anomaly-based systems usually comprise a training phase where the normal behavior is learnt, and a test phase where the knowledge acquired is used to tag the requests.

  A known anomaly system was presented by Kruegel et al. in 2005. It uses a number of models that capture the normal behavior of web traffic [Kruegel et al., 2005b]. Other examples of anomaly-based systems are explained in [El-Alfy and Al-Obeidat, 2014] and [Lin et al., 2012].

- **Hybrid systems.** They combine the two approaches mentioned before. The paper by Tombini et al. studies different possibilities of combining anomaly and misuse detection. The authors conclude that depending on the input dataset and on the designer objective, each combination presents certain advantages [Tombini et al., 2004]. After measuring some characteristics on web servers the authors propose an architecture where an anomaly detector acts in first place and it feeds a misuse detector with potentially intrusive events.

  Depren et al. proposed an IDS architecture utilizing both anomaly and misuse detection approaches. This hybrid architecture consists of an anomaly detection module, a misuse detection module and a decision support system combining the results of these two detection modules [Depren et al., 2005].

  Bringas et al. presented ESIDE-Depian [Bringas et al., 2008]. As misuse detection module the system uses Snort and as anomaly detection component, a Bayesian network algorithm is applied.

  The paper by Sandip et al. also presents a hybrid IPS [Sandip et al., 2012]. The signature module employs conditional random fields to detect known intrusions in real time. Conditional random fields are a framework for building probabilistic models to segment and label sequence data [Lafferty et al., 2001]. The anomaly detection module uses the outlier detection provided by the conditional random field to detect unknown intrusions.

  The advantage of hybrid systems is that they can detect more attacks than signature-based or anomaly-based systems individually, while raising less false positives. However they require more time and resources to process the requests.

2.2.2 Structure of intrusion detection systems

According to the Common Intrusion Detection Framework (CIDF) [Schnackenberg and Tung, 1999], IDSs are composed by four components: event generators, event analyzers, response units and event databases [Porras et al., 1999]. They are explained next:

- **Event generators** obtain events from a computational environment and turn the occurrences of the environment into objects with a format that can be used by the subsequent component of the system. For example, the decoding process would be included in this phase.

- **Event analyzers** take the objects from the previous component and analyze them. Here two subcomponents can be distinguished: preprocessor and detection engine. The first one performs all actions before the classification of the incoming requests, such as parsing the requests, feature extraction and feature selection. The detection engine analyzes the requests searching for intrusions. These two stages will be further analyzed later in this chapter.

- **Event databases** store events for later retrieval.

- **Response units** consume objects to carry out some kind of action. The action depends on the type of system. IDSs raise an alarm when intrusions are found, while IPSs block the requests to avoid them reaching the target server.

2.2.3 Web application firewalls

IDSs are general systems that can analyze traffic corresponding to different protocols, such as HTTP, FTP, TCP, etc. Each of these protocols has different characteristics. When analyzing HTTP traffic, conventional firewalls operating at network and transport layers are not enough to protect against web-specific attacks. To be effective, systems analyzing traffic at the application layer are necessary.

Web application firewalls are particular IDSs specialized on analyzing HTTP traffic in order to detect web attacks.

According to WASC, a WAF is "an intermediary device, sitting between a web client and a web server, analyzing Open System Interconnection (OSI) Layer-7 messages for violations in the programmed security policy. A web application firewall is used as a security device protecting the web server from attack" [WASC, 2004]. Furthermore, WASC states that "WAFs solutions are capable of preventing attacks that network firewalls and intrusion detection systems cannot, and they do not require modification of the application source code" [WASC, 2006].

It is important that WAFs are designed to be both effective and efficient. Being effective means detecting attacks while not raising false alarms. Besides detecting correctly, being efficient implies low computational complexity and resource consumption. This is the reason why efficiency is critical in WAFs operating in real-time environments or in scenarios with resource constraints.

2.3 Preprocessing

Data preprocessing is extensively recognized as an important stage in anomaly detection [Davis and Clark, 2011]. Preprocessing comprises all tasks carried out before the formal processing of data, like dataset creation, data cleaning, normalization, transformation, feature extraction and feature selection [Kotsiantis et al., 2006]. From them, this thesis focuses on dataset creation, feature extraction and feature selection. Dataset creation is covered in Chapter 3 while feature extraction and feature selection are explained in this section as well as in Sec. 5.3.1 and Sec. 5.3.2.

Preprocessing requires a high amount of effort in terms of both resources and time, then it is important to pay attention to this previous step. According to Kurgan and Musilek, it is usually assumed that about half of the effort required for a project is spent on data preparation [Kurgan and Musilek, 2006]. The preprocessing step does not only affect the efficiency of the data storage but also the performance of the detection system [Davis and Clark, 2011].

Many intrusion detection systems are designed by applying feature extraction, feature selection and a classification algorithm. These two preprocessing steps are explained next.

2.3.1 Feature extraction

Feature extraction consists in determining representative features from the original data. Its aim is to obtain the appropriate features that represent regularities of the original dataset. This phase is crucial for the success of the classification algorithms [Lim et al., 2007].

Features can be either extracted by looking at patterns in the data, or by looking at the relationships between patterns. For the last option, data mining methods, such as association rule learning and frequent episode for sequence analysis, are used for feature extraction [Davis and Clark, 2011]. Association rule learning is a method for discovering relations between variables in large databases [Piatetsky-Shapiro, 1991]. An association rule is an expression $X \Rightarrow Y$, where X and Y are sets of items. This expression means that transactions of a database that contains X, tend to contain Y [Agrawal and Srikant, 1994]. An episode is a collection of events describing actions of users or systems that occur relatively close to each other in a given partial order. The technique consists in discovering frequently occurring episodes in a sequence in order to produce rules for describing or predicting the behavior of the sequence [Mannila et al., 1997].

Feature extraction can be done either manually, automatically or combining both of them:

- Many professionals rely on their **expert knowledge** to define a set of features that can distinguish between normal traffic and attacks. This approach often leads to high and reliable attack detection rates. However, it is a manual process and not quickly adaptive to changing network environments. It has been applied in intrusion detection in works like [Kruegel et al., 2005b], [Sriraghavan, 2008] or [Criscione and Zanero, 2009].

- **Automatic methods** overcome the drawback of being manual. Nevertheless frequently their results are less precise than those obtained by means of expert knowledge methods. One of the methods generally used in intrusion detection for feature extraction are n-grams. They are explained in Sec. 2.3.1.1.

- A third alternative is **combining the two previously mentioned methods**. However, this option has not been much explored in intrusion detection. One of the few papers found about that is [Rieck, 2009]. In this work Rieck extracted both manual and automatic features. However, the experiments were performed for every isolated group of features and their results did not consider the combination of features from different nature [Rieck, 2009]. Kloft et al. proposed a feature selection method. They made experiments with a uniform mixture of n-grams and expert knowledge features [Kloft et al., 2008].

2.3.1.1 N-grams

Models based on n-grams originate from the field of information retrieval and natural language processing. They are "language-independent" models that consist in fixed length overlapping symbol segments [Kowalski, 2010]. N-grams are not a classification technique themselves. They are usually applied for feature extraction or they are used in combination with other techniques for the classification phase. One of the advantages of this model is that it does not need domain knowledge, thus it is easy to automate. In the case of intrusion detection, if a payload is considered a string, then a n-gram is a substring of n characters. Let S be a space of all possible n-grams ($n \geq 1$). When working with byte sequences, S has the size of 2^{8n} (considering 8-bits representation for each character): $S = \{n\text{-grams}_i | i = 1 \ldots 2^{8n}\}$. According to that, the number of 1-grams is 256, and this amount grows exponentially when n increases.

N-grams have been largely applied in intrusion detection, ranging from HIDSs to NIDSs, and from 1-grams to higher order n-grams. Generally, the method to extract features from n-grams is counting the number of appearances of each n-gram in a string. In the case of web traffic, the string is the HTTP request. In the following, a review of the literature in the application of n-grams to intrusion detection is presented.

Regarding HIDSs, Forrest et. al applied n-grams to system calls [Forrest et al., 1996]. This work is based on Artificial Immune Systems (AISs). AISs are algorithms and systems that use the human immune system as inspiration [Greensmith et al., 2010]. Several concepts from the immune field have been extracted and applied to solve real world science and engineering problems. Forrest et al. defined the sense of self and non-self for privileged Unix processes. In this context, self refers to normal processes and non-self to attacks. The concept of self was defined by using normal patterns of short sequences of system calls. In order to detect the non-self data on system call, a set of detectors were produced.

AccessMiner [Lanzi et al., 2010], presented in 2010, was designed for malware protection. Instead of building models on individual applications, it defines the normal behavior as sequences of system calls of a broad set of benign applications. To define the normal behavior, n-grams were applied to model short sequences of system calls.

PHAD (Packet Header Anomaly Detector) detects attacks using anomaly models based on n-gram statistics [Mahoney and Chan, 2001].

In relation to 1-grams, Wang et al. presented in 2004 their famous payload-based anomaly detector PAYL [Wang and Stolfo, 2004]. This system models payloads by using their byte value distribution. Following the anomaly approach, a model of the statistical distribution of 1-grams in normal requests is stored. When incoming requests are received for detection, their 1-gram distribution is analyzed. Each n-gram not present in the normal model increments the anomaly score of the packet. If the final anomaly score exceeds a predefined threshold, then the packet is tagged as anomalous.

The anomaly detector Anagram [Wang et al., 2006] increases the security of PAYL by modeling a mixture of high-order n-grams ($n > 1$). This mixture is designed to detect anomalous and suspicious network packet payloads. The system uses Bloom filters. They are essentially bit arrays of m bits, where any individual bit i is set if the hash of an input value, $\mod m$, is i. Anagram uses a Bloom filter to store n-grams of normal packets and another one for known attacks. In the test phase, the n-grams of the test packet are compared to both normal and attack filters. Payloads that contain too many n-grams not present in the normal Bloom filter, or present in the attack Bloom filter, are classified as anomalous.

Higher order n-grams have been also employed by Bolzoni and Etalle, who applied n-grams of different length for the algorithms used by their anomaly-based NIDS [Bolzoni et al., 2009].

Variations of n-gram extraction have also been applied. Rieck and Laskov introduced the concept of variable length n-grams (words). They presented a detection system based on language models [Rieck and Laskov, 2007], [Rieck and Laskov, 2006]. Their method proceeds by extracting language features, such as n-grams and words from connection payloads, before applying unsupervised anomaly detection. In order to differentiate between attacks and normal data, their proposal computes similarity measures between language models. These measures can be used to generate attack signatures.

Another variation was presented by Perdisci et al. Their technique, called $2v$-gram, consists of measuring features by using a sliding window. In $2v$-gram, this window has the peculiarity of covering two bytes which are v positions apart from each other in the payload. Their system McPAD is designed to mainly detect shell-code attacks in various forms [Perdisci et al., 2009]. It also acts successfully with advanced polymorphic blending attacks. Nevertheless, it does not perform well when the attacker tries to spread the attack over several attack packets.

Since this dissertation is focused on detecting attacks over the HTTP protocol, special emphasis is done in the review of systems specialized in detecting intrusions over web traffic.

***N*-grams on web traffic.** Regarding the particular case of web traffic, there is a variety of works applying n-grams to the inspection of the payloads.

Naiman analyzes statistically contiguous sequences of n system calls. These system calls proceed from processes generated by an HTTP daemon [Naiman, 2004]. The motivation is that occurrences of enough new n-grams in some localized time frame constitute evidence of innovative behavior, which is considered anomalous.

TokDoc is a prototype of a reverse proxy. Its particularity is that besides acting as an IDS, it can work as an IPS, healing the malicious requests [Krueger et al., 2010]. It has several detectors, being one of them based on 2-grams.

A variant of n-grams was employed in Spectrogram [Song et al., 2009]. Spectrogram is a machine learning based statistical anomaly detection sensor. It defends against web-layer code-injection attacks, such as Hypertext Preprocessor (PHP) file inclusion, SQL-injection, XSS and memory-layer exploits. It has an inference model that tracks the n-gram level transitions within a string. The gram size is an adjustable parameter of the system. Denoting x_i as the i^{th} character within a string, the likelihood of a n-gram is calculated as the likelihood of x_n, that is conditioned by the $n-1$ preceding characters. This strategy takes advantage of the overlapping nature of n-grams within an input string.

Some systems also employ a sliding window of width n instead of using n-grams. For instance, this is the case of HMMPayl [Ariu et al., 2011], that extracts features by using sliding windows over the sequence of bytes that represent the payloads. Their system later analyzes the features extracted by applying Hidden Markov Models. Markov chains are explained in Sec. 2.4.1.2 and Sec. 4.2.

2.3.2 Feature selection

A feature selection method finds the minimum set of features that maximizes the performance of a classification algorithm. This methodology follows the principle of parsimony (or Ockham's razor). This principle advocates for using models and procedures that contain all that is necessary for modeling but nothing more [Hawkins, 2004]. By reducing the number of features without negatively affecting the detection results, feature selection increases the available processing time and reduces the required system resources.

Feature selection is specially helpful when the number of features is high. This is the case of n-grams, where the number of possible n-grams increases exponentially with the value of n. This increase usually leads to the so called *curse of dimensionality* and to the computational complexity problem. The curse of dimensionality was coined by Bellman in 1961 to refer the fact that many algorithms that can cope with low dimensions, do not work well on high dimensionality. In machine learning, generalization can become exponentially harder as the dimensionality grows [Domingos, 2012]. Dimensionality is measured by the number of features. It should be considered that when the number of dimensions is elevated, generally more traffic is required for the training step.

Feature selection counts with a number of benefits [Nguyen, 2012], such as:

- General data reduction: limiting storage requirements.
- Performance improvement: increasing the algorithm speed.

- Data understanding: gaining knowledge about the process that generated the data.

Feature selection has been applied to intrusion detection. Li et al. experimentally tested that their system achieves better detection results and lower computational costs when feature selection is applied than when it is not [Li et al., 2009].

In machine learning, feature selection methods are typically classified into three categories: wrapper, filter and embedded models [Guyon et al., 2006], [Liu and Motoda, 2007]. They differ in the way they interact with the classifier.

- The **wrapper approach** aims to improve the results of the specific classifiers they work with. This approach employs the performance of learning algorithms to assess the quality of the features and select them accordingly. Using the learning algorithm implies a high consumption of time and computational resources. Examples of the wrapper approach are given in Sec. 2.3.2.1.

- The **filter model** directly considers statistical characteristics of a dataset without involving learning algorithms. Due to its computational efficiency, the filter method is used to select features from high-dimensional datasets. This is generally the case of datasets used in intrusion detection. A major challenge in the IDS feature selection process is to choose appropriate measures that can precisely determine the relevance and the relation between features of a given dataset. The relevance and the relation of features are usually characterized in terms of correlation or mutual information. Systems applying this approach are shown in Sec. 2.3.2.2.

- The **embedded model** of feature selection does not separate learning from feature selection. The embedded model integrates the selection of features in the model building [Nguyen, 2012]. An example of such model is the decision tree induction algorithm, which selects a feature for each intermediate node (those that are not leafs). No further section is shown for this type of systems.

2.3.2.1 Wrapper model

Within the wrapper approach, a big variety of algorithms has been applied in the intrusion detection context. Some of them are reviewed next. One of the examples was presented by Kloft et al., who generalized the vanilla Support Vector Data Description (SVDD) algorithm [Kloft et al., 2008]. SVDD obtains a spherically shaped boundary around a dataset that can be used to detect outliers [Tax and Duin, 2004].

Other examples, like [Sivatha Sindhu et al., 2012] and [Tsang et al., 2007], use genetic algorithms.

Li et al. used a modified random mutation hill climbing (RMHC) algorithm as search strategy to specify a candidate subset for evaluation. Hill climbing is a family of optimization techniques of local search. It is an iterative algorithm that reaches the best solution by replacing an element by another one in case the last one is nearer to the top of the hill. The process continues until no further improvements can be found [Simon, 2013]. Random mutation hill climbing selects randomly the next element to be examined. Li et al. also employed a modified linear Support Vector Machines (SVM) iterative procedure as wrapper approach to obtain the optimum feature subset [Li et al., 2009]. SVM are supervised learning models that aim to separate categories in a high dimensional feature space, by learning hyperplanes separated as much as possible. These models make possible to predict the category of a point based on where the representation of the point is located in the space.

Other authors used SVM together with Simulated Annealing (SA) to select features [Lin et al., 2012]. SA is a generic probabilistic metaheuristic to find an approximation to the global optimum of a given function in a large search space. In order to obtain a more extensive search, the algorithm slowly decreases the probability of accepting worse solutions as it explores the solution space [Van Laarhoven and Aarts, 1987].

De La Hoz et al. proposed a multi-objective approach, using the Non-dominated Sorting Genetic Algorithm II (NSGA-II) for feature selection [De la Hoz et al., 2014]. NSGA is a genetic-based algorithm for multi-objective optimization. NSGA- II was created to improve NSGA. In multi-objective optimization, trade-offs need to be taken between two or more conflicting objectives in order to reach optimal solutions. In this type of problems, there is not a single solution that simultaneously optimizes all objectives, but a number of optimal solutions, that are called Pareto optimals. Pareto optimals are those solutions where none of the objective functions can be improved without degrading some of the other objectives [Deb et al., 2002]. The two objectives considered by De La Hoz et al. were maximizing the classifier performance and minimizing the number of features. Afterward, this technique was applied over a Growing Hierarchical Self-Organizing Map (GHSOM) classifier. The Self-Organizing Map (SOM, also called Kohonen-Map) is a type of Artificial Neural Network (ANN) that performs unsupervised learning. The model consists of a number of neural processing units with a weight vector. SOMs differentiate from other ANNs because they use a neighborhood function to preserve the topological properties of the input space. GHSOMs overcome some disadvantages of SOMs, such as its static architecture and lack of representation of hierarchical relation. The GHSOM is a hierarchical structure of several layers, whose number of units, maps and layers are determined during the unsupervised learning process [Yang et al., 2010].

2.3.2.2 Filter model

Several works that apply the filter method can be found in literature. This section reviews some of them. Amiri et al. applied two algorithms for feature selection: linear correlation coefficient and mutual information [Amiri et al., 2011].

Correlation and mutual information are frequently used in feature selection. Examples of algorithms that use such criteria are the known Correlation Feature Selection (*CFS*) and the Minimal Redundancy Maximal Relevance (*mRMR*) measures. These measures are explained in Sec. 5.3.2. In their work, Nguyen et al. fused and generalized these measures in the so called *GeFS* measure [Nguyen et al., 2010b]. In [Nguyen et al., 2010c], the authors compared *GeFS* with other known algorithms, like SVM-wrapper, Markov-blanket and Classification & Regression Tree (CART). This is the measure chosen for feature selection in this thesis.

Principal Component Analysis (PCA) is a popular method used for dimensionality reduction. According to [Abdi and Williams, 2010], it is a multivariate technique that analyzes a data table where observations are described by several inter-correlated quantitative dependent variables. The goal of PCA is to extract the important information from the data table, to represent it as a set of new orthogonal variables called principal components, and to display patterns of similarity of the observations and variables as points in maps. This technique is frequently used to preprocess data and reduce the dimensionality of a space, what helps to perform detection more efficiently. Examples of its application in intrusion detection are [Wang et al., 2004], [Wang and Battiti, 2006], [Bouzida et al., 2004], [Shyu et al., 2003], [Bouzida et al., 2004], [Fonseca et al., 2008], [Jamdagni et al., 2013], [Zargar and Baghaie, 2012] and [Zhang and White, 2007]. Most of these works have been experimentally evaluated using the *KDD 99* dataset [University of California, 1999]. This dataset is a collection of simulated raw TCP dump data. The training dataset, collected during seven weeks, consists of 494 021 records. The test dataset covers two weeks of traffic. In each connection there are 41 attributes describing different features of the connection and a label assigned to each either as an attack type or as normal [Olusola et al., 2010].

2.4 Detection techniques

According to the classification of IDSs presented above in regard to their location and detection method, this thesis is focused on NIDSs and anomaly-based systems respectively.

Within the anomaly detection field, diverse types of algorithms have been used to perform intrusion detection. Although there are different proposals for classifying algorithms, a commonly accepted one is the classification used in [García-Teodoro et al., 2009]. It consists of three main categories:

- **Knowledge-based** techniques. They try to capture the behavior from the available system data [Debar et al., 1999]. These techniques rely on highly qualified expert knowledge, that might be insufficient in cases such as polimorphic or zero-day attacks. Polymorphic attacks are able to change its appearance with every instance, making its detection more difficult [Fogla et al., 2006].

- **Stochastic-based** methods. They observe the activity of the network and generate stochastic profiles to represent its behavior. As new events are processed, the system profiles are updated. Typically, the current profile is compared to the learned models. In case that current activity differs more than a threshold from the stored one, it is flagged as anomalous [García-Teodoro et al., 2009].

- **Machine learning** is concerned with how to construct computer programs that automatically learn with experience. This automation capability is very useful in the cyberspace, where big amounts of data need to be handled.

  Machine learning systems are based on the establishment of an explicit or implicit model that allows to categorize the analyzed patterns [Tsai et al., 2009].

This work is focused on the last two categories of techniques. Their advantage is that they do not need so much expert knowledge. Additionally, the applicability of machine learning and stochastic techniques coincides in many cases [García-Teodoro et al., 2009], what makes feasible to apply both of them in similar environments.

Next, the most important systems using stochastic techniques and machine learning are revised.

2.4.1 Stochastic techniques

Within stochastic techniques, statistical methods and Markov chains are the most widely used in intrusion detection. Next, a review of IDSs using these techniques is presented.

2.4.1.1 Statistical-based techniques

Statistical methods define models that are usually represented by thresholds, probabilities and basic statistical operators. Like stochastic models, statistical systems compare the incoming traffic to the learned models. In case the traffic is different enough to the learned models, the behavior is labeled as abnormal.

Regarding IDSs, several statistical-based systems have been designed. The first systems were univariate, such as Haystack [Smaha, 1988], Intrusion Detection Expert System(IDES) [Denning and Neumann, 1985], [Lunt et al., 1992] and its later version Next-Generation Intrusion Detection Expert System (NIDES) [Anderson et al., 1994], [Anderson et al., 1995]. Later, Ye et al. proposed a system based on a multi-variate statistical technique that considers the correlations between two or more metrics [Ye et al., 2002]. Mahoney and Chan also presented various systems that apply learning techniques in order to detect anomalies in the network traffic: Packet Header Anomaly Detector [Mahoney and Chan, 2001], LEarning Rules for Anomaly Detection (LERAD) [Mahoney and Chan, 2002a] and Application Layer Anomaly Detector (ALAD) [Mahoney and Chan, 2002b]. These systems use time-based models where the probability of an event depends on the time since it occurred last.

Statistical techniques on web traffic. Since this thesis focuses on web traffic, statistical-based systems specialized in detecting intrusions over web traffic are analyzed in this section.

In anomaly-based systems, a typical approach is combining multiple anomaly detectors, that employ different techniques, to describe the normal behavior of a target web application. Those activities differing from those models are tagged as anomalous. The system proposed in [Kruegel et al., 2005b] follows this approach. This system inspects log files to analyze individual Uniform Resource Identifiers (URIs) and parameters of GET requests whose return code is between 200 and 300. It means they are successful requests. The authors use a set of models for detecting malicious activity: attribute length, attribute character distribution, structure of the parameters (regularity of the non-printable characters), detection of anomalous values for an attribute (different values for a fixed-value attribute), attribute presence or absence, attribute order, access frequency, inter-request time delay, and invocation order of the server-side programs. Note that these models are obtained by means of expert knowledge. In their system, the task of a model is to assign a probability value either to the whole query or to one of the query's attributes. It is possible to associate several models to an attribute or to the whole query. Each model contributes to the calculation of the anomaly score, that is calculated as a weighted sum of the probability values returned by the models associated to the attribute or query. Note that in this approach, every model associated to the attribute or query needs to be processed for the calculation of the anomaly score. A query is reported as anomalous if any of the attribute or query anomaly scores is above the corresponding detection threshold. The assumption behind this scheme is that those feature values with a sufficiently low probability indicate a potential attack. A model can operate either in training or test mode. In the training mode, the system creates profiles for the normal values and establishes thresholds as the maximum anomaly score plus an adjustable percentage. In the test mode, anomaly scores are calculated and anomalous queries are reported. Their system is experimentally evaluated making use of traffic from universities in California and Vienna, and from a Google server. This approach was later generalized in [Robertson et al., 2006], where a technique to automatically translate suspicious web requests into anomaly signatures is presented.

A paper following the same scheme as Kruegel's work is [Sriraghavan and Lucchese, 2008]. It also monitors successful GET requests that contain parameters. The system automatically creates parameter profiles that are built using a multi-model approach. The following models are employed: unknown program identification, unknown parameter identification, attribute length, attribute character distribution and implementation of 6-bin character distribution algorithm. About the last model, the authors prove that 3-bin is more efficient than 6-bin. 3-bin distinguishes between letters, digits and non-alphanumeric characters. While 6-bin considers the relative frequency of 256 ASCII characters. Following the anomaly approach, these models have a learning phase and a detection one. For the experiments, this system uses attack-free traffic from the *DARPA 99* dataset and attacks from the *ECML/PKDD* dataset. Similarly to the system proposed in this thesis, automatic learning ensures that the system can be used with different types of web application environments, without the need of manual configuration.

Masibty [Criscione and Zanero, 2009] can act as an attack detector or a blocker. It uses the concept of entry point (EP) to deal with complex web applications. An EP is basically an augmented URI that is further specialized depending on parameters, session context and other influences. Masibty is specialized on the detection of SQL injection and XSS attacks. Its architecture presents different modules for anomaly detection. For example, for detecting SQL injection attacks it contains engines regarding the order, presence, numbers, token, distribution and length of the request parameters. Similarly to papers presented before, this system combines the results from all anomaly engines in order to calculate a single anomaly score for the whole request. For testing, the authors use their own dataset. In the experiments, some attacks are incorporated into the training dataset.

2.4.1.2 Markov chains

Markov chains are mathematical systems that undergo transitions from one state to another, between a finite or countable number of possible states. They are random processes usually characterized as memoryless: the next state depends only on the current state and not on the sequence of events that precedes it [Freedman, 1971].

During the training phase, the probabilities associated to the transitions are estimated to capture the normal behavior of the target system. The detection of anomalies is then carried out by comparing the anomaly score obtained for the observed sequences with a fixed threshold [García-Teodoro et al., 2009].

Markov chains have been used in intrusion detection. They have been applied in the context of host protection by authors such as Ye et al., that presented an anomaly detection technique based on Markov chains to study system calls of Solaris systems [Ye et al., 2004]. The paper by Bakhoum applies Markov chains in the context of network intrusion detection [Bakhoum, 2011a].

A variant of Markov chains are Hidden Markov Models (HMM). They are Markov processes with unobserved (hidden) states. In contrast to Markov models, in a HMM the state is not directly visible. In Markov chains the state is directly visible to the observer and therefore the state transition probabilities are the only parameters. The output of a HMM, that is dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by a HMM gives some information about the sequence of states [Elliott et al., 1995].

HMMs have been also applied to both host and network environments for improving their security. In order to represent normal behavior and recognize intrusions in system call datasets, Warrender et al. used a HMM with fully connected states (i.e., transitions are allowed from any state to any other state) [Warrender et al., 1999]. Yeung et al. employed HMMs, besides other techniques, in an anomaly system that models program profiles based on Unix system calls and user profiles based on Unix shell commands [Yeung and Ding, 2003]. Arnes et al. applied HMMs for a risk assessment tool integrated with an IDS [Arnes et al., 2006].

Markov chains on web traffic. In the particular case of systems inspecting web traffic, Markovian techniques have been extensively applied.

Estévez-Tapiador et al. presented an anomaly WAF that uses Markov chains to model HTTP traffic [Estévez-Tapiador et al., 2004]. Two variants of the system were proposed. The first one is an extension of the payload histograms, using an alphabet corresponding to ASCII code. For every character in a normal request, it calculates the probability that it is in the first position within the payload and the probability that it is followed by any other character. If the probability of a incoming request exceeds a given threshold, it is deemed as anomalous. The second variant incorporates protocol-dependent information. The idea behind it is that the probability of occurrence of certain strings within each section of the payload is not uniform throughout the request. In this approach the HTTP payloads are segmented into blocks of characters according to protocol delimiters. These blocks are stored in a dictionary, that is filled in the training phase by splitting normal requests. In the test phase, the dictionary is used for evaluating the normality of the incoming requests. For the experiments, a modification of the popular *DARPA 99* dataset is employed, taking normal traffic from the cited dataset and manually generating web attacks.

As mentioned, Spectrogram is focused on detecting local and remote file inclusion, XSS and SQLi attacks [Song et al., 2009]. It learns to recognize legitime content and structure of the web-layer script arguments. The rest of the HTTP request is not analyzed. It uses a configurable number of Markov chains to obtain a final likelihood score of the request being normal. Each Markov chain is in charge of calculating the likelihood of each character and then recovering the geometric mean of individual likelihoods. Differently to other works, that consider that the probability of a character depends only on the preceding character, Spectrogram considers that the probability of a character depends on the $n-1$ preceding characters. For the experiments the authors used their own dataset.

As mentioned, besides detecting anomalies in web traffic, TokDoc puts remedy to the detected attacks [Krueger et al., 2010]. It is done by locating and substituting anomalous values by others that are closer to them and, at the same time, are normal. In this way, it avoids malicious requests reaching a protected web server. Within the detectors used, one of them is based on Markov chains. For the construction of Markov chains, the authors consider 256 ASCII characters in the analysis of the arguments. For the experiments, their own created datasets were used.

HMMs have also been applied to web attack detection. One of them is HMM-Web, by Corona et al. It shows that HMMs are effective in detecting a wide range of either known or unknown attacks against web applications [Corona et al., 2009]. For the analysis of the requests two groups of characters are considered: alphanumeric and non-alphanumeric. Similarly to Spectrogram, HMM-Web uses an ensemble of HMMs to increase the performance, concretely it uses an ensemble per attribute and another one for the sequence of attributes. Then, it uses a maximum rule to select the model in the ensemble that best describes the analyzed sequence.

In [Hosseinkhani et al., 2011], an extension of HMM-Web is presented. It deploys a second dimension in order to improve the detection of input validation attacks in web applications. That is, the approach proposes a training phase with two dimensions: in the first one, the IDS learns normal values of the web application attributes. In the second one, the sequence of normal probabilities of these attributes is modeled. Both the training and detection phases use HMMs. After employing a group of HMM, the maximum probability rule is used to select the model in the ensemble that best describes the analyzed input value. For the experiments, a mixture of the *DARPA 99* and the *ECML/PKDD* datasets is used. A drawback of this system is that the second dimension introduces a delay in the detection process.

HMMPayl puts emphasis on reducing the false positive rate and for that, it exploits the power of HMM in modeling sequences of data [Ariu et al., 2011]. An ensemble of HMMs is used in parallel, being each HMM initialized with different random values. Then, a selector is used for choosing the best classifier selection strategy. Instead of analyzing the whole payload, they choose which sequences of the payload are used to classify it. Experiments are carried out using both public and private datasets.

2.4.2 Machine learning

Machine learning (ML) is a subfield of Artificial Intelligence that can help in the automation of processes. In 1959, Arthur Samuel defined it as a "field of study that gives computers the ability to learn without being explicitly programmed" [Samuel, 1959].

ML has been extensively applied in intrusion detection [Tsai et al., 2009]. Detection systems based on ML allow to quickly detect attacks while demanding much less manual work. Because of this reason, the approach is becoming increasingly important for computer security, especially when considering the huge amount of network data that IDSs need to analyze [Nguyen, 2012].

ML comprises a big range of algorithms that have been traditionally classified into two different approaches: supervised and unsupervised learning. In supervised learning algorithms the labels of the training dataset are available for the learning process. Examples of supervised learning algorithms are decision trees and support vector machines. Unsupervised learning algorithms, such as the K-means clustering, can still learn the normal and abnormal behavior of the dataset without their labels.

Several works have been presented in intrusion detection that apply ML algorithms. Some of them are:

- Bayesian networks [Kruegel et al., 2003], [Ihler et al., 2006], [Barbara et al., 2001], [Valdes and Skinner, 2000].

- Artificial Neural Networks (ANN) and Self-Organizing Maps (SOM) [Ramadas et al., 2003], [Xydas et al., 2008], [Ramos and Abraham, 2005], [Khalkhali, 2011], [Yang et al., 2010], [Bolzoni et al., 2006], [Depren et al., 2005], [Corchado and Herrero, 2011], [Ibrahim, 2010].

- Genetic Algorithms [Li, 2004], [Pillai et al., 2004], [Saniee Abadeh et al., 2007], [Abadeh et al., 2011].

- Clustering and K-nearest Neighbors [Zanero and Savaresi, 2004], [Zanero, 2006], [Hautamäki et al., 2004], [Sequeira and Zaki, 2002], [Liao and Vemuri, 2002], [Breunig et al., 2000], [Di Crescenzo et al., 2005], [Das et al., 2009], [Corona and Giacinto, 2010], [Dessiatnikoff et al., 2011], [Li et al., 2008], [Kirchner, 2010], [Davanzo et al., 2011].

- Decision Trees [Pfahringer, 2000], [Balon-Perin, 2012], [Depren et al., 2005], [Sangkatsanee et al., 2011a], [Sivatha Sindhu et al., 2012], [Rao et al., 2011], [Lin et al., 2012], [García et al., 2006], [Muniyandi et al., 2012].

- Support Vector Machines [Sung and Mukkamala, 2003], [Bolzoni et al., 2009], [Shrivastava and Jain, 2011], [Düssel et al., 2008].

Within the algorithms cited above, decision trees have been chosen in this dissertation to distinguish between normal and anomalous traffic. This family of algorithms is one of the most popular [Wu et al., 2007] and experimentally successful of the machine learning algorithms.

According to SAS company, “decision trees are tools for multiple variable analysis and can support in decisions using a tree-like graph” [De Ville, 2006]. They are widely used for classification and they can predict the output of new or unseen observations. One of their advantages is that human-understandable rules can be directly derived from them.

For the construction of the tree, the object of analysis is reflected in the root. The branches of the tree are built by considering the relationship between the object of analysis and one or more fields that serve as input fields to create the branches. The leaf nodes represent classification categories.

A proof of the successful application of decision trees in intrusion detection is the work by Pfahringer. He won the famous DARPA intrusion detection contest [Lippmann et al., 2000a] with an algorithm based on decision trees. The author constructed an ensemble of ten decision trees run fifty times [Pfahringer, 2000]. The ensemble is constructed by applying bagging and boosting, besides minimizing the error cost of predicting specific classes. According to Quinlan, "bagging and boosting form a set of classifiers that are combined by voting; bagging by generating replicated bootstrap samples of the data, and boosting by adjusting the weights of training instances" [Quinlan, 1996].

Depren applied a J.48 decision tree to classify various types of attack in the misuse module of a hybrid IDS [Depren et al., 2005], performing experiments on the *KDD 99* dataset.

Later, Sangkatsanee et. al studied the performance of several supervised machine learning techniques on intrusion detection [Sangkatsanee et al., 2011a]. These authors used decision trees, a rule-based learning algorithm, two neural networks and two Bayesian algorithms. From the various techniques mentioned, the experimental results showed that decision trees can outperform all other techniques. Then, the authors propose a detection engine of real-time IDSs based on decision trees.

Balon-Perin applied several machine learning algorithms, being one of them an ensemble of decision trees [Balon-Perin, 2012]. Each ensemble was specialized on the detection of one class of attack. Additionally, bagging techniques were used to increase the accuracy of IDSs.

Decision trees have also been used in conjunction with other techniques. An example is the work from Rao et al. [Rao et al., 2011], where decision trees are used together with clustering. Then, it mixes supervised and unsupervised learning. Concretely, the authors use K-means clustering and a ID3 decision tree to classify anomalous and normal activities in a computer network. In this system, the decision tree algorithm is used to refine the decision boundaries by learning the subgroups within the clusters created by the K-means algorithm.

Muniyandi et al. followed the same idea of the last mentioned paper, but using the C4.5 decision tree instead of the ID3 [Muniyandi et al., 2012].

The approach by Sivatha Sindhu et al. mixed neural networks and decision trees [Sivatha Sindhu et al., 2012]. The system was experimentally evaluated with a family of six decision tree classifiers, namely decision stump, C4.5, naïve Bayes' tree, random forest, random tree and representative tree model.

Lin et al. apply decision trees and simulated annealing. These techniques are used to obtain decision rules for new attacks and to improve the accuracy of classification [Lin et al., 2012].

Decision trees on web traffic. As can be seen, decision trees have been extensively applied for detecting intrusions in network traffic. They have proven to be very effective in this field, even outperforming other techniques. However, they have been scarcely applied to the particular case of web traffic.

One of the few works found about it is [García et al., 2006]. This paper applies an ID3 decision tree to distinguish a number of web attacks, including some of their variants. in particular, the system is able to detect the following attacks: SQL injection, XSS, code injection and directory traversal. The authors use their own-built dataset for the experimentation stage.

An advantage of using decision trees is that their rules are easy to read, thus, the root of an attack can be easily understood.

2.5 Importance of the number of training requests

The training phase is critical in intrusion detection systems. Depending on how the training is performed, the system will be able to perform detection with more or less precision. The shorter the training phase is, the least computational time and resources are necessary. However, if the system is not trained with enough requests, it will not be effective against attacks. Not having sufficient data to completely determine a correct classifier or having a too big dataset might lead to certain machine learning algorithms to overfitting. Overfitting produces a classifier not grounded in reality that is only modeling random peculiarities in the data [Domingos, 2012], [Hawkins, 2004]. Therefore, it is important to study how many requests should be chosen to train the system so that the performance results are maximized and, at the same time, the resource and time consumption are minimized. This critical issue has not received much attention in intrusion detection. Except for [Bolzoni and Etalle, 2008], that presents experiments with variations on the number of requests, usually the whole dataset is used by default to carry out the experiments in intrusion detection. Considering that statistical techniques require a certain number of requests in order to obtain meaningful results, this study turns specially interesting in this area.

The idea behind the adequate number of training requests is satisfying the principle of parsimony: including the necessary training requests, but no more. This criteria also leads to a better detection.

Studying the influence of the number of training requests on the performance of the system would allow to 1) determine how many training requests should be gathered in the data acquisition process to achieve certain results on the detection system and, 2) to estimate the time and resources necessary for the training phase (also for retraining in case it is necessary).

2.6 Conclusions

After analyzing the mentioned papers in the literature, a group of conclusions and open questions can be drawn:

- Regarding feature extraction, n-grams is a commonly used strategy to build features in intrusion detection. However, the combination of manually and automatically extracted features has been scarcely explored in the field.

- Feature selection improves both the performance and computational cost of intrusion detection systems. The *GeFS* measure for feature selection has been successfully applied to network traffic. However its performance on web traffic is an open question.

- Generally, statistical-based learning systems need to evaluate all detection models in order to establish the anomaly score of the request, that is, to decide about its normality/abnormality. Algorithms reducing resource and time consumption are necessary.

- Markov chains are effective in distinguishing attacks and normal traffic. Thus, they can be successfully applied to intrusion detection.
- Machine learning has been extensively used in intrusion detection. Decision trees have been proven to be effective in intrusion detection. Although these precedents make them promising to be applied in other scenarios, they have not been much used in web traffic analysis.
- Little attention has been paid to the study of how the number of requests used in the training phase affects the performance of the system. The knowledge acquired with such study allows to know, for example, the amount of requests necessary to achieve a certain detection result. Getting this type of information is useful in the data acquisition process, as well as in the estimation of time and resources necessary for training and retraining the systems.

The remaining of this thesis gives answers to these open questions.

Chapter 3

Data acquisition for web intrusion detection

"Scientia potentia est. - Information is power."

— Francis Bacon

Adequate datasets are of vital importance for training and testing WAFs. This chapter studies the characteristics that a dataset should present to be considered appropriate for this purpose. Unfortunately, finding datasets that meet these characteristics is not an easy task. The study of the most used datasets in intrusion detection reveals that most of them present a number of drawbacks to be applied to the evaluation of WAFs. In order to solve this situation, this thesis provides a new dataset called *CSIC* that satisfies the conditions established for the proper evaluation of WAFs. The dataset is being used by the scientific community. The characteristics of the dataset, as well as the process followed to create it are explained in this chapter. Additionally, the current applications of the dataset in the scientific community are presented. Lastly, conclusions are shown.

3.1 Problems in data acquisition for intrusion detection

Counting with appropriate datasets to train and test WAFs is critical. The quality of these datasets directly influences the evaluation of these systems. However, in the web intrusion detection community there is a scarcity of standard and common datasets to evaluate these systems. In relation to this issue, several authors claimed that "the most significant challenge that an evaluation faces is the lack of appropriate public datasets for assessing anomaly detection systems" [Sommer and Paxson, 2010], [Tavallaee et al., 2010]. Due to this scarcity, many researchers opted to create their own datasets, many of which are of private use. The problem of this situation is that it hinders the comparison between different systems. Furthermore, many of the existing datasets present a series of disadvantages that difficult their use in web detection systems. This makes noticeable that the necessity of counting with labeled and adequate datasets to train, test and compare WAFs is not covered in web intrusion detection. Fulfilling this gap is the motivation of this chapter.

3.1.1 Requirements for adequate datasets

In order to adequately configure and evaluate web intrusion detection systems it is necessary that the dataset used satisfies a series of requirements:

- It is convenient that it is publicly accessible. This allows other researchers use it and compare their systems.
- It should contain HTTP traffic, since this is the type of traffic WAFs analyze.
- The dataset needs to be labeled. Otherwise, it is not possible to evaluate the performance of WAFs.
- It should contain at least two classes: normal and attack. Anomalies could also be included. Anomalous traffic is further explained in Sec. 3.4.
- The dataset should contain a variety of modern attacks and their variations. According to Symantec, web attacks have evolved significantly

in the last years [Wood, 2014]. Including modern attacks allows to check if the system is able to detect current web attacks.

- Additionally, it is desirable that the traffic contains realistic values and it is not anonymized, with the purpose of not losing realism.

3.1.2 Evaluation of existing datasets

Unfortunately, difficulties associated to obtaining adequate datasets to evaluate WAFs are manifold. As mentioned before, there is a scarcity of labeled and appropriate datasets to evaluate web intrusion detection systems. Furthermore, many existing datasets face a number of troubles. Several existing datasets have been analyzed to check whether they satisfy the conditions presented in the previous section for considering a dataset as adequate. In regard to that, a number of problems have been found:

- **Dataset not labeled**. A common problem is that requests do not have a label indicating the class they belong to. The dataset is simply captured traffic. However, this is not enough for evaluating WAFs. Labels are necessary to measure how well the system is able to classify the instances.
- **Many datasets do not contain HTTP traffic**. For example, the *LBNL* dataset contains network traffic. This dataset contains traces with full header network information, but without payload [LBNL and ICSI, 2005]. The payload contains the information belonging to the application layer.
- **Many datasets are not publicly available or they might be difficult to obtain**. When the datasets are private, they are not usable by the scientific community. This does not make possible the comparison of systems between each other. In some cases, the reason for the obscurity is the privacy of data, what impedes sharing the traffic.

  In other cases, the datasets are partially available, i.e., they are only accessible for selected researchers. This is the case of the *UNB ISCX* intrusion detection evaluation dataset [Shiravi et al., 2012]. It is based on the concept of profiles, which contain an abstract representation of events and behaviors seen on the network. There are two types of profiles: the first one describes an attack scenario and the second one extracts mathematical distributions or behaviors of applications, protocols or low level network entities. Agents are used to generate HTTP traffic, among other protocols, from the profiles created. This dataset is only available for selected researchers and it is necessary to apply for it. The dataset takes several weeks to be obtained and in our case, it has not been possible to get access to it yet.

The *ECML/PKDD* dataset, that was generated for the European Conference on Machine Learning and Principles/ Practice of Knowledge Discovery in Databases (ECML/PKDD) Challenge in 2007, is also partially available [Raïssi et al., 2007]. In this case, it has been possible to obtain it. This dataset is labeled and contains exclusively HTTP traffic.

- **The dataset is not updated**. Since new web attacks are constantly appearing, it is important that the dataset is up to date. It implies containing modern attacks in order to adequately test the effectiveness of WAFs in contemporary environments.

  The *DARPA* dataset [Lippmann et al., 2000a], [Lippmann et al., 2000b] was presented in 1998 and 1999 by MIT. It contains network traffic, including HTTP traffic. This dataset is one of the most used ones for evaluating intrusion detection systems. However, the *DARPA* dataset has been criticized by the intrusion detection system community [McHugh, 2000], [Brown et al., 2009]. One of the reasons is that it is out of date and it does not include many of the modern attacks, making it not adequate for evaluating current WAFs. In fact, some researchers ([Estévez-Tapiador et al., 2004], [Hosseinkhani et al., 2011]) have used this dataset in conjunction with others, or with their own created attacks, to overcome this disadvantage.

