Universidad de Deusto

Detección de tráfico de control de botnets modelizando el flujo de los paquetes de red

Félix Brezo Fernández

Bilbao, agosto de 2013

Universidad de Deusto

Detección de tráfico de control de botnets modelizando el flujo de los paquetes de red

Tesis doctoral presentada por Félix Brezo Fernández
dentro del Programa de Doctorado en
Ingeniería para la Sociedad de la Información y Desarrollo Sostenible

Dirigida por el
Dr. Pablo García Bringas
y por el
Dr. Igor Santos Grueiro

El doctorando Los directores

Bilbao, agosto de 2013

A los que he decepcionado ya
y a los que estoy por decepcionar aún,
porque la verdadera decepción
es la cara oscura de un lujo solamente reservado
para aquellos que alguna vez creyeron en mí con pasión.

Resumen

La historia del *malware* va intrínsecamente ligada al desarrollo de las tecnologías de la información y de la computación. Sin embargo, no es menos cierto que las motivaciones de los escritores de aplicaciones maliciosas han evolucionado. La fama y la gloria anhelada por los primeros programadores ha dado paso a una industria completamente organizada en torno a las muy variadas aplicaciones delictivas de un espacio cada vez más democratizado y dependiente del buen funcionamiento de las infraestructuras tecnológicas que lo dan soporte.

Es en este contexto donde surge la amenaza de las *botnets*: colecciones de equipos infectados con *malware* que son controlados remotamente por sus operadores, también conocidos como *botmasters*. Los mecanismos que puede llevar a cabo el *botmaster* para monetizar las infecciones son muy variados: desde el envío masivo de correo basura, hasta el robo de credenciales bancarias, la ejecución de ataques de denegación de servicio distribuida y la sustracción de información sensible de las empresas. La principal novedad que presenta esta modalidad de crimen organizado es la sencillez con la que las nuevas variantes de *malware* son concebidas por lo que se hace necesario que dichas aplicaciones tengan una curva de aprendizaje no excesivamente pronunciada.

Así, una de las posibilidades más explotadas recientemente es la utilización de paneles de *Command & Control* instalados en servidores web y accesibles desde un navegador. Es así como la necesidad da lugar a paneles que terminan por convertirse en avanzadas herramientas de gestión de los nodos infectados desde las que enviar órdenes y distribuir las tareas a realizar. De esta manera, y como complemento de los enfoques tradicionales que detectan la infección a partir de la identificación de ejecutables en los equipos de usuario, se propone un modelo de

detección que utilice atributos propios de los paquetes salientes desde/hacia un equipo para identificar la existencia de una conexión maliciosa.

Para ello, se hace necesario mejorar los sistemas de caracterización de las conexiones actuales mediante la definición de una metodología para el análisis de muestras de tráfico. El objetivo es que estas puedan ser utilizadas para la identificación de amenazas relativas a las *botnets* y, concretamente, a los canales de *Command & Control* que las gestionan. El proceso ha requerido del desarrollo de un modelo vectorial de representación de las conexiones que permita un análisis de los paquetes de acuerdo a la identificación del problema como una tarea de clasificación supervisada sobre la que aplicar diferentes algoritmos empleados en otras áreas de conocimiento dentro de la inteligencia artificial. Así, con el horizonte de avanzar el estado del arte en una línea que permita mejorar la forma en que la amenaza de las *botnets* es estudiada, se han facilitado metodologías aplicables al desarrollo actual de las *botnets* y que permitan ampliar las capacidades de los detectores de *malware* tradicionales.

Abstract

Malware history has been inherently linked to the development of the information technologies and Computer Science. However, it is not less true that the initial motivations of malware writers have evolved significantly from the fame and the glory seeked by the first malware programmers. The current situation is no longer the same: large organized ciberdellictive groups have been extended in the mercy However, it is not less certain that the ultimate motivations of the malicious applications writers have also evolved: from the fame and glory seeked by the first computer programmers, the times have led to the proliferation of a fully organized industry capable of exploiting the diverse criminal trends to be exploited in a cyberspace more democratized and dependant of the technological infraestructures that give support to it.

This is the context where the threat of botnets emerges: the management of huge collections of malware-infected machines remotely controlled by their operators (or botmasters) has become a business. The mechanisms that can develop a botmaster to get economical benefit from the infections can vary a lot: from the massive delivery of spam, to bank credentials steal, the performance of Distributed Denial of Service attacks or the obtention of sensitive information from the companies, bringing to a new level the concept of industrial espionage. The main nouvealty that presents this new variation of organized crime is the easeness with which the new variants of malware are being conceived including software with softened learning curves.

Thus, one of the most exploited posibilities recently is the usage of Command & Control installed on web servers and easily accesible using a browser. That is the way in which the need of advanced management tools has lead to the usafe of HTTP botnet related malware solutions. Because of the aforementioned

and as a complement to the traditional malware detection approaches capable of identifying infections in the user devices, we propose a detection model using the attributes of the outgoing and incoming packets from a host to identify the existence of a malicious connection become from a botnet.

So as to achieve this goal, it is neccessary to improve the current characterization systems of the connections by means of the definition of a methodology capable of performing traffic samples analysis that might belong to the Command & Control channel from a botnet. With such objective, we are going to develop firstly a vectorial representation model for connections. This model will allow us to conceive the task of analysing the packets in a connection as a supervised classification problem in which to apply the different algorithms employed in other fields of knowledge linked to artificial intelligence (AI).

In any case, the final goal is to advance the stat eof the art in a line that would allow us to improve the way in which the threat of botnets is studied, providing a methodology fully applicable to the current development of this networks and permitting to improve the traditional capabilities of the signature-based malware detection solutions.

Agradecimientos

Mucha gente piensa que una tesis es la obra de una vida. Recuerdo ahora unas palabras de un profesor del programa de doctorado de la Universidad de Deusto, Mario Piattini, quien, quizás para descargar algo de responsabilidad, nos definía allá por el mes de noviembre de 2012 cómo teníamos que afrontar nuestra tesis doctoral: «Vuestra tesis no es la obra de vuestras vidas, si no la primera obra del resto de vuestras vidas». Sea como fuere y sea lo que fuere, el proceso se ha visto culminado y aquí estamos redactando unos agradecimientos que se tienen que remontar mucho antes siquiera de la concepción del primer boceto de documento de lo que es el documento que hoy tienes (y perdóname el tuteo anónimo lector) entre manos.

No puedo negar que sería poco coherente que la historia de los agradecimientos de esta tesis empezara con la elaboración del manuscrito final. La historia ha de remontarse algún tiempo atrás, a algún lugar de la Facultad de Ingeniería (entonces aún oficialmente ESIDE) allá por el año 2009. Uno, que entonces era más joven y (aún más) inexperto, de salto en salto por los despachos de la facultad terminaba en el S3lab. Un laboratorio amplio, con un graffiti al fondo (peculiar historia la de la pintada) que lo llenaba de carácter y con unos también más jóvenes ayudantes de investigación. Recuerdo pronunciar unas primeras palabras dubitativas: «Buenas tardes... ¿Dónde puedo encontrar a Pablo?». Un sonriente y aún desconocido Borja Sanz me señalaba un despacho de tono pistacho que me había pasado a mano derecha. Primera torpeza.

Recuerdo la conversación con Pablo como si hubiera sido hace nada y recuerdo haberle contado con ganas por qué quería hacer seguridad informática. No sé si le habré convencido aún de lo en serio que me tomé aquella conversación, pero el caso es que Pablo se prestó a darme una oportunidad que no me podía

permitir el lujo de desaprovechar. Era el comienzo de la aventura. A partir de ahí, todo se sucedió con bastante rapidez. Me presentaron a un tal Igor Santos del que me avisó (también Borja) del que tendría que tener cuidado: tenía unas ideas extrañas que terminaba sacando... Vaya que sí. Lo que sé y lo que aprendí trabajando a su lado, es lo que me ha permitido llegar hasta donde estoy. Aprendí el método, aprendí a pegarme con cantidades ingentes de datos, aprendí a ser riguroso en la redacción y en la corrección de *papers*, aprendí a presentarlos (con más o menos éxito al principio, pero aprendí), aprendí lo duro que es tener que repetir los experimentos porque se te han escapado *pequeños* detalles en la construcción de representaciones... Hay pocas palabras que puedan expresar lo que, gratuitamente y quizás sin darse cuenta, me ayudó a crecer en estos años.

Son muchas las personas con las que he tenido el placer de compartir laboratorio. Mencionados Pablo, Borja e Igor, no me puedo olvidar de José (aunque para mí será siempre Sete). Con él he desarrollado gran parte de los últimos trabajos que han dado pie a esta tesis doctoral. Pero si me quedara solo con eso estaría dejando de lado lo más importante de todos esos ratos de *cacharreo*, horas de *botnets* para BRIANA (el TFM que compartimos en el Máster de Seguridad) y más horas aún de análisis de tráfico y revisión de *papers* y artículos. Esas cosas unen lo que unen.

Me quiero acordar también de Jorge. Además de un gran socio del Athletic y de ser un mediocre jugador de squash (aunque, desde luego, mejor que yo), me ha ayudado a darle un último empujón a esta tesis. Nos quedan proyectos por ahí pendientes.

No puedo dejar de acordarme también del resto de compañeros: de Juan y los partidos de pádel, de Javi y esos momentos compartidos programando *bots* en C a la carrera o escuchando grupos que jamás se me habría ocurrido poner a mí, de Iker y de su apoyo con la documentación y experimentos, de Carlos y los piques con su Baskonia (sé que eras tú el de los hackeos), de Mikel y las conversaciones sobre interfaces de usuario, de Patxi y su intercambio de links por las redes sociales, de los comentarios a la investigación del tío con más talento que conozco, Xabi Ugarte, de Xabi Cantero y su manejo indescriptible de las hojas de cálculo, Linux y Dvorak (buena herencia, Xabi), de Sen-

doa y una licencia de radioaficionado, de Aitor y su *bielsismo*, de Agustín y esos debates sobre el estado del mundo...

Ha sido mucha gente la que me ha apoyado y con la que me he encontrado a gusto durante muchos años en esta época universitaria... Pero también desde fuera. Es el momento de acordarme de mis padres y mi hermana que han apoyado cada idea que he tenido en estos 26 años, por loca, idealista o retorcida que pareciera. Me enseñaron a ser curioso, a cuestionarme las cosas y a tratar de ser lo más noble posible. Las palabras que de vez en cuando me repito delante del espejo tratan de recoger esa filosofía: «*Nobleza. Discreción. Esfuerzo.*». No dudéis de que lo seguiré a rajatabla...

Recuerdo también que, por una casualidad de esas que tiene la vida (mi hermana Olga lo saber mejor que nadie), terminé haciendo un Máster de Análisis de Inteligencia. Y ahí Yaiza fue un descubrimiento postrero. Es de esas personas con las que da gusto pensar, hablar, trabajar y mejorar porque compartes con ellas inquietudes y pensamientos sin a veces siquiera llegar a expresarlos: conexión *N-to-N*. Casualidad o destino, me considero muy afortunado.

En ese listado de personas que a uno le quedan para siempre después de tanto tiempo hay unas cuantas a las que debo una o varias disculpas por no haber conseguido sacar tiempo para ellas. Desde Nacho o Tomás, hasta Miquel, José María, Dani, Marta, Beñat, Jon, Diego o Imanol. Todos ellos han tenido que soportar fines de semana en los que uno se encuentra desganado y los pone a la cola de sus tareas a sabiendas de que hay que cuidar las amistades, y más aún, las mejores. Seguro que me dejo alguno importante pero, como ya dijo alguien por ahí: «*Son todos los que están, aunque no están todos los que son*».

Amigos... Este es mi techo en cuanto a formación universitaria. Con aciertos pero también con muchos errores de los que tocaba aprender. Para mí es la culminación de un ciclo desde el que comienzo una etapa nueva de mi vida que no queda en este documento, si no que llevaré conmigo allá donde termine. Muchas gracias. De veras.

Félix Brezo, doctorando

Índice general

Índice de figuras **xv**

Índice de tablas **xix**

1 La amenaza asimétrica con rostro de personaje histórico **1**

2 Introducción: un campo abonado para las *botnets* **7**
2.1 Relevancia del proyecto: un fenómeno global 8
2.2 Hipótesis de partida y objetivos 10
2.3 Metodología 11
2.3.1 El método científico 11
2.3.2 Fases en las que se divide el proyecto 15
2.3.3 Estructura de la tesis doctoral 17

3 Revisión bibliográfica **21**
3.1 Surgimiento de la industria del *malware* 22
3.1.1 Una breve historia del *malware* 22
3.1.2 Situación actual del *malware* 25
3.2 Tipos de *malware* 28
3.3 La amenaza de las *botnets* 36
3.3.1 Usos ciberdelictivos 37
3.3.2 Comportamiento general 41
3.3.3 Arquitectura típica de una *botnet* 42
3.3.3.1 La filosofía *pull* 45
3.3.3.2 La filosofía *push* 46
3.3.4 Evolución de los canales de *Command & Control* 47
3.3.4.1 IRC 47
3.3.4.2 P2P 49

3.3.4.3 HTTP/HTTPS 51
3.3.4.4 Nuevas tendencias: *malware* social y dispositivos móviles 52
3.4 Principales contramedidas y líneas defensivas 53
3.4.1 Técnicas de detección de *botnets* 55
3.4.1.1 Detección por medio de firmas 55
3.4.1.2 Detección de comportamiento cooperativo . 55
3.4.1.3 Detección de comportamiento ofensivo . . . 58
3.4.1.4 Detección del usuario final 58
3.4.2 Técnicas de desactivación 58
3.5 Sumario 60

4 Modelado del tráfico de red **63**
4.1 Flujos de información y tráfico de red 64
4.1.1 El proceso de estandarización 64
4.1.2 Abstracción del modelo 65
4.2 Obtención de representaciones 65
4.2.1 Construcción de representaciones de un paquete . . . 66
4.2.2 Construcción de representaciones de una cantidad indefinida de paquetes 68
4.2.2.1 Medidas de tendencia central 70
4.2.2.2 Medidas de dispersión 72
4.2.2.3 Generalización de la representación 72
4.2.3 Generación de conjuntos de datos procesables 73
4.3 Sumario 75

5 Metodología para la obtención de muestras **77**
5.1 Metodología de obtención de tráfico 78
5.1.1 Despliegue de los canales de *Command & Control* . . . 78
5.1.2 Preparación de los sujetos 78
5.2 Generación de tráfico benigno 80
5.3 Selección de muestras maliciosas 81
5.4 Procesamiento de las capturas 81
5.4.1 Preprocesamiento 83
5.4.2 Generación de ficheros auxiliares de procesamiento . 83
5.5 Limitaciones 84
5.5.1 Aspectos técnicos 85
5.5.2 El factor humano 87
5.6 Sumario 88

6 Detección de tráfico procedente de canales de *Command & Control* 89
6.1 Aprendizaje supervisado . . . 90
6.1.1 El método *Support Vector Machines* (SVM) . . . 91
6.1.2 Método *Bagging* . . . 93
6.1.3 Árboles de decisión . . . 94
6.1.3.1 Algoritmo C4.5 . . . 95
6.1.3.2 El método *Random Forest* . . . 96
6.1.4 Redes bayesianas . . . 97
6.1.4.1 Clasificador *Naïve Bayes* . . . 99
6.1.4.2 Algoritmo de aprendizaje K2 . . . 99
6.1.4.3 Método de aprendizaje *Hill Climbing* . . . 100
6.1.4.4 El método *Tree Augmented Naïve* (TAN) . . . 100
6.1.5 El algoritmo *K-Nearest Neighbours* (KNN) . . . 101
6.1.6 Perceptrones . . . 102
6.1.6.1 *Voted Perceptron* . . . 103
6.1.6.2 Multilayer Perceptron (MLP) . . . 103
6.2 Metodología general . . . 103
6.2.1 Aplicación de la validación cruzada . . . 107
6.2.2 *Resampling* y balanceo de datos de entrenamiento . . . 108
6.3 Validación empírica . . . 108
6.3.1 Aplicación del modelo a conexiones representadas por sus paquetes . . . 109
6.3.1.1 Capacidad del modelo para detectar la presencia de tráfico procedente de una única *botnet* determinada . . . 111
6.3.1.2 Capacidad del modelo para detectar de forma genérica el tráfico de control procedente de varias *botnets* . . . 116
6.3.2 Aplicación del modelo a conexiones representadas por secuencias de *n-paquetes* . . . 118
6.3.2.1 Capacidad del modelo para detectar la presencia de tráfico procedente de una única *botnet* determinada . . . 118
6.3.2.2 Capacidad del modelo para detectar de forma genérica el tráfico de control procedente de varias *botnets* . . . 127
6.4 Implicaciones . . . 131
6.5 Sumario . . . 135

7 Conclusiones **137**
7.1 Síntesis de la validación del sistema 138
7.2 Resumen de los resultados 139
7.3 Aplicaciones de la investigación 140
7.4 Limitaciones 141
7.5 Líneas futuras de trabajo 142
7.6 Consideraciones finales 144
7.7 Otros méritos científicos 151

Bibliografía **153**

Apéndices **177**

A Grado de similitud de los paquetes relativos a las muestras de tráfico utilizadas **179**
A.1 Uso de métricas de similitud para representaciones vectoriales 180
A.1.1 Similitud del coseno: base teórica 180
A.2 Adaptación de las métricas al proceso de modelado 184
A.2.1 Ejecución de las comparaciones 185
A.2.2 Representación de la complejidad del tráfico de una familia 185
A.3 Sumario 188

B Fichas técnicas de las muestras de *malware* empleadas **189**
B.1 Flu 189
B.1.1 Características 190
B.1.2 Otras consideraciones y ficha técnica 193
B.2 Prablinha 193
B.2.1 Características 193
B.2.2 Otras consideraciones 198
B.3 Warbot 198
B.3.1 Características 199
B.3.2 Otras consideraciones 200

Índice alfabético **201**

Índice de figuras

2.1 Example of ontology extract 12
2.2 Cronograma de las tareas en que se ha dividido la tesis doctoral. 14

3.1 Evolución del número de muestras maliciosas almacenadas en la base de datos de AV-Test. *Fuente: AV-Test.* 26
3.2 Evolución del número de ataques provocados por *ransomware*. *Fuente: McAfee.* 27
3.3 Diferentes topologías para la implementación del canal de *Command & Control* de una *botnet*. 45

4.1 Comparación de los niveles propuestos por la pila OSI y la pila de protocolos TCP/IP. 65
4.2 Obtención de las características de un paquete. 67
4.3 Obtención de las características de un paquete. 69
4.4 Obtención de los intervalos temporales normalizados. 73
4.5 Muestra de un fichero *.arff* con representaciones de un paquete de longitud generado a partir de una muestra de tráfico de control de Warbot. 74

5.1 Example of ontology extract 82

6.1 Ejemplo de un clasificador SVM en un espacio bi-dimensional. 92
6.2 Un ejemplo de árbol de decisión. 95
6.3 Ejemplo de red bayesiana. 98
6.4 Ejemplo del funcionamiento de kNN en un espacio bidimensional para $k = 4$ vecinos. 101
6.5 Aplicación de la métrica de balanceo. 109

6.6 Evolución del porcentaje de acierto en la clasificación (*Acc.*) en base a la longitud de las secuencias utilizadas para el etiquetado de tráfico benigno o de *Command & Control* de las *botnets* analizadas: Flu, subfiguras 6.6(a) y 6.6(b); Prablinha, subfiguras 6.6(c) y 6.6(d); y Warbot, subfiguras 6.6(e) y 6.6(f). 119

6.7 Evolución de los índices de detección (TPR) en función de la longitud de las secuencias utilizadas para el etiquetado de tráfico benigno o de *Command & Control* de las *botnets* analizadas: Flu, subfiguras 6.7(a) y 6.7(b); Prablinha, subfiguras 6.7(c) y 6.7(d); y Warbot, subfiguras 6.7(e) y 6.7(f). 121

6.8 Evolución de la efectividad del etiquetado (PPV) en función de la longitud de las secuencias utilizadas para el etiquetado de tráfico benigno o de *Command & Control* de las *botnets* analizadas: Flu, subfiguras 6.8(a) y 6.8(b); Prablinha, subfiguras 6.8(c) y 6.8(d); y Warbot, subfiguras 6.8(e) y 6.8(f). . . . 123

6.9 Evolución del valor de la *f-measure* según la longitud de las secuencias utilizadas para el etiquetado de tráfico benigno o de *Command & Control* de las *botnets* analizadas: Flu, subfiguras 6.9(a) y 6.9(b); Prablinha, subfiguras 6.9(c) y 6.9(d); y Warbot, subfiguras 6.9(e) y 6.9(f). 124

6.10 Evolución del valor del área por debajo de la curva ROC (AUC) según la longitud de las secuencias utilizadas para el etiquetado de tráfico benigno o de *Command & Control* de las *botnets* analizadas: Flu, subfiguras 6.10(a) y 6.10(b); Prablinha, subfiguras 6.10(c) y 6.10(d); y Warbot, subfiguras 6.10(e) y 6.10(f). 126

6.11 Evolución del nivel de acierto en la clasificación (*Acc.*) según la longitud de las secuencias utilizadas para el etiquetado de tráfico genérico benigno o de *Command & Control* de las *botnets* analizadas. 127

6.12 Evolución de los índices de detección (TPR) obtenidos para la detección de muestras maliciosas según la longitud de las secuencias utilizadas para el etiquetado de tráfico genérico benigno o de *Command & Control* de las *botnets* analizadas. . 128

6.13 Evolución del valor de exactitud (PPV) según la longitud de las secuencias utilizadas para el etiquetado de tráfico genérico benigno o de *Command & Control* de las *botnets* analizadas. 129

6.14 Evolución del valor de la *f-measure* según la longitud de las secuencias utilizadas para el etiquetado de tráfico genérico benigno o de *Command & Control* de las *botnets* analizadas. . 130
6.15 Evolución del valor del área por debajo de la curva ROC (AUC) según la longitud de las secuencias utilizadas para el etiquetado de tráfico genérico benigno o de *Command & Control* de las *botnets* analizadas. 131

A.1 Determinación del signo del seno y del coseno del ángulo que forman diferentes vectores con el eje de abscisas (o eje X). 181
A.2 Determinación del signo del seno y del coseno del ángulo que forman diferentes vectores con el eje de abscisas (o eje X). 184
A.3 Funciones de distribución de las muestras de tráfico de *Command & Control* de las *botnets* analizadas en esta tesis doctoral: Flu, subfigura A.3(a); Prablinha, subfigura A.3(b); y Warbot, subfigura A.3(c). 186
A.4 Representación del grado de complejidad de las representaciones de las *botnets* Flu, Prablinha y Warbot. 188

B.1 Captura de pantalla del panel de control de la *botnet* Flu. . . 191
B.2 Captura de pantalla del panel de control de la *botnet* Prablinha.194
B.3 Llamadas públicas a los métodos de cifrado de Prablinha. . . 197
B.4 Algoritmo de cifrado de cadenas de texto de Prablinha. . . . 197
B.5 Captura de pantalla del panel de control de la *botnet* Warbot. 199

Índice de tablas

6.1 Resultados de detección de tráfico procedente de la *botnet* Flu empleando los atributos de paquetes individuales. 112
6.2 Resultados de detección de tráfico procedente de la *botnet* Prablinha empleando los atributos de paquetes individuales. 113
6.3 Resultados de detección de tráfico procedente de la *botnet* Warbot empleando los atributos de paquetes individuales. . . 115
6.4 Resultados generales de detección de tráfico etiquetado como genérico procedente de la *botnets* analizadas empleando los atributos de paquetes individuales. 117
6.5 Estimación de posibilidades de que una secuencia de paquetes correspondientes a una conexión determinada haya sido correctamente clasificada. 134

A.1 Ejemplos de los valores que se obtendrían a la hora de calcular la similitud del coseno de los vectores representados. Se entiende que el valor del ángulo α es de aproximadamente 30º.183
A.2 Matriz de resultados de la comparación de n muestras de tráfico. 185

B.1 Criptogramas obtenidos tras ser cifrados con el algoritmo por defecto de Prablinha. 198

«El mundo entero es un escenario. Y todo lo demás... es vodevil».

V, personaje principal del cómic *V de Vendetta* de Alan Moore (1953 – ...)

Capítulo

1

La amenaza asimétrica con rostro de personaje histórico

La historia del *hacktivismo* va ligada a la simbología de un personaje histórico bastante desconocido más allá de la tradición anglosajona. La efigie de Guy Fawkes la encontramos en las máscaras con las que una gran cantidad de activistas cubren sus rostros en los miles de comunicados y vídeos que se pueden encontrar en la red. El propio Guy —o Guido, como sería conocido en la Europa latina tras varios pasos por España e Italia— fue un conspirador católico inglés que sería hallado culpable de orquestar la conocida como la *Conspiración de la Pólvora* a principios del siglo XVII. Eran tiempos inciertos para una Europa colonialista salpicada por numerosas guerras de religión que enmascaraban intereses de un carácter mucho más terrenal. De hecho, el período que iba de finales del siglo XVI hasta mediados del siglo siguiente estaría marcada por uno de los conflictos bélicos más prorrogados de la historia: la conocida como *Guerra de los Ochenta Años* o *Guerra de Flandes*.

Esta contienda no se daría finalmente por concluida hasta bien entrado el siglo XVII con la firma el 30 de enero de 1648 del Tratado de Münster [Cas56]. Este acuerdo, como parte de la Paz de Westfalia [Ell99], establecía la independencia definitiva de las Siete Provincias Unidas neerlandesas

de la corona española después de un siglo bajo el control de la casa de los Habsburgo, que aún reinarían en España hasta el fallecimiento sin descendencia de Carlos II en 1700.

Décadas antes de estos sucesos, en el Reino de Escocia, María I Estuardo (católica) fue obligada a abdicar en su hijo el infante Jacobo de solo trece meses de edad. Así es como la reina de los escoceses, salpicada por los rumores de ordenar el asesinato de su segundo marido a manos del que terminaría siendo su tercer esposo, no volvería a ver jamás a su hijo. El infante Jacobo, educado en una estricta doctrina calvinista pese a haber sido bautizado católico como su madre, tendría que hacer frente años después a numerosos incidentes de índole religiosa a los que no permanecería ajeno.

Es este el contexto en el que Guy Fawkes, por su pasado militar y sus conocimientos de explosivos, asumiría el rol de *brazo ejecutor* del movimiento restauracionista católico en la corona británica tras haber buscado apoyos sin éxito en la España de Felipe III. En represalia por las duras medidas tomadas contra los católicos romanos tras la entronización del Rey Jacobo y como primera parte de su plan, los conspiradores pretendían volar el Palacio de Westminster, sede aún hoy de las dos cámaras del Parlamento del Reino Unidio en Londres, con todos los representantes de la corte dentro. Además, se contemplaba reclamar como heredera legítima de la corona tras el fallecimiento del monarca a sus hija Isabel con la que se esperaba volver al paraguas de Roma [Fra10].

Sin embargo, el Rey Jacobo VI de Escocia y I de Inglaterra sería advertido de la conspiración por Lord Monteagle, procediendo a dar la orden de vigilar los sótanos y subterráneos del Parlamento en donde Guy Fawkes sería descubierto el cinco de noviembre de 1605 junto con gran cantidad de barriles de pólvora y material incendiario. Tras ser torturado, Fawkes sería finalmente condenado a muerte en enero del año siguiente y desde entonces el día cinco de noviembre tiene lugar la festividad de la *Bonfire Night* o Noche de Guy Fawkes. En ella se celebra anualmente la salvación del rey y del Parlamento y sería obligatoria en el Reino Unido y todos los territorios al amparo del imperio británico hasta 1859. La historia perduraría por décadas como celebaración nacional británica dando lugar a toda una serie de manifestaciones en la cultura británica, tanto gastronómicas —desde las *jacked potatoes* asadas en la hoguera hasta el *bonfire toffee*— como populares a través de canciones o la quema de efigies de personajes célebres detestados por el pueblo.

Sin embargo, al margen de esta festividad, el carácter global del personaje de Guy Fawkes y de su historia no sería más difundido hasta la década

de los ochenta del siglo XX, cuando el escritor de novelas gráficas Alan Moore lo utilizaría para enmascarar a un personaje sediento de venganza de una de sus novelas gráficas más reconocidas: *V de Vendetta*.

La versión cinematográfica del cómic, dirigida por James McTeigue con un presupuesto de 54 millones de dólares, protagonizada por Natalie Portman y Hugo Weaving y distribuida por Warner Bros., asaltaría las carteleras del año 2006 con un éxito absoluto cosechando a su paso una legión de seguidores y más de 132 millones de dólares. En la gran pantalla y de forma similar a como ocurría en el original, V trataba de abrirse paso en el seno de una sociedad corrupta a caballo entre la venganza por su sufrimiento personal del pasado y los deseos de liberar a su país de un régimen totalitario. A pesar del distanciamiento de Alan Moore, quien terminaría por distanciarse del proyecto por discrepancias con respecto al desarrollo del guión [Gol06], este último elemento pasaba a cobrar una importancia aún mayor en los 131 minutos de grabación hasta el punto de reconvertir al personaje de V en un icono casi político muy cercano ideológicamente al anarquismo.

En cualquier caso, una de las imágenes más famosas y recordadas del film es el discurso televisado y cargado de ideología que V dio en la televisión de Londres. La tecnología con la que el gobierno había monopolizado los medios de comunicación para emitir transmisiones obligatorias con fines claramente propagandísticos, era reutilazada por V para dar más difusión a su mensaje:

«*¡Buenas tardes, Londres! Permitid que, primero, me disculpe por la interrupción. Yo, como muchos de ustedes, aprecio la comodidad de la rutina diaria, la seguridad de lo familiar, la tranquilidad de la monotonía. A mí, me gusta tanto como a vosotros. Pero con el espíritu de conmemorar los importantes acontecimientos del pasado —normalmente asociados con la muerte de alguien o el fin de alguna terrible y sangrienta batalla y que se celebran con una fiesta nacional—, he pensado que podríamos celebrar este cinco de noviembre —un día que, lamentablemente, ya nadie recuerda— tomándonos cinco minutos de nuestra ajetreada vida para sentarnos y charlar un poco.*

»*Hay, claro está, personas que no quieren que hablemos. Sospecho que, en este momento, estarán dando órdenes por teléfono, y que hombres armados ya vienen en camino. ¿Por qué? Porque mientras puedan utilizarán la fuerza. ¿Para qué el diálogo? Sin embargo, las palabras siempre conservarán su poder, las palabras hacen posible que algo tome significado y, si se escuchan, enuncian la verdad. Y la verdad es, que en este país, algo va muy mal, ¿no? Crueldad e injusticia, intolerancia y opresión. Antes tenías libertad para objetar, para pensar y decir lo que pensabas. Ahora, tienes censores y sistemas de vigilancia que nos coartan*

para que nos conformemos y nos convirtamos en sumisos.

»*¿Cómo ha podido ocurrir esto? ¿Quién es el culpable? Bueno, ciertamente, unos son más responsables que otros y tendrán que rendir cuentas. Pero, la verdad sea dicha, si estás buscando un culpable, sólo tienes que mirarte en el espejo. ¿Por qué lo hiciste? Porque tenías miedo. ¿Y quién no? Guerras, terror, enfermedades. Había una plaga de problemas que conspiraron para corromper vuestros sentidos y sorberos el sentido común. El temor pudo con vosotros y, presas del pánico, acudisteis al actual lider, Adam Sandler. Os prometió orden, os prometió paz. Y todo cuanto os pidió a cambio fue vuestra silenciosa y obediente sumisión.*

»*Anoche intenté poner fin a ese silencio. Anoche destruí el Old Bailey para recordar a este país lo que ha olvidado. Hace más de cuatrocientos años un gran ciudadano deseó que el cinco de noviembre quedara grabado en nuestra memoria. Su esperanza era hacer recordar al mundo que justicia, igualdad y libertad son algo más que palabras; son metas alcanzables. Así que si no abrís los ojos, si seguís ajenos a los crímenes de este gobierno, entonces os sugiero que permitáis que el cinco de noviembre pase sin pena ni gloria. Pero si veis lo que yo veo, si sentís lo que yo siento y si perseguís lo que yo persigo, entonces, os pido que os unáis a mí, dentro de un año, ante las puertas del parlamento, y juntos, les haremos vivir un cinco de noviembre que jamás, jamás nadie olvidará.*»

De forma reiterada, estas alusiones al cinco de noviembre —hasta en el propio nombre del personaje *V* cinco en números romanos— son continuas. Además, en dicho discurso televisado y a lo largo de los diez tomos del cómic, el protagonista emplea una máscara del propio Guy Fawkes con la que pretende reflejar que su lucha es también la del ciudadano anónimo, dueño de su libertad y de su destino. Quizás por los tintes revolucionarios del cómic, quizás por la defensa que hace el personaje de los derechos civiles, quizás por la insistencia permanente de la importancia del ciudadano anónimo como individuo parte de un colectivo, ha sido precisamente la efigie de este personaje histórico, Guy Fawkes, la que se ha convertido con posterioridad en uno de los elementos más característicos de muchos colectivos activistas mucho más allá de sus sentimientos de venganza personal.

Pero esta contextualización no tendría sentido si no la emplazamos en la era de Internet y de las comunicaciones. A día de hoy, Anonymous ha terminado por integrar estos símbolos como elemento fundamental de su propia identidad. Es así como es habitual encontrar comunicados, vídeos y fotografías en los que presuntos interlocutores de estos grupos se cubren el rostro con estas máscaras a imagen y semejanza del protagonista del cómic en lo que denominan su particular lucha contra el poder establecido.

Si a ello le sumamos el aderezo de una no siempre real sensación de se-

guridad y anonimato al opinar desde el otro lado de la pantalla —los casos de monitorización de la red desvelados recientemente pueden ser un buen ejemplo de ellos [Jon13, GM13, Flo13, Ram13, Rai12, Har12]—, damos con los ingredientes necesarios para alimentar una corriente en la que es cada vez más complejo identificar a otros estados como los rivales a batir. La democratización de la tecnología y del acceso a Internet y la complejidad de un mundo en el que podemos trabajar desde casa pero manejar un equipo afincado en un país extranjero que controla una red más móvil que nunca, nos llevan a una realidad en la que las amenazas cobrarán tintes más asimétricos que nunca: el enemigo será cada vez más transparente y más difícil de identificar como una única persona física, jurídica u organizativa porque se fundirá con instituciones y ciudadanos.

Esto no es ajeno a todos. Son muchos los colectivos que operan en la red que son conscientes de este cambio de paradigma y que ya lo están aprovechando como vehículo para promover sus propias causas, ya sean políticas, económicas o personales. Los que queremos un Internet libre pero seguro, justo y respetuoso, y en paz pero reivindicativo, tenemos la responsabilidad de adecuar las tecnologías de la seguridad para que todos podamos explorar —y explorar— al máximo las posibilidades que nos brinda la era de la información. Los tiempos han cambiado. Las amenazas y, específicamente, las ciberamenazas, también de modo que sigue siendo prioritario adaptarse a ellas para que la red se mantenga como un espacio de protección para el ciudadano.

«Cést en faisant nímporte quoi, quón deviant nímporte qui»
Traducción metafórica: «Es haciendo cualquier cosa, como nos convertimos en un cualquiera».

Rémi Gaillard, humorista francés (1975 – ...).

Capítulo 2

Introducción: un campo abonado para las *botnets*

El origen del término *botnet* surge como fusión de las voces inglesas *robot-networks*. A día de hoy las redes de ordenadores secuestrados, que representan cerca del 25 % de todos los intentos de conexión maliciosos, son una mezcla de amenazas estrechamente relacionadas con otras áreas de *malware*: pueden propagarse como los gusanos, sirven para la ocultación de virus informáticos y permiten además el control remoto de la máquina infectada por parte de terceros. Estas circunstancias, junto con otros pruebas relacionadas con la escritura de código colaborativa a partir de iniciativas *open-source* o de código abierto —como ocurre con muestras de diversas familias como SDBot, Agobot o variantes del propio Zeus, cuyo código está comentado incluso por diferentes autores— permiten la proliferación de una amplia gama de variantes, modificaciones y mutaciones de los *bots* en función de la finalidad específica a la que las quiera dedicar su autor.

Así, hoy en día, los tipos de ataques que pueden ser llevados a cabo empleando este tipo de redes son muy variados: desde los sencillos pero efectivos ataques distribuidos de denegación de servicio —o DDoS, por las siglas en inglés de *Distributed Denial of Service attacks*— [CJM] o el robo masivo de credenciales o información confidencial [BOB+], hasta ataques que explotan nuevas oportunidades de monetización de la infección como

el *click-frauding* [IH05], el robo de carteras de divisas virtuales como Bitcoin [BGB] o movimientos aún más sofisticados que se hacen con el control de máquinas industriales con los más diversos fines y de dudosa procedencia como Stuxnet [MMDC13]. Son sus capacidades y la utilización de estas herramientas por parte de un número cada vez mayor de cibercriminales por un lado y por organizaciones con diversos fines políticos por otro, las que han llevado a que las Estrategias Nacionales de Seguridad de las principales potencias empiecen a hablar de conceptos como las ciberamenazas o el ciberterrorismo en el marco de lo que algunos se atreven a bautizar como el quinto campo de batalla moderno más allá de los cuatro tradicionales —tierra, mar, aire y espacio—: el ciberespacio.