- **Traffic is anonymized**. Privacy concerns are frequently a cause for anonymizing data. The preprocessing process that implies the anonymization of the dataset can lead to the loss of realism and can also negatively affect the quality of detection results. The previously mentioned *ECML/PKDD* dataset is an example of anonymized traffic.

Except for the attack part, all parts of its requests are anonymized. Hence, there are not two requests addressing to the same web application. This characteristic makes it unusable for a number of systems, for example [Hosseinkhani et al., 2011].

This problem also affects to the *LBNL* dataset. The traffic of this dataset has been anonymized to remove information that could identify an individual (Internet Protocol) IP.

- **Not balanced amount of normal traffic and attacks**. This problem is notable, for instance, in traffic gathered at warfare competitions. An example are datasets created in DEFCON Capture the Flag (CTF) [The Shmoo Group, 2011]. Since it was generated during the competition, this traffic mainly consists of intrusive traffic [Kruegel et al., 2005a]. In scenarios different from adversarial environments, this characteristic might make traffic unrealistic, since the proportion of attacks might result disproportionate in relation to the amount of normal traffic. In relation to this topic, and trying to overcome some of the current inconvenients, Sangaster et al. studied how to generate useful datasets by collecting traffic from warfare competitions [Sangster et al., 2009].

The cited problems have been obtained from the analysis of the datasets most used in intrusion detection. Table 3.1 shows, for each of the analyzed datasets, whether they satisfy or not the conditions shown in Sec. 3.1.1 for being considered an adequate dataset. The table shows that none of them satisfies all requirements.

Table 3.1 – Analysis of existing datasets satisfying requirements for an adequate dataset for evaluating WAFs.

Dataset	Public access	HTTP traffic	Labeled	Two classes	Modern attacks	Non anonymized
UNB ISC	✗	✓	✓	✓	✓	✓
ECML/PKDD	✗	✓	✓	✓	✓	✗
LBNL	✓	✗	✓	✓	✗	✗
DEFCON	✓	✓	✓	✗	✓	✓
DARPA 98/99	✓	✓	✓	✓	✗	✓
Captured traffic	✓	✓	✗	✗	✓	✓

3.2 Solution: generating a new dataset

Given the problems associated to the existing datasets, a new dataset was created in this dissertation to overcome the exposed drawbacks.

For the generation of the new dataset, three alternatives for obtaining web traffic were studied:

- Publishing a testbed web application on the Internet.
- Surfing the testbed web application in a controlled environment.
- Generating traffic artificially.

Next, these possibilities are analyzed in more detail:

- **Publishing the application on the Internet** allows to obtain realistic and reliable traffic. In this case, normal traffic is gathered but, unfortunately, attacks are also collected, since in the Internet there are hackers and malicious programs that try to hack web applications.

  The drawback of this alternative is that collected traffic is not labeled, unless it is done under controlled conditions (this is the next option). As mentioned, merely collecting web traffic is not enough for adequate datasets. It needs to be labeled. Manually labeling the huge amount of data necessary to evaluate a WAF is not a feasible option. In fact, it would be like manually solving the intrusion detection problem. For this reason, this alternative was discarded.

- Another possibility is **surfing the application in a controlled environment**. In this alternative, users have the role of either normal user or attacker. The differentiation of roles makes possible to label the requests of the dataset. The class label is in correspondence to the role of the user generating the request. This alternative offers more control of the traffic targeting the application and, then, of the content and realism of the dataset. The drawback of this option is that, as thousands of requests are needed, collecting users to generate the traffic might not be easy or cheap.

- **Artificially generating traffic**. A main advantage of this option is that it ensures the traffic to be correctly labeled. Although the traffic is not real, it allows to create the dataset exactly as desired. This gives flexibility to decide what type of traffic (normal or attack) will be included in the dataset, as well as which types of attacks. Because of the mentioned reasons and, given the drawbacks of the previous alternatives, this option was considered the most suitable one for our purposes.

This option was successfully applied to create the *CSIC* dataset. The characteristics and the generation process of this dataset are explained in the next sections.

3.3 Characteristics of the *CSIC* dataset

The *CSIC* dataset was created in our department in 2010. As contribution of this work, it was designed with the aim to overcome the described drawbacks of existing datasets. A public dataset, usable by the whole scientific community, allows the comparison of different detection systems.

In total, the *CSIC* dataset contains 36 000 normal requests and more than 25 000 anomalous requests. The requests are labeled either as normal or anomalous. Regarding the generation of attacks, both static and dynamic attacks were generated, including modern web attacks such as SQL injection, buffer overflow, information gathering, CRLF injection, cross-site scripting, server side include and parameter tampering.

Table 3.2 shows that the *CSIC* dataset satisfies all requirements defined for adequately evaluating WAFs. This table includes the *CSIC* dataset in the previously presented Table 3.1 in order to compare this dataset with the previously analyzed ones. The table shows that the *CSIC* dataset is the only one that satisfies the desired requirements.

The *CSIC* dataset is publicly available at *http://www.isi.csic.es/dataset*. There, three files can be found: one file for training and two files for testing. The training file contains 20 MB of only normal traffic. Regarding the testing datasets, one of them contains 20 MB of normal traffic and the other one 15.7 MB of anomalous data.

Table 3.2 – Analysis of existing datasets and the *CSIC* dataset satisfying requirements for an adequate dataset for evaluating WAFs.

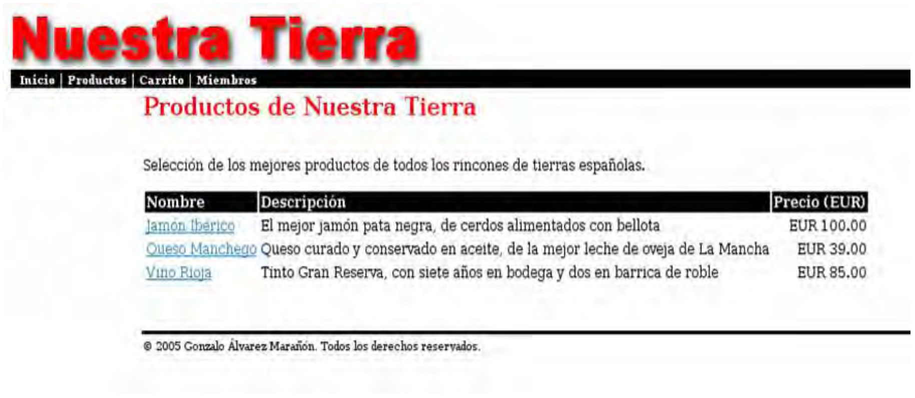
Dataset	Public access	HTTP traffic	Labeled	Two classes	Modern attacks	Not anonymized
UNB ISC	✗	✓	✓	✓	✓	✓
ECML/PKDD	✗	✓	✓	✓	✓	✗
LBNL	✓	✗	✓	✓	✗	✗
DEFCON	✓	✓	✓	✗	✓	✓
DARPA 98/99	✓	✓	✓	✓	✗	✓
Captured traffic	✓	✓	✗	✗	✓	✓
CSIC	✓	✓	✓	✓	✓	✓

3.4 Generation process

Since in the generation process of the dataset it was necessary to generate attacks, targeting web applications published in the Internet was not a feasible option. Hence, an ad hoc environment was built for our purposes. A web application was specially created for this goal. Also, a web server and a WAF to protect it were deployed in a virtualized environment. Virtualized environments have several advantages, such as resource saving, data privacy preservation, security and flexibility [Sahoo et al., 2010], [Li et al., 2015]. The WAFs presented in this thesis are explained in Chapters 4 and 5.

Although the target web application was not published in the Internet, it has the same structure and functionalities as real applications, in order to make it the most realistic possible. It consists of an e-commerce web application, developed with JSP and running under Apache Tomcat.

For the generation of the dataset, it was considered that targeting a single web application would be enough for our purposes. The designed application is composed of several web pages that allow users to do actions such as buying items with a shopping cart or registering by providing their personal data. Figures 3.1, 3.2 and 3.3 show screenshots of different pages belonging to the web application. In particular, Fig. 3.1 presents a list of products that can be bought in the e-commerce application, called “Nuestra Tierra”. Figure 3.2 shows the shopping cart when a client bough some products. And Fig. 3.3


Nuestra Tierra

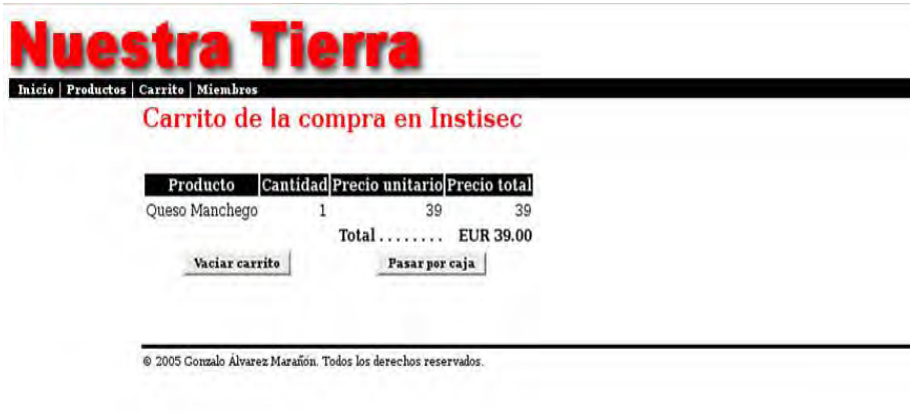
Productos de Nuestra Tierra

Selección de los mejores productos de todos los rincones de tierras españolas.

Nombre	Descripción	Precio (EUR)
Jamón Ibérico	El mejor jamón pata negra, de cerdos alimentados con bellota	EUR 100.00
Queso Manchego	Queso curado y conservado en aceite, de la mejor leche de oveja de La Mancha	EUR 39.00
Vino Rioja	Tinto Gran Reserva, con siete años en bodega y dos en barrica de roble	EUR 85.00

© 2005 Gonzalo Álvarez Marañón. Todos los derechos reservados.

Figure 3.1 – Screenshot of the e-commerce application showing available products.


Nuestra Tierra

Carrito de la compra en Instisec

Producto	Cantidad	Precio unitario	Precio total
Queso Manchego	1	39	39
		Total........	EUR 39.00

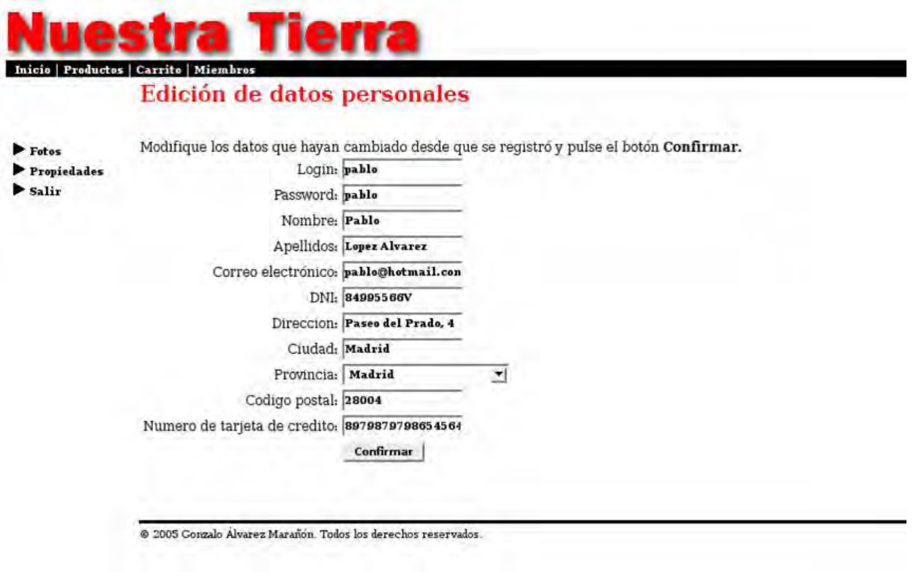
© 2005 Gonzalo Álvarez Marañón. Todos los derechos reservados.

Figure 3.2 – Screenshot of the e-commerce application showing an example of the shopping cart when some products are bought.

contains the registration page, where users introduce their personal data in the fields.

As usually happens with modern web applications, some pages of the e-commerce application admit arguments. Examples of parameters of this website are: a product name that a user wants to buy, the amount of it, a user's address in the registration process or his/her telephone number.

The generation of the traffic is made with the help of dictionaries. Dictionaries are data files that contain real data that can be used to fulfill the values of the website arguments. For example, in the case of our web shopping, dictionaries contain values corresponding to product names, user names, user addresses, telephone numbers, etc. This data is used to fill in the values of



Edición de datos personales

Modifique los datos que hayan cambiado desde que se registró y pulse el botón **Confirmar.**

Login:	pablo
Password:	pablo
Nombre:	Pablo
Apellidos:	Lopez Alvarez
Correo electrónico:	pablo@hotmail.con
DNI:	84995566V
Direccion:	Paseo del Prado, 4
Ciudad:	Madrid
Provincia:	Madrid
Codigo postal:	28004
Numero de tarjeta de credito:	8979879798654564

© 2005 Gonzalo Álvarez Marañón. Todos los derechos reservados.

Figure 3.3 – Aspect of the web application for the registration process.

the arguments in the traffic generation process. In particular, there are two dictionaries per argument in the web application: one with normal values, and another one with anomalous values. All data collected for dictionaries was extracted from real databases, with the aim of making the collected data as much realistic as possible.

Once the dictionaries are built, the generation of both normal and anomalous traffic can take place. In order to do that, all publicly available pages of the web application are visited (similarly to what a spider would do). In case the visited web page has arguments or cookies, their values are filled out with data taken from the dictionaries (normal or anomalous, depending on the type of traffic to be generated). The data values are chosen randomly.

Contrarily to some papers in the literature, like [Kruegel et al., 2005b], that use background traffic as normal traffic, the generation process of this dataset allows to make sure that normal traffic does not contain any attack. This is important for the training of anomaly-based WAFs, since if attacks are learned in this approach, they might not be detected.

The dataset contains individual HTTP requests. One of the advantages of the dataset is that it is not necessary to take care of packet fragmentation, what facilitates its use. Additionally, the generation process of the dataset does not need to deal with logs. This is beneficial since it can be a tedious task. In fact, some tools, such as Logstash, were born to help with the problems associated to handling logs.

In addition to attacks, the *CSIC* dataset contains anomalous traffic. Both categories are labeled as anomalous. Anomalous traffic makes reference to requests that are not normal but do not have attack intention. For example, a telephone number that contains letters. These requests could be generated, for instance, as consequence of user typographic errors. Anomalies could be used twofold: 1) when they are used in the training phase, they simulate noise in the traffic. Then it is possible to test the system under those circumstances. 2) When anomalies are included in the test stage, they allow to exhaustively test anomaly-based systems, checking how well they reject everything that is different from the established normal behavior.

Note that the dataset is designed to satisfy the requirements mentioned in Sec. 3.1.1 for adequately evaluate WAFs. Since it is publicly available, it is usable by researchers to evaluate and compare their systems. The dataset is labeled and it contains normal and anomalous HTTP traffic, including a variety of modern web attacks. It addresses a realistic web application and the values of the arguments are taken from real databases. Additionally, the traffic is not anonymized.

This dataset is used to experimentally train and test the different WAFs proposed next in this thesis.

3.5 Applications of the *CSIC* dataset

As mentioned, the motivation of making the *CSIC* dataset public is that the scientific community can use it. And indeed, the dataset is being used by other researchers.

Kozik et al. used the *CSIC* dataset to evaluate the behavior of their security algorithms. These algorithms are designed to detect injection attacks, in particular, SQLi and XSS [Kozik et al., 2014a], [Kozik et al., 2014b].

We have certainty that researchers at Alzahra University (Iran) of are also using the dataset for anomaly detection on web server logs. They are even doing research on what features extract from it, besides the ones presented in Sec. 4.5 and Sec. 5.4.2 of this dissertation.

Furthermore, a version of this dataset in comma-separated values (CSV) format has been done by Scully at Aberystwyth University [Scully, 2015]. This format facilitates the use of the dataset with tools such as Weka [Hall et al., 2009], that provides a wide range of machine learning algorithms.

3.6 Conclusions

The main conclusions of this chapter are the following:

- Obtaining public labeled and suitable datasets for training and testing WAFs is not an easy task.

- Most of the existing datasets present drawbacks that render them inappropriate for evaluating WAFs.

- The proposed *CSIC* dataset satisfies the requirements desired for adequately evaluating WAFs. Among other characteristics, it is public, labeled, and its data is not anonymized, what makes it ideal for this purpose. The creation of the dataset helps to solve an important problem that was opened in the field: the scarce of labeled and adequate datasets to evaluate WAFs. Furthermore, it makes possible to compare different web detection systems.

- The *CSIC* dataset is currently being used by the scientific community.

Chapter 4

Stochastic techniques for web intrusion detection

"Mathematics is the alphabet in which God has written the universe."

— Galileo Galilei

This chapter presents two WAFs based on stochastic techniques. To perform detection, one of the WAFs applies statistical algorithms and the other one Markov chains. An explanation about the main concepts of Markov chains is included in this chapter. Additionally, the general architecture of the systems, as well as their design, are presented. The design of the systems comprises two phases. First, the preprocessing step takes place, where features are extracted. The processing phase consists of both training the system and later testing it. The detection algorithms make use of two models. One of them makes reference to the length of certain elements of the HTTP request and, the other one, to properties about their characters. These models, together with other information, are stored in the Normal Behavior Description (NBD) file during the training phase. The structure of this file is described. During the detection process, the information stored in the NBD file is retrieved to decide about the normality/abnormality of the incoming requests. The systems are experimentally evaluated using the *CSIC* dataset. After establishing the experimental setup, the results of the experiments are shown and discussed. A study about the influence of the number of training requests is also included in this chapter. Finally, the conclusions drawn are presented.

4.1 Introduction

Stochastic techniques have been extensively used in intrusion detection. Several references about it can be found in Sec. 2.4.1. In this chapter, two WAFs based on stochastic techniques are proposed. The systems operate at the application layer and follow an anomaly-based approach. The objective is constructing systems that reach high-speed and high detection, while keeping a simple design. One of the proposed systems applies statistical techniques for detection, and the other one Markov chains. These techniques are chosen because they are the stochastic algorithms most widely used in intrusion detection. Both proposed systems share a common structure, given that both are based on stochastic techniques. However, each of them presents a different implementation, according to the particularities of the algorithm used.

4.2 Markov chains concepts

In order to better understand the implementation of the Markovian system presented in this chapter, the main concepts of this technique are explained in this section.

A Markov chain is defined by a set of N states $\Gamma = \{S_1, S_2, \dots, S_N\}$ and, by the pair of probability matrices, Π and A [Feller, 1968], [Ramana, 2007]. The matrices express the temporal evolution of the system from a statistical point of view. Concretely, $\Pi = \pi_i, \forall i \in [1, N]$, is a vector that indicates the probability of the i-th state being the first element of the temporal sequence of observations:

$$\pi_i = P(q_1 = S_i),$$

where q_t represents the current state of the model at time t and S_i the elements of the temporal sequence of observations. This vector has the following constraints:

$$\pi_i \geq 0, \forall i \in [1, N] : \sum_{i=1}^{N} \pi_i = 1.$$

Matrix A, $A = a_{ij}, \forall i, j \in [1, N]$, represents the transition probabilities between states. Given that the system is in a state i at some time t, the matrix gives the probability of reaching the state j at time $t + 1$. The matrix of transition probabilities can be estimated as follows:

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_i) = \frac{P(q_t = S_i \cap q_{t+1} = S_j)}{P(q_t = S_i)}.$$

Matrix A has the following two constraints:

$$a_{ij} \geq 0, \forall i, j \in [1, N] : \sum_{j} a_{ij} = 1, \forall i \in [1, N].$$

The probability $P_j^{(t)}$ of state j at time t is given recursively by

$$P_j^{(1)} = \pi_j, P_j^{(t)} = \sum_{i=1}^{N} P_i^{(t-1)} a_{ij}, t > 1.$$

Markov models have two stages: the first one is learning, where Γ, Π and A are learned. The second stage is evaluation. During this stage it is checked whether an observed sequence is recognized by the learned Markov chain. The details about how they are implemented are explained in next sections of this chapter.

Next, the general architecture of both stochastic systems is explained.

4.3 General architecture

The systems presented analyze HTTP traffic in order to protect a target web application. They act as reverse proxies. A reverse proxy is a type of proxy server that directs client requests to the appropriate back-end server. The architecture of the designed WAFs is shown in Fig. 4.1. The objective of a WAF is to accurately distinguish whether the received requests are suspicious or not. For that, it receives as input a collection of HTTP requests $r_1, r_2, \ldots, r_n$. This input comes from a client that targets the protected web server. After processing an input request, the system outputs a single bit a_i for the corresponding r_i. This output indicates whether the request has been classified as normal or anomalous. The proposed WAFs are programmed in Java. The communications between the client and the WAF, and between the WAF and the server, are implemented via sockets. The systems can be located between the client and the server, or they could be included as a module of the web server to be protected.

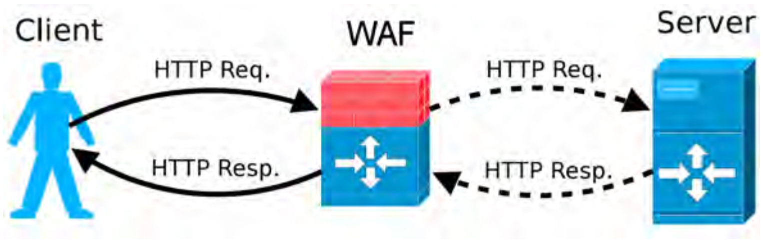


Figure 4.1 – Web Application Firewall Architecture.

Detection takes place at the application layer. Analyzing the payload of the requests, and not only the headers of the TCP packet, makes possible to be more efficient and detect more types of web attacks. According to WASC, the analysis developed at the application layer is capable of detecting web attacks that cannot be detected when working on lower OSI layers [WASC, 2006].

The proposed WAFs can act online or offline. Online systems are those connected to the network, handling traffic on demand. Contrarily, offline systems are disconnected from the network, typically analyzing requests from a dataset.

The incoming requests are processed individually. This means that those requests analyzed previously do not influence the decision about the classification of the present request. Thus, these WAFs are focused on detecting attacks that involve a single request. Some systems in the state of the art are specialized on detecting a specific type of attack, like [Valeur et al., 2005] in detecting SQL attacks or [Kirda et al., 2009] and [Wurzinger et al., 2009] in detecting XSS. In contrast to this type of systems, the WAFs presented in this work are designed to detect multiple types of web attacks, such as SQLi, XSS, buffer overflow, server side include or parameter tampering, among others.

Additionally, the whole HTTP request is analyzed. This characteristic is different from other works, that analyze only some parts of the request, like [Kruegel and Vigna, 2003] and [Song et al., 2009]. This decision was taken based on the fact that attacks can be included in any part of the request. For instance, XSS attacks could be embedded into the *User-Agent* header, or cookies could be used as part of web attacks (cookie hijacking) [Riley et al., 2010]. Then, in order to detect these attacks, the whole request needs to be analyzed.

4.4 Design

The proposed systems follow an **anomaly-based approach**. In this approach the normal behavior of the web application is characterized and those requests that deviate more than a threshold from the specified behavior are deemed as anomalous. This approach was further explained in Sec. 2.2.1.3.

The proposed systems are designed following two stages: preprocessing and processing. These stages are explained next. This design is shared by both systems since they apply stochastic algorithms.

- **Preprocessing**. When an individual request is received, it is firstly decoded. Then, **feature extraction** takes place: those features considered relevant for detecting attacks are extracted from the request. It is done by means of expert knowledge about web attacks. For extracting features, the request is parsed into its components: HTTP method, resource, version of the HTTP protocol, headers and arguments. The resource is composed of directories and a file. In this thesis, arguments include both arguments of POST requests and parameters in the query of GET requests. Headers and arguments are further split into their name and value. Figure 2.1 on Sec. 2.1.2 shows an example of an HTTP request and its components.

  With the aim of detecting both static and dynamic attacks, the following features are extracted from the different request components:

  - Method name.
  - Path and name of the resource targeted.
  - For each header: header name and stochastic models of the header value.
  - For each argument: argument name and stochastic models of the argument value.

  Argument and header values are considered more critical from the web security perspective since attacks are more frequently embedded in these components. Thus, stochastic models are applied to capture the properties of these components. The models are implemented differently depending on the algorithm used, statistical or Markovian. Detection models are explained in Sec. 4.5.

  The features extracted are stored in the Normal Behavior Description file. Details of this file are given in Sec. 4.6.

  The proposed approach is applied at token level [Krueger et al., 2010]. Features are extracted from each component individually, not from the whole request. This is useful due that each component might have a different nature. Furthermore, each argument or header might have different properties. For example, the character distribution of a zip code is different from the one of a user's name argument. According to Kruegel et al., "systems that focus on web-based attacks show that by taking advantage of the specificity of a particular application domain is possible to achieve better detection results" [Kruegel et al., 2005b]. Since it gives more precision than if the features were extracted for the whole request or for a group of components, it is the option chosen to design WAFs in this thesis.

- **Processing**. The processing stage of a request is different depending on the phase of the system: training or test. During the **training phase**, the features extracted are used to learn the normal behavior of the web application. It implies filling the NBD file and training the corresponding detection models.

  In the **test phase**, incoming traffic is classified. In order to do that, it is checked whether the features of the incoming request match the normal behavior stored in the NBD file. If any feature does not match the normal behavior, the whole request is deemed as anomalous. Otherwise, it is classified as normal.

After the test phase, the request is encoded and sent to the web server. It is important to clarify that the action performed in this step depends on whether the system acts as an IDS or IPS. The presented systems can work on both modes. In the IDS mode, these systems send all requests to the server and raise an alarm when the request has been classified as anomalous. In the IPS mode, the systems only forward normal requests. That is, anomalous ones are blocked.

An scheme of the presented design can be seen in Fig. 4.2.

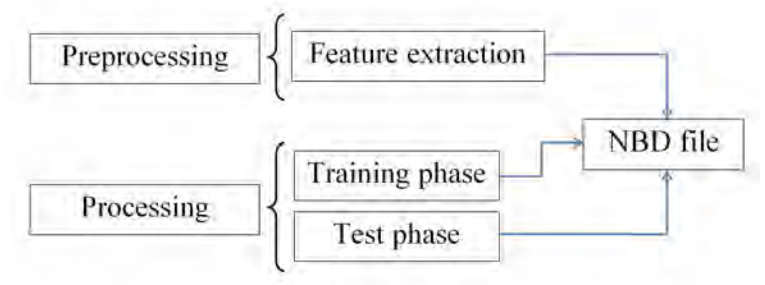


Figure 4.2 – Design structure of stochastic-based WAFs.

Next, detection models are presented.

4.5 Detection models

Detection models are applied to characterize the behavior of argument and header values. Recall that models are applied to every argument or header present in the request and that the models are applied individually to each of them, what gives more precision in the detection.

These detection models are based on normality intervals that define a range within the argument or header values are considered normal. Values falling outside the intervals are considered anomalous. The limits of the intervals are learned during the training phase. If only normal traffic is used during the training phase, the limits of the intervals establish thresholds between what is considered normal or anomalous.

Both stochastic systems use two detection models. One of the models analyzes the length of the argument/header value and the other, calculates stochastic properties of its characters. The detection models are implemented differently depending on the algorithm used. Next, details about statistical and Markovian detection models are given.

4.5.1 Statistical detection models

As mentioned, the statistical approach uses two detection models: 1) length and, 2) character distribution of the argument/header values. These models are explained next.

- **Length model**. Length is a useful criterion to detect attacks due to the fact that, on the one hand, values of normal requests do not usually contain many bytes and, on the other hand, many web attacks use a considerable large amount of input characters (such as code injection, XSS and buffer overflow). This model captures the length of the argument/header value. The limits of the interval are established as the minimum and the maximum length of the argument/header values seen in the training. In previous experiments, we set these limits to different values, as it will be explained later.

- **Character distribution model**. Several intervals are defined for this model. By using our expert knowledge about web attacks, we observed that not all characters have the same importance in web attacks. Special characters are particularly relevant for the detection of numerous web attacks. Furthermore, Sriraghavan and Lucchese proved that considering letters, digits and non-alphanumeric characters is more efficient than considering the relative frequency of the 256 ASCII characters [Sriraghavan and Lucchese, 2008]. Therefore, instead of considering the 256 ASCII characters individually, the statistical WAF models the characters of the argument/header values into three groups:

- Letters.
- Digits.
- Non-alphanumeric characters.

Besides the improvement in the detection, an advantage of considering these three groups is that it accelerates the training and checking processes of the models.

Percentages of the character distribution, according to these three groups, were used as features to build the model. It is composed of three intervals:

- Percentage of letters.
- Percentage of digits.
- Percentage of non-alphanumeric characters included in a set of non-alphanumeric characters allowed for the corresponding argument/header. This set is formed during the training of the system. That is, non-alphanumeric characters corresponding to the argument/header value are included into the set.

Similarly to the length model, the limits of the intervals are fixed with the minimum and maximum percentages found on the training argument/header values. Next, an example of how the lower limit of the letter interval is calculated. At the beginning its value is set to the percentage of letters corresponding to the first header/argument analyzed. If the percentage of the following values analyzed is lower than the established limit, the limit of the interval is replaced by the new percentage.

As mentioned, different limits of the intervals were considered in a previous version of the model. In that case, instead of the minimum and the maximum values, the mean (μ) and the standard deviation (σ) were used to calculate the limits. The width of the interval was regulated by a sensibility parameter $s \in [0.2, 4]$:

$$[\mu - \sigma \cdot s, \mu + \sigma \cdot s].$$

From the experiments, it was observed that if the sensibility parameter was not high enough, some normal values fell outside the interval in the test phase. Particularly, it happened in those cases that were not close enough to the mean. Since the criteria of establishing the limits of the interval with the minimum and maximum values makes the system more efficient, this option was preferred.

It is remarkable that the approach proposed has less models than most of the existing statistical WAFs. For example, Kruegel et al. uses eight detection models [Kruegel et al., 2005b]. Sriraghavan and Lucchese use five models [Sriraghavan and Lucchese, 2008]. And the paper from Criscione and Zanero employs six models for detecting SQL attacks [Criscione and Zanero, 2009]. Provided that it does not harm the detection, using a low amount of simple models allows to save time and resources, at the same time that simplifies the design of the system.

4.5.2 Markovian detection models

The system based on Markov chains also uses two detection models for the characterization of the application normal behavior: the first model considers the length of the argument/header value and the second its structure. Note that structure is different from character distribution. The structure does not only consider the percentage of each type of character, but also the order of the characters. It captures which character is followed by another one.

- **Length model.** This model uses the length of an argument or header value in order to detect anomalies.

  Similarly to the statistical case, this model is based on intervals that define the range of normality for the argument/header values. However, it also presents some differences. One difference is that the Markovian model makes the assumption that the length of normal argument/header values follows a Gaussian distribution. This distribution is used to calculate the thresholds of the length interval:

  – Instead of considering the minimum length of the training values as in the statistical case, the Markovian model sets the lower limit of the interval to zero. It was done since from the security point of view, it is not necessary to restrict the lower limit. That is, a string is not dangerous because the fact of being short.

– The upper limit is established as follows: given a probability, the corresponding percentile of the Gaussian distribution is taken. For example, if the probability is fixed to 0.9, the upper limit is the value below which 90% of the observations fall.

  For a normal distribution, the percentile is calculated with this equation:

$$percentile = \mu + z_{score} \cdot \sigma,$$

  where μ is the mean, σ the standard deviation and z_{score}, also known as standard score [Bluman, 2007]), is a measure to know how far a data point is above or below the population mean, expressed in standard deviations.

  Then, the limits of the length interval are set as follows:

$$[0, percentile].$$

  The probability p associated to the calculation of the corresponding percentile is a parameter of the model. Its values can be configured by the IT operator to test the behavior of the system under those conditions (see Sec. 4.9).

- **Structure model.** Markov chains are ideal for modeling the structure of tokens, since they can capture the order in which characters are distributed. Recall that Markov chains are defined by Γ, Π and A. These concepts were explained in Sec. 4.2.

  The knowledge about the different states reached by the system, Γ, is obtained though the observation of the system outcomes $\Theta = O_i$, that are considered as possible states of the system. In our Markovian system, the states correspond to the different types of characters: l (letter), d (digit) and non-alphanumeric characters (such as *,(,),-,', etc). Note that again, the characters are grouped, what makes possible to reduce the number of states of the Markov model. However, in this case each non-alphanumeric character constitutes an state itself. Contrarily to the statistical model, there is not a generic state that covers all non-alphanumeric characters.

The probability of matrix A and vector Π are estimated during the training phase. The idea is that this model captures the normal distribution of the characters forming the argument/header values. Each argument and each header has its own Markov chain, what allows to capture the particular structure of the corresponding values. During the test phase, it is checked whether a given sequence of observations is recognized by the previously estimated model or not.

For the learning phase, the simplest generalization of the Markov model was considered. In it, every observed state depends on the previous one, and only on the previous one. Then, the matrix of probabilities of transitions can be estimated by:

$$a_{ij} = \frac{P(q_t = O_i \cap q_{t+1} = O_j)}{P(q_t = O_i)}.$$

The two probabilistic terms in the previous expression can be calculated by counting occurrences of the states in the observed values. Similarly, Π can be estimated by counting the occurrences of states in the first character of the observed training values. For example, in the case of an argument with value "Markov", the only state is l. The state l of vector Π takes value 1, corresponding to "M". The transition from state l to l in matrix A takes value 5.

All stochastic models are stored in the NBD file, that is explained next.

4.6 Normal behavior description file

The normal behavior learnt about the web application under study is stored in a normal behavior description (NBD) file. It is implemented as an XML file. XML files have the advantages of being universal and easily readable.

The structure of the NBD file is related to the components of HTTP requests. It has the following nodes:

- **Methods**. This node includes a whitelist of the allowed HTTP methods.

- **Headers**. It contains a list of the admitted HTTP headers and a characterization of their values. Header values are described in a *rule* node that contains the two detection models explained in Sec. 4.5. The models are implemented differently depending on the system.
- **Directories**. This node has a tree-like hierarchy, in close correspondence to the web application's directory structure:
  - Directories of resources belonging to the web application are represented by their *name* in a *directory* node, allowing to nest directories. There is a node for each subdirectory in the resource.
  - The web page of the resources (file) is characterized by its *name* in a *file* node, placed within the corresponding *directory* node.
- **Arguments**. Each argument of the web page is defined by its name and value in an *argument* node, holding from the corresponding *file* node. The values of the arguments are characterized by the statistical or Markovian models, depending on the system. The structure of the NBD file for these models is explained below.

4.6.1 Particularities of the NBD file for the statistical system

In this section the particularities of the structure of the NBD file corresponding to the statistical detection models are explained. Statistical models are present in the NBD file for each argument, and each header, in order to characterize the statistical properties of their values. The intervals corresponding to the statistical length and character distribution models are represented within the *stats* node of the NBD file. The *stats* node corresponding to the length model contains these elements:

- minLength.
- maxLength.

The *stats* node corresponding to the character distribution model contains the following elements:

- minLetter.
- maxLetter.
- minDigit.
- maxDigit.
- minSpecial.
- maxSpecial.
- special.

All elements starting with *min* represent the lower limit of the corresponding interval. They are calculated as the minimum value seen in the training argument/header values. For that, the length of the first element is established as the minimum at the beginning. Then, if during the training phase the feature of the incoming value is lower than the current limit in the NBD file, the lower limit is updated. Analogously, elements starting with *max* elements refer to the upper limit. It is calculated similarly but with the maximum value. The element called *special* represents the set of non-alphanumeric characters allowed for the corresponding argument/header value. Recall that the statistical values are extracted for each argument/header independently, since their values could have totally different properties.

Figure 4.3 shows an example of the structure of the NBD file configured for the target e-commerce web application presented in Sec. 3.4. In the example, the file "add.jsp" in the "public" directory represents a page in charge of adding a new product to the shopping cart. This page has two arguments: the name of the product that the customer wishes to buy and the amount of items selected. As an example, argument "amount" is shown. The allowed values of this argument have a maximum length of 3, then its maximum value is 999. These values are fully formed by digits, i.e., no letters or special characters are permitted for this argument.

```
<configuration>
<methods>
  <method>GET</method>
  <method>POST</method>
</methods>
<headers>
  <rule name="Accept-Charset">
        <stats minLength="5" maxLength="20"
               minLetter="4" maxLetter="90"
               minDigit="2"  maxDigit="60"
               minSpecial="2"  maxSpecial="25"
               special="-"/>
  </rule>
</headers>
<directories>
  <directory name="shop">
    <file name="index.jsp"/>
    <directory name="public">
      <file name="add.jsp">
        <argument name="amount">
          <stats minLength="1" maxLength="3"
                 minLetter="0" maxLetter="0"
                 minDigit="100"  maxDigit="100"
                 minSpecial="0"  maxSpecial="0"
                 special=""/>
        </argument>
        ...
```

Figure 4.3 – NBD file example for the statistical algorithm.

4.6.2 Particularities of the NBD file for the Markovian system

In this section, the structure of the NBD file related to Markovian models is presented. The length model is represented by the *length* node in the file. This node is defined by the lower and upper limits of the length interval. The *Markov model* node contains information about the structure of the values represented by a Markov chain. This node describes the states Γ, vector Π, and matrix A of the Markovian model. This model is updated dynamically as long as the training requests are received. Note that only the states found in the training values will be captured by the model. For example, a zip code will only have the state d. In this case, states l and non-alphanumeric characters are not needed to define the allowed values of this argument. This implies that the dimension of matrix A is not fixed. It is minimized as much as possible for each token. This fact helps to reduce the memory consumption to store the model and the time needed to check if a string matches it.

Figure 4.4 shows an example of the NBD file configured for "Nuestra Tierra" web application. The example is presented for the same web page "add.jsp". Regarding the length model, the lower limit is fixed to zero. The upper limit is calculated as the percentile of the Gaussian distribution. The "amount" parameter contains only digits. As there is only one state, vector Π and matrix A have only one element (100% of the characters are digits). In the case of the "product name" argument, the possible states are: letter and space (non-alphanumeric character). Then, the dimension of Π is 1x2 and A is 2x2. As vector Π reflects, the name of a product can only start with a letter. Matrix A contains the following knowledge: in 90% of the cases the character following a letter is another letter, and in 10% of the cases it is a space. After a space always a letter is following.

4.7 Detection process

Detection process takes place during the test phase. In this step the system is already trained and it is ready for operation. The goal of this phase is to correctly classify the incoming requests as normal or anomalous.

An schema of the detection process is depicted in Fig. 4.5. It consists of a succession of steps that are in charge of testing whether the request satisfies the normal behavior learned or not. To do that, the first step is extracting features from the incoming request. Then, the detection process takes place. In it, it is successively checked whether different components of the request match their corresponding section in the NBD file. This file has been previously filled with those values considered normal for the target web application.

```
<configuration>
<methods>
  <method>GET</method>
  <method>POST</method>
</methods>
<headers>
<rule name="Accept-Language">
<length lowerLimit="0" upperLimit="2"/>
<markovModel A="1.0" Pi="1.0" states="l"/>
</rule>
</headers>

<directories>
  <directory name="shop">
    <file name="index.jsp"/>
    <directory name="public">
      <file name="add.jsp">
        <argument name="amount">
<length lowerLimit="0" upperLimit="2.29"/>
<markovModel A="1.0" Pi="1.0" states="d"/>
    </argument>
        <argument name="productName">
<length lowerLimit="0" upperLimit="14.56"/>
<markovModel A="0.9 0.1 1.0 0.0" Pi="1.0 0.0" states="l, space"/>
    </argument>

...

      </file>
    </directory>
  </directory>
</directories>
```

Figure 4.4 – NBD file example for the Markov chain algorithm.

The detection process is composed of the following steps:

1. The process starts checking the HTTP method. It is done by checking whether the method belongs to the whitelist contained in the NBD file or not. In case the method is not present, the whole request is rejected and the detection process ends. Otherwise the detection process continues.

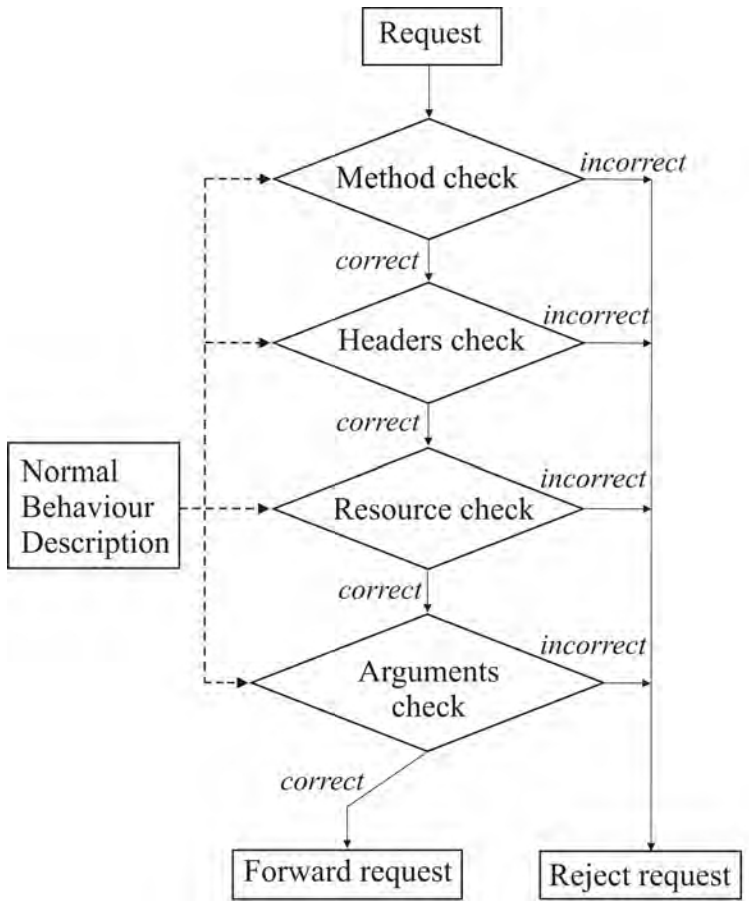


Figure 4.5 – Detection process flow.

2. Headers are checked. For each header, the process is the following:

    (a) Firstly, it is checked whether the header name appears in the NBD file.

    (b) If so, its value is checked. To consider a header as valid, its value should match both detection models. It is done differently depending on the system (statistical or Markovian). The detection process of these models is later explained in this section.

3. Regarding resources, it is checked if both the path (directories) and the file accessed are in correspondence with the tree structure of the NBD nodes.

4. If there is any argument, each of them is checked in a way similar to headers' checking:

(a) Firstly, it is checked whether the argument is allowed for the targeted resource or not. To be allowed, the argument name should be included in the list of arguments of the corresponding resource.

(b) If so, the value of the argument is tested. The argument should satisfy both detection models. Checking these models is done following the same process carried out with headers. It is later explained in this section.

Since there is no previous information about which header or argument could be more prone to be an attack, they are checked in the same order as they appear in the request. As can be observed, only when it is confirmed that every part of the request is normal, the request is tagged as normal. As soon as any part of the request does not match the normal criteria established, the whole request is classified as anomalous. It is remarkable that this process has a design where not all modules need to be evaluated. Some of them can be skipped under certain circumstances. In particular, these circumstances happen when any part of the request does not pass any of the defense lines. In that case, the request is directly rejected. This fact makes possible to accelerate the decision process. Even more, it is new in comparison to most statistical-based works in the literature, like [Kruegel et al., 2005b], [Cheng, 2009], [Sriraghavan and Lucchese, 2008], [Criscione and Zanero, 2009]. These works usually need to evaluate every model before deciding about the normality/abnormality of the request.

Since this process checks different parts of the request, it allows the detection of both static and dynamic attacks.

4.7.1 Detection process in statistical models

The process to check whether a specific argument/header value of the incoming request satisfies the statistical models follows this structure:

- First, the length model is checked. To satisfy this model, the length of the argument/header value to be tested should fall inside the corresponding length interval in the NBD file. Since this interval specifies the limits of normal values, they are classified as normal. Otherwise, they are considered anomalous.

- Second, the character distribution model is analyzed. To be considered normal, the value to be checked should be included inside the three intervals defined in the NBD file, that correspond to the letter, digit and non-alphanumeric character distribution. Additionally, its special characters should be contained in the corresponding special set. If any of the thresholds are bypassed (i.e., the values fall outside the intervals) or the value contains non-allowed special characters, the request is labeled as anomalous.

4.7.2 Detection process in Markovian models

The process to check whether the Markovian model is satisfied is the following:

- First, the length model is evaluated by checking if the length of the incoming argument/header value is inside the corresponding interval of allowed lengths.

- Second, to evaluate whether a given observed value is recognized by the previously estimated Markov chain, the following approach is adopted:

  Given a Markov chain $\lambda = (\Gamma, A, \Pi)$ and a sequence of observed symbols $O = O_1, O_2, \ldots, O_T$, the sequence is recognized by the Markov chain if the probability of the sequence being generated by the Markov chain ($P[O|\lambda]$) exceeds an established threshold. Recall that the state corresponding to the characters is checked, not the character itself.

  Based on [Estévez-Tapiador et al., 2004], the representation on a logarithmic scale of the Maximum A-posteriori Probability (MAP) is used to evaluate if the observed sequence has been generated by the Markov chain:

$$LogMAP(O, \lambda) = log(\pi_{O_1}) + \sum_{t=1}^{T-1} log(a_{O_t O_{t+1}}). \tag{4.1}$$

  Since it is logarithmic, in this measure no probability can be zero. A usual technique to avoid zero values is performing a previous *smoothing* of the Markov chain. A simple way of smoothing the model is setting those probabilities lower than a given threshold, to a fixed value called ϵ.

As long as the observations fit the model, the "LogMAP" function (4.1) does not have abrupt changes of slope. However, if there is any unexpected symbol, there will be an abrupt change of slope. Detecting these changes would mean detecting anomalies. To detect them, an approximation of the derivative of the "LogMAP" function can be used:

$$D_W(t) = |LogMAP(t) - \frac{1}{W}\sum_{i=1}^{W} LogMAP(t-i)|, \tag{4.2}$$

where W is the window size, that can take values $W = 1, 2, 3, \ldots$. The last term in (4.2) is the mean of the last W outputs. This equation supplies an output for each symbol analyzed in the sequence. When the output exceeds a fixed threshold τ, the sequence of observed symbols is classified as anomalous. ϵ and τ are configurable parameters of the model.

4.8 Experimental setup

In this section the setup carried out for the experimental stage is explained.

Given that the *CSIC* dataset satisfies the requirements established in Sec. 3.1.1 to be considered appropriate for evaluating WAFs, it is used to conduct the experiments to evaluate the proposed systems.

Recall that the training dataset contains only normal traffic. Since the algorithms are anomaly-based, the models learn correctly the normal behavior of the web application when they are trained with only normal traffic. Although a certain amount of noise would be tolerable for the stochastic algorithms, in anomaly-based systems, attacks included in the training dataset could result undetected in the test phase. This is why they were not included in the training dataset for these experiments.

The systems perform a two-class detection: the requests are classified as either normal or abnormal.

With the aim of studying how the variation of the number of training requests influences the detection results of the system, a number of experiments are run. Let the number of experiments be $M = 15$. These experiments are performed with an increasing number of training requests. The number of requests used in each experiment is $tr_i = 2^i - 1, \forall i \in [1, M]$. That is, the number of training requests ranges from 1 to 32 767.

The number of possible combinations to choose tr_i normal requests among the total number of normal requests in the *CSIC* dataset (36 000) is very high. Since trying all possibilities is infeasible, a few of them (H) have been randomly chosen. In particular, $H = 10$. There are different options for choosing requests, like stratified, quota or random [Ellison et al., 2009]. Within them, the random method has been selected due that there is no previous information about how the system is going to be attacked. More about experiments is argued in Sec. 4.10.

The procedure for each experiment is the following: the system is trained with tr_i requests randomly chosen. Then the system is tested with a fixed subset of testing requests, that contains te normal requests and te anomalous requests, with $te = 1000$.

After the test phase the performance of the detection algorithm is evaluated, as it is explained next.

4.9 Evaluation measures

Evaluating IDSs is a hard task. Sommer and Paxson even state that "evaluation turns out to be more difficult than building the detector itself" [Sommer and Paxson, 2010]. They point out the difficulty in finding the right data to test the systems and complications in interpreting the results as the main obstacles for evaluating IDSs. Difficulties in gathering data were addressed in Sec. 3.1.

Despite these difficulties, several measures can be used to evaluate IDSs performance. Before introducing these measures, a few concepts are introduced first.

Table 4.1 shows the contingency matrix of a detection system. If a real attack is detected, the system obtains a True Positive (TP). Otherwise it is a

Table 4.1 – Contingency matrix for detection systems.

		Detection classification	
		Negative	Positive
Real label	Negative	TN	FP
	Positive	FN	TP

False Negative (FN). If a normal event is signaled as normal by the detector, it is a True Negative (TN). It is a False Positive (FP) if it is classified as an attack.

For evaluating IDSs, detection rate and false positive rate are measures generally used in the field. Detection rate (also known as recall) measures the number of alarms that are indeed attacks. It is calculated as follows:

$$DR = \frac{TP}{TP + FN}.$$

The false positive rate indicates the number of alarms that are not really attacks, i.e., false alarms:

$$FPR = \frac{FP}{FP + TN}.$$

DR y FPR take values between 0 and 100 and are expressed as a percentage.

The objective of IDSs is maximizing the detection rate while minimizing the false positive rate. As can be seen, these objectives are antagonistic, then, they are not easy to optimize simultaneously. The challenge of IDSs is to reach a tradeoff between DR and FPR.

DR and FPR are usually plotted in Receiver Operating Characteristic (ROC) curves [Provost et al., 1998]. According to Bradley, a ROC curve provides a way to visually represent the trade-off between false positive and detection rates by varying a parameter [Bradley, 1997]. Since plotting DR vs. FPR represents one point in the graph when all parameters are fixed, the variation of the parameter is used to draw the whole curve. For example, it is common to use the detection threshold as a parameter of the ROC curve.

In our case, the parameter chosen for the graph is the number of requests used in the training phase. This parameter is used for both the statistical and

Markovian algorithms. As mentioned, the value of the parameter is increased exponentially. Experiments are performed as follows. For every number of requests used for the training (tr_i), the system is tested subsequently. This allows to observe the influence of the parameter on the detection results. This process is repeated H times.

In the case of Markov chains, additionally, the parameters of the Markov chain are used in the ROC curve to test their behavior when they take different values:

- Parameter p corresponds to the probability used to calculate the percentile of the length model. This model was explained in Sec. 4.5. The values tested for p are $p = 0.9$, $p = 0.95$ and $p = 0.99$.
- The parameter ϵ is used to smooth the Markov model. As it was mentioned in Sec. 4.7, it is assigned to values in the transition probabilities whose values in the trained model are 0 or lower than a threshold. When ϵ takes low values, the system is more sensible to deviations with respect to the normal behavior. Following the study of Estevez-Tapiador et al. about the values of this parameter, the values tested are: $\epsilon = 10^{-15}$, $\epsilon = 10^{-10}$, $\epsilon = 10^{-8}$ and $\epsilon = 10^{-4}$ [Estévez-Tapiador et al., 2004].
- Parameter τ is the threshold used to decide whether a token value is normal or anomalous. If the parameter takes low values, the detection is more prone to false positives. On the contrary, if it is very high, it is not possible to detect attacks with a low level of abnormality. Then, different values have been tried to conclude which one leads to better detection results: $\tau = 30, 40, 50, 60, 70, 80, 90, 100, 200, 300$.

4.10 Results

This section presents the detection results obtained in the experimental stage by both systems. Results are presented according to the different evaluation measures previously shown.

As mentioned in the previous sections, given a number of training requests, the experiments are performed H times. This reduces the possible effect of

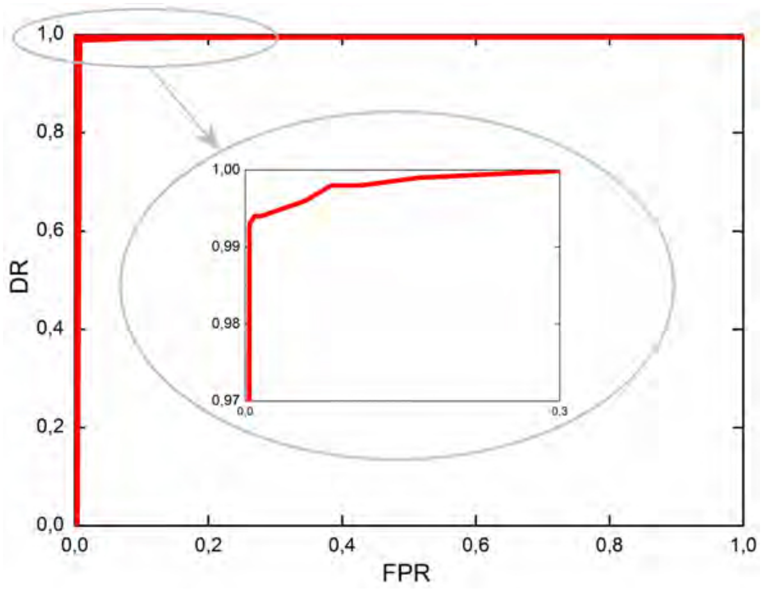


Figure 4.6 – ROC curve of the statistical algorithm. The parameter used is the number of training requests.

randomness and shows better the actual behavior of the WAFs. The results of the system are expressed as the mean and standard deviation of the H trials.

The results of the statistical-based system are shown in Fig. 4.6. It plots the ROC curve (DR vs. FPR) of the algorithm. The parameter chosen is the number of requests used in the training phase. A zoom of the upper left corner is shown for more clarity.

The results obtained for Markov chains can be seen in Fig. 4.7. Additionally, for Markov chains, a study of the behavior of the systems for various values of the parameters has been done. Figure 4.8 shows the ROC curve corresponding to different values of the parameter p, Fig. 4.9 for parameter ϵ and Fig. 4.10 for parameter τ. These results are discussed in Sec. 4.11. Moreover, the values of the parameters have been studied according to the incremental number of requests used to train the system. Figure 4.11 and Fig. 4.12 show the DR and FPR corresponding to parameter p. Figure. 4.13 and Fig. 4.14 present results for parameter ϵ. And Fig. 4.15 and Fig. 4.16 for parameter τ. These figures show a small figure inside with a zoom of the graph to facilitate the visualization of the results.

A discussion of these results is given in the next section.

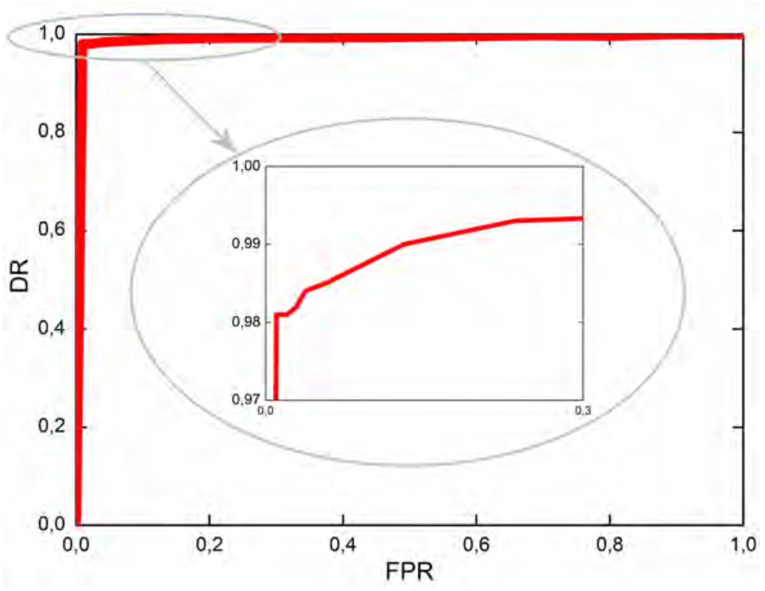


Figure 4.7 – ROC curve of the Markovian algorithm. The parameter used is the number of training requests.

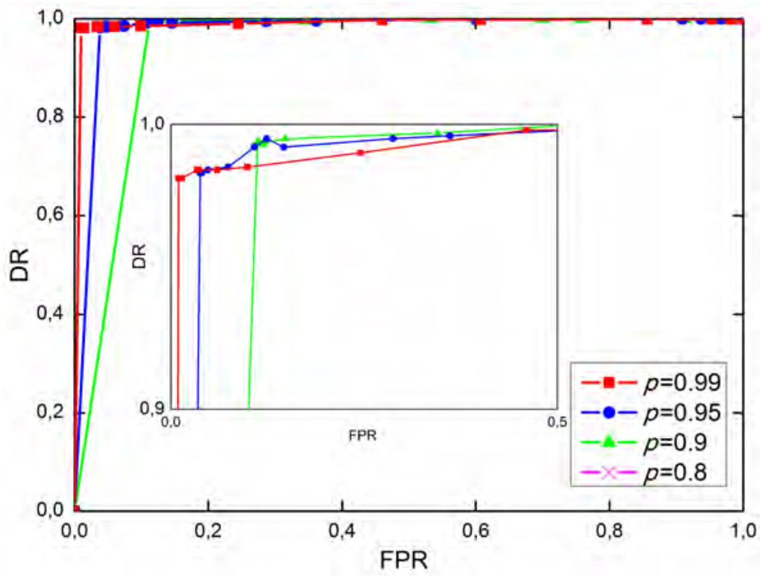


Figure 4.8 – ROC curve of the Markovian algorithm. The parameter used is p.

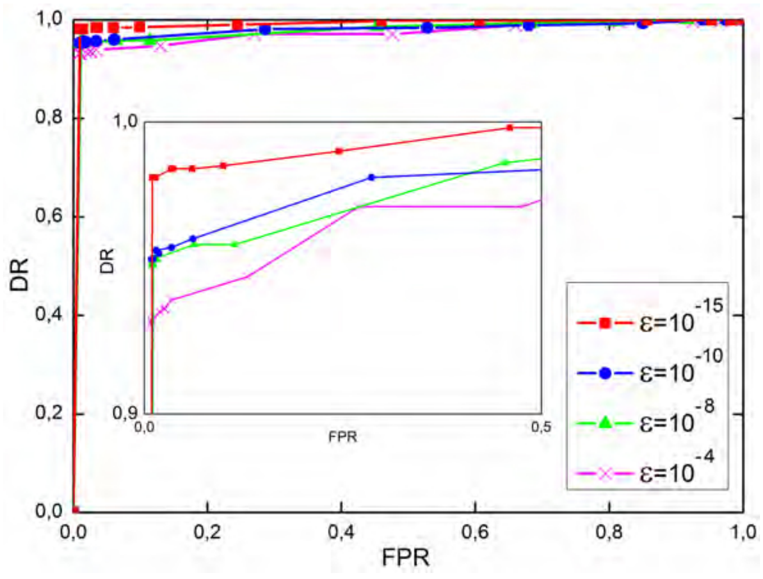


Figure 4.9 – ROC curve of the Markovian algorithm. The parameter used is ϵ.

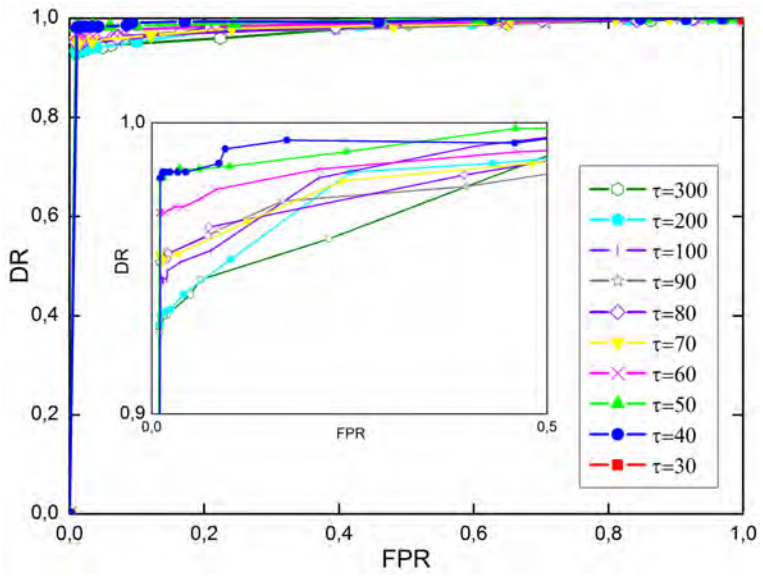


Figure 4.10 – ROC curve of the Markovian algorithm. The parameter used is τ.

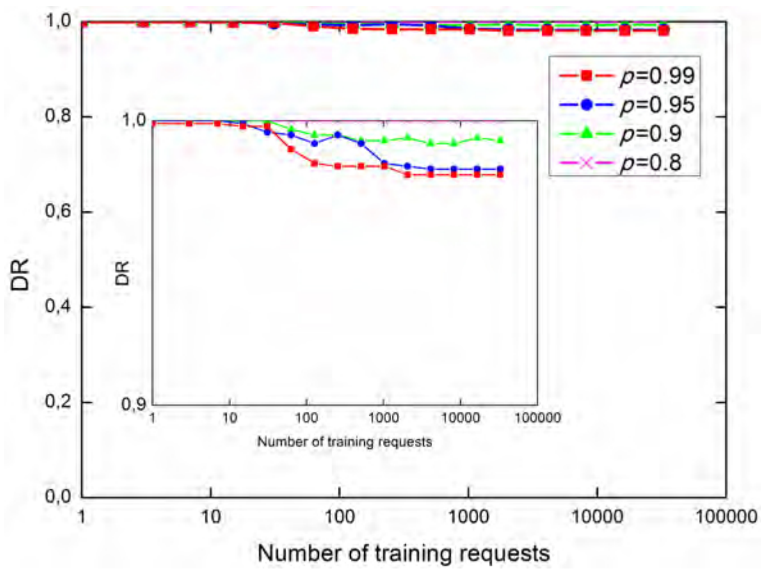


Figure 4.11 – DR according to the increase of the number of training requests for the Markovian algorithm. The parameter used is p.

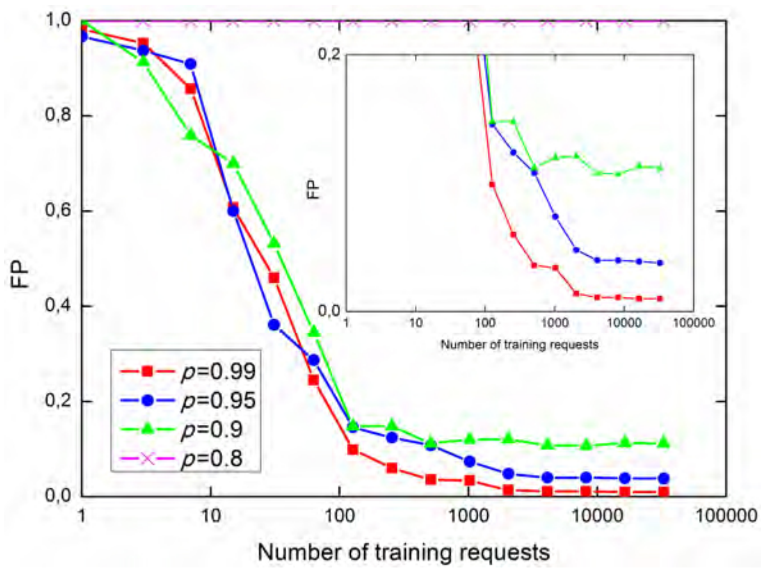


Figure 4.12 – FPR according to the increase of the number of training requests for the Markovian algorithm. The parameter used is p.

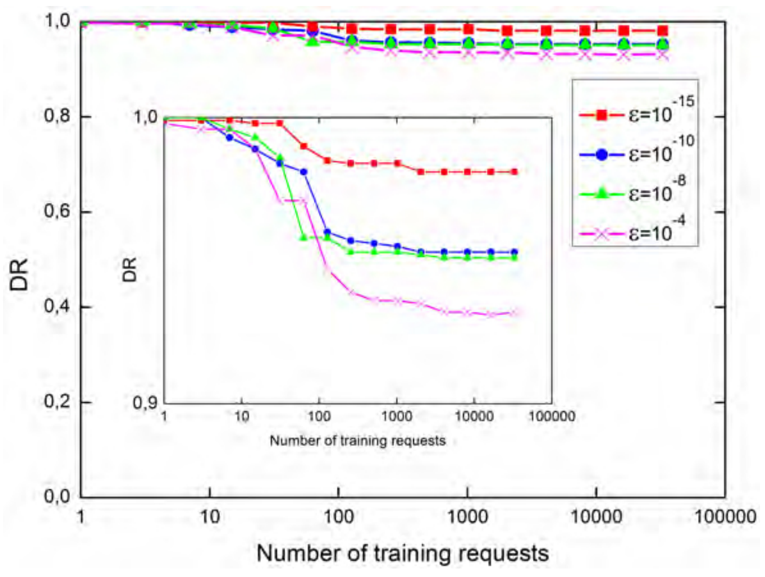


Figure 4.13 – DR according to the increase of the number of training requests for the Markovian algorithm. The parameter used is ϵ.

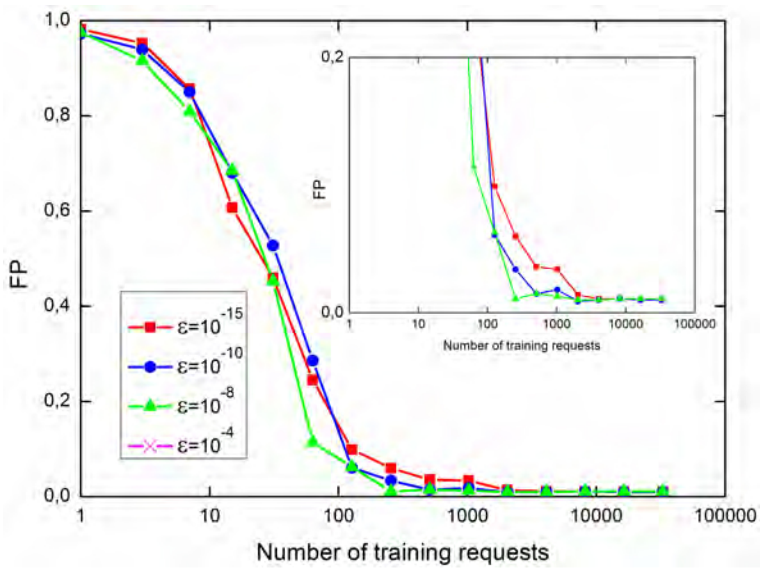


Figure 4.14 – FPR according to the increase of the number of training requests for the Markovian algorithm. The parameter used is ϵ.

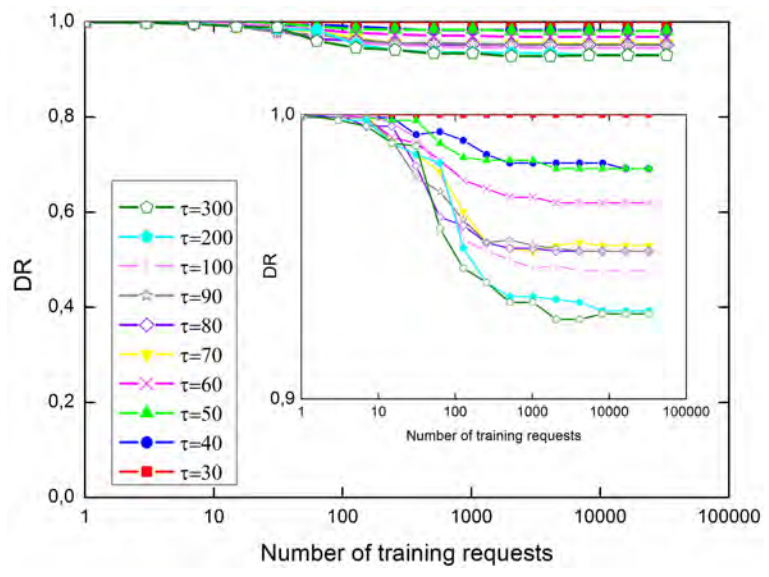


Figure 4.15 – DR according to the increase of the number of training requests for the Markovian algorithm. The parameter used is τ.

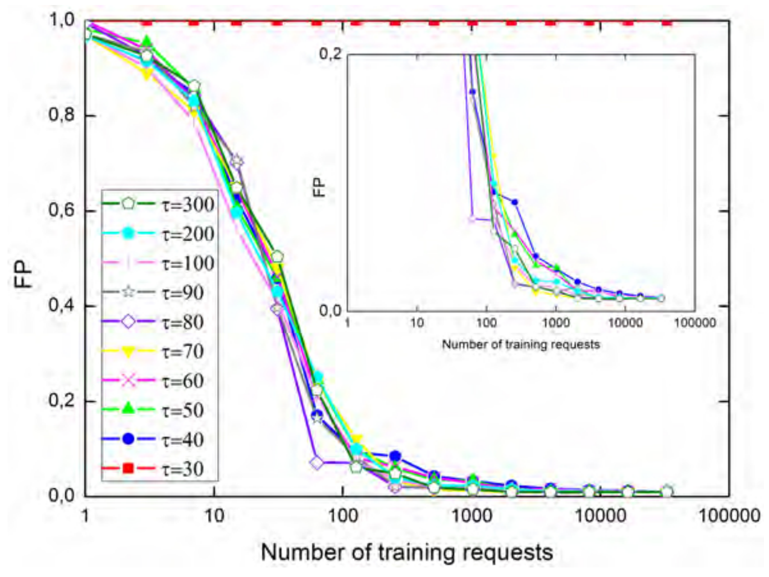


Figure 4.16 – FPR according to the increase of the number of training requests for the Markovian algorithm. The parameter used is τ.

The influence of the number of training requests in the results can be better appreciated in the following figures and tables. Figure 4.17 presents the results for statistical techniques. It reflects the influence of the training requests on the detection results (DR and FPR). Training requests are represented in a logarithmic scale. These results are detailed in Table 4.2 that shows, for each iteration, the corresponding number of training requests, as well as the mean (σ) and standard deviation (μ) of the DR, FPR and processing time.

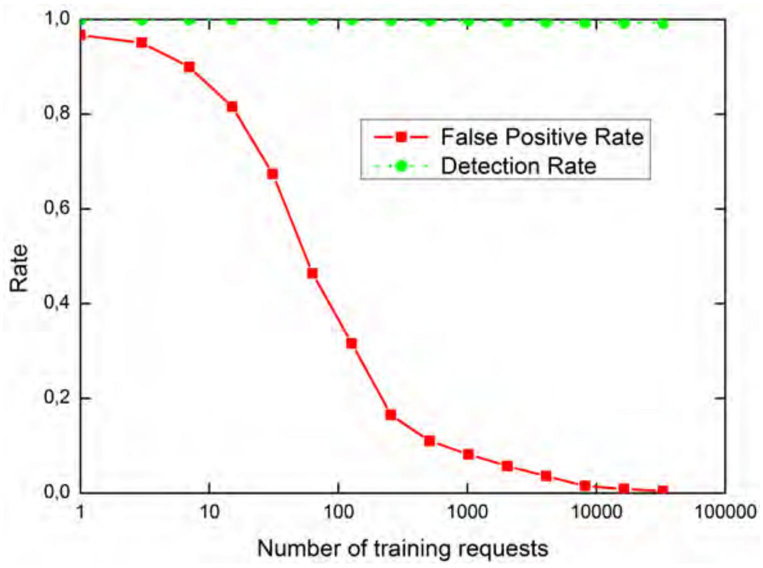


Figure 4.17 – Study of the influence of the number of requests on the detection results of statistical algorithm. DR and FPR are plotted vs. the number of training requests.

Regarding Markov chains, Fig. 4.18 shows the influence of the number of training requests in the results of the algorithm. The detection results presented above, as well as the processing time of the Markovian algorithm are summarized in Table 4.3. This table shows the mean and standard deviation of DR, FPR and processing time for each tr_i.

Table 4.2 – Detection results and processing time for the statistical algorithm.

Iteration	Num. Training Requests	DR (%)		FPR (%)		Processing Time (ms/request)	
		μ	σ	μ	σ	μ	σ
1	1	100	0	96.7	1.68	1751	2.81
2	3	100	0	95.1	2.34	393.66	2.54
3	7	100	0	90.0	2.27	179.95	2.39
4	15	100	0	81.6	3.41	86.17	2.34
5	31	100	0	67.4	4.01	43.8	1.87
6	63	100	0	46.4	4.32	19.58	1.62
7	127	100	0	31.6	3.57	11.2	1.53
8	255	99.9	0.05	16.5	1.60	5.19	1.27
9	511	99.8	0.05	11.0	1.69	2.77	1.05
10	1023	99.8	0.05	8.2	0.74	1.79	1.01
11	2047	99.6	0.14	5.7	0.83	1.1	0.88
12	4095	99.5	0.14	3.6	0.45	0.85	0.80
13	8191	99.4	0	1.5	0.12	0.68	0.61
14	16 383	99.4	0	0.9	0.05	0.63	0.34
15	32 767	99.3	0.05	0.4	0.08	0.59	0.29

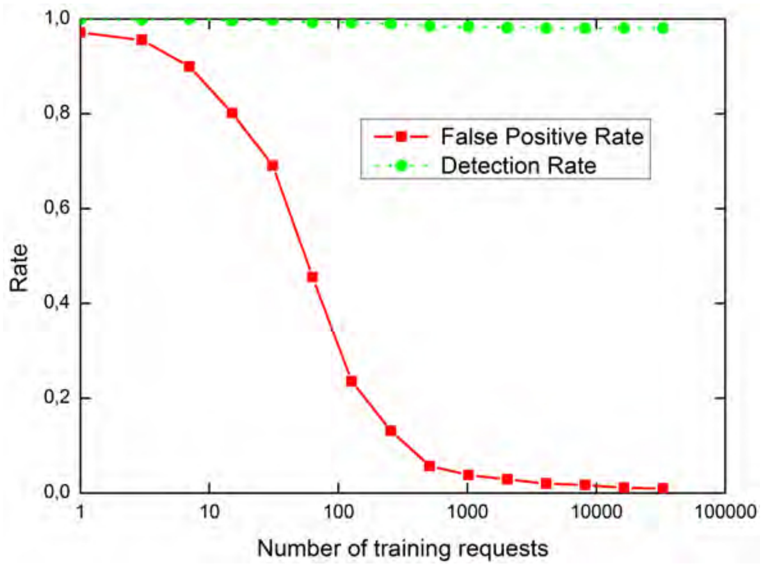


Figure 4.18 – Study of the influence of the number of requests on the detection results of Markovian algorithm. DR and FPR are plotted vs. the number of training requests.

Table 4.3 – Detection results and processing time for the Markov chains algorithm.

Iteration	Num. Training Requests	DR (%)		FPR (%)		Processing Time (ms/request)	
		μ	σ	μ	σ	μ	σ
1	1	100	0	97.2	0.70	2519	2.70
2	3	99.9	0.05	95.6	1.51	493	2.35
3	7	100	0	90.0	1.18	202.81	1.89
4	15	99.7	0.12	80.2	3.03	186.22	1.66
5	31	99.8	0.19	69.1	3.17	50.83	1.58
6	63	99.4	0.20	45.6	1.96	37.69	1.29
7	127	99.3	0.09	23.6	2.96	20.01	0.52
8	255	99	0.29	13.1	0.88	17.53	0.95
9	511	98.5	0.14	5.7	0.97	13.78	0.88
10	1023	98.4	0.05	3.8	0.59	43.45	0.69
11	2047	98.2	0.08	2.9	0.34	10.32	0.45
12	4095	98.1	0.05	2.0	0.22	10.16	0.29
13	8191	98.1	0.05	1.7	0.37	9.93	0.07
14	16 383	98.1	0	1.1	0	8.87	0.07
15	32 767	98.1	0	1.0	0.05	7.9	0.09

4.11 Discussion

Next, the results presented previously for the statistical and Markovian systems are analyzed.

4.11.1 Discussion of statistical results

Table 4.2 reveals that when the statistical WAF is trained with 16 383 requests, it is able to achieve a mean DR of 99.4% while the mean FPR is 0.9%. Results of Table 4.2 are represented graphically in Fig. 4.6. The corresponding standard deviations are 0% and 0.05% respectively. A property of the Gaussian distribution is that 95.4% of the values are 2σ away from the mean value. Assuming a Gaussian distribution in the example before, it means that for 95.4% of the values, the DR remains 99.4%, and FPR ranges from 0.8% to 1%. Analyzing the three columns corresponding to σ in the table, it can be seen that for FPR and time, the standard deviation decreases as long as the system is trained with a higher number of requests. In the case of the DR, σ takes values near zero.

Considering that the standard deviation takes values near zero, especially when the system is trained with a higher number of requests, we did not continue performing more experiments. We considered that taking H possibilities was representative enough. These results presented correspond to the mean of the $H = 10$ runs. It is expected that the standard deviation would be even smaller when a higher number of experiments is performed. Since σ takes small values, when DR and FPR results are given in this document without specifying σ, they refer to the mean value.

The mean processing time corresponding to the example above is 0.63 ms/request. All processing time measurements have been obtained with an Intel core i7 CPU at 2.40 GHz and 8GB RAM, SO Windows 8, 64 bits.

A possibility to obtain even lower processing time values would be to study the viability of parallelization in some parts of the process. Potential points to introduce parallelization in the design of the presented WAFs are the different steps of the detection process presented in Sec. 4.7 or the checking of the different headers/arguments. Some works also have been done in the direction of running Markov chains in parallel [Gopal and Casella, 2011], [Angelino, 2014]. However, a thorough study of this issue falls outside the scope of this thesis.

Representing the results graphically, the ROC curve of Fig. 4.17 shows that when a low amount of training requests is used, the algorithm rejects most requests. That is, DR is very high but false positive rate is also very high. As long as the system is trained with more requests, the trend of the FP is to decrease progressively. Three stages can be observed in the graph. At the beginning (from 1 to 255 training requests), the reduction of the FPR is very sharp. When the system starts being trained with more requests (until 8191 training requests), FPR decreases more slowly. In the last stage, when the system is trained with higher amounts of requests, the fall is very soft. Meanwhile, DR keeps almost invariable. DR has the highest value possible (100%) at the beginning and, later, it is very lightly reduced. These results reflect that when the algorithm does not have enough information, it cannot detect correctly. When more information is provided, the system can detect intrusions while raising scarce false alarms, until a point where the addition of more requests is not really providing new knowledge to the algorithm.

In comparison with other papers in the literature, the presented systems use a low amount of requests. For example, TokDoc [Krueger et al., 2010] uses a third of the requests of the *FIRST08* dataset for the training phase, that is, 484 040 requests. It was also trained with a third part of the *BLOG09* dataset, i.e., 393 980 requests. Our statistical algorithm is able to reach a mean DR of 99.4% and a mean FPR of 0.9%, while only using 16 383 requests. The advantages of needing a low number of requests were explained in Sec. 2.5.

All types of attacks included in the *CSIC* dataset have been successfully detected by the statistical algorithm.

4.11.2 Discussion of Markovian results

Table 4.3 shows that with 32 767 training requests the Markovian system reaches a mean of 98.1% DR and 1% FPR. The associated σ are 0% and 0.05%. These results can be seen graphically represented in Fig. 4.7. In Table 4.3 it can be observed that σ decreases when the system is trained with more requests and it takes values near zero. The mean processing time associated to these results is 7.9 ms/request.

Regarding the study of Markov chain parameters, Fig. 4.8 shows that the optimal value for parameter p is $p = 0.99$. The best value is chosen as the one that achieves the best balance between a high DR and a low FPR. Study of parameter p reveals that when its value is 0.8, the system cannot classify correctly. When $p = 0.8$, DR and FPR are both 1, regardless the number of requests used to train the system. This means that all requests analyzed by the system are tagged as attacks. Therefore, in the case of our Markovian system, this value is not meaningful to distinguish between normal and anomalous requests. Hence, this parameter value should not be used.

The study of parameter ϵ in Fig. 4.9 shows that the best results are achieved when $\epsilon = 10^{-15}$. In the case of τ, the best value is $\tau = 50$. The influence of this parameter can be seen in Fig. 4.10. Similarly to what happened with parameter p, when $\tau = 30$ the systems cannot distinguish between normal and abnormal values, then, this value should be discarded.

Analyzing results in Fig. 4.18, it can be seen that, similarly to the statistical system, the results pass through three different stages in the study of the influence of training requests. Firstly, the DR and FPR are both high when the algorithm is fed with few requests. As long as more requests are used in the training, the trend of the algorithm is to achieve better results, until the algorithm does not improve anymore, even when more traces are provided.

This behavior of the system is also reflected in the study of parameters. It is shown in Fig. 4.11 and Fig. 4.12 for parameter p, Fig. 4.13 and Fig. 4.14 for parameter ϵ and Fig. 4.15 and Fig. 4.16 for parameter τ. These figures show DR and FPR respectively, for different values of the parameters when the number of training requests increases. Again, the results of the Markovian algorithm improve when the number of training requests increases, until a point where results hardly increase.

Optimal values for parameters, $p = 0.99$, $\epsilon = 10^{-15}$ and $\tau = 50$, were used for the results shown in Table 4.3.

4.12 Conclusions

The main conclusions drawn from this chapter are:

- The proposed statistical and Markov chains algorithms are able to correctly distinguish attacks from normal traffic. The whole request is analyzed for intrusion detection, what allows detecting attacks in any part of an HTTP request.

- The presented algorithms achieve high detection rates. Statistical techniques reach a mean DR of 99.4% and a mean FPR of 0.9%, with 0% and 0.05% values for the standard deviation respectively. Markov chains achieve 98.1% DR and 1% FPR in average, with the same standard deviation values than in the previous case. Additionally, the algorithms are able to detect zero-day attacks. Moreover, all types of attacks in the *CSIC* dataset are detected.

- The algorithms proposed are high-speed. Using 16 383 training requests the statistical techniques reach a mean processing time of 0.63 ms/request, with 0.34 standard deviation. This technique can even reach up to 0.59 ms/request when trained with 32 767 requests. In the case of Markov chains, the processing time for 32 767 training requests is 7.9 ms/request, with a standard deviation of 0.09 ms/request.

- The algorithms are designed following the simplicity principle, while not negatively affecting the detection results. Only two simple models, applied to argument/header values, are used for detection. The rest of the detection is done by means of whitelists. Besides than effective, the detection process is designed to be efficient. It means that for each request only the necessary components are evaluated. When any of the normality conditions is not satisfied, the process does not need to continue any longer.

- The study regarding the influence of the number of training requests reveals that it is possible to achieve a mean FPR of 1% with a reduced number of requests in comparison to the amounts that have been typically used in intrusion detection so far. The experiments show that the statistical technique presented uses 16 383 requests to achieve a mean FPR of 0.9% and Markov chains use 32 767 to achieve a mean FPR of 1%.

  This study also confirms the expected trend in the behavior of the algorithms: they detect correctly only when enough training requests are received. However, after some point, increasing the number of requests does not incorporate new knowledge to the learned behavior.

- The *CSIC* dataset can be successfully used to evaluate stochastic-based WAFs. The fact of testing different systems with the same dataset allows their comparison.

Chapter 5

Machine learning techniques for web intrusion detection

"Intelligence is the ability to adapt to change."

— Stephen Hawking

This chapter proposes a WAF based on machine leaning techniques. In particular, it is based on decision trees. After analyzing the general architecture of the WAF, its design is presented. For this system, the preprocessing phase comprises feature extraction and feature selection. New feature extraction methods are proposed. For feature selection, the *GeFS* measure is used. Processing corresponds to the classification process, that is performed with the help of decision trees. The system is evaluated using two datasets: our *CSIC* dataset and the publicly available *ECML/PKDD* one. After fixing the experimental settings, the results obtained from the experiments are presented. Experiments show that the new combination methods improve the results of both expert knowledge and n-grams separately. Additionally, a study of the influence of the number of training requests is given. Results are analyzed from different points of view. A comparison of stochastic and ML systems is also included. Finally, the conclusions drawn are presented.

5.1 Introduction

The automation capabilities of machine learning have been applied to many computer areas, among them, to intrusion detection. Some examples of IDSs based on ML are presented in [Chen et al., 2005], [Mulay et al., 2010] and [Sangkatsanee et al., 2011b]. The previous chapter proposed WAFs that apply stochastic techniques for performing detection. This chapter also presents a WAF, but in this case, the system uses ML techniques. Like in stochastic systems, the ML-based WAF follows the anomaly approach. The objective is again designing high-speed and high detection systems that have a simple design. The design of the system includes a preprocessing and a processing stage. In the preprocessing stage, feature extraction and feature selection are applied. In particular, new feature extraction methods are proposed in this chapter. These methods combine expert knowledge and n-grams, and they prove to be more effective than these two techniques separately. Within the variety of ML algorithms, decision trees are chosen for the classification stage due to their successful application in intrusion detection. Four different decision tree algorithms are applied, namely C4.5, CART, Random Tree and Random Forest. Further details of the system and the experiments carried out are explained in this chapter.

5.2 General architecture

The architecture of the ML system has many similarities with the stochastic systems presented in the previous chapter. The system works at the application layer analyzing the whole payload of HTTP requests. After analyzing the incoming requests, the system outputs its classification decision about the normality or abnormality of the request. The system detects a variety of web attacks that involve a single request.

The ML-based WAF analyzes the payloads at both token and request levels. It makes the approach more complete from a security point of view, since more attacks can be detected. Including the request level is useful in cases where all tokens satisfy the detection requirements but the whole request does not. For example, when the length of all tokens are within the length limits but the whole request is not. It does not mean that the analysis at token level is not necessary. As it was argued before, it allows to detect many web attacks that otherwise would go undetected.

5.3 Design

The design of the system counts with two stages: preprocessing and processing. An scheme of this design is shown in Fig. 5.1.

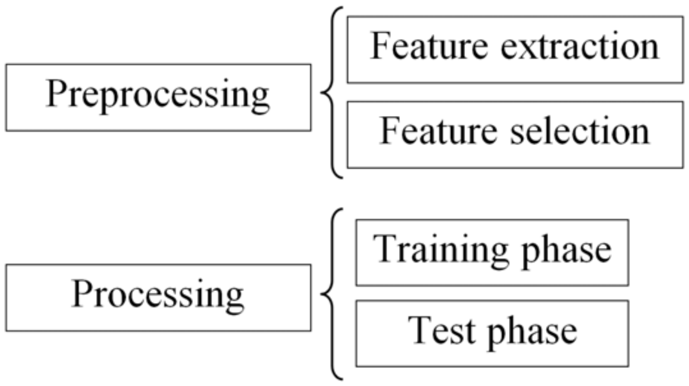


Figure 5.1 – Design structure of ML-based WAF.

- **Preprocessing**. This stage consists of **feature extraction and feature selection**. Notions of these two processes can be found in Sec. 2.3. Regarding feature extraction, in intrusion detection it is usually done by applying either manual or automatic techniques. This thesis proposes new feature extraction methods for ML by combining both techniques. The previously proposed stochastic systems use expert knowledge to extract features. Differently, the ML system combines expert knowledge features with n-gram ones. The objective is that the proposed methods improve the detection results of the separate techniques at the same time that they consume low resources. These feature extraction methods proposed are explained in Sec. 5.3.1.

In order to guarantee that this combination consumes low-resources, feature selection is applied to reduce the number of irrelevant and redundant features. From the feature extraction categories presented in Sec. 2.3.1, the filter model was chosen, given that it requires less computational resources. In particular, the *GeFS* measure for intrusion detection is applied. This measure has been successfully tested with network traffic before [Nguyen et al., 2010b], [Nguyen et al., 2011]. However, it has not been applied to web traffic so far. In this dissertation its behavior in HTTP traffic is studied. Details about this measure are given in Sec. 5.3.2.

- **Processing**. This step comprises the **classification** phase. ML include a set of algorithms, such as Bayesian networks, artificial neural networks, genetic algorithms, support vector machines or decision trees. From these algorithms, decision trees have been widely and successfully applied to intrusion detection. For this reason, they are chosen for the classification process of the ML system. This election is based on the fact that this family of algorithms is one of the most popular machine learning algorithms [Wu et al., 2007]. Additionally, decision trees have been proven to obtain experimentally successful results in intrusion detection. In fact, the winner of the famous DARPA intrusion detection contest [Lippmann et al., 2000a] was an algorithm based on decision trees [Pfahringer, 2000]. This characteristics make them promising to be applied to web attack detection. Decision trees have been rarely employed to classify web traffic. One of the goals of this thesis is studying their performance in the classification of this type of traffic.

  According to No-Free-Lunch Theorem, there are no universal classifiers for every kind of data [Wolpert, 1996], [Wolpert, 2001]. Since there is no standard classification algorithm for WAFs, four decision tree algorithms have been applied: C4.5, Classification And Regression Tree, Random Tree and Random Forest. They have a training and a test phase. These algorithms are further explained in Sec. 5.3.3.

5.3.1 Feature extraction

This thesis proposes new methods for feature extraction. These methods combine expert knowledge and n-gram features. A description of n-grams can be found in Sec. 2.3.1.1. Three combined extraction methods are proposed:

- ***Combine-select***. This alternative firstly mixes all features extracted by expert knowledge and n-grams. As the number of n-grams often leads to dimensionality problems, afterwards, feature selection is applied in order to reduce the number of features. In the seldom cases where the combination of automatic and manual features was applied in the literature (see Sec. 2.3.1), this alternative was used. More alternatives are presented next.
- ***Select-combine***. This alternative mixes expert knowledge features that have been previously selected. Differently to the previous option, feature selection is performed firstly and, the resulting features, are mixed subsequently.
- ***Select-n-gram-combine***. This alternative follows the idea of *select-combine*, however, it only takes the selected values of n-grams. Expert knowledge features are not selected because they come from knowledge of web attacks experts. Therefore, the subset derived from this alternative is composed, on the one hand, by expert knowledge features and, on the other hand, by the selected features from n-grams.