2.1 Relevancia del proyecto: un fenómeno global

Es así como el fenómeno de las *botnets* se ha convertido en un tema de interés también para los diferentes organismos dedicados a la seguridad informática a nivel mundial. Ya desde 2008, algunas agencias de seguridad como la Europol se han visto obligadas a preparar a sus profesionales para hacer frente a las amenazas que estas redes traen a la privacidad, el anonimato y la seguridad de empresas y de administraciones [Eur08]. Lo cierto es que las oportunidades que ofrecen estas complejas redes de ordenadores conectados y la potencial capacidad de cálculo que proporcionaría su gestión no han pasado por alto ni para los profesionales de la informática que se dedican a codificar soluciones distribuidas para enfrentarse a problemas complejos de computación —como el proyecto SETI [ACK+02]— ni tampoco para los creadores de *malware*, *spammers* y otros delincuentes cibernéticos.

Como se va a poder comprobar a lo largo de esta disertación, los escritores de *malware* están adaptando la forma en que presentan sus aplicaciones de cara a suavizar lo más posible la curva de aprendizaje del usuario final a la hora de administrar sus herramientas. Hasta tal punto es así que es tendencia hacer posible que los administradores de algunas de las más grandes redes de *bots* no tengan por qué ser grandes expertos en informática. Este es el caso de la *botnet* Mariposa, desarticulada en 2010 en el marco de la Operación Butterfly dirigida por la Guardia Civil y coordinada con diferentes organismos europeos y Panda Software [Cor10]. En ella, tres personas con escasa formación técnica —una de ellas de Balmaseda (Bizkaia)— fueron detenidas acusadas de ser los administradores de una red de cerca de cua-

tro millones de ordenadores infectados, la más grande detectada hasta la fecha.

De esta manera, las *botnets*, como colección de equipos infectados y manipulados de forma remota por criminales cibernéticos, van a evolucionar de forma natural en cuanto a complejidad constituyendo una amenaza cada vez mayor y más sólidamente asentada en el día a día. Es en este contexto en el que surge un nuevo concepto: HaaS. La terminología de HaaS (*Hacking as a Service*) o también conocida habitualmente como CaaS (*Crime as a Service*) [BS] deriva del SaaS (*Software as a Service*) utilizado en ingeniería del software. *SaaS* es un modelo de distribución de software donde se presta el servicio de mantenimiento y el soporte del mismo desde un único punto. Sabiendo esto, Haas o CaaS se pueden definir como variantes de SaaS pero orientados a dar cobertura a ciberdelincuentes.

En esta línea, un informe presentado por la empresa de seguridad informática Kaspersky, como parte del Technology Day 2011 [IT10], mostraba que los *botnets* son una de las amenazas más potentes en la actualidad para la estabilidad del ciberespacio, con perspectivas de seguir una «evolución dramática» hasta 2020, en gran parte debido a la incorporación de cada vez más y más dispositivos móviles con conexión a Internet y a una creciente potencia de cálculo. En la misma línea, Trend Micro creaba en enero de 2013 el primer mapa mundial de *botnets* en el que ya se identificaban más de seiscientos servidores de *Command & Control* que habían recibido conexiones de hasta medio millón de equipos infectados [Cla13]. Se trata por tanto de una amenaza real con amplias capacidades para ser monetizada con posterioridad. De hecho, una vez que la infección haya tenido lugar, el *bot-herder* o *botmaster* puede generar grandes cantidades de *spam*, lanzar ataques contra los sitios web más importantes o llevar a cabo campañas de fraude de publicidad en línea mediante la generación automática de clics en *banners* de su propiedad.

En las últimas fechas, además, está cobrando una importancia adicional el aprovechamiento de la potencia de procesamiento, almacenamiento y ancho de banda de los ordenadores personales y de las redes a las que estos están asociados. Hoy día, la mera disposición de tiempo de CPU o el acceso a determinados recursos de red están siendo explotados económicamente de formas anteriormente no concebidas, pero que permiten una monetización real de la infección en un muy corto período de tiempo. Muchas de estas acciones se ejecutan además en *background* —o segundo plano— como resultado de que el atacante ponga todos los medios a su alcance para evitar que el verdadero dueño de la máquina llegue a conocer la infección.

Esta situación puede tener consecuencias dramáticas a la hora de realizar un seguimiento de la autoría de los ataques ya que el rastreo podría llevar a un usuario inocente en donde la pista se confundiera entre otras muchas que lleven hasta el verdadero responsable del incidente.

2.2 Hipótesis de partida y objetivos

En este contexto, la definición de metodologías capaces de hacer frente de una forma efectiva a la amenaza de las *botnets* se torna fundamental. Así, el autor enuncia a continuación una metodología para el análisis de muestras de tráfico de *botnet* pertenecientes a canales de *Command & Control* de modo que puedan ser utilizadas para la identificación de amenazas relativas a este fenómeno empleando algoritmos de clasificación supervisada. La hipótesis en la que se basa esta tesis doctoral es la siguiente:

«*El uso de características atómicas para detectar aquellas comunicaciones que compongan el proceso de envío y recepción de comandos maliciosos hacia o desde las víctimas infectadas por* malware *concebido para el despliegue de una* botnet».

Objetivos generales

Con el objetivo último de poder corroborar o rechazar la hipótesis de partida anterior, se han definido los siguientes objetivos generales:

1. Desarrollar una metodología que permita la obtención de muestras de tráfico como objetos de estudio en investigaciones relacionadas con el análisis de tráfico.

2. Obtener un modelo de representación de las conexiones que permita la clasificación de paquetes en base a sus características.

3. Obtener un modelo de representación de las conexiones que permita la clasificación de secuencias de paquetes de diferentes longitudes procedentes de una misma conexión.

Para garantizar la consecución de los anteriores objetivos generales, se han definido los siguientes objetivos específicos:

1. Agrupar las conexiones y anonimizar las muestras para eliminar la posibilidad de que aparezcan sesgos —en la forma de direcciones IP o puertos— que trivialicen los resultados obtenidos para esas muestras.

2. Seleccionar aquellos atributos representativos de las conexiones maliciosas y filtrar del resto de características.

3. Establecer medidas que permitan la obtención de patrones relativos a la frecuencia con la que se producen las comunicaciones en base a diferentes valores de tendencia central.

4. Generar una metodología capaz de representar muestras de tráfico de secuencias de paquetes de diferentes longitudes.

5. Evaluar la capacidad de que este modelo de representación sea capaz de identificar tráfico perteneciente a canales de *Command & Control*.

6. Determinar cuáles son aquellos clasificadores supervisados que mejores resultados generales obtienen tras ser entrenados con muestras procedentes de *botnets*diferentes.

2.3 Metodología

Como resultado del proceso de revisión bibliográfica anterior y una vez enmarcado el contexto en el que se encuadra la tesis doctoral y definida la hipótesis de partida, el doctorando presenta en este apartado la parte del documento destinada a la explicación de la metodología que se ha empleado para la consecución de los objetivos ya definidos. En las páginas que siguen se procederá a detallar la aplicación del método científico que dará cobertura a esta tesis doctoral, en la que se incluyen las fases en las que se ha dividido el trabajo, las tareas realizadas en cada fase, una breve aproximación a la metodología a utilizar por cada tarea y el cronograma del desarrollo de la tesis en el tiempo.

2.3.1 El método científico

El método científico establece una serie de procedimientos a seguir para la validación de todo trabajo experimental. A lo largo del documento y dado su eminente carácter científico, no se podía dejar de lado la aplicación rigurosa de este de cara a garantizar una validación objetiva y estricta de la hipótesis de partida tal y como se recoge gráficamente en la figura 2.1. En base a ello, a continuación se definen las tareas que constituirán el trabajo doctoral aquí descrito.

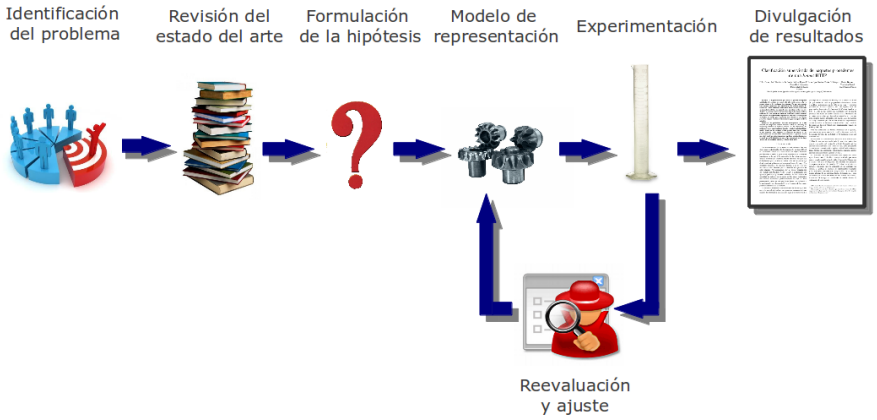


Figura 2.1: Aplicación del método científico.

1. **Identificación del problema**. El problema que las *botnets* trae consigo una gran cantidad de problemáticas. Su uso sencillo y la gran cantidad de vectores de infección existentes, así como un uso cada vez más masificado de los sistemas informáticos, han dado lugar a un cambio de paradigma

2. **Estudio del estado del arte**. La identificación del problema es solamente el primer paso del proceso. El investigador deberá ser capaz de conocer y manejar las técnicas más actuales aplicadas en dominios similares de cara a ser capaz de enfrentarse a la problemática de la forma más eficiente posible para tratar de anticipar cuáles van a ser los principales obstáculos a los que se va a tener que enfrentar a lo largo de la investigación.

3. **Formulación de hipótesis**. Una vez analizado el estado del arte, el investigador, como parte inicial de su trabajo doctoral formulará la hipótesis fundamental que se tratará de verificar o rechazar en esta disertación.

4. **Diseño de un modelo de representación**. En base a los estudios realizados y a la línea de trabajo marcada en la propia hipótesis, se dise-

ñará un modelo de representación de las conexiones que permita su utilización con la clasificación supervisada. Se propondrán varias soluciones posibles: desde la utilización de paquetes individuales hasta el uso de secuencias de estos.

5. **Experimentación**. La fase de experimentación conllevará la realización de los experimentos programados empleando los modelos de representación ya definidos. Será ejecutada en cuatro subfases:

   (a) Selección de candidatos.

   (b) Obtención de muestras de tráfico.

   (c) Aplicación de los modelos de representación para generar nuevos muestras o *samples*.

   (d) Planificación del proceso de ejecución.

   (e) Ejecución de los experimentos en sí mismos.

6. **Reevaluación y ajuste del problema**. La fase de reevaluación de los resultados es uno de los apartados más relevantes del método científico dado que la principal fortaleza del método científico es precisamente el componente de retroalimentación intrínseco a todo proceso de mejora continua. La capacidad del investigador para analizar los resultados y su habilidad para comprender los motivos de las desviaciones obtenidas así como sus conocimientos sobre técnicas complementarias que puedan proveer de una respuesta más satisfactoria a la problemática analizada es clave en este punto.

7. **Presentación de resultados ante la comunidad científica**. Una parte inherente del trabajo científico es la divulgación de los resultados de las investigaciones llevadas a cabo. En este punto, se hace referencia a la necesidad de hacer públicos estos trabajos y de dar a conocer a la comunidad científica los aportes realizados y, sobre todo, las líneas de trabajo futuras que emanan de este trabajo, así como facilitar la replicabilidad de los experimentos aquí propuestos de cara a futuras correcciones y esfuerzos que pudieran reducir la incertidumbre de las propuestas aquí realizadas.

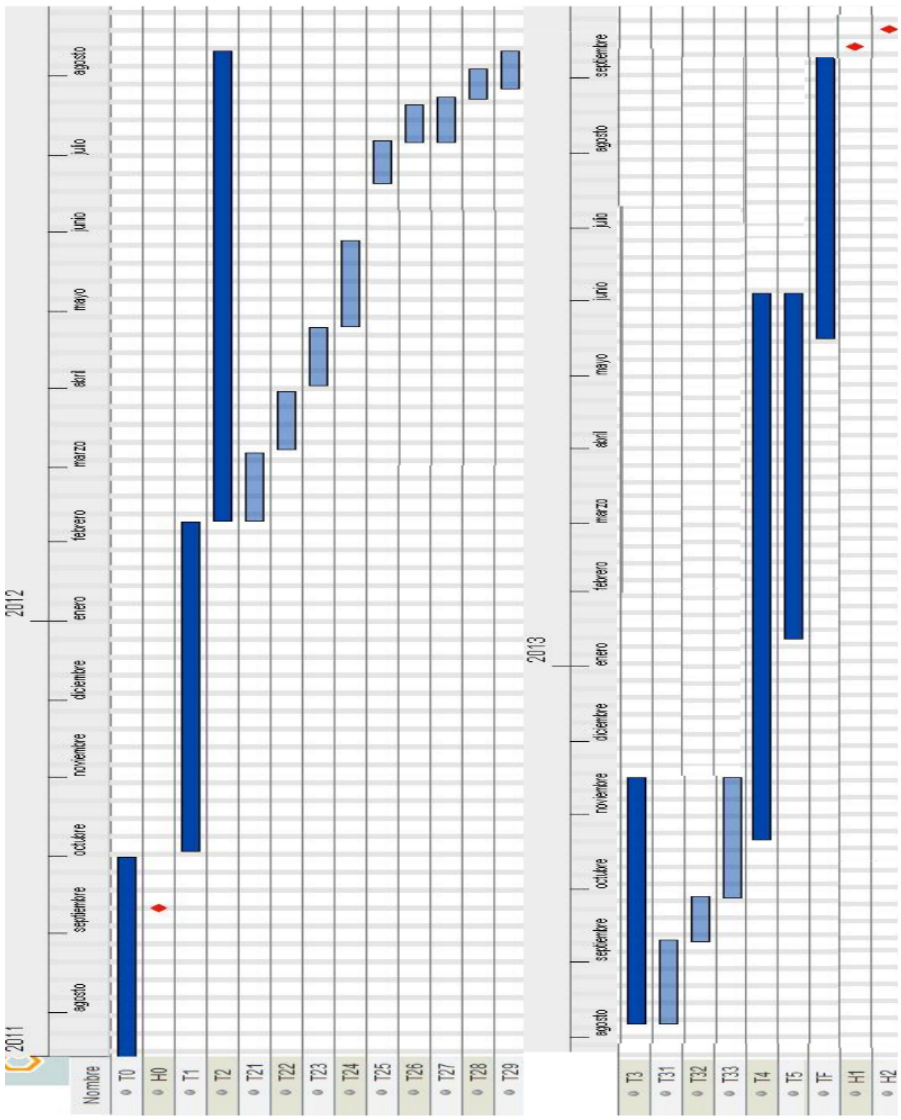


Figura 2.2: Cronograma de las tareas en que se ha dividido la tesis doctoral.

2.3.2 Fases en las que se divide el proyecto

Con tal fin, se ha propuesto la división del proceso de desarrollo de la tesis doctoral tal y como se puede ver en la figura 2.2. Las tareas recogidas en ella se definen en orden cronológico[1] a continuación.

- **Tarea 0: Formulación de la propuesta de proyecto de investigación** (T_0). La tarea T_0 constituye el procedimiento llevado a cabo para la formulación de la propuesta de proyecto que respaldaría a la postre el trabajo del doctorando a partir del 1 de octubre de 2011: la beca FPI (Formación de Personal de Investigación) ofrecida por Deiker, oficina dependiente del Vicerrectorado de Investigación, Innovación y Transferencia de la Universidad de Deusto que se ocupa de la Captación, Promoción, Gestión y Transferencia de la Investigación (esta tarea no se refleja en la figura por ser una tarea complementaria).

- **Tarea 1: revisión del estado del arte e identificación del problema** (T_1). La tarea T_1 consiste en la elaboración de una revisión exhaustiva del estado del arte actual en lo que a las técnicas de análisis de trafico se refiere para la detección de tráfico procedente de canales de *Command & Control*. El objetivo es conseguir que el investigador se forme una opinión lo más ajustada posible de cuáles pueden ser las líneas de trabajo susceptibles de ser explotadas para hacer frente al problema de partida.

- **Tarea 2: modelado de tráfico simple** (T_2). La tarea T_2 consiste en la proposición de un modelado de tráfico simple que emplea los atributos propios de los paquetes como características a utilizar para obtener las representaciones. Para validar la eficacia del modelo se ha considerado el problema como un asunto de clasificación supervisada. La tarea T_2 se puede dividir en las siguientes subtareas:

  - *Tarea T_{21}: diseño del modelo de representación simple.*
  - *Tarea T_{22}: obtención de muestras benignas.*
  - *Tarea T_{23}: obtención de muestras de tráfico de Flu.*
  - *Tarea T_{24}: ejecución de experimentos de clasificación supervisada con Flu.*

[1] Las fechas se corresponden con las fechas reales de la ejecución de la tesis doctoral, cuyo registro y depósito fueron fijados para el mes de septiembre de 2013.

  - *Tarea T_{25}: obtención de muestras de tráfico de Prablinha.*
  - *Tarea T_{26}: ejecución de experimentos de clasificación supervisada con Prablinha.*
  - *Tarea T_{27}: obtención de muestras de tráfico de Warbot.*
  - *Tarea T_{28}: ejecución de experimentos de clasificación supervisada con Warbot.*
  - *Tarea T_{29}: ejecución de experimentos de clasificación supervisada genéricos.*

- **Tarea 3: modelado de tráfico por secuencias de n-paquetes** (T_3). En la tarea T_3 se buscará ampliar el alcance del modelo de representación anterior ampliando las características a aquellas propias de secuencias de n paquetes mediante la combinación de todas aquellas pertenecientes a los paquetes que las componen. La tarea T_3 se puede dividir en tres subtareas:

  - Tarea T_{31}: diseño del modelo de representación de muestras empleando atributos pertenecientes a secuencias de n-paquetes.
  - Tarea T_{32}: construcción de representaciones de n-paquetes.
  - Tarea T_{33}: *ejecución de experimentos de clasificación supervisada con Flu, Prablinha, Warbot y tráfico genérico*. En este apartado se ejecutan las pruebas para las $n-1$ representaciones constituidas por los atributos de $n = k \rightarrow \forall k \in \{2,3,4,5,6,7,8,9,10,11,12\}$ paquetes consecutivos procedentes de una misma conexión.

- **Tarea 4: redacción de la tesis** (T_4). La tarea T_4 está constituida por el proceso de redacción de la tesis doctoral en sí misma. En esta tarea se incluye también la redacción de anexos, diseño de contenidos gráficos y maquetación, entre otros aspectos.

- **Tarea 5: revisión** (T_5). En la tarea T_5 se recoge el espacio temporal reservado para la revisión del manuscrito por parte del doctorando.

- **Tarea final: registro y presentación** (T_F). Por último, la tarea T_F hace referencia a la formulación del proceso de registro, depósito, constitución del tribunal y preparación de la defensa de la presente tesis doctoral.

Asimismo, en el gráfico anterior se han fijado también una serie de hitos que han constituido algunos de los fechas más relevantes para la consecución del proyecto.

- *Memoria del proyecto* (H_0): se trata del documento que se entregaba en verano de 2011 para presentar una primera aproximación de la investigación a realizar.
- *Registro de la tesis* (H_1): es la fecha en que se tiene estimado el registro del trabajo doctoral tras la obtención de las calificaciones de los cursos de doctorado.
- *Depósito de la tesis* (H_2): es la fecha en que se tiene estimada la realización del depósito de la tesis una vez haya sido obtenido el registro de la misma.

En las tareas T_{24}, T_{26}, T_{28}, T_{29} y T_{33} ha procedido a emplear 23 algoritmos de clasificación supervisada diferentes que implementan diferentes técnicas de aprendizaje: desde árboles de decisión hasta redes bayesianas con diferentes configuraciones y máquinas de soporte vectorial, pasando por perceptrones simples y complejos y por el análisis de los k vecinos más próximos.

2.3.3 Estructura de la tesis doctoral

La estructura de los contenidos del documento final de la tesis doctoral presenta un índice similar al siguiente.

Capítulo 2

En el capítulo 2 se identificará de forma general la problemática que trae consigo el fenómeno de las *botnets* a la hora de comprometer el nivel de seguridad del ciberespacio actual, resaltando aquellos aspectos que hacen especialmente interesante para el investigador el atajo de dichos problemas. Asimismo, se definirá la hipótesis fundamental del trabajo y la metodología a desarrollar para verificarla de acuerdo a las características propias del método científico.

Capítulo 3

En el capítulo 3 se presenta el estado del arte relativo al análisis y detección del fenómeno *botnet*, situando al lector en el estadío actual de la investiga-

ción del fenómeno, empezando por una breve historia genérica del *malware* y terminando con la evolución hacia el modelo de amenazas que son analizadas en este documento.

Capítulo 4

En el capítulo 4 se detalla el proceso de modelado de las muestras de cara a la generación de las diferentes representaciones experimentales que serán empleadas en procesos posteriores.

Capítulo 5

En el capítulo 5 se define la metodología a seguir para la obtención de muestras de tráfico experimentales, detallando tanto las características de las muestras de *malware* malicioso analizado como el proceso de obtención de tráfico legítimo para su posterior análisis. Se subraya en este capítulo la importancia de los esfuerzos por obtener muestras de tráfico benignas lo más asépticas posibles y lo más ajustadas al tráfico que se puede dar en un entorno real.

Capítulo 6

En el capítulo 6 se expone el proceso de validación experimental al que se ha sometido la hipótesis de partida presentada anteriormente, presentando los resultados obtenidos tras plantear el problema como un problema de clasificación supervisada. Los experimentos propuestos en este apartado se pueden clasificar en tres tipos atendiendo a la forma de etiquetado de las muestras: detección de tráfico procedente de una *botnet* determinada en tráfico benigno, detección de una *botnet* en un entorno donde se incluye tráfico benigno y malicioso procedente de otras *botnets* y detección genérica de tráfico de *botnet*.

Capítulo 7

En el capítulo 7 se recogen las principales conclusiones extraídas de esta investigación doctoral y se realiza un ejercicio de prospectiva a la hora de definir cuáles serán las principales líneas de trabajo futuro que guiarán los pasos de escritores de *malware* y analistas de seguridad en adelante, haciendo un especial hincapié en aquellos aspectos que puedan suponer un ataque al sistema de detección propuesto.

Anexos

Adicionalmente, se incluyen en la parte final una serie de anexos que complementan la información presentada en este documento. Principalmente, se constituye como información complementaria a la ya expuesta en el documento principal y que desarrolla aspectos cuya inclusión detallada en dicho manuscrito podría desviar la atención de las verdaderas aportaciones del trabajo principal. De esta manera, se incluyen como anexos las fichas técnicas de las muestras de *malware* empleadas así como la definición detallada de algunas características propias de las muestras de tráfico analizadas, además de un índice alfabético de los términos más relevantes utilizados a lo largo del trabajo y de las publicaciones que han dado soporte previo a este trabajo.

«Nunc minerva, postea palas» («Primero la sabiduría, después la guerra»).

Lema de la Academia de Ingenieros Militares del Ejército de España

Capítulo

3

Revisión bibliográfica

El mundo actual y su economía y relaciones sociales, regidos todos por un uso generalizado de los equipos informáticos y de Internet, ha contribuido al surgimiento de modelos de negocio delictivos que explotan las vulnerabilidades de los sistemas informáticos con fines eminentemente económicos. Las aplicaciones maliciosas actuales que permiten a los ciberdelincuentes hacer esto posible son la evolución de aquellas que en el pasado dañaban equipos y ordenadores indistintamente, pero más orientado desde la perspectiva de la obtención de un beneficio económico. Así, todas ellas en su conjunto constituyen el mundo del *malware*. Precisamente, dentro de las diferentes variantes que se pueden encontrar, las *botnets* son solamente un tipo de herramienta mucho más específica que combina las capacidades de otras tantas, pero para ello necesitamos antes hacer un pequeño repaso de la situación actual de estas herramientas y de cuáles son las posibilidades.

En este capítulo se va a recorrer, desde sus orígenes, la historia de un fenómeno que ha sentado las bases al panorama actual en donde las *botnets* son uno de los máximos exponentes de las amenazas informáticas. Se va a desarrollar en las páginas que siguen el fenómeno desde su surgimiento, hasta la clasificación e identificación de las muestras actuales pasando por una revisión exhaustiva del estado del arte para su detección y desactivación.

3.1 Surgimiento de la industria del *malware*

Los tiempos y los avances tecnológicos han cambiado la forma en que los escritores de *malware* han entendido su negocio. La definición más usual del término tiene su origen en las palabras inglesas *malicious* y *software* y hace referencia a «cualquier programa informático diseñado con la intención de dañar ordenadores, redes o información» [SPDB09]. De la fama y la gloria anhelada por algunos hackers de finales del siglo XX, se ha dado paso a un mundo mucho más profesionalizado en el que los conceptos de *crimen como servicio* —*Crime as a Service* o CaaS— y de cibercrimen campan a sus anchas. En las páginas que siguen, y como introducción contextual a la situación actual del *malware*, procedemos a enumerar cuál es la realidad de las aplicaciones maliciosas hoy en día, sentando las bases para lo que son ahora las nuevas amenazas del ciberespacio del Siglo XXI pese a sus raíces mucho más primitivas.

3.1.1 Una breve historia del *malware*

Es en 1949 cuando Von Neumann estableció la idea de los programas almacenados y cuando expuso la *Teoría y organización de autómatas complejos* [VN49], donde presentaba por primera vez la posibilidad de desarrollar pequeños programas auto-replicantes y capaces de tomar el control de otros de similar estructura mediante la gestión del acceso a la memoria que estos utilizan. Si bien es cierto que el concepto tiene miles de aplicaciones en la ciencia —por ejemplo, en campos ligados a la inteligencia artificial [Neg05]— no resultaba enrevesado pensar en aplicaciones maliciosas de la teoría expuesta por Von Neumann que dieran lugar al nacimiento de los virus informáticos, programas que se reproducirían a sí mismos el mayor número de veces posible en memoria para llegar a ocupar, si tenían posibilidad, la totalidad del espacio lógico restante.

De hecho, ya en 1959, en los laboratorios de Bell Computer, tres jóvenes programadores, Robert Thomas Morris, Douglas Mcllroy y Victor Vysottsky crearon un juego denominado Core War [Dew84], basado en la teoría de Von Neumann con el objetivo de que diferentes programas combatieran entre sí tratando de ocupar toda la memoria de la máquina para eliminar así a los oponentes. Este juego es considerado el precursor de los virus informáticos.

Fue en 1972 cuando Robert Thomas Morris creó el que es considerado como un programa autoreplicamente precursor directo de los virus infor-

máticos: el Creeper —o enredadera— que era capaz de infectar máquinas IBM 360 de la red ARPANET[1] y que emitía un mensaje en pantalla que decía «Soy una enredadera, atrápame si puedes» [Par05]. Para eliminarlo, se creó, precisamente, otro *malware*: el llamado Reaper —en inglés, *segadora*— que estaba programado para buscarlo y eliminarlo [Bra90] convirtiéndose en lo que algunos consideran un primer embrión de los antivirus modernos.

En la década de los ochenta y, a medida que los PC ganaban popularidad, cada vez más gente se acercaba a la informática y experimentaba con sus propios programas. Esto dio lugar a los primeros desarrolladores de programas dañinos y en 1981, un estudiante de instituto, Richard Skrenta escribe el primer virus de amplia reproducción: Elk Cloner [Paq00, BDM10], que infectaba ficheros del sector de arranque en sistemas Apple DOS, contando el número de veces que arrancaba el equipo y al llegar a 50 mostraba un poema.

En 1984, Frederick B. Cohen [Coh87] acuña por primera vez el término virus informático en uno de sus estudios definiéndolo como un «programa que puede infectar a otros programas incluyendo una copia posiblemente evolucionada de sí mismo» convirtiéndose en el primer investigador que hacía frente a los aspectos prácticos y teóricos del *malware* [Adl]. De hecho, el propio Cohen describiría en trabajos posteriores algunas de las primeras contramedidas para luchar contra los virus informáticos en numerosos artículos [Coh89, Coh91]. Mientras tanto, mediada la década de los ochenta, hacía su aparición el virus Jerusalem o Viernes 13, que era capaz de infectar archivos *.exe* y *.com* [Coh94]. Fue detectado por primera vez por la Universidad Hebrea de Jerusalén llegando a ser uno de los virus más famosos y reproducidos de la historia [Kee02].

En los noventa, fueron numerosas las investigaciones que analizarían el fenómeno desde diferentes puntos de vista. Eugene Spafford proponía en su «Computer Viruses-A Form of Artificial Life?» [Spa90] la aplicación de una *checklist* para tratar de determinar si los virus informáticos encajaban con la definición que se daba a la *vida artificial*, vigente desde un estudio de 1986 por Christopher G. Langton [Lan86]. Dicho listado de características incluían, entre otras, aspectos como su crecimiento, reproducción, metabolismo o relación con el entorno. Spafford llegaba a la conclusión de que estos programas estaban muy cerca de lo que se podría considerar vida artificial y sugería la ampliación del término *vida* poniendo en entredicho las

[1] Antecesora del Internet moderno, la *Advanced Research Projects Agency Network* fue creada por el *Department of Defense* (DOD) de los EEUU como via de comunicación y sentando las bases para la implantación posterior del protocolo TCP-IP.

implicaciones éticas de la creación descontrolada de virus.

Por su parte, Adleman analizaba en 1990 cómo combatir y aplicar técnicas de desinfección de programas de baja o moderada capacidad destructiva [Adl], mientras Kephart proponía estructuras de control de los equipos de una compañía lo más centralizadas posible para facilitar la detección y desinfección [KWC93] llegándose a afirmar que la problemática del *malware* y de los virus informáticos era un tema cuyas repercusiones se habían sobredimensionado [KSCW97]. Como se irá viendo, estos planteamientos, por exceso, defecto o atemporalidad, se han quedado completamente obsoletos.

Así, ya en 1999, surgía el gusano Happy, desarrollado por el francés Spanska, creando una nueva corriente en cuanto a la propagación del *malware* que persiste hasta el día de hoy: el envío de gusanos por correo electrónico [Szo05]. Un año más tarde y tras volver a pasar el mundo de la informática al primer plano con la psicosis del conocido como *Efecto 2000* [Qui98] tuvo lugar un hecho que acapararía una repercusión mediática aún mayor debido a los daños reales ocasionados por la infección tan masiva que produjo: se trataba del gusano ILoveYou o VBS/LoveLetter [Che03]. En mayo de 2000, este *malware*, haciendo uso de técnicas de ingeniería social, infectaba a los usuarios a través del correo electrónico consiguiendo afectar a millones de equipos a nivel mundial y poniendo de manifiesto la necesidad de crear organismos que dieran respuesta a las amenazas que acontecían en la red [Rho00].

Comenzaba así la época de grandes epidemias masivas que tuvieron su punto álgido en el año 2004 [GPJ04]. Fue en ese año cuando aparecieron gusanos como el Mydoom, el Netsky, el Sasser, Blaster o el Bagle, que buscaban alarmar lo más posible a la comunidad informática tratando de tener la mayor repercusión y reconocimiento posible.

Sin embargo, estas tendencias cambian de rumbo a partir del año 2005 [BSBdV]. Es este el más duro y el último de las grandes epidemias masivas. Los creadores de *malware* se dieron cuenta de que sus conocimientos servirían para mucho más que para tener repercusión mediática. Era tiempo de convertir sus capacidades y conocimientos en una forma de vida. Tras cinco años de tendencia sostenida, los virus tal y como los conocíamos fueron dando paso a gusanos y troyanos encargados de formar redes de *bots* con fines puramente económicos. Los programadores comprendieron que la escritura de *malware* era un entretenimiento que podía convertirse en un negocio muy rentable.

La mejor prueba de ello fue la aparición de nuevas formas de monetizar

estas infecciones: desde la violación de los sistemas *pay-per-click* de las campañas de publicidad en la red hasta los denominados troyanos bancarios. Con el objetivo de favorecer su propagación, es una realidad que su expansión viene cada vez más motivada por la existencia de miles de variantes de cada uno de ellos. Por ello, los creadores del *malware*, a sabiendas de que su detección por los sistemas de detección de firmas tradicionales se estaba tornando cada vez más efectiva, fueron desarrollando herramientas que modificaban el código de los mismos y, a veces, las propias características de la infección [YZA08, KSS, BHB+] para garantizar su pervivencia en un ecosistema cada vez más *amenazado* por las soluciones antivirales comerciales.

3.1.2 Situación actual del *malware*

Desde el comienzo de la aparición de *malware* se han llevado diferentes estadísticas del daño causado por dicho *software*. El crecimiento exponencial que han experimentado las bases de datos de algunas firmas de antivirus, como AV-Test o Kaspersky [Gos11], disparaba exponencialmente la cantidad de *malware* encontrada a finales de la primera década del siglo XXI 3.1. Podemos ver que desde el año 2006 hay un aumento significativo en la cantidad de *malware* encontrado. Este hecho sólo se podría explicar por el incremento de la capacidad de cómputo de los centros de procesamiento de los antivirus (incluyendo en ellos las tecnologías en la nube o *in-the-cloud*), el desarrollo de nuevas técnicas de detección automática, implementación de nuevas heurísticas, virtualización y nuevos métodos de análisis del comportamiento como ejemplo de los nuevos métodos que la industria de *anti-malware* ha desarrollado para incrementar la calidad de la protección ofrecida.

Así, la mayoría de soluciones que ofrecían las casas antivirus durante la década de los 90, como la monitorización del trafico o el escaneo periódico del equipo, eran soluciones indispensables mínimas ya en el año 2006. Para 2009, debido a la creciente diferenciación de los ataques y, principalmente, a la aparición de *malware* polimórfico con especiales capacidades para mutar su código [BHB+], quedaron obsoletas y tuvieron que implementar nuevas técnicas de detección más fiables y robustas. La revolución de la industria en cuanto a la implantación de novedosas técnicas de detección viene motivada por la necesidad de poder hacer frente a una problemática, la del *malware*, cada vez más compleja.

Paralelamente, durante los años 2009 y 2010, ocurren dos factores muy

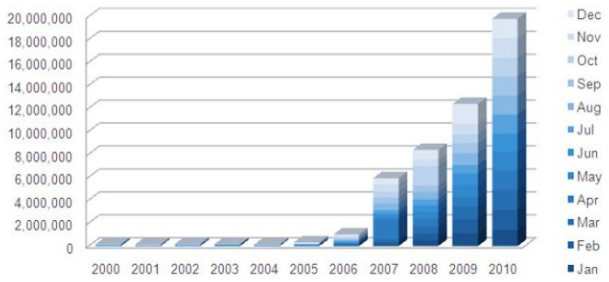


Figura 3.1: Evolución del número de muestras maliciosas almacenadas en la base de datos de AV-Test. *Fuente: AV-Test.*

relevantes para el estudio de esta gráfica. En primer lugar, durante el año 2009 tuvo lugar un caso excepcional de epidemia cibernética con el gusano Conficker, el cual infectó millones de máquinas alrededor del mundo. Las formas de propagación de dicho gusano iban desde la fuerza bruta en ataques de red, hasta la propagación por medio de USB infectados. Dicho funcionamiento se explicará en el siguiente apartado debido a la importancia de dicho gusano dentro de este ámbito.

Por su parte, el año 2010 tampoco pasó desapercibido puesto que fue el año del troyano Zeus —también conocido como ZeuS— que, además de infectar miles de equipos, tenía la característica de ser uno de los *malware* más avanzados y complejos nunca creados por la complejidad de los ataques de robo de credenciales que se implementaban en él. Zeus, será también estudiado más a fondo en el siguiente apartado.

Como dato también muy importante durante el segundo semestre de 2010 hay que tener en cuenta el gusano Stuxnet, el cual para ser analizado completamente y entendido por la comunidad de expertos se necesitaron de tres meses completos para la comprensión completa de sus capacidades.

Por otro lado, empiezan a aparecer un nuevo tipo de troyanos: los conocidos como *Trojan-Ransom* o también *ransomware*. Se trata de una especie de troyanos maliciosos, que mantienen una parte —a veces incluso la totalidad— del ordenador de la víctima secuestrada a la espera del pago de lo que se podría considerar como un rescate. Para ello, este tipo de *malware* cifra los datos, a veces incluso empleando algoritmos triviales como XOR[1] [SH12], para luego exigir el pago de una cantidad utilizando métodos anónimos de pago. El éxito de este tipo de ataques radica en dos características

[1] Si ejecutamos una operación XOR sobre un fichero el sistema de descifrado sería tan trivial como volver a ejecutar exactamente la misma operación, obteniéndose como resultado el fichero original.

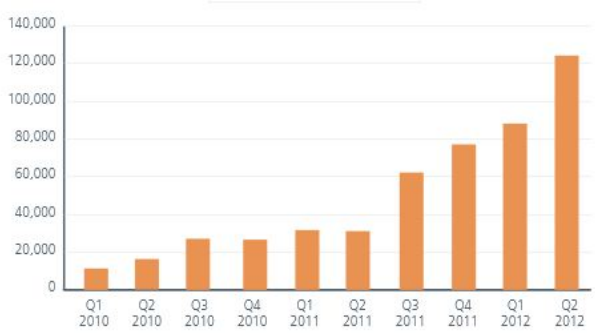


Figura 3.2: Evolución del número de ataques provocados por *ransomware*. *Fuente: McAfee.*

que los hacen posibles. Por un lado, que las cantidades solicitadas no suelen ser demasiado elevadas y, por otro, que el desbloqueo se hace efectivo tras el pago al menos en apariencia. Es decir, los ciberdelincuentes prefieren no plantearlo como una estafa que ofrezca un servicio que luego no se provea, sino que prefieren hacerlo efectivo para que, en caso de que se corra el rumor, poder explotar esta infección en más ocasiones. Sin embargo, este tipo de ataques, no son recientes. De hecho, uno de los primeros troyanos vistos en PC, el AIDS-Trojan, documentado en 1989, actuaba exactamente de esta manera. Aunque por muchos años estos ataques han sido muy esporádicos, últimamente se están dando con mayor frecuencia.

Otro ejemplo, puede ser el conocido en España como *virus de la Policía* que actuaba de una forma similar. Infectaba equipos y mostraba una advertencia a los usuarios haciéndose pasar por el Cuerpo Nacional de Policía y acusaba a la víctima de poseer pornografía infantil, exigiendo a la víctima el pago de antemano de una cantidad para evitar una investigación más exhaustiva, tal y como define Francois Paget en un informe desarrollado para McAfee en 2012 [Pag12] y en el que se puede apreciar el incremento de los ataques de este tipo que se recoge en la figura 3.2.

En este contexto, la evolución de la amenaza del *malware* ha atraído también a nuevos actores que se interesan por él como herramienta de ciberespionaje. De esta manera, más allá de los ataques contra la reputación que tanto preocupan por la viralidad con la que se pueden difundir falsos testimonios, los ataques dirigidos que hacen uso de *exploits*[1] para explo-

[1] Un *exploit* es una pieza de *software*, fragmento de datos o secuencia de comandos y/o acciones, utilizada con el fin de aprovechar una vulnerabilidad de seguridad de un sistema de información para conseguir un comportamiento no deseado del mismo como accesos

tar vulnerabilidades identificadas en los sistemas de una empresa, se están constituyendo como algunas de las nuevas amenazas contra los intereses corporativos.

3.2 Tipos de *malware*

Es habitual encontrar referencias al *malware* empleando la terminología de *virus* o *virus informático*. Como se verá más adelante, es esta una apreciación manifiestamente inexacta dado que los virus son solamente una de las formas en las que se pueden presentar las aplicaciones maliciosas. Por este motivo, se va a proceder a clasificar los tipos de *software* malicioso que se pueden encontrar en la actualidad. En una clasificación básica que atiende a las funcionalidades propiamente dichas de las aplicaciones maliciosas, estas pueden agruparse en: *virus*, *hoax*, troyanos, *spyware*, bombas lógicas, gusanos o *bots*. Existen, a su vez, varias subdivisiones internas en función de las características y funcionalidades específicas de cada programa, así como algunas muestras que incorporan aspectos que se pueden encontrar también en el resto de los grupos.

- **Virus**. Un virus informático es un tipo de *malware* que tiene por objeto alterar el normal funcionamiento de la computadora sin el permiso o el conocimiento del usuario. Los virus habitualmente reemplazan archivos ejecutables por otros ficheros *infectados* con el código de éste que ha inyectado su carga maliciosa —o *payload* en inglés— en el fichero legítimo original. La propagación de un virus informático es conceptualmente simple ya que tiene lugar a través de la ejecución de un programa que ha sido infectado previamente. Aunque la principal característica de los virus informáticos es precisamente la de propagarse a través de un *software* ya *infectado* y, aunque no se replican por sí mismos porque no tienen esa facultad que sí tienen los gusanos informáticos, pueden estar codificados con muy distintos objetivos: desde las simples bromas de sus orígenes hasta la provocación de daños importantes en los sistemas, el bloqueo de las redes informáticas generando tráfico inútil o la extracción de datos almacenados en el equipo.

  Es entonces cuando el código del virus queda residente en la memo-

no autorizados, la toma de control del sistema o ataques de denegación de servicio.

ria RAM[1] de la computadora, aun cuando el programa que lo contenía haya terminado de ejecutarse. De esta manera, el virus podría tomar el control de los servicios básicos del sistema operativo, infectando con posterioridad, archivos ejecutables que puedan ser llamados a ejecución en el futuro. Finalmente se añade el código del virus al programa infectado y se graba en el disco, con lo cual el proceso de replicado se completa. Históricamente, algunas de las acciones más habitualmente achacadas a los virus, pero no por ello las únicas, han sido su capacidad de:

  - Unirse a un programa instalado en el ordenador permitiendo su propagación.
  - Mostrar en la pantalla mensajes o imágenes humorísticas, generalmente molestas.
  - Ralentizar o bloquear el ordenador implementando diversas modalidades de ataques de denegación de servicio local.
  - Destruir la información almacenada en el disco, en algunos casos vital para el sistema, con el fin último de impedir el funcionamiento del equipo.
  - Reducir el espacio en el disco.
  - Molestar al usuario cerrando ventanas, moviendo el ratón o cifrando y descifrando contenidos arbitrariamente y pidiendo un *rescate* por ellos.

- **Caballos de Troya o troyanos**. El término troyano o caballo de Troya proviene de su homólogo anglosajón *Trojan Horse*. En general, podemos hablar de la existencia de un troyano cuando nos enfrentamos a un programa ejecutable con fines maliciosos que, aunque no se propaga por sí solo, tiene como objetivo alojarse en las computadoras para permitir el acceso a usuarios externos a través de una red local o de Internet. Un troyano no es en sí un virus ya que posee una finalidad diferente. El virus es un huésped destructivo mientras que el troyano no necesariamente provoca daños. Para que un programa sea catalogado como troyano tiene que acceder y controlar la máquina anfitriona sin ser advertido, normalmente tras instalarse bajo una

[1] La RAM o memoria de acceso aleatorio —en inglés *Random Access Memory*— es la memoria desde donde el procesador recibe las instrucciones y en donde almacena los resultados, utilizándose como memoria de trabajo para el sistema operativo.

apariencia inocua como puede ser una aplicación conocida, una imagen o un archivo de música entre otros, siendo esta la característica que le da nombre.

Una vez instalado podría incluso tratar de aparentar la realización de una función útil —aunque cierto tipo de troyanos optan únicamente por permanecer ocultos— mientras internamente lleva a cabo otras tareas de las que el usuario no es consciente. Una de sus funciones principales podría ser la de espiar el equipo infectado instalando un *software* de acceso remoto que permitiera monitorizar al usuario legítimo. Existen distintos tipos de troyanos clasificados en función de la forma de penetración en los sistemas y el daño que pueden causar:

- Troyanos de acceso remoto. Permiten llevar a cabo el control remoto de la máquina infectada ejecutando desde el exterior todo tipo de acciones.
- Troyanos de *spam*. Son herramientas utilizadas para el envío automático de correo basura o *spam*, como forma de monetizar la infección
- Troyanos *stealer*. Se trata de utilidades de destrucción o robo de datos que puende haber sido creados *ad hoc* como parte de una práctica de espionaje industrial. A estos efectos, se entiende como espionaje industrial toda aquella «actividad sistemática de obtención clandestina de información protegida por su propietario en el ámbito de la inteligencia económica» [Nav07].
- *Trojan-Clicker*. Por su parte, este tipo de *malware* emula la acción de hacer clic llevando la navegación del usuario a URL fraudulentas o servicios de publicidad *online* legítimos en una practica conocida como *click-frauding*. Estuvo presente únicamente a comienzos del año 2010, con un descenso en picado paralelo a los *Trojan-Downloader*.
- Troyanos *proxy*. Un *proxy* es un programa o dispositivo que realiza una acción en representación de otro. En otras palabras si una máquina A solicita un recurso a una tercera C, lo hará mediante una petición a B de modo que C no sepa que la petición procedió originalmente de A. Su finalidad más habitual es la de convertirse en un *servidor proxy*, utilizado para interceptar las conexiones de red que un cliente hace a un servidor de destino, bien sea por motivos de seguridad, rendimiento o anonimato. Por tanto, los

troyanos *proxy* asumen la identidad de la máquina infectada ante otras computadoras, pudiendo utilizar estas como plataforma para la ejecución de nuevos ataques.

- Troyanos FTP. El protocolo FTP —siglas en inglés de *File Transfer Protocol* o protocolo de transferencia de archivos— es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (*Transmission Control Protocol*), basado en la arquitectura cliente-servidor que permite que, desde un cliente, se pueda conectar a un servidor para descargar o enviar archivos y acceder a sus contenidos. Este tipo de aplicaciones son utilizadas para añadir o copiar datos hacia o desde la computadora infectada, que pueden incluir desde actualizaciones de las versiones del *malware* hasta la extracción de información sensible.
- Deshabilitadores de programas de seguridad como antivirus, cortafuegos o *firewalls*.
- Troyanos de ataque de denegación de servicio. Se contempla en este punto la posibilidad de ejecutar ataques contra los servidores hasta su bloqueo tanto locales como distribuidos.
- Troyanos URL. Estos troyanos cambiaban la forma en que la máquina infectada se conectaba a Internet para dirigir el usuario a conexiones de alto coste y poder monetizar así la infección. Tuvieron su apogeo cuando las líneas de acceso a Internet más utilizadas hacían uso de conexiones de módem permitiendo que el atacante monetizase la infección mediante llamadas de tarificación adicional [GC].

- ***Spyware* o programas espía**. Por *spyware* se entiende un programa que se instala en un ordenador furtivamente para recopilar información acerca de los hábitos de uso del equipo en el que ha sido instalado y enviar dichos contenidos a empresas publicitarias y otras organizaciones interesadas, incluyendo organismos oficiales para la investigación de delitos relacionados con la propiedad intelectual con toda la problemática legal que ello conlleva. Sus teóricos sitúan sus orígenes a gran escala en el proyecto Magic Lantern [Bri01]. Magic Lantern es identificado como un supuesto programa espía concebido por el FBI a comienzos de siglo para recabar información sobre los usuarios.

  Los programas espía pueden ser instalados en un ordenador a tra-

vés de un troyano o de un virus distribuido por correo electrónico, aunque también podrían estar ocultos en programas aparentemente inocuos. La información que habitualmente recogen va desde mensajes y contactos del correo electrónico, hasta datos sobre la conexión a Internet, historiales y costumbres de navegación o información insertada en formularios —como claves de correo, números de tarjeta de crédito, contraseñas, etc.—. Gran parte de esta información, almacenada en las *cookies*[1], puede ser bloqueada por el usuario dado que su uso no está normalmente vetado a este.

Conviene no confundir el *spyware* con el conocido como *adware*, que es aquel *software* que durante su funcionamiento despliega publicidad en forma de ventanas, *pop-ups* —ventanas emergentes— o barras de herramientas que se instalan como complemento de algunas aplicaciones para subvencionar económica y legítimamente otro tipo de proyectos. No hay que perder de vista que los usuarios terminan aceptando las condiciones de estos programas y no son forzados a instalarlos. En este sentido, no son considerados, al menos por definición, como programas *spyware* porque no están destinados a la obtención fraudulenta de información sobre los hábitos de navegación.

- ***Hoax***. Aunque los *hoax* no son *malware* en sí mismos y ni tan siquiera tienen las características que definen a un *software*, por lo que tampoco tienen la capacidad de reproducirse por sí mismos, sí que se suelen situar como una de las vías de infección más habituales dentro del mundo del *malware*. En realidad, se trata de mensajes de contenido falso que incitan al usuario a hacer copias y enviarlas a sus contactos. Suelen apelar a sentimientos morales —«Ayuda a un niño enfermo de cáncer»—, al espíritu solidario –especialmente, después de tener lugar catástrofes humanitarias de cualquier tipo como ya ocurriera con el terremoto de enero de 2010 en Haití [Roj10]— o a fenómenos relacionados con la histeria colectiva que puede generar la propagación de determinadas noticias —como el «aviso de un nuevo virus peligrosísimo»— y, en cualquier caso, tratan de aprovecharse de la falta de experiencia de los internautas novatos. Más recientemente, la muerte del Presidente venezolano Hugo Chávez ha causado un efecto

[1] Una *cookie* es un fragmento de información que se almacena en el disco duro del visitante de una página web a través de su navegador, a petición del servidor de la página. Esta información puede ser luego recuperada por el servidor en posteriores visitas. En ocasiones también se le llama *huella*.

similar ya que fue aprovechado por los cibercriminales para enviar enlaces maliciosos mediante correos electrónico que comentaban aspectos relacionados con su fallecimiento [Mej13].

- **Bombas lógicas**. Una bomba lógica es una parte de código insertada intencionadamente en un programa informático que permanece oculta hasta que se cumplen una serie de condiciones preprogramadas, momento en que se ejecuta una acción maliciosa concreta. Por poner un ejemplo, un programador puede ocultar una pieza de código que comience a borrar archivos en el momento en que sea despedido de la compañía a través de un *trigger* o disparador de base de datos que se active al cambiar la condición de trabajador del programador. Este fue el caso de Yung-Hsun Lin, un administrador de sistemas de Medco Health que fue condenado a dos años y medio de prisión y a pagar una multa de 81 200 dólares como compensación por colocar una bomba lógica en los sistemas de la compañía en octubre de 2003 [Vij07]. El *software* entonces estaba preparado para eliminar información almacenada en hasta setenta servidores de la compañía.

  Estas técnicas también son utilizadas por otras variantes de *malware* para esparcirse antes de ser detectadas. Muchos virus atacan sus sistemas huéspedes en fechas específicas, tales como el viernes 13, el anglosajón *April fools'day* o el día de los inocentes. Así, los troyanos que se activan en ciertas fechas son también llamados frecuentemente *bombas de tiempo*.

  Nótese que para que un *software* sea considerado una bomba lógica, la acción ejecutada debe ser indeseada y desconocida para el usuario legítimo del mismo. Por ejemplo, los programas *demos* o de prueba, que desactivan una cierta funcionalidad después de un tiempo prefijado, no son identificados como bombas lógicas.

- **Gusanos**. Un gusano —también llamado *IWorm* del inglés, *I* de internet y *worm* de gusano— es un *malware* que tiene la propiedad de duplicarse a sí mismo y que se sirve de las funcionalidades automáticas de un sistema operativo que generalmente permanecen invisibles al usuario.

  A diferencia de un virus, un gusano no precisa alterar los archivos de los programas para propagarse, sino que reside en memoria y se puede duplicar a sí mismo. Los gusanos casi siempre causan problemas en la red —aunque sea simplemente consumiendo ancho de banda—,

mientras que los virus tienen como principal objetivo infectar o corromper los archivos de la computadora que atacan.

Es habitual detectar la presencia de gusanos en un sistema cuando, debido a su incontrolada replicación, sus recursos se consumen hasta el punto de que las tareas ordinarias del mismo son excesivamente lentas o, simplemente, no pueden ejecutarse. Se suelen servir de las redes de computadoras para enviar copias de sí mismos a otros nodos —es decir, a otros terminales de la red— y son capaces de llevar esto a cabo sin la intervención del usuario, propagándose a través de Internet y empleando diversos vehículos para la infección, como los protocolos SMTP, IRC[1] o P2P entre otros.

- ***Bots***. Por *bot* se entiende todo ordenador secuestrado que forma parte de una red clandestina de equipos controlados por un *botmaster* [2]. El fenómeno *botnet* (fruto de la combinación de las voces inglesas *software robot networks*) no es casual ni tampoco especialmente reciente. El desarrollo de complejas redes de equipos conectados y las posibilidades computacionales y funcionales que su coordinación puede proveer, no ha pasado por alto para el conjunto de profesionales de la informática ni tampoco para los escritores de *malware*, *spammers* u otros delincuentes cibernéticos, que pueden utilizar como canales de control el protocolo IRC, las redes P2P o conexiones HTTP y HTTPS.

  Es por tanto un tema recurrente que viene preocupando a las agencias de seguridad más importantes del mundo desde hace tiempo, hasta el punto de, ya en 2008, haber obligado a algunas organizaciones como la Europol a preparar a sus profesionales para poder hacer frente a las amenazas que este tipo de redes traen consigo en materia de privacidad, anonimato y seguridad corporativa [Eur08]). En el caso de España, no es hasta principios de 2013 cuando se crea por fin el primer mando militar conjunto en materia de ciberdefensa [Cal13]. Paralelamente, se han llevado a cabo diversas operaciones policiales como la *Operación Bot Roast* liderada por el Federal Bureau of Investigation en 2007 [Off07], por la que el propio FBI detectó una *botnet* con un millón de ordenadores comprometidos [New07]. Sin embargo, es precisamente por su relación con el problema del *spam*[3] —o envío

[1] IRC —siglas en inglés de Internet Relay Chat— es un protocolo de comunicación en tiempo real basado en texto que permite conversaciones entre dos o más personas.

[2] El *botmaster* es la persona u organización que controla una *botnet*.

[3] El *spam* —del inglés, *spiced ham*— es una carne enlatada distribuida por la Hormel

masivo de correo basura—, que la lucha contra este tipo de redes contraía más importancia originalmente. Aún hoy en día, la práctica totalidad del correo basura enviado a nivel mundial se realiza a través de máquinas bajo control directo de los operadores de *spam* que se sirven de las *botnets* para llevar a cabo dichos envíos en su papel de *botmasters* [for11]. De acuerdo con un informe de SpamHaus[1], aproximadamente 100 *spammers* son responsables por sí solos del 80 % del tráfico de dichos contenidos maliciosos en Estados Unidos y Europa. Ya en 2004 se estimó que más de dos terceras partes del *spam* enviado en esas fechas procedía de las *botnets*. Symantec, en pleno apogeo del fenómeno a mediados de 2009, llegó a tasar en el 85 % anual alcanzando un 90,4 % del tráfico mundial de correos electrónicos en el mes de mayo del mismo año [sym09].

Hoy día sus aplicaciones son mucho más variadas. Desde ataques de denegación de servicio hasta robo masivo de credenciales o creación de pasarelas para camuflar otro tipo de actuaciones o transacciones. Las posibilidades que ofrecen y la versatilidad de las infecciones en un mundo cada vez más conectado ponen a las *botnets* en boca de todos.

En este trayecto hemos repasado algunas de las amenazas informáticas más comunes. Sin embargo, de entre todos los tipos de muestras aquí recogidos, quedarse únicamente con uno de ellos sería imprudente. La principal fortaleza de las nuevas variantes es su capacidad para ejecutar todo tipo de actividades de forma remota y anónima. Así, pese a que en el pasado la diferenciación entre los tipos de *malware* era mucho más evidente en función de para qué objetivo específico eran concebidos, la combinación de sus características al servicio de un fin malicioso común tiene que ser considerada a la hora de dimensionar las amenazas que se ciernan en adelante sobre toda clase de redes y equipos.

Foods Corporation desde 1937 que tuvo bastante éxito entre los combatientes norteamericanos durante la II Guerra Mundial. En 1970, la serie de comedia de la BBC, *Monty Python's Flying Circus*, llevo a cabo un *sketch* en el que se mostraba una cafetería en la que no se servía nada más que *Spam*, motivo que impulsaría a la sociedad moderna a usar ese término en honor de dicho corto.

[1] ROKSO: Register Of Known Spam Operations http://www.spamhaus.org/rokso

3.3 La amenaza de las *botnets*

Los *bots* que nos encontramos en la actualidad y que suponen ya cerca del 25% de los intentos maliciosos de conexión, son una mezcla de amenazas anteriores muy relacionadas con otras áreas del *malware*. Se propagan como gusanos, se ocultan como muchos virus y permiten el control remoto de la máquina infectada por terceras entidades. Estas circunstancias, junto con otras evidencias que relacionan la escritura de código con un trabajo cooperativo —como sucede con SDBot, cuyo código está comentado incluso por distintos autores—, dan pie a la proliferación de un gran abanico de variantes, modificaciones y mutaciones en función del objetivo específico para el que sean escritos.

El ya mencionado informe de Kaspersky señalaba que «en los próximos años el espionaje comercial, el robo de bases de datos y los ataques a la reputación de las empresas tendrán una gran demanda en el mercado negro» [Gos11], para lo cual una de las estimaciones que se realizaba era la utilización creciente de redes *botnet* como principal herramienta de obtención fraudulenta.

Con todo ello, son muchas las empresas punteras del sector que en los últimos seis años han hecho un gran esfuerzo contra la amenaza de las *botnets* a través de asociaciones público-privadas que aglutinan instituciones relacionadas con la industria, la educación, el gobierno y la seguridad de todo el planeta. Fruto de estas colaboraciones, han sido ya numerosas las operaciones policiales que se han llevado a cabo. La primera y una de las más importante fue la «Operación Bot Roast» acometida en 2007 [Off07]. Según el blog oficial de Microsoft [Cra10] se estima que el fenómeno *botnet* —también conocido como *bot-herding* o *pastoreo de equipos*— fue capaz de obtener hasta $780 millones a través del envío de *spam* en el período 2008-2009. Entre el 3 y 21 de diciembre de 2009, Waledac fue responsable del envío de 651 millones de mensajes de *spam* dirigidos exclusivamente a las cuentas de Hotmail enviando ofertas y otras estafas en línea relacionados con las drogas, productos falsificados, puestos de trabajo, acciones, etc.

Por poner un ejemplo, el 25 de febrero de 2010, Microsoft, socios de la industria y las comunidades académicas y jurídicas anunciaron un exitoso esfuerzo de colaboración para desactivar la *botnet* win32/Waledac [Cra10]. Se estima que Waledac había infectado a cientos de miles de computadoras alrededor del mundo y tenía la capacidad para enviar más de 1500 millones de mensajes de *spam* al día.

Al mismo tiempo, también en febrero de 2010, la Guardia Civil espa-

ñola llevó a cabo una acción en colaboración con Panda Labs, por la que desmanteló la *botnet* Mariposa con 12,7 millones de ordenadores comprometidos tanto de usuarios como de empresas de hasta 190 países [Cor10]. La *botnet* Mariposa se convertía así en la más grande operación coordinada por una entidad española en Internet hasta la fecha.

3.3.1 **Usos ciberdelictivos**

La infección de un equipo por parte de una *botnet* puede presentar muy diferentes sintomatologías, algunas de ellas, no necesariamente evidentes. Las nuevas tendencias en el mundo del *malware* están dando lugar a infecciones cuya primera seña de identidad es la de intentar pasar lo más desapercibido posible para el usuario, es decir, no causar ni la más mínima molestia al usuario que le pudiera recordar que tiene la base de virus desactualizada o que no ha instalado el último parche de seguridad de su visor de ficheros .pdf. Por ello, el mayor riesgo que entraña en términos económicos es la gran cantidad de utilidades que se instalan junto con dicho *software*. Hay que ser conscientes de que en la actualidad los ordenadores ya no solamente almacenan información sensible. La mera capacidad de cómputo de los equipos y su conectividad son también activos importantes que pueden ser monetizados de muy diversas formas. Esto, junto con el uso de las técnicas de ofuscación que tienen el objetivo de ocultar los métodos y las características propias del *software* malicioso en sí [UPSB11, UPSB$^+$], incrementan de forma exponencial la complejidad de la búsqueda de muestras maliciosas en el terminal objeto de estudio. Por lo general, las funcionalidades implementadas varían en función de la complejidad del objetivo final real del programador, pero, por lo general, hay que asumir la presencia de diferentes combinaciones de entre algunas de las siguientes características:

- **Soporte DDoS**. En seguridad informática, un ataque de denegación de servicio, también llamado ataque DoS (acrónimo de *Denial of Service*), es un ataque a un sistema informático o red que causa que un servicio o recurso sea inaccesible para los usuarios legítimos. Por lo general causa la pérdida de conectividad de red por el consumo del ancho de banda de la red de la víctima o la sobrecarga de los recursos computacionales del sistema de la víctima. Estos ataques pueden haber sido provocados de forma interna —por ejemplo, un proceso que consuma los puertos de entrada del terminal— o externa —por medio de la generación de un número indeterminado de peticiones

para crear un cuello de botella o *bottleneck* artificial—. Una versión más sofisticada de este último la constituyen los ataques DDoS o de denegación de servicio distribuida —*Distributed Denial of Service*— . Un ataque de estas características consiste básicamente en la ejecución paralela de ataques de denegación de servicio desde orígenes diversos y con un único destinatario y deben ser tratados como problemas de control de congestión cuyos efectos pueden ser bloqueados, o al menos limitados, utilizando técnicas adecuadas en los *routers* para rechazar aquellos paquetes con más posibilidades de ser parte del ataque [IB02, FRBH]. Es el ataque más común perpetrado por ejemplo, por colectivos que operan a nivel internacional como Anonymous, tanto por su sencillez como por su impacto y ha sido empleado contra todo tipo de instituciones, tanto públicas y con fines políticos [Got07, les, Naz07, Mar08, Wil08, Naz09a] como contra organismos privados generalmente relacionados con entidades [Hun08, Pau10, Bak11, Per] que puntualmente dieron cobertura financiera a determinados movimientos contra los que Anonymous ha ido protestando [MD11, Sau11].

- **Robo de credenciales**. El robo de credenciales es una de las más intrusivas vías de monetización de las infecciones. Esta información puede ser obtenida aprovechando la información registrada de forma automática en los equipos, en los formularios del navegador o las claves de registro de algunos programas, pero también existen técnicas que hacen uso de herramientas de *keylogging* para llevar a cabo este robo de credenciales. Entendemos por *keylogger* un dispositivo, bien *software* o bien *hardware*, que a menudo se utiliza como un tipo de demonio de *software* malicioso y cuyo objetivo es abrir una puerta para capturar nombres de usuario y contraseñas registrando las pulsaciones del teclado y guardándolas en un archivo o enviándolas a través de Internet. La información a obtener puede ser muy variada: desde nombres de usuario y contraseñas de cuentas de las redes sociales [BSBK], hasta datos bancarios relativos a las tarjetas de crédito como el número de cuenta, la fecha de expiración de la tarjeta o el código de seguridad CVV [Pem05], pasando por información de similares características relativa a entidades de dinero electrónico [Hyp06] o incluso cuentas de plataformas de juego en línea [RB13].

  En esta línea, los laboratorios de Kasperky, Kaspersky Labs, desactivaron en marzo de 2012 el canal de *Command & Control* de Hlux/-

Kelihos, una *botnet* con la peculiaridad de estar especializada en el robo de carteras de Bitcoin [Kas12]. Bitcoin se puede definir como una criptodivisa distribuida ajena a cualquier tipo de control centralizado [Bre12, BGB] que permite hacer pagos empleando como medio un protocolo criptográfico diseñado a tal fin por Satoshi Nakamoto [Nak09].

Teniendo en cuenta que Bitcoin desarrolla un sistema de pago con el objetivo final de sustituir y no solamente simular el funcionamiento de las divisas tradicionales, las implicaciones y posibilidades ciberdelictivas son todavía mayores. Aún siendo conocida principalmente en entornos *underground*, la repercusión de esta criptodivisa ha dado lugar a un volumen de transacciones valoradas en alrededor de 3,5 millones de dólares diarios a principios de 2013[1] con una capitalización del mercado que ronda los 500 millones de dólares.

- **Ejecución remota de programas protegidos**. Con objetivos muy diferentes, las *botnets*, haciendo uso de sus capacidades de propagación como gusanos, pueden ser el primer paso de una infección mucho más profunda en el sistema. Esto es así dado que los atacantes pueden obtener el control total de los equipos infectados para realizar diferentes operaciones encubiertas sin que la víctima se percate de ello, como la actualización del *software* o la descarga de nuevos contenidos maliciosos.

- **Ciberespionaje**. El espionaje económico —o industrial— es un concepto recurrente en la historia de las empresas. Miguel Ángel Esteban [Nav07] lo definía en el *Glosario de Inteligencia* del Ministerio de Defensa de España como: «[Toda aquella] actividad sistemática de obtención clandestina de información protegida por su propietario en el ámbito de la inteligencia económica. El espionaje económico está prohibido para la práctica de la inteligencia competitiva, ya que su realización exige realizar acciones ilegales o alejadas de la deontología profesional como la corrupción, el soborno, la violación de comunicaciones, etc.». Los tiempos actuales no son ajenos a la gran cantidad de información que puede ser recopilada proveniente de las grandes corporaciones. Esto hace que muchas *botnets* se centren en la compra-venta de información sensible extraída de las organizaciones

[1] Información extraída de la página web Blockchain.info que monitoriza la información económica relativa al mercado de Bitcoins. URL: http://blockchain.info/es/stats

explotando las posibilidades que ofrece el llevar a cabo tareas de recopilación de información y de facilitación del fenómeno de la fuga de datos.

- ***Click-frauding***. El *click-frauding* hace referencia a un tipo de fraude en Internet que consiste en la generación artificial de tráfico fraudulento y en el que el objetivo del ataque son los servicios de *pay-per-click* —como por ejemplo Google AdSense[1]—. En verdad, el *click-frauding* se puede realizar manualmente o con la ayuda de ciertos tipos de *software*, entre los que también podemos incluir las redes de *bots*. Por medio de estas herramientas, los estafadores pueden manipular los sistemas de facturación que están detrás del destinatario de la publicación aumentando artificialmente el tráfico, el número de impresiones y/o la cantidad de clics ejecutados por los usuarios. La ventaja de utilizar ordenadores *zombie* reside en que los sistemas de detección de fraude que usan las compañías tendrán muchas más dificultades para identificar estos comportamientos al camuflarse las conexiones detrás de diferentes orígenes y no poderse utilizar como criterio de clasificación la procedencia del clic en sí misma.

- **Explotación de la capacidad de cómputo**. Hoy en día, los escritores de *malware* han encontrado nuevas formas de monetizar las infecciones y de sacarles valor. Una de las más recientes, es la utilización de estas para explotar la capacidad de cómputo de las máquinas infectadas en el desarrollo de cálculos que generan empleando los tiempos muertos de ejecución de forma similar a como se usan en proyectos de computación distribuida[2] [LS04] sin fines maliciosos. Al respecto, la sección de Ciberinteligencia e Inteligencia criminal —en inglés, *CyberIntelligence and Criminal Intelligence*— del FBI [SS12] ya identificaba en un informe filtrado a mediados de 2012 que existían muestras de Zeus[3] especialmente destinadas para operar en segundo plano utili-

[1] Google Adsense es un servicio de publicidad en línea de Google que permite a los editores obtener ingresos mediante la colocación de anuncios en sus sitios web, ya sean en forma textual, de imágenes o de otro tipo de publicidad interactiva más avanzada.

[2] La computación distribuida es un modelo para resolver problemas de computación masiva utilizando un gran número de equipos conectados a una red de telecomunicaciones distribuida en la que los terminales empleados están separados físicamente. Una característica fundamental de estos equipos es que sean muy confiables y tolerantes a los fallos que se puedan derivar de la caída de uno de los nodos de la red sin que esto conlleve grandes implicaciones para la continuación de las tareas de cálculo.

[3] Versión del troyano Zbot especializada en el reclutamiento de equipos como parte de

zando los recursos computacionales de la víctima en la sombra. Hay que tener en cuenta que, a día de hoy, la obtención de *bitcoins* empleando la capacidad de cómputo no es considerada como una opción rentable si no se cuenta con el *hardware* de computación GPU apropiado. De no ser así, los ingresos obtenidos no son capaces de cubrir los costes de energía de la generación. Sin embargo, si dicha actividad se realiza ejecutando en los ordenadores de las víctimas, clandestinamente y en segundo plano, los beneficios obtenidos pueden ser aprovechados íntegramente por el atacante evadiendo así los costes de generación a los que tendrá que hacer frente la víctima.

Todas estas utilidades pueden emplearse también como herramientas para llevar a cabo ataques con fines militares. De hecho, los sucesos acontecidos en Estonia en 2007 [Naz07] pusieron de manifiesto que la utilización de las *botnets* podía poner en jaque la estabilidad y el normal funcionamiento de las instituciones de todo un país (con independencia de cuál fuera el patrocinador del ataque). A raíz de aquellos eventos, se creó por parte de la OTAN/NATO el Centro de Excelencia para la Cooperación en Cyber Defensa —CCDCOE, por sus siglas en inglés: *Cooperative Cyber Defence Centre of Excellence*— con sede, desde agosto de 2008, en la capital de Estonia, Tallin.

3.3.2 **Comportamiento general**

Profundizando en lo que respecta a la construcción de una *botnet*, se pueden distinguir tres etapas principales en su período de maduración antes de que estas sean completamente funcionales.

1. **Selección de candidatos**. Esto se hace normalmente a través de binarios compartidos en otras redes P2P, *parasitando* el protocolo implementado para transferencia de archivos tradicionales [WWACZ07]. Sin embargo, el tamaño de la *botnet* se limitará a la cantidad de usuarios conectados a ese servidor en particular, algo que ha dado lugar a nuevas formas de propagación, empleando el correo electrónico o la mensajería instantánea y usando como vehículo la compartición de archivos troyanizados incluso a través aplicaciones de carácter social.

2. **Infección**. Para ello, se hace uso de troyanos, gusanos y/o virus y la

una *botnet* y originalmente concebido para la sustracción de información bancaria.

ayuda habitual de las siempre efectivas técnicas de ingeniería social[1]. El atacante tratará de explotar la inexperiencia de los usuarios menos conscientes, reclutando nuevos miembros para asegurar así el éxito de la infección y la expansión de la red. Los creadores de *malware*, por lo general, fomentan la promoción de sus *botnets* ofreciendo contenidos y archivos de alta demanda en Internet como ficheros *crackeados* de videojuegos, generadores de código, películas, imágenes, etc. que, en verdad, contienen archivos infectados.

3. **Propagación de la *botnet***. Si el equipo recientemente infectado, no pertenece a ninguna red P2P, se facilita el contacto con el *botmaster*, aplicando la técnica tradicional de las conexiones *Peer To Peer*[2]: una lista de las ubicaciones de los miembros de la *botnet* puede ser accedida, bien porque haya sido previamente codificada en el propio binario malicioso —en inglés, *hardcoded*— como hace Trojan.Peacomm [GSN+07], o bien porque haga uso de un archivo externo de configuración. La contrapartida principal de este enfoque es que si esta lista cae en manos de los defensores, la integridad completa de la red de *bots* y su expansión estaría totalmente comprometida. Como alternativa, las listas actualizadas dinámicamente ofrecen muchos aspectos interesantes desde el punto de vista de su protección y estabilidad. Se trata de listados actualizados por combinación entre aquellos que poseen sus vecinos y que generan nuevas partes de la lista de miembros a contactar sin poner en peligro el entramado de nodos que componen la red.

3.3.3 Arquitectura típica de una *botnet*

La principal fortaleza de las *botnets* radica en el potencial de tener una red de ordenadores conectados muy polivalente y que se pueden controlar con

[1] La ingeniería social es una técnica por la que se trata de obtener información, acceso o privilegios en un sistema mediante la aplicación de la máxima de que «los usuarios son el eslabón débil» en la cadena de seguridad. En la práctica, un ingeniero social usará comúnmente el teléfono o Internet para manipular a su víctima, fingiendo ser, por ejemplo, empleados de algún banco, de empresas subcontratadas o compañeros de trabajo. La gran disponibilidad de información laboral existente en redes sociales como Linkedin puede ser utilizada para perpetrar estos ataques [HKNT].

[2] Una red peer-to-peer, P2P por sus siglas en inglés, es una red de computadoras en la que en todos o algunos aspectos funcionan sin clientes ni servidores fijos, sino una serie de nodos que se comportan como iguales entre sí. Es decir, actúan simultáneamente como clientes y servidores respecto a los demás nodos de la red.

gran fluidez y versatilidad de forma remota. Con todo, los ataques que se han venido perpetrando contra proveedores de servicios bancarios en la red como medida de represión contra las restricciones interpuestas por numerosas compañías a Wikileaks [Sei, BS], pusieron de manifiesto que, aún en el año 2011, los canales de *Command & Control* que empleaban como medio IRC seguían siendo utilizados para transmitir órdenes en las *botnets* empleadas por el colectivo Anonymous [Lil, Inf11]. Sin embargo, se hace necesario desarrollar técnicas más sofisticadas para el desarrollo de los nuevos sistemas de detección de *botnets* para afrontar las nuevas técnicas de *Command & Control* que están siendo empleadas por los escritores de *malware*: desde el uso de mensajes privados en las redes P2P, hasta la parasitación de servicios de terceros como Twitter o Pastebin, pasando por toda clase de peticiones HTTP contra servidores web u otros canales tradicionales.

Para lograr este objetivo, hay visiones muy diferentes sobre cómo tratar los problemas de comunicación entre las entidades pertenecientes a la red de *bots*, por lo que existen distintas arquitecturas tal y como se puede ver en la figura 3.3.

- **Centralizada** —fig. 3.3(a)—. En las topologías centralizadas, todos los nodos infectados se conectan directamente a un servidor central controlado por el *botmaster*. Son menos complejas desde el punto de vista de su arquitectura pero crean un único, a la par que crítico, punto de fallo. Por lo general, se define como arquitectura centralizada a aquella que consiste en un nodo central que distribuye los mensajes entre los clientes de la red. Se caracterizan por:

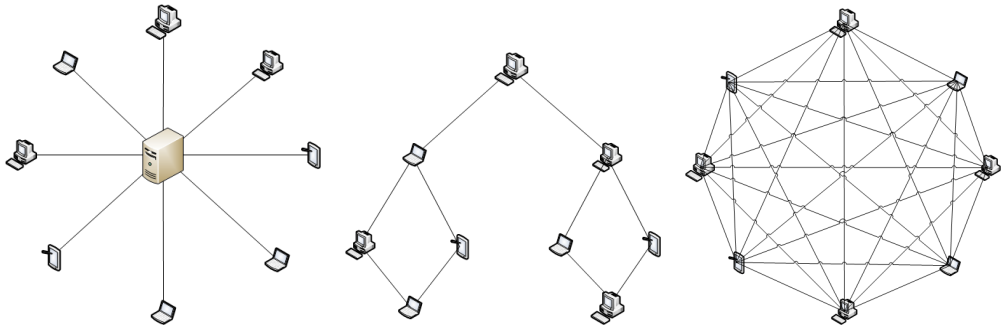
  - Baja latencia, debido al pequeño número de saltos necesarios para transmitir las órdenes del *botmaster*.
  - Conexión directa para enviar las órdenes a los nodos de distribución, lo que en la práctica podría comprometer la seguridad de la red en caso de detección accidental de un nodo.
  - Implementación a través de diferentes protocolos de comunicación, aunque la mayoría suelen utilizar IRC y HTPP (o HTTPS).