An scheme of these three alternatives can be seen in Fig. 5.2. Each of the alternatives generates different subsets of features, receiving the same name than the corresponding alternative. That is, the corresponding subsets are called *combine-select*, *select-combine* and *select-n-gram-combine*. These subsets are called “combination cases”. In contrast, *expert knowledge* and *n-gram* subsets are referred to as “basic cases”.

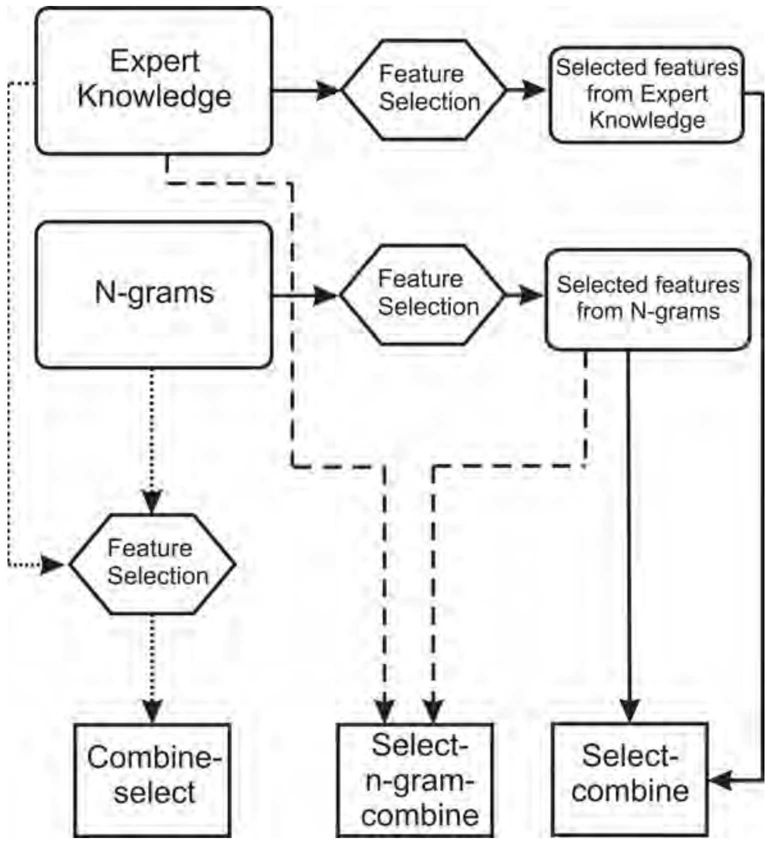


Figure 5.2 – Structure of the three proposed combination alternatives.

5.3.2 Feature selection

The *GeFS* measure, proposed by Nguyen et al., is used in this thesis for feature selection. The main concepts of the *GeFS* measure are explained next. For further details see [Nguyen et al., 2010b].

The feature selection problem can be defined as finding $x = (x_1, \dots, x_n) \in \{0,1\}^n$ that maximizes the function $GeFS(x)$:

$$\max_{x \in \{0,1\}^n} GeFS(x) = \frac{a_0 + \sum_{i=1}^{n} A_i(x) x_i}{b_0 + \sum_{i=1}^{n} B_i(x) x_i}, \tag{5.1}$$

where binary values of the variable x_i indicate the appearance ($x_i = 1$) or the absence ($x_i = 0$) of the feature f_i; a_0, b_0 are constants; $A_i(x)$, $B_i(x)$ are linear functions of variables $x_1, \dots, x_n$ and n is number of features.

There are several feature selection measures that can be represented by this form, such as the *Correlation Feature Selection* and *Minimal Redundancy Maximal Relevance* measures.

Correlation Feature Selection measure: The Correlation Feature Selection measure evaluates subsets of features on the basis of the following hypothesis: *"Good feature subsets contain features highly correlated with the classification, yet uncorrelated to each other"* [Hall, 1999].

Considering the merit of a feature subset S with k features as

$$Merit_{S_k} = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}},$$

the *CFS* criterion tries to maximize the merit of the subset of features:

$$\max_{S_k} \left\{ \frac{r_{cf_1} + r_{cf_2} + \dots + r_{cf_k}}{\sqrt{k + 2(r_{f_1 f_2} + .. + r_{f_i f_j} + .. + r_{f_k f_1})}} \right\}; \tag{5.2}$$

$\overline{r_{cf}}$ represents the average value of all feature-classification correlations and $\overline{r_{ff}}$ is the average value of all feature-feature correlations.

By using binary values of the variable x_i to indicate the appearance ($x_i = 1$) or the absence ($x_i = 0$) of the feature f_i in the globally optimal feature set, the expression (5.2) can be rewritten as an optimization problem:

$$\max_{x\in\{0,1\}^n}\left\{\frac{(\sum_{i=1}^n a_i x_i)^2}{\sum_{i=1}^n x_i+\sum_{i\neq j} 2b_{ij}x_i x_j}\right\}.$$

Minimal Redundancy Maximal Relevance measure: This method considers relevant features and redundant features simultaneously. For the given feature set S and the class c, $D(S,c)$ represents the relevance of S for the class c and $R(S)$ the redundancy of all features in set S. Then the *mRMR* measure is defined as

$$\max_S\{D(S,c)-R(S)\}. \tag{5.3}$$

Relevance is defined as

$$D(S,c)=\frac{1}{|S|}\sum_{f_i\in S} I(f_i;c),$$

and redundancy

$$R(S)=\frac{1}{|S|^2}\sum_{f_i,f_j\in S} I(f_i;f_j),$$

where $I(f_a,f_b)$ is the mutual information function, with f_a and f_b being features or the class label.

In the *mRMR* criterion, the binary values of the variable x_i are used in order to indicate the appearance ($x_i=1$) or the absence ($x_i=0$) of the feature f_i in the globally optimal feature set. The mutual information values $I(f_i;c)$, $I(f_i;f_j)$ are denoted by constants c_i, a_{ij}, respectively. Suppose that there are n full-set features, then, the problem (5.3) can be described as an optimization problem:

$$\max_{x\in\{0,1\}^n}\left\{\frac{\sum_{i=1}^n c_i x_i}{\sum_{i=1}^n x_i}-\frac{\sum_{i,j=1}^n a_{ij}x_i x_j}{(\sum_{i=1}^n x_i)^2}\right\}.$$

To solve the feature selection problem 5.1, it is transformed into a mixed 0-1 linear programming problem, which is later solved by using the branch and bound algorithm.

Next, the **search strategy** for obtaining relevant features and the criterion to choose the **appropriate instance of the *GeFS* measure** are detailed:

- ***Step 1***: Analyze the statistical properties of the given dataset in order to choose the appropriate instance (*CFS* or *mRMR*) of the *GeFS* measure. The criterion to choose the appropriate instance in each case is the following: the *CFS* measure is chosen if the dataset has many features that are linearly correlated to the class label and to each other. Otherwise, the *mRMR* measure is selected.

  In the case of web traffic, the methodology applied to implement this step is twofold:

  1. First, the corresponding subset of features is visualized in the two-dimensional space to get a plot matrix. In the matrix, each element represents the distribution of data points depending on, either the values of a feature and the class label, or the values of two features.
  2. With the aim of verifying the observations from the graphics, the next step is to calculate correlation coefficients between features. For that, the commonly used Pearson's correlation coefficient [Lutu, 2010] is used. These coefficients take values between -1 and 1. According to Cohen's criteria for interpreting correlations, values lower than 0.1 have no practical significance [Lutu, 2010]. Then, it is considered that there are linear correlations between features when at least half of the coefficients are greater than 0.1 [Nguyen, 2012].

  This methodology can be seen graphically in Fig. 5.3.

- ***Step 2***: According to the choice from *Step 1*, construct the optimization problem in 5.1 for the *CFS* or *mRMR* measures. In this step, expert knowledge can be used by assigning the value 1 to the variable if the feature is relevant and the value 0 otherwise.

- ***Step 3***: Transform the optimization problem of the *GeFS* measure to a mixed 0-1 linear programming (M01LP) problem, which is to be solved by means of a branch and bound algorithm. A non-zero integer value of x_i from the optimal solution x indicates the relevance of the feature f_i regarding the *GeFS* measure.

These steps are to be followed for every subset in which feature selection is applied.

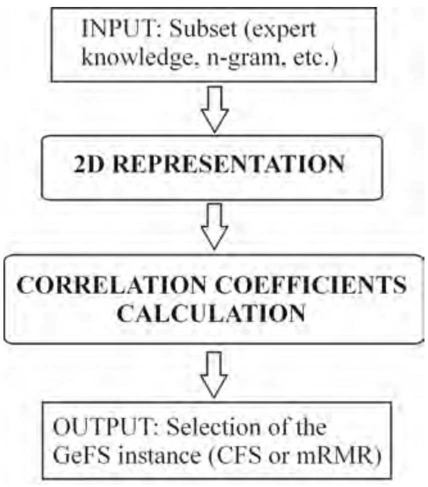

INPUT: Subset (expert knowledge, n-gram, etc.)
2D REPRESENTATION
CORRELATION COEFFICIENTS CALCULATION
OUTPUT: Selection of the GeFS instance (CFS or mRMR)

Figure 5.3 – Scheme for the selection of the appropriate instance of the *GeFS* measure.

5.3.3 Classification

The intrusion detection problem has been frequently formulated as a pattern recognition task [Nguyen, 2012]. Thus, it can be seen as a classification problem, where the requests should be classified as normal or anomalous.

Decision tree algorithms are predictive models that can be used as classifiers. They are very useful when massive volume of data need to be analyzed. In this work, the classification algorithms are in charge of deciding whether to label the incoming request as normal or anomalous. An advantage of decision trees is that they construct human-understandable classification rules, which could be later included in a signature-based detection system. In relation to that, papers like [Robertson et al., 2006] and [Bolzoni and Etalle, 2008] studied how to derive rules from an anomaly detection system.

As mentioned, four decision trees have been applied as classification algorithms in the ML system. Following, a brief explanation of each algorithm is presented. Further details about the algorithms can be found in [Duda et al., 2001] and [Wu et al., 2007].

- **C4.5** was introduced by Ross Quinlan. It is an algorithm used to generate decision trees that are built from a set of training data using the concept of information entropy [Quinlan, 1993]. At each node of the tree, C4.5 chooses the attribute that most effectively splits its samples into differentiated classes. For that, the criterion is to choose the attribute that gives the highest normalized information gain. Then the C4.5 algorithm recurs on the smaller sublists. The initial tree is pruned to avoid overfitting. Overfitting was explained in Sec. 2.5.

- **CART** stands for Classification And Regression Tree. It was popularized by Breiman et al. It is a recursive partitioning method that builds trees for predicting continuous dependent variables (regression) and categorical predictor variables (classification) [Breiman et al., 1984]. CART is a non-parametric algorithm. It generates a binary decision tree that is constructed by splitting the node that best differentiates the target variable into two child nodes repeatedly. It starts with the root node, that contains the whole learning sample. Important aspects of CART are deciding when the tree is complete and assigning a class to each terminal node.

- **Random trees** include the idea of selecting features randomly. This idea was introduced independently by Ho [Ho, 1995], [Ho, 1998] and Amit and Geman [Amit and Geman, 1997]. The implementation used in this thesis constructs a tree that considers K randomly chosen attributes at each node. It does not perform pruning to reduce the size of the decision tree. According to Olaru and Wehenkel, the goal of pruning "is to provide a good compromise between a model's simplicity and its predictive accuracy, by removing irrelevant parts of the model" [Olaru and Wehenkel, 2003].

- **Random forest** is an ensemble classifier that consists of many decision trees. Its output class is the mode of the class' output by individual trees. The algorithm for inducing a random forest was developed by Breiman [Breiman, 2001], Cutler and Stevens [Cutler and Stevens, 2006].

Figure 5.4 shows an example of a decision tree extracted from the Weka software [Hall et al., 2009]. It is built by the C4.5 algorithm over the *select-n-gram-combine* subset of the *ECML/PKDD* dataset when it is trained with 255 requests. The values in the leafs represent the class: normal is represented as 0 and abnormal as 1. The number in brackets mean the number of instances that fall into that category. The tree can be interpreted as follows: if the incoming payload contains ≤ 1 character '(' (attribute x85), then the request is classified as normal. Otherwise, if it has none or one character ')' (attribute x83), then the request is also considered normal. If this is not the case, it is analyzed if attribute x29 is lower or equal to 7. Attribute x29 represents the number of keywords in the path. In that case, the request is classified as anomalous. Otherwise, the length of the header "User-Agent" (attribute x13) is checked to label the request. This label is anomalous if x13 ≤ 66 and normal otherwise. As can be seen, it is easy to derive rules from a decision tree.

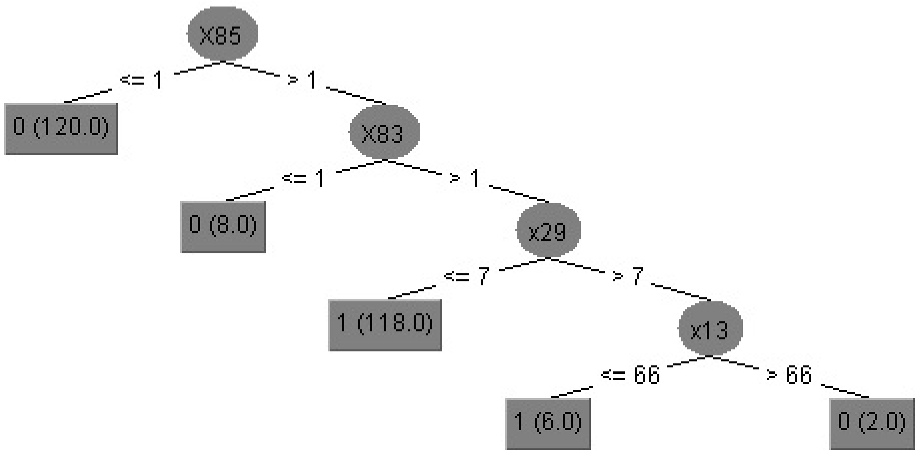


Figure 5.4 – Example of decision tree built with the C4.5 algoritm.

5.4 Experimental setup

This section explains the experimental setup of the experiments performed with the ML-based WAF. First, the datasets used are explained. Then, settings regarding the preprocessing stage are presented. At last, settings for the study of the influence of the training requests in the detection results are given.

5.4.1 Datasets

In this chapter, two datasets are used to experimentally evaluate the detection algorithms: our *CSIC* dataset and the *ECML/PKDD* dataset.

The *CSIC* dataset was presented in Chapter 3. It was employed in the previous chapter for evaluating stochastic algorithms. The advantage of using this dataset for stochastic and ML algorithms is that it makes possible the comparison of both techniques.

Additionally, the publicly available *ECML/PKDD* dataset is used. This dataset was generated for the ECML/PKDD 2007 Discovery Challenge [Raïssi et al., 2007]. It is composed of 50 000 samples, including 20% of attacks. The dataset is divided into the training and the test sets. The training set is employed for the experiments. Requests are labeled with specifications of normal traffic or the following attack classes: cross-site scripting, SQL injection, LDAP injection, XPATH Injection, path traversal, command execution and server-side include (SSI) attacks. Although the requests of this dataset are anonymized, we consider interesting to evaluate the system using a public dataset.

This dataset could not be used with stochastic systems due to the characteristics of these systems and the anonymity of the dataset. The fact that the *ECML/PKDD* dataset is anonymized implies that there are not two requests addressing the same web application. This makes the dataset unusable for the stochastic algorithms designed in this thesis given that they use the string corresponding to the resource as part of the detection process. The XML file generated would be huge and there would be no repeated resources, what does not make sense from the statistical point of view, since statistical algorithms need a number of requests in order to obtain meaningful results.

In the case of ML, the system extracts features from the resources, that is, they use properties of the resources, like their length, instead of the string of the resource name itself. These characteristics make the *ECML/PKDD* dataset applicable in the ML case.

5.4.2 Preprocessing Settings

In this section, the settings for the experiments regarding the preprocessing stage are presented.

The design mentioned in Sec. 5.3 is applied to all subsets of features: *expert knowledge*, *n-gram*, *combine-select*, *select-combine* and *select-n-gram-combine*. This section explains, for each subset, the settings related to feature extraction and feature selection. Regarding the first step, it is explained how each technique extracts features. For feature selection, the adequate instance of the *GeFS* measure is chosen for each subset, following the criterion that was given in Sec. 5.3.2. Additionally, it is shown how the opposite choice of the *GeFS* measure would negatively affect the results.

5.4.2.1 Expert knowledge

Next, feature extraction and feature selection settings for the *expert knowledge* subset are presented.

Feature extraction. By means of our expert knowledge and taking as inspiration [Rieck, 2009], various features that are considered relevant for web attack detection have been extracted. These 30 features are shown in Table 5.1.

As can be seen in the table, some of the features refer to the length of different parts of the request, since length is an important aspect to be considered in the detection of attacks such as buffer-overflow or XSS. Other group of features makes reference to the appearance of certain types of characters. In particular, to the number of appearances of letters, digits and non-alphanumeric characters in the path and argument values. Note that, differently to the stochastic case, the appearance of these characters is not analyzed in headers. The importance of distinguishing between these groups of characters was explained in Sec. 4.5. Going deeper into this idea, expert feature extraction in ML distinguishes two categories of non-alphanumeric characters, depending on whether they have a special meaning in certain programming languages or not.

Table 5.1 – Names of 30 expert knowledge features that are considered relevant for the detection of web attacks for the *ECML/PKDD* dataset.

Feature Name	Symbol
Length of the request	$\diamond$
Length of the path	
Length of the arguments	$\diamond$ $\star$
Length of the header "Accept"	†
Length of the header "Accept-Encoding"	
Length of the header "Accept-Charset"	
Length of the header "Accept-Language"	
Length of the header "Cookie"	
Length of the header "Content-Length"	
Length of the header "Content-Type"	
Length of the Host	
Length of the header "Referer"	
Length of the header "User-Agent"	
Method identifier	
Number of arguments	
Number of letters in the arguments	
Number of digits in the arguments	
Number of 'special' char in the arguments	● $\diamond$ $\star$
Number of other char in the arguments	● $\diamond$ $\star$
Number of letters in the path	
Number of digits in the path	
Number of 'special' char in the path	
Number of other char in path	
Number of cookies	
Minimum byte value in the request	$\diamond$
Maximum byte value in the request	
Number of distinct bytes	† $\star$
Entropy	$\diamond$
Number of keywords in the path	$\star$
Number of keywords in the arguments	

The symbol ● refers to features selected by the *CFS* for the *expert knowledge* subset, $\diamond$ to features selected by the *mRMR* for the *expert knowledge* subset, † to the characters selected by *CFS* for the *combine-select* subset and $\star$ to the characters selected by *mRMR* for the *combine-select* subset. (For the complete set of features selected for the *combine-select* alternative, see also Table 5.3).

Therefore, four kind of characters are considered for machine learning:

- Letters.
- Digits.
- Non-alphanumeric characters that have a special meaning in a set of programming languages, SQL and Javascript in our case. This type of characters are referred in Table 5.1 and Table 5.2 as 'special' chars.
- Other characters, i.e., non-alphanumeric characters that are not included in the third category.

Another feature is built by studying the entropy [Shannon, 2001] of bytes composing requests. Following the idea of Rieck, other group of features is built by counting the appearances of certain keywords in different parts of the request. These keywords have been previously included in a list that contains terms with special meaning in certain programming languages that are often used in injection attacks, like SQL and Javascript.

Feature selection. According to the methodology to choose the appropriate instance of the *GeFS* measure exposed in Sec. 5.3.2, the next steps are followed:

- Firstly, features are visualized in the two-dimensional space. The resulting graph represents the distribution of data points. All combinations have been exhaustively studied, that is, feature vs. feature and class vs. feature, for each feature and each class. As example, a few of these plots are shown in Fig. 5.5. It shows the data point distribution of the *expert knowledge* subset of the *ECML/PKDD* dataset. In the first example (a), feature "length of the arguments" is plotted vs. feature "length of the path". Differently, in the second example (b), the "number of letters in the arguments" is plotted vs. the "length of the path". Both normal and anomalous requests are used in the representation. These figures show that there is a non-linear relationship between the features extracted by means of expert knowledge.

Table 5.2 – Names of 30 expert knowledge features that are considered relevant for the detection of web attacks for the *CSIC* dataset.

Feature Name	Symbol
Length of the request	$\pm\ \ominus$
Length of the path	$\pm\ \ominus$
Length of the arguments	$\pm\ \ominus$
Length of the header "Accept"	$\cap$
Length of the header "Accept-Encoding"	$\cap$
Length of the header "Accept-Charset"	$\cap$
Length of the header "Accept-Language"	$\cap$
Length of the header "Cookie"	$\cap$
Length of the header "Content-Length"	$\cap$
Length of the header "Content-Type"	
Length of the Host	$\cap$
Length of the header "Referer"	$\cap$
Length of the header "User-Agent"	$\cap$
Method identifier	
Number of arguments	$\pm$
Number of letters in the arguments	$\pm\ \ominus$
Number of digits in the arguments	$\pm\ \ominus$
Number of 'special' char in the arguments	$\pm\ \cap\ \ominus$
Number of other char in the arguments	
Number of letters in the path	$\pm\ \ominus$
Number of digits in the path	$\pm\ \cap$
Number of 'special' char in the path	$\pm$
Number of other char in path	$\cap$
Number of cookies	$\cap$
Minimum byte value in the request	$\triangle$
Maximum byte value in the request	
Number of distinct bytes	$\ominus$
Entropy	$\pm\ \cap$
Number of keywords in the path	$\ominus$
Number of keywords in the arguments	

The symbol $\pm$ refers to features selected by the *CFS* for the *expert knowledge* subset, $\cap$ to features selected by the *mRMR* for the *expert knowledge* subset, $\triangle$ to the characters selected by *CFS* for the *combine-select* subset and $\ominus$ to the characters selected by *mRMR* for the *combine-select* subset. (For the complete set of features selected for the *combine-select* alternative, see also Table 5.4)

Examples for the *CSIC* dataset are shown in Fig. 5.6. They show feature "length of header 'Accept-Encoding' " vs. "length of the request" and feature "length of the arguments" vs. "length of the path". In this case, there are linear relationships between the features.

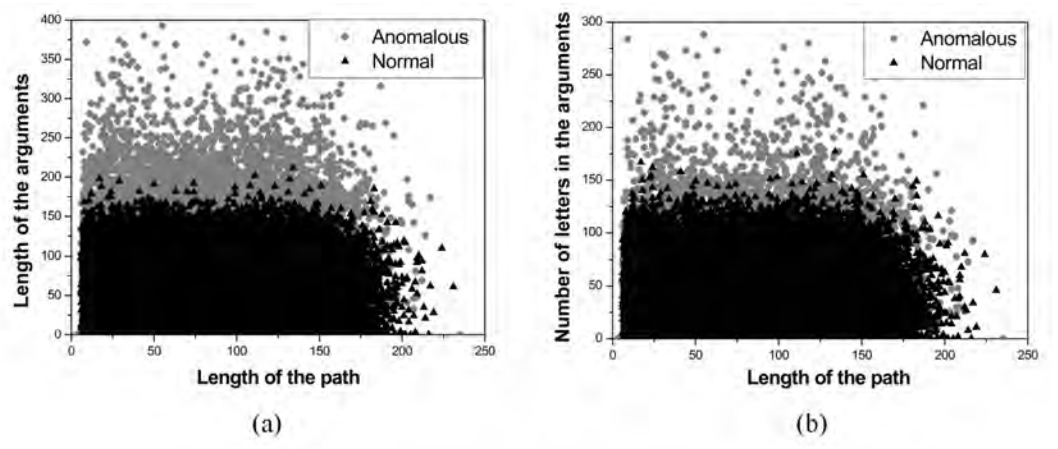


Figure 5.5 – Examples of the *expert knowledge* data point distribution for the *ECML/PKDD* dataset. (a) Feature "Length of the arguments" vs. feature "Length of the path". (b) Feature "Number of letters in the arguments" vs. feature "Length of the path".

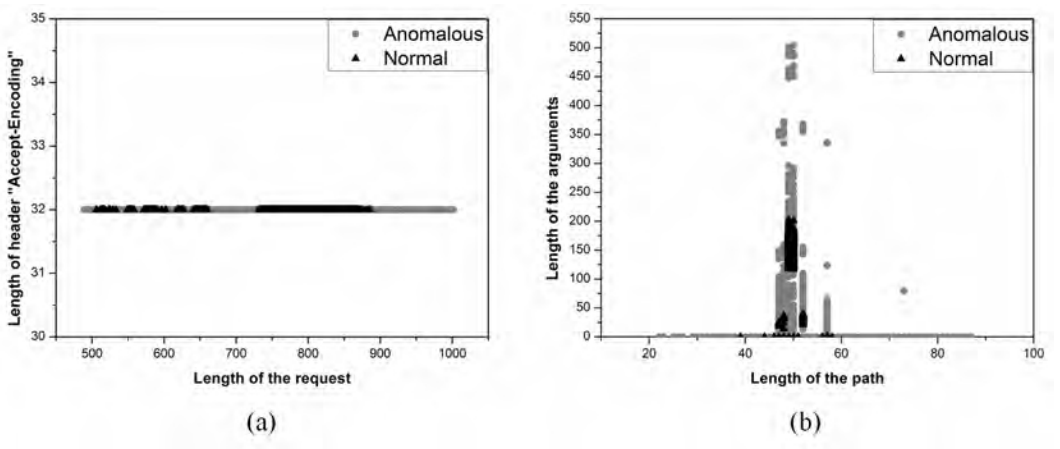


Figure 5.6 – Examples of the *expert knowledge* data point distribution for the *CSIC* dataset. (a) Feature "Length of header 'Accept-Encoding' " vs. feature "Length of the request". (b) Feature "Length of the arguments" vs. feature "Length of the path".

- In order to verify the observations from the graphics, the correlation coefficients between the features have been calculated. For the

ECML/PKDD dataset, more than 83% of the correlation coefficients are lower than 0.09. According to the criterion established in step 1 of Sec. 5.3.2, that considers that there are linear correlations between features when at least half of the coefficients are greater than 0.1, it can be stated that in this case, non-linear relationships are the most representative. Recall that the criterion explained in Sec. 5.3.2 establishes that the *CFS* measure is chosen if the dataset has many features that are linearly correlated to each other and *mRMR*, otherwise. Therefore, in this case the *mRMR* measure is chosen.

In the case of the *CSIC* dataset, more than 63% of the correlation coefficients are greater than 0.1, hence, the instance of the *GeFS* measure chosen is *CFS*.

Once the instances are selected, they are applied for selecting features. Figure 5.7 (a) represents, for the *ECML/PKDD* dataset, the number of features of the full subset (subset with all features, i.e., before feature selection) and the number of features after applying the *CFS* and *mRMR* instances. As can be seen, the number of features is reduced by 93% with *CFS* and 80% with *mRMR*. Although the selected instance for the *ECML/PKDD* dataset (*mRMR*) is not the one that reduces the most the number of features, this measure is more appropriate than *CFS* when detection results are also considered. It will be further explained in Sec. 5.5. The information about which particular features are selected by each measure can be found in Table 5.1. In the table, symbols $\bullet$ and $\diamond$ are used to represent the selection of each instance of the *GeFS* measure.

The number of features before and after feature selection for the case of the *CSIC* dataset is shown in Fig. 5.7 (b). The selected instance (*CFS*) reduces 63.3% of the features, while *mRMR* does it by 53.3%. In Table 5.2, symbols $\pm$ and $\cap$ are used to indicate which particular features are selected by the *GeFS* instances.

Note that in both Table 5.1 and Table 5.2, many features have several symbols next to them, which means that they have been selected by different feature selection instances and in diverse subsets. This indicates that they are important features for detecting attacks in the studied datasets.

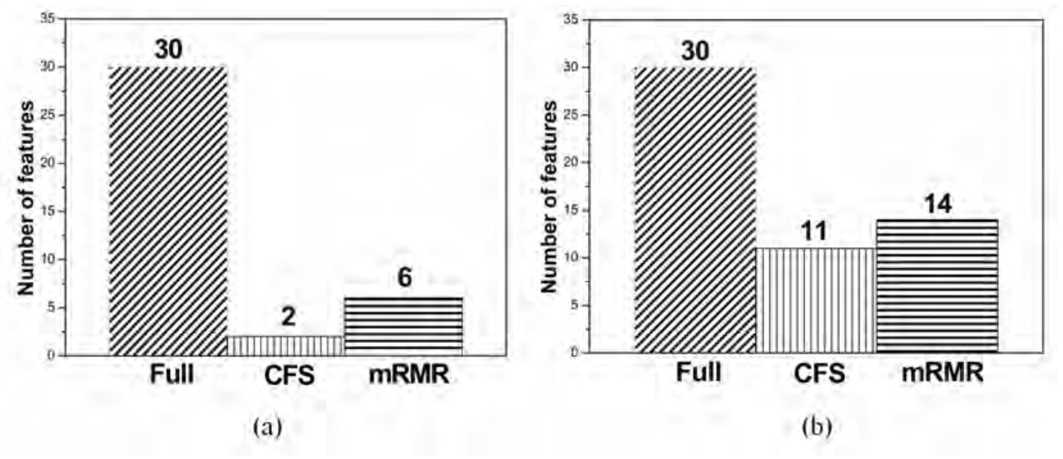


Figure 5.7 – Number of features for the *expert knowledge* subset. (a) For the *ECML/PKDD* dataset. (b) For the *CSIC* dataset.

5.4.2.2 *N*-grams

The settings regarding feature extraction and feature selection for n-grams are explained next.

Feature extraction. For extracting features with n-grams in the ML system, n-grams are set to $n = 1$. This value has been chosen since $n = 1$ is the simplest case. It should be considered that in web traffic there are no long strings since they are limited by the request length. In those cases, small values for n are usually chosen. Although it could be expected that the results would improve as n increases, there are papers in the literature using n-grams in HTTP traffic which show that it is not necessarily the case, like [Song et al., 2009] and [Perdisci et al., 2009]. Furthermore, cases with $n > 1$ require high cost in time and computational complexity, what is not appropriate for algorithms operating in real environment or scenarios with resource constraints. Therefore cases with $n > 1$ have not been considered in this thesis.

Following the formula given in Sec. 2.3.1.1 for calculating the number of n-grams, $S = \{n\text{-grams}_i | i = 1 \ldots 2^{8n}\}$, the number of all possible 1-grams is 256. Recall that the ML-based system analyzes the whole request. When analyzing the requests of the studied datasets, it was revealed that not all 1-grams were present. The result obtained was that only 96 features (37.5% of 256) appear at least once in the *ECML/PKDD* dataset and 114 (44.5%) in the case of the *CSIC* dataset. The features corresponding to the *ECML/PKDD* dataset are listed in Table 5.3 and those corresponding to the *CSIC* dataset in Table 5.4.

Table 5.3 – 96 characters appearing in the *ECML/PKDD* dataset at least once.

Character	Symbol	Character	Symbol	Character	Symbol	Character	Symbol
k	⋆	5		g		LF	• ◇
a	⋆	3		=		;	
_		:	◇	"	⋆	I	
9		0		D		h	
z		x		$	•	Q	
W		f		H		E	
R		4		Y		.	
7		,		6		q	
p		e		r		L	
i		b		&		m	
@		C		Space	• ◇	s	⋆
2		Z		A		u	
c		F		v		1	
y		LF		?		l	
M		j		8		n	
d		-	•	t	†	P	
*		K		/		G	
V		U		w		T	
S		o		N		J	
+	•	B		'	•	]	
O		X		%	•	)	◇ •
|	⋆	(	• ◇	\	• ⋆	~	
<	•	>		[	•	!	•
#	•	{		`	• †	}	

Symbol • refers to the 15 characters selected by the *CFS* measure for n-grams, ◇ to the 5 characters selected by the *mRMR* measure for n-grams, † to the characters selected by *CFS* for the *combine-select* alternative and ⋆ to the characters selected by *mRMR* for the *combine-select* alternative. For the complete set of features selected for the *combine-select* alternative see also Table 5.1.

With the assumption that normal traffic payloads are different from attack ones, the automatic method for extracting features from the requests is the following: given an HTTP request *req*, a feature vector of *req* is constructed as $x_{req} = (x_1, x_2, \ldots, x_\lambda)$, where x_i is the number of appearances of $n\text{-}gram_i$ in *req* and λ the number of 1-grams appearing at least once in the corresponding dataset. The vector constructed for every request represents the number of appearances of the corresponding n-gram in the HTTP request. In our case, 1-grams correspond to individual characters.

Table 5.4 – 114 characters appearing in the *CSIC* dataset at least once.

Character	Symbol	Character	Symbol	Character	Symbol	Character	Symbol
k		J		5		g	
a	∩	3	⊖	=	∩	;	± ⊖
:		O		I	±	9	
0		D		h		z	
x		+	±	[		B	
f		H		E		4	±
.		7		,		6	
q	∩	p	⊖	e		r	
)		L	± ∩	i	⊖	b	
m		C		Space		s	
2		A		u		c	
F		1		LF		l	
M		j		8		n	∩
d	∩ ⊖	-	± ∩ ⊖	t	∩ ⊖	P	⊖
*		K		/	±	G	
U		(		T		S	
o		N		R		&	±
ñ		?	△	V		y	
w		ó		v		@	
á		Q		%	±	~	
ç		é		ú		X	
W		Á		í		#	
<	±	>		’	±	_	
à		!		"		ä	
ü		ù		ö		Z	
ò		º		À		É	
Í		Ñ		Ó		è	
ì		ï		|	±	Y	
$		Ú	△				

Symbol ± refers to features selected by the *CFS* measure for the n-grams, ∩ to features selected by the *mRMR* for n-grams, △ to the characters selected by *CFS* for the *combine-select* subset and ⊖ to the characters selected by *mRMR* for the *combine-select* subset. For the complete set of features selected for the *combine-select* alternative see also Table 5.2.

Feature selection. The scheme cited in Sec. 5.3.2 has been applied for feature selection:

- Examples of the visualization of n-gram features in the two-dimensional space are shown in Fig. 5.8 for the *ECML/PKDD* dataset. Figure 5.9 shows examples for the *CSIC* dataset. As can be observed, there are linear relations between the n-gram features of both datasets.

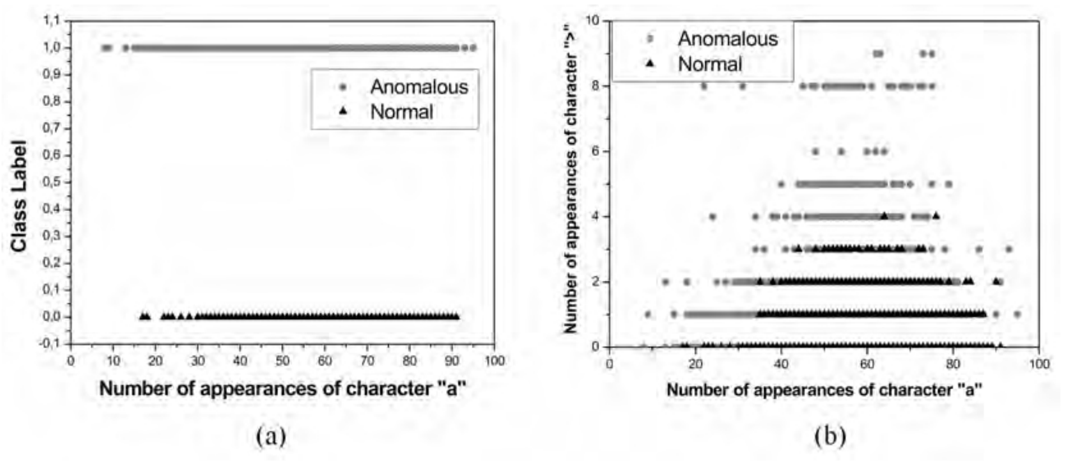


Figure 5.8 – Examples of the *n-gram* data point distribution of the *ECML/PKDD* dataset. (a) Feature “Class label” vs. feature “Number of appearances of character ‘a’ ”. (b) Feature “Number of appearances of character ‘>’ ” vs. feature “Number of appearances of character ‘a’ ”.

- The study of the correlation coefficients reveals the following: more than 52% of the correlation coefficients are greater than 0.1 for the *ECML/PKDD* dataset. According to the criterion explained before, it means that there are linear relationships, and then, the selected measure for the *n-gram* subset is *CFS*. For the *CSIC* dataset, 57% of the coefficients are higher than 0.1, hence, *CFS* is selected. This confirms the observations from the graphs.

  Figure 5.10 (a) shows the number of features for the full-set of the *ECML/PKDD* dataset and the number of features after feature selection. It can be seen that *CFS* reduces 84% (from 96 to 15) of the irrelevant or redundant features for detecting web attacks, while *mRMR* reduces 95% of them.

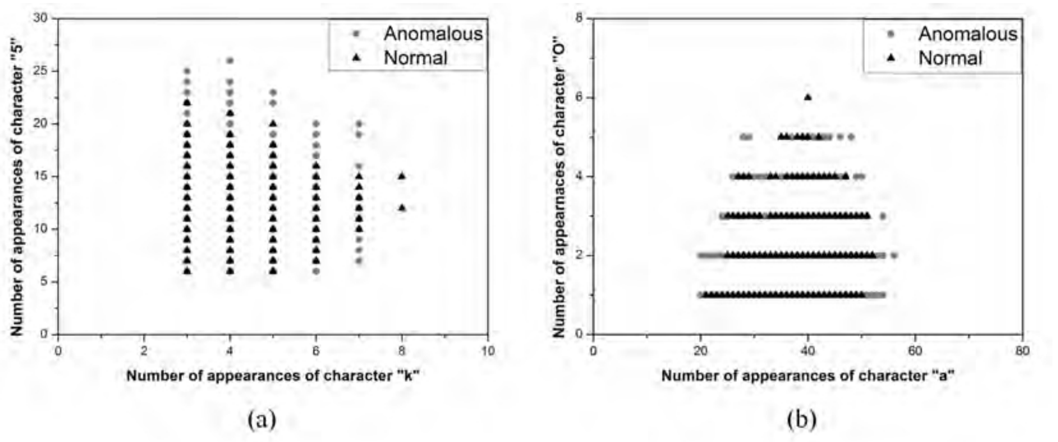


Figure 5.9 – Examples of the *n-gram* data point distribution of the *CSIC* dataset. (a) Feature "Number of appearances of character '5' " vs. feature "Number of appearances of character 'k' ". (b) Feature "Number of appearances of character 'O' " vs. feature "Number of appearances of character 'a' ".

In the case of the *CSIC* dataset, the reduction is represented in Fig. 5.10 (b), where the *mRMR* instance reduces the number of features almost by 93%, while *CFS* does it by 89.5%.

The specific features selected by each *GeFS* instance can be observed in Table 5.3 for the *ECML/PKDD* dataset (represented with symbols • and ⋄), and in Table 5.4 for the *CSIC* dataset (symbolized by ± and ∩). It is remarkable that, indeed, some of the 1-grams selected are critical for the detection of web attacks. For instance the quotation mark (') is included in many SQL injection attacks, characters '<' and '>' are typically appearing in scripts such as the ones used in XSS attacks, and '%' has a special meaning in SQL or shell script.

5.4.2.3 Combination

Next, settings for the combination cases are shown.

Feature extraction. After selecting the appropriate feature selection instance for the *expert knowledge* and *n-gram* subsets of features, the exact number of features of the three combination subsets can be specified:

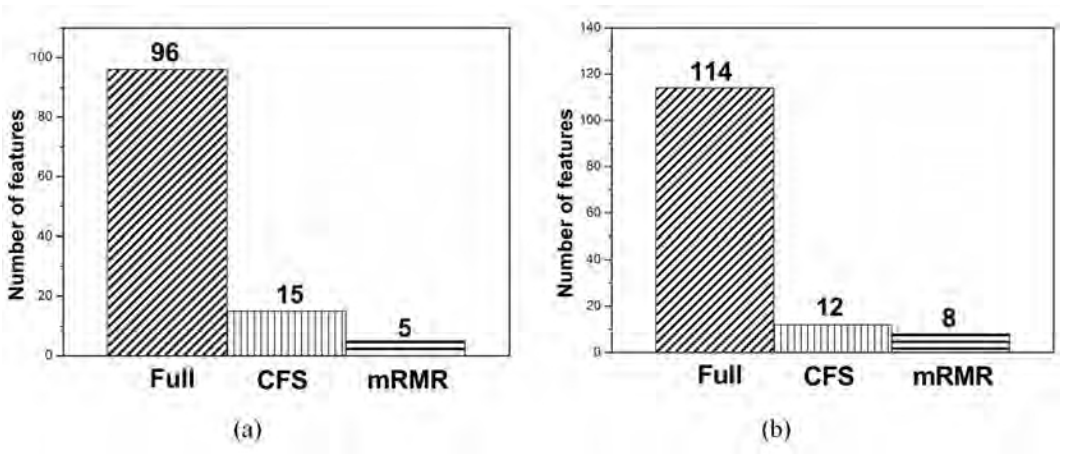


Figure 5.10 – Number of features for the *n-gram* subset.
(a) For the *ECML/PKDD* dataset. (b) For the *CSIC* dataset.

- *Combine-select.* In the case of the *ECML/PKDD* dataset, the resulting subset of the *combine-select* alternative is composed of 126 features in total, corresponding to the combination of 30 features from expert knowledge and 96 features from n-grams. For the *CSIC* dataset, the subset is composed of 30 features from expert knowledge and 114 features from n-grams, resulting in a total of 144 features. The process of feature selection applied to these subsets is later explained in this section.

- *Select-combine.* Recall that this alternative applies first feature selection and then combines the features. There are four options for generating the subsets of features, corresponding to the two instances of the *GeFS* measure (*CFS* and *mRMR*) for selecting features from expert knowledge and n-grams. The reasonable subset to be constructed is the one composed of the features selected by the appropriate instances of the *GeFS* measure. Then, for the *ECML/PKDD* dataset the subset is the one constituted by 15 n-gram features (selected by *CFS*) plus 6 expert knowledge features (selected by *mRMR*). For the *CSIC* dataset, it is formed by 11 expert knowledge features (selected by *CFS*) and 12 n-gram features (selected by *CFS*). These subsets are called *mRMR+CFS* and *CFS+CFS* due to the selected instances respectively. Additionally, another subset with features selected by the non-chosen *GeFS* instance is also shown in the Results section (Sec. 5.5), with the purpose to see how the opposite choice of feature selection methods would negatively affect the detection. Therefore, the new subset *mRMR+mRMR* is added to the previous subsets. This subset is composed of 11 features $(6 + 5)$ in the case of the *ECML/PKDD* dataset and 22 $(14 + 8)$ in the case of the *CSIC* one. Since the features that compose these two subsets are already selected, it is not necessary to apply feature selection again.

- *Select-n-gram-combine.* As only *n*-grams are selected with this alternative, there are two possible subsets to be considered, corresponding to the two *GeFS* instances possible for the *expert knowledge* subset. These two subsets are called *expert+CFS* and *expert+mRMR*. In this case, since both datasets use the *CFS* instance for *n*-grams, the subset chosen for the experiments is *expert+CFS*. The results are also shown for the *expert+mRMR* subset.

As a summary, Table 5.5 shows the structure of the subsets corresponding to each combination alternative for the *ECML/PKDD* dataset. An equivalent table is shown in Table 5.6 for the *CSIC* dataset. The number of features of the *combine-select* subset corresponds to the full subset, that is, before feature selection. Next section explains the subset after feature selection.

Table 5.5 – Description of the subsets corresponding to each combination alternative for the *ECML/PKDD* dataset.

Alternative	Subset Name	Expert Knowledge Features	*N*-gram Features	Total Number of Features
Combine-select	Combine-select	30	96	126
Select-combine	*mRMR+CFS*	6 (*mRMR*)	15 (*CFS*)	21
	mRMR+mRMR	6 (*mRMR*)	5 (*mRMR*)	11
Select-n-gram-combine	*Expert+CFS*	30	15 (*CFS*)	45
	Expert+mRMR	30	5 (*mRMR*)	35

Feature selection. In the case of the combination subsets, feature selection is applied after feature extraction only in the case of the *combine-select* subset. The statistical properties of the *select-combine* and *select-n-gram-combine* subsets are not necessary to be studied given that the features in these subsets have been already selected in the basic cases.

Since the *combine-select* subset contains a higher amount of features and they have manual and automatic nature, it is specially useful to apply feature

Table 5.6 – Description of the subsets corresponding to each combination alternative for the *CSIC* dataset.