  La centralización a menudo no es completa. En función del tamaño de la red y de la complejidad de las operaciones que se requieran se puede dar lugar a estructuras centralizadas dependientes de un nodo central.

- **No estructurada** —fig. 3.3(b)—. Una topología descentralizada o no estructurada provee una protección adicional y un incremento de la versatilidad de la que puede hacer uso el *botmaster*, quien podrá desplegar sus comandos a través de diferentes máquinas que actúen como servidores. Esta arquitectura lleva al extremo el concepto de las redes P2P, ya que trata de hacer invisible la mera existencia de la red de comunicaciones entre los propios sistemas infectados, incluso en el seno de su propia *botnet*. La idea es que cada vez que el *botmaster* quiere transmitir algún tipo de comando, los cifre, a la espera de que cualquier escaneo aleatorio realizado con éxito a través de Internet detecte otro bot perteneciente a la misma *botnet*. Así, se evita comprometer la seguridad de toda la red en el caso de interceptación de un miembro de ella. La principal contrapartida de llevar a cabo una implementación de estas características radica en la creciente complejidad de la red, dado que su gestión se vuelve más difícil al existir una latencia mayor. Esto implica que a la hora de perpetrar ciertos ataques sean notablemente menos eficaces.

- ***Peer-to-peer*** —fig. 3.3(c)—. En una topología *peer-to-peer* todos y cada uno de los nodos de una red pueden actuar, bien como clientes o bien como servidores. La principal ventaja de este enfoque radica en la flexibilidad y en la capacidad para afrontar su detección empleando técnicas basadas en el análisis de las conexiones. Tienen la ventaja de ser más difíciles de desestabilizar, ya que no tienen una base única y central desde la que enviar las órdenes de emisión y/o compartición de recursos e información, haciendo uso de las instalaciones de las tradicionales redes P2P que permiten unos *ratios* de conexión y desconexión elevados sin que se resienta la funcionalidad de la misma. Adicionalmente, cada nodo tiene una mayor complejidad estructural porque todos ellos pueden actuar como ambos, cliente y servidor siendo más difíciles de interceptar y estudiar. Algunos ejemplos de arquitecturas de redes P2P son la mencionada Trojan.Peacomm [GSN+07] y Stormnet. Hay que tener en cuenta que el hecho de contar con una arquitectura P2P no obliga de ninguna forma a que la emisión de los comandos se haga a través de estas redes de intercambio de archivos al uso (eMule, Kazaa, etc.), siendo posible encontrar casos de redes de *bots* con una arquitectura P2P cuyo canal de comunicación es implementado a través de canales IRC tal y como se detalla más adelante.

Los canales de comunicación de órdenes pueden aplicar una de dos fi-



(a) Centralizada. (b) No estructurada. (c) P2P.

Figura 3.3: Diferentes topologías para la implementación del canal de *Command & Control* de una *botnet*.

losofías: *push* o *pull*. En las páginas que siguen se detallarán los elementos que caracterizan a cada una de ellas.

3.3.3.1 La filosofía *pull*

La filosofía *pull* es también conocida como de *publicación/suscripción*, debido a la forma de obtener las nuevas listas de instrucciones a llevar a cabo. Estas son publicadas por el *botmaster* y convenientemente notificadas a través de un servicio de suscripción. Su uso, especialmente común en las redes centralizadas, puede variar en función del protocolo mediante el cual el canal de C&C se implementa en la red de *bots*:

- **Basados en IRC**. Los *bots* están conectados periódicamente a un canal IRC dado, a la espera de más instrucciones, siendo capaces de retransmitir las órdenes en tiempo real, ya sea a través de mensajes privados (PRIVMSG IRC) o mediante la publicación de mensajes temáticos (TOPIC). Necesitan contar por tanto con un cliente IRC que les dé acceso a los protocolos de chat. Estos clientes son a menudo encapsulados en el propio *malware*.

- **Basados en redes de intercambio de archivos P2P**. En el momento en que se quiere realizar la búsqueda de un archivo, una consulta se transfiere a través de la red, recibiendo una respuesta positiva si un miembro de la misma red de *bots* la contiene. Es entonces cuando se inicia el protocolo de comunicación al haberse utilizado la posesión de dicho archivo como mecanismo de identificación de los nodos en el seno de la red.

- **Basados en HTTP**. El *botmaster* informa a los ordenadores infectados de la nueva lista de comandos a ejecutar al actualizar los contenidos de una página web que los *bots* visitan de forma periódica. Esto expone a los nodos de los canales de comunicación ya que el acceso a esta web es a menudo rastreable con facilidad mediante la implantación de técnicas de monitorización de trafico que permitan decidir qué tipo de URL bloquear. Para dar respuesta a esta problemática, los escritores de *malware* han dado paso a técnicas que permitan camuflar el destino real de las conexiones mediante técnicas de *fast-flux*.

Mención especial merecen las *botnets* parasitarias, cuyo canal de comunicación es también bastante simple. En primer lugar, un *bot* es seleccionado para realizar una búsqueda específica por un título concreto —que puede ser fijo de antemano o ser calculado por algún tipo de algoritmo— y que permite a cada *bot* identificar 1) otro archivo que se debe encontrar para revelar los comandos para ejecutar o 2) la respuesta de búsqueda la que el comando enviado se codificará sin necesidad de descargar ningún archivo. Esta filosofía usa *in-band messages* —comunes en el tráfico P2P— que permiten que el envío y recepción de comandos se confunda con el tráfico legítimo dentro de la red, dificultando su detección y complicando el análisis del tráfico, al requerir este enfoque de un proceso adicional de filtrado previo para evitar el ruido.

3.3.3.2 La filosofía *push*

Esta filosofía es también conocida como de escucha pasiva o *passive listening*. Este enfoque se caracteriza porque los nodos esperan a que el operador les contacte para transferirles las instrucciones a ser ejecutadas y que, en su caso, estas sean transmitidas a los demás miembros de la red. Se evita así la comunicación a través de conexiones regulares con el servidor central y se reducen, por tanto, las posibilidades de detección mediante el modelado del tráfico de salida de la computadora infectada. En una red de intercambio de archivos, la decisión de cuáles serán los elementos infectados que serán informados de los nuevos comandos a ejecutar se puede simplificar tratando de encontrar un archivo en particular que esté presente en los discos duros de todos los miembros de esa misma *botnet*. Así, se enviarán sólo a los miembros que poseían dicho archivo las directivas apropiadas ya que habrán sido identificados como miembros de la red de *bots*.

3.3.4 Evolución de los canales de *Command & Control*

La comunicación entre los *bots* ha sufrido grandes cambios desde el uso de los clientes de IRC, a menudo creadas sobre la marcha por los creadores de *malware* para alcanzar mayores niveles de anonimato y la independencia de algún punto más complejo de conexiones punto a punto. Además, para facilitar el crecimiento exponencial de tales redes, es necesario asegurarse de que esas órdenes enviadas por el *botmaster* no se pierdan sin ser recibidas, interpretadas y/o ejecutadas por el mayor número de pares infectados posible. Es precisamente un mecanismo de comunicación eficaz entre los *zombies* el que determina, a la vez, la topología de red y su capacidad para evitar la detección y asegurar su solidez mediante la apertura de nuevos canales de *Command & Control*. Para satisfacer este objetivo se analizan a continuación los canales de diseminación de comandos más habituales para la propagación de las órdenes siguiendo la clasificación propuesta por el autor en artículos ya publicados [BSBdV, BGdlPS+].

3.3.4.1 IRC

Las *botnets* basadas en IRC —del inglés, *Internet Relay Chat*— supusieron el estadío inicial de la amenaza de las *botnets* con un impacto más notorio entre los años 2005 y 2008 [ZHH+]. Los nodos infectados que formaban parte de estas redes se conectaban de forma periódica a un canal de IRC a la espera de nuevas órdenes a ejecutar, siendo capaces de retransmitir los comandos al vuelo, bien a través de mensajes privados de IRC (PRIVMSG IRC) o bien a través de la publicación de mensajes temáticos (TOPIC). Por aquel entonces, los *bots* eran una de las principales herramientas utilizadas para la ejecución de ataques distribuidos de denegación de servicio necesitando, muchas veces como única herramienta, versiones ligeras de clientes de IRC. Algunas de las familias más conocidos se recogen a continuación:

- **GTbot**. Abreviatura de *Global Threat bot*, GTbot se puede definir como una versión de mIRC —habitualmente renombrada como *temp.exe*— que era ejecutada en modo sigiloso y de forma oculta. Utilizada con frecuencia junto con el programa Hide Window [Cas] para permitir su ejecución sigilosa, contenía un cierto número de *scripts* para mIRC que son los que le daban la verdadera funcionalidad al *malware*. Este programa era habitualmente descargado por usuarios de las salas IRC cuando, mediante técnicas de ingeniería social se les instaba a ejecutar una aplicación que creían inocua. Una vez instalado, el troyano se

unía por sí solo a un canal de IRC a la espera de nuevos comandos para ser ejecutados por el administrador de la *botnet*.

- **SDbot.** SDbot es un troyano de puerta trasera que permite a un atacante conseguir acceso remoto al equipo infectado para ejecutar toda clase de acciones maliciosas que comprometan la confidencialidad del usuario y/o le impidan desarrollar sus tareas habituales. Para ello, SDbot también utiliza su propio cliente de IRC como canal de C&C. Entre sus funcionalidades maliciosas se incluyen el lanzamiento de ataques de denegación de servicio así como la descarga y posterior ejecución de otros ficheros infectados en el equipo. De cualquier manera, SDbot requiere de la intervención del usuario para propagarse. Esto ha derivado en una gran cantidad de metodologías que emplean una gran cantidad de herramientas que van desde disquetes o CD-ROM, hasta mensajes de correo electrónico con adjuntos infectados, descargas de Internet, transferencias de ficheros vía FTP, canales IRC, aplicaciones de compartición de ficheros o redes P2P, etc.

- **Agobot**. Agobot, habitualmente referido también como Gaobot, es una familia de gusanos informáticos cuyo código fuente fue liberado bajo la versión 2 de la GNU GPL (*General Public License*) de la Free Software Foundation[1] especialmente diseñados para el despliegue de una *botnet*. El propio código fuente del *malware* se describe como «un *bot* IRC para Win32/Linux» y su primera implementación conocida es atribuida a Axel «Ago» Gembe, un programador alemán también conocido por apropiarse del código fuente del videojuego Half Life 2 de Valve [Pou08]. Agobot está escrito en C++ y unas partes en ensamblador y hace uso de programación orientada a objetos con soporte multihilo. Navegando por el proyecto de su distribución, un programador medio puede identificar con facilidad la utilidad de cada uno de los módulos *.cpp* que implementan las diferentes funcionalidades del *software*. En este sentido, Agobot era otro ejemplo de una *botnet* que requería de pocos o muy escasos conocimientos de programación para ser usada y/o modificada con rapidez dada la modularidad con la que fue concebido.

- **Rbot**. Rbot es un gusano que se expande a través de las comunicaciones internas de una red. El programa buscaba y listaba las carpetas

[1] La Free Software Foundation es una organización sin ánimo de lucro que defiende la utilización de licencias libres frente a licencias privativas.

compartidas por algunas de las aplicaciones de compartición y descarga de ficheros más conocidas (especialmente P2P) en donde dejaba una copia de sí mismo para facilitar su difusión como una descarga legitima adicional.Disponía de algunas características que le permitían actuar de puerta trasera capaz de desplegar funcionalidades de *botnet* empleando IRC como canal. El gusano utilizaba sus propios medios para conectarse y configurar un canal de IRC a la espera de las órdenes que le enviaría el atacante posteriormente. Entre otros ataques, algunos de los comandos que se podían ejecutar en las máquinas infectadas se puede incluir la ejecución de ataques de denegación de servicio distribuida, la obtención fraudulenta de claves de registro (como identificadores de diferentes productos comerciales así como *CD-Keys*), la adición o el borrado de nuevas descargas compartidas, grupos o usuarios, etc.

3.3.4.2 **P2P**

Tradicionalmente propagadas a través de la compartición de binarios infectados en redes punto a punto que utilizan el protocolo P2P. Tienen la ventaja de ser más difíciles de desestabilizar al no tener un único núcleo desde el que enviar la información o desde el que compartir recursos o información. Existen ejemplos muy exitosos de redes con esta arquitectura que son capaces de aprovechar las facilidades que ofrecen las redes P2P tradicionales para explotar unos elevados *ratios* de conexión y desconexión, como lo son Trojan.Peacomm y Stormnet [GSN+07].

Mención especial merecen en este punto las conocidas como *botnets parasitarias* [WWACZ07, WWACZ09]. Este tipo de redes utilizan como canal de comunicación servicios de terceros que están permanentemente *online* para comunicarse con las víctimas. En el caso de las aplicaciones de compartición de ficheros, el proceso se desarrolla como sigue. En primer lugar, un *zombie* es seleccionado para ejecutar una búsqueda específica de un título en dicha red. Este titulo puede ser algún documento dado o calculado de una forma aleatoria por algún tipo de algoritmo generador que pueda ser identificado posteriormente. La ejecución de la búsqueda puede tener dos objetivos: comunicar a las víctimas qué fichero han de encontrar para dar con los comandos a ejecutar o indicar al resto de los nodos de la red en qué respuesta de búsqueda se van a codificar estos, lo que evitaría a los *bots* la descarga del fichero de órdenes. De esta manera, el *botmaster* solamente recibirá respuestas afirmativas si un miembro de la misma *botnet* es

capaz de procesar estos comandos para iniciar el protocolo de comunicación. Esta filosofía, utiliza los llamados *in-band-messages* —comunes en el seno del tráfico P2P— para permitir el envío y recepción de órdenes [NR09]. La ventaja es que el tráfico se puede confundir con facilidad entre el tráfico legitimo de la red, dificultando su detección y complicando también el análisis de tráfico al requerir de grandes cantidades de recursos para analizar la totalidad de los mensajes *al vuelo* y de un proceso de prefiltrado de las comunicaciones.

En esta línea, Shishir Nagaraja *et al.* [NMH+10] trabajaron en la detección de redes P2P desarrollando investigaciones para tratar de detectar si los ISP —Internet Service Provider— o proveedores de Internet eran capaces de detectar las eficientes estructuras de comunicaciones de las redes P2P de cara a utilizar a estos como una primera línea de defensa ante las *botnets*. Los autores trataban de explotar las capacidades de los ISP, de los administradores de las redes corporativas y de los IDS para detectar patrones en estas comunicaciones debido a la gran cantidad de tráfico malicioso que circula por las instalaciones virtuales que están bajo su control, identificando estos como un buen punto de partida para detectar este tráfico.

Un ejemplo de este tipo de arquitecturas es IMMONIA. IMMONIA se puede definir como un *bot* de arquitectura *peer-to-peer* diseñado para expandirse por sus propios medios empleando pataformas IRC. Escrito originalmente en AutoIt v3[1]. Siendo codificado empleando un lenguaje de *scripting*, su código[2], atribuible a un programador conocido como Hypoz en el año 2011, está plenamente documentado y ampliamente comentado. El aspecto más interesante a resaltar es su estructura P2P descentralizada lanzada desde plataformas IRC implementando una gran complejidad estructural y permitiendo que cada uno de los nodos de la red puedan comportarse tanto como cliente como servidor, lo que dificulta su intercepción, monitorización y estudio.

La evolución natural de estas estructuras tenderá a la explotación de recursos y servicios que estén siempre activos. Aplicaciones como Twitter, Facebook o Pastebin, con una gran cantidad de usuarios legítimos, se convierten aquí en un marco ideal para camuflar este tipo de comunicaciones maliciosas [BGdlPS+].

[1] AutoIt v3 es un lenguaje de *scripting* similar a BASIC concebido para la automatización de tareas relacionadas con la interfaz de Windows entre otras tareas propias de los lenguajes de *scripting*. Su extensión por defecto es *.au3*.

[2] A fecha de esta publicación aún disponible en la dirección http://www.mediafire.com/?1wdq1o076amww03.

3.3.4.3 **HTTP/HTTPS**

En las *botnet* basadas en HTTP el *botmaster* informa a los nodos infectados de las nuevas órdenes a ejecutar por medio de la actualización de los contenidos en una página web que los ordenadores *zombie* visitarán periódicamente. Algunos de los ejemplos más modernos son Flu y Prablinha —con los que trabajaremos en esta tesis doctoral— y el propio Zeus. La *botnet* Zeus es una evolución de un troyano conocido por el robo de información bancaria por el método de *man-in-the-browser* y de la captura de datos procedentes de los formularios, siendo principalmente distribuido a través de sitios de descargas o por medio de esquemas de *phising*.Identificada en primer lugar en julio de 2007 cuando fue usado para extraer información del Departamento de Transporte de los Estados Unidos [Off07], aunque no fue hasta marzo de 2009 cuando se consumó su expansión definitiva.Por aquel entonces, Zeus había infectado ya más de 3,6 millones de ordenadores solamente en Estados Unidos. Binsaleeh *et al.* presentó en 2010 un análisis de ingeniería inversa sobre el *toolkit* de herramientas de Zeus y lo definió como «la más novedosa y poderosa herramienta para el cibercrimen que ha emergido en la comunidad *underground* de Internet para el control de *botnets*» [BOB+10].

Su código fuente, con comentarios en ruso, fue filtrado en mayo de 2011 tras ser vendido en diferentes foros *underground* [Sel]. Bien balanceado y programado, el canal de *Command & Control* que puede utilizar el atacante ha sido escrito en PHP mientras que el *bot* que correrá en las máquinas infectadas fue codificado en C++. Si se hace caso a su documentación, los desarrolladores de Zeus han llegado a trabajar en más de 15 actualizaciones y modificaciones de dicho *malware* solamente en 2010 [Rag] incluyendo soporte para IPv6, FTP o POP3.

Las siguientes son solamente algunas de las principales funcionalidades que implementan algunas de las versiones de Zeus: intercepción de tráfico del navegador web, FTP o POP3 entre otros; modificación de las páginas legítimas al vuelo mediante técnicas de inyección de *iframes* o *HTML injection*[1].

[1] La inyección de código HTML es a menudo conocida como *Cross-Site Scripting* o XSS —para la captura de información bancaria o la redirección del sistema hacia páginas controladas por el atacante—; recolección de información personal en la máquina infectada, bien mediante técnicas de *keylogging* o bien recopilando toda clase de información abierta como *cookies* del navegador y certificados o monitorización de cuándo una de las víctimas está intercambiando información con un banco determinado para poder capturar sesiones al vuelo.

3.3.4.4 Nuevas tendencias: *malware* social y dispositivos móviles

La expansión de las redes sociales ofrece un especial atractivo para los creadores de nuevas aplicaciones maliciosas. De hecho, los *botmasters* están empezando a explotar las redes sociales como Twitter [Naz09b] como nuevos medios para la divulgación de comandos. Se crea así el concepto de *parasitación* de las redes sociales. Como la existencia de estas, por su propia naturaleza, depende de estar disponibles permanentemente *online* veinticuatro horas al día y siete días a la semana, los programadores están encontrando en ellas una nuevo medio para albergar con éxito los canales de *Command & Control* que las gobiernan [KMXS10] con un riesgo de desactivación mucho menor y grandes posibilidades de evitar la detección camufladas en tráfico de red legítimo. Conscientes de esta situación, las nuevas redes sociales emplean técnicas que tratan de poner coto a la creación de cuentas de usuario de forma automatizada empleando diferentes metodologías [BPH+, SKV].

Adicionalmente, hay que considerar la realidad de que el usuario es el vector de infección de gran eficacia, en parte porque los propios usuarios confían en las sugerencias de sus contactos de cara a visitar y chequear los enlaces que reciben. Esto es de gran relevancia si además se tiene en cuenta que los usuarios tendemos a ofrecer grandes dosis de credibilidad a los contenidos que nos sugieren nuestros contactos por encima de aquellos contenidos sugeridos por páginas desconocidas. Esta realidad, que aprovecha la vulnerabilidad del usuario de las redes sociales como principal vector de infección, ganará más adeptos en el futuro [LLLF, FLJ+].

A esto hay que sumar la creciente evolución del *malware* para dispositivos móviles [Dun09, FFC+] y la falsa sensación de muchos usuarios de que estos dispositivos son más seguros que sus equipos de sobremesa [CFSW] pese a que la información almacenada es, habitualmente, de un carácter especialmente privado: SMS, fotografías, números de telefóno, etc. Algunos estudios, como el de Wilson *et al.* [WY11] ponía de manifiesto que solamente el 33,9% de los encuestados era consciente que, de alguna forma, podía llegar a infectarse su dispositivo móvil. Si nos fijamos en la cantidad de usuarios con algún tipo de *software* antivirus instalado, la cifra apenas llegaba al 15,4% del total de usuarios, poniendo de manifiesto nuevamente la escasa concienciación acerca de la amenaza.

Además, se dan otras circunstancias que alimentan este fenómeno. El incremento de las capacidades de estos dispositivos así como su uso cada vez más extendido [Lea05, Law08] y la facilidad de monetizar las infecciones por medio de la facturación de servicios fraudulentos de tarificación

adicional empleando, por ejemplo, SMS, los trabajos de autores como Yong *et al.* [Yf06] y Lian *et al.* [Ly08] han dado lugar al desarrollo de numerosas técnicas para la detección de este tipo de comportamientos basadas en la identificación prematura de estos [ASS, ZWHZ] .

El mundo de las *botnets* no ha sido ajeno a estas realidades [LT11]. Algunos autores han llegado a diseñar arquitecturas distribuidas que empleaban únicamente el envío de mensajes cortos o SMS como medio para la transmisión de comandos en *botnets* con arquitecturas P2P [ZSH]. Sin embargo, las capacidades de los nuevos dispositivos, cada vez más próximas a las de equipos de sobremesa, y su utilización para efectuar todo tipo de transacciones *online*, dejan caer que no será esta la única forma de infección que preocupará a los analistas de seguridad del futuro y que estas irán asemejando su complejidad y alcance a la del *malware* ya existente para ordenadores personales.

3.4 Principales contramedidas y líneas defensivas

En el momento de la detección y seguimiento de redes de *bots*, hay dos enfoques que aplican dos técnicas diferentes: la activa, por lo general basadas en *honeypots* como la desarrollada en The Honeynet Project[1] [Spi03] y la pasiva que consiste en una monitorización exhaustiva del tráfico de la red. Así, el método activo trata de detectar y luego de ganar acceso a una *botnet* simulando, por ejemplo, el funcionamiento de un *bot*. Sin embargo, dada la naturaleza de la conexión, este comportamiento puede ser detectado por los operadores de la *botnet* y ponerlos sobre aviso para que puedan tomar medidas adicionales e impedir futuras incursiones. Por el contrario, la vía pasiva se basa en el estudio de los flujos de información en el entorno de la red, sin establecer vínculos de comunicación directa con los nodos infectados y observando el tráfico de red que llega a la *darknet* [SG09].

El término *darknet* es comúnmente utilizado para definir una serie de redes y tecnologías que permiten a los usuarios copiar y compartir contenido digital sin que sea posible conocer ciertos detalles de las descargas que se han hecho ni el origen de los usuarios que lo perpetraron. Aún así, Seewald *et al.* [SG09] lo utilizan en su publicación para definir un rango de IP que no están disponibles para las máquinas reales y que en caso de que sean accedidas, esto ocurra porque dichos enlaces corresponden a conexiones maliciosas.

[1] http://www.honeynet.org/

La principal ventaja de la aplicación puramente pasiva de este enfoque es que su existencia no puede ser detectada por el *botmaster* dado que no hay comunicación con los miembros de la red. Esto permite extraer mucha información interesante con respecto a la estructura de la red así como para el establecimiento de mecanismos de alerta temprana y prevención en el caso de detectar la contaminación de la misma. Sin embargo, se encontró que los enfoques un poco más flexibles, como los consistentes en el reenvío de paquetes ACK para aceptar conexiones entrantes [BCJ+05] aumenta significativamente el volumen de información para analizar a costa de aumentar al mismo tiempo el riesgo de ser detectado por el *botmaster*. En este sentido, algunas de las líneas principales de trabajo futuro posibles en la detección de *botnets* están determinadas por la identificación de ciertos síntomas que son comunes a la mayoría de las infecciones.

La detección de tráfico DNS puede ser un buen punto de partida en el análisis de los intentos de conexión de cada entidad de la *botnet* a la red en sí misma, pero sólo será eficaz si los canales de comunicación y de *Command & Control* se conocen de antemano. Esto permitiría trabajar en la misma línea que se ha hecho para detectar intrusiones en, por ejemplo, redes corporativas y públicas con mecanismos de detección de tráfico anómalo [Bri, BPPS].

A niveles de red local, es interesante la aplicación del concepto de *honeypot* o *tarros de miel*. En seguridad, se denomina *honeypot* al *software* o conjunto de computadores cuya intención es atraer a los atacantes, simulando ser sistemas vulnerables o débiles a los ataques. Se trata de una herramienta de seguridad informática utilizada para recoger información sobre los atacantes y las técnicas que estos utilizan. De esta manera, los *honeypots* pueden distraer el foco del ataque de las máquinas más importantes del sistema, y advertir rápidamente al administrador de este de que se está produciendo un incidente de seguridad, además de permitir un examen en profundidad del atacante, durante y después de la crisis que afecta al *honeypot*. Su utilidad radica en que es bastante común encontrar intentos de infección en aquellos equipos que pertenecen a la misma LAN que un *bot* que ha sido secuestrado y que se mantiene operativo, por lo que pueden ser utilizados para detectar todo tipo de actividades maliciosas colocándolos como cebos [dCdFD+].

3.4.1 Técnicas de detección de *botnets*

Una vez diferenciados los dos principales enfoques de análisis de las *botnets*, en esta subsección se procede a enumerar algunos de los métodos de detección documentados ya utilizados en el pasado, así como sus limitaciones y posibles áreas de expansión del conocimiento. Estos métodos incluyen tanto, un enfoque más tradicional, como puede ser la detección por medio de firmas y otro más centrado en el análisis de tráfico en función de las características de la información enviada y recibida.

3.4.1.1 Detección por medio de firmas

Efectivo y todavía en uso, el uso de la detección de firmas se enfrenta actualmente a los desafíos planteados por las técnicas modernas capaces de evitar los sistemas polimórficos. Sin embargo, la vigilancia pasiva ha demostrado tener aún algunas historias de éxito muy relevantes. Goebel *et al.* [GH07] ha sabido combinar las listas de expresiones regulares en los apodos de IRC —previamente etiquetados como sospechosos—, junto con un análisis de *n-gramas* para estudiar si una conversación particular a través de canales habituales de comunicación pertenece a un equipo comprometido. Por contra, a pesar de los problemas relacionados con las nuevas generaciones de *applets*[1] que se conectarán a los clientes de IRC con nombres generados automáticamente y que introducirán algunos falsos positivos, su principal limitación es la necesidad de etiquetar estos *nicks* —o por lo menos una parte de ellos en el caso de las técnicas semisupervisadas— para poder afrontar el análisis con posibilidades reales de detectar efectivamente comportamientos maliciosos. De forma similar, Blinkey *et al.* [Rac04] contrasta una lista de direcciones IP de un determinado canal de IRC, con las direcciones IP que ejecutaron en el pasado acciones maliciosas de exploración con el fin de identificar el comportamiento malintencionados y de aislar estos intentos de conexión.

3.4.1.2 Detección de comportamiento cooperativo

Por lo general, las investigaciones que van por este camino se orientan hacia el estudio de las actividades relacionadas con las comunicaciones entre las máquinas infectadas, con el objetivo final de utilizar esta información para

[1] En ciencias de la computación, un *applet* es toda aquella aplicación que ejecuta una tarea específica que es ejecutada, habitualmente como un *plug-in* en el marco de un programa mayor.

optimizar las técnicas de seguimiento. Hay dos tipos de tráfico a detectar: las respuestas a las peticiones de la *botmaster* —con resultados parciales o totales, incluyendo el estado actual de la operación— y el tráfico utilizado para determinar la tarea solicitada por el propio *botmaster*, entendidas como actividades puramente maliciosas aquellas en donde aparece el envío de *spam*, conexiones masivas a ciertos servidores de un ataque DDoS entre otras actuaciones. La idea es determinar qué ataques estadísticos que busquen perfiles de comportamiento que difieran de los realizados por un usuario estándar no infectado pueden ser más apropiados, a fin de seleccionar los nodos de la red con más posibilidades de estar formando parte de una *botnet*. El objetivo final es detectar *botnets* basados en la filosofía *pull* mediante la identificación de un número de intentos de conexión desproporcionado utilizando análisis aleatorios y detectar aquellos equipos responsables de un gran número de conexiones de entrada/salida que puedan corresponder a los nodos desde los que se gestiona la red centralizada.

Chois *et al.* propusieron la detección de *botnets* tratando de capturar comportamientos de grupo anómalos en el tráfico de red [CLK]. Posteriormente, evolucionaron dicho enfoque para la identificación de *botnets* amparándose en criterios similares [CL12] pero implementando un modelo comparable al que ya presentara Villamarín *et al.* empleando como criterio la similitud de tráfico DNS [VSB].

Más adelante, se realizaría el cálculo de la distancia entre el flujo de datos controlados en máquinas sospechosas y un modelo de comunicación IRC predefinido mediante el análisis del tráfico en la capa de transporte [KRH07], con las ventajas ya mencionadas asociadas a la realización de un punto de vista totalmente pasivo —haciendo imperceptible el análisis para el *botmaster*—. Este enfoque ha tenido éxito para la detección de redes de *bots* que utilizan las comunicaciones cifradas con un ratio de falsos positivos inferior al 2%. Sin embargo, aún no hay estudios que confirmen la eficacia de estos sistemas con los controladores de *botnets* utilizan canales de C&C a través de HTTP o P2P.

Otras aproximaciones han buscado desarrollar mecanismos de clasificación temática de las conversaciones de los chats de IRC para la identificación de aquellas que son de origen humano. La idea consistía en ser capaces de detectar comunicaciones propias de un flujo automatizado habitualmente relacionadas con el uso de consultas periódicas o la identificación de una desproporcionada cantidad de información manejada. Dewes *et al.* [DWF03] establecían que en una conversación de chat IRC típica el lector normalmente recibe hasta diez veces más datos de los que envía. Así,

mediante la observación y el perfilado de los usuarios que interactúan entre sí, las conversaciones anormales podrían ser detectadas permitiendo un estudio adicional basado únicamente en el volumen de información que fluye entre dos nodos sospechosos.

Dubenforder *et al.* optaron por realizar un análisis del flujo de información a través de determinados puertos IRC [RDP04] dependiendo de las características particulares de la aplicación de la *botnet*. Una de las principales conclusiones extraídas se centraba en la utilización sistemática de determinados puertos. Así ocurría, por ejemplo, con el puerto número 6667, asociado al control de los infectados por *botnets* SDBot y en el que casi un 35 % del tráfico encaminado a través de él era vinculado a tráfico utilizado por la *botnet* para llevar a cabo su actividades [Bor07].

Por su parte, Gu *et al.* [GZL08] hicieron un interesante análisis de la forma de detectar el tráfico de red anómalo basado en HTTP y/o IRC tratando de encontrar patrones de similitudes espacio-temporales en las comunicaciones entre los miembros de una *botnet* utilizando la naturaleza programática de los propios *bots*. Los mensajes que se identifican son los de solicitud de información del sistema, rastreando la red y tratando de identificar máquinas que efectúan la comunicación de similares características, de modo que si se detectara un número determinado de máquinas que se prestaran a la ejecución de comandos de idénticas características, se podría decir que una *botnet* nueva ha sido detectada. La característica más interesante de esta aproximación reside en el hecho de no necesitar evadir el uso de determinadas técnicas criptográficas o de codificación para identificar el contenido de una conversación dada. El método se basa en la comparación de frecuencias y lagunas temporales mediante la complejidad Kolgomorov como propuso Wehner en 2007 [Weh07] para la detección de gusanos.

Esta filosofía cambia en función de la metodología *push*/*pull* utilizada por la *botnet* que vayan a ser examinadas. Existen algunos casos de estudio sobre HTTP e IRC —como BotSniffer, implementado como un *plug-in* para Airsnort [GZL08]—, así como listados de los comandos de teclado que implementan algunas variantes de Agobot, SpyBot o SDbot. Sin embargo, aunque estos comandos fueran codificados para ocultar el verdadero contenido de los mensajes, la actividad de los nodos infectados todavía podía ser detectada debido a la necesidad de estos de responder a las peticiones formuladas por el *botmaster*. Estas comunicaciones se realizan siempre en un manera mucho más coherente y uniforme a las comunicaciones humanas tradicionales. Es por eso que las *botnets*, teniendo en cuenta las técnicas actuales para generar azar, tratan de evitar, o al menos confundir, estos

sistemas mediante la modificación de los intervalos de tiempo entre las interacciones.

3.4.1.3 **Detección de comportamiento ofensivo**

Este enfoque, más orientado a la detección de actividades maliciosas por sí mismas, supone que las *botnets* envían cantidades masivas de información (correo no deseado, los intentos de conexión fantasma, etc.) en períodos relativamente cortos de tiempo. Así, Xie *et al.* [XYA+08] utilizan el volumen de datos enviados durante estos períodos, junto con la información obtenida de los servidores de correo electrónico no deseado, para el seguimiento de estos contenidos antes de que causen un daño real —por ejemplo, el envío masivo de *spam*— y la prevención de las consecuencias mediante la identificación de los nodos que son responsables para llevar a cabo esta tarea.

3.4.1.4 **Detección del usuario final**

Por otro lado, Ormerod *et al.* [OWD+] propusieron la desactivación desde un punto de vista menos técnico. El equipo de Ormerod defendió la eficacia de intentar perseguir y desanimar a aquellos usuarios finales que tratan de monetizar los credenciales robados utilizando estas herramientas, combatiendo las vulnerabilidades y monitorizando los rastros que dejan los *kits* especializados en robos. Muy en la misma línea, Riccardi *et al.* [ROL+] desarrollaron sus esfuerzos en mitigar el impacto de las *botnets* financieras identificando y facilitando una puntuación *maliciosa* a todos aquellas comunicaciones provenientes de nodos sospechosos con el fin de tener una medida de evaluación de la reputación de cada conexión. Para lograr este objetivo, sin embargo, los autores necesitan de un proceso de compartición de información mucho más fluido entre todos los individuos que toman parte en el proceso de detección y seguimiento de los cibercriminales aunando esfuerzos entre las Fuerzas y Cuerpos de Seguridad y los analistas de seguridad informática e investigadores.

3.4.2 **Técnicas de desactivación**

Las obligaciones del analista de seguridad, sin embargo, no terminan en el momento de la detección de la existencia de una *botnet*. Ser capaz de poner en cuarentena las conexiones maliciosas es una alternativa pasiva y poco

intrusiva, pero existen diferentes técnicas que amplían este enfoque con actividades mucho más agresivas.

- Cierre de la comunicación de C&C para evitar que se propaguen aún más comandos por medio del canal especificado y, por lo tanto, limitar así su capacidad operativa primero y sus formas de propagación después.

- Cierre físico de la red poniéndola en cuarentena —y en una etapa temprana de su desarrollo— para completar el proceso de desinfección de los nodos comprometidos uno por uno con las técnicas antivirales tradicionales en cada *host*.

- Utilización de técnicas de ataque de envenenamiento de índices. Por lo general, estas técnicas han venido siendo utilizadas por las empresas para evitar la redistribución de contenido de *software*, vídeo y otros elementos protegidos por derechos de autor mediante la inclusión de registros falsos [LNR06, LH06]. Aplicando esto a la esfera de las *botnets*, implicaría la detección de los archivos que serán buscados por las máquinas infectadas —con *honeypots*, por ejemplo— a fin de distribuir después los archivos con los mismos nombres clave pero con ninguna información relevante en ellos, por lo que es más difícil para los nodos de comunicarse eficientemente con sus otros miembros infectados [GSN+07]. Sin embargo, también es cierto que existe literatura capaz de esquivar este problema [SKK08] que podría aislar a los ataques de *botnets*, evitando la propagación de la identidad y el emplazamiento de sus nodos mediante el uso de un modelo estándar de clave pública. Este hecho ha alertado sin duda a la comunidad científica acerca de los problemas que implicaría un aumento en la complejidad de los canales de comunicación tradicionales en la próxima generación de redes de *bots*.

- Uso de técnicas de ataque Sybil o *Sybil attack*[1] en inglés. El objetivo es infiltrarse a través de un nodo en la lista de *bots* infectados con el fin último de poder sabotear las comunicaciones desde el interior mediante la modificación —y posterior publicación— de *hashes*, archivos o comandos corruptos [VAJ07]. La principal ventaja es que el cambio de algunas pocas piezas es ya suficiente para bloquear la difusión de

[1] El ataque de Sybil en la seguridad informática es un ataque en el que un sistema de reputación se subvierte mediante la creación de identidades en las redes peer-to-peer.

los comandos de la red como archivos totalmente diferentes a los que se buscaba.

- *Blacklisting* de comandos y/o miembros. Detectados en las primeras etapas de las comunicaciones sospechosas, se podrían aplicar métodos que los identificaran y mitigaran la velocidad de expansión de la red.

- El secuestro de la red mediante el envío de comandos para desactivarla. El inconveniente es que muchas *botnets* utilizan actualmente los sistemas de cifrado asimétricos [SPW02] para evitar la detección de uno de los nodos que pudieran comprometer la seguridad del resto de la red.

3.5 Sumario

En este capítulo se han recogido los principales esfuerzos llevados a cabo por la comunidad científica para la detección de *botnets*. Así, se han revisado algunos de los conceptos que han permitido la evolución natural del *malware* tradicional hacia enfoques mucho más beneficiosos desde el punto de vista económico mucho más allá de la filosofía eminentemente romántica de las aplicaciones maliciosas de los ochenta y noventa. En este sentido, las siempre crecientes cifras de ciberdelitos y los indicadores empleados para determinar la evolución de estas amenazas son claros: la profesionalización de esta industria es un hecho consumado que abona el terreno de un ecosistema cada vez más propicio para la aparición de muestras con unas capacidades, no solamente más dañinas, sino también con mayor capacidad para llevar a cabo una explotación económica de las infecciones, incluso por parte de atacantes sin demasiados conocimientos.

Ante un contexto como este, se han analizado las características que hacen de la construcción y despliegue de *botnets* un concepto interesante desde el punto de vista del atacante. En este capítulo se ha analizado la evolución desde sus orígenes, sus arquitecturas típicas y los canales de control más utilizados de cara a identificar las nuevas líneas de trabajo que ayuden a la comunidad científica al control de una realidad como esta.

Para ello, se ha procedido a enumerar algunas de las principales líneas de defensa frente al fenómeno *botnet*. Empezando por las principales medidas de detección de *botnets* empleando toda clase de reconocimiento de patrones en el envío y recepción de comandos y terminando con aquellas aproximaciones más directamente relacionados con la desactivación y

puesta en cuarentena de los servicios maliciosos, se ha querido hacer una revisión exhaustiva del estado del arte que permita contextualizar las líneas de acción que se han tomado en el desarrollo de la presente tesis doctoral.

«Algunas de las hazañas más grandes de la humanidad han sido obra de personas que no eran lo bastante listas para comprender que eran imposibles».

Doug Larson, columnista estadounidense (1926 – ...).

Capítulo

4

Modelado del tráfico de red

Una de las partes más importantes de todo proceso de investigación es el desarrollo de un modelo que permita una representación fidedigna del objeto de estudio. En la tesis doctoral que nos ocupa, el objeto de estudio no es otro que las comunicaciones interceptadas entre los servidores y redes a los que trata de acceder un usuario y el propio terminal desde el que lo hace. Conocer el funcionamiento básico de estas redes y, por ende, cómo se produce el intercambio de información entre iguales en una red es fundamental para entender el alcance real del modelo propuesto. Así, en este capítulo se propone que los atributos empleados para representar dichas conexiones sean obtenidos directamente de los paquetes que las componen, extrayendo las características concretas que cada uno de ellos almacenen y construyendo representaciones a partir de estas.

Para satisfacer estos objetivos y en base a estas premisas se define en este capítulo el proceso de obtención del modelo propuesto. Así, el resto del capítulo está estructurado como sigue. La sección 1 define el funcionamiento general de las redes de comunicaciones como paso previo a definir un modelo funcional. La sección 2 establece la abstracción del modelo de representación a utilizar en esta tesis doctoral. La sección 3 define el proceso de generación de las diferentes representaciones. Por último, la sección 4 sintetiza las conclusiones relativas al proceso de modelado.

4.1 Flujos de información y tráfico de red

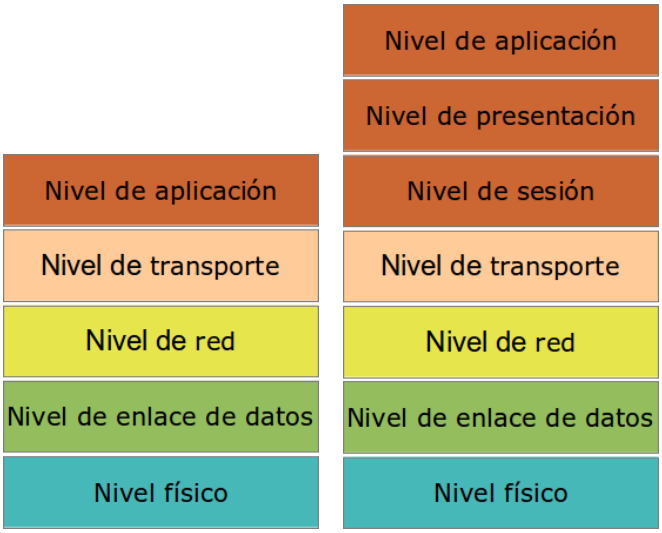
El funcionamiento de las redes de ordenadores se basa en una idea conceptualmente sencilla: trocear la información en unidades de pequeño tamaño que viajan de forma independiente hasta su destino en donde serán ensamblados de nuevo para reconstruir el contenido original y permitir su procesamiento. Sin embargo, la evolución disgregada de las tecnologías de comunicación a mediados de los ochenta ponía de manifiesto un grave problema: multitud de nuevas empresas se encontraban desarrollando tecnologías con diferentes formas de entender las comunicaciones que además eran implementadas de forma distintas y, muchas veces, incompatibles [Zim80, Sta04].

4.1.1 El proceso de estandarización

Es en ese contexto en el que se detecta la necesidad de establecer una serie de mecanismos que estandaricen la forma en que las máquinas se comunican unas con otras. Surge así la pila de protocolos TCP-IP —véase figura 4.1(a)— que constaba de cinco niveles o capas cada una de las cuales con un objetivo concreto: el nivel físico, el nivel de enlace de datos, el nivel de red, el nivel de transporte y el nivel de aplicación.

Más adelante, la Organización Internacional para la Estandarización (ISO) fijaría un nuevo marco de referencia para definir arquitecturas que permitan la conexión y la interactividad entre sistemas de comunicación de diferentes fabricantes: el modelo de interconexión de sistemas abiertos o ISO/IEC 7498-1 [fSEC+94] también conocido como la *pila OSI* —del inglés *Open System Interconnection*, véase figura 4.1(b)— y que añadía dos capas nuevas a las de la pila TCP-IP para ampliar la capa de aplicación: las de sesión y presentación.

De cualquier manera, el funcionamiento básico de los protocolos es muy similar. En el emisor, una aplicación que quiere comunicarse con otra a través de la red encapsula la información y se la sirve a la capa inmediatamente *inferior* para que vaya añadiendo capas que le permitan identificar la dirección física de los equipos de origen y destino, el puerto, etc. Así, después de que los datos se transfieran por la red y sean recibidos por el destinatario, el proceso se produce en sentido inverso: cada una de las capas va extrayendo la información que sabe manejar sirviendo a la capa inmediatamente *superior* el resto del campo de datos. Este proceso se desarrolla así cada vez que una de las dos entidades pretende ponerse en comunicación



(a) Pila TCP-IP. (b) Pila OSI.

Figura 4.1: Comparación de los niveles propuestos por la pila OSI y la pila de protocolos TCP/IP.

con la otra.

4.1.2 Abstracción del modelo

Conociendo el funcionamiento de las tecnologías que facilitan la conexión entre equipos, es mucho más sencillo hacerse a la idea del modelo propuesto en esta tesis doctoral: la utilización de las características individuales de estos para determinar la procedencia común del tráfico de una *botnet* atendiendo a la evolución de sus atributos atómicos e independientemente —y esto es lo relevante— de cuál sea el origen de la conexión. Para ello será necesario que los mensajes enviados y reenviados desde un mismo nodo sean pertinentemente anonimizados y se eliminen los elementos que identifican a las conexiones a la hora de construir las representaciones.

4.2 Obtención de representaciones

Los paquetes de conexiones maliciosas y conexiones etiquetadas como relativas a tráfico benigno serán agrupados de cara a su posterior secuenciación. En este punto, si atendemos a la longitud de las secuencias podemos dividir el experimento en dos partes:

- **Secuencias de un paquete**. El objetivo es analizar si las características propias de los paquetes enviados como parte del entramado de comunicaciones de control en cada una de las *botnets* analizadas, es suficiente para realizar la detección o, para al menos, asignar un valor estimativo de confiabilidad que permita clasificar la información recibida en esa conexión.

Sin embargo, se es consciente de que el *payload* —o carga maliciosa— que puede traer consigo un paquete en sí no es muy significativa. Por ello, se propone emplear métricas que tengan más en consideración la evolución de las características de los paquetes en un período de tiempo determinado.

- **Secuencias compuestas por una cantidad de paquetes superior a uno**. Aunque otros enfoques han venido aplicando transformaciones sobre la totalidad de los datos relativos a una misma conexión [BBR+], en esta disertación se va a realizar una atomización más precisa de las características de una conexión, agrupando los paquetes recibidos en cadenas de paquetes de diferentes longitudes.

Por simplificar los conceptos, se podría considerar la observación de paquetes individuales como la obtención de una instantánea del estado de la conexión en un tiempo t_0, siendo mucho más representativo el seguimiento dinámico de las variables en el tiempo t que pasa desde que se recibe un paquete hasta que se recibe el enésimo paquete de esa misma conexión. De cualquier manera, en las páginas que siguen se profundizará más en detalle en cada uno de estos enfoques y en la forma en que son generadas las diferentes representaciones.

4.2.1 Construcción de representaciones de un paquete

Para secuencias de $n = 1$ paquete la representación de la conexión estará formada por los atributos correspondientes a cada uno de los paquetes de la misma, almacenando los valores númericos para su posterior evaluación con herramientas de inteligencia artificial. Con respecto a la no inclusión de valores relativos a la separación temporal de los paquetes de una misma conexión en las representaciones de un único paquete, en experimentos anteriores ya publicados [BGdlPUP+b] en los que sí que se incluía dicho valor, el autor ha podido comprobar cómo el peso del tiempo transcurrido desde que se recibe un paquete hasta que se recibe el siguiente perteneciente a esa misma conexión no es relevante para la clasificación. Podríamos llamar a

este tiempo *tiempo inter-representación* o *tiempo exterior* t_e en contraposición con el tiempo *tiempo intra-representación* o *tiempo interior* t_i que se detallará más adelante. La obtención de unos valores muy bajos al aplicar algoritmos como *information gain* [BGdlPUP+b], ponía de manifiesto una realidad que se hacía patente para secuencias de $n = 1$ paquetes. En la figura 4.2 se define el proceso que se sigue para la obtención de las características de un paquete.

entrada: Un conjunto C de c conexiones junto con los atributos que representan a los p paquetes observados para esa conexión en una muestra de tráfico
foreach $i \in P$ **do**
    // Se filtrarán atributos relativos a protocolos no relevantes que puedan encontrarse solamente en tráfico benigno
    $att \leftarrow filtrar(p.getAtributos())$;
    // Se comprueba si la conexión actual ya ha sido observada
    if $att[con] \notin conexiones$ **then**
        // Se crea la nueva conexión
        $conexiones.append(newConexion(con))$
    end
    // Se añaden los atributos como una nueva representación de *con*
    $conexiones[con].append(att)$;
end
salida : Un array bidimensional que contiene una lista de los atributos de los paquetes disponibles para cada conexión

Figura 4.2: Obtención de las características de un paquete.

En primer lugar se filtrarán aquellos atributos existentes en el conjunto de datos que pertenezcan a protocolos que no sean directamente utilizados por los protocolos de la *botnet* para evitar, en primer lugar, posibles desviaciones que pudieran acontecer y, en segundo lugar, para permitir una mayor agilidad en los cálculos centrándolos en aquellas características interesantes. La aparición de estos atributos recaerá en el conjunto de datos benignos que contengan referencias a protocolos como UDP que no utilizan el resto de aplicaciones maliciosas.

A continuación, se comprobará si la conexión que se está analizando —identificando esta por dirección IP y puerto— ha sido ya observada. En caso negativo, se creará una nueva representación para, seguidamente, añadir el listado de atributos del nuevo paquete como una nueva representación de un paquete de esta conexión. El atributo de la clase de la representación es obtenido en función del punto al que se conecta el nodo, ya que en la experimentación conocemos qué direcciones IP corresponden a las conexiones maliciosas, algo que no ocurre en la vida real.

4.2.2 Construcción de representaciones de una cantidad indefinida de paquetes

En contraposición con lo ya expuesto en el apartado anterior y para ampliar el enfoque ahí descrito, para secuencias de $n = i$ paquetes ($\forall i \in \mathbb{N}$ tal que $1 <= i <= 12$) se contará con más características: los atributos del primer paquete, los del segundo, etc., y así sucesivamente hasta alcanzar el enésimo paquete que componga dicha representación. Adicionalmente, se han añadido como parte de la representación una serie de variables complementarias que almacenan diferentes medidas de tendencia central y medidas de dispersión relativas al intervalo de *tiempo intra-representación* —o también *interior*— que pasa entre la recepción de un paquete y del siguiente de la secuencia de paquetes analizada.

Básicamente, la metodología se puede definir como un proceso de concatenación de las características individuales de cada paquete ordenado cronológicamente atendiendo al momento de su observación. De esta manera, en la figura 4.3, se muestra el pseudo-código relativo a la construcción de secuencias de hasta n paquetes que dan lugar a las representaciones combinadas teniendo como punto de partida la información utilizada en el proceso descrito en la figura 4.2.

Dado un conjunto de datos C formado por c conexiones de las que se disponen una serie de paquetes, el proceso de generación de representaciones de hasta $n = 12$ recorrerá cada una de las c conexiones generando representaciones a partir de la concatenación de los atributos de un número n paquetes sucesivos. Adicionalmente, se incluirán variables que tratan de ponderar la separación espacio-temporal entre los paquetes que componen dicha representación. Esta separación se puede calcular ya que se cuenta con el *timestamp* del momento en que fue observado cada paquete. Como se puede ver, además del tipo de la conexión, entre las variables que se in-

entrada: Un array bidimensional A que contiene una lista de los atributos de los paquetes disponibles para cada conexión c del conjunto de datos C

```
muestras ← [];
foreach c ∈ C do
    nPaq ← len(c);
    // El proceso se realizará tantas veces como longitud de las representaciones se desee. En este caso: hasta n = 12
    foreach i ∈ range(2, n) do
        // En una conexión c se podrán obtener nPaq − i + 1 secuencias de longitud i
        foreach j ∈ c do
            rep ← [];
            tiempos ← [];
            // Se seleccionan los i paquetes que formarán la representación
            foreach k ∈ i do
                rep.append(j + k);
                tiempos.append(j + k);
            end
        end
        // Se calculan los tiempos interiores t_i
        attM ← calculoTi(tiempos);
        rep.append(attM);
        // Se añade el tipo de la muestra como último atributo de la muestra
        rep.append(attTipo);
        // Se añade la muestra completa al listado de muestras del conjunto de datos
        muestras.append(rep);
    end
end
```

salida : Un array bidimensional que contiene una lista de los atributos de los paquetes disponibles para cada conexión

Figura 4.3: Obtención de las características de un paquete.

cluirán al final de cada secuencia de paquetes se encuentran las medidas de tendencia central y medidas de dispersión que se definen a continuación.

4.2.2.1 **Medidas de tendencia central**

Las medidas de tendencia central son una de las herramientas de las que se dispone para resumir la información relativa a un conjunto de datos en único escalar. Este valor representa el centro de la distribución de datos y se conoce como parámetro de tendencia central o medida de tendencia central. Aunque la media aritmética es probablemente uno de los parámetros estadísticos más extendidos, existen también otras medidas que deben ser analizadas. En las páginas que siguen se recogen las características que definen cada una de las medidas de tendencia central consideradas.

- **Media aritmética**. En matemáticas y estadística la media aritmética —a menudo referidas como media o promedio— expresa la tendencia central de una colección de números tomada como la suma de todos ellos dividida por el tamaño de dicha colección. Matemáticamente, la media aritmética $\bar{x}$ de una colección de n elementos $\{a_1, a_2, \cdots a_n\} \in \mathbb{R}$ se puede expresar como en la ecuación (4.1):

$$\bar{x} = \frac{\sum_{i=1}^{n} a_i}{n} = \frac{(a_1 + a_2 + \cdots + a_n)}{n} \tag{4.1}$$

  A pesar de que la media aritmética es utilizada habitualmente como medida de tendencia central, no es un estadístico robusto dado que puede ser ampliamente influenciado por la aparición de valores atípicos por lo que en determinados casos, como para distribuciones asimétricas, se prefieren otro tipo de descriptores como la mediana.

- **Media cuadrática**. La media cuadrática —en inglés *Root Mean Square* (RMS)— es una medida estadística de tendencia central de la magnitud de una cantidad variable. Se puede calcular tanto para una función de variación continua, como ya es usada en ingeniería eléctrica para calcular la potencia disipada por una resistencia por la que circula una intensidad $I(t)$ que varía en el tiempo, o a partir de una serie de valores discretos, como es el caso. De forma general, la media cuadrática x_{RMS} de una colección de n elementos $\{a_1, a_2, \cdots a_n\} \in \mathbb{R}$ se puede expresar como en la ecuación (4.2):

$$x_{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i 2} = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n}} \tag{4.2}$$

Es utilizada en aquellos contextos en los que se pueda estar interesado en obtener promedios que no recojan los efectos de medidas con signo positivo o negativo. En nuestro caso, se formulará la media cuadrática x_{RMS} para aquellas secuencia de n elementos $\{a_1, a_2, \cdots a_n\} \in \mathbb{R}$ que representarán la variación de tiempo —positiva y negativa— con respecto al tiempo a_1 esperado desde que se recibió el primer hasta que se recibió el segundo paquete.

- **Media geométrica**. Es habitualmente utilizada como medida de tendencia central de valores relativos. Matemáticamente, la media geométrica $\bar{x}$ de una colección de n elementos $\{a_1, a_2, \cdots a_n\} \in \mathbb{R}$ se puede expresar como en la ecuación (4.3):

$$\bar{x} = \sqrt[n]{\prod_{i=1}^{n} x_i} = \sqrt[n]{x_1 * x_2 * \cdots * x_n} \tag{4.3}$$

- **Media armónica**. La media armónica es una medida de tendencia central que resulta poco influida por la existencia de determinados valores mucho más grandes que resto, siendo en cambio sensible a valores mucho más pequeños. Matemáticamente, la media armónica H de una colección de n elementos $\{a_1, a_2, \cdots a_n\} \in \mathbb{R}$ se puede expresar como en la ecuación (4.4):

$$H = \frac{n}{\sum_{i=1}^{n} \frac{1}{a_i}} = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_n}} \tag{4.4}$$

  Se suele utilizar para promediar velocidades, tiempos, rendimientos, etc. en los que la media aritmética no es suficientemente representativa.

- **Mediana**. Matemáticamente, la mediana M_e de una colección de un número impar de elementos n ($\forall n = 2k + 1$ tal que $k \in \mathbb{R}$) ordenados en orden creciente $\{a_1, a_2, \cdots a_n\} \in \mathbb{R}$ será el valor que ocupe la posición central y que se determina en la ecuación (4.5), mientras que la mediana de una colección de un número par de elementos n ($\forall n = 2k$ tal que $k \in \mathbb{R}$) será la media aritmética de los dos valores centrales, lo que se puede expresar como en la ecuación (4.6):

$$M_e = a_{\frac{n+1}{2}} \tag{4.5}$$

$$M_e = \frac{a_{\frac{n}{2}} + a_{\frac{n}{2}+1}}{2} \tag{4.6}$$

4.2.2.2 **Medidas de dispersión**

Otra herramienta que nos sirve para determinar las características de los conjuntos de datos son las medidas de dispersión o variabilidad. Estas medidas indican si los diferentes valores de un conjunto de datos están más o menos alejados del valor central, siendo mayores los coeficientes para valores muy dispersos y menores para valores que varían menos entre ellos.

- **Desviación típica**. Matemáticamente, la desviación típica σ de una colección de n elementos $\{a_1, a_2, \cdots a_n\} \in \mathbb{R}$ se puede expresar como en la ecuación (4.7):

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{4.7}$$

4.2.2.3 **Generalización de la representación**

A la hora de establecer comparaciones no es aconsejable manejar magnitudes absolutas, ya que las unidades no tienen por qué ser siempre comparables [CLT13]. Hay que tener en cuenta la necesidad de afrontar convenientemente la problemática del emplazamiento de los canales de control. Se hace necesario ocuparse de los posibles sesgos en los que se podría incurrir si se considera el tiempo que transcurre entre la recepción de un paquete y del siguiente para evitar que el método se convierta en una herramienta de detección de servidores más o menos alejados. En otras palabras, los paquetes que se reciban desde una *botnet* determinada van a estar separados por un número de saltos o *hops* entre los nodos de emisión y recepción concretos. Si se mantienen los valores absolutos de estas distancias, se podría terminar obteniendo un modelo que únicamente detecte conexiones en servidores situados a un número de saltos concreto de la víctima, lo cual sería un resultado pobre y no interesa. De esta manera, se plantea la posibilidad de calcular cuál será la variación media del tiempo de recepción con respecto al primer paquete aplicando el algoritmo de la figura 4.4.

En esta aproximación se pretende dar una respuesta que considere la separación entre el primer y el segundo paquete como una unidad, y que pondere el resto de variaciones en base a esa misma unidad. Por tanto, en el caso que nos ocupa, la medida de tendencia central seleccionada es la media geométrica dado que es la que pondera de una forma más eficaz y ajustada variaciones relativas. Así ocurre con la diferencia de tiempo normalizada existente entre los paquetes de una misma secuencia con respecto al valor de separación temporal observado entre los dos primeros de esa misma

entrada: Un conjunto T de n separaciones temporales en
milisegundos.
`// tNormalizado es una lista de los atributos normalizados`
$tNormalizado \leftarrow []$;
foreach $i \in T$ **do**
  `// Se saca factor común del primer elemento`
  $factComn \leftarrow i/T[0]$;
  $tNormalizado.append(factComn)$;
end
salida : Obtención de una lista de intervalos temporales
normalizados.

Figura 4.4: Obtención de los intervalos temporales normalizados.

secuencia.

En vista de los atributos anteriormente seleccionados, se puede decir que, para representaciones de secuencias de n paquetes definidos cada uno de ellos individualmente por a atributos y un número t de atributos espacio-temporales, el número total de atributos A que compondrán la representación vendrá dado por la ecuación (4.8):

$$A = a * n + t + 1 \tag{4.8}$$

en donde 1 es el atributo que define la clase de la representación. En nuestro caso: *bot* o legítimo.

4.2.3 Generación de conjuntos de datos procesables

La herramienta de minería de datos utilizada, WEKA[1], permite la introducción de los conjuntos de entrenamiento en formatos sencillos como *.csv* —del inglés *Comma Separated Values*— o incluso consultas sobre bases de datos. Sin embargo, dada la sencillez de la estructura de los fichero con formato *.arff* se ha optado por estos últimos. Su contenido se puede dividir en dos apartados: la cabecera y los datos.

- **Cabecera**. La cabecera contiene los datos relativos al formato de los atributos —usando la etiqueta *@attribute* seguida del nombre del atributo y el formato— que componen la representación de cada una de

[1] WEKA es una herramienta de aprendizaje automático desarrollada por investigadores de la Universidad de Waikato la cual se ha sido ampliamente utilizada por la comunidad científica con diferentes propósitos.

```
1   @relation Exp113_War_i_1
2   @attribute http.content_length__0 numeric
3   @attribute http.content_length_header__0 numeric
4   @attribute http.request__0 numeric
5   @attribute http.response__0 numeric
6   @attribute http.response.code__0 numeric
7   @attribute tcp.flags.res__0 numeric
8   @attribute tcp.flags.reset__0 numeric
9   @attribute tcp.flags.syn__0 numeric
10  .
11  .
12  .
13  @attribute tcp.segment__0 numeric
14  @attribute tcp.segment.count__0 numeric
15  @attribute class {botWarbot,legitime}
16  @data
```

Figura 4.5: Muestra de un fichero *.arff* con representaciones de un paquete de longitud generado a partir de una muestra de tráfico de control de Warbot.

las muestras, así como el nombre de la relación —definido empleando la etiqueta *@relation*—. Como los datos empleados en esta tesis doctoral son todos de tipo numérico, a excepción de la clase del paquete o secuencia de paquetes, la clase de esos atributos será *number* en todo caso. Nótese que se ha incluido un número que indica el paquete al que pertenece cada uno de los atributos.

- **Datos**. Introducidos por la etiqueta *@data*, contienen el conjunto de representaciones de secuencias de paquetes listas para clasificar. Cada uno de los valores, separados por comas, vendrá precedido del índice que representa al atributo que cuantifican.

En la figura 4.5 se muestra un ejemplo de uno de los ficheros utilizados para un experimento de clasificación de paquetes procedentes de la *botnet* WarBot para secuencias de $n = 1$ paquetes. Por este motivo todos los atributos del ejemplo contienen un sufijo en su identificador formado por *<atributo>__0* que indica que son características extraídas del primer —y en este caso único— paquete de la representación.

4.3 Sumario

En este capítulo se han introducido dos métodos para la representación de conexiones. Aunque ambos se basan en la extracción de características de los paquetes que se intercambian entre el servidor de control y los nodos infectados, se puede considerar al segundo, que es la evolución del primero, como un método de representación conceptualmente diferente.

Así, el primero de ellos se basa en la extracción de características de los paquetes individuales que se envían y reciben en una misma conexión. Los atributos empleados para dicha representación han sido pertinentemente filtrados para evitar sesgos y han sido incluidos en las representaciones compuestas por los atributos de cada paquete. El segundo se puede definir como la construcción de representaciones en base a los atributos independientes de cada paquete y a la separación espacio-temporal relativa existente entre los paquetes que componen una misma representación mediante la monitorización de los tiempos medios de interconexión entre terminales.

Las principales fortalezas de un método de estas características son dos. En primer lugar que es un método transparente al contenido de la conexión —cada vez con más frecuencia cifrado— ya que trata de extraer patrones de comportamiento comunicativo y no patrones relativos al contenido en sí. En segundo lugar que, dada la naturaleza de los protocolos de comunicación actuales, la carga maliciosa o *payload* de un paquete independiente es limitada lo que permite un mayor grado de tolerancia con respecto a los falsos positivos que en otros áreas de la detección de comportamientos maliciosos como la identificación de *malware*, en donde un fichero individual es una amenaza potencialmente superior.

Por último, se ha incluido también la forma en que esta información va a ser transferida a la herramienta Weka. El formato seleccionado es el formato propio de esta: ficheros *.arff* con una estructura definida para cada experimento en función del número de características que se incluyan en cada representación.

«Se necesita poco para hacer las cosas bien, pero menos aún para hacerlas mal».

Paul Bocuse, cocinero francés (1926–...).

Capítulo 5

Metodología para la obtención de muestras

Como ya se ha visto, toda experimentación del área de las Ciencias de la Computación requiere la identificación de un problema susceptible de ser modelado para la realización posterior de los cálculos pertinentes. Esta tesis doctoral no podía ser una excepción: para enfrentarnos a la problemática ya definida anteriormente de la proliferación de las *botnets* como herramientas ciberdelictivas, se ha propuesto una metodología de trabajo capaz de modelizar el tráfico de red con el objetivo último de identificar patrones en el flujo mismo de la red. En este punto, contar con un método robusto y eficiente para la obtención de muestras de tráfico se torna fundamental de cara a poder realizar una correcta interpretación de los resultados.

Para ello, este capítulo se ha estructurado como sigue. La sección 1 recoge el estado de la práctica actual, repasando algunas de las técnicas empleadas en el pasado para tratar de dar respuesta a este problema. La sección 2 detalla las características de las muestras de *malware* seleccionadas para esta investigación. La sección 3 desglosa la metodología ejecutada para la obtención de muestras de tráfico útiles para la investigación. La sección 4 define el procedimiento que se hace necesario llevar a cabo para anonimizar convenientemente las muestras de tráfico y garantizar que el proceso posterior. La sección 5 repasa algunas de las limitaciones que presenta este enfoque de modelado de tráfico.

5.1 Metodología de obtención de tráfico

La metodología para la obtención de muestras se centrará en recabar información sobre dos tipos de tráfico claramente diferenciados: por un lado, paquetes pertenecientes a conexiones legítimas llevadas a cabo por usuarios autorizados y, por otro, paquetes pertenecientes a equipos infectados con aplicaciones maliciosas que se intentan comunicar con el servidor desde el que el *botmaster* emitirá las órdenes y comandos a ejecutar. Seguidamente a la infección, el investigador procede a realizar diferentes solicitudes de comandos a ejecutar entre las que se encontrarán —en función de las características propias de cada muestra— algunas de las siguientes: ejecución remota de comandos, transferencia de ficheros, realización de ataques, *keylogging*, etc. Todo ello con el objetivo último de hacer la extracción de muestras lo más transparentemente posible y evitar la inclusión de cualquier información que pudiera desvirtuar los experimentos e introducir sesgos indeseados.

5.1.1 Despliegue de los canales de *Command & Control*

En este sentido, la máquina de control tendrá instalada el servidor PHP con MySQL para almacenar las tablas de la base de datos que contiene las referencias y los datos relativos a las máquinas infectadas así como los ataques realizados y toda clase de información de interés para el operador. El escenario ideal contempla la instalación de la interfaz de *Command & Control* en un servidor externo con el fin de simular lo mejor posible un despliegue real y lograr evitar que parámetros como el número de saltos de los paquetes o el reenvío de aquellos perdidos, pudieran introducir sesgos en la clasificación que aparecerían si la instalación se hubiera llevado a cabo en un entorno de laboratorio. En todo caso, los servidores de *Command & Control* serán alojados en máquinas Ubuntu 12.04 en las que se habrán dispuesto previamente todos los requisitos para su funcionamiento, como servidores PHP, base de datos MySql o permisos de lectura y escritura para permitir la transferencia de ficheros.

5.1.2 Preparación de los sujetos

Las características propias de cada una de las muestras de *malware* fuerzan al investigador a configurar los equipos de una forma determinada para que estos puedan asimilar la infección. En este apartado se recogerán aquellos

aspectos más técnicos a tener en cuenta para poder replicar el funcionamiento real de una *botnet*. La filosofía de trabajo propuesta en el artículo está basada en un IDS[1] basado en red. Es decir, el punto de escucha de tráfico esta situado a la salida los propios nodos infectados de la red. Conviene destacar en este punto las cada vez mayores capacidades del *malware* para detectar su ejecución en entornos virtuales. En el caso de que esto ocurra, podrían ser insuficientes estas medidas, ya que podría ocurrir que el tráfico escuchado no contuviera conexiones maliciosas que comprometieran la existencia de la red.

Las máquinas infectadas estarán corriendo sobre sistemas Windows XP con .Net Framework 3.5, lo que es condición indispensable para este experimento ya que la versión de Flu empleada ha sido compilada en C# bajo dicho *framework*. El código fuente de dicho *malware* ha sido parcialmente recodificado para infectar máquinas Windows al uso de una forma controlada incluyendo algunas nuevas funcionalidades no implementadas en la versión original, como el envío remoto de *spam*, la captura de imágenes desde la *webcam* del usuario y la ejecución de ataques de denegación de servicio.

En este caso, los equipos infectados serán máquinas con Windows XP (x32) que contarán con las instalaciones de los *framework* que requieran cada una de ellas por separado. Se trata de copias en las que la cantidad de servicios instalada se limitará a una configuración básica, sin programas adicionales que pudieran distorsionar el funcionamiento normal de la máquina infectada. Para monitorizar su funcionalidad, la actividad del ejecutable instalado en las víctimas correrá entonces en segundo plano aprovechando las capacidades de ocultación implantadas por sus desarrolladores, realizando las pertinentes conexiones a la dirección IP asignada en su creación en busca del fichero *.xml*, en el que, tras ejecutar los métodos de parseo, encontrará la lista actual de comandos a ejecutar. Asimismo, las características de Prablinha, que necesitan de un comportamiento similar, lo convierten en un nuevo ejemplo de este funcionamiento.

Herramientas de *sniffing*

El *traffic sniffing* o la monitorización del tráfico se realizará empleando una de las herramientas más conocidas: Wireshark. Wireshark es una herramienta de análisis de paquetes multiplataforma y distribuida con licencia GPL anteriormente conocida como Ethereal. Es habitualmente utilizada

[1] Del inglés, *Intrusion Detection System*.

para la monitorización de problemas relacionados con la conectividad de la red, el análisis de tráfico y el desarrollo de software o nuevos protocolos de comunicación.

5.2 Generación de tráfico benigno

La obtención de tráfico legítimo o benigno se ha realizado teniendo presentes estudios de hábitos de uso de Internet recientes, como el Estudio sobre hábitos en Internet e identidad digital de las marcas llevado a cabo en colaboración con el ICEMD, Instituto de Economía Digital [Int12]. Así, los autores consideran que el error introducido es aceptable ya que fueron instados a consultar páginas de uso cotidiano para un ciudadano promedio [LLBG11] como el acceso a las redes sociales, la consulta del correo electrónico o sitios de información, la visualización de vídeos en *streaming* o la ejecución de búsquedas y consulta de sitios de noticias entre otros, tratando de replicar con la máxima fidelidad posible el comportamiento de un usuario estándar llevando a cabo las ya mencionadas tareas

Para este experimento se ha contado con la colaboración de algunos alumnos del Máster Universitario de Seguridad de la Información[1] —en adelante MUSI— impartido por la Universidad de Deusto como parte de su oferta de postgrado. El procedimiento para la obtención de las muestras fue el siguiente:

1. Notificación previa al departamento de sistemas de la Universidad de Deusto para la apertura temporal de las conexiones con origen o destino en las máquinas a utilizar por los alumnos. Este paso es necesario dado que la política de seguridad de un centro universitario es sensiblemente más estricta que la de una red doméstica al filtrar y bloquear gran cantidad de tráfico con el objetivo de evitar un uso fraudulento de una red abierta de acceso público.

2. Desinfección de los equipos como paso previo para evitar la inclusión del sesgo que podría introducir la existencia de una infección previa que introdujera alteraciones que pudieran ser consideradas —erróneamente— tráfico benigno.

3. Instalación y configuración de la herramienta de monitorización de tráfico en los equipos.

[1] Máster Universitario de Seguridad de la Información de la Universidad de Deusto: http://www.masterseguridad.deusto.es

4. Notificación a los alumnos de las tareas a realizar y de cuál será la utilización de la información empleada, así como para recordar cuáles son las tareas de más interés para el experimento.

5. Inicio del proceso de captura y monitorización del tráfico.

6. Finalización, recabación de muestras, desinstalación de las aplicaciones y notificación al departamento de sistemas para restaurar los protocolos de seguridad habituales.

Una vez llevado a cabo este proceso, se contará con una serie de muestras correspondientes a sesiones legítimas y que habrán sido almacenadas en el formato *.pcap* para su posterior tratamiento como parte del conjunto de datos de entrenamiento.

5.3 Selección de muestras maliciosas

Las dificultades existentes para la obtención de muestras de tráfico procedentes de un entorno real fuerza a los investigadores a recrear escenarios similares en un entorno de laboratorio. Para ello, se ha contado con cinco reconocidas familias de *malware* empleadas para el despliegue de *botnets*. Sin embargo, éstas han sido modificadas para conectarse a los servidores —controlados por los investigadores— desde los que se enviarán los comandos maliciosos.

Las muestras seleccionadas corresponden a variantes de *malware* de las siguientes familias: Flu (subsección B.1), Prablinha (subsección B.2) y Warbot (subsección B.3). Las características de las muestras se discuten en profundidad en el anexo B.

5.4 Procesamiento de las capturas

Las muestras de tráfico recabadas en el apartado anterior no son procesables de forma directa por los algoritmos que se describirán más adelante. Por ello, se hace necesario llevar a cabo un procesamiento previo de cara a obtener representaciones susceptibles de ser empleadas por la herramienta de la Universidad de Waikato WEKA que implementa diferentes algoritmos de *machine learning* o aprendizaje automáticco.

En este sentido, la información monitorizada por Wireshark y almacenada en archivos con formato *.pcap*, es empleada como entrada principal

para una serie de procesos que darán como resultado un fichero *.arff* en el que quede representado el conjunto de datos de entrenamiento. En la figura 5.1 se muestra gráficamente la secuencia de tareas a realizar que serán descritas con mayor detalle a lo largo de las siguientes páginas:

1. Obtención del fichero *.pdml* con los datos de muestra monitorizados por Wireshark.

2. Prefiltrado de características relevantes y eliminación de componentes ajenos.

3. Generación de los *m* conjuntos de datos agrupando las representaciones de cada conexión en cadenas de *n* paquetes con el tráfico pertinentemente anonimizado.

4. Obtención de los *m* conjuntos de datos en un formato procesable (*.arff*) que permita su utilización en los experimentos.

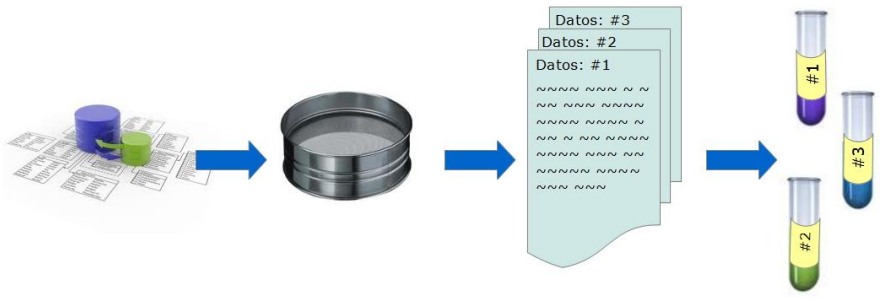


Figura 5.1: Proceso de generación del modelo de representación.

5.4.1 Preprocesamiento

El manejo de los ficheros *.pcap* en bruto no es sencillo. Pcap está concebido para dar cuenta de todos aquellos atributos con los que cuentan los paquetes que fluyen en la red observada a un nivel muy bajo, permitiéndose, gracias a herramientas como Wireshark, el filtrado de aquellos que satisfagan las restricciones del observador.