Alternative	Subset Name	Expert Knowledge Features	*N*-gram Features	Total Number of Features
Combine-select	Combine-select	30	114	144
Select-combine	*CFS+CFS*	11 (*CFS*)	12 (*CFS*)	23
	mRMR+mRMR	14 (*mRMR*)	8 (*mRMR*)	22
Select-n-gram-combine	*Expert+CFS*	30	12 (*CFS*)	42
	Expert+mRMR	30	8 (*mRMR*)	38

selection in this case, in order to reduce the number of redundant and irrelevant features. The instance of the *GeFS* measure is selected as follows:

- Two examples of the data point distribution of the *combine-select* subset for the *ECML/PKDD* dataset are represented in Fig. 5.11. They show that there are not linear relationships between features. The examples for the *CSIC* dataset, represented in Fig. 5.12, also show no linear relationships.

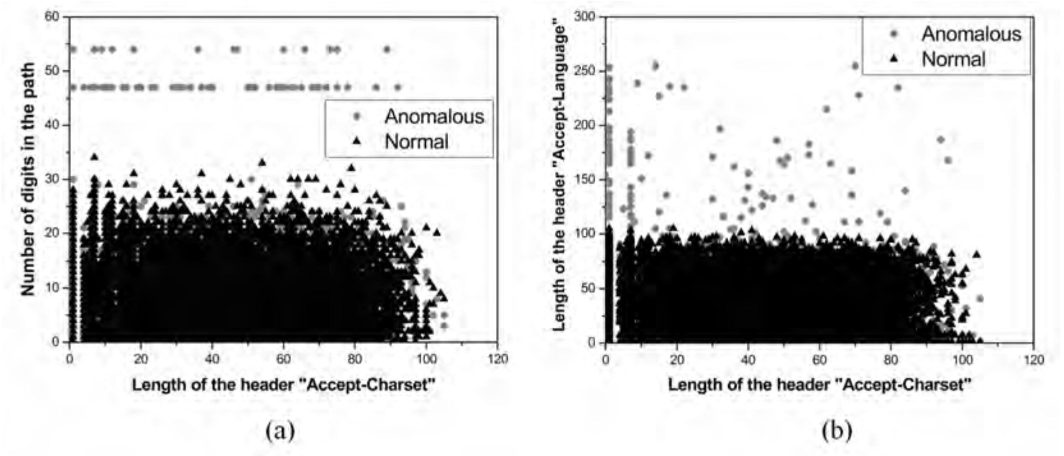


Figure 5.11 – Examples of the *combine-select* data point distribution of the *ECML/PKDD* dataset. (a) Feature "Number of digits in the path" vs. feature "Length of the header 'Accept-Charset' ". (b) Feature "Length of the header 'Accept-Language"' vs. feature "Length of the header 'Accept-Charset' ".

- The analysis of the correlation coefficients corresponding to the *combine-select* subset confirms the observations of the graphs. For the *ECML/PKDD* dataset, more than 76% of the coefficients are lower than 0.09, what means that there are non-linear relations between the features

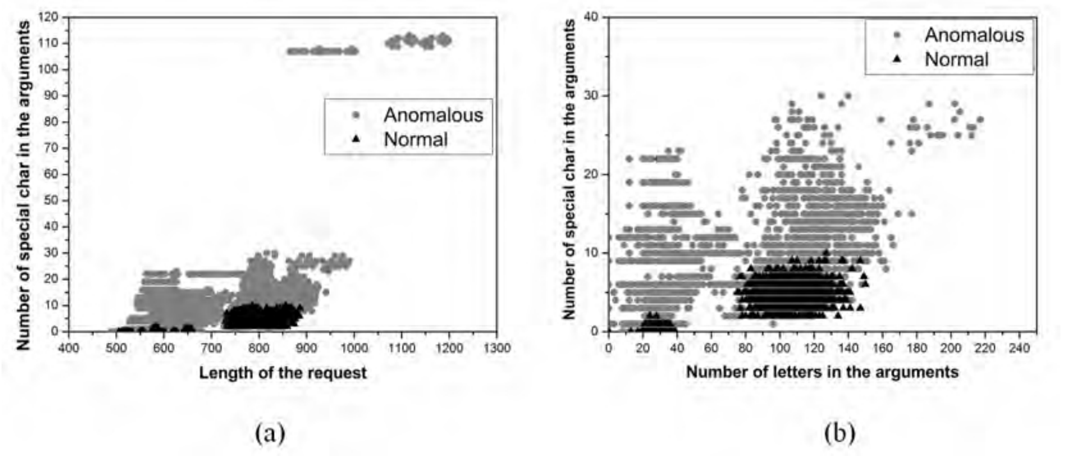


Figure 5.12 – Examples of the *combine-select* data point distribution of the *CSIC* dataset. (a) Feature "Number of special char in the arguments" vs. feature "Length of the request ". (b) Feature "Number of special char in the arguments"' vs. feature "Number of letters in the arguments") of the *CSIC* dataset.

of this subset. Therefore, the *mRMR* measure is the convenient one for this case. In the case of the *CSIC* dataset, 66% of the coefficients are lower than 0.09, then the *mRMR* measure is also chosen in this case.

After selecting the corresponding instance of the *GeFS* measure, it is applied for selecting features in the *combine-select* subset. For the *ECML/PKDD* dataset, Fig. 5.13 (a) shows the number of features of the full subset and for the subsets after feature selection. Although the instance selected in this case is *mRMR*, both instances are shown for comparison. The reduction in the number of features for the *combine-select* subset rises up to 91.27% for the *mRMR* instance and to 96.83% for *CFS*.

The corresponding graph for the *CSIC* dataset is represented in Fig. 5.13 (b). The reduction is 97.92% for *CFS* and 88.2% for *mRMR*. *mRMR* is the selected instance. Although it is not the instance that reduces the most the number of features, it can lead to better detection results, as will be seen in Sec. 5.5.

The features selected from the *combine-select* subset of the *ECML/PKDD* dataset are represented in Table 5.1 and Table 5.3. Symbol † is used for *CFS* and ⋆ for *mRMR*. For the *CSIC* dataset, symbols △ and ⊖ are used in Table 5.2 and Table 5.4 to represent the selected features. Note that, for both datasets,

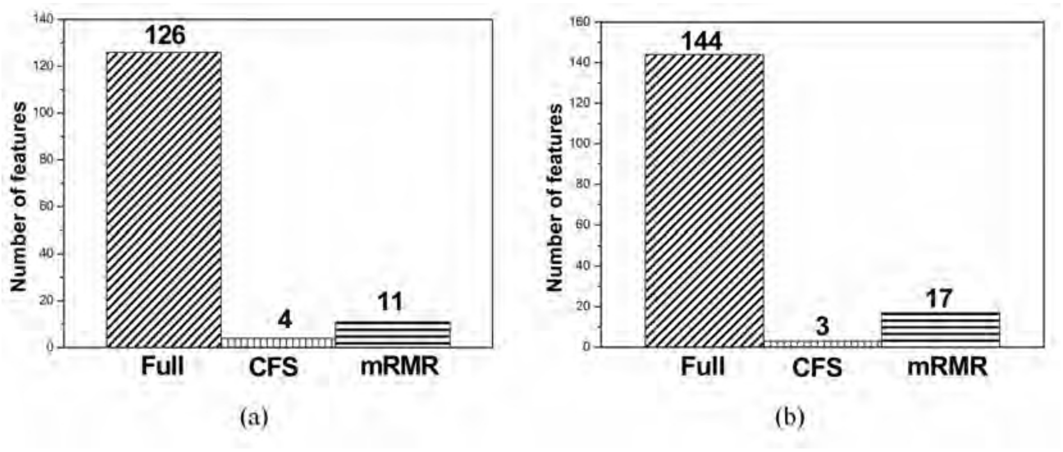


Figure 5.13 – Number of features for the *combine-select* subset. (a) For the *ECML/PKDD* dataset. (b) For the *CSIC* dataset.

many features selected by the *CFS* and *mRMR* measures are the same in the case of *expert knowledge*, *n-grams* as well as in the *select-combine* case, what indicates that these features are significant for the detection of web attacks.

The number of features of the *select-combine* and *select-n-gram-combine* subsets of the *ECML/PKDD* and *CSIC* datasets can be seen in Fig. 5.14 (a) and (b) respectively.

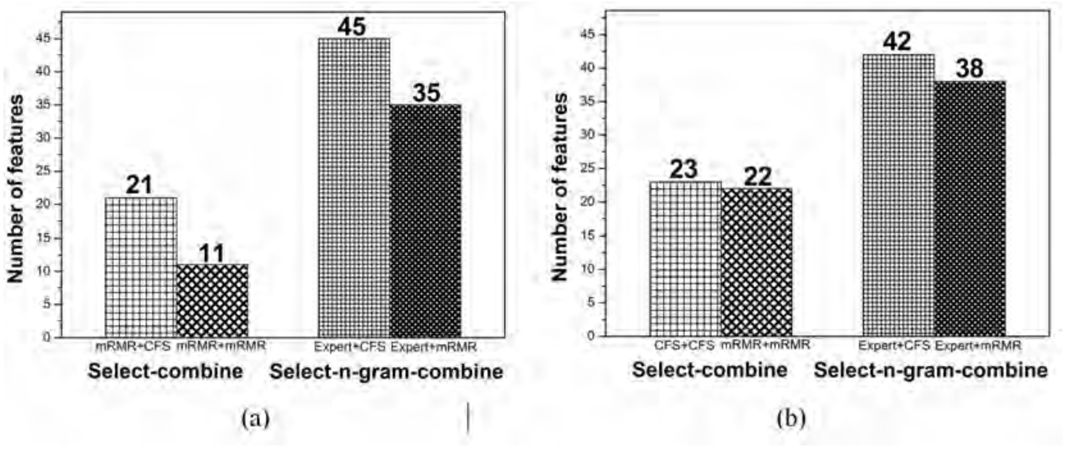


Figure 5.14 – Number of features of the *select-combine* and *select-n-gram-combine* subsets. (a) For the *ECML/PKDD* dataset. (b) For the *CSIC* dataset.

In summary, Table 5.7 shows the *GeFS* instances chosen for every subset of the *ECML/PKDD* and *CSIC* datasets. As can be seen, the appropriate instances of the *GeFS* measure are the same chosen for both datasets in all cases, except for the *expert knowledge* subset.

Table 5.7 – *GeFS* instance chosen for each subset of the *ECML/PKDD* and *CSIC* datasets.

Subset	*GeFS* instance	
	ECML/PKDD	*CSIC*
Expert Knowledge	*mRMR*	*CFS*
n-gram	*CFS*	*CFS*
Combine-select	*mRMR*	*mRMR*
Select-combine	Nothing	Nothing
Select-n-gram-combine	Nothing	Nothing

5.4.2.4 Classification algorithms

As mentioned, the experiments are carried out with the four decision trees: C4.5, CART, Random Tree and Random Forest. These algorithms are described in Sec. 5.3.3. These algorithms receive as input the subsets of features. In particular, they are applied to analyze all subsets of features (*expert knowledge*, *n-gram*, *combine-select*, *select-combine* and *select-n-gram-combine*), both before and after feature selection.

Decision trees have two stages: training and test. In the training phase, normal and anomalous requests feed the algorithm to obtain a model that classifies the traffic. In the test phase the knowledge previously learned is used to check how well the algorithm detects, i.e, if the requests are correctly classified as normal or anomalous.

The implementation of decision trees used are those provided by the Weka software (University of Waikato, Hamilton, New Zealand) [Hall et al., 2009]. Cross validation is a validation technique for assessing how the results of the algorithm will generalize to another data set. In N-fold cross validation, the dataset is divided into N equally sized subsets. It is run N times. In each of them, one of the subsets is used for testing and the remaining $N-1$ are used for training the algorithm. The results are averaged and standard deviation is calculated [Kubat, 2015]. The use of cross validation allows to obtain more reliable results. In our case, experiments are conducted with 10-fold cross validation. According to Kohavi, 10-fold stratified cross validation is an optimal method for real world datasets [Kohavi, 1995]. The remaining setting values used are those set by default in the Weka software.

5.4.3 Settings for the study of the influence of training requests in the detection results

As mentioned, decision trees have a training and a test phase. For this kind of algorithms, both normal and anomalous requests are used to train the system. In this way, the algorithm can learn how to classify the instances (two-class detection: normal and anomalous). Both types of traffic are also used in the test period.

In order to study how the number of training requests influences the performance of the system, once all the previous subsets of features are evaluated, the one that reaches the best results is chosen for the study of how the number of requests influences the algorithm's detection results.

For that, a number of $M = 15$ experiments is run, with an increasing number of requests per experiment. Like in the stochastic case, the number of training requests used in each experiment is given by the formula $tr_i = 2^i - 1, \forall i \in [1, M]$. In each experiment, the system is firstly trained with tr_i requests randomly chosen, and then, it is tested with a fixed subset of requests. This fixed subset is composed of te normal request plus te anomalous requests, with $te = 1000$. The M experiments are run $H = 10$ times. Each time, different possibilities for choosing tr_i requests are analyzed. Running the experiments multiple times helps to truly reflect the behavior of the WAF, independently of which training requests are chosen. Randomness has been picked for request sampling since it is not possible to know in advance which type of attack the system is going to receive. The next section shows the results obtained from the experiments.

5.5 Results

This section shows, on the one hand, the results of the different subsets, in order to determine the effectiveness of the combined methods proposed. On the other hand, the results of the study about the influence of the number of training requests in the detection capacity of the WAF are presented.

5.5.1 Results of the subsets

With the purpose of validating whether the proposed combined feature extraction methods are more effective than individual techniques, the results corresponding to the two basic cases are shown first, and then, those corresponding to the three combination alternatives. Detection results are presented in terms of DR and FPR. Additionally, the number of features used by each technique and its processing time are also included. The number of features used gives a measure of the resource consumption. The achieved results are measured by means of detection rate and false positive rate.

5.5.1.1 Expert knowledge

Regarding expert knowledge, Table 5.8 shows the detection results of the four classification algorithms for the *ECML/PKDD* dataset. The rows show the results for the different decision trees used and, finally, the average value. The column *full-set* shows the results before feature selection and the *CFS* and *mRMR* columns correspond to the results after feature selection. In the table, the selected *GeFS* instance is highlighted with bold letters (*mRMR* in the case of expert knowledge).

When considering also the number of features, shown in Fig. 5.7 (a), it is noticeable that the *GeFS* measure greatly reduces the number of features, while it keeps almost the same detection results. Although *CFS* reduces the number of features (from 30 to 2) more than *mRMR* (from 30 to 6), its detection results are worse, with lower DR (89.1% vs. 91.22%) and higher FPR (23.18% vs. 15.3%). This fact shows that the method used to choose the proper instance of the *GeFS* leads to select the instance that reaches better results, although it might not be the one that selects a lower amount of features. Note that the

Table 5.8 – Detection rate and false positive rate of four decision trees performed on the *expert knowledge* subset of the *ECML/PKDD* dataset.

Classifiers	Detection Rate (%)			False Positive Rate (%)		
	Full-set	*CFS*	*mRMR*	Full-set	*CFS*	*mRMR*
C4.5	95.42	89.07	**92.30**	7.8	23.2	**15.1**
CART	95.51	89.12	**92.23**	7.9	23.1	**14.7**
Random Tree	92.43	89.11	**88.81**	10.6	23.2	**16.3**
Random Forest	95.80	89.10	**91.53**	8.1	23.2	**15.1**
Average	**94.79**	**89.10**	**91.22**	**8.6**	**23.18**	**15.3**

Bold letters are used to highlight the selected subset, corresponding to the chosen *GeFS* instance.

goal of reducing the number of features and improving the detection results are contradictory, therefore, a tradeoff should be made. Feature selection attempts to reduce the number of features without negatively affecting the detection results.

This reduction in the number of features also implies a decrease in the processing time measurements of the system. Table 5.9 shows the processing time values associated to the results presented in Table 5.8. The third column shows the difference in ms/request between the first and second column, that is, before and after feature selection. The fourth column represents the improvement percentage that this difference represents. Note that the average improvement is calculated as the percentage that the average difference represents, not as the average of the four values in the improvement column. The average difference is calculated as the difference between the average values of the first and second columns. It can be seen that feature selection signifies an average reduction of 67.33% in the processing time (from 1.01 ms/request to 0.33 ms/request). All processing time measurements have been obtained with an Intel core i7 CPU at 2.40 GHz and 8GB RAM, SO Windows 8, 64 bits. Processing times should be taken as approximate values, as other factors could have interfered in the measurements, such as other tasks running in the operating system or memory availability.

Results for expert knowledge over the *CSIC* dataset are shown in Table 5.10. As opposed to the *ECML/PKDD* dataset, the instance chosen for the *CSIC* dataset is *CFS*, which is highlighted in bold type. In this case, Fig. 5.7 (b)

Table 5.9 – Processing time and improvement for the *expert knowledge* subset of the *ECML/PKDD* dataset.

Classifiers	Processing Time (ms/req.)		Improvement	
	Full-set	*mRMR*	Difference	Improvement(%)
C4.5	2.12	0.24	1.88	88.68
CART	0.64	0.13	0.51	79.69
Random Tree	0.22	0.12	0.10	45.45
Random Forest	1.07	0.84	0.23	21.50
Average	**1.01**	**0.33**	**0.68**	**67.33**

shows that *CFS* is the instance that reduces the most the number of features (it uses 11 features) and also the instance that gets better results (93.55% detection rate vs. 75.5% for *mRMR*).

Table 5.10 – Detection rate and false positive rate of four decision trees performed on the *expert knowledge* subset of the *CSIC* dataset.

Classifiers	Detection Rate (%)			False Positive Rate (%)		
	Full-set	*CFS*	*mRMR*	Full-set	*CFS*	*mRMR*
C4.5	94.98	**94.07**	79.8	5.3	**6.8**	25.7
CART	94.31	**93.72**	79.86	7.0	**6.8**	25.3
Random Tree	92.76	**92.71**	71.36	7.7	**7.8**	30.6
Random Forest	93.93	**93.69**	71.70	6.0	**7.2**	30.5
Average	**93.99**	**93.55**	**75.5**	**6.5**	**7.15**	**28.02**

As Table 5.11 shows, in this case the average improvement in the processing time is 43.18%.

5.5.1.2 *N*-grams

Table 5.12 summarizes the performance of the four decision trees over the *n-gram* subset of the *ECML/PKDD* dataset. The results confirm that the selection of the *GeFS* instance leads to better results. In this case it is *CFS*, that is highlighted in bold letters in the table. It can be observed that when the instance is chosen differently to the linear correlation criterion explained in Sec. 5.3.2, the results are negatively affected. This shows the importance

Table 5.11 – Processing time and improvement for the *expert knowledge* subset of the *CSIC* dataset.

Classifiers	Processing Time (ms/req.)		Improvement	
	Full-set	*CFS*	Difference	Improvement(%)
C4.5	1.22	0.42	0.80	65.57
CART	0.52	0.25	0.27	51.92
Random Tree	0.16	0.13	0.03	18.75
Random Forest	1.61	1.21	0.40	24.84
Average	**0.88**	**0.50**	**0.38**	**43.18**

of the proper selection of the *GeFS* instance. Considering the number of features in Fig. 5.10 (a), as well as the detection results in Table 5.12, it can be observed that besides reducing the number of features from 96 to 15, the *CFS* measure is even able to improve the detection results (from 92.99% to 93.47%). Furthermore, the *GeFS* measure is able to reduce the resource consumption. This can be appreciated in Table 5.13, that shows an average improvement of 90.82% in the processing time after applying feature selection.

Table 5.12 – Detection rate and false positive rate of four decision trees performed on the *n-gram* subset of the *ECML/PKDD* dataset.

Classifiers	Detection Rate (%)			False Positive Rate (%)		
	Full-set	*CFS*	*mRMR*	Full-set	*CFS*	*mRMR*
C4.5	94.12	**94.12**	89.95	9.2	**10.4**	18.7
CART	94.92	**94.16**	90.03	8.7	**10.2**	18.3
Random Tree	88.55	**91.90**	88.73	15.7	**11.9**	22.0
Random Forest	94.41	**93.70**	89.12	11.0	**10.7**	20.7
Average	**92.99**	**93.47**	**89.45**	**11.15**	**10.8**	**19.9**

For the *CSIC* dataset, the instance of the *GeFS* measure chosen is also *CFS*. Considering the number of features in Fig. 5.10 (b) and the results in Table 5.14, it can be seen that, in this case, the feature selection measure is able to reduce the number of features by 84%, at the expense of reducing the detection rate by 2.16% (from 84.59% to 82.43%). Table 5.15 shows that the processing time improvement for the *n-gram* subset of the *CSIC* dataset ascends to 85.48% in average.

Table 5.13 – Processing time and improvement for the *n-gram* subset of the *ECML/PKDD* dataset.

Classifiers	Processing Time (ms/req.)		Improvement	
	Full-set	*CFS*	Difference	Improvement(%)
C4.5	9.0	1.05	7.95	88.33
CART	8.73	0.38	8.35	95.65
Random Tree	0.36	0.25	0.11	30.56
Random Forest	2.39	0.21	2.18	91.21
Average	**5.12**	**0.47**	**4.65**	**90.82**

Table 5.14 – Detection rate and false positive rate of four decision trees performed on the *n-gram* subset of the *CSIC* dataset.

Classifiers	Detection Rate (%)			False Positive Rate (%)		
	Full-set	*CFS*	*mRMR*	Full-set	*CFS*	*mRMR*
C4.5	84.61	**82.45**	76.77	16.8	**24.4**	25.6
CART	85.74	**82.54**	77.63	17.2	**23.9**	26.0
Random Tree	81.93	**82.34**	78.39	19.3	**23.7**	25.5
Random Forest	86.09	**82.39**	78.61	17.1	**23.4**	25.1
Average	**84.59**	**82.43**	**77.85**	**17.6**	**23.85**	**25.55**

5.5.1.3 Combination cases

In this section, the results corresponding to the three combination alternatives are shown.

- *Combine-select*. The results of this subset for the *ECML/PKDD* dataset are shown in Table 5.16. As expected, it is confirmed that the *GeFS* instance chosen (*mRMR*) gets the best results. It reduces the number of features from 126 to 11, as can be seen in Fig. 5.13 (a), while the detection results vary from 96.13% to 91.38%, as Table 5.16 shows. This *GeFS* instance gets an average improvement of the processing time of 40%, as can be seen in Table 5.17.

  Like for the *ECML/PKDD* dataset, the appropriate *GeFS* instance for the *CSIC* dataset is *mRMR*. The detection results obtained before and after applying this instance are shown in Table 5.18. The number of

Table 5.15 – Processing time and improvement for the *n-gram* subset of the *CSIC* dataset.

Classifiers	Processing Time (ms/req.)		Improvement	
	Full-set	*CFS*	Difference	Improvement(%)
C4.5	9.26	0.60	8.66	93.52
CART	4.80	0.24	4.56	95
Random Tree	0.29	0.15	0.14	48.28
Random Forest	2.72	1.5	1.22	44.85
Average	**4.27**	**0.62**	**3.65**	**85.48**

Table 5.16 – Detection rate and false positive rate of four decision trees performed on the *combine-select* subset of the *ECML/PKDD* dataset.

Classifiers	Detection Rate (%)			False Positive Rate (%)		
	Full-set	*CFS*	*mRMR*	Full-set	*CFS*	*mRMR*
C4.5	97.34	76.06	**92.42**	4.0	46.4	**14.2**
CART	97.41	76.10	**92.52**	4.2	46.2	**14.3**
Random Tree	92.78	72.52	**88.18**	10.0	48.9	**15.9**
Random Forest	96.98	72.32	**92.41**	5.6	47.3	**14.5**
Average	**96.13**	**74.25**	**91.38**	**5.95**	**47.2**	**14.72**

features can be seen in Fig. 5.13 (b). In this case, besides reducing the number of features from 144 to 17, the feature selection algorithm is also able to increase the detection rate, from 92.69% to 93.61%. Therefore, in comparison with the full set of features, it improves the resource consumption and the detection results at the same time. This fact shows that a lower number of features does not necessary mean to obtain worse results. Regarding the processing time, Table 5.19 shows that the *mRMR* measure is able to reduce it by 28.28%.

- *Select-combine.* In relation to this alternative, the results of the subsets *mRMR+CFS* and *mRMR+mRMR* corresponding to the *ECML/PKDD* dataset are shown in Table 5.20. Recall that the *mRMR+CFS* subset (highlighted with bold letters in the table) is created with the features selected by the appropriate instances of the *GeFS* measure. The *mRMR+mRMR* subset is created for the sake of comparison, choosing

Table 5.17 – Processing time and improvement for the *combine-select* subset of the *ECML/PKDD* dataset.

Classifiers	Processing Time (ms/req.)		Improvement	
	Full-set	*mRMR*	Difference	Improvement(%)
C4.5	0.87	0.60	0.27	31.03
CART	0.62	0.22	0.40	64.52
Random Tree	0.16	0.10	0.06	37.5
Random Forest	1.56	0.99	0.57	36.54
Average	**0.80**	**0.48**	**0.32**	**40**

Table 5.18 – Detection rate and false positive rate of four decision trees performed on the *combine-select* subset of the *CSIC* dataset.

Classifiers	Detection Rate (%)			False Positive Rate (%)		
	Full-set	*CFS*	*mRMR*	Full-set	*CFS*	*mRMR*
C4.5	95.06	63.73	**93.89**	5.3	39.1	**6.5**
CART	94.20	63.74	**94.05**	6.6	39.2	**6.4**
Random Tree	88.62	63.75	**92.61**	12.1	39.3	**7.9**
Random Forest	92.87	63.75	**93.89**	8.7	39.3	**7.1**
Average	**92.69**	**63.74**	**93.61**	**8.18**	**39.23**	**6.98**

Table 5.19 – Processing time and improvement for the *combine-select* subset of the *CSIC* dataset.

Classifiers	Processing Time (ms/req.)		Improvement	
	Full-set	*mRMR*	Difference	Improvement(%)
C4.5	1.62	0.70	0.92	56.79
CART	0.57	0.37	0.20	35.09
Random Tree	0.16	0.11	0.05	31.25
Random Forest	1.60	1.07	0.53	33.13
Average	**0.99**	**0.71**	**0.28**	**28.28**

purposely the opposite *GeFS* instance to see how this fact influences the results. In fact, as expected, the results reflect that the *mRMR+CFS* subset gets a detection rate of 96.88% while the *mRMR+mRMR* subset achieves 93.71%. These subsets contain 21 and 11 features respectively.

Table 5.20 – Detection rate and false positive rate of four decision trees performed on the *select-combine* subsets of the *ECML/PKDD* dataset.

Classifiers	Detection Rate (%)		False Positive Rate (%)	
	mRMR+CFS	*mRMR+mRMR*	*mRMR+CFS*	*mRMR+mRMR*
C4.5	**97.09**	94.52	**4.8**	10.2
CART	**96.94**	94.55	**4.9**	10.1
Random Tree	**95.81**	91.45	**5.8**	11.6
Random Forest	**97.56**	94.34	**4.4**	10.3
Average	**96.88**	**93.71**	**4.98**	**10.55**

In the case of the *CSIC* dataset, as the instances of the *GeFS* measure chosen are different, the subsets created are *CFS+CFS* and *mRMR+mRMR*. The first subset has 23 features and the second one, created for comparison, contains 22. As can be seen in Table 5.21, the *CFS+CFS* subset gets a DR of 94.07%, while *mRMR+mRMR* gets 82.74%, due to the selection of the opposite *GeFS* instance.

Table 5.21 – Detection rate and false positive rate of four decision trees performed on the *select-combine* subsets of the *CSIC* dataset.

Classifiers	Detection Rate (%)		False Positive Rate (%)	
	CFS+CFS	*mRMR+mRMR*	*CFS+CFS*	*mRMR+mRMR*
C4.5	**94.5**	84.40	**6.0**	20.4
CART	**94.25**	84.55	**6.3**	20.1
Random Tree	**93.23**	80.43	**7.2**	20.8
Random Forest	**94.29**	81.57	**6.7**	20.3
Average	**94.07**	**82.74**	**6.55**	**20.4**

- *Select-n-gram-combine.* The results for this alternative are shown in Table 5.22 for the *ECML/PKDD* dataset and, in Table 5.23, for the *CSIC* dataset.

  In both cases the subsets created are called *expert+CFS* and *expert+mRMR*. Again, the performance of the last subset is shown for comparison. In the case of the *ECML/PKDD* dataset, the subsets have 45 and 35 features respectively, achieving 97.18% and 96.35% detection rates. In relation to processing time, as in the case of the *select-combine* alternative the features are already selected, it is not

Table 5.22 – Detection rate and false positive rate of four decision trees performed on the *select-n-gram-combine* subsets of the *ECML/PKDD* dataset.

Classifiers	Detection Rate (%)		False Positive Rate (%)	
	Expert+CFS	*Expert+mRMR*	*Expert+CFS*	*Expert+mRMR*
C4.5	**97.63**	96.69	**3.8**	5.3
CART	**97.58**	96.82	**4.0**	5.0
Random Tree	**95.69**	94.74	**5.9**	7.4
Random Forest	**97.84**	97.16	**3.8**	4.8
Average	**97.18**	**96.35**	**4.38**	**5.63**

Table 5.23 – Detection rate and false positive rate of four decision trees performed on the *select-n-gram-combine* subsets of the *CSIC* dataset.

Classifiers	Detection Rate (%)		False Positive Rate (%)	
	Expert+CFS	*Expert+mRMR*	*Expert+CFS*	*Expert+mRMR*
C4.5	**95.25**	94.36	**5.1**	5.9
CART	**94.78**	94.26	**5.6**	6.0
Random Tree	**93.12**	92.45	**7.4**	8.0
Random Forest	**94.47**	93.91	**6.6**	7.1
Average	**94.41**	**93.75**	**6.18**	**6.75**

possible to calculate the percentage of improvement before and after feature selection. Table 5.24 is presented to show the processing times of the four decision trees for the three combination alternatives for the *ECML/PKDD* dataset.

Results for the *CSIC* dataset show the same trend, obtaining better detection results for the *expert+CFS* subset (94.41% vs. 93.75%). The processing time corresponding to the three combination alternatives for the *CSIC* dataset are shown in Table 5.25.

5.5.2 Study of the influence of the training requests in the detection results

In this section, the influence of the number of training requests over the detection results of the system is analyzed. According to the results presented in the previous section, the *select-n-gram-combine* subset is the one that reaches

Table 5.24 – Processing time for the three combination subsets of the *ECML/PKDD* dataset.

Classifiers	Processing Time (ms/req.)		
	Combine-select	Select-combine	Select-n-gram-combine
C4.5	0.60	1.58	3.73
CART	0.22	0.67	1.84
Random Tree	0.10	0.15	0.30
Random Forest	0.99	1.34	2.82
Average	**0.48**	**0.94**	**2.17**

Table 5.25 – Processing time for the three combination subsets of the *CSIC* dataset.

Classifiers	Processing Time (ms/req.)		
	Combine-select	Select-combine	Select-n-gram-combine
C4.5	0.70	1.33	2.63
CART	0.37	0.41	0.84
Random Tree	0.16	0.18	0.24
Random Forest	1.60	0.19	2.24
Average	**0.71**	**0.53**	**1.49**

the highest detection rate. Therefore, this is the subset chosen for studying the variability of the results depending on the number of training requests.

The results of the experiments are given by calculating the mean and the standard deviation of the H runs for every decision tree. The mean DR results for each particular decision tree can be seen in Fig. 5.15, that represents the influence of the number of training requests on the DR for the *ECML/PKDD* dataset.

Similarly, Fig. 5.16 represents the mean DR in relation to the different number of training requests. It corresponds to the classification of the *CSIC* dataset performed by the four decision trees.

Additionally, for each run, the mean of the results produced by the four applied decision trees are calculated. The results of the system for the *ECML/PKDD* dataset are shown in Table 5.26. For each iteration, it contains the number of training requests, and the mean (μ) and standard deviation (σ)

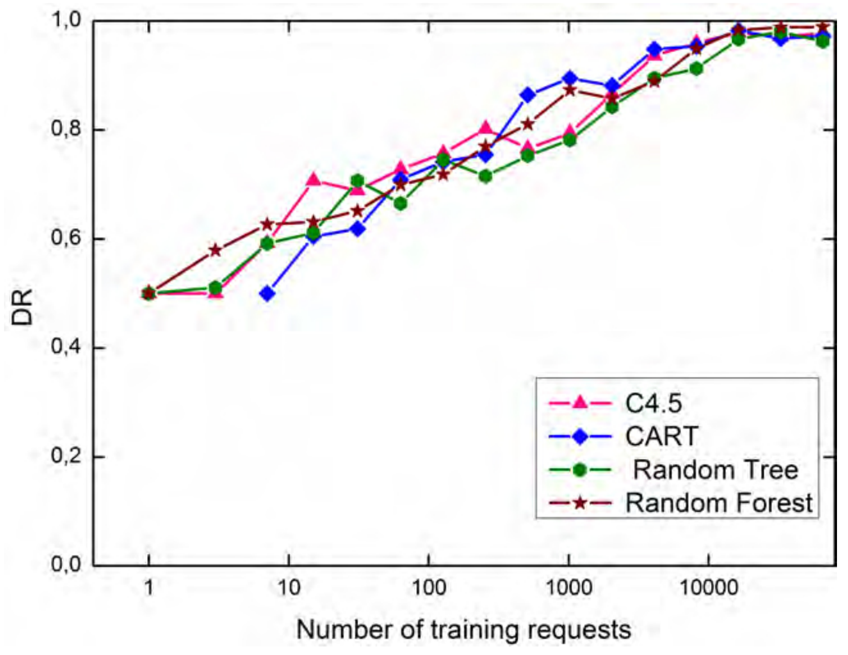


Figure 5.15 – Study of the influence of the number of requests on the detection results of four decision trees. *ECML/PKDD* dataset. The detection rate is plotted vs. the number of training requests.

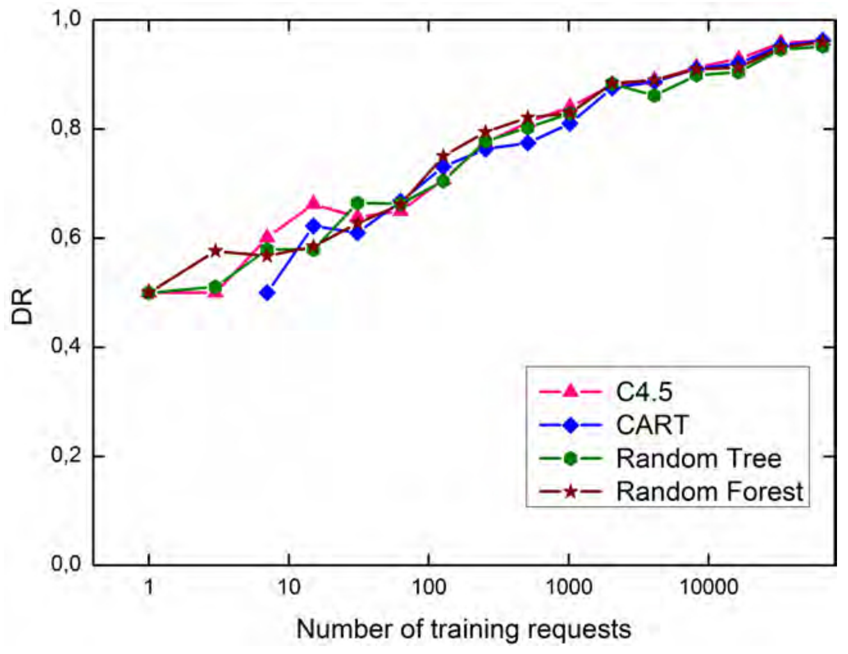


Figure 5.16 – Study of the influence of the number of requests on the detection results of four decision trees. *CSIC* dataset. The detection rate is plotted vs. the number of training requests.

of the DR, FPR and processing time. The equivalent results for the *CSIC* dataset are presented in Table 5.27.

Although other tables have been shown previously in relation to processing times, we decided to take as reference those contained in Table 5.26 and Table 5.27, since those values are obtained considering H runs and are calculated for different amounts of training requests.

Table 5.26 – Detection results and processing time for the *ECML/PKDD* dataset.

Iteration	Num. Training Requests	DR (%)		FPR (%)		Processing Time (ms/request)	
		μ	σ	μ	σ	μ	σ
1	1	50.0	0	50.0	0	1000	0
2	3	53.0	2.32	47.0	2.33	333.3	0.16
3	7	57.8	3.45	42.3	3.14	142.8	0.12
4	15	63.9	2.51	36.2	2.50	66.6	0
5	31	66.7	2.67	33.4	2.28	32.2	0
6	63	70.0	2.80	30.0	2.60	15.9	0.35
7	127	74.0	2.60	26.0	2.52	7.9	0.12
8	255	76.0	2.80	24.0	2.85	3.9	0
9	511	79.8	3.02	20.2	2.72	1.9	0
10	1023	83.6	2.88	16.4	2.56	1.0	0
11	2047	86.2	2.76	13.8	2.26	0.5	0.24
12	4095	91.7	1.10	8.6	1.01	0.2	0.35
13	8191	94.4	0.91	5.6	0.91	0.2	0.24
14	16 383	97.8	0.28	2.2	0.27	0.5	0.30
15	32 767	97.7	0	2.3	0	1.0	0.50

These results are discussed in the next section.

5.6 Discussion

This section discusses several aspects about the results previously presented: first, basic cases are compared to each other. Then, results of basic cases are compared with those of combination alternatives, to check if the proposed combinations outperform the basic cases separately. A comparison of the three alternatives is also shown, to conclude which of them is the most appropriate in a given scenario. Additionally, decision trees are analyzed and a comparison of

Table 5.27 – Detection results and processing time for the *CSIC* dataset.

Iteration	Num. Training Requests	DR (%)		FPR (%)		Processing Time (ms/request)	
		μ	σ	μ	σ	μ	σ
1	1	50.0	0	50.0	0	1000	0
2	3	52.9	2.57	47.1	2.57	333.3	0
3	7	56.2	2.33	43.8	2.93	142.8	0
4	15	61.2	3.04	38.8	2.54	66.6	0
5	31	63.5	2.51	36.7	2.86	32.2	0
6	63	66.1	2.94	34.0	2.53	15.9	0
7	127	72.3	2.88	27.7	1.87	7.9	0
8	255	77.8	2.52	22.3	1.53	3.9	0
9	511	80.3	2.68	19.8	1.67	1.9	0
10	1023	82.7	2.56	17.3	1.75	1.0	0.14
11	2047	88.1	1.29	12.0	1.83	0.5	0.35
12	4095	88.7	0.74	11.3	0.44	0.3	0.35
13	8191	90.8	0.56	9.2	0.35	0.3	0.35
14	16 383	91.7	0.37	8.4	0.26	0.4	0.35
15	32 767	95.1	0.22	4.9	0.13	0.7	0.14

both datasets is given. Furthermore, in this section, the results of the stochastic systems presented in the previous chapter and the results of the ML system are compared.

In this discussion, the number of features, processing time and detection performance are considered a criteria for evaluating the results achieved by each alternative. The mean values of DR, FPR and processing time are used for comparisons, that is, standard deviation is not considered. For simplicity, in some comparisons only the detection rate values are cited, but, in general, the same comments also apply to the false positive rate.

The results for both the *ECML/PKDD* and *CSIC* datasets are included in order to obtain a more solid comparison, i.e., to avoid that conclusions are biased by the dataset used.

5.6.1 Comparison between basic cases

In this section, the results obtained with expert knowledge are compared to those obtained with n-grams. Results are compared in terms of detection results, number of features and processing time.

Table 5.28 presents the results corresponding to the subsets of the *ECML/PKDD* dataset, both before feature selection (full subset) and after (*CFS* or *mRMR* as corresponds). The selected instance of the *GeFS* measure is highlighted in bold letters. The table shows that before feature selection, expert knowledge reaches a higher detection rate and uses a lower number of features than n-grams. Then, expert knowledge is clearly more reliable than n-grams before feature selection.

Table 5.28 – Comparison between basic cases. *ECML/PKDD* dataset.

Subset/ Alternatives	Expert Knowledge			N-gram		
	DR (%)	NF	P. Time	DR (%)	NF	P. Time
Full subset	94.79	30	1.01	92.99	96	5.12
CFS	89.10	2	0.15	**93.47**	**15**	**0.47**
mRMR	**91.22**	**6**	**0.33**	89.45	5	0.33

DR stands for detection rate, NF for number of features and P. Time for processing time. Processing time is measured in ms/request. The selected instance of the *GeFS* measure is highlighted in bold letters.

The results after feature selection are compared according to the selected instance in each case, i.e., *mRMR* for expert knowledge and *CFS* for n-grams. Regarding the detection rate, n-grams get better results than expert knowledge (91.22% vs. 93.47%). Additionally, n-grams use a higher number of features. When the number of features is higher the processing time increases as well. In this case, it is not clear which feature extraction method is better. Thus it depends on which criterion is the most important for the given scenario, either increasing detection results or reducing the resource consumption.

The results for the *CSIC* dataset are analyzed in Table 5.29. In this case, it is clear that expert knowledge is better than n-grams, given that before and after feature selection expert knowledge reaches higher detection rates with a lower number of features. Additionally, the processing time is lower. Given the results, it can be said that, in general, expert knowledge is able to distinguish better between attacks and normal traffic than n-grams.

Table 5.29 – Comparison between basic cases. *CSIC* dataset.

Subset/ Alternatives	Expert Knowledge			N-gram		
	DR (%)	NF	P. Time	DR (%)	NF	P. Time
Full subset	93.99	30	0.88	89.59	114	4.27
CFS	**93.55**	**11**	**0.50**	**82.43**	**12**	**0.62**
mRMR	75.50	14	0.97	77.85	8	0.56

5.6.2 Combination alternatives vs. basic cases

In this section, every combination alternative is compared to both basic cases.

- *Combine-select*. Table 5.30 presents the results for the *combine-select* subset for the *ECML/PKDD* dataset. Results for the basic cases are also shown in order to facilitate the comparison.

  Table 5.30 – Comparison of *combine-select* with basic cases. *ECML/PKDD* dataset.

Subset/ Alternatives	Expert Knowledge			N-gram			Combine-select		
	DR (%)	NF	P. Time	DR (%)	NF	P. Time	DR (%)	NF	P. Time
Full subset	94.79	30	1.01	92.99	96	5.12	96.13	126	0.80
CFS	89.10	2	0.15	**93.47**	**15**	**0.47**	74.25	4	0.30
mRMR	**91.22**	**6**	**0.33**	89.45	5	0.33	**91.38**	**11**	**0.48**

  Before feature selection, *combine-select* achieves better results than both basic cases (96.13% vs. 94.79% and 92.99%) by using a higher number of features (126 features vs. 30 and 96). After feature selection, it gets better results than expert knowledge (91.38% vs. 91.22%), but not than n-grams (91.38% vs. 93.47%). Note that the results are compared according to the selected instance of the *GeFS* measure for each case.

The combination uses more features than expert knowledge (11 vs. 6) and it requires more processing time, while it uses less features than n-grams (11 vs. 15) and spends almost the same processing time than n-grams. The results show that the trend is: the higher the number of features, the better the detection results. This is not necessarily obvious given that, if the features added are highly correlated to the existing ones, they might not provide new information and their results might not improve. This fact indicates that the methods proposed in this thesis extract features that are quite independent from each other and they provide useful information to distinguish normal requests from attacks. In general, the higher the number of requests, the higher the processing time, although not always there is a linear relation between them.

In the case of the *CSIC* dataset, Table 5.31 shows that in this case the combination does not improve the results of both basic subsets before feature selection (92.69% vs. 93.99% and 84.59%), but it does after applying *GeFS* (93.61% vs. 93.55% and 82.43%). For the full subset, the combination significantly improves the performance of n-grams by using more features. The number of features after feature selection is 17 for the combination vs. 11 for expert knowledge and 12 for n-grams. Although the combination uses more features, its processing time is lower than in the case of n-grams.

Table 5.31 – Comparison of *combine-select* with basic cases. *CSIC* dataset.

Subset/ Alternatives	Expert Knowledge			N-gram			Combine-select		
	DR (%)	NF	P. Time	DR (%)	NF	P. Time	DR (%)	NF	P. Time
Full subset	93.99	30	0.88	84.59	114	4.27	92.69	144	0.99
CFS	**93.55**	**11**	**0.50**	**82.43**	**12**	**0.62**	63.74	3	0.16
mRMR	75.50	14	0.97	77.85	8	0.56	**93.61**	**17**	**0.56**

- *Select-combine*. The results corresponding to the *ECML/PKDD* dataset for the two subsets of this alternative are summarized in Table 5.32. Note that in this alternative the full subset (before selection) is not presented, since *select-combine* includes feature selection. Because of that, it is not compared with basic cases before selecting features.

Table 5.32 – Comparison of *select-combine* with basic cases. *ECML/PKDD* dataset.