Para esta experimentación se ha aprovechado la posibilidad de exportar los datos en bruto a un fichero mucho más legible, comprensible y fácilmente interpretable —tanto para el investigador como para las aplicaciones de parseo de datos—: el formato propio de Wireshark denominado *Packet Detailed Markup Language* (PDML) con extensión *.pdml*. El manejo de este tipo de ficheros es sencillo dado que su estructura es similar a la de un fichero *.xml* y se pueden emplear con facilidad los parseadores tradicionales usados con este tipo de extensiones.

5.4.2 Generación de ficheros auxiliares de procesamiento

El resultado ha de ser la generación de un fichero con aquellas características consideradas relevantes, listas para ser parseadas y posteriormente procesadas. Uno de los primeros pasos a realizar es la secuenciación de todos los paquetes relativos a una misma conexión para su posterior agrupamiento en cadenas de n paquetes consecutivos. El objetivo es extraer características como la longitud o el tamaño de las cabeceras, los valores de los *flags* y otros atributos atómicos como el intervalo de tiempo entre paquetes para generar representaciones puntuales de la conexión. Por cada paquete se extraerán aquellos atributos relevantes para la clasificación en función de las características concretas de las conexiones maliciosas a analizar.

Vamos a entender una cadena identificativa de un paquete p_i perteneciente a una captura de tráfico de p paquetes tal y como se muestra en la ecuación 5.1:

$$p_i = IP\_src_1 : Port\_src_1 : IP\_dst_1 : Port\_dst_1 \tag{5.1}$$

La estructura estándar del fichero intermedio es la que se puede ver en la ecuación 5.2 de la que se han extraído n atributos estáticos $a_{i1}, a_{i2}, \cdots, a_{in}$ de cada paquete además del propio *timestamp* T_i —entendiendo como tal el *UNIX timestamp* o cantidad de milisegundos transcurridos desde las 00:00:00 del 1 de enero de 1970— y del tipo de la conexión $tCon_i$, etiqueta colocada

de forma automática ya que el origen y destino de la conexión son elementos conocidos por el investigador.

$$c = \begin{bmatrix} p_1 & T_1 & a_{11} & a_{12} & \cdots & a_{1n} & tCon_1 \\ p_2 & T_2 & a_{21} & a_{22} & \cdots & a_{2n} & tCon_2 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ p_p & T_p & a_{p1} & a_{p2} & \cdots & a_{pn} & tCon_p \end{bmatrix} \tag{5.2}$$

Como resultado de este proceso, para cada muestra de tráfico se obtendrá un fichero de texto que almacenará los atributos atómicos observados y en el que se verá representado el array bidimensional anterior. La ventaja de obtener este fichero intermedio es que se hace más sencillo su cruce posterior con otros ficheros auxiliares de tráfico benigno y malicioso para generar nuevos conjuntos de entrenamiento que las combinen.

De cara a desarrollar representaciones combinadas de secuencias de *n-paquetes* pertenecientes a una misma conexión, se hace indispensable almacenar a lo largo de todo el proceso intermedio de obtención de los conjuntos de datos esta información, a sabiendas de que no será incluida en el fichero *.arff* final. Con esta información y si asumimos que no existe ningún proceso de más bajo nivel encargado de ocultar el destino de las mismas —por ejemplo, empleando la redirección del tráfico a través de una serie de *proxies* o usando *darknets* como Tor[1]—, resultaría trivial desarrollar cualquier proceso de clasificación de conexiones: en ese caso, el mero hecho de la identificación de las IP origen o destino bastarían para conocer la reputación de la conexión. Por ello, en el método que se propone en esta tesis doctoral atributos como las IP y los puertos de origen y destino no serán empleados como características relevantes para el etiquetado de las muestras. Esto permitirá también la identificación de tráfico malicioso procedente de servidores de *Command & Control* situados en localizaciones de reciente aparición y sobre las que no se dispone de ningún tipo de información acerca de cuál es su cometido real.

5.5 Limitaciones

Si bien es cierto que la potencial aparición de sesgos ha sido tenida en cuenta desde un primer momento, en esta sección se quiere hacer un especial hincapié en la forma en que estas problemáticas han sido tratadas. En la subsección 5.5.1 se recogen especificades de las redes de comunicaciones

[1] http://www.torproject.org

que podrían cuestionar la forma en que ha sido concebido el experimento y cómo el doctorando se ha enfrentado a ellas para erradicar la inclusión de estos sesgos. Asimismo, en la subsección 5.5.2 se recogerán problemáticas más relacionadas con la forma en la que el factor humano afecta al proceso de obtención de muestras de tráfico real.

5.5.1 Aspectos técnicos

Uno de los aspectos a tener encuenta a la hora de monitorizar tráfico en un entorno de laboratorio es el número de saltos que los paquetes tienen que dar desde el servidor que simula el *hosting* de una plataforma fraudulenta hasta las máquinas infectadas.

Supongamos que un investigador emplaza servidores de *Command & Control* de una *botnet* en un alojamiento gestionado por la propia institución para la que trabaja. En el proceso experimental de recogida de muestras de tráfico y a la hora de escuchar el tráfico generado hacia dichos servidores *próximos* los paquetes realizarán una serie de n saltos —*hops*— hasta alcanzar su objetivo. Sin embargo, a la hora de conectarse a servidores externos de servicios legítimos la cantidad m de saltos —y por tanto— la distancia será mayor dado que la localización física de estos no está afincada en la institución. A continuación se proponen dos alternativas para tratar esta problemática.

Formas de enfrentarse al problema

El principal mecanismo para evitar este problema se basa en conseguir que la distribución espacio-temporal del envío y recepción de los paquetes sea independiente de la localización de los servidores en sí mismos y que dependa únicamente de las características concretas del *malware* implementado para desplegar la *botnet*. El problema al que hay que hacer frente es el sobreajuste que se puede producir ante valores de tiempo demasiado relacionados con el emplazamiento del servidor. Se puede decir que existen en este punto dos formas de afrontarlo:

1. **Distribuir los servidores por la red**. Este enfoque se centraría en colocar los servidores en alojamientos físicos alejados del punto en el que se produce la observación de modo que los paquetes que tengan dicho origen o destino tengan que efectuar un número similar de saltos para alcanzarlo. La contrapartida es que, aún estando situados en servidores externos, estos tampoco deberían ser los mismos

y deberían encontrarse a *distancias* lógicas asépticas para que se evitara la detección *por proximidad* que pudiera introducir nuevamente el mismo sesgo. Asimismo, existirían también otras alternativas más originales para simular el alejamiento de una red determinada. Una posibilidad podría considerar la canalización de todo el tráfico, tanto benigno como malicioso, a través de un *proxy* externo situado en un emplazamiento neutral de modo que toda la información tuviera que viajar hacia/desde él. Aunque a la postre la problemática subyacente sería la misma —se corre el riesgo de que lo que se termine por detectar sea la distancia hasta el alojamiento del servidor de *Command & Control*—, el error cometido sería inferior si la elección del canal estuviera bien realizada. La utilización de *darknets* como Tor podrían ayudar en esta labor al reasignar el tráfico a través de canales cifrados al vuelo introduciendo confusión en lo que a la separación de los paquetes se refiere.

2. **Normalizar los espacios temporales**. La opción por la que se ha optado en esta tesis doctoral va un poco más allá, proponiendo la normalización de la separación espacio-temporal entre paquetes de una misma representación. A la hora de definir estos atributos se ha rechazado la idea de tener en cuenta la separación temporal entre paquetes tomada como valores absolutos. Para ello se ha considerado relevante relativizar los tiempos tomando como tiempo base la separación entre los dos primeros paquetes y ajustando el resto de separaciones como variaciones con respecto a dicho valor.

   Un ejemplo dejará esto más claro. Considerése una serie de paquetes c_{00}, c_{01} y c_{02} relativos a una conexión C_0 —vamos a decir— *próxima* y observados en los instantes de tiempo $T_{00} = 10ms$, $T_{01} = 30ms$ y $T_{02} = 60ms$. La lista de separación entre estos no será tenida en cuenta como $\{20ms, 30ms\}$ si no como $\{1,\ 1,5\}$. Pongamos ahora que se considerara una conexión similar C_1 (tal que $C_1 \approx C_2$) emplazada en un servidor —vamos a decir— *más alejado* y formada por una serie de paquetes c_{10}, c_{11} y c_{12} observados en los instantes de tiempo $T_{10} = 100ms$, $T_{11} = 300ms$ y $T_{12} = 600ms$. La lista de separación entre ellos en términos absolutos sería de $\{200ms, 300ms\}$, pero esta separación podría ser atribuible a conceptos relacionados con la calidad del servicio, la intensidad de la señal o la disponibilidad de la red. Nuestro enfoque propone la utilización de los valores relativizados de dicha separación como criterio disgregador: es decir, $\{1,\ 1,5\}$.

Con la adopción de esta segunda metodología con la que tomaremos en consideración la separación espacio-temporal de los paquetes, se da respuesta a dos cuestiones: por un lado, se pretende minimizar el sesgo que introduce la distancia del emplazamiento real del servidor de control y, por otro, se propone un método transparente para la consideración de dichas atribuciones en el futuro.

5.5.2 El factor humano

Las muestras de tráfico almacenadas corresponden a diferentes sesiones de tráfico normal llevado a cabo por estudiantes del Máster Universitario de Seguridad de la Información de la Universidad de Deusto de modo que se procedería a la monitorización de las conexiones que realizaran durante las siguientes horas de navegación. Como nota adicional, y por las implicaciones legales que conlleva el almacenamiento de tráfico, se hacía necesario la aceptación tácita de estos estudiantes de su participación en el experimento que se iba a desarrollar.

Si bien es cierto que el proceso de obtención de las muestras de tráfico está pensado para minimizar la aparición de sesgos y evitar un entrenamiento erróneo y artificial de los clasificadores, no se puede obviar que la ejecución de estas pruebas conlleva algunas limitaciones. Así, la obligación legal —aunque también ética y moral— de notificar a los usuarios las monitorización, implica también la introducción de un cierto sesgo en cuanto al contenido final de las comunicaciones, ya que un usuario recientemente advertido de la ejecución de un experimento no será habitualmente dado a visitar páginas relativas a sitios pornográficos o entidades financieras.

Se cree conveniente dejar constancia de esta situación de cara a poner sobre aviso al lector acerca de las posibles desviaciones. Los aspectos legales que entraña la ejecución de este tipo de experimentos no pueden ser ajenos al investigador de cara a la protección de la información almacenada, más aún si en ella podrían incluirse información sensible como nombres de usuario o contraseñas.

Asimismo, hay que tener en cuenta que la simulación de comportamiento real tiene otras limitaciones susceptibles de introducir sesgos. La principal alteración que introduce este enfoque viene dada por el efecto disuasorio que tiene el mero hecho de que el usuario sepa que su tráfico esté siendo monitorizado. Esto es relevante dado que, aún de forma involuntaria, el usuario puede introducir hábitos de navegación con los que extreme las precauciones artificialmente a la hora de evitar acceder a páginas o

servicios de riesgo como plataformas de pago o sitios con contenidos para adultos a los que podría acceder si no existiera esa monitorización. Se asume por tanto que los usuarios serán más cuidadosos de lo normal con el tratamiento de su propia información al saber que están participando en un experimento, una sensación que las va a acompañar a lo largo del mismo.

En este sentido, la alternativa era realizar el proceso de obtención de las muestras de tráfico legítimo en un entorno real. Sin embargo, desde el punto de vista metodológico este enfoque tiene las mismas contraindicaciones que los métodos de detección de anomalías: la posibilidad de que se filtre tráfico ilegítimo en las muestras normalizadas y etiquetadas como legítimas es una realidad, al no poderse garantizar la desinfección previa de estos equipos. Este tipo de *filtraciones* deben ser consideradas ya que podrían introducir sesgos y errores en los sistemas de aprendizaje automático utilizados.

De cualquier manera, se considera que el error introducido es aceptable ya que los sujetos fueron instados a consultar páginas de uso cotidiano para un ciudadano promedio [LLBG11] como el acceso a las redes sociales, la consulta del correo electrónico o sitios de información, la visualización de vídeos en *streaming* o la ejecución de búsquedas, entre otros.

5.6 **Sumario**

En este capítulo se ha definido el procedimiento para la obtención de muestras de tráfico susceptibles de ser utilizadas para entrenar los modelos de aprendizaje automático definidos en el capítulo siguiente. Para ello se ha definido todo el proceso de obtención de muestras correspondientes a tráfico legítimo en función de comportamientos estándar de los usuarios a la hora de navegar por Internet. De igual manera, se han introducido algunas de las características de los ejecutables utilizados para las infecciones controladas así como los pasos tomados para un despliegue efectivo de las mismas.

Asimismo, se han definido también cómo se han extraído las características relevantes para la investigación y cómo se ha desarrollado el proceso de segregación de aquellas que —por ser direcciones IP o puertos— eran susceptibles de introducir sesgos en la clasificación. De igual forma, también se han puesto sobre la mesa aspectos importantes en lo que respecta a la inclusión de sesgos en las muestras y se han detallado las medidas que el doctorando ha puesto en marcha para evadir estas problemáticas.

«El pesimista se queja del viento; el optimista espera que cambie; el realista ajusta las velas».

William George Ward, teólogo y matemático inglés (1812–1882).

Capítulo

6

Detección de tráfico procedente de canales de *Command & Control*

El doctorando presenta en esta tesis doctoral un modelo experimental que hace uso de las técnicas de modelado descritas anteriormente. Una vez satisfecho el objetivo de generar una cantidad de tráfico tal que puede ser empleada por algunos de los algoritmos implementados en Weka para la clasificación de tráfico, se procederá a aplicar una metodología que permitirá optimizar y anticipar el proceso de categorización de conexiones en base a los atributos ya definidos anteriormente.

De esta manera, la propuesta que se desarrolla en esta tesis doctoral considera la identificación de comunicaciones HTTP de *Command & Control* como un problema de clasificación supervisada. Esta elección no es arbitraria: se ha extrapolado la metodología de trabajo de detección de *botnets* IRC [GZL08] también aplicada con éxito en otros campos tan variados como la botánica [ČC09], las imágenes [ZZ12], la detección de *malware* [SBN+10a, IA, ITBV12] o la clasificación de *software* en general [CX12]. Para ello, se aplicarán a las diferentes formas de enunciar el problema, algoritmos de aprendizaje automático que trabajen en esa línea, a sabiendas de la gran cantidad de esfuerzos adicionales que requiere el proceso de eti-

quetado en la utilización de métodos supervisados frente a otro tipo de aproximaciones [TT12].

En este contexto, las contribuciones que se han realizado son las siguientes:

- Se muestra la forma de utilizar los atributos de los paquetes —y de las secuencias de estos— como modelo de representación para la detección y categorización de conexiones empleando técnicas de aprendizaje supervisado.

- Se provee de una validación empírica del método supervisado en un estudio exhaustivo sobre la idoneidad de diferentes modelos de aprendizaje automático para detectar conexiones pertenecientes a familias de *botnets* conocidas.

- Se demuestra la obtención de unas tasas de identificación suficientemente altas como para poder contar con una categorización *a priori* de las conexiones que se establezcan.

- Se determina la capacidad del modelo para detectar conexiones genéricas de *botnet* en un bosque de conexiones maliciosas.

El resto de este capítulo está estructurado como sigue. La sección 6.1 define las técnicas de aprendizaje supervisado empleadas para la clasificación. La sección 6.2 recoge la metodología desarrollada para los experimentos. La sección 6.3 detalla los resultados obtenidos en el proceso de validación empírica expuesto. La sección 6.4 discute cuáles son las implicaciones de estos resultados y sus posibles aplicaciones en modelos futuros de detección. Por último, la sección 6.5 resume las principales conclusiones a extraer de este capítulo.

6.1 Aprendizaje supervisado

De igual manera a como ya hicieran en el pasado Livadas *et al.* para detectar *botnets* controladas por IRC [LWLS06], se ha planteado el problema como un problema de clasificación supervisada. En este tipo de problemas, se estudia un fenómeno representado por un vector X en R^d que puede ser clasificado de K maneras diferentes de acuerdo a un vector Y de *etiquetas*. El aprendizaje supervisado y la utilización de algoritmos de clasificación hacen uso de un conjunto de datos previamente etiquetado [Kot] que, en

nuestro caso, se corresponde con las etiquetas de tráfico *legítimo* o procedente de una *botnet* en función del origen de las representaciones utilizadas para el experimento.

Con tal fin, se dispone de un conjunto de entrenamiento $D_n = \{(X_i, Y_i)\}_{i=1}^{n}$, donde X_i representa los sucesos correspondientes al fenómeno X mientras que Y_i es la etiqueta que lo sitúa en la categoría que el clasificador asume como correcta. Por ejemplo, para el caso de longitudes de secuencia de $n = i$ paquetes ($\forall 1 <= i <= 12$), estaremos hablando de una secuencia X_i definida por una serie de atributos que lo representan, siendo Y_i la categoría asignada a dicha secuencia según las estimaciones de cada clasificador. Así, conociendo las características concretas de las conexiones asociadas al tráfico malicioso generado de forma experimental por parte de los autores, se ha podido etiquetar con facilidad la finalidad de cada uno de los paquetes como de *bot* o *legítimo* para generar los *datasets* de entrenamiento empleados en la sección 6.3.

En las páginas que siguen se revisan los diferentes algoritmos de aprendizaje supervisado utilizados para afrontar el problema de la detección de tráfico de *Command & Control* como un problema de clasificación supervisada. Se ha optado por comparar el rendimiento de los diferentes algoritmos de clasificación dadas las ocasionalmente notables diferencias de efectividad que se pueden observar en experimentos similares llevados a cabo en otros ámbitos como la detección de errores en modelos de calidad del *software* [SKM09] o la clasificación automática de comentarios en sitios web sociales [SdlPSPL+12] entre otros.

6.1.1 El método *Support Vector Machines* (SVM)

Los clasificadores SVM —también conocidas como máquinas de soporte vectorial en castellano— se basan en un hiperplano que divide una representación n-dimensional de los datos en dos regiones separadas. Se define el hiperplano como aquel espacio que maximiza el margen m entre dos clases, también llamadas *regiones espaciales*. Este margen se define a su vez como la distancia más larga entre los ejemplos de las clases, calculada a partir de la distancia de los especímenes más cercanos al hiperplano —véase la figura 6.1—. La idea es que, haciendo uso de un conjunto de datos de entrenamiento, un SVM genere un modelo lineal capaz de predecir si un nuevo espécimen desconocido pertenece a una u otra categoría aplicando la citada división del hiperplano. Para ello, las máquinas de soporte vectorial hacen uso de diferentes funciones de kernel.

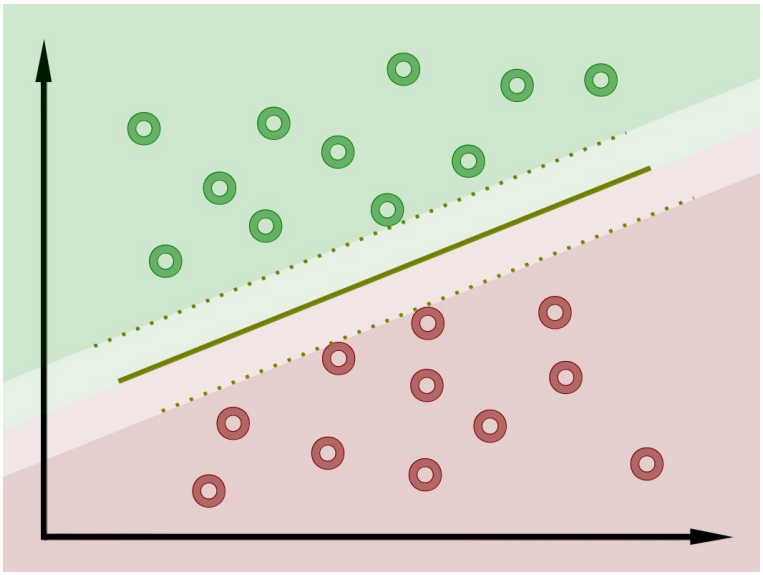


Figura 6.1: Ejemplo de un clasificador SVM en un espacio bi-dimensional.

Funciones de *kernel*

Si se utiliza la *forma dual*[1] de la representación cuadrática para representar al hiperplano, se revela que la tarea de clasificación es únicamente una función de los vectores de soporte [Bis06]. En esta tesis, utilizamos las siguientes funciones de *kernel* como formas de medir cómo de lejos se encuentran las diferentes entidades: un *kernel polinomial*, *polinomial normalizado* [AW99] y [AW99]

- Los ***kernels* polinomiales** [AW99] han sido utilizados con éxito en los últimos años con SVM especialmente usados para afrontar problemas relacionados con el procesamiento del lenguaje natural tanto en su versión normal:

$$k(x, xy) = (x \cdot y)^d \tag{6.1}$$

como en su versión normalizada:

$$K(x, y) = \frac{(x \cdot y)^d}{\sqrt{(x \cdot y)^d}} \tag{6.2}$$

[1] En programación lineal, el problema primario y la forma dual con complementarios. Una solución a cualquiera de ellos determina la solución a ambos.

En ambos casos, d representa el grado del polinomio. Es habitual no encontrarse con valores de d superiores a 2, ya que se ha observado que se presentan problemas de sobreajuste a los datos de muestra [CHC+10]. Por situaciones de sobreajuste hay que entender aquellas situaciones que se dan cuando uno de los clasificadores ofrecen resultados óptimos para el conjunto de datos de entrenamiento utilizados pero este no es capaz de acomodarse a la variedad de datos que otros proveen. Esta característica puede dar lugar a errores en la aplicación de este algoritmo de clasificación en entornos reales [SLTW04].

- ***Kernel Radial Basis Function* (RBF)** [AW99]. Es uno de los más populares utilizados en la clasificación empleando máquinas de soporte vectorial [CHC+10]. El *kernel* RBF entre dos muestras x e y representadas por vectores de atributos se define como:

$$K(x,y) = \exp(-\frac{||x-y||_2^2}{2\sigma^2}) \tag{6.3}$$

- ***Pearson VII Universal Kernel (PUK)***. Utilizada con frecuencia en el campo de la espectroscopia[1], es una alternativa a las propuestas de *kernels* anteriores [UMB06]. Se define como:

$$k(x,y) = \frac{1}{\left[1 + \left(\frac{\sqrt{||x-y||^2}\cdot\sqrt{2^{\frac{1}{\omega}}-1}}{\delta}\right)^2\right]^{\omega}} \tag{6.4}$$

6.1.2 Método *Bagging*

Bagging —abreviatura de *bootstrap aggregating*— fue un método propuesto por Leo Breiman en 1994 [Bre94] para mejorar los métodos tradicionales de clasificación mediante la combinación de clasificaciones sobre conjuntos de datos de entrenamiento generados aleatoriamente. El método *Bagging* es un meta-algoritmo diseñado para mejorar la estabilidad y precisión de otros algoritmos de aprendizaje automático en procesos de clasificación estadística y regresión. Una de las cualidades del método es que puede ser utilizado para proveer estimaciones mejoradas sobre las probabilidades reales de

[1] La espectroscopia es el estudio de la interacción entre la radiación electromagnética y la materia, con absorción o emisión de energía radiante. Tiene aplicaciones en campos como la astronomía, la física y la química.

un nodo al utilizar valores estimados de salida y no las salidas observadas tal cual, lo que a la postre permitiría reducir la varianza y minimizar los problemas relacionados con el sobreajuste [Bre96b]. Aunque es habitualmente utilizado para los árboles de decisión también puede ser utilizado para cualquier tipo de método.

Así, dado un conjunto D de datos estándar de tamaño n, Bagging genera un número m de nuevos conjuntos de entrenamiento D_i, cada uno de ellos de un tamaño $n' < n$, extrayendo a continuación muestras de D uniformemente. Mediante la extración de muestras con reemplazamiento, algunas de las observaciones pueden ser repetidas en cada uno de los conjuntos D_i, lo que ocurrirá con más frecuencia cuanto más próximo a n sea n'. En el caso particular de un conjunto de datos D_i formado por un número de mustras $n' = n$, se espera que se cuente con $1 - \frac{1}{e}$ de muestras únicas de D —aproximadamente, $\pm 63,2\,\%$ de ellas—, siendo el resto muestras duplicadas [Bre94]. Este tipo de muestreo es conocido como *bootstrap sample*. Los m modelos generados son ajustados utilizando las m muestras y son combinados promediando la salida —para regresiones— o votando —en el caso de la clasificación—.

Según Breiman, Bagging deriva en «mejoras sustanciales» de aquellos modelos inestables [Bre96a], entre los que se incluyen las redes neuronales o los árboles de clasificación y de regresión. Por este motivo, el algoritmo utilizado es REPTree, un árbol de decisión de aprendizaje rápido y que también ha sido utilizado en el pasado para mejorar la precisión de los procesos de *clustering* o agrupamiento [DF03]. Por otro lado, es también cierto que este método podría ver moderadamente perjudicado el rendimiento de los métodos estables, como el de KNN.

6.1.3 Árboles de decisión

Los árboles de decisión —véase la figura 6.2— son un tipo de clasificadores de aprendizaje automático que pueden ser representados gráficamente como árboles. Los nodos interiores representan las condiciones o estados posibles de las variables del problema y los nodos finales u *hojas* constituyen la decisión final del clasificador [Qui86].

Un árbol de decisión es, formalmente, un grafo $G = (V, E)$ que consiste en un conjunto no vacío de nodos finitos V y un conjunto de aristas E. Si el conjunto de las aristas E está compuesto a su vez por bi-tuplas ordenadas de nodos de la forma (v, w), entonces se puede decir que el grafo G es dirigido.

Un *camino* en el grafo G se define como una secuencia de aristas de la

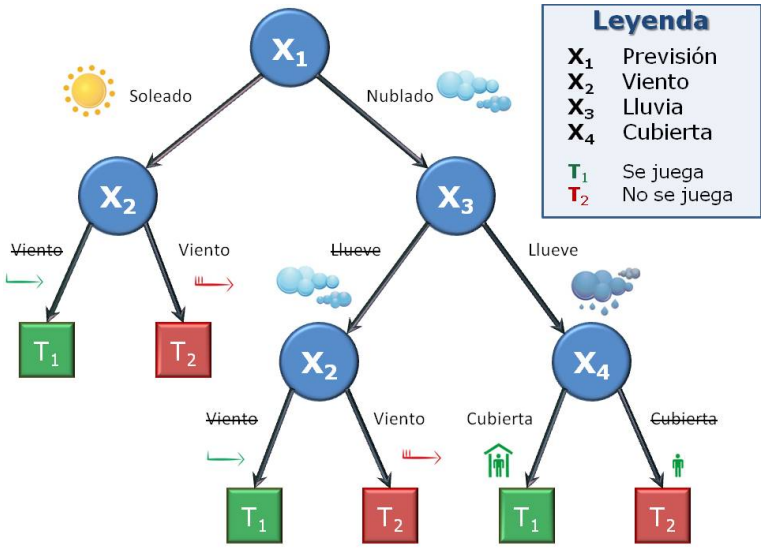


Figura 6.2: Un ejemplo de árbol de decisión. Los nodos interiores representan los diferentes estados que una variable puede tomar, mientras que los nodos finales u hojas son la decisión final que el clasificador puede tomar.

forma $(v_1, v_2), (v_2, v_3), ..., (v_{n-1}, v_n)$. Los caminos se pueden expresar por el origen, su final y la *distancia* recorrida, entendiendo como tal el mínimo número de aristas desde el origen hasta el final. Además, si (v, w) es una arista en el árbol, entonces se considera a v *nodo padre* de w, mientras que w sería el *nodo hijo* de v. El único nodo en el árbol sin padres se denomina *nodo raíz*. Cualquier otro nodo en el árbol, es un *nodo interno*. Para construir la representación gráfica de un árbol, se contesta a un conjunto de preguntas binarias.

Hay varios algoritmos de aprendizaje supervisado, que se utilizan para aprender la estructura de un árbol mediante un conjunto de datos etiquetados. En este trabajo, hemos utilizado *random forest* —en castellano, bosque aleatorio—, que es una combinación o *ensemble* —combinación de varios clasificadores en uno más fuerte— de árboles de decisión construidos al azar. Además, hemos utilizado J48, que es la implementación de WEKA [Gar] del algoritmo C4.5 [Qui93].

6.1.3.1 **Algoritmo C4.5**

C4.5 crea árboles de decisión dada una cantidad de información de entrenamiento usando el concepto de entropía de la información [Sal94]. En

cada nodo del árbol de decisión, el algoritmo elige un atributo de los datos que más eficazmente dividen el conjunto de muestras en subconjuntos enriquecidos en una clase u otra utilizando como criterio de selección la diferencia de entropía o la ya mencionada *normalized information gain*). Como ya se ha dicho, J48 es una implementación de código abierto para Weka del algoritmo C4.5 [qui, Qui93] que a su vez es una extensión del algoritmo ID3 [Qui86]. El algoritmo construye árboles de decisión utilizando un conjunto de datos etiquetados y mediante el concepto de *entropía de la información*, que mide la incertidumbre asociada con una variable aleatoria. Este concepto, también conocido como *entropía de Shannon* [Sha51, SW49], mide el valor esperado de información (*Information Gain*, IG) dentro de un mensaje.

Para generar un árbol de decisión mediante el algoritmo C4.5, se requiere un conjunto de datos de entrenamiento $S = \{s_1, s_2, ..., s_{n-1}, s_n\}$ previamente etiquetado. Así, los datos de entrenamiento conforman un *corpus* de ejemplos $s_i = x_1, x_2, ..., x_m$ ya clasificados en los que $x_1, x_2, ..., x_{m-1}, x_m$ representan los m atributos o características del ejemplo. El atributo m de cada representación corresponderá con alguna de las o etiquetas $C = \{c_1, c_2, ..., c_{o-1}, c_o\}$ y representarán la clase a la que pertenece dicho espécimen s_i.

Para cada nodo del árbol, el algoritmo C4.5 elige el atributo dentro de los datos que, de forma más eficiente, divide el conjunto restante de especímenes en subconjuntos de una clase o de la otra. El criterio para dividir el conjunto es la *ganancia de información* o *Information Gain* (IG), la cual calcula la dependencia estadística entre dos variables aleatorias. De este modo, el atributo con el mayor IG se selecciona para ser el nodo que divida el nuevo conjunto de datos. Este proceso se repite recursivamente hasta que no se puedan realizar más divisiones del conjunto de datos.

C4.5 es capaz de manejar tanto atributos continuos como discretos, datos sin completar, atributos con diferentes costes y, además, puede podar[1] los árboles después de generarlos.

6.1.3.2 El método *Random Forest*

El algoritmo *Random Forest* se trata de un clasificador de agregación desarrollado por Leo Breiman [Bre01b] y que consta de un abanico de subárboles de decisión —o *decision trees* en inglés— elegidos de forma aleatoria. *Random Forest* mejora la precisión en la clasificación ya que en la construcción de cada clasificador individual se introduce un componente estocás-

[1] La poda de árboles de decisión es un proceso de optimización de estos.

tico, bien en la partición del espacio —es decir, introduciendo elementos aleatorios en la propia construcción de los árboles haciendo del sistema uno no determinista— o bien en la muestra de entrenamiento [CdSI10].

Para la clasificación de una instancia, se genera entonces un vector de entrada para cada uno de los árboles de la combinación. De esta manera cada árbol genera una clasificación siendo el clasificador global el que elija aquella que haya obtenido una mayor cantidad de votos.

Como ya se ha comentado, a la hora de generar los árboles de decisión aleatorios que componen a un *Random Forest*, se elige un subconjunto aleatorio del conjunto total de datos. Para ello, el método utiliza una discriminación estocástica [Kle96].

Seguidamente, se calcula la mejor división posible para ese subconjunto de características. Se repite el proceso para cada nodo del árbol de decisión sin llegar a efectuar una poda del mismo una vez que el árbol ha sido creado —al contrario que en la mayoría de métodos para la construcción de árboles de decisión—.

Es de esta manera como el algoritmo *Random Forest* produce un clasificador preciso para muchos conjuntos de datos, manteniendo su precisión incluso cuando existe un gran número de variables de entrada. Además, estima la importancia de las mismas a la hora de determinar la clasificación final y es capaz de enfrentarse a datos incompletos. Sin embargo, suelen ser bastante susceptibles a sobreajustarse al conjunto de datos, especialmente en tareas con una gran cantidad de datos ruidosos en donde el proceso de recogida no ha sido realizado de forma correcta.

6.1.4 **Redes bayesianas**

El teorema de Bayes [Bay63] es la base para la inferencia bayesiana, un método estadístico de razonamiento que determina, basándose en un número de observaciones, la probabilidad de que una hipótesis dada sea cierta. Una de sus principales características es que el teorema de Bayes ajusta la probabilidades tan pronto como se dé con nuevas observaciones.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{6.5}$$

De acuerdo con su formulación clásica (véase la ecuación 6.5), dados dos eventos A y B, la probabilidad condicionada $P(A|B)$ de que ocurra A si ha ocurrido B, se puede obtener si sabemos la probabilidad de que ocurra A,

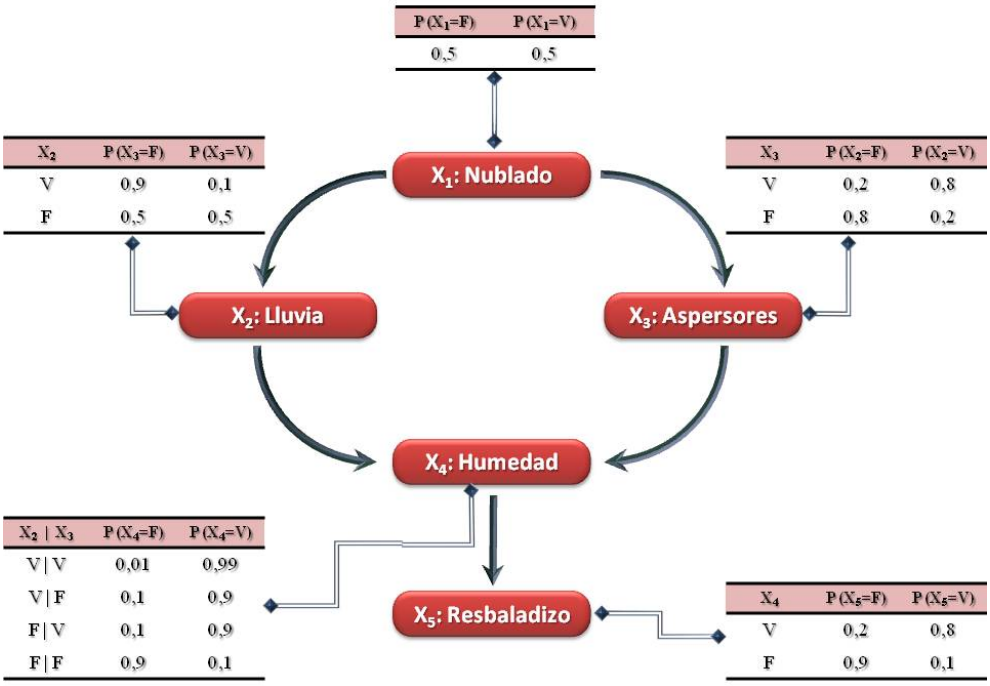


Figura 6.3: Ejemplo de red bayesiana.

$P(A)$, la probabilidad de que ocurra B, $P(B)$, y la probabilidad condicionada de B dado A, $P(B|A)$.

Podemos decir por tanto que las redes bayesianas —véase la figura 6.3— conforman un modelo probabilístico que representa una colección de variables aleatorias —y también sus dependencias condicionales— mediante un grafo dirigido (DAG) que indica explícitamente la influencia causal entre ellas. Formalmente, son grafos dirigidos acíclicos con una función de distribución probabilística asociada que representa la fuerza de las relaciones mantenidas [CGH96]. Los nodos de dicho grafo representan variables —que pueden ser tanto premisas como conclusiones— mientras que las aristas representan dependencias condicionales entre las variables. Para las necesidades aquí expuestas, la capacidad más importante de las redes bayesianas es su habilidad para inferir la probabilidad de que cierta hipótesis sea cierta: en nuestro caso, la probabilidad de que un paquete o una secuencia de ellos proceda de un canal de *Command & Control* de una *botnet*.

Existen varios algoritmos para el aprendizaje de la estructura de una red bayesiana. En esta tesis, se utiliza K2 [CH91], *Hill Climbing* o *escalada de colinas* [RN03] y *Tree Augmented Naïve* (TAN) [GGP+97], además del

clasificador *Naïve Bayes* (clasificador bayesiano ingenuo) [Lew98].

6.1.4.1 Clasificador *Naïve Bayes*

Un clasificador bayesiano ingenuo —del inglés, *Naïve Bayes*— es un clasificador probabilístico basado en el teorema de Bayes que aplica una serie de operaciones de simplificación. La idea es que si el número de variables independientes que se manejan es demasiado grande, carece de sentido aplicar tablas de probabilidad [JWCY07], de ahí las reducciones que simplifican la muestra y que le dan el apelativo de *Bayes ingenuo*. Asimismo, asume que no existe ningún tipo de dependencia entre los nodos de la red bayesiana [CGH96]. Por tanto, cada nodo en la red está únicamente vinculado con el nodo conclusión de la misma.

El clasificador combina el modelo bayesiano con una regla de decisión (véase la ecuación 6.6):

$$\underset{c \in C}{\text{argmáx}}\ P(c) = \prod_{i=1}^{n} P(F_i = f_i | C = c) \tag{6.6}$$

donde cada f_i corresponde a cada una de las características conocidas del espécimen a clasificar.

6.1.4.2 Algoritmo de aprendizaje K2

El algoritmo K2 busca la estructura de red más probable dado un conjunto de datos. Para esto, el algoritmo adapta un algoritmo *hill climbing* restringiendo el orden en las variables [GGP+97]. La diferencia de K2 con el *hill climbing* sencillo es que este algoritmo no considera las aristas de la primera red ingenua para borrarlas posteriormente.

El algoritmo K2 comienza asumiendo que un nodo no tiene padre para, seguidamente, añadir de forma incremental los padres cuya adición incremente la probabilidad de la estructura resultante (véase la ecuación 6.7.

$$f(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r-1)!}{(N_{i,j} + r_i - 1)!} \cdot \prod_{k=1}^{r_i} \alpha_{i,j,k}! \tag{6.7}$$

donde π_i es el conjunto de padres del nodo x_i. q_i es el cardinal del conjunto phi_i, que contiene todas las posibles instancias de los nodos padre de x_i en el conjunto de datos D. Por otro lado, r_i es el cardinal del conjunto V_i que contiene todos los posibles valores para el nodo x_i. $\alpha_{i,j,k}$ es el número de

especímenes dentro de D donde un atributo x_i se instancia con su valor k-ésimo, y los nodos padre de x_i en π_i se instancian con la j-ésima instancia en ϕ_i.

Ocasionalmente, el algoritmo termina de añadir padres a los nodos. Esto sucede cuando al añadir un nodo padre no se incrementa la probabilidad de la estructura de la red [LRS] y, por tanto, el algoritmo finaliza su ejecución, dando como resultado la estructura de la red bayesiana final.

6.1.4.3 **Método de aprendizaje *Hill Climbing***

Hill Climbing [RN03] es un método de búsqueda local. El algoritmo intenta maximizar una función heurística $f(x)$. Con este fin, comienza con una solución aleatoria e iterativamente realiza pequeños cambios a esta, consiguiendo mejoras en cada iteración. Cuando el algoritmo no puede conseguir más mejoras en la función heurística, este finaliza. A pesar de que el objetivo es encontrar una solución cercana a la óptima, no se puede garantizar que el algoritmo *Hill Climbing* encuentre la mejor solución y, en muchas ocasiones, se obtiene un máximo local[1].

Como algoritmo de aprendizaje bayesiano, añade, borra y da la vuelta a las aristas basándose en una función heurística. La búsqueda no se restringe por un orden en las variables, al contrario que lo que hace K2.

La heurística que se suele emplear para buscar las relaciones entre los nodos, es la función de puntuación bayesiana, denominada también *Bayesian scoring function* [DGP08].

6.1.4.4 **El método *Tree Augmented Naïve* (TAN)**

El método TAN es un método que determina el peso máximo que abarca un árbol para luego devolver una red bayesiana ingenua aumentada con dicho árbol. Este método mejoraba los resultados obtenidos por *Naïve Bayes* en términos de precisión [FGG97] mediante la imposición del árbol de decisión sobre la estructura bayesiana con la ventaja de que mantiene la simplicidad computacional —al no necesitar la realización de búsquedas— y la robustez característica del clasificador *Naïve Bayes*.

[1] Un máximo local se corresponde con el máximo valor encontrado en un intervalo de observación pudiendo ser este inferior a otros valores obtenidos para una misma función fuera de dicho intervalo.

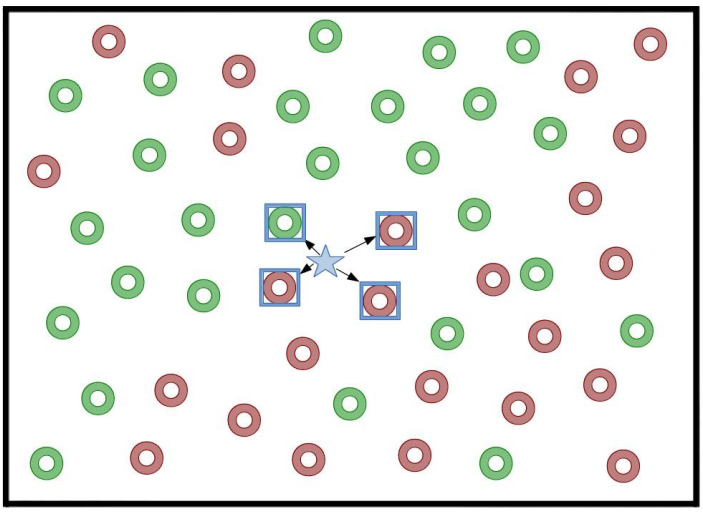


Figura 6.4: Ejemplo del funcionamiento de kNN en un espacio bidimensional para $k = 4$ vecinos.

6.1.5 El algoritmo *K-Nearest Neighbours* (KNN)

El algoritmo KNN [FH52] es uno de los algoritmos de clasificación más simples de los relativos a las técnicas de aprendizaje automático. Se trata de un método de clasificación basado en el análisis de los k vecinos más próximos en el espacio analizado ($\forall k \in \mathbb{N}$). En este caso y dada la simplicidad del algoritmo, los valores de k empleados para comprobar la efectividad del mismo han sido $k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$ a fin de determinar si la toma en consideración de más vecinos mejora significativamente los resultados.

En la fase de entrenamiento de este algoritmo, se utiliza un conjunto de instancias de entrenamiento $S = \{s_1, s_2, ..., s_{n-1}, s_n\}$ donde n es el número de variables de cada instancia (e. g., los atributos que aparecen procedentes de cada paquete o secuencia de paquetes). Así, estos datos son representados posteriormente en un espacio de entrenamiento —véase la figura 6.4—.

En la fase de clasificación se mide la distancia entre los especímenes del conjunto de datos de entrenamiento y la instancia a clasificar. La distancia entre dos puntos $P = (p_1, p_2, \cdots, p_n)$ y $Q = (q_1, q_2, \cdots, q_n)$ representados en n dimensiones ($P, Q \in \mathbb{R}^n$) se puede medir utilizando cualquier métrica de distancia. Para ello, la herramienta Weka implementa diferentes métricas

como la distancia Minkowski $d_{Min}(P,Q)$, definida de forma general como:

$$d_{Min}(P,Q) = (\sum_{i=1}^{n} |p_i - q_i|^p)^{\frac{1}{p}} \tag{6.8}$$

en donde la conocida como distancia Chebyshev $d_{Che}(P,Q)$ es un caso particular para $n = \infty$ y que también se puede expresar como:

$$d_{Che}(P,Q) = max_n(|p_i - q_i|) \tag{6.9}$$

Asimismo al caso particular de $p = 1$ se le conoce como distancia Manhattan $d_{Man}(P,Q)$, también definida como:

$$d_{Man}(P,Q) = |p_1 - q_1| + |p_2 - q_2| + \cdots + |p_n - q_n| \tag{6.10}$$

Sin embargo, en el caso de esta experimentación se emplea la distancia euclídea $d_E(P,Q)$ (caso particular de la distancia de Minkowski para $p = 2$) entendida como la raíz cuadrada del sumatorio de los cuadrados de las diferencias entre los n coeficientes de dos puntos dados (véase la ecuación 6.11):

$$d_E(P,Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=0}^{n} (p_i - q_i)^2} \tag{6.11}$$

Aunque existen varios métodos para determinar la clase de una instancia, la técnica más utilizada consiste en elegir la clase de la instancia como aquella más común entre los k puntos del espacio de entrenamiento más cercanos al espécimen a clasificar.

6.1.6 **Perceptrones**

Los perceptrones pueden ser considerados como algoritmos de clasificación supervisada de la familia de los clasificadores lineales. Sin embargo, por esto mismo, sus principales limitaciones residen en su incapacidad manifiesta para separar conjuntos de datos que no son linealmente separables [BO02]. En esta tesis doctoral hemos trabajado con dos aproximaciones: la del perceptrón por votación y la del perceptón multicapa o *multilayer*.

6.1.6.1 *Voted Perceptron*

Yoaund Freund y Robert E. Schapire propusieron una versión votada del perceptrón simple [FS99] para enfrentarse a las limitaciones que presentaba el algoritmo general a la hora de ser entrenado para reconocer una gran cantidad de patrones. Yoaund Freund y Robert E. Schapire fueron capaces de recuperar el interés en esta modalidad después de proponer una formulación votada del algoritmo original [FS99]. Este enfoque ha sido utilizado con éxito para solventar problemas de clasificación binaria o relacionados con el procesamiento del lenguaje natural [Col] y para afrontar situaciones relacionadas con las estrategias de entrenamiento distribuidas phttp://as.com/ara aprender complejos modelos estructurales, aún reconociendo lo costoso del proceso de entrenamiento [MHM].

6.1.6.2 **Multilayer Perceptron (MLP)**

El modelo de MLP es un modelo *feedforward* de redes neuronales artificiales que mapea conjuntos de datos de entrada a un conjunto de salida correspondiente utilizado con éxito en campos como la clasificación de *malware* [SDB$^+$] o las ciencias atmosféricas [GD98]. La principal ventaja de MLP es que puede distinguir datos que no son linealmente separables, lo que, por otra parte, constituía una de las principales limitaciones del perceptrón lineal simple como ya se ha dicho. Un MLP consta de una red neuronal artificial formada por capas de nodos en un grafo dirigido, con cada capa completamente conectada a la siguiente. Este modelo utiliza una técnica de aprendizaje supervisado de *propagación hacia atrás* o *backpropagation* para el entrenamiento de su clasificador. La técnica de *backpropagation* o aprendizaje de propagación hacia atrás consiste en minimizar un error por medio de descenso de gradiente, por lo que la parte esencial del algoritmo es el cálculo de las derivadas parciales de dicho error con respecto a los parámetros de la red neuronal.

6.2 Metodología general

Con el objetivo de poner a prueba la eficiencia de los clasificadores, se ejecutarán tres experimentos diferenciados para secuencias de longitud de hasta $n = 12$ paquetes, en los cuales se aplicarán los algoritmos de clasificación ya empleados con éxito en otros ámbitos en investigaciones complementarias publicadas en la XXXVIII Conferencia Latinoamericana en

Informática (CLEI 2012) por Brezo *et al.* [BGdlPUP+b] enunciados anteriormente contra cada uno de los conjuntos de datos definidos posteriormente. Los experimentos se agruparán en dos bloques diferenciados: la utilización de un primer conjunto de datos de prueba formado por atributos observados en paquetes individuales y un segundo compuesto por representaciones que consideran secuencias de estos. Cada uno de estos criterios tratará de dar respuesta a los siguientes aspectos de la detección:

- La identificación de cada una de las *botnets* por separado entre tráfico benigno.
- La identificación de cada una de las *botnets* en una muestra de tráfico benigno en la que también se cuente con tráfico malicioso procedente de otras *botnets*.
- La identificación de tráfico genérico de *Command & Control* en el seno de una muestra de tráfico benigno. Para generar esas muestras de tráfico genérico se etiquetarán como pertenecientes a la misma clase todas y cada una de las secuencias de paquetes de todas las muestras de *malware*.

Para dar respuesta a estas cuestiones se han recabado muestras de tráfico procedentes de diferentes *botnets* y muestras de tráfico legítimo correspondientes a sesiones de tráfico real. Utilizando este conjunto de datos, queremos responder a las siguientes preguntas de investigación:

> «¿Cuáles son los mejores clasificadores supervisados para la detección de dichos canales de *Command & Control* empleando los atributos de paquetes aislados?»

> «¿Cuáles son las posibilidades que ofrece el método para la identificación de una *botnet* en medio de un *bosque* de tráfico legítimo y malicioso?»

> «¿Con qué eficacia se pueden utilizar muestras genéricas de tráfico de control de *botnet* para identificar tráfico malicioso?»

> «¿Cuál es el tamaño óptimo de secuencias de *n-paquetes* para la detección de los canales de *Command & Control* de una *botnet* dada utilizando aprendizaje supervisado?»

Las primera cuestión se refiere a la utilización de atributos aislados como características representativas del estado de una conexión de *botnet* determinada lo que se describe en el apartado 6.3.1.1 de la subsección 6.3.1. Asimismo, en esa misma subsección, se analizan en el apartado 6.3.1.2 las capacidades del modelo de representación para identificar el tráfico buscado en las dos siguientes cuestiones respectivamente. La última de las preguntas de investigación pone sobre la mesa los experimentos desarrollados en profundidad en la subsección 6.3.2, tanto para identificar tráfico perteneciente a familias de *botnets* —apartado 6.3.2.1—, como para relativas a tráfico genérico de *botnet*—apartado 6.3.2.2—.
La evaluación de cada uno de los métodos se va a realizar en base a diferentes parámetros habitualmenter utilizados en la evaluación del rendimiento de los métodos de clasificación [Pow07].

- ***True Positive Ratio* (TPR), tasa de verdaderos positivos o sensibilidad —en inglés, *sensitivity* y en inglés también *recall*—**. Conceptualmente, en *text mining*, se identifica como la probabilidad de que un documento elegido aleatoriamente sea verdaderamente relevante en la búsqueda. En nuestro caso, se puede definir como la tasa de paquetes maliciosos detectados correctamente de entre todos los analizados por el clasificador. Se calcula —véase la ecuación 6.12— dividiendo el número de paquetes maliciosos correctamente clasificados (TP) entre el total de muestras maliciosas extraídas, ya sean *true positives* o *false negatives* ($TP + FN$):

$$TPR = \frac{TP}{TP + FN} \tag{6.12}$$

- ***Exactitud* ($Acc.$, del inglés *Accuracy*)**. La exactitud $Acc.$ —véase la ecuación 6.13— se calcula dividiendo en este caso el total de aciertos entre el número de instancias que componen la totalidad del conjunto de datos.

$$Acc. = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.13}$$

- ***Possitive Predictive Value* (PPV)**. El PPV —en inglés, también *precision*, concepto que no hay que confundir con *accuracy*— es un valor que presenta la posibilidad de que un resultado positivo refleje la condición que ha sido testeada, es decir, de que una muestra etiquetada como positiva sea efectivamente un verdadero positivo. Se define ma-

temáticamente en la ecuación 6.14 tal y como sigue:

$$PPV = \frac{TP}{TP + FP} \tag{6.14}$$

En donde el *False Positive Ratio* (FPR) es la tasa de falsos positivos o errores de tipo I. Estos errores —también llamados de tipo alfa (α)— son aquellos que comete un investigador al no aceptar la hipótesis nula H_0 cuando esta es verdadera para la población objeto de estudio. Por tanto, en nuestro caso, en el que la hipótesis nula H_0 se corresponde con que un paquete sea legítimo, la tasa de falsos positivos o FPR se puede definir como el número de paquetes legítimos incorrectamente clasificados como procedentes de una *botnet*. Se calcula —véase la ecuación 6.15— dividiendo el número de muestras de paquetes correspondientes a tráfico de *botnet* cuya clasificación se erró (FP) entre el número total de muestras ($FP + TN$).

$$FPR = \frac{FP}{FP + TN} \tag{6.15}$$

Relacionada con el FPR, se denomina *especificidad* —también tasa de verdaderos negativos o TNR— a la probabilidad de clasificar correctamente a un individuo cuyo estado real sea el definido como negativo y también se puede expresar como en la ecuación 6.16:

$$TNR = 1 - FPR \tag{6.16}$$

- ***F-measure*** **(F)**. La *F-measure* o F —véase la ecuación 6.17— es una métrica típica de *Information Retrieval* (IR) para evaluar el resultado de una búsqueda que determina hasta qué punto los grupos obtenidos se asemejan a los que se hubieran logrado con una categorización manual. Esta medida se puede definir como:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} = 2 \cdot \frac{PPV \cdot TPR}{PPV + TPR} \tag{6.17}$$

De este modo si sustituimos en F los valores de PPV y TPR podremos

simplificar la fórmula como sigue:

$$
\begin{aligned}
F &= 2 \cdot \frac{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}} = \\
&= 2 \cdot \frac{\frac{TP^2}{(TP+FP)\cdot(TP+FN)}}{\frac{TP\cdot(TP+FN)+TP\cdot(TP+FP)}{(TP+FP)\cdot(TP+FN)}} = \\
&= 2 \cdot \frac{TP^{\cancel{2}} \cdot \cancel{(TP+FP)} \cdot \cancel{(TP+FN)}}{\cancel{(TP+FP)} \cdot \cancel{(TP+FN)} \cdot \cancel{TP} \cdot (TP+FN+TP+FP)} = \\
&= \frac{2TP}{2TP+FN+FP}
\end{aligned}
\tag{6.18}
$$

- ***Area Under ROC Curve*** **(**$AURC$ **o** AUC**)**. El significado del área por debajo de la curva ROC es el establecimiento de la relación entre los falsos negativos y los falsos positivos [SKM09] y se obtiene representando, para cada posible elección de valores de corte la TPR en el eje de ordenadas y la FPR en el eje de abscisas. Empleada originalmente por el ejército estadounidense en investigaciones tras el ataque de Pearl Harbor de 1941 para detectar correctamente las firmas de la señal de rádar de las aeronaves japonesas [GS+66], se ha venido utilizando desde finales de siglo XX como métrica de comparación y evaluación de diferentes algoritmos de clasificación [Spa]. Se suele usar para generar estadísticas que representan el rendimiento —o la efectividad, en un sentido más amplio— de un clasificador en un sistema binario[1] como herramienta para seleccionar modelos posiblemente óptimos y descartar aquellos subóptimos. Aunque no proporciona información sobre el buen comportamiento del modelo, el AUC ayuda a determinar la validez de una distribución de datos a lo largo de una serie dada de condiciones predictorias [LJVR07].

6.2.1 Aplicación de la validación cruzada

Una de las formas de evaluar el comportamiento de los clasificadores es utilizar técnicas de *cross validation* o validación cruzada. Estas técnicas dividen

[1] En el caso de aquellos experimentos cuyo etiquetado no fuera binario, se deberá usar el término *weighted* AUC que aglutina los valores AUC de cada uno de los subconjuntos binarios de entrenamiento en que se divida el problema. Entenderemos por etiquetado no binario aquel experimento cuyo conjunto de datos puede ser etiquetado en más de dos clases.

un conjunto de datos en k conjuntos de entrenamiento diferentes formados por el $[100-(k-1)\cdot\frac{100}{k}]\%$ de las muestras. Tras el entrenamiento, el modelo será validado a su vez con el $(\frac{100}{k})\%$ restante de las muestras. El error del sistema E se puede definir como sigue:

$$E = \frac{1}{k}\cdot\sum_{i=1}^{k} E_i \tag{6.19}$$

siendo E_i el error propio de cada una de las iteraciones. El valor de k utilizado en Weka para esta experimentación es de $k = 10$, por lo que cada conjunto de datos utilizado se divide en diez conjuntos de entrenamiento y de prueba: los primeros estarán formados por el 90 % de todas las representaciones mientras que los segundos, los utilizados para analizar las capacidades del modelo, por el 10 % restante.

6.2.2 *Resampling* y balanceo de datos de entrenamiento

Como ya se ha sugerido anteriormente, la cantidad de paquetes de las muestras puede oscilar notablemente. En el caso de los experimentos realizados, la cantidad de paquetes correspondientes a tráfico benigno en comparación con el tráfico malicioso generado en cada uno de los *hosts* ha sido desde 4:1 hasta 80:1. Sin embargo, la utilización de datos de entrenamiento tan desbalanceados puede introducir sesgos en la clasificación si no se tienen en cuenta [BJZY12]. Por ello, se ha optado por utilizar técnicas de *resampling* que reajusten la cantidad de paquetes de cada clase como paso previo a la aplicación de los diferentes algoritmos de clasificación.

El método seleccionado para la ejecución es el que se muestra en la figura 6.5 implementado en Weka como *Spread Subsample*. En este caso, el algoritmo optará por la reducción del conjunto de datos de entrenamiento a una cantidad de muestras que contenga, por clase, el menor número de instancias de entre las clases candidatas: es decir, en caso de que en la muestra de trafico original se hayan recogido más muestras legítimas que maliciosas ($b >= m$) se procederá a seleccionar solamente un número m de muestras benignas para entrenar los modelos.

6.3 Validación empírica

En esta sección se procederá a la recogida de datos de los experimentos llevados a cabo para esta tesis doctoral. Como ya se ha explicado anterior-

entrada: Conjunto M de m representaciones maliciosas $\{r_1, r_2, \cdots, r_m\}$
y conjunto B de b representaciones legítimas $\{r_1, r_2, \cdots, r_b\}$.
$resto \leftarrow |(m-b)|$;
if $m > b$ **then**
  // Como $m > b$: filtramos representaciones de M
  $M' \leftarrow FiltrarRepresentaciones(M, resto)$;
  // Se ha actualizado el conjunto de datos de modo que M' pase a tener solo b representaciones maliciosas $\{r_1, r_2, \cdots, r_b\}$.
  $Entrenarmodelo(M', B)$;
end
else
  // Como $b >= m$: filtramos representaciones de B
  $B' \leftarrow FiltrarRepresentaciones(B, resto)$;
  // Se ha actualizado el conjunto de datos de modo que B'pase a tener solo m representaciones legítimas $\{r_1, r_2, \cdots, r_m\}$.
  $Entrenarmodelo(M, B')$;
end

Figura 6.5: Aplicación de la métrica de balanceo.

mente, el proceso se va a dividir en dos apartados: los relativos al análisis de los resultados para representaciones de paquetes individuales y los relativos a aquellas formadas por representaciones que integran los atributos de una secuencia de paquetes. En cualquier caso, para ambas representaciones se van a agrupar los resultados en base a las dos técnicas de etiquetado diferentes ya expuestas: tráfico de una sola *botnet* y tráfico genérico.

6.3.1 Aplicación del modelo a conexiones representadas por sus paquetes

En este apartado se recogerán los experimentos relativos a representaciones comopuestas únicamente por secuencias de $n = 1$ paquete. Se ampliarán a continuación los resultados de los más de un centenar de experimentos de clasificación llevados a cabo para validar este modelo de representación concreto.

Los algoritmos de aprendizaje utilizados para estos apartados son los

siguientes:

- **Support Vector Machines**. Se ha empleado el algoritmo *Sequential Minimal Optimisation* (SMO) [Pla99] utilizando para los experimentos diferentes tipos de kernel *kernel polinomial* [AW99], un *kernel polinomial normalizado* [AW99], un *kernel Pearson VII* [UMB06] y un *kernel Radial Basis Function* (RBF) [AW99].

- **Árboles de decisión**. Se han utilizado *Random Forest* [Bre01a] y la implementación del algoritmo C4.5 [Qui93] realizada por equipo de WEKA [Gar], J48.

- **Redes bayesianas**. Se han usado diferentes algoritmos de aprendizaje estructural: K2 [CH91], *Hill Climbing* [RN03] y *Tree Augmented Naïve* (TAN) [GGP+97], además de testear la efecicacia de un clasificador de *Bayes ingenuo* o *Naïve Bayes* [Lew98].

- **Bagging**. Se ha utilizado la implementación del método Bagging en Weka con el algoritmo de aprendizaje rápido REPTree utilizado para evaluar la actuación de determinadas individuos en plataformas de juego *online* [SHS].

- **Perceptrones**. Se han utilizado diferentes tipos de perceptrones: los perceptrones multicapa (MLP) [GD98] y el perceptrón por votación [FS99].

- **K-Nearest Neighbour (KNN)**. Hemos realizado experimentos en el rango desde $k = 1$ hasta $k = 10$ para entrenar los algoritmos, con el fin de corroborar la hipótesis ya planteada por los autores [BGdlPUP+b] de que la efectividad de estos mismos no oscilaba en exceso en función del número de vecinos a tomar en consideración para dar respuesta a este tipo de problemas.

Para evaluar las capacidades de cada clasificador, hemos utilizado las métricas que hemos descrito con anterioridad para los apartados 6.3.1.1 y 6.3.1.2: exactitud (*Acc.*), tasa de verdaderos positivos (TPR), valor positivo de predicción (PPV), *f-measure* y área por debajo de la curva ROC (AUC).

En primer lugar, se han extraído las características atómicas individuales que definen cada uno de los paquetes de una determinada conexión. En total, y en función de la *botnet* analizada, éstas oscilaban entre las 23 y las 30 características. En otros trabajos en los que ha colaborado el autor

[SBN+10b, SSL+, SDB+, SBUPB13] la cantidad de atributos era sensiblemente más numerosa y se hacía necesario reducir al máximo los esfuerzos realizados empleando técnicas como *Information Gain* [Föl73] para facilitar las tareas de clasificación. Sin embargo, no ha sido necesario seleccionar aquellos de más relevancia dado que el coste computacional de considerarlos todos o una muestra aún más reducida de ellos se puede considerar marginal desde el punto de vista experimental.

6.3.1.1 **Capacidad del modelo para detectar la presencia de tráfico procedente de una única *botnet* determinada**

En la tabla 6.1 se recogen los resultados obtenidos para muestras de tráfico procedentes de la familia Flu. Estos resultados amplían los presentados por el autor en [BGdlPUP+a, BGdlPSB13] empleando más algoritmos.

Como se puede observar el mejor clasificador en términos de exactitud final es el clasificador *Random Forest*, con la capacidad para etiquetar correctamente el 82,53 % del total de las muestras. Sin embargo, cabe destacar que los resultados en términos de exactitud son, en general, bastante homogéneos: tanto la implementación del algoritmo C4.5 en Weka, como el método *Bagging* o las diferentes variantes del algoritmo KNN obtienen valores similares superiores al 80 % de las clasificaciones correctas. Con respecto al algoritmo KNN parece confirmarse para Flu la hipótesis de que el incremento de un mayor número de vecinos a la hora de tomar la decisión de clasificar la muestra en un grupo u otro no aporta una mejora perceptible: muy al contrario, se aprecia una ligera pero continuada tendencia a la baja.

En cuanto a la tasa de verdaderos positivos, en general las tasas de detección son bastante elevadas salvo en el caso de *Bayes Net* K2, que se sitúa en un 0,6. Si bien es cierto que el resultado de este algoritmo puede parecer pobre, es necesario también ponerlo en su contexto comparándolo con el PPV, que relaciona la tasa de verdaderos positivos —o TPR— con la tasa de falsos positivos —o FPR—. En este caso, un valor tan próximo a 1 en dicho apartado establece un mayor índice de fiabilidad de las clasificaciones positivas, mientras que valores más próximos a 0,5 —como es el caso de Naïve Bayes— indicarían la escasa fiabilidad de que una muestra identificada como maliciosa sea verdaderamente maliciosa. En este caso, destaca la eficacia del clasificador *Bayes Net* usando como algoritmo de búsqueda tanto K2 como *Hill Climbing*.

Con respecto a los valores realativos a la *f-measure* es nuevamente *Bayes*

Tabla 6.1: Resultados de detección de tráfico procedente de la *botnet* Flu empleando los atributos de paquetes individuales.

Clasificador	***Acc.* (%)**	**TPR**	**PPV**	***F-measure***	**AURC**
SMO Polik.	74,30	0,94	0,68	0,78	0,75
SMO Polik. N.	78,22	0,94	0,71	0,81	0,78
SMO RBF	70,23	0,95	0,64	0,76	0,71
SMO Pear-VII	81,91	1,00	0,74	0,84	0,82
Bagging REPT	82,17	1,00	0,74	0,85	0,93
Naive Bayes	68,12	0,95	0,62	0,75	0,79
Bayes Net K2	79,03	0,60	0,97	0,74	0,92
Bayes Net Hill	78,92	0,61	0,95	0,74	0,92
Bayes Net TAN	82,31	1,00	0,74	0,85	0,93
C4.5	82,14	1,00	0,74	0,85	0,85
Rand. Forest	82,48	1,00	0,75	0,85	0,93
MLP	81,59	0,94	0,76	0,83	0,92
Voted Per.	66,16	0,75	0,64	0,68	0,66
KNN k=1	82,40	1,00	0,74	0,85	0,93
KNN k=2	82,28	1,00	0,74	0,85	0,93
KNN k=3	82,20	1,00	0,74	0,85	0,93
KNN k=4	82,02	1,00	0,74	0,85	0,93
KNN k=5	81,82	1,00	0,74	0,84	0,92
KNN k=6	81,69	1,00	0,74	0,84	0,92
KNN k=7	81,70	1,00	0,74	0,84	0,92
KNN k=8	81,67	1,00	0,74	0,84	0,92
KNN k=9	81,67	1,00	0,74	0,84	0,92
KNN k=10	81,67	1,00	0,74	0,84	0,92

Net K2 el que obtiene una puntuación más elevada que contrasta de forma manifiesta con el valor obtenido por el clasificador bayesiano ingenuo, de apenas 0,57. En cuanto al valor de la curva ROC, que determina el ajuste y la idoneidad de las muestras representadas para determinar la bondad de dicha selección, los valores más altos se sitúan por encima de 0,90, destacando *Bagging*, *Random Forest* y KNN con valores de AUC de hasta 0,93.

En la tabla 6.2 se presentan los resultados obtenidos por cada uno de los algoritmos de clasificación para el conjunto de datos de Prablinha. Como se puede observar, el porcentaje de acierto en la clasificación es, en general, ligeramente inferior a los resultados obtenidos con la muestra de Flu. Sin

Tabla 6.2: Resultados de detección de tráfico procedente de la *botnet* Prablinha empleando los atributos de paquetes individuales.

Clasificador	***Acc.* (%)**	**TPR**	**PPV**	***F-measure***	**AURC**
SMO Polik.	67,32	0,83	0,63	0,72	0,67
SMO Polik. N.	67,50	0,86	0,63	0,73	0,67
SMO RBF	65,59	0,86	0,61	0,72	0,65
SMO Pear-VII	70,92	0,78	0,68	0,73	0,71
Bagging REPT	79,26	0,94	0,72	0,82	0,89
Naive Bayes	57,81	0,28	0,72	0,40	0,64
BayesNet K2	73,33	0,97	0,66	0,79	0,83
Bayes Net Hill	73,44	0,97	0,66	0,79	0,83
Bayes Net TAN	74,09	0,95	0,67	0,79	0,84
C4.5	79,08	0,94	0,72	0,82	0,88
Rand. Forest	79,73	0,94	0,72	0,83	0,89
MLP	66,63	0,43	0,84	0,55	0,79
Voted Per.	59,47	0,68	0,58	0,63	0,60
KNN k=1	79,59	0,94	0,72	0,83	0,89
KNN k=2	79,37	0,95	0,73	0,82	0,89
KNN k=3	79,16	0,94	0,72	0,82	0,89
KNN k=4	78,51	0,94	0,71	0,82	0,89
KNN k=5	77,66	0,93	0,71	0,81	0,88
KNN k=6	77,37	0,93	0,70	0,81	0,87
KNN k=7	77,24	0,92	0,71	0,81	0,87
KNN k=8	76,88	0,92	0,70	0,80	0,87
KNN k=9	76,39	0,92	0,70	0,80	0,87
KNN k=10	75,97	0,91	0,69	0,79	0,86

embargo, aquí sí que se pueden observar diferencias notables en función del tipo de algoritmo de clasificación empleado. Los mejores son, por este orden, *Random Forest*, KNN para $k = 1$, $k = 2$ y $k = 3$, *Bagging* y la implementación en Weka del algoritmo C4.5 (J48). Todos ellos obtienen un porcentaje de acierto en la clasificación muy cercano al 80 % y siempre superior al 79 %. Los peores resultados obtenidos, con diferencia, corresponden al algoritmo *Voted Perceptron* —que clasifica correctamente el 59,47 % de las muestras— y, especialmente, *Naïve Bayes*, que apenas clasificaba correctamente el 57,81 % de las mismas.

En cuanto a la tasa de verdaderos positivos, la más alta de todas las

obtenidas corresponde a la aplicación de los algoritmos de búsqueda K2 y *Hill Climbing* con redes bayesianas, consiguiéndose alcanzar valores de 0,97. Sin embargo, la exactitud de dichos aciertos es más bien limitada en comparación con el resto, ya que estos algoritmos obtienen valores de PPV de apenas un 0,66. Esto quiere decir que, a pesar de identificar correctamente como maliciosas la mayor parte de las muestras maliciosas, también etiquetaban como tales una cantidad elevada de muestras benignas lo que generará una gran cantidad de falsos positivos. Si se comparan ambos criterios, los resultados más positivos los obtiene el algoritmo KNN para $k = 2$ que consigue un valor de TPR de 0,95 y un 0,73 de PPV. Mención aparte merece *Multilayer Perceptron*. Pese a que apenas consigue un valor de 0,43 de TPR obtiene la puntuación más alta en lo que a PPV se refiere: es decir, es el algoritmo que, una vez etiquetado un paquete como positivo, más probabilidades tiene de haber acertado en dicha clasificación.

En cuanto a la *f-measure* destacan en el espectro superior KNN para $k = 1$ y *Random Forest* —con 0,83, los valores más altos de todo el estudio— y, en el inferior, MLP y *Naïve Bayes* con 0,55 y 0,40. En cuanto a la bondad de los conjuntos de datos utilizados, medida esta magnitud empleando los valores por debajo de la curva ROC destacan ambos algoritmos de árboles de decisión, el método *Bagging* y KNN en sus diferentes versiones con valores que alcanzan 0,89.

Merece también un comentario la observación de que la inclusión de más vecinos para el método de clasificación de los k vecinos más próximos no supone, en general, ninguna mejora. De hecho, la tendencia es a la inversa: a más vecinos utilizados resultados —ligeramente— menos fiables.

En la tabla 6.3 se presentan los resultados obtenidos por cada uno de los algoritmos de clasificación entrenados con el conjunto de datos de muestra de Warbot. A diferencia de los resultados anteriores, se pone de manifiesto que el porcentaje de acierto en la clasificación es el superior de todo el estudio realizado, sugiriendo una mayor sencillez de las comunicaciones empleadas. El porcentaje de acierto en la clasificación oscila entre el 79,61 % del peor clasificador (*Naïve Bayes*) y el 87,29 % obtenido por KNN para $k = 1$ y SMO entrenado con un *kernel* polinomial. En cualquier caso, resultados muy positivos.

La tasa de verdaderos positivos TPR pone aún más de manifiesto una mayor sencillez en la clasificación. El resultado más bajo obtenido es 0,75 obtenido por todos los clasificadores a excepción de MLP (0,77) y *Naïve Bayes* (1,00). Estos dos únicos resultados atípicos difieren del resto al obtener *Naïve Bayes* un 0,70 de PPV —en otras palabras, solamente el 70 % de las

Tabla 6.3: Resultados de detección de tráfico procedente de la *botnet* Warbot empleando los atributos de paquetes individuales.

Clasificador	***Acc.* (%)**	**TPR**	**PPV**	***F-measure***	**AURC**
SMO Polik.	87,29	0,75	1,00	0,86	0,88
SMO Polik. N.	87,19	0,75	1,00	0,86	0,87
SMO RBF	87,19	0,75	1,00	0,86	0,87
SMO Pear-VII	87,29	0,75	1,00	0,86	0,88
Bagging REPT	87,19	0,75	1,00	0,86	0,93
Naive Bayes	79,61	1,00	0,70	0,83	0,95
BayesNet K2	86,63	0,75	0,99	0,85	0,95
Bayes Net Hill	86,63	0,75	0,99	0,85	0,95
Bayes Net TAN	87,13	0,75	1,00	0,86	0,95
C4.5	87,19	0,75	1,00	0,86	0,87
Rand. Forest	87,26	0,75	1,00	0,86	0,96
MLP	86,95	0,77	0,97	0,86	0,96
Voted Per.	87,15	0,75	1,00	0,86	0,87
KNN k=1	87,29	0,75	1,00	0,86	0,96
KNN k=2	87,19	0,75	1,00	0,86	0,96
KNN k=3	87,19	0,75	1,00	0,86	0,96
KNN k=4	87,19	0,75	1,00	0,86	0,96
KNN k=5	87,19	0,75	1,00	0,86	0,96
KNN k=6	87,19	0,75	1,00	0,86	0,96
KNN k=7	87,19	0,75	1,00	0,86	0,96
KNN k=8	87,19	0,75	1,00	0,86	0,96
KNN k=9	87,19	0,75	1,00	0,86	0,96
KNN k=10	87,19	0,75	1,00	0,86	0,96

muestras etiquetadas como maliciosas lo son— y MLP un valor de 0,97. Contrastan con los resultados que obtienen el resto de clasificadores que es, en general, de 1,00: es decir, la totalidad de las clasificaciones maliciosas corresponden efectivamente a paquetes procedentes de una *botnet*.

Los valores de *f-measure* son nuevamente muy homogéneos: varían poco entre un máximo de 0,86 y un mínimo de 0,83 —obtenido nuevamente por el clasificador de Bayes ingenuo. La bondad de los conjuntos de datos utilizada es también muy elevada aunque oscila ligeramente más que la *f-measure*. Tanto *Random Forest* como MLP y la totalidad de las variantes de KNN obtienen en este punto un valor de 0,96 —el máximo del estudio en

este caso— por un mínimo de 0,87 obtenido por SMO entrenado con kernel polinomial normalizado y con RBF, la implementación en Weka de C4.5 y *Voted Perceptron*.

La principal conclusión a extraer con respecto a Warbot es que los paquetes de dicha *botnet* son identificados, por su estructura, como sensiblemente diferentes al tráfico real HTTP y la práctica totalidad de los algoritmos puede identificar un número muy significativo de ellos entre tráfico normal. Comentario adicional merece también el hecho de que el algoritmo KNN parece no mostrar variaciones reseñables en cuanto a su eficacia en función del número de vecinos utilizados.

6.3.1.2 **Capacidad del modelo para detectar de forma genérica el tráfico de control procedente de varias *botnets***

De cualquier manera, los resultados ya comentados consiguen altos índices de detección para el tráfico proveniente de *botnets* con canales de *Command & Control* basado en HTTP y HTTPS muy concretos. El siguiente paso es comprobar la eficacia del sistema para la detección del tráfico procedente de otras redes de cara a diseñar un sistema capaz de identificar de forma genérica el tráfico de control generado incluso por muestras maliciosas de familias de *malware* lo más variadas posible.