Subset/ Alternatives	Expert Knowledge			N-gram			Select-combine		
	DR (%)	NF	P. Time	DR (%)	NF	P. Time	DR (%)	NF	P. Time
CFS	89.10	2	0.15	**93.47**	**15**	**0.47**	-	-	-
mRMR	**91.22**	**6**	**0.33**	89.45	5	0.33	-	-	-
mRMR+CFS	-	-	-	-	-	-	**96.88**	**21**	**0.94**
mRMR+mRMR	-	-	-	-	-	-	93.71	11	0.55

The results of the selected *mRMR+CFS* subset notably improve the rates of the two basic cases separately (96.88% vs. 91.22% and 93.47%), while using more features (21 in contrast to 6 and 15). The processing time is higher than the times of basic cases. Besides the number of features, in feature selection, the detection results achieved is also an important factor to be considered. In our case, it can be seen that the criterion employed to choose the *GeFS* instance leads to select the one that reaches higher results, although it neither guarantees the minimum number of features in all cases nor the minimum processing time. However, in the *ECML/PKDD* dataset it is noticeable that even the subset with the non-selected instance (*mRMR+mRMR*) improves the detection of both basic cases, and not necessarily using a higher number of features. This reinforces the idea that creating a combined subset conducts to achieving better detection results. For the *mRMR+mRMR* subset, the processing time is lower than for the *mRMR+CFS* subset, although it is higher than for both basic cases.

The results for the *CSIC* dataset are presented in Table 5.33, together with the basic cases to make easier the analysis of the results. In this case, the selected subset (called *CFS+CFS*) achieves better detection results than both basic cases (94.07% vs. 93.55% and 82.43%). The number of features is also higher (23 vs. 11 and 12), while the processing time is lower (0.53 vs. 0.88 and 0.62).

In conclusion, this combination is able to improve the detection results of the basic cases individually, at the expense of employing a higher number of features. Anyway, in intrusion detection, using 21 and 23 features, as in our case, is not considered a high number of features and it is quickly

Table 5.33 – Comparison of *select-combine* with basic cases. *CSIC* dataset.

Subset/ Alternatives	Expert Knowledge			*N*-gram			Select-combine		
	DR (%)	NF	P. Time	DR (%)	NF	P. Time	DR (%)	NF	P. Time
CFS	**93.55**	**11**	**0.88**	**82.43**	**12**	**0.62**	-	-	-
mRMR	75.50	14	0.97	77.85	8	0.56	-	-	-
CFS+CFS	-	-	-	-	-	-	**94.07**	**23**	**0.53**
mRMR+mRMR	-	-	-	-	-	-	82.74	22	1.06

processable by most detection systems. Although the number of features is higher, in this case the processing time of the combination outperforms both basic cases.

- *Select-n-gram-combine.* To facilitate the discussion, a summary for the results of this alternative for the *ECML/PKDD* dataset is presented in Table 5.34.

Table 5.34 – Comparison of *select-n-gram-combine* with basic cases. *ECML/PKDD* dataset.

Subset/ Alternatives	Expert Knowledge			*N*-gram			Select-*n*-gram-combine		
	DR (%)	NF	P. Time	DR (%)	NF	P. Time	DR (%)	NF	P. Time
CFS	89.10	2	0.15	**93.47**	**15**	**0.47**	-	-	-
mRMR	**91.22**	**6**	**0.33**	89.45	5	0.33	-	-	-
Expert+CFS	-	-	-	-	-	-	**97.18**	**45**	**2.17**
Expert+mRMR	-	-	-	-	-	-	96.35	35	1.4

The selected subset of this combination alternative (*expert+CFS*) improves the results of both basic cases separately (97.18% vs. 91.22% and 93.47%) by using a higher number of features (45 vs. 6 and 15). In this case, the increment in the number of features is translated into a rise in the processing time. The *expert+mRMR* subset, although not being optimal in feature selection, improves the detection of basic cases with a higher number of features. This fact shows that the combination can improve the detection results, and that the criterion used for the *GeFS* measure leads to the selection of the instance that reaches higher detection results.

Regarding the *CSIC* dataset, its results can be studied by looking at Table 5.35. In this case, the behavior is the same than in the previous dataset: the combination improves the detection results of the two basic cases individually, by selecting a higher number of features and increasing the processing time.

Table 5.35 – Comparison of *select-n-gram-combine* with basic cases. *CSIC* dataset.

Subset/ Alternatives	Expert Knowledge			*N*-gram			Select-*n*-gram-combine		
	DR (%)	NF	P. Time	DR (%)	NF	P. Time	DR (%)	NF	P. Time
CFS	**93.55**	**11**	**0.50**	**82.43**	**12**	**0.62**	-	-	-
mRMR	75.50	14	0.97	77.85	8	0.56	-	-	-
Expert+CFS	-	-	-	-	-	-	**94.41**	**42**	**1.49**
Expert+mRMR	-	-	-	-	-	-	93.75	38	1.25

In summary, except in the *combine-select* case of the *ECML/PKDD* dataset, the combination alternatives reach higher results than basic cases separately. In all analyzed cases, the detection results are better when the number of features is higher. And when this happens, generally, the processing time is also higher. However, it was shown that this implication is not necessarily true in all cases and that their correlation is not necessarily linear. The subset with lower processing time is expert knowledge. As the conclusions drawn above hold true for both *CSIC* and *ECML/PKDD* datasets, they lead to think that this behavior is independent of the dataset used. The improvement of detection results is usually not in line to reducing the number of features, i.e., the more one objective is fulfilled, the lest fulfilled is the other one. Because of that, the best solution depends on the scenario to be protected and on which factors are more critical for it. When the detection results are important, then the combination alternatives are recommended. In environments with restricted resources, combination alternatives that do not increase the processing time could be used, that is, *combine-select* or *select-combine* might be potentially employed. Another alternative for restricted resource scenarios is applying basic cases, that could result more appropriate due to their reduced usage of features.

5.6.3 Comparison between combination alternatives

In this subsection, the combination alternatives are compared to each other. The results for the *ECML/PKDD* dataset of the three alternatives are summarized in Table 5.36 for facilitating the comparison. Recall that in the case of *combine-select* alternative, results before and after feature selection have been calculated. However, for the *select-combine* and *select-n-gram-combine* alternatives, results can only be obtained including feature selection. In order to properly compare these three alternatives, the results shown for the *combine-select* alternative are exclusively calculated after feature selection.

Table 5.36 – Detection results, number of features and processing time for the three combination alternatives. *ECML/PKDD* dataset.

Subset	DR (%)	NF (%)	P. Time (ms/req.)
Combine-select	91.38	11	0.48
Select-combine	96.88	21	0.94
Select-n-gram-combine	97.18	45	2.17

Table 5.36 shows that the best detection results are reached by the *select-n-gram-combine* alternative. However, it is also the alternative using a higher number of features and requiring a higher processing time. Again, the trend is that the detection results are better as long as the number of features increases. This is not necessarily obvious as it depends on the information provided by each feature. The processing time rises as long as the number of features does.

For the *CSIC* dataset, the summary can be found in Table 5.37. The behavior regarding the detection results and number of features is the same as for the *ECML/PKDD* dataset: the best detection for the *select-n-gram-combine* alternative, that uses more features. However, the processing time does not necessarily increase as long as the number of features grows, which has positive implications for the performance of the system.

Table 5.37 – Detection results, number of features and processing time for the three combination alternatives. *CSIC* dataset.

Subset	DR (%)	NF (%)	P. Time (ms/req.)
Combine-select	93.61	17	0.71
Select-combine	94.07	23	0.53
Select-n-gram-combine	94.41	42	1.49

5.6.4 Comparison between decision trees

In this section, the results of the different classifiers applied to solve the current intrusion detection problem, over the particular studied datasets, are analyzed.

From the results in Sec. 5.5, it seems that, in general, the C4.5, CART and Random Forest obtain similar results, while the performance of Random Tree is slightly lower.

If the analysis is performed per dataset, the trend is that for *ECML/PKDD*, Random Forest and CART reach slightly better results, while for the *CSIC* dataset, CART and C4.5 are faintly more effective.

Regarding the analysis of processing time, in general Random Tree is the most efficient one (it reaches the lowest processing time), while Random Forest and C4.5 require more milliseconds for processing each request. This conclusions are aligned with the nature of each algorithm.

In average, there are no big differences between the classifiers. This tends to satisfy the No-Free-Lunch Theorem, that states that there are no a priori distinctions between learning algorithms [Wolpert, 1996] and that "any two optimization algorithms are equivalent when their performance is averaged across all possible problems" [Wolpert and Macready, 2005].

5.6.5 Comparison between datasets

In this section, results of both datasets are compared. Regarding expert knowledge, features have different relationships among them: for the *CSIC* dataset they are linear, while for the *ECML/PKDD* dataset they are not linear.

Differently, in the *n-gram* subset, both datasets have linear relationships. In this case the detection results for the *CSIC* dataset are not as high as those for the *ECML/PKDD* dataset.

About the combinations, the detection results for the *ECML/PKDD* are slightly higher than for the *CSIC* dataset. It could be said then that the *CSIC* dataset is more challenging for the detection algorithms than *ECML/PKDD*. Anyway, both datasets follow a similar trend.

For both basic cases and *combine-select*, the processing times of the *ECML/PKDD* dataset are lightly lower than those corresponding to *CSIC*. For *select-combine* and *select-n-gram-combine* the situation is the opposite.

5.6.6 General comparison

In this section, several graphs are shown as a summary for facilitating the comparison of the different subsets in terms of the detection results and the number of features. Figure 5.17 represents the DR and FPR of all the subsets for the *ECML/PKDD* dataset. In the figure, the size of the circles is used to represent the number of features, being the coordinates of the alternative the point in the center of the circles. The best solution would be the one closer to the (0,1) point with the minimum circle size, that is, better detection results and lower number of features. Figure 5.18 shows in detail the left-upper area of Fig. 5.17, in order to facilitate the observation of the results.

As can be seen in Fig. 5.17 and Fig. 5.18, the best results are reached by the three combination alternatives, that are closer to the (0,1) point and some of them do not use a high number of features.

The summary for the *CSIC* dataset is represented in Fig. 5.19. Figure 5.20 and Fig. 5.21 show the previous figure in more detail for better appreciation (the last figure augments the dotted squared area in Fig. 5.20). These graphs show that the best solutions are again the combinations, and some expert knowledge options.

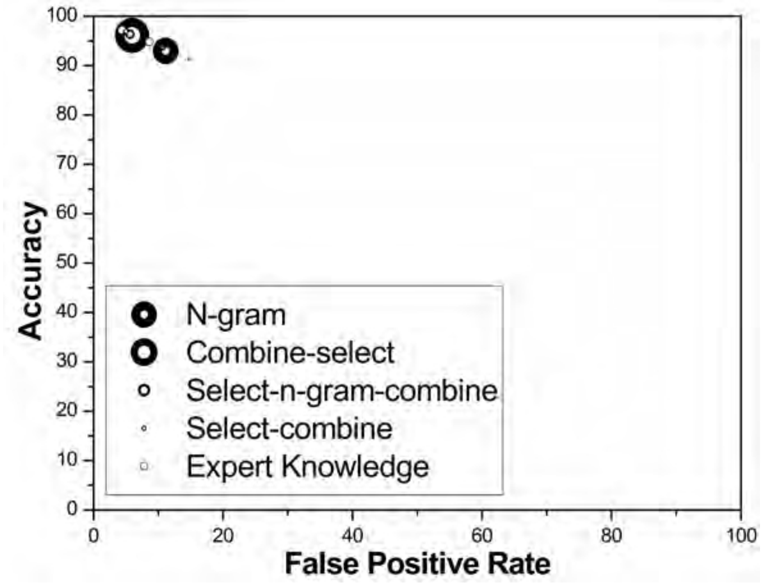


Figure 5.17 – DR and FPR for all the subsets of the *ECML/PKDD* dataset. The size of the symbols represents the number of features.

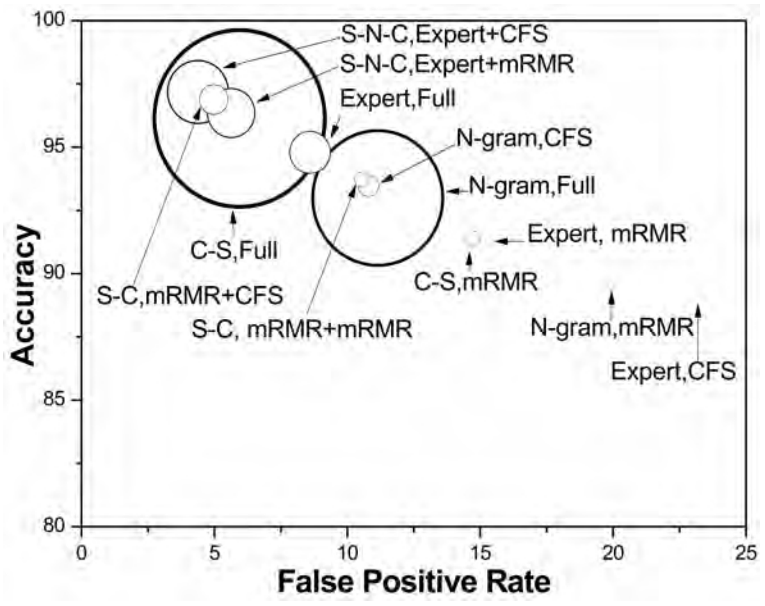


Figure 5.18 – Detail of DR and FPR for the subsets of the *ECML/PKDD* dataset. C-S stands for *combine-select*, S-C for *select-combine* and S-N-C for *select-n-gram-combine*.

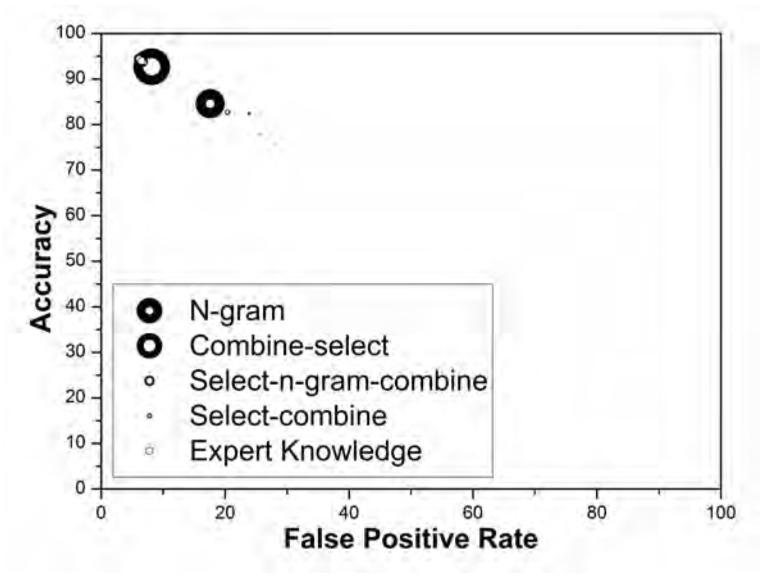


Figure 5.19 – DR and FPR for all the subsets of the *CSIC* dataset. The size of the symbols represents the number of features.

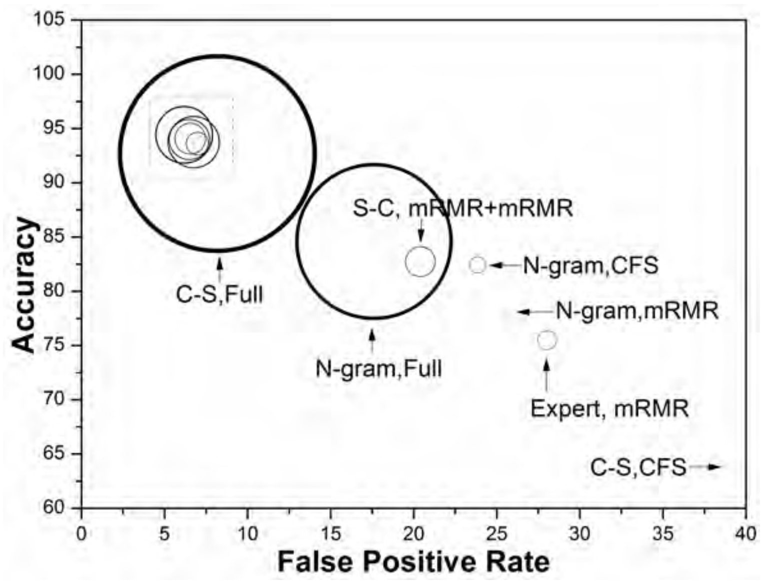


Figure 5.20 – Detail of Fig. 5.19 regarding DR and FPR for the subsets for the *CSIC* dataset. C-S stands for *combine-select*, S-C for *select-combine* and S-N-C for *select-n-gram-combine*.

5.6.7 Influence of the number of training requests

Table 5.26 reveals that for the *ECML/PKDD* dataset, the average of the four decision trees can obtain a mean DR of 97.8% and a mean FPR of 2.2%, by

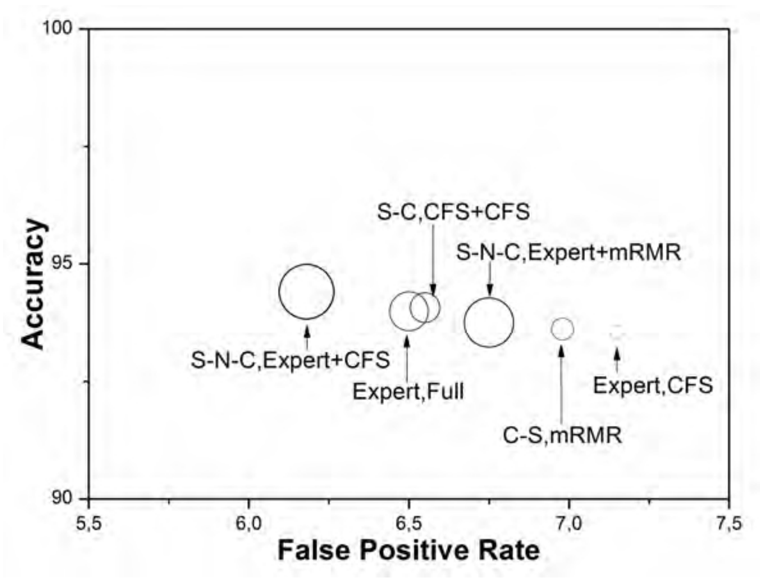


Figure 5.21 – Detail of Fig. 5.20 regarding DR and FPR for the subsets for the *CSIC* dataset. C-S stands for *combine-select*, S-C for *select-combine* and S-N-C for *select-n-gram-combine*.

using 16 383 training requests. The standard deviation of DR is 0.28% and that of FPR is 0.27%. Regarding processing time, the mean value is 0.5 ms/request with $\sigma = 0.30\%$.

The table shows that the general trend followed by the results when the number of training requests increments is the following: DR augments progressively while FPR decreases. For both measures, σ fluctuates until 511 requests, where it decreases until it reaches the 0 value. The processing time keeps falling until 8191 requests, where it starts growing again.

This trend can be also seen graphically in Fig. 5.22 for the *ECML/PKDD* dataset. It shows the mean DR and FPR (regarding the H runs) for the mean of the four employed decision trees.

Several runs are carried out to guarantee that the behavior of the system is captured. Observing the figure it can be seen how, at the beginning, the algorithms do not have enough knowledge and then, both mean DR and FPR are near 50%. This means that half of the attacks are detected and half of the alarms are false alarms. This behavior is similar to a random classification system with the same probability of classifying requests into each class (normal or anomalous). As long as the system is trained with more requests DR

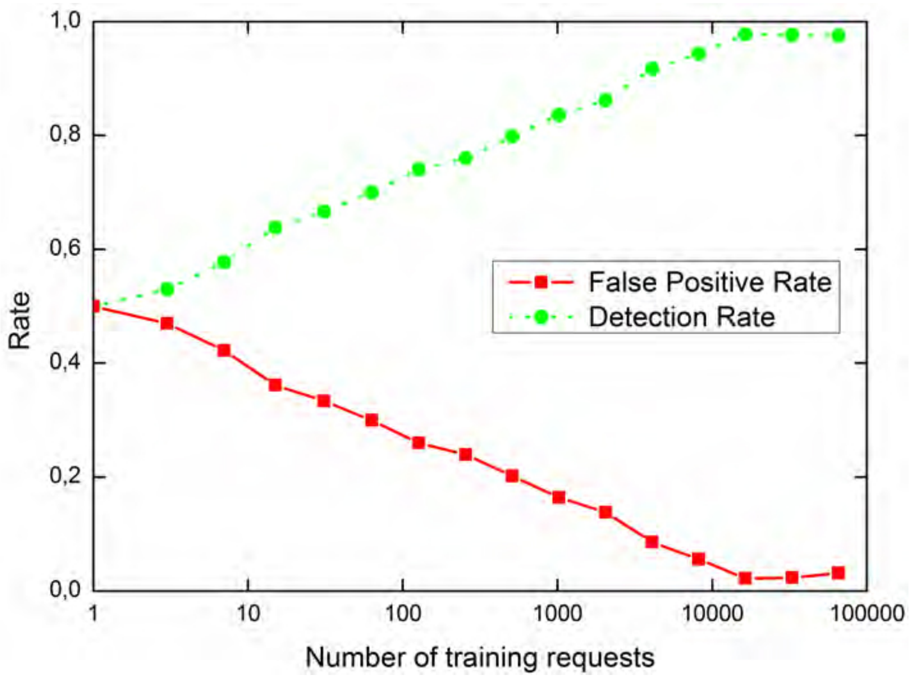


Figure 5.22 – Study of the influence of the number of requests on the detection results of the mean of four decision trees. *ECML/PKDD* dataset.

progressively increases while FPR does the opposite, achieving results more aligned to our objective.

For the *CSIC* dataset, Table 5.27 shows that mean results rise up to 95.1% DR (with $\sigma = 0.22\%$) and 4.9% FPR ($\sigma = 0.13\%$) when training the system with 32 767 requests. In this case, the mean processing time is 0.7 ms/request and the standard deviation is 0.14%.

The influence of the increase in the number of training requests in the results of the WAF is summarized next: the mean DR gradually rises, in contrast to the mean FPR, that decreases progressively. This is represented graphically in Fig. 5.23. Analyzing the deviation of these measures in Table 5.27, it can be seen that it fluctuates until it stabilizes and takes a decreasing trend. The required processing time is more and more reduced until 8191 requests, where it starts growing. The standard deviation for the processing time keeps 0 until 511 requests.

Since in both datasets σ takes small values, when this document talks about the results, it refers to the mean value. At this point it should be mentioned

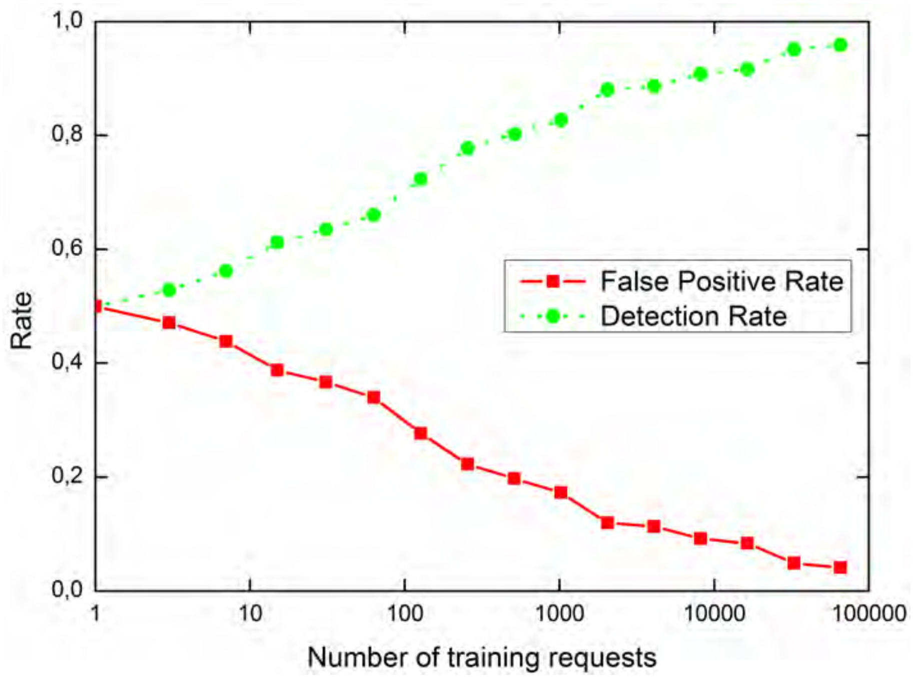


Figure 5.23 – Study of the influence of the number of requests on the detection results of the mean of four decision trees. *CSIC* dataset.

that the fact that the mean FPR is higher than 0.01 (threshold frequently used in the IDS field) is not a problem when it is considered that, nowadays, the output of current IDSs is processed by Security Information and Event Management (SIEM) systems, that correlate data and discard false positives.

The results shown correspond to $H = 10$ runs. Given the results obtained for the mean and standard deviation, it was considered that performing H runs was representative enough for our purposes. It is expected that the standard deviation would be smaller when more experiments are conducted.

The processing times in these tables are taken as reference since they are obtained out of H runs and they are calculated for the different values of training requests.

Note that Fig. 5.15 and Fig. 5.16 previously presented, that show the results for each of the decision trees individually, present a similar behavior to Fig. 5.22 and Fig. 5.23 respectively, that show the results of the four decision trees in average.

5.6.8 Comparison with stochastic systems

The aim of this section is comparing the detection algorithms presented in this chapter and in Chapter 4, that is, comparing the behavior of statistical techniques, Markov chains and decision trees.

In order to compare different systems, a common dataset is needed. Since the experiments of all the studied systems are conducted using the *CSIC* dataset, the comparison is possible.

The systems are compared in regard to the following criteria: detection results, processing time and number of training requests needed. It should be considered that the implementations of the systems have some differences that could affect the results. Furthermore, the selection of the training requests is a random process, therefore, it could also influence the results and comparisons. It is important to mention that comparisons and conclusions are circumscribed to the *CSIC* dataset. Conclusions might be different when a different dataset is analyzed.

5.6.8.1 Comparison of the characteristics of the systems

This section shows a comparison between some characteristics of the systems that might influence their results:

- Feature extraction for both stochastic systems is done manually (expert knowledge). However, in the ML system feature extraction methods that combine both manual and automatic techniques are proposed.

- Given the characteristics of each algorithm, stochastic systems are trained with only normal traffic and decision trees with both normal and anomalous traffic. This fact is reflected in Fig. 4.17 and Fig. 5.23. As a consequence, stochastic systems deny everything when they are trained with few requests, i.e., DR is high but FPR too. This behavior reflects the "denying everything unless explicitly allowed" approach of the system. Differently, decision trees are trained with both normal and anomalous requests. Then, when decision trees are trained with few requests, the system behaves almost randomly. In other words, DR and FPR are 50%.

- In the stochastic approach the features are constructed at the token level. However, the machine learning system works at both the token and request levels.

- Stochastic systems distinguish three types of characters: letters, digits and non-alphanumeric characters. Decision trees distinguish whether the non-alphanumeric characters have a meaning in certain programming languages or not. Moreover, the character distribution of headers is considered in stochastic systems but not in the ML one.

- In the case of the statistical algorithms, the length limits are determined by the minimum and the maximum values seen in the training data. However, in the Markov chains the lower limit is set to 0 and the upper limit is the parameter of the WAF for the Gaussian distribution.

- The ML system applies feature selection. However, it was not considered necessary to apply it in stochastic systems because the number of extracted features is low.

5.6.8.2 Comparison of the results

In this subsection, the proposed detection systems are compared according to several criteria: detection results, number of needed training requests and processing time. For the comparison, the mean values are used and standard deviation values have been omitted. Recall that small differences in the results should not be considered very decisive, due to the previously mentioned differences in the design of the systems and the influence of random aspects of the systems, that may affect their results. Moreover, the fact that training requests are chosen randomly also has an influence.

- **Detection results**. The detection results of the three systems are presented in Table 5.38, in the 'DR' and 'FPR' columns. Recall that comparisons are performed using the *CSIC* dataset. When comparing the results, it can be observed that, for the number of training requests used, stochastic algorithms perform better than decision trees (higher DR and lower FPR). However, between statistical techniques and Markov chains the difference is not much.

Table 5.38 – Number of training requests, processing time and results achieved by each detection algorithm. DR: Detection Rate, FPR: False Positive Rate, NTrR: Number of Training Requests, NTR: Number of Test Requests, PT: Processing Time (ms/request), MPT: Minimum Processing Time (ms/request). *CSIC* dataset.

Technique	DR(%)	FPR(%)	NTrR	NTR	PT	MPT
Statistical	99.4	0.9	16 383	2000	0.63	0.59
Markov chain	98.1	1.0	32 767	2000	7.9	7.9
Decision Trees	95.1	4.9	32 767	2000	0.7	0.3

- **Number of training requests**. Table 5.38 reflects that statistical techniques use half of the requests required by Markov chains in order to get almost the same detection results. Whereas decision trees are trained with the same number of training requests than Markov chains and detection results are a bit lower.

  Therefore, statistical techniques are those that require the lowest number of training request to achieve 0.9% FPR. Markov chains and decision trees are less recommendable in environments where obtaining traffic is difficult or where the amount of traffic is limited.

  As mentioned, in all cases, the systems are tested with 1000 normal plus 1000 abnormal requests.

- **Processing Time**. Since the processing time of the system makes reference to the milliseconds consumed to process a single request, the lower the processing time, the more efficient the system is. Processing time is shown in Table 5.38 in two columns. On the one hand, column ‘PT’ shows the processing time corresponding to the number of requests used to train the system, that is represented in the ‘NTrR’ column. On the other hand, column ‘MPT’ refers to the minimum processing time that the algorithm reached, regardless of the number of training requests.

  Results reveal that the slowest algorithm is the one that uses Markov chains. Column ‘MPT’ shows that decision trees can reach the fastest processing time, hence, they are recommended when the processing time of the system is an important factor in the detection system.

In summary, it can be stated that stochastic systems reach the best detection results and decision trees achieve the best processing time. The most adequate technique for each target scenario depends on which of the previous aspects has a higher importance in each particular case.

5.7 Conclusions

The conclusions drawn in this chapter are summarized next:

- Decision trees can be successfully applied to the classification of web traffic. They are effective in the classification of this type of traffic besides in network traffic. Four decision trees have been used for the experiments: C4.5, CART, Random Tree and Random Forest.
- The decision trees employed for detection reach high detection results. For the *ECML/PKDD* dataset, the average results of the four decision trees reach 97.8% DR and 2.2% FPR. These results correspond to the mean of the results for 16 383 training requests and for the $H = 10$ runs of the experiments using the *select-n-gram-combine* subset. For the *CSIC* dataset, the results are 95.1% DR and 4.9% FPR in average for the same subset with 32 767 training requests and $H = 10$. In the experiments, decision trees are able to detect the different attacks included in both the *CSIC* and *ECML/PKDD* datasets, even zero-day attacks.
- The algorithms are high-speed. For the *ECML/PKDD* dataset, when decision trees are trained with 16 383 requests, the processing time is 0.5 ms/request, with 0.3% standard deviation. However, the algorithms can process requests up to 0.2 ms/request and 0.24% standard deviation, when trained with 8191 requests. In the case of the *CSIC* dataset, the processing time is 0.7 ms/request for 32 767 training requests, with 0.14% standard deviation. The algorithms can reach higher speeds when trained with 4095 and 8191 requests. In those cases the processing time is 0.3 ms/request, with 0.35% standard deviation.
- The *GeFS* measure successfully selects features in web traffic (in addition to network traffic). This measure reduces the number of irrelevant and

redundant features and decreases the processing time. Besides that, it is able to improve the detection results in two cases: firstly, in the case of *combine-select* subset of the *CSIC* dataset, where the reduction was 88.2% (from 144 to 17 features) and the DR improvement of 0.92% (from 92.69% to 93.61%). Secondly, in the case of n-grams of the *ECML/PKDD* dataset, that reaches an improvement of 0.48% in DR (from 92.99% to 93.47%) with a reduction of 84.38% in the number of features (from 96 to 15).

- New feature extraction methods are proposed. They combine expert knowledge and n-gram features. Concretely, three combination alternatives are proposed:
  - *Combine-select*. The first alternative mixes all the features extracted by n-grams and expert knowledge and applies feature selection afterwards.
  - *Select-combine*. In contrast, this alternative mixes the features already selected from expert knowledge and n-grams.
  - *Select-n-gram-combine*. This alternative is a variation of the second one. It combines expert knowledge features (not selected) with the selected ones from n-grams.

  The resource consumption of the proposed extraction methods is reduced by the application of feature selection. However, for both datasets, the three combination alternatives improve the detection results of the methods individually, satisfying the proposed objectives.

  Regarding the number of features, the combination alternatives use a higher number of features than individual techniques. In general, when the number of features grows higher, the processing time also does. However, it is not always the case (like for example, with the case of the *select-combine* alternative for the *CSIC* dataset). From the three combination alternatives, the *select-n-gram-combine* is the one that uses more features (45 features for the *ECML/PKDD* dataset and 42 for the *CSIC* dataset) and also the one that reaches the highest results. *Combine-select* is the option that uses the lowest number of features (11 for *ECML/PKDD* and 17 for the *CSIC* dataset) and the one that achieves lower mean detection results (91.38% and 93.61% for the *ECML/PKDD* and *CSIC* datasets respectively).

In general, in intrusion detection, the number of features used in this thesis is considered low and, considering the capacity of modern systems, it is acceptable. However, in scenarios with high resource limitations, alternatives using a lower number of features (for example *combine-select*) could result more suitable. These conclusions apply to both datasets studied in this thesis.

- The study of how the number of training requests influences the performance of the system reveals a funnel-shape trend of the system (the higher the number of training requests, the best detection results). For the *ECML/PKDD* dataset, 16 383 requests are enough to achieve a mean FPR of 2.2%. For the *CSIC* dataset 32 767 requests are used to achieve a mean FPR of 4.9%.

- The comparison of statistical techniques, Markov chains and decision trees concludes that stochastic systems reach the best detection results and decision trees can potentially achieve the best processing time. The statistical system is the one that needs the least training request for reaching those results.

Chapter 6

Conclusions, contributions and future work

"Tell me and I forget, teach me and I may remember, involve me and I learn."
— Benjamin Franklin

This chapter firstly shows a brief summary of the present thesis. Then, the conclusions extracted from the research are described. The contributions of the thesis, as well as the publications derived, are shown. Finally, future research lines are drawn.

6.1 Summary

Web applications take more and more part in our daily lives. They are becoming increasingly popular and complex in all sorts of environments, ranging from e-commerce applications to banking. This fact makes web applications very attractive for attackers, who intend to exploit web vulnerabilities. Web applications are threatened, therefore, it is necessary to protect them. In order to detect web-specific attacks, detection mechanisms need to be placed at the application layer. This is the objective of WAFs: analyzing HTTP traffic with the aim of detecting web intrusions. These systems are a particular case of IDSs, with the particularity of being specialized on the analysis of web traffic. One of the benefits of IDSs and WAFs is that they protect a target web application without the necessity of modifying its source code.

The main objective of this thesis is developing intrusion detection systems that are able to accurately detect web attacks with a low resource consumption, high speed and a simple design. In order to achieve this goal, various techniques have been used for detection: stochastic-based techniques and machine learning. The proposed systems follow an anomaly-based approach. Regarding stochastic techniques, two methods have been applied: statistical-based algorithms and Markov chains. Regarding machine learning, four decision trees have been used to detect web attacks, namely C4.5, CART, Random Tree and Random Forest.

Additional aspects of the web attack detection have been addressed in this thesis. On the one hand, it studies the influence that the number of requests used in the training phase produces over the detection capacity of the system. On the other hand, it carries out a study about which features are more efficient and effective for web intrusion detection. For that purpose, firstly, three feature extraction methods have been studied: expert knowledge, n-grams and a combination of the two previous options. They have been analyzed in order to determine which one leads to the best detection results and the lowest resource consumption. Secondly, the extracted features have been selected by means of the GeFS measure.

The proposed detection systems have been experimentally tested. For that, HTTP traffic is needed. Counting with appropriate datasets for training and testing detection methods is critical. However, in the web intrusion detection field, gathering labeled and proper traffic faces several difficulties. The main problem identified is the scarcity of labeled HTTP datasets [Sommer and Paxson, 2010], [Tavallaee et al., 2010]. In this thesis, the *ECML/PKDD* dataset, generated for the ECML/PKDD Discovery Challenge, has been used. However, the disadvantage of this dataset is that most parts of the requests are anonymized, which complicates the thorough evaluation of the detection algorithm's performance. Therefore, a common and public dataset would be necessary to adequately evaluate the systems. It would also allow the comparison of different schemes and techniques. This has been the motivation for generating the *CSIC* dataset. It is a new and publicly available HTTP dataset that is being used by the web intrusion detection community to evaluate their detection systems. It consists of labeled normal and anomalous requests, including multiple modern attacks. Furthermore, it is not anonymized.

After this summary, the conclusions obtained during the development of this work are given.

6.2 Conclusions

The most relevant conclusions derived from this thesis are:

- **Stochastic and machine learning algorithms can be successfully used in anomaly-based web attack detection.**

  They are able to distinguish between normal and anomalous traffic with low time and resource consumption. Both approaches have been able to detect zero-day attacks.

- **The goal of building high-detection WAFs has been achieved.**

  Results show that the best detection results are reached by the statistical system.

  The performance of the systems is measured as the mean of the detection rate and the false positive rate. The values obtained from the experiments are the following:

  – In relation to the statistical based system, it is able to reach a detection rate of 99.4% and a false positive rate of 0.9%.

  – The Markovian system achieves a detection rate of 98.1% and 1% false positive rate. These results have been reached with the following parameter values for the Markov chain: $\tau = 50$, $\epsilon = 10^{-15}$ and $p = 0.99$.

  – Regarding ML techniques, the results have been calculated as the average of the detection rate of four decision trees (C4.5, CART, Random Tree and Random Forest). Results have been shown for two datasets: in the case of the *CSIC* dataset the detection rate rises up to 95.1% when the false positive rate is 4.9%. For the *ECML/PKDD* dataset, the DR reaches 97.8% with 2.2% FPR. Note that FPRs higher than 0.01 are not problematic, given that SIEM systems analyze the output of IDSs and correlate data, being able to reduce the amount of false positives.

- **The influence of the number of training requests on the detection results has been studied.**

  For that, $M = 15$ experiments that use an increasing number of training requests, from 1 to 32 767 have been performed. They have been run $H = 10$ times, choosing different samples of training requests.

  Contrarily to what would be expected, the behavior of the systems is not always linear, consequently, using more training requests does not always imply better results.

  Experiments have revealed the following:

  – In the case of the statistical WAF, when using 16 383 requests to train the system, a false positive rate of 0.9% is reached. When the training is done with higher amounts of requests, it is possible to progressively reduce the false positive rate, until reaching a FPR of 0.4%. The detection rate remains almost invariable: from 99.4% to 99.3%. This is achieved with 32 767 training requests.

  – In the case of Markov chains, 32 767 requests are necessary to obtain 1% FPR. It is noticeable that when 8191 requests are employed, the FPR is already 1.7%.

  – In relation to ML, the experiments show that with 32 767 requests from the *CSIC* dataset, the system reaches a FPR of 4.9%. The rate decreases to 2.2% in the case of the *ECML/PKDD* dataset when 16 383 requests are employed.

  In summary, the statistical method and the ML system (in the case of the *ECML/PKDD* dataset) are the ones that need the lowest number of requests for training the system (16 383), while not decrementing its detection capacity. Markov chains and the ML system for the *CSIC* dataset use a double amount of requests.

- **The goal of building high-speed detection systems has been achieved.**

  The experiments reveal the following processing time, depending on the detection technique used:

  - The statistical algorithm is able to process requests up to a rate of 0.59 ms/request (i.e., processing a single request takes 0.00059 s).
  - The processing time is 7.9 ms/request for Markov chains.
  - In the case of ML, a mean of the processing time of the four decision trees is taken. The experiments show that ML can reach a processing time of 0.3 ms/request for the *CSIC* dataset. In the case of the *ECML/PKDD* dataset, the processing time decreases to 0.2 ms/request. Note that as the mean of the algorithms is taken, using a specific algorithm results could even improve. In particular, Random Tree is the fastest decision tree (it can be 7 times faster than C4.5).

  All processing time measurements have been obtained with an Intel core i7 CPU at 2.40 GHz and 8GB RAM, SO Windows 8, 64 bits.

  In conclusion, from the detection techniques studied, ML algorithms are those that get the minimum processing time. Thus, from the experiments it has been concluded that even in the case where ML requires a higher number of training requests, it is faster.

- **The proposed feature extraction methods, that combine expert knowledge and *n*-grams, have improved the detection results of both techniques separately.**

  Furthermore, feature selection has been applied to reduce irrelevant and redundant features. Using a low number of features results in low resource consumption.

  Comparing the results of the three combination alternatives proposed, namely *combine-select*, *select-combine* and *select-n-gram-combine*, it can be concluded that the first one is the option that reduces the most the number of features, but it also gets the lowest results. Contrarily, *select-n-gram-combine* reaches the best detection results but it needs the highest number of features. *Select-combine* is an intermediate case between the other two alternatives.

In general, a lower number of features implies lower processing times. However, it was shown that in some cases, like for the *select-combine* subset of the *CSIC* dataset, the combination was even able to reduce the processing time.

- **The systems consume low resources. Experiments have shown that not only for network traffic, but also for web traffic, the GeFS measure is able to reduce the number of redundant and irrelevant features, while keeping the detection results.**

  The resource consumption is measured according to the number of features. Besides reducing the number of features, GeFS can even make the results improve, like in the case of n-grams and *combine-select*.

  The experiments have been carried out using two datasets (*CSIC* and *ECML/PKDD*) for expert knowledge, n-grams and the three combination alternatives. Expert knowledge and n-grams are called basic cases and the rest, combination ones. The study about the type of relationship between features has revealed that in most of the cases there are linear relationships between the features, considering basic cases and combination. When there are linear relations, the CFS has been used for reducing the number of features. The experiments show that, in most of the cases, for the two datasets studied, there are more linear relations between the features than non-linear ones.

- **A new publicly available and labeled dataset, the *CSIC* dataset, has been created.**

  The dataset contains exclusively HTTP traffic, and its requests are labeled as normal or anomalous. The fact that the dataset is labeled makes possible to evaluate the detection abilities of the web detection system. Specifically, the dataset contains around 36 000 normal requests and 25 000 anomalous ones. Furthermore, the *CSIC* dataset includes modern web attacks such as SQL injection, buffer overflow, XSS, server side include and so on, satisfying the necessity of evaluating the behavior of the systems towards modern attacks. Considering the quick speed of the evolution of web attacks nowadays, it is very important that the protection mechanisms are ready to respond adequately to those threats. Other advantage of the dataset is that it includes realistic values and that it is not anonymized.

This dataset is more challenging than the *ECML/PKDD* dataset for the evaluation of decision tree algorithms.

Given that the dataset is publicly available, it provides a common framework to the scientific community for evaluating WAFs, facilitating the comparison of different web intrusion systems. In fact, the dataset is currently being used by the scientific community to evaluate their web attack detection systems [Kozik et al., 2014a], [Kozik et al., 2014b]. Even, versions with other formats of the dataset have been created by other researchers [Scully, 2015].

In summary, it can be seen that the objectives established in Sec. 1.4 have been satisfied.

- **The behavior of the proposed detection techniques (statistical methods, Markov chains and decision trees) has been compared using the *CSIC* dataset to study which of them is the most recommendable in each scenario.**

  This study reveals that, depending on the scenario and on the pursued goal, the recommendable system should be chosen as follows: when the goal is to achieve high detection results, then the statistical-based system is the most appropriate one. This is also the most suitable option when a low amount of requests is available. However, if the primordial interest is a low processing time of the system, then decision trees are optimal.

6.3 Contributions

In this section, the main contributions of this thesis are presented:

1. **Two new high-speed, high-detection, low resource consumption and simple designed stochastic-based anomaly web application firewalls have been developed.**

   One of the systems applies statistical techniques for the detection and the other is based on Markov chains.

   Several relevant aspects of the algorithms to be highlighted are:

   - In general, the statistical-based anomaly WAFs in the literature make use of several models to decide about the normality/abnormality of the requests, like [Kruegel et al., 2005b], [Criscione and Zanero, 2009]. These models range from low to high complexity. In order to calculate the total anomaly rate of a request, these systems need to evaluate every model. Contrarily, the systems proposed in this thesis are designed to consider each model as autonomous. That is, the models contribute to the final decision but each one also has decision autonomy, not being merely another part of the general abnormality decision. This implies that, in some cases, the approach proposed does not need to evaluate all models before making a decision about the abnormality of the request. This happens when a model decides that the request is not normal. In that case, it is not necessary to continue with the detection process. As soon as the request does not satisfy any of the model requirements for normality, the request is tagged as anomalous. The request is only marked as normal when it satisfies the requirements of all models. This design makes the algorithm simpler, faster and low resource consumer than when the whole detection process needs to be completed in all cases.
   - Regarding the number of models, the proposed stochastic approach makes use of two simple models, while the detection performance of the system is not deflated. Therefore, this approach reduces the complexity of web anomaly detection systems, showing that few and

simple models can be effective for detecting web attacks without a negative impact in the detection results. Additionally, it reduces time, resource consumption and processing time.

- While most of the systems analyze some parts of the request, these WAFs analyze the whole request, including headers. This makes possible to detect attacks embedded in any part of the request.
- Additionally, the stochastic approach has been applied at the argument level, that is, the system learns the structure of a particular argument. This allows to model the argument more precisely than considering a bigger portion of the request, making the detection more accurate.
- Instead of associating the states of the Markov chains to the 256 ASCII characters, as it is usually done in intrusion detection, in this dissertation the characters have been grouped as letters, digits or non-alphanumeric characters. Sriraghavan and Lucchese proved that such grouping of characters does not reduce the performance in the detection of web attacks [Sriraghavan and Lucchese, 2008] and, at the same time, it makes possible to reduce the number of the states in the Markov chain. The smaller the size of Markov matrices is, the lower the time and resource consumption is. The processing time reached by the Markovian system is 7.9 ms/request.
- These systems have been evaluated by using the public and labelled *CSIC* dataset.
- These characteristics make the systems high-speed, high-detection, low resource consumer and simple, satisfying the objectives of this thesis.

2. **Successful application of C4.5, CART, Random Tree and Random Forest decision trees to web intrusion detection for the first time.**

   Decision trees have been proven to be very effective in solving the general intrusion detection problem, however, they have been hardly used in the web attack field. Although the ID3 decision tree has been previously applied to web traffic, as far as we know, it is the first time that C4.5, CART, Random Tree, Random Forest are applied to this type of traffic.

The study about the behavior of these decision trees over HTTP traffic reveals that they can be successfully used to solve the web attack detection problem.

3. **Study of how the number of requests used in the training phase influences the detection accuracy of the system.**

   Several experiments have been carried out by incrementally increasing the number of training requests, in order to study how their variation affects the detection results of the stochastic and ML detection systems. Additionally, this study allows to establish how many requests are necessary to achieve a particularly desired result concerning the detection capability of the system. These aspects have been scarcely studied in the intrusion detection state of the art. The experiments show that the number of training requests to be used depends on the algorithm analyzed. The research developed reveals that for statistical techniques it is not necessary to employ the whole training dataset to train the system. In fact, for the *CSIC* dataset, with approximately a quarter (16 383) of the requests that compose the dataset, a false positive rate of 1% is reached. This number of requests also reaches the lowest FPR for the ML system when the *ECML/PKDD* dataset is used. The need of a low number of training requests, without negatively affecting the detection results, enjoys several advantages: on the one hand, the traffic is easier to gather; on the other hand, the training phase is shorter. The second point is also important in case the system needs periodical retraining.

4. **Proposal of new methods for combining manual and automatic features for machine learning that achieve better detection results than both techniques separately and consume low resources.**

   The proposed methods mix expert knowledge (manual extraction) and n-grams (automatic extraction). In the few cases where such a combination is applied in the literature, it is done by first combining the features and selecting them afterwards. Within the combined feature extraction methods, as a novelty, this thesis proposes different alternatives to implement the combination. Besides the combination method previously mentioned, other new methods have been introduced: one of them consists in selecting the features of the manual and automatic approaches and, later, combining them. The other proposal is similar to the previously mentioned one, but it only takes the selected values of n-grams. The combination alternatives improve the detection results of the manual or automatic techniques separately. Additionally, feature selection has been applied to reduce the number of features and, consequently, to decrease the resource consumption. In some cases, the combination methods have been even able to reduce the processing time.

5. **Successful application of the *GeFS* measure to web traffic for the first time.**

   Although this feature selection measure has been successfully tested for network traffic [Nguyen et al., 2010b], [Nguyen et al., 2010a], it has not been applied before to HTTP traffic. The experiments conducted obtain evidence of its successful capability of selecting features that allow to distinguish between web attacks and normal traffic.

6. **Comparison of the statistical, Markovian and decision tree algorithms in the detection of web attacks.**

   These three detection techniques have been compared using the same reference dataset (*CSIC*), what allows to make comparisons between the algorithms and extract conclusions about their performance. Different properties, such as detection results, processing time and number of needed training requests have been compared, making possible to conclude which algorithm is the most appropriate for a given scenario.

7. **Generation of the *CSIC* dataset: a new public and labeled dataset that provides a common framework to evaluate and compare WAFs.**

   In order to avoid the problems of some other existing datasets, the *CSIC* dataset provides public-access and labeled HTTP traffic that contains modern attacks and that is not anonymized. The dataset is currently being used by the IDS community for the evaluation of their web attack detection systems. Even other formats of the dataset have been created by the scientific community in order to enlarge its possibilities of use.

A number of **papers has been published** as a result of this research:

1. *C. Torrano-Gimenez, A. Perez-Villegas, G. Alvarez. An anomaly-based web application firewall. In Conference on Security and Cryptography (SECRYPT 2009), pages 23–38, 2009. http://jigpal.oxfordjournals.org/content/21/4/560.abstract (last accessed August 2015).*

2. *C. Torrano-Gimenez, A. Perez-Villegas, G. Alvarez. A self-learning anomaly-based web application firewall. Advances in Intelligent and Soft Computing, 63: 85–92, 2009. http://link.springer.com/chapter/10.1007%2F978-3-642-04091-7_11 (last accessed August 2015).*

3. *C. Torrano-Gimenez, A. Perez-Villegas, G. Alvarez. An anomaly-based approach for intrusion detection in web traffic. Journal of Information Assurance and Security, 5(4): 446–454, 2010. http://digital.csic.es/bitstream/10261/40544/1/ARTICULOS315428 %5B1%5D.pdf (last accessed August 2015).*

4. *A. Perez-Villegas, C. Torrano-Gimenez, G. Alvarez. Applying Markov chains to web intrusion detection. In Reunión Española sobre Criptología y Seguridad de la Información (RECSI 2010), pages 361–366, 2010.*

5. *H. T. Nguyen, C. Torrano-Gimenez, G. Alvarez, S. Petrovic, K. Franke. Application of the generic feature selection measure in detection of web attacks. In International Workshop in Computational Intelligence in Security for Information Systems (CISIS 11), pages 25–32, 2011. http://www.researchgate.net/profile/Hai_Nguyen14/publication/ 221271490_Application_of_the_Generic_Feature_Selection_Measure _in_Detection_of_Web_Attacks/links/09e41512b868ff2954000000.pdf (last accessed August 2015).*

6. *C. Torrano-Gimenez, H. T. Nguyen, G. Alvarez, S. Petrovic, K. Franke. Applying feature selection to payload-based web application firewalls. In International Workshop on Security and Communication Networks (IWSCN 11), pages 75–81, 2011. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6827720 (last accessed August 2015).*

7. *C. Torrano-Gimenez, H. T. Nguyen, G. Alvarez, and K. Franke. Combining expert knowledge with automatic feature extraction for reliable web attack detection. Security and Communication Networks, 2012. http://onlinelibrary.wiley.com/doi/10.1002/sec.603/abstract (last accessed August 2015).*

8. *H. T. Nguyen, C. Torrano-Gimenez, G. Alvarez, K. Franke, S. Petrovic. Enhancing the effectiveness of web application firewalls by generic feature selection. Logic Journal of the IGPL, 21(4): 560–570, 2013. http://jigpal.oxfordjournals.org/content/21/4/560.abstract (last accessed August 2015).*

This thesis has also been presented in the Security Conference RootedCon 2015 [Rooted, 2015]. *C. Torrano-Gimenez. Doing research about web application firewalls. RootedCon 2015. http://es.slideshare.net/ctorranog.*

Additional publications related to web security have been done during the thesis period:

1. *C. Torrano-Gimenez, A. Perez-Villegas, G. Alvarez. WASAT- a new authorization security analysis tool. In Web Application Security conference (IBWAS09), pages 39–49, 2009. http://elibrary.palcomtech.ac.id/wp-content/uploads/Web-Application-Security.pdf#page=50 (last accessed August 2015).*

2. *M. Balduzzi, C. Torrano-Gimenez, D. Balzarotti, E. Kirda. Automated discovery of parameter pollution vulnerabilities in web applications. In Network and Distributed System Security Symposium (NDSS), 2011.* ***BEST PAPER AWARD.*** *http://www.iseclab.org/people/embyte/papers/hpp.pdf (last accessed August 2015).*

3. *C. Torrano-Gimenez, A. Perez-Villegas, G. Alvarez. TORPEDA: Un conjunto de datos ampliable para la evaluación de cortafuegos de aplicaciones web. In Reunión Española sobre Criptología y Seguridad de la Información (RECSI 2012), pages 77–82, 2012. http://recsi2012.mondragon.edu/es/programa/recsi2012_submission_73.pdf (last accessed August 2015).*

4. *S. Pastrana, C. Torrano-Gimenez, H. T. Nguyen, A. Orfila. Anomalous web payload detection: Evaluating the resilience of 1-gram based classifiers. In Proceedings of the* 8*th* *Conference on Intelligent Distributed Computing (IDC), pages 195–201, 2014. http://www.seg.inf.uc3m.es/papers/2014IDC.pdf (last accessed August 2015).*

Various **research stays abroad** have been completed during the thesis period:

1. University of Nottingham (U.K.), from June to August 2008. The stay, hosted by Dr. Uwe Aickelin, was dedicated to the application of an immune-inpired algorithm to web attack detection.

2. Technique University of Munich (Germany), from May to July 2009. With the guidance of Dr. Gerhard Münz, the study of the application of statistic techniques and information theory to the detection of automated web attacks was carried out.

3. Research Center Eurecom (Sophia Antipolis, France), that is part of the International Secure Systems Lab (ISECLAB), from March to May 2010. In collaboration with Dr. Engin Kirda, a tool that discovers HTTP parameter pollution vulnerabilities (PAPAS) was developed.

6.4 Future work

Different detection systems against web attacks have been presented in this work. The contributions derived from this research still leave place for extended work and related open research lines. In the following, the possible future work is presented:

- **Statistical characterization of other parts of the request**. The stochastic detection systems proposed have been designed such that not all parts of the systems are ready to dynamically adjust to the addition of new web pages. Then, when new pages are included to the protected web application, the system needs to be retrained in order to update the normal behavior to the new definition of the application. To overcome this drawback, the idea is to change the ongoing definition of the resources in the XML file. Instead of using the current static definition of the resources, writing exactly in the NBD file the string corresponding to the resource name, the proposal is to statistically characterize the name of the resources. This is what is done with the values of the arguments. It allows to capture the structure of the resources in a more general way, what facilitates their matching when they experiment dynamic changes. The same idea can be applied to names of arguments and headers.

- **Feature extraction with high order *n*-grams**. For the experiments performed, monograms have been used. Although in Sec. 5.4.2.2 it was explained why this decision was made, it would be interesting to investigate how the selection of higher order n-grams ($n > 1$) affects the detection results of the systems.

- **Considering the presence of noise in the training data**. The presented stochastic systems have been trained with only normal traffic (without noise in data). However, only normal traffic might be difficult to obtain, therefore in some works, like [Criscione and Zanero, 2009] and [Corona et al., 2009], their authors have tested how resistent the systems are to certain amount of noise in the training traffic (usually around 1%).

  In case of noise, the proposed stochastic systems would register illegal characters in the NBD file. Since argument/header values are characterized by statistical intervals or Markov chains, a certain amount of noise in the training traffic could be tolerated. However, as the remaining part of the NBD file includes strings literally, attacks containing illegal strings would not be detected. In order to avoid that, we are already working in the development of a kind of filtering process, which is based on the assumption that noise within traffic is less frequent than normal traffic.

The proposed process should be applied to different items: methods, header name, directories, files and argument names. The idea of the filter method is deleting from the NBD file those elements that are considered noise in the traffic. This provokes that noise in certain parts of the requests is not learned and therefore, it does not affect the detection results. A study of how much noise the algorithms can tolerate will be further investigated as future work.

- **Detecting multi-request web attacks**. The systems presented have been designed to detect attacks that involve a single request. An extension of the systems would require detecting web attacks that involve several requests, such as fuzzing or password cracking. In the detection of these attacks, it is probable that independently analyzing the requests does not give an indication of attack. Nevertheless, analyzing the requests as a group, the attack is revealed. This idea can also be extended to detect APTs, that operate in *low-and-slow* mode. Besides how to defend against these attacks, this research topic also involves obtaining traffic to test how well the detection algorithms are able to distinguish this kind of attacks from normal traffic.

- **Sampling training requests**. The contribution on the influence of training requests in the detection results opens interesting research lines related to the optimization of the training phase of detection systems. The shorter the training phase, the better, as less computational time and resources are necessary. However if the system is not trained with enough requests, it will not be effective in the detection. Thus, the proposal for future work is optimizing the training period, by minimizing the training requests while the system obtains enough information for an effective detection. In order to do that, the requests that supply more information to the system should be selected. Although this topic has been studied in regard to network traffic ([Androulidakis and Papavassiliou, 2008], [Androulidakis et al., 2009],[Bakhoum, 2011b], [Bartos and Rehak, 2012]), it has not received much attention in relation to web traffic yet.

  Studying the requests that optimize the training process is also highly related to adequate ways of creating datasets, which is a key issue in evaluating IDSs.

- **Adaptive systems**. As mentioned, the systems presented have been designed so that when the protected web application suffers changes, the algorithms need to be retrained. Furthermore, the environment or the traffic might also change. In that case, the current solution would also require retraining. To overcome this drawback, adaptive IDSs readjust automatically to these changes. For that, the idea is that instead of stopping learning immediately after the training phase, the algorithms continue acquiring knowledge during the operation of the system. That is, the system learns from the incorrectly classified instances.

  This approach copes with several problems related to IDSs: it reduces high false positives rates, avoids retraining and the systems adapt to changing environments. It is an interesting future research line.

- **Application to other contexts**. The algorithms presented can be applied to detect anomalies in other types of traffic, like FTP or Domain Name System (DNS). They can also be applied to detect different kinds of threats, such as APT, insiders or information leakage. In order to do that, the systems should be fed with the appropriate traffic.

References

[Abadeh et al., 2011] Abadeh, M. S., Mohamadi, H., and Habibi, J. (2011). Design and analysis of genetic fuzzy systems for intrusion detection in computer networks. *Expert Systems with Applications*, 38(6):7067–7075. http://www.sciencedirect.com/science/article/pii/S0957417410013692 (last accessed August 2015). 2.4.2

[Abdi and Williams, 2010] Abdi, H. and Williams, L. J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459. http://wwwpub.utdallas.edu/~herve/abdi-wireCS-PCA2010-inpress.pdf (last accessed August 2015). 2.3.2.2

[Acunetix, 2014] Acunetix (2014). Cross-Site Scripting (XSS) Attack. https://www.acunetix.com/websitesecurity/cross-site-scripting/ (last accessed August 2015). 2.1.3

[Agrawal and Srikant, 1994] Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. http://dl.acm.org/citation.cfm?id=645920.672836 (last accessed August 2015). 2.3.1

[Alvarez and Petrović, 2003a] Alvarez, G. and Petrović, S. (2003). A new taxonomy of web attacks suitable for efficient encoding. *Computers & Security*, 22(5):435–449. http://www.sciencedirect.com/science/article/pii/S0167404803005121 (last accessed August 2015). 2.1.3

[Alvarez and Petrović, 2003b] Alvarez, G. and Petrović, S. (2003). A taxonomy of web attacks. In *Proceedings of ICWE*, pages 295–298. http://dx.doi.org/10.1007/3-540-45068-8_56 (last accessed August 2015). 2.1.3

[Amiri et al., 2011] Amiri, F., Yousefi, M. M. R., Lucas, C., Shakery, A., and Yazdani, N. (2011). Mutual information-based feature selection for intrusion detection systems. *Journal of Network and Computer Applications*, 34(4):1184–1199. http://dx.doi.org/10.1016/j.jnca.2011.01.002 (last accessed August 2015). 2.3.2.2

[Amit and Geman, 1997] Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588. http://www.cse.buffalo.edu/~jcorso/t/555pdf/1997_AmitGemanNCb.pdf (last accessed August 2015). 5.3.3

[Anderson et al., 1994] Anderson, D., Frivold, T., Tamaru, A., and Valdes, A. (1994). Next generation intrusion detection expert system (NIDES), software users manual. http://www.csl.sri.com/papers/7sri/7sri.pdf (last accessed August 2015). 2.4.1.1

[Anderson et al., 1995] Anderson, D., Lunt, T. F., Javitz, H., Tamaru, A., and Valdes, A. (1995). Detecting unusual program behavior using the statistical components of NIDES. http://csl.sri.com/papers/5sri/5sri.pdf (last accessed August 2015). 2.4.1.1

[Androulidakis et al., 2009] Androulidakis, G., Chatzigiannakis, V., and Papavassiliou, S. (2009). Network anomaly detection and classification via opportunistic sampling. *IEEE Network, Special issue title on recent developments in network intrusion detection*, 23(1):6–12. 6.4

[Androulidakis and Papavassiliou, 2008] Androulidakis, G. and Papavassiliou, S. (2008). Improving network anomaly detection via selective flow-based sampling. *Communications, IET*, 2(3):399–409. 6.4

[Angelino, 2014] Angelino, E. L. (2014). *Accelerating Markov chain Monte Carlo via parallel predictive prefetching*. PhD thesis, The School of Engineering and Applied Sciences, Harvard University. 4.11.1

[Anitha and Vaidehi, 2006] Anitha, A. and Vaidehi, V. (2006). Context based application level intrusion detection system. In *Proceedings of the International conference on Networking and Services*, ICNS '06, page 16, Washington, DC, USA. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1690488&abstractAccess=no&userType=inst (last accessed August 2015). 2.2.1.3

[Ariu et al., 2011] Ariu, D., Tronci, R., and Giacinto, G. (2011). HMMPayl: An intrusion detection system based on Hidden Markov Models. *Computers & Security*, 30(4):221–241. http://pralab.diee.unica.it/sites/default/files/Ariu_COSE2011.pdf (last accessed August 2015). 2.2.1.2, 2.3.1.1, 2.4.1.2

[Arnes et al., 2006] Arnes, A., Valeur, F., Vigna, G., and Kemmerer, R. A. (2006). Using Hidden Markov Models to evaluate the risks of intrusions. In *Proceedings of the 9th international conference on Recent Advances in Intrusion Detection*, RAID'06, pages 145–164, Berlin, Heidelberg. http://cs.ucsb.edu/~vigna/publications/2006_arnes_valeur_vigna_kemmerer_RAID.pdf (last accessed August 2015). 2.4.1.2

[Atzori et al., 2010] Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805. 1.1

[Bakhoum, 2011a] Bakhoum, E. (2011). Intrusion detection model based on selective packet sampling. *EURASIP Journal on Information Security*, 2011:1–12. http://jis.eurasipjournals.com/content/pdf/1687-417X-2011-2.pdf (last accessed August 2015). 2.4.1.2

[Bakhoum, 2011b] Bakhoum, E. G. (2011). Intrusion detection model based on selective packet sampling. *EURASIP J. Information Security*, 2011:1–12. http://jis.eurasipjournals.com/content/pdf/1687-417X-2011-2.pdf (last accessed August 2015). 6.4

[Balduzzi et al., 2011] Balduzzi, M., Torrano Gimenez, C., Balzarotti, D., and Kirda, E. (2011). Automated discovery of parameter pollution vulnerabilities in web appplications. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, San Diego, California, USA. http://www.isoc.org/isoc/conferences/ndss/11/pdf/6_1.pdf (last accessed August 2015). 2.1.3

[Balon-Perin, 2012] Balon-Perin, A. (2012). Ensemble-based methods for intrusion detection. Master's thesis, Department of Computer and Information Science. Norwegian University of Science and Technology (NTNU), Trondheim (Norway). http://code.ulb.ac.be/dbfiles/Bal2012mastersthesis.pdf (last accessed August 2015). 2.4.2

[Barbara et al., 2001] Barbara, D., Wu, N., and Jajodia, S. (2001). Detecting novel network intrusions using Bayes estimators. In *Proceedings of SIAM Internatinal Conference on Data Mining*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.3388&rep=rep1&type=pdf (last accessed August 2015). 2.4.2

[Barber, 2001] Barber, R. (2001). Hackers profiled– who are they and what are their motivations? *Computer Fraud & Security*, 2001(2):14–17. http://www.sciencedirect.com/science/article/pii/S1361372301020176 (last accessed August 2015). 1.1

[Barnett et al., 2013] Barnett, R. C., Coleman, J., Shezaf, O., Grossman, J., and Auger, R. (2013). Web hacking incident database. http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database (last accessed August 2015). 2.1.3

[Bartos and Rehak, 2012] Bartos, K. and Rehak, M. (2012). Towards efficient flow sampling technique for anomaly detection. In *Traffic Monitoring and Analysis*, volume 7189 of *Lecture Notes in Computer Science*, pages 93–106. http://pam2012.ftw.at/TMA/papers/TMA2012paper16.pdf (last accessed August 2015). 6.4

[Bilge and Dumitras, 2012] Bilge, L. and Dumitras, T. (2012). Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS12, pages 833–844, New York, NY, USA. http://users.ece.cmu.edu/~tdumitra/public_documents/bilge12_zero_day.pdf (last accessed August 2015). 2.1.3

[Bluman, 2007] Bluman, A. G. (2007). *Elementary Statistics: A Step by Step Approach*. Mc Graw Hill. 4.5.2

[Bojinov et al., 2009] Bojinov, H., Bursztein, E., Lovett, E., and Boneh, D. (2009). Embedded management interfaces: emerging massive insecurity. In *Proceedings of Black Hat USA*. http://cdn.ly.tl/talks/embedded-management-interfaces-emerging-massive-insecurity-paper.pdf (last accessed August 2015). 1.1

[Bolzoni and Etalle, 2008] Bolzoni, D. and Etalle, S. (2008). Boosting web intrusion detection systems by inferring positive signatures. In *Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part II On the Move to Meaningful Internet Systems*, OTM '08, pages 938–955, Berlin, Heidelberg. http://doc.utwente.nl/64826/1/bolzoni_etalle_sphinx.pdf (last accessed August 2015). 2.5, 5.3.3

[Bolzoni et al., 2006] Bolzoni, D., Etalle, S., and Hartel, P. (2006). POSEIDON: a 2-tier anomaly-based network intrusion detection system. In *Proceedings of the 4th IEEE International Workshop on Information Assurance, IWIA*, pages 144–156. http://arxiv.org/pdf/cs/0511043 (last accessed August 2015). 2.4.2

[Bolzoni et al., 2009] Bolzoni, D., Etalle, S., and Hartel, P. H. (2009). Panacea: Automating attack classification for anomaly-based network intrusion detection systems. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, pages 1–20, Berlin, Heidelberg. http://doc.utwente.nl/65446/1/bh09.pdf (last accessed August 2015). 2.3.1.1, 2.4.2

[Bouzida et al., 2004] Bouzida, Y., Cuppens, F., Cuppens-Boulahia, N., and Gombault, S. (2004). Efficient intrusion detection using principal component analysis. In *Proceedings of the troisième Conférence sur la Sécurité et Architectures Réseaux (SAR)*, La Londe, France. http://yacine.bouzida.free.fr//Articles/2004SAR.pdf (last accessed August 2015). 2.3.2.2

[Bradley, 1997] Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159. http://lamda.nju.edu.cn/yuy/dm07/auc.pdf (last accessed August 2015). 4.9

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32. https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf (last accessed August 2015). 5.3.3

[Breiman et al., 1984] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth. 5.3.3

[Breunig et al., 2000] Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD'00, pages 93–104, New York, NY, USA. http://www.dbs.ifi.lmu.de/Publikationen/Papers/LOF.pdf (last accessed August 2015). 2.4.2

[Brezo et al., 2013] Brezo, F., de la Puerta, J. G., Ugarte-Pedrero, X., Santos, I., Bringas, P. G., and Barroso, D. (2013). A supervised classification approach for detecting packets originated in a HTTP-based botnet. *CLEI Electronic Journal*, 16(3). http://www.clei.org/cleiej/papers/v16i3p2.pdf (last accessed August 2015). 1.1

[Bringas et al., 2008] Bringas, P. G., Penya, Y. K., Paraboschi, S., and Salvaneschi, P. (2008). Bayesian-networks-based misuse and anomaly prevention system. In *Proceedings of the Tenth International Conference on Enterprise Information Systems*, pages 12–16. http://paginaspersonales.deusto.es/ypenya/publi/penya_ICEIS08_Bayesian-networks-basedmisuseandanomalypreventionsystem.pdf (last accessed August 2015). 2.2.1.3

[Brown et al., 2009] Brown, C., Cowperthwaite, A., Hijazi, A., and Somayaji, A. (2009). Analysis of the 1999 DARPA/Lincoln Laboratory IDS Evaluation Data with NetADHICT. In *Proceedings of IEEE International Conference on Computational Intelligence for Security and Defense Applications*, pages 67–73, Piscataway, NY, USA. https://www.ccsl.carleton.ca/~carson/papers/cisda09-netadhict.pdf (last accessed August 2015). 3.1.2

[Burkeman, 2009] Burkeman, O. (2009). Forty years of the internet: how the world changed for ever. The Guardian. http://www.theguardian.com/technology/2009/oct/23/internet-40-history-arpanet (last accessed August 2015). 1.1

[Camacho et al., 2014] Camacho, J., Macia-Fernandez, G., Diaz-Verdejo, J., and Garcia-Teodoro, P. (2014). Tackling the big data 4 vs for anomaly detection. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 500–505. 1.1

[Cárdenas et al., 2013] Cárdenas, A. A., Manadhata, P. K., and Rajan, S. (2013). Big data analytics for security intelligence. Technical report, Cloud Security Alliance. https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Big_Data_Analytics_for_Security_Intelligence.pdf (last accessed August 2015). 1.1

[Carettoni and di Paola, 2009] Carettoni, L. and di Paola, S. (2009). HTTP parameter pollution. In *Proceedings of OWASP AppSec Europe 2009*. http://www.owasp.org/images/b/ba/AppsecEU09_CarettoniDiPaola_v0.8.pdf (last accessed August 2015). 2.1.3

[Chen et al., 2005] Chen, W.-H., Hsu, S.-H., and Shen, H.-P. (2005). Application of SVM and ANN for intrusion detection. *Computers & Operations Research*, 32:2617–2634. http://ece.ut.ac.ir/DBRG/seminars/AdvancedDB/2006/Moshtaghi/Resources/Application%20of%20SVM%20and%20ANN%20for%20intrusion%20detection.pdf (last accessed August 2015). 5.1

[Cheng, 2009] Cheng, S.-Y. (2009). A design and implementation of hybrid web application IDS built with Snort and anomaly detection. Master's thesis, Institute of Computer and Communication Engineering, National Cheng Kung University, Taiwan (R.O.C.). 4.7

[Corchado and Herrero, 2011] Corchado, E. and Herrero, A. (2011). Neural visualization of network traffic data for intrusion detection. *Applied Soft Computing*, 11(2):2042–2056. http://gicap.ubu.es/publications/2011/PDF/2011_c01_Neural_Visualization_of_Network_Traffic_Data.pdf (last accessed August 2015). 2.4.2

[Corona et al., 2009] Corona, I., Ariu, D., and Giacinto, G. (2009). HMM-Web: A Framework for the Detection of Attacks Against Web Applications. In *Proceedings of the 2009 IEEE International Conference on Communications*, ICC'09, pages 747–752, Piscataway, NJ, USA. http://pralab.diee.unica.it/sites/default/files/Corona_ICC2009.pdf (last accessed August 2015). 2.4.1.2, 6.4

[Corona and Giacinto, 2010] Corona, I. and Giacinto, G. (2010). Detection of server-side web attacks. In *Proceedings of WAPA*, volume 11 of *JMLR Proceedings*, pages 160–166. http://jmlr.org/proceedings/papers/v11/corona10a/corona10a.pdf (last accessed August 2015). 2.4.2

[Criscione and Zanero, 2009] Criscione, C. and Zanero, S. (2009). Masibty: An anomaly based intrusion prevention system for web applications. In *Proceedings of Black Hat-Europe*. http://www.blackhat.com/presentations/bh-europe-09/Zanero_Criscione/BlackHat-Europe-2009-Zanero-Criscione-Masibty-Web-App-Firewall-wp.pdf (last accessed August 2015). 2.2.1.1, 2.3.1, 2.4.1.1, 4.5.1, 4.7, 1, 6.4

[Curry et al., 2013] Curry, S., Kirda, E., Schwartz, E., Stewart, W. H., and Yoran, A. (2013). Big data fuels intelligence-driven security. Technical report, RSA, EMC^2. http://www.emc.com/collateral/industry-overview/big-data-fuels-intelligence-driven-security-io.pdf (last accessed August 2015). 1.1

[Cutler and Stevens, 2006] Cutler, A. and Stevens, J. R. (2006). [23] random forests for microarrays. In *DNA Microarrays, Part B: Databases and Statistics*, volume 411 of *Methods in Enzymology*, pages 422–432. http://www.sciencedirect.com/science/article/pii/S007668790611023X (last accessed August 2015). 5.3.3

[Das et al., 2009] Das, D., Sharma, U., and Bhattacharyya, D. K. (2009). A web intrusion detection mechanism based on feature based data clustering. In *Proceedings of IEEE International Advance Computing Conference (IACC 2009)*, Patiala, India. 2.4.2

[Davanzo et al., 2011] Davanzo, G., Medvet, E., and Bartoli, A. (2011). Anomaly detection techniques for a web defacement monitoring service. *Expert Systems with Applications*, 38(10):12521–12530. http://dx.doi.org/10.1016/j.eswa.2011.04.038 (last accessed August 2015). 2.4.2

[Davis and Clark, 2011] Davis, J. J. and Clark, A. J. (2011). Data preprocessing for anomaly based network intrusion detection: A review. *Computers & Security*, 30(6-7):353–375. http://dx.doi.org/10.1016/j.cose.2011.05.008 (last accessed August 2015). 2.3, 2.3.1

[De Ville, 2006] De Ville, B. (2006). *Decision Trees –What Are They?*. Chapter 1 of Decision Trees for Business Intelligence and Data Mining: Using SAS Enterprise Miner. SAS Institute Inc., Cary, NC, USA. http://support.sas.com/publishing/pubcat/chaps/57587.pdf (last accessed August 2015). 2.4.2

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Transactions on Evolutionary Computation*, 6(2):182–197. http://dx.doi.org/10.1109/4235.996017 (last accessed August 2015). 2.3.2.1

[Debar et al., 1999] Debar, H., Dacier, M., and Wespi, A. (1999). Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822. http://galaxy.cs.lamar.edu/~bsun/seminar/example_papers/IDS_taxonomy.pdf (last accessed August 2015). 2.4

[Denning and Neumann, 1985] Denning, D. and Neumann, P. (1985). Requirements and model for IDES. A real-time intrusion detection system. Technical report, Computer Science Laboratory, SRI

International, Menlo Park, CA. http://faculty.nps.edu/dedennin/publications/IDESReportSRI1985.pdf (last accessed August 2015). 2.4.1.1

[Depren et al., 2005] Depren, O., Topallar, M., Anarim, E., and Ciliz, M. K. (2005). An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Systems with Applications*, 29(4):713–722. http://www.sciencedirect.com/science/article/pii/S0957417405000989 (last accessed August 2015). 2.2.1.3, 2.4.2

[Dessiatnikoff et al., 2011] Dessiatnikoff, A., Akrout, R., Alata, E., Kaaniche, M., and Nicomette, V. (2011). A clustering approach for web vulnerabilities detection. In *Proceedings of the* 17*th* *IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 194–203. http://hal.inria.fr/docs/00/75/52/12/PDF/prdc2011.pdf (last accessed August 2015). 2.4.2

[Di Crescenzo et al., 2005] Di Crescenzo, G., Ghosh, A., and Talpade, R. (2005). Towards a theory of intrusion detection. In *Proceedings of the* 10*th* *European conference on Research in Computer Security*, ESORICS'05, pages 267–286, Berlin, Heidelberg. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.8499&rep=rep1&type=pdf (last accessed August 2015). 2.4.2

[Domingos, 2012] Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87. http://doi.acm.org/10.1145/2347736.2347755 (last accessed August 2015). 2.3.2, 2.5

[Duda et al., 2001] Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. Wiley, New York, 2*nd* edition. 5.3.3

[Düssel et al., 2008] Düssel, Patrick and Gehl, Christian and Laskov, Pavel and Rieck, Konrad (2008). Incorporation of application layer protocol syntax into anomaly detection. In *Proceedings of ICISS*, volume 5352 of *Lecture Notes in Computer Science*, pages 188–202.

http://www-rsec.cs.uni-tuebingen.de/laskov/papers/iciss2008.pdf (last accessed August 2015). 2.4.2

[Ebeling and Nicolis, 1992] Ebeling, W. and Nicolis, G. (1992). Word frequency and entropy of symbolic sequences: a dynamical perspective. *Chaos, Solitons & Fractals*, 2(6):635–650. http://www.sciencedirect.com/science/article/pii/096007799290058U (last accessed August 2015).

[El-Alfy and Al-Obeidat, 2014] El-Alfy, E.-S. M. and Al-Obeidat, F. N. (2014). A multicriterion fuzzy classification method with greedy attribute selection for anomaly-based intrusion detection. *Procedia Computer Science*, 34(0):55–62. http://www.readcube.com/articles/10.1016/j.procs.2014.07.037 (last accessed August 2015). 2.2.1.3

[Elliott et al., 1995] Elliott, R. J., Aggoun, L., and Moore, J. B. (1995). *Hidden Markov Models. Estimation and control.* Springer. 2.4.1.2

[Ellison et al., 2009] Ellison, S. L. R., Barwick, V. J., and Farrant, T. J. D. (2009). *Practical Statistics for the Analytical Scientist.* The Royal Society of Chemistry. 4.8

[Estévez-Tapiador et al., 2004] Estévez-Tapiador, J. M., García-Teodoro, P., and Díaz-Verdejo, J. E. (2004). Measuring normality in HTTP traffic for anomaly-based intrusion detection. *Computer Networks*, 45(2):175–193. http://www.seg.inf.uc3m.es/papers/2004comnet.pdf (last accessed August 2015). 2.4.1.2, 3.1.2, 4.7.2, 4.9

[Feller, 1968] Feller, W. (1968). *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley. 4.2

[Fielding et al., 1999] Fielding, R. T., Gettys, J., Mogul, J. C., Nielsen, H. F., Masinter, L., Leach, P. J., and Berners-Lee, T. (1999). Hypertext transfer protocol – HTTP/1.1. RFC2616. https://www.ietf.org/rfc/rfc2616.txt (last accessed August 2015). 2.1.2

[Fogla et al., 2006] Fogla, P., Sharif, M., Perdisci, R., Kolesnikov, O., and Lee, W. (2006). Polymorphic blending attacks. In *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, USENIX-SS'06, Berkeley, CA, USA. USENIX Association. http://dl.acm.org/citation.cfm?id=1267336.1267353 (last accessed August 2015). 2.4

[Fonseca et al., 2008] Fonseca, I. L., Pérez, F. M., Fernández, R. L., Gimeno, F. J. M., and Martínez-Abarca, J. A. G. (2008). Método para la detección de intrusos mediante redes neuronales basado en la reducción de características. In *Proceedings of Desarrollo de grandes aplicaciones de red: V Jornadas, JDARE 2008*. http://rua.ua.es/dspace/bitstream/10045/15687/1/JDARE-08-H.pdf (last accessed August 2015). 2.3.2.2

[Forrest et al., 1996] Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A. (1996). A sense of self for Unix processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, SP '96, pages 120–128, Washington, DC, USA. http://www.cs.unm.edu/~forrest/publications/ieee-sp-96-unix.pdf (last accessed August 2015). 2.2.1.2, 2.3.1.1

[Freedman, 1971] Freedman, D. (1971). *Markov Chains*. Holden-Day. 2.4.1.2

[García et al., 2006] García, V. H., Monroy, R., and Quintana, M. (2006). Web attack detection using ID3. In Debenham, J., editor, *Professional Practice in Artificial Intelligence*, volume 218 of *IFIP International Federation for Information Processing*, pages 323–332. http://pitagoras.usach.cl/~gfelipe/wcc/papers/Symposium/Article_34-Garcia.pdf (last accessed August 2015). 2.4.2, 2.4.2

[García-Teodoro et al., 2009] García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., and Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28:18–28.

http://web.info.uvt.ro/~dzaharie/cne2013/proiecte/aplicatii/intrusion_detection_systems/IDS_general.pdf (last accessed August 2015). 1.2, 2.4, 2.4.1.2

[Gopal and Casella, 2011] Gopal, V. and Casella, G. (2011). Running regenerative Markov chains in parallel. http://www.stat.ufl.edu/archived/casella/Papers/Gopal-Casella.pdf (last accessed August 2015). 4.11.1

[Goyal and Aggarwal, 2012] Goyal, M. K. and Aggarwal, A. (2012). Composing signatures for misuse intrusion detection system using genetic algorithm in an offline environment. In *Proceedings of ACITY (1)*, volume 176 of *Advances in Intelligent Systems and Computing*, pages 151–157. http://www.bibsonomy.org/bibtex/24c7e655f213e994767508fa3e80c6f0a/dblp (last accessed August 2015). 2.2.1.3

[Greensmith et al., 2010] Greensmith, J., Whitbrook, A., and Aickelin, U. (2010). Artificial immune systems. In *Handbook of Metaheuristics*, pages 421–448. Springer. 2.3.1.1

[Grove, 2010] Grove, R. F. (2010). *Web Based Application Development*. Jones and Bartlett Publishers. 2.1.1, 2.1.2

[Guyon et al., 2006] Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L. A. (2006). *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. 2.3.2

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explorations Newsletter*, 11(1):10–18. http://www.cms.waikato.ac.nz/~ml/publications/2009/weka_update.pdf (last accessed August 2015). 3.5, 5.3.3, 5.4.2.4

[Hall, 1999] Hall, M. A. (1999). *Correlation-based Feature Subset Selection for Machine Learning*. PhD thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand. http://www.cs.waikato.ac.nz/~mhall/thesis.pdf (last accessed August 2015). 5.3.2

[Hautamäki et al., 2004] Hautamäki, V., Kärkkäinen, I., and Fränti, P. (2004). Outlier detection using k-nearest neighbour graph. In *Proceedings of* 17th *International Conference on the Pattern Recognition ICPR'04*, volume 3, pages 430–433. ftp://ftp.cs.joensuu.fi/pub/franti/papers/Hautamaki/P2.pdf (last accessed August 2015). 2.4.2

[Hawkins, 2004] Hawkins, D. (2004). The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44:1–12. http://www.cse.hcmut.edu.vn/~chauvtn/data_mining/Reading/Chapter4-Classification/2004TheProblemofOverfitting.pdf (last accessed August 2015). 2.3.2, 2.5

[Higgins, 2009] Higgins, K. J. (2009). SANS Report: 60 percent of all attacks hit web applications, most in the U.S. Dark Reading. http://www.darkreading.com/risk/sans-report-60--of-all-attacks-hit-web-applications-most-in-the-us/d/d-id/1131937? (last accessed August 2015). 2.1.3

[Ho, 1995] Ho, T. K. (1995). Random decision forests. In *Proceedings of ICDAR*, pages 278–282. ftp://cm.bell-labs.com/who/tkh/papers/odt.pdf (last accessed August 2015). 5.3.3

[Ho, 1998] Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844. http://dx.doi.org/10.1109/34.709601 (last accessed August 2015). 5.3.3

[Hosseinkhani et al., 2011] Hosseinkhani, M., Tarameshloo, E., and Sadeghiyan, B. (2011). A two dimensional approach for detecting input validation attacks based on HMM. In *Proceedings of International Conference on Database and Data Mining (ICDDM 2011)*. http://ceit.aut.ac.ir/~87131084/abst/2DimensionalWEBIDS.pdf (last accessed August 2015). 2.4.1.2, 3.1.2

[Ibrahim, 2010] Ibrahim, L. M. (2010). Anomaly network intrusion detection system based on distributed time-delay neural network (DTDNN). *Journal of Engineering Science and Technology*, 5:457–471. http://jestec.taylors.edu.my/Vol5Issue4December10/Vol_5_4_457_471_L.M.Ibrahim.pdf (last accessed August 2015). 2.4.2

[Ihler et al., 2006] Ihler, A., Hutchins, J., and Smyth, P. (2006). Adaptive event detection with time-varying Poisson processes. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 207–216, New York, NY, USA. http://www.datalab.uci.edu/papers/event_detection_kdd06.pdf (last accessed August 2015). 2.4.2

[ITU, 2015] ICT Data and Statistics Division, ITU (2015). ICT facts & figures. the world in 2015. Technical report, International Telecommunication Union (ITU). http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf. 1.1

[Jamdagni et al., 2013] Jamdagni, A., Tan, Z., He, X., Nanda, P., and Liu, R. P. (2013). RePIDS: A multi tier Real-time Payload-based Intrusion Detection System. *Computer Networks*, 57(3):811–824. http://www.sciencedirect.com/science/article/pii/S1389128612003544 (last accessed August 2015). 2.3.2.2

[Jifa and Lingling, 2014] Jifa, G. and Lingling, Z. (2014). Data, dikw, big data and data science. *Procedia Computer Science*, 31(0):814 – 821. 2nd International Conference on Information Technology and Quantitative Management, {ITQM} 2014. 1.1

[Jovanovic et al., 2006] Jovanovic, N., Kruegel, C., and Kirda, E. (2006). Pixy: A static analysis tool for detecting web application vulnerabilities (short paper). In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, SP '06, pages 258–263, Washington, DC, USA. http://infsec.uni-trier.de/download/teachingSS2009/IT-Sicherheit-II/literatur/pixy.pdf (last accessed August 2015). 1.2

[Khalkhali, 2011] Khalkhali, I. (2011). Web Anomaly Host-based IDS with Enhanced Custom Log file: A Machine Learning Approach. Master's thesis, Sharif University of Technology, Kish Island (Iran). 2.4.2

[Kirchner, 2010] Kirchner, M. (2010). A framework for detecting anomalies in HTTP traffic using instance-based learning and K-nearest neighbor classification. In *Proceedings of the* 2^{nd} *International Workshop on Security and Communication Networks (IWSCN)*, pages 1–8. 2.4.2

[Kirda et al., 2009] Kirda, E., Jovanovic, N., Kruegel, C., and Vigna, G. (2009). Client-side cross-site scripting protection. *Computers & Security*, 28(7):592–604. https://www.cs.ucsb.edu/~chris/research/doc/compsec09_noxes.pdf (last accessed August 2015). 4.3

[Kloft et al., 2008] Kloft, M., Brefeld, U., Düessel, P., Gehl, C., and Laskov, P. (2008). Automatic feature selection for anomaly detection. In *Proceedings of the* 1^{st} *ACM workshop on Workshop on AISec*, AISec '08, pages 71–76, New York, NY, USA. https://www.kma.informatik.tu-darmstadt.de/fileadmin/user_upload/Group_KMA/kma_publications/aisec19-kloft.pdf (last accessed August 2015). 2.3.1, 2.3.2.1

[Kohavi, 1995] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of IJCAI*, pages 1137–1145. http://www.cs.iastate.edu/~jtian/cs573/Papers/Kohavi-IJCAI-95.pdf (last accessed August 2015). 5.4.2.4

[Kotsiantis et al., 2006] Kotsiantis, S., Kanellopoulos, D., and Pintelas, P. (2006). Data preprocessing for supervised learning. *International Journal of Computer Science*, 1(2):111–117. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.132.5127&rep=rep1&type=pdf (last accessed August 2015). 1.2, 2.3

[Kowalski, 2010] Kowalski, G. (2010). *Information Retrieval Architecture and Algorithms*. Springer. 2.3.1.1

[Kozik et al., 2014a] Kozik, R., Choraś, M., Renk, R., and Hołubowicz, W. (2014). Modelling http requests with regular expressions for detection of cyber attacks targeted at web applications. In *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14*, volume 299 of *Advances in Intelligent Systems and Computing*, pages 527–535. Springer International Publishing. `http://dx.doi.org/10.1007/978-3-319-07995-0_52` (last accessed August 2015). 3.5, 6.2

[Kozik et al., 2014b] Kozik, R., Choraś, M., Renk, R., and Hołubowicz, W. (2014). A proposal of algorithm for web applications cyber attack detection. In *Computer Information Systems and Industrial Management*, volume 8838 of *Lecture Notes in Computer Science*, pages 680–687. Springer Berlin Heidelberg. `http://dx.doi.org/10.1007/978-3-662-45237-0_61` (last accessed August 2015). 3.5, 6.2

[Kruegel et al., 2003] Kruegel, C., Mutz, D., Robertson, W., and Valeur, F. (2003). Bayesian event classification for intrusion detection. In *Proceedings of the* 19th *Annual Computer Security Applications Conference.*, pages 14–23. `https://www.cs.ucsb.edu/~chris/research/doc/2003_07.pdf` (last accessed August 2015). 2.4.2

[Kruegel et al., 2005a] Kruegel, C., Valeur, F., and Vigna, G. (2005). *Intrusion Detection and Correlation: Challenges and Solutions (Advances in Information Security)*. Springer. 3.1.2

[Kruegel and Vigna, 2003] Kruegel, C. and Vigna, G. (2003). Anomaly detection of web-based attacks. In *Proceedings of the* 10th *ACM conference on Computer and communications security*, CCS '03, pages 251–261, New York, NY, USA. `http://www.cs.ucsb.edu/~vigna/publications/2003_kruegel_vigna_ccs03.pdf` (last accessed August 2015). 4.3

[Kruegel et al., 2005b] Kruegel, C., Vigna, G., and Robertson, W. (2005). A multi-model approach to the detection of web-based attacks. *Computer Networks*, 48(5):717–738.

http://packetstorm.foofus.com/papers/IDS/nids/A-Multi-model-Approach-to-the-Detection-of-Web-based-Attacks.pdf (last accessed August 2015). 2.2.1.2, 2.2.1.3, 2.3.1, 2.4.1.1, 3.4, 4.4, 4.5.1, 4.7, 1

[Krueger et al., 2010] Krueger, T., Gehl, C., Rieck, K., and Laskov, P. (2010). TokDoc: a self-healing web application firewall. In *Proceedings of SAC*, pages 1846–1853. http://user.informatik.uni-goettingen.de/~krieck/docs/2010b-sac.pdf (last accessed August 2015). 2.2.1.1, 2.3.1.1, 2.4.1.2, 4.4, 4.11.1

[Kubat, 2015] Kubat, M. (2015). *An Introduction to Machine Learning*. Springer. http://dx.doi.org/10.1007/978-3-319-20010-1. 5.4.2.4

[Kurgan and Musilek, 2006] Kurgan, L. A. and Musilek, P. (2006). A survey of knowledge discovery and data mining process models. *The Knowledge Engineering Review*, 21(1):1–24. http://biomine.ece.ualberta.ca/papers/KER-KDDM2006.pdf (last accessed August 2015). 2.3

[De la Hoz et al., 2014] De la Hoz, E. D., de la Hoz, E., Ortiz, A., Ortega, J., and Martínez-Álvarez, A. (2014). Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps. *Knowledge-Based Systems*, 71(1): 322–338. http://www.sciencedirect.com/science/article/pii/S0950705114002950 (last accessed August 2015). 2.3.2.1

[Lafferty et al., 2001] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. http://dl.acm.org/citation.cfm?id=645530.655813 (last accessed August 2015). 2.2.1.3

[Lai et al., 2008] Lai, J.-Y., Wu, J.-S., Chen, S.-J., Wu, C.-H., and Yang, C.-H. (2008). Designing a taxonomy of web attacks. In *Proceedings of the 2008 International Conference on Convergence and Hybrid Information Technology*, ICHIT '08, pages 278–282, Washington, DC, USA. http://crypto.nknu.edu.tw/psnl/publications/2008ICHIT_taxonomy.pdf (last accessed August 2015). 2.1.3

[Lanzi et al., 2010] Lanzi, A., Balzarotti, D., Kruegel, C., Christodorescu, M., and Kirda, E. (2010). AccessMiner: Using system-centric models for malware protection. In *Proceedings of ACM Conference on Computer and Communications Security*, pages 399–412. http://seclab.nu/static/publications/ccs2010malware.pdf (last accessed August 2015). 2.2.1.2, 2.3.1.1

[LBNL and ICSI, 2005] Lawrence Berkeley National Laboratory and ICSI (2005). LBNL/ICSI Enterprise Tracing Project. http://www.icir.org/enterprise-tracing/ (last accessed August 2015). 3.1.2

[Li et al., 2015] Li, S.-H., Yen, D. C., Chen, S.-C., Chen, P. S., Lu, W.-H., and Cho, C.-C. (2015). Effects of virtualization on information security. *Computer Standards & Interfaces*, (0). 3.4

[Li, 2004] Li, W. (2004). Using genetic algorithm for network intrusion detection. In *Proceedings of the United States Department of Energy Cyber Security Group 2004 Training Conference*, pages 24–27. http://bit.csc.lsu.edu/~jianhua/krish-1.pdf (last accessed August 2015). 2.4.2

[Li et al., 2008] Li, Y., Guo, L., Tian, Z.-H., and Lu, T.-B. (2008). A lightweight web server anomaly detection method based on transductive scheme and genetic algorithms. *Computer Communications*, 31(17):4018–4025. http://dx.doi.org/10.1016/j.comcom.2008.08.009 (last accessed August 2015). 2.4.2

[Li et al., 2009] Li, Y., Wang, J.-L., Tian, Z., Lu, T., and Young, C. (2009). Building lightweight intrusion detection system using wrapper-based feature selection mechanisms. *Computers & Security*, 28(6):466–475. http://dx.doi.org/10.1016/j.cose.2009.01.001 (last accessed August 2015). 2.3.2, 2.3.2.1

[Liao and Vemuri, 2002] Liao, Y. and Vemuri, V. (2002). Use of K-nearest neighbor classifier for intrusion detection. *Computers & Security*, 21(5):439–448. http://www.sciencedirect.com/science/article/pii/S016740480200514X (last accessed August 2015). 2.4.2

[Lim et al., 2007] Lim, S. H., Wang, L.-L., and DeJong, G. (2007). Explanation-based feature construction. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 931–936. http://ijcai.org/papers07/Papers/IJCAI07-150.pdf (last accessed August 2015). 2.3.1

[Lin et al., 2012] Lin, S.-W., Ying, K.-C., Lee, C.-Y., and Lee, Z.-J. (2012). An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection. *Applied Soft Computing*, 12(10):3285–3290. https://www.researchgate.net/profile/Kuo-Ching_Ying/publication/257635569_An_intelligent_algorithm_with_feature_selection_and_decision_rules_applied_to_anomaly_intrusion_detection/links/0deec53c86e863a069000000?origin=publication_list&ev=pub_srch_pub_xdl (last accessed August 2015). 2.2.1.3, 2.3.2.1, 2.4.2

[Lippmann et al., 2000a] Lippmann, R., Fried, D., Graf, I., Haines, J., Kendall, K., McClung, D., Weber, D., Webster, S., Wyschogrod, D., Cunningham, R., and Zissman, M. (2000). Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In *Proceedings of the DARPA Information Survivability Conference and Exposition, DISCEX '00*, volume 2, pages 12–26. http://www.ll.mit.edu/ideval/files/Evaluating_IDs_DARPA_1998.pdf (last accessed August 2015). 2.4.2, 3.1.2, 5.3

[Lippmann et al., 2000b] Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., and Das, K. (2000). The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, 34(4):579–595. http://www.ll.mit.edu/ideval/files/1999Eval-ComputerNetworks2000.pdf (last accessed August 2015). 3.1.2

[Liu and Motoda, 2007] Liu, H. and Motoda, H. (2007). *Computational Methods of Feature Selection (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*. Chapman & Hall/CRC. 2.3.2

[Lunt et al., 1992] Lunt, T., Tamaru, A., Gilham, F., Jagannathm, R., Jalali, C., and P.G. Neumann, H.S. Javitz, A. V. T. G. (1992). A real-time intrusion detection expert system (IDES). Technical report, Computer Science Laboratory, SRI International,, Menlo Park, CA, USA. http://www.csl.sri.com/papers/9sri/9sri.pdf (last accessed August 2015). 2.4.1.1

[Lutu, 2010] Lutu, P. E. N. (2010). *Dataset selection for aggregate model implementation in predictive data mining*. PhD thesis, University of Pretoria. http://cirg.cs.up.ac.za/thesis/PENLutu.pdf (last accessed August 2015). 2

[Mahoney and Chan, 2001] Mahoney, M. V. and Chan, P. K. (2001). PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic. Technical Report CS-2001-4 2004, Florida Institute of Technology. https://msdn.cs.fit.edu/media/TechnicalReports/cs-2001-04.pdf (last accessed August 2015). 2.3.1.1, 2.4.1.1

[Mahoney and Chan, 2002a] Mahoney, M. V. and Chan, P. K. (2002). Learning models of network traffic for detecting novel attacks. Technical Report CS-2002-08, Florida Institute of Technology. https://www.cs.fit.edu/media/TechnicalReports/cs-2002-08.pdf (last accessed August 2015). 2.4.1.1

[Mahoney and Chan, 2002b] Mahoney, M. V. and Chan, P. K. (2002). Learning nonstationary models of normal network traffic for detecting novel attacks. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 376–385, New York, NY, USA. http://www.ideal.ece.utexas.edu/courses/ee379k/papers/mahoney02novelattacks.pdf (last accessed August 2015). 2.4.1.1

[Mannila et al., 1997] Mannila, H., Toivonen, H., and Inkeri Verkamo, A. (1997). Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, 1(3):259–289. http://dx.doi.org/10.1023/A:1009748302351 (last accessed August 2015). 2.3.1

[McHugh, 2000] McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security*, 3(4):262–294. https://www.cs.nmt.edu/~infosec/CritiqueonTestingIDS.pdf (last accessed August 2015). 3.1.2

[Meng and Kwok, 2014] Meng, Y. and Kwok, L.-F. (2014). Adaptive non-critical alarm reduction using hash-based contextual signatures in intrusion detection. *Computer Communications*, 38(0):50–59. https://www.researchgate.net/profile/Weizhi_Meng/publication/260250039_Adaptive_Non-Critical_Alarm_Reduction_Using_Hash-based_Contextual_Signatures_in_Intrusion_Detection/links/0c960535a806c5039e000000?origin=publication_list&ev=pub_srch_pub_xdl (last accessed August 2015). 2.2.1.3

[MISP, 2015] MISP (2015). Malware information sharing platform. http://www.misp-project.org/ (last accessed August 2015). 2.2.1.3

[Mitchell, 1998] Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA. 2.2.1.3

[MITRE, 2012] MITRE (2012). Common attack pattern enumeration and classification. http://capec.mitre.org/ (last accessed August 2015). 2.1.3

[MITRE, 2014a] MITRE (2014). Common vulnerabilities and exposures (CVE). https://cve.mitre.org/ (last accessed August 2015). 1.2

[MITRE, 2014b] MITRE (2014). Common Weakness Enumeration, version 2.8. http://cwe.mitre.org/data/published/cwe_v2.8.pdf (last accessed August 2015). 2.1.3

[Mulay et al., 2010] Mulay, S. A., Devale, P., and Garje, G. (2010). Intrusion detection system using support vector machine and decision tree. *International Journal of Computer Applications*, 3(3):40–43. http://www.ijcaonline.org/volume3/number3/pxc387993.pdf (last accessed August 2015). 5.1

[Muniyandi et al., 2012] Muniyandi, A. P., Rajeswari, R., and Rajaram, R. (2012). Network anomaly detection by cascading K-means clustering and C4.5 decision tree algorithm. *Procedia Engineering*, 30(0):174–182. http://www.sciencedirect.com/science/article/pii/S1877705812008594 (last accessed August 2015). 2.4.2

[Naiman, 2004] Naiman, D. Q. (2004). Statistical anomaly detection via HTTPD data analysis. *Computational Statistics & Data Analysis*, 45(1):51–67. http://www.sciencedirect.com/science/article/pii/S0167947303001154 (last accessed August 2015). 2.2.1.2, 2.3.1.1

[Newsome and Song, 2005] Newsome, J. and Song, D. (2005). Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *Proceedings of NDSS05*. http://valgrind.org/docs/newsome2005.pdf (last accessed August 2015). 1.2

[Nguyen, 2012] Nguyen, H. T. (2012). *Reliable Machine Learning Algorithms for Intrusion Detection Systems*. PhD thesis, Faculty of Computer Science and Media Technology Gjøvik University College. http://hdl.handle.net/11250/144371 (last accessed August 2015). 2.3.2, 2.4.2, 2, 5.3.3

[Nguyen et al., 2010a] Nguyen, H. T., Franke, K., and Petrović, S. (2010). Improving effectiveness of intrusion detection by correlation feature selection. In *Proceedings of the International Conference on Availability, Reliability and Security (ARES)*, pages 17–24. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5438117&abstractAccess=no&userType=inst (last accessed August 2015). 5

[Nguyen et al., 2010b] Nguyen, H. T., Franke, K., and Petrović, S. (2010). Towards a generic feature-selection measure for intrusion detection. In *Proceedings of the 20th International Conference on Pattern Recognition*, ICPR'10, pages 1529–1532, Washington, DC, USA. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5597038&abstractAccess=no&userType=inst (last accessed August 2015). 2.3.2.2, 5.3, 5.3.2, 5

[Nguyen et al., 2011] Nguyen, H. T., Franke, K., and Petrović, S. (2011). Improving effectiveness of intrusion detection by correlation feature selection. *International Journal of Mobile Computing and Multimedia Communications*, 3(1):21–34. http://www.researchgate.net/publication/220265542_Improving_Effectiveness_of_Intrusion_Detection_by_Correlation_Feature_Selection/file/79e415065422243f45.pdf (last accessed August 2015). 5.3

[Nguyen et al., 2010c] Nguyen, H. T., Petrović, S., and Franke, K. (2010). A comparison of feature-selection methods for intrusion detection. In *Computer Network Security*, volume 6258 of *Lecture Notes in Computer Science*, pages 242–255. Springer Berlin Heidelberg. http://profs.info.uaic.ro/~alaiba/pub/fluxuri-2014/Cercetare%202014/Computer%20Network%20Security.pdf#page=253 (last accessed August 2015). 2.3.2.2

[Nguyen-Tuong et al., 2005] Nguyen-Tuong, A., Guarnieri, S., Greene, D., Shirley, J., and Evans, D. (2005). Automatically hardening web applications using precise tainting. In *Proceedings of the 20th IFIP International Information Security Conference*, pages 372–382. http://dependability.cs.virginia.edu/publications/2005/sec2005.pdf (last accessed August 2015). 1.2

[Olaru and Wehenkel, 2003] Olaru, C. and Wehenkel, L. (2003). A complete fuzzy decision tree technique. *Fuzzy sets and systems*, 138(2):221–254. 5.3.3

[Olusola et al., 2010] Olusola, A. A., Oladele, A. S., and Abosede, D. O. (2010). Analysis of KDD'99 intrusion detection dataset for selection of relevance features. In *World Congress on Engineering and Computer Science, Int. Assoc. Engn*, pages 162–168. 2.3.2.2

[OWASP, 2013] OWASP (2013). Top 10. Technical report. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project (last accessed August 2015). 1.2, 2.1.3, 2.1.3

[Paxson, 2015] Paxson, V. (2015). The Bro network security monitor. https://www.bro.org/ (last accessed August 2015). 2.2.1.3

[Perdisci et al., 2009] Perdisci, R., Ariu, D., Fogla, P., Giacinto, G., and Lee, W. (2009). McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks*, 53(6):864–881. https://pralab.diee.unica.it/sites/default/files/Perdisci_COMNET2009.pdf (last accessed August 2015). 2.3.1.1, 5.4.2.2

[Pfahringer, 2000] Pfahringer, B. (2000). Winning the KDD99 classification cup: Bagged boosting. *SIGKDD Exploration Newsletter*, 1(2):65–66. ftp://ftp.cse.buffalo.edu/users/azhang/disc/disc01/cd1/out/websites\/kdd_explorations_full/pfahringer.ps (last accessed August 2015). 2.4.2, 5.3

[Piatetsky-Shapiro, 1991] Piatetsky-Shapiro, G. (1991). Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases*. AAAI/MIT Press, Cambridge, MA. 2.3.1

[Pillai et al., 2004] Pillai, M. M., Eloff, J. H. P., and Venter, H. S. (2004). An approach to implement a network intrusion detection system using genetic algorithms. In *Proceedings of the 2004 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*, SAICSIT'04, pages 221–228, Republic of South Africa. http://dl.acm.org/citation.cfm?id=1035053.1035080 (last accessed August 2015). 2.4.2

[Porras et al., 1999] Porras, P., Schnackenberg, D., Staniford-Chen, S., Davis, Stillman, M., and Wu, F. (1999). The common intrusion detection framework architecture. http://gost.isi.edu/cidf/drafts/architecture.txt (last accessed August 2015). 2.2.2

[Provost et al., 1998] Provost, F., Fawcett, T., and Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In Kaufmann, M., editor, *Proceedings of the International Conference on Machine Learning*, San Mateo, CA, USA. http://eecs.wsu.edu/~holder/courses/cse6363/spr04/pubs/Provost98.pdf (last accessed August 2015). 4.9

[Quinlan, 1993] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning, 1st ed.* Morgan Kaufmann. 5.3.3

[Quinlan, 1996] Quinlan, J. R. (1996). Bagging, boosting, and C4.5. In *AAAI/IAAI, Vol. 1*, pages 725–730. 2.4.2

[Raïssi et al., 2007] Raïssi, C., Brissaud, J., Dray, G., Poncelet, P., Roche, M., and Teisseire, M. (2007). Web analyzing traffic challenge: Description and results. In *Proceedings of the Discovery Challenge ECML/PKDD'2007*, pages 47–52. http://www.lirmm.fr/~poncelet/publications/papers/awt_raissi.pdf (last accessed August 2015). 3.1.2, 5.4.1

[Ramadas et al., 2003] Ramadas, M., Ostermann, S., and Tjaden, B. (2003). Detecting anomalous network traffic with self-organizing maps. In *Recent Advances in Intrusion Detection*, volume 2820 of *Lecture Notes in Computer Science*, pages 36–54. Springer Berlin Heidelberg. https://etd.ohiolink.edu/!etd.send_file?accession=ohiou1049472005&disposition=inline (last accessed August 2015). 2.4.2

[Ramana, 2007] Ramana, B. V. (2007). *Higher Engineering Mathematics*. McGraw-Hill. 4.2

[Ramos and Abraham, 2005] Ramos, V. and Abraham, A. (2005). ANTIDS: Self organized ant-based clustering model for intrusion detection system. In *Proceedings of the 4th IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology*, pages 977–986. http://arxiv.org/ftp/cs/papers/0412/0412068.pdf (last accessed August 2015). 2.4.2

[Rao et al., 2011] Rao, K. H., Srinivas, G., Damodhar, A., and Krishna, M. V. (2011). Implementation of anomaly detection technique using machine learning algorithms. *International Journal of Computer Science and Telecommunications*, 2:25–31. http://www.ijcst.org/Volume2/Issue3/p5_2_3.pdf (last accessed August 2015). 2.4.2

[Real Time Statistics Project, 2014] Real Time Statistics Project (2014). Internet live stats. http://www.internetlivestats.com/ (last accessed August 2015). 1.1

[Rieck, 2009] Rieck, K. (2009). *Machine Learning for Application-Layer Intrusion Detection*. PhD thesis, IV Faculty, Technical University, Berlin (Germany). http://user.informatik.uni-goettingen.de/~krieck/docs/2009-diss.pdf (last accessed August 2015). 2.3.1, 5.4.2.1

[Rieck and Laskov, 2006] Rieck, K. and Laskov, P. (2006). Detecting unknown network attacks using language models. In *Proceedings of the Third International Conference on Detection of Intrusions and Malware & Vulnerability Assessment*, DIMVA'06, pages 74–90, Berlin, Heidelberg. http://www-rsec.cs.uni-tuebingen.de/laskov/papers/dimva2006.pdf (last accessed August 2015). 2.3.1.1

[Rieck and Laskov, 2007] Rieck, K. and Laskov, P. (2007). Language models for detection of unknown attacks in network traffic. *Journal in Computer Virology*, 2(4):243–256. http://eprints.pascal-network.org/archive/00002788/01/jicv06.pdf (last accessed August 2015). 2.3.1.1

[Riley et al., 2010] Riley, R. D., Ali, N. M., Al-Senaidi, K. S., and Al-Kuwari, A. L. (2010). Empowering users against sidejacking attacks. In *Proceedings of SIGCOMM*, pages 435–436.

http://conferences.sigcomm.org/sigcomm/2010/papers/sigcomm/p435.pdf (last accessed August 2015). 4.3

[Robertson et al., 2006] Robertson, W. K., Vigna, G., Krügel, C., and Kemmerer, R. A. (2006). Using generalization and characterization techniques in the anomaly-based detection of web attacks. In *Proceedings of NDSS*. http://iseclab.org/papers/webfuzzing.pdf (last accessed August 2015). 2.4.1.1, 5.3.3

[Roesch, 1999] Roesch, M. (1999). Snort - lightweight intrusion detection for networks. In *Proceedings of the* 13th *USENIX Conference on System Administration*, LISA '99, pages 229–238, Berkeley, CA, USA. http://static.usenix.org/publications/library/proceedings/lisa99/full_papers/roesch/roesch.pdf (last accessed August 2015). 2.2.1.3

[Rollett et al., 2007] Rollett, H., Lux, M., Strohmaier, M., Dosinger, G., and Tochtermann, K. (2007). The Web 2.0 way of learning with technologies. *International Journal of Learning Technology*, 3(1):87–107. http://www.markusstrohmaier.info/documents/2007_JoLT_Learning.pdf (last accessed August 2015). 1.1

[Rooted, 2015] Rooted (2015). Rootedcon. http://www.rootedcon.com (last accessed August 2015). 6.3

[Ryck et al., 2011] Ryck, P. D., Desmet, L., Philippaerts, P., and Piessens, F. (2011). A security analysis of next generation web standards. Technical report, The European Network and Information Security Agency (ENISA). https://lirias.kuleuven.be/bitstream/123456789/317385/1/NG_Web_Security.pdf (last accessed August 2015). 2.2.1.3

[Sahoo et al., 2010] Sahoo, J., Mohapatra, S., and Lath, R. (2010). Virtualization: A survey on concepts, taxonomy and associated security issues. In *Computer and Network Technology (ICCNT), 2010 Second International Conference on*, pages 222–226. 3.4

[Salem et al., 2008] Salem, M. B., Hershkop, S., and Stolfo, S. J. (2008). A survey of insider attack detection research. Technical report, Columbia University, New York, NY (USA). http://www.dtic.mil/dtic/tr/fulltext/u2/a519455.pdf (last accessed August 2015). 2.1.3

[Samuel, 1959] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229. http://researcher.watson.ibm.com/researcher/files/us-beygel/samuel-checkers.pdf (last accessed August 2015). 2.4.2

[Sandip et al., 2012] Sandip, A. S., Ajit, A. M., and Bapusaheb, J. D. (2012). An improved approach for signature and anomaly based intrusion detection and prevention. In *Proceedings on International Conference in Computational Intelligence (IJCA)*. http://www.docin.com/p-463259849.html (last accessed August 2015). 2.2.1.3

[Sangkatsanee et al., 2011a] Sangkatsanee, P., Wattanapongsakorn, N., and Charnsripinyo, C. (2011). Practical real-time intrusion detection using machine learning approaches. *Computer Communications*, 34(18):2227–2235. http://dx.doi.org/10.1016/j.comcom.2011.07.001 (last accessed August 2015). 2.4.2

[Sangkatsanee et al., 2011b] Sangkatsanee, P., Wattanapongsakorn, N., and Charnsripinyo, C. (2011). Practical real-time intrusion detection using machine learning approaches. *Computer Communications*, 34(18):2227–2235. http://dx.doi.org/10.1016/j.comcom.2011.07.001 (last accessed August 2015). 5.1

[Sangster et al., 2009] Sangster, B., O'Connor, T. J., Cook, T., Fanelli, R., Dean, E., Adams, W. J., Morrell, C., and Conti, G. (2009). Toward instrumenting network warfare competitions to generate labeled datasets. In *Proceedings of the* 2nd *Conference on Cyber Security Experimentation and Test*, CSET'09, pages 9–9. http://dl.acm.org/citation.cfm?id=1855481.1855490 (last accessed August 2015). 3.1.2

[Saniee Abadeh et al., 2007] Saniee Abadeh, M., Habibi, J., Barzegar, Z., and Sergi, M. (2007). A parallel genetic local search algorithm for intrusion detection in computer networks. *Engineering Applications of Artificial Intelligence*, 20(8):1058–1069. http://dx.doi.org/10.1016/j.engappai.2007.02.007 (last accessed August 2015). 2.4.2

[SANS Institute, 2015] SANS Institute. Critical security controls. http://www.sans.org/critical-security-controls (last accessed August 2015). 1.2

[Scarfone and Mell, 2007] Scarfone, K. and Mell, P. (2007). Guide to intrusion detection and prevention systems. Technical Report SP 800-94, NIST. http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf (last accessed August 2015). 2.2, 2.2.1.1, 2.2.1.2

[Schnackenberg and Tung, 1999] Schnackenberg, D. and Tung, B. (1999). The common intrusion detection framework. http://gost.isi.edu/cidf (last accessed August 2015). 2.2.2

[Scholte et al., 2011] Scholte, T., Balzarotti, D., and Kirda, E. (2011). Quo vadis? A study of the evolution of input validation vulnerabilities in web applications. In *Proceedings of Financial Cryptography*, pages 284–298. http://iseclab.org/papers/vuln_fcds.pdf (last accessed August 2015). 1.2

[Schwartz et al., 2010] Schwartz, E. J., Avgerinos, T., and Brumley, D. (2010). All you ever wanted to know about dynamic taint analysis and forward symbolic execution (but might have been afraid to ask). In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 317–331, Oakland, CA (USA). http://users.ece.cmu.edu/~ejschwar/papers/oakland10.pdf (last accessed August 2015). 1.2

[Scully, 2015] Scully, Peter M. D. (2015). CSIC 2010 HTTP dataset in CSV format (for weka analysis). Intelligent Robotics Research Group, Dept. of Computer Science Aberystwyth University, Ceredigion, Wales. http://users.aber.ac.uk/pds7/csic_dataset/csic2010http.html (last accessed August 2015). 3.5, 6.2

[Seo et al., 2004] Seo, J., Kim, H.-S., Cho, S., and Cha, S. (2004). Web server attack categorization based on root causes and their locations. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)*, ITCC '04, pages 90–96, Washington, DC, USA. http://priyono.lecture.ub.ac.id/files/2015/03/Web-Server-Attack-Categorization-based-on-Root-Causes-and-Their-Locations.pdf (last accessed August 2015). 2.1.3

[Sequeira and Zaki, 2002] Sequeira, K. and Zaki, M. (2002). ADMIT: anomaly-based data mining for intrusions. In *Proceedings of the* 8*th* *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 386–395, New York, NY, USA. http://www.cs.rpi.edu/~zaki/PaperDir/SIGKDD02-admit.pdf (last accessed August 2015). 2.4.2

[Shannon, 2001] Shannon, C. E. (2001). A mathematical theory of communication. *SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55. http://doi.acm.org/10.1145/584091.584093 (last accessed August 2015). 5.4.2.1

[Shelly and Frydenberg, 2011] Shelly, G. B. and Frydenberg, M. (2011). *Web 2.0: Concepts and Applications*. Course Technology. Cengage Learning. 1.1

[Shiravi et al., 2012] Shiravi, A., Shiravi, H., Tavallaee, M., and Ghorbani, A. A. (2012). Towards developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357–374. http://www.sciencedirect.com/science/article/pii/S0167404811001672 (last accessed August 2015). 3.1.2

[Shrivastava and Jain, 2011] Shrivastava, S. K. and Jain, P. (2011). Effective anomaly based intrusion detection using rough set theory and support vector machine. *International Journal of Computer Applications*, 18(3):35–41. http://www.ijcaonline.org/volume18/number3/pxc3872906.pdf (last accessed August 2015). 2.4.2

[Shyu et al., 2003] Shyu, M.-l., Chen, S.-c., Sarinnapakorn, K., and Chang, L. (2003). A novel anomaly detection scheme based on principal component classifier. In *Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the 3rd IEEE International Conference on Data Mining (ICDM'03)*, pages 172–179. http://www.dtic.mil/dtic/tr/fulltext/u2/a465712.pdf (last accessed August 2015). 2.3.2.2

[Simon, 2013] Simon, D. (2013). *Evolutionary Optimization Algorithms*. John Wiley & Sons. 2.3.2.1

[Sivatha Sindhu et al., 2012] Sivatha Sindhu, S. S., Geetha, S., and Kannan, A. (2012). Decision tree based light weight intrusion detection using a wrapper approach. *Expert Systems with Applications*, 39(1):129–141. http://dx.doi.org/10.1016/j.eswa.2011.06.013 (last accessed August 2015). 2.3.2.1, 2.4.2

[Smaha, 1988] Smaha, S. (1988). Haystack: An intrusion detection system. In *Proceedings of the 4th Aerospace Computer Security Applications Conference*, pages 37–44. http://homeostasis.scs.carleton.ca/~soma/id-2007w/readings/smaha-haystack.pdf (last accessed August 2015). 2.4.1.1

[Sommer and Paxson, 2010] Sommer, R. and Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 305–316, Washington, DC, USA. http://dx.doi.org/10.1109/SP.2010.25 (last accessed August 2015). 1.3, 2.2, 3.1, 4.9, 6.1

[Song et al., 2009] Song, Y., Keromytis, A. D., and Stolfo, S. J. (2009). Spectrogram: A mixture-of-Markov-chains model for anomaly detection in web traffic. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*. http://academiccommons.columbia.edu/download/fedora_content/download/ac:145295/CONTENT/ndss_09_07.pdf (last accessed August 2015). 2.3.1.1, 2.4.1.2, 4.3, 5.4.2.2

[Sriraghavan and Lucchese, 2008] Sriraghavan, R. and Lucchese, L. (2008). Data processing and anomaly detection in web-based applications. In *Proceedings of IEEE Workshop on Machine Learning for Signal Processing (MLSP)*, pages 187–192. 2.4.1.1, 4.5.1, 4.7, 1

[Sriraghavan, 2008] Sriraghavan, R. G. (2008). Data processing and anomaly detection in web-based applications. Master's thesis, Oregon State University (USA). http://ir.library.oregonstate.edu/xmlui/bitstream/handle/1957/8176/GaarudapuramSriraghavan_Rajagopal_Masters_Thesis.pdf?sequence=1 (last accessed August 2015). 2.3.1

[Stuttard and Pinto, 2007] Stuttard, D. and Pinto, M. (2007). *The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws*. Wiley Publishing, Inc., Indianapolis (USA). 1.1, 1.1

[Sung and Mukkamala, 2003] Sung, A. H. and Mukkamala, S. (2003). Identifying important features for intrusion detection using support vector machines and neural networks. In *Proceedings of the 2003 Symposium on Applications and the Internet*, SAINT '03, pages 209–216, Washington, DC, USA. http://dl.acm.org/citation.cfm?id=827273.829224 (last accessed August 2015). 2.4.2

[Tankard, 2011] Tankard, C. (2011). Advanced Persistent threats and how to monitor and deter them. *Network Security*, 2011(8):16–19. http://dx.doi.org/10.1016/s1353-4858(11)70086-1 (last accessed August 2015). 1.1

[Tavallaee et al., 2010] Tavallaee, M., Stakhanova, N., and Ghorbani, A. A. (2010). Toward credible evaluation of anomaly-based intrusion-detection methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(5):516–524. http://www.cs.unb.ca/~natalia/SMC.pdf (last accessed August 2015). 3.1, 6.1

[Tax and Duin, 2004] Tax, D. and Duin, R. (2004). Support vector data description. *Machine Learning*, 54(1):45–66. http://dx.doi.org/10.1023/B:MACH.0000008084.60811.49 (last accessed August 2015). 2.3.2.1

[The Shmoo Group, 2011] The Shmoo Group (2011). Defcon. http://cctf.shmoo.com (last accessed August 2015). 3.1.2

[Tombini et al., 2004] Tombini, E., Debar, H., Me, L., and Ducasse, M. (2004). A serial combination of anomaly and misuse idses applied to HTTP traffic. In *Proceedings of the 20th Annual Computer Security Applications Conference*, ACSAC '04, pages 428–437, Washington, DC, USA. https://acsac.org/2004/papers/60.pdf (last accessed August 2015). 2.2.1.3

[Tsai et al., 2009] Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., and Lin, W.-Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994–12000. http://www.sciencedirect.com/science/article/pii/S0957417409004801 (last accessed August 2015). 2.4, 2.4.2

[Tsang et al., 2007] Tsang, C.-H., Kwong, S., and Wang, H. (2007). Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. *Pattern Recognition*, 40(9):2373–2391. http://www.tongji.edu.cn/~hanliwang/papers/J10_PR_2007.pdf (last accessed August 2015). 2.3.2.1

[University of California, 1999] Irvine, University of California. (1999). KDD cup 1999 data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (last accessed August 2015). 2.3.2.2

[Valdes and Skinner, 2000] Valdes, A. and Skinner, K. (2000). Adaptive, model-based monitoring for cyber attack detection. In *Recent Advances in Intrusion Detection*, volume 1907 of *Lecture Notes in Computer Science*, pages 80–93. Springer Berlin Heidelberg. http://www.sdl.sri.com/papers/adaptbn/adaptbn.pdf (last accessed August 2015). 2.4.2

[Valeur et al., 2005] Valeur, F., Mutz, D., and Vigna, G. (2005). A Learning-based Approach to the Detection of SQL Attacks. In *Proceedings of the Second International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA'05, pages 123–140, Berlin, Heidelberg.

http://www.cs.ucsb.edu/~vigna/publications/2005_valeur_mutz_vigna_dimva05.pdf (last accessed August 2015). 4.3

[Van Laarhoven and Aarts, 1987] Van Laarhoven, P. and Aarts, E. (1987). *Simulated Annealing: Theory and Applications*. Springer Science & Business Media. 2.3.2.1

[Varadarajan, 2012] Varadarajan, G. K. (2012). Web application attack analysis using Bro IDS. Technical report, SANS Institute. InfoSec Reading Room. http://www.sans.org/reading-room/whitepapers/detection/web-application-attack-analysis-bro-ids-34042 (last accessed August 2015). 2.2.1.3

[Wang et al., 2006] Wang, K., Parekh, J. J., and Stolfo, S. J. (2006). Anagram: A Content Anomaly Detector Resistant to Mimicry Attack. In *Proceedings of the* 9*th* *international conference on Recent Advances in Intrusion Detection*, RAID'06, pages 226–248, Berlin, Heidelberg. http://academiccommons.columbia.edu/download/fedora_content/download/ac:110542/CONTENT/cucs-020-06.pdf (last accessed August 2015). 2.3.1.1

[Wang and Stolfo, 2004] Wang, K. and Stolfo, S. J. (2004). Anomalous payload-based network intrusion detection. In *Proceedings of RAID04*. http://www.covert.io/research-papers/security/PAYL-AnomalousPayload-basedNetworkIntrusionDetection.pdf (last accessed August 2015). 2.3.1.1

[Wang and Battiti, 2006] Wang, W. and Battiti, R. (2006). Identifying intrusions in computer networks with principal component analysis. In *Proceedings of the First International Conference on Availability, Reliability and Security*, International Conference on Availability, Reliability and Security (ARES'06), pages 270–279, Washington, DC, USA. http://rtm.science.unitn.it/~battiti/archive/ares2006.pdf (last accessed August 2015). 2.3.2.2

[Wang et al., 2004] Wang, W., Guan, X., and Zhang, X. (2004). A novel intrusion detection method based on principle component analysis in computer security. In *Advances in Neural Networks - ISNN 2004*, volume 3174 of *Lecture Notes in Computer Science*, pages 657–662. http://link.springer.com/chapter/10.1007/978-3-540-28648-6_105 (last accessed August 2015). 2.3.2.2

[Warrender et al., 1999] Warrender, C., Forrest, S., and Pearlmutter, B. (1999). Detecting Intrusions Using System Calls: Alternative Data Models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy, 1999.*, pages 133–145. http://wenke.gtisc.gatech.edu/ids-readings/system_call_models.pdf (last accessed August 2015). 2.4.1.2

[WASC, 2004] WASC (2004). Web Security Glossary. Technical report, Web Application Security Consortium. http://www.webappsec.org/projects/glossary/v1/wasc_glossary_02262004.pdf (last accessed August 2015). 2.2.3

[WASC, 2006] WASC (2006). Web Application Firewall Evaluation Criteria. Technical Report Version 1.0, Web Application Security Consortium. https://files.pbworks.com/download/2jyXbiH7Lf/webappsec/13247061/wasc-wafec-v1.0.pdf (last accessed August 2015). 2.2.3, 4.3

[WASC, 2008] WASC (2008). Web application security statistics. http://projects.webappsec.org/f/WASS-SS-2008.pdf (last accessed Oct. 2014). 2.2

[WASC, 2010] Web Application Security Consortium (2010). Threat Classification v2.0. Technical report. http://projects.webappsec.org/w/page/13246978/ThreatClassification (last accessed August 2015). 2.1.3

[Wassermann and Su, 2008] Wassermann, G. and Su, Z. (2008). Static detection of cross-site scripting vulnerabilities. In *Proceedings of International Conference on Software Engineering (ICSE)*, pages

171–180. http://rosaec.snu.ac.kr/meet/file/20090204paperd.pdf (last accessed August 2015). 1.2

[Wharton, 2009] Wharton, U. o. P. (2009). A world transformed: What are the top 30 innovations of the last 30 years? http://knowledge.wharton.upenn.edu/article/a-world-transformed-what-are-the-top-30-innovations-of-the-last-30-years/ (last accessed August 2015). 1.1

[Wolpert and Macready, 2005] Wolpert and Macready (2005). Coevolutionary Free Lunches. *IEEE Transactions on Evolutionary Computation*, 9(6):721–735. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.2425&rep=rep1&type=pdf (last accessed August 2015). 5.6.4

[Wolpert, 1996] Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390. https://www.researchgate.net/profile/David_Wolpert/publication/2755783_The_Lack_of_A_Priori_Distinctions_Between_Learning_Algorithms/links/54242c890cf238c6ea6e973c?origin=publication_list&ev=pub_srch_pub_xdl (last accessed August 2015). 5.3, 5.6.4

[Wolpert, 2001] Wolpert, D. H. (2001). The supervised learning no-free-lunch theorems. In *Proceedings of 6th Online World Conference on Soft Computing in Industrial Applications*, pages 25–42. http://www.no-free-lunch.org/Wolp01a.pdf (last accessed August 2015). 5.3

[Wondracek et al., 2010] Wondracek, G., Holz, T., Platzer, C., Kirda, E., and Kruegel, C. (2010). Is the internet for porn? an insight into the online adult industry. In *Proceedings of Workshop on the Economics of Information Security (WEIS)*. http://iseclab.org/papers/weis2010.pdf. 1.1

[Wood, 2014] Wood, P. (2014). 2014 internet security threat report. Technical Report 19, Symantec Corporation.

http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf (last accessed August 2015). 1.1, 2.1.3, 3.1.1

[Wu et al., 2007] Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J., and Steinberg, D. (2007). Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37. http://www.cs.umd.edu/~samir/498/10Algorithms-08.pdf (last accessed August 2015). 2.4.2, 5.3, 5.3.3

[Wurzinger et al., 2009] Wurzinger, P., Platzer, C., Ludl, C., Kirda, E., and Kruegel, C. (2009). Swap: Mitigating xss attacks using a reverse proxy. In *Proceedings of the 2009 ICSE Workshop on Software Engineering for Secure Systems*, IWSESS'09, pages 33–39, Washington, DC, USA. http://www.iseclab.org/papers/swap.pdf (last accessed August 2015). 4.3

[Xydas et al., 2008] Xydas, I., Miaoulis, G., Bonnefoi, P.-F., Plemenos, D., and Ghazanfarpour, D. (2008). Using an evolutionary neural network for web intrusion detection. In *Proceedings of the* 26th *IASTED International Conference on Artificial Intelligence and Applications*, AIA '08, pages 258–265, Anaheim, CA, USA. http://users.teiath.gr/yxydas/Paper3_xydas.pdf (last accessed August 2015). 2.4.2

[Yang et al., 2010] Yang, Y., Jiang, D., and Xia, M. (2010). Using improved GHSOM for intrusion detection. *Journal of Information Assurance and Security (JIAS)*, 5(3):232–239. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.182.6219&rep=rep1&type=pdf (last accessed August 2015). 2.3.2.1, 2.4.2

[Ye et al., 2002] Ye, N., Emran, S. M., Chen, Q., and Vilbert, S. (2002). Multivariate statistical analysis of audit trails for host-based intrusion detection. *IEEE Transactions on Computers*, 51(7):810–820. http://dl.acm.org/citation.cfm?id=627211 (last accessed August 2015). 2.4.1.1

[Ye et al., 2004] Ye, N., Zhang, Y., and Borror, C. M. (2004). Robustness of the Markov-chain model for cyber-attack detection. *IEEE Transactions on Reliability*, 53(1):116–123. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1282169&abstractAccess=no&userType=inst (last accessed August 2015). 2.4.1.2

[Yeung and Ding, 2003] Yeung, D.-Y. and Ding, Y. (2003). Host-based intrusion detection using dynamic and static behavioral models. *Pattern Recognition*, 36(1):229–243. http://repository.ust.hk/dspace/bitstream/1783.1/2495/1/yeung.pr2003.pdf (last accessed August 2015). 2.4.1.2

[Zanero, 2006] Zanero, S. (2006). *Unsupervised Learning Algorithms for Intrusion Detection*. PhD thesis, Dipartimento di Elettronica e Informazione. Politecnico di Milano (Italy). http://home.dei.polimi.it/zanero/papers/tesi_zanero_online.pdf (last accessed August 2015). 2.4.2

[Zanero and Savaresi, 2004] Zanero, S. and Savaresi, S. M. (2004). Unsupervised learning techniques for an intrusion detection system. In *Proceedings of the 2004 ACM symposium on Applied computing*, SAC '04, pages 412–419, New York, NY, USA. http://home.dei.polimi.it/zanero/papers/IDS-SAC.pdf (last accessed August 2015). 2.4.2

[Zargar and Baghaie, 2012] Zargar, G. R. and Baghaie, T. (2012). Category-Based Intrusion Detection Using PCA. *Journal of Information Security*, 3(4):259–271. http://www.scirp.org/journal/PaperDownload.aspx?DOI=10.4236/jis.2012.34033 (last accessed August 2015). 2.3.2.2

[Zhang and White, 2007] Zhang, L. and White, G. B. (2007). Anomaly detection for application level network attacks using payload keywords. In *Proceedings of the IEEE Symposium on Computational Intelligence in Security and Defense Applications CISDA 2007*, pages 178–185. 2.3.2.2

Glossary

Term	**Meaning**
AIS	Artificial Immune System
ALAD	Application Layer Anomaly Detector
ANN	Artificial Neural Network
APT	Advanced Persistent Threat
CART	Classification And Regression Tree
CIDF	Common Intrusion Detection Framework
CFS	Correlation Feature Selection
CR	Carriage Return
CSIC	Consejo Superior de Investigaciones Científicas
CSV	Comma-Separated Values
CTF	Capture The Flag
DNS	Domain Name System
DR	Detection Rate
DT	Decision Tree
ECML/PKDD	European Conference on Machine Learning and Principles/ Practice of Knowledge Discovery in Databases
ENISA	European Union Agency for Network and Information Security
EP	Entry Point
GeFS	Generic Feature Selection
GHSOM	Growing Hierarchical Self-Organising Map
FN	False Negative
FP	False Positive
FPR	False Positive Rate
FTP	File Transfer Protocol
HIDS	Host-based Intrusion Detection System
HMM	Hidden Markov Model
HPP	HTTP Parameter Pollution
HTTP	Hypertext Transfer Protocol
IDES	Intrusion Detection Expert System

Term	**Meaning**
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
IT	Information Technology
ITU	International Telecommunication Union
LERAD	LEarning Rules for Anomaly Detection
LDAP	Lightweight Directory Access Protocol
LF	Line Feed
MAP	Maximum A-posteriori Probability
ML	Machine Learning
mRMR	Minimal Redundancy Maximal Relevance
MPT	Minimum Processing Time
NBD	Normal Behavior Description
NIDES	Next-Generation Intrusion Detection Expert System
NIDS	Network Intrusion Detection Systems
NIST	National Institute of Standards and Technology
NSGA	Non-dominated Sorting Genetic Algorithm
NTR	Number of Test Requests
NTrR	Number of Training Requests
OS	Operating System
OSI	Open System Interconnection
OWASP	Open Web Application Security Project
PAPAS	PArameter Pollution Analysis System
PCA	Principal Component Analysis
PHAD	Packet Header Anomaly Detector
PHP	Hypertext Preprocessor
PT	Processing Time
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
RMHC	Random Mutation Hill Climbing
ROC	Receiver Operating Characteristic
SA	Simulated Annealing
SIEM	Security Information and Event Management
SOM	Self-Organizing Maps
SQL	Structured Query Language
SSI	Server-Side Include
SVDD	Support Vector Data Description
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive

Term	**Meaning**
URI	Uniform Resource Identifier
VPN	Virtual Private Network
WAF	Web Application Firewall
WASC	Web Application Security Consortium
XSS	Cross-site Scripting

Alphabetical index