Así, en la tabla 6.4 se presentan los resultados para el experimento consistente en el etiquetado de tráfico como genérico o malicioso únicamente, sin hacer distinciones de la familia a la que pertenecen. El mejor resultado en términos de precisión lo obtiene el clasificador KNN para $k = 1$, disminuyendo esta paulatinamente cada vez que se incluye un vecino más en la clasificación. Sin embargo, otros algoritmos como C4.5, *Bagging* o las redes bayesianas entrenadas con TAN también han obtenido resultados muy similares en torno al 80 %. El peor resultado vuelve a obtenerlo el perceptrón por votación, con apenas un 55,10 % de las muestras bien clasificadas, puesto que ha terminado ocupando independientemente de cuál haya sido la métrica de comparación utilizada.

En lo que a la tasa de verdaderos positivos se refiere, los resultados son en general también muy elevados, alcanzando cotas de más de 0,90 para una gran variedad de clasificadores con una fiabilidad de clasificaciones correctas en torno al setenta por ciento (0,70). De cualquier manera, conviene resaltar la capacidad de los clasificadores bayesianos que emplean tanto K2 como *Hill Climbing* como algoritmos de búsqueda: aunque ambos apenas pueden identificar el 60 % de las muestras etiquetadas como maliciosas,

Tabla 6.4: Resultados generales de detección de tráfico procedente de la *botnets* analizadas empleando los atributos de paquetes individuales

Modelo	***Acc.* (%)**	**TPR**	**PPV**	***F-measure***	**AURC**
SMO Polik.	74,22	0,92	0,67	0,79	0,74
SMO Polik. N.	75,47	0,91	0,68	0,79	0,75
SMO RBF	69,58	0,93	0,63	0,76	0,69
SMO Pear-VII	77,53	0,93	0,70	0,81	0,77
Bagging REPT	80,46	0,98	0,72	0,84	0,91
Naive Bayes	68,21	0,54	0,77	0,63	0,78
Bayes Net K2	77,08	0,59	0,94	0,73	0,90
Bayes Net Hill	77,01	0,59	0,94	0,73	0,90
Bayes Net TAN	79,94	0,97	0,71	0,83	0,91
C4.5	80,47	0,98	0,72	0,84	0,89
Rand. Forest	80,13	0,93	0,74	0,82	0,92
Voted Per.	55,10	0,55	0,55	0,56	0,55
KNN k=1	80,54	0,98	0,72	0,84	0,92
KNN k=2	80,18	0,98	0,72	0,84	0,91
KNN k=3	80,27	0,97	0,72	0,84	0,91
KNN k=4	79,98	0,97	0,71	0,83	0,91
KNN k=5	79,86	0,97	0,71	0,83	0,91
KNN k=6	79,70	0,97	0,71	0,83	0,91
KNN k=7	79,59	0,97	0,71	0,83	0,91
KNN k=8	79,46	0,97	0,71	0,83	0,91
KNN k=9	79,35	0,97	0,71	0,83	0,91
KNN k=10	79,27	0,97	0,71	0,83	0,91

el 94% de las así señaladas serán efectivamente muestras procedentes del canal de *Command & Control* de una *botnet*.

La *f-measure* presenta resultados más variados. Existe una variación perceptible que oscila entre el 0,56 —de *Voted Perceptron*— y el 0,84 de *Bagging*, C4.5 o KNN, que son los que obtienen unos valores más altos en este apartado. El valor del área por debajo de la curva ROC más alto obtenido es de 0,92 para *Random Forest* y KNN, manteniéndose constante en general para todos los clasificadores a excepción de las máquinas de soporte vectorial —que se sitúan entre un 0,7 y un 0,8— y del ya mencionado *Voted Perceptron*.

6.3.2 Aplicación del modelo a conexiones representadas por secuencias de *n-paquetes*

En este apartado se recogerán los experimentos relativos a representaciones compuestas por los atributos extraídos de secuencias de paquetes. Se detallan a continuación los resultados obtenidos de estos experimentos comparando a su vez la evolución de estos resultados conforme se va ampliando la longitud de las secuencias y profundizando en los resultados más allá de lo ya introducido en otras publicaciones del autor [BGdlPB].

6.3.2.1 Capacidad del modelo para detectar la presencia de tráfico procedente de una única *botnet* determinada

En este apartado se va a proceder a analizar la evolución de las diferentes métricas de evaluación según se va ampliando el número de paquetes que se incluyen en cada representación de las muestras de tráfico de cada una de las *botnets* analizadas en el conjunto de datos. Se analizarán, por este orden, la evolución del porcentaje de acierto en la clasificación —figura A.3—, el TPR —figura 6.7—, el PPV —figura 6.8—, el valor de *f-measure* —figura 6.9— y $AURC$ —figura 6.10—. Cada una de estas figuras está subdividida en seis subfiguras representadas en una matriz de tres filas por dos columnas: en donde cada una de las filas dará cobertura a los resultados de una de las *botnets* analizadas mientras que, para una mejor claridad,se presentarán en cada columna los diferentes algoritmos.

- En la de la izquierda los algoritmos SMO entrenados con diferentes *kernels*, el método *Bagging*, los clasificadores bayesianos —incluyendo *Naïve Bayes* y redes bayesianas— y los árboles de decisión C4.5 y *Random Forest* .
- En la de la derecha figuran los clasificadores basados en perceptrones —tanto multicapa como por votación— y los basados en los k vecinos más próximos —$\forall k \in \{1, 2, 4, 5, 6, 7, 8, 9, 10\}$—.

De esta forma, en las figuras A.3 se puede ver la evolución del porcentaje de acierto en la clasificación. En general se puede observar que existe una importante mejora de los diferentes clasificadores cuanto más información contengan las representaciones con las que estos se entrenan, si bien es cierto que la mejora es mucho más acusada cuando se amplía la longitud de las secuencias de $n = 1$ hasta $n = 4$ que a partir de $n = 6$ donde parece producirse un estancamiento general.

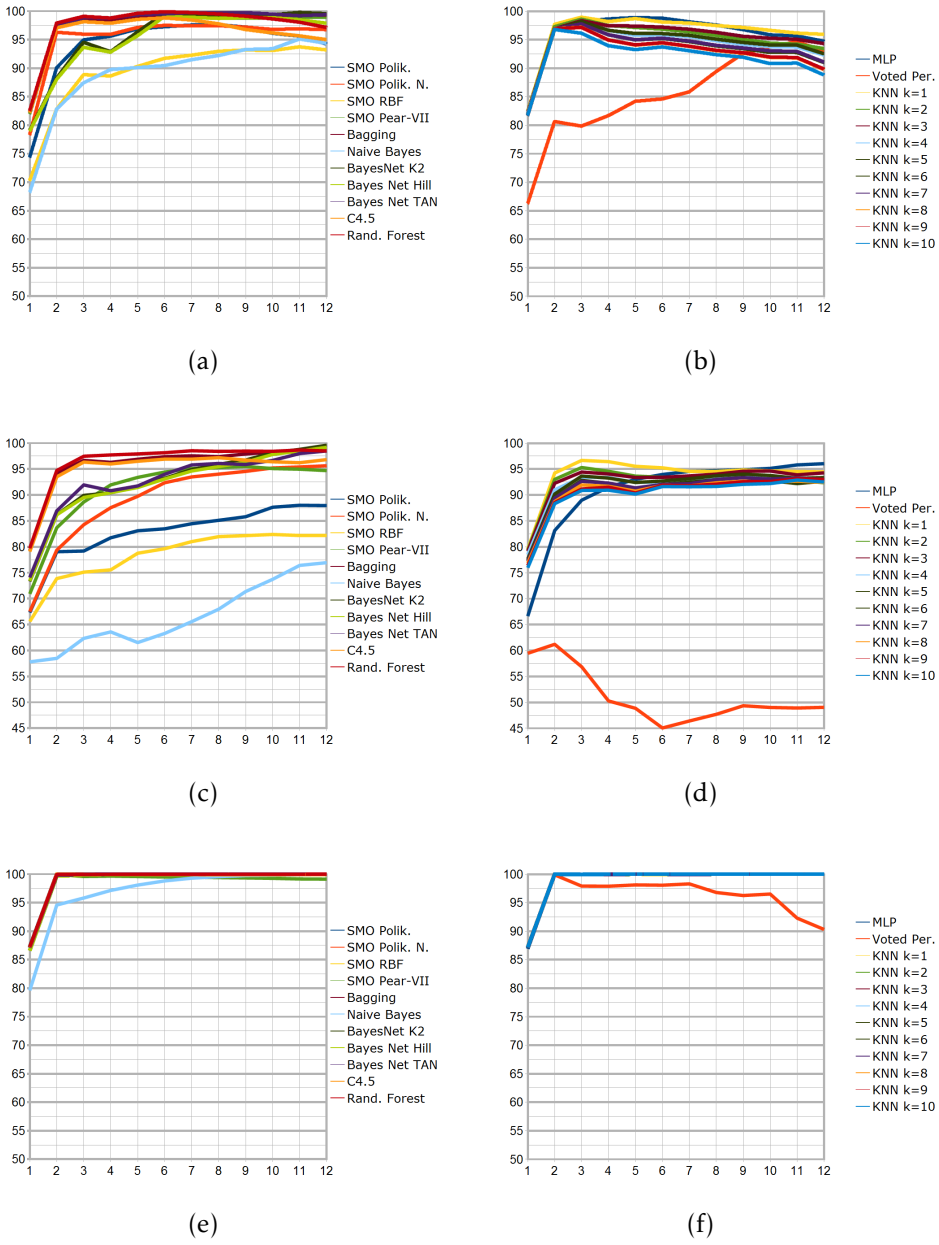


Figura 6.6: Evolución del porcentaje de acierto en la clasificación (*Acc.*) en base a la longitud de las secuencias utilizadas para el etiquetado de tráfico benigno o de *Command & Control* de las *botnets* analizadas: Flu, subfiguras 6.6(a) y 6.6(b); Prablinha, subfiguras 6.6(c) y 6.6(d); y Warbot, subfiguras 6.6(e) y 6.6(f).

Mención aparte merece el caso de KNN, en el que se observa una ligera tendencia a la baja para el conjunto de datos de Flu, pero no así de Prablinha y Warbot. Esta peculiaridad podría estar relacionada con el hecho de que Flu generalmente inicia conexiones a puertos diferentes a partir de los doce a quince paquetes, provocando que el número de muestras del conjunto de datos de entrenamiento no sea suficiente como para realizar clasificaciones correctas. Como ya se comentaba en la sección 6.3.1.1, parece haber una gran unanimidad entre los clasificadores a la hora de etiquetar las muestras de Warbot, sugiriendo una gran homogeneidad de las peticiones efectuadas y diferencias perceptibles con el tráfico legítimo.

Como notas negativas, conviene puntualizar los malos resultados de *Naïve Bayes* y las redes bayesianas entrenadas con RBF que son en general los peores clasificadores en la mayor parte de las métricas utilizadas. Conviene destacar también la escasa fiabilidad achacable a *Voted Perceptron* que, en el caso de Prablinha por ejemplo, obtiene unos resultados particularmente pobres de en torno al 50 %.

En las figuras 6.7 se muestra la evolución de la tasa de verdaderos positivos (TPR) para la detección de tráfico de *Command & Control* en función del número de paquetes incluidos en la representación. Los resultados en este punto difieren mucho más que en el apartado anterior si se exceptúan los relativos a Warbot que se mantienen constantes y homogéneos para la práctica totalidad de los clasificadores empleados tal y como se puede comprobar en las subfiguras 6.7(e) y 6.7(f). Este punto consolida la idea de que las representaciones de dicha *botnet* son mucho más fáciles de detectar que el resto.

En lo que respecta a Flu —subfiguras 6.7(b) y 6.7(b)—se observan en general altos niveles de detección con algunas excepciones: redes bayesianas entrenadas con los algoritmos K2 y *Hill Climbing* y, muy especialmente *Voted Perceptron*. Por otro lado, se observa una ligera tendencia a la baja —más acusada a partir de longitudes de representación superiores a $n = 7$ paquetes— en las detecciones en algoritmos como *Naïve Bayes* o las máquinas de soporte vectorial entrenadas con *Pearson-VII*. Los algoritmos de clasificación en función de los k vecinos más próximos obtienen siempre tasas superiores al 0,95.

Los resultados de Prablinha —subfiguras 6.7(c) y 6.7(d)—- son algo más variados. Por un lado, se pone de manifiesto la escasa capacidad de *Naïve Bayes* para detectar muestras maliciosas, aunque es cierto que esta va creciendo conforme aumenta la longitud de las secuencias. A pesar de que otros algoritmos tampoco se muestran especialmente capaces –.como es el

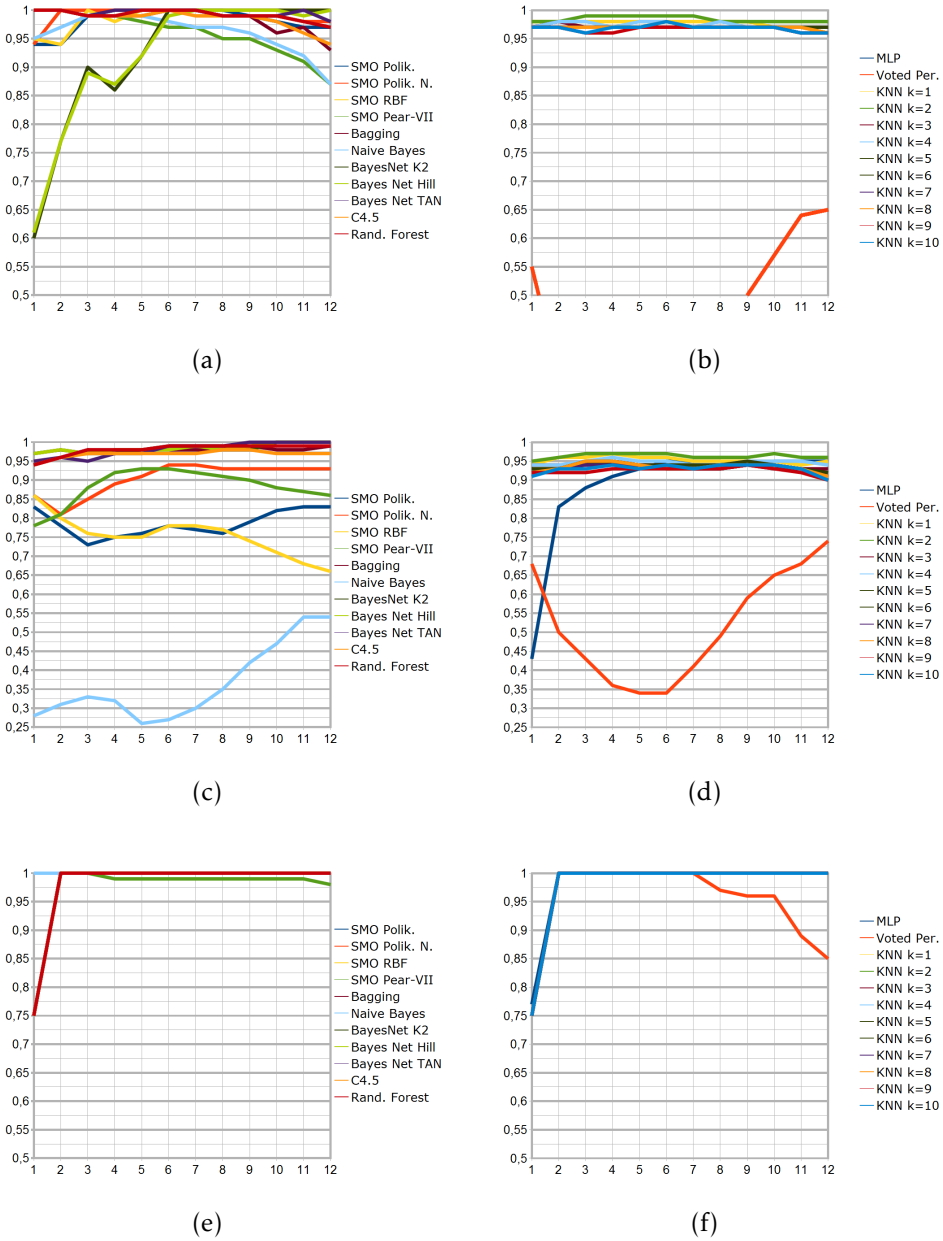


Figura 6.7: Evolución de los índices de detección (TPR) en función de la longitud de las secuencias utilizadas para el etiquetado de tráfico benigno o de *Command & Control* de las *botnets* analizadas: Flu, subfiguras 6.7(a) y 6.7(b); Prablinha, subfiguras 6.7(c) y 6.7(d); y Warbot, subfiguras 6.7(e) y 6.7(f).

caso de SMO con un algoritmo de búsqueda RBF o con *kernel* polinomial— sí que es cierto que existen otros, como C4.5, *Bagging*, *Random Forest* o las redes bayesianas con el algoritmo de búsqueda TAN, que obtienen tasas de detección superiores a 0,95.

Con respecto a los basados en perceptrones, nuevamente es *Voted Perceptron* el que peores resultados obtiene. Cabe destacar que el perceptrón multicapa mejora sensiblemente su tasa de detección de verdaderos positivos desde menos de 0,45 hasta valores superiores a 0,8 con la mera inclusión de un paquete adicional en la representación. En lo que al algoritmo KNN se refiere, los resultados difieren de forma irrelevante obteniéndose los valores más elevados para $k = 2$.

En las figuras 6.8 se muestra la evolución del PPV para la detección de tráfico de *Command & Control* en función del número de paquetes incluidos en la representación. Esta métrica tiene que ser atendida en paralelo a la anterior ya que compara la efectividad de las clasificaciones positivas ponderando aquellas realizadas correctamente con el número de falsos positivos obtenidos en el proceso. Es decir, relativiza los resultados anteriores poniéndolos en consonancia con el porcentaje final de aciertos obtenidos.

En general, se puede observar un crecimiento perceptible conforme va aumentando la longitud de las secuencias empleadas para la clasificación. Dicho crecimiento es más acusado en el rango de secuencias de $n = 1$ hasta $n = 3$ con ligeras correcciones para valores más elevados. Como caso particular dentro de estas gráficas cabe destacar el elevado valor obtenido por el clasificador bayesiano, tanto empleando el algoritmo K2 como *Hill Climbing* si se comparan los resultados, incluso para representaciones de $n = 1$ paquete —véase subfigura 6.8(a)—, con los pobres resultados obtenidos en la tasa de verdaderos positivos o TPR —subfigura 6.7(a)— en la que estos obtenían unos resultados más discretos en comparación con el resto de clasificadores utilizados. Esto pone de manifiesto que incluso para aquellas representaciones más pequeñas, la bondad de las clasificaciones positivas bien realizadas es excepcionalmente alta, lo que tiene sin duda un gran valor para el analista de *malware* dado que podrá fiarse mucho más de aquellos resultados positivos. Nuevamente, prácticamente todos los clasificadores parecen ponerse de acuerdo a la hora de etiquetar las representaciones del conjunto de datos de Warbot —subfiguras 6.8(e) y 6.8(f)—.

En las figuras 6.9 se muestra la evolución de los valores de la *f-measure* para la detección de tráfico de *Command & Control* en función del número de paquetes incluidos en la representación. En el caso de la *f-measure* se observa una peculiaridad más o menos compartida por todas las mues-

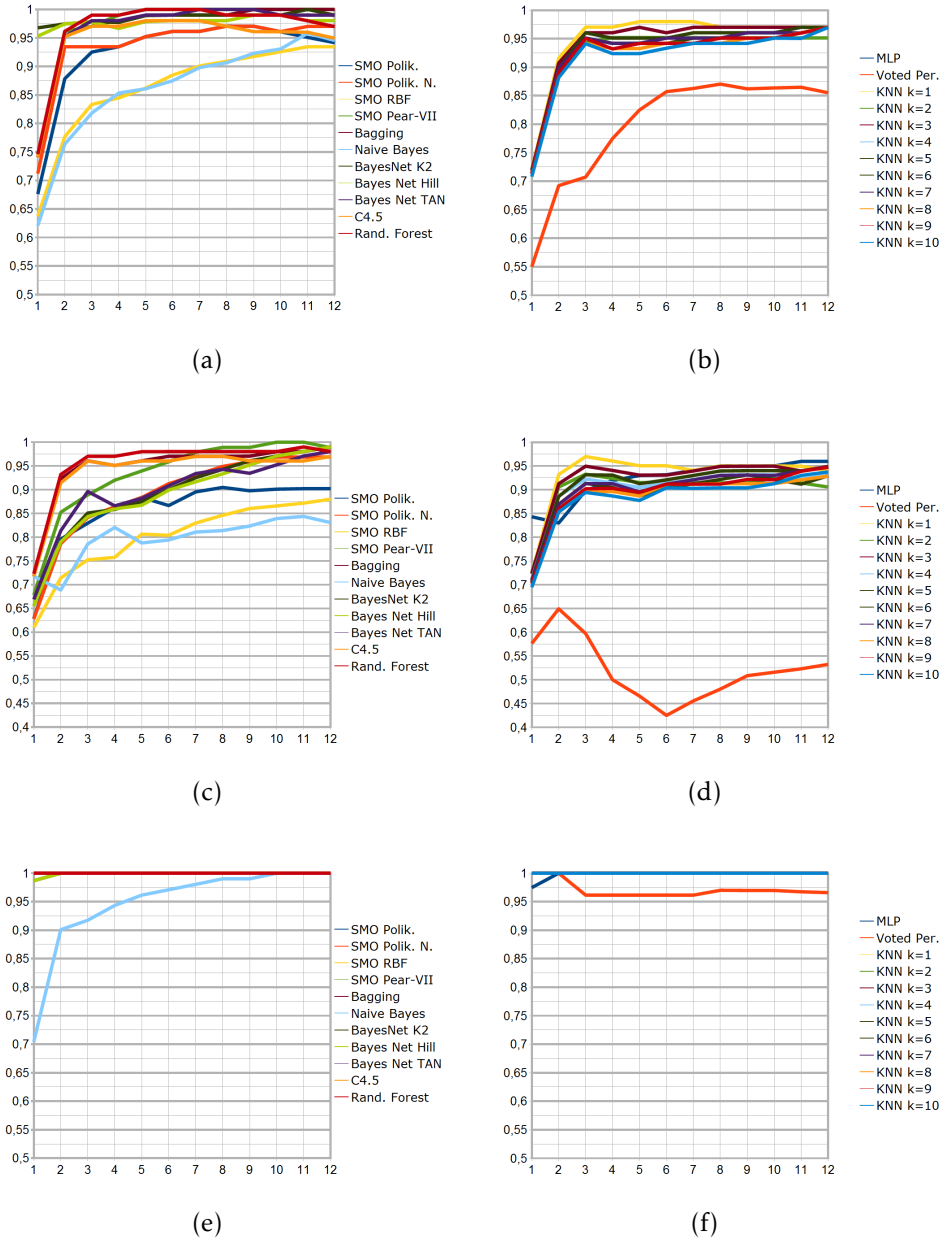


Figura 6.8: Evolución de la efectividad del etiquetado (PPV) en función de la longitud de las secuencias utilizadas para el etiquetado de tráfico benigno o de *Command & Control* de las *botnets* analizadas: Flu, subfiguras 6.8(a) y 6.8(b); Prablinha, subfiguras 6.8(c) y 6.8(d); y Warbot, subfiguras 6.8(e) y 6.8(f).

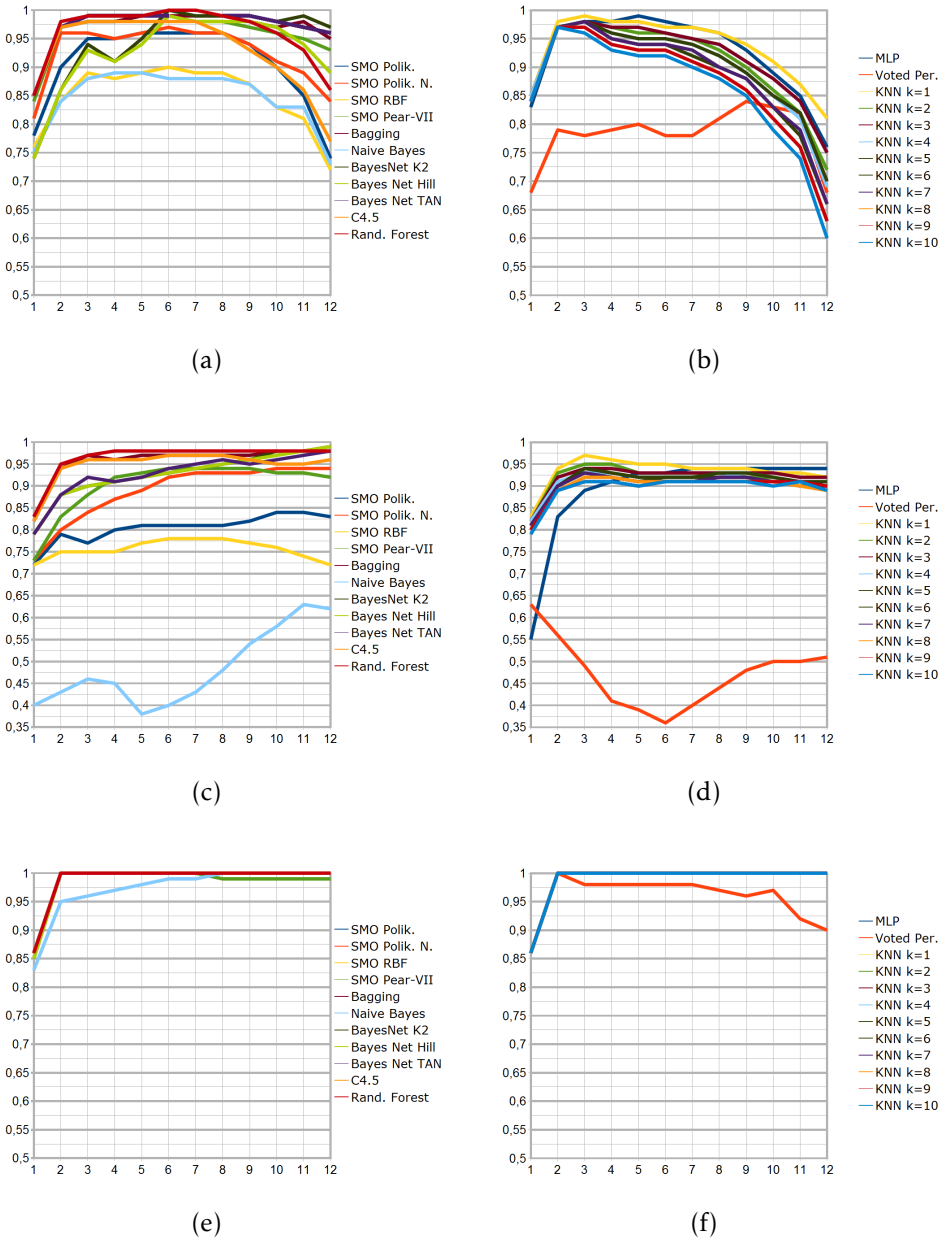


Figura 6.9: Evolución del valor de la *f-measure* según la longitud de las secuencias utilizadas para el etiquetado de tráfico benigno o de *Command & Control* de las *botnets* analizadas: Flu, subfiguras 6.9(a) y 6.9(b); Prablinha, subfiguras 6.9(c) y 6.9(d); y Warbot, subfiguras 6.9(e) y 6.9(f).

tras y es un descenso más o menos acusado en función del clasificador a partir de longitudes de secuencias superiores a $n = 7$ paquetes en el caso de Flu —subfiguras 6.9(a) y 6.9(b)— especialmente acusada en el caso de los clasificadores KNN. Mientras tanto, el conjunto de datos de Prablinha —subfiguras 6.9(c) y 6.9(d)— parece ser capaz de contener dicho decremento aunque la evolución de algunos resultados parece sugerir un bajón para secuencias de paquetes de longitudes superiores. En el caso de Warbot, aunque con algún matiz, para valores superiores a $n = 2$ y $n = 3$ se obtienen valores de 1.

Por último, en las figuras 6.10 se muestra la evolución de los valores del área por debajo de la curva ROC para la detección de tráfico de *Command & Control* en función del número de paquetes incluidos en la representación. Estos valores vienen a representar la evolución de la bondad del conjunto de datos conforme se va ampliando la longitud de las secuencias para cada una de las familias.

En el caso de Warbot —subfiguras 6.10(e) y 6.10(f)— se puede ver como en general, a partir de $n = 2$ los modelos obtienen un valor de la curva ROC de 1,00 indicando la facilidad con la que estos han sido entrenados para dicho conjunto de datos.

Asimismo, en el caso de Flu —subfiguras 6.10(a) y 6.10(b)— los resultados son también muy positivos, con gran cantidad de representaciones obteniendo valores por encima de 0,95 para una gran variedad de clasificadores. Por abajo, la nota negativa la ponen SMO entrenado con RBF, *Naïve Bayes* y *Voted Perceptron* si bien es cierto que, aunque parecen empezar a mostrar síntomas de estancamiento a partir de $n = 12$, ven sus valores en continuo crecimiento conforme se van ampliando el número de paquetes por cada una de las muestras representadas.

Para Prablinha —subfiguras 6.10(c) y 6.10(c)—, se pueden dividir los valores obtenidos en dos grupos. Por un lado, aquellos que evolucionan hasta obtener valoraciones superiores a 0,95 a partir de $n = 3$ y entre los que se encuentran la totalidad de los clasificadores KNN, MLP, los árboles de decisión o *Bagging*. Por otro, el resto de clasificadores entre los que se pueden destacar los casos de *Voted Perceptron*, bayes ingenuo o las máquinas de soporte vectorial entrenadas con *kernel* RBF o *polikernel*. En este sentido, la fase de estancamiento de los valores por debajo de la curva ROC parece sobrevenirse antes que en el caso de Flu.

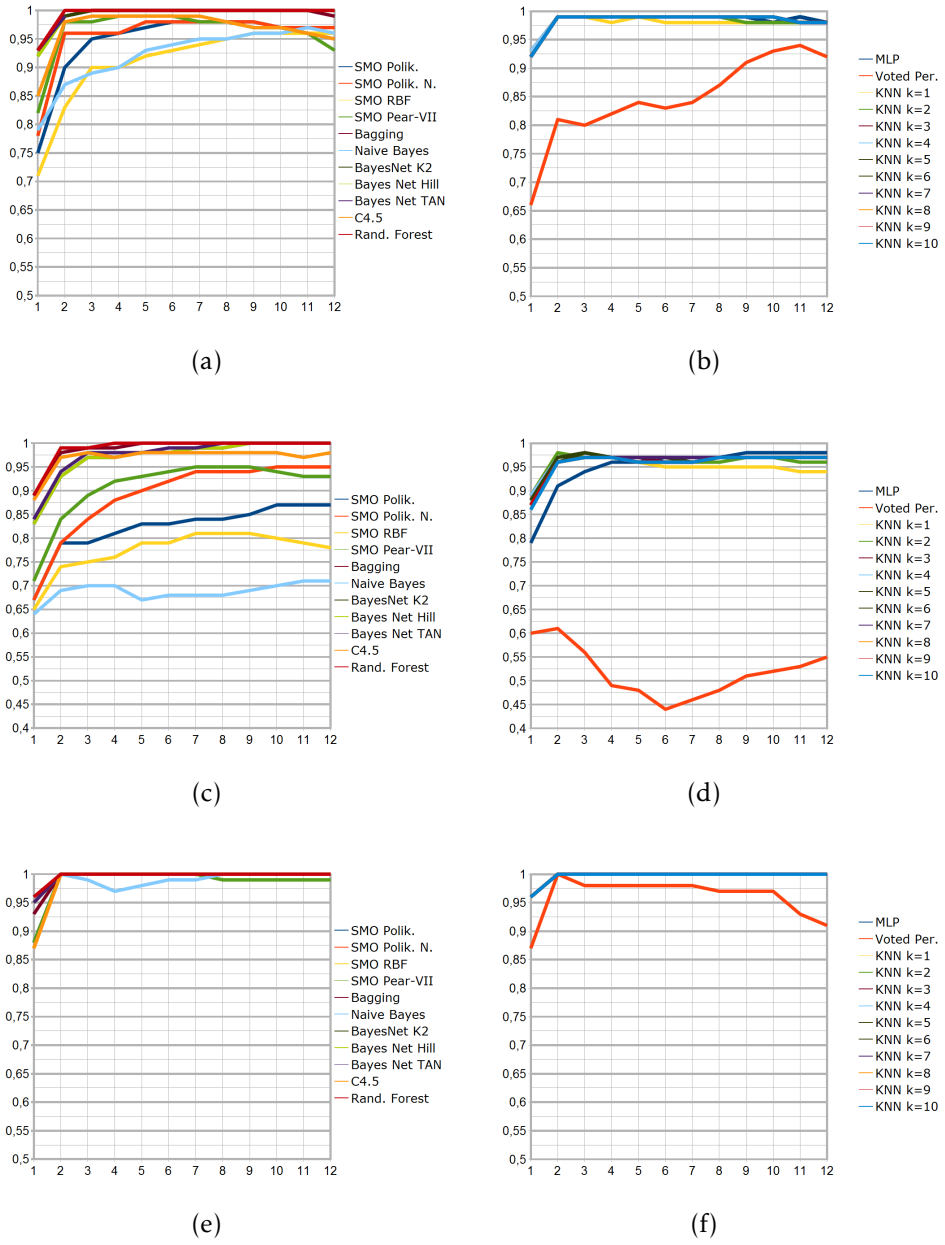
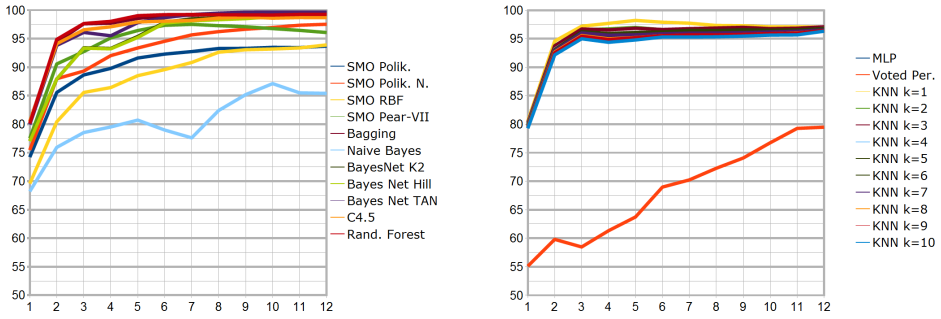


Figura 6.10: Evolución del valor del área por debajo de la curva ROC (*AUC*) según la longitud de las secuencias utilizadas para el etiquetado de tráfico benigno o de *Command & Control* de las *botnets* analizadas: Flu, subfiguras 6.10(a) y 6.10(b); Prablinha, subfiguras 6.10(c) y 6.10(d); y Warbot, subfiguras 6.10(e) y 6.10(f).



(a) (b)

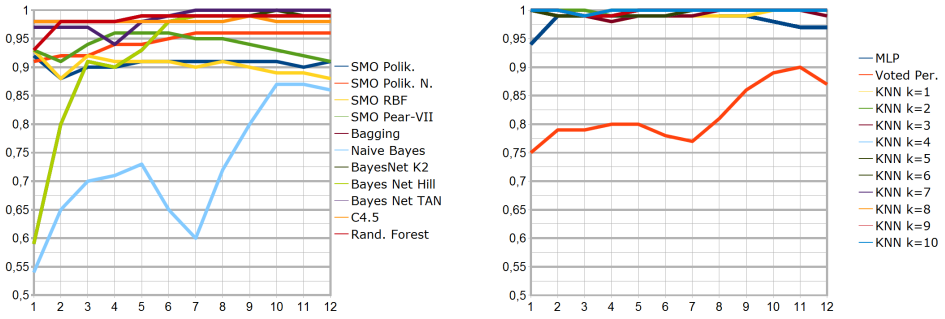
Figura 6.11: Evolución del nivel de acierto en la clasificación (*Acc.*) según la longitud de las secuencias utilizadas para el etiquetado de tráfico genérico benigno o de *Command & Control* de las *botnets* analizadas.

6.3.2.2 Capacidad del modelo para detectar de forma genérica el tráfico de control procedente de varias *botnets*

Vistos los resultados obtenidos en el apartado anterior, de igual forma a como se hiciera en el apartado 6.3.1.2, se va a proceder a analizar cuál es la evolución de este enfoque si se considera la totalidad del tráfico de *botnet* objeto de estudio como tráfico genérico de *Command & Control*. Se analizarán, por este orden, la evolución del porcentaje de acierto en la clasificación —figura 6.11—, el TPR —figura 6.12—, el PPV —figura 6.13—, el valor de *f-measure* —figura 6.14— y AUC —figura 6.15—. Cada una de estas figuras está subdividida a su vez en dos subfiguras representadas en una matriz de una fila por dos columnas, presentando cada columna los resultados para los diferentes algoritmos:

- En la de la izquierda los algoritmos SMO entrenados con los diferentes *kernels*, el método *Bagging*, los clasificadores bayesianos —incluyendo *Naïve Bayes* y redes bayesianas— y los árboles de decisión C4.5 y *Random Forest* .
- En la de la derecha figuran los clasificadores basados en perceptrones —tanto multicapa como por votación— y los basados en los k vecinos más próximos ($\forall k \in \{1, 2, 4, 5, 6, 7, 8, 9, 10\}$).

De esta forma, en la figura 6.11 se puede ver la evolución del porcentaje de acierto en la clasificación para tráfico genérico. Se puede observar que existe una importante mejora de los clasificadores a mayor número de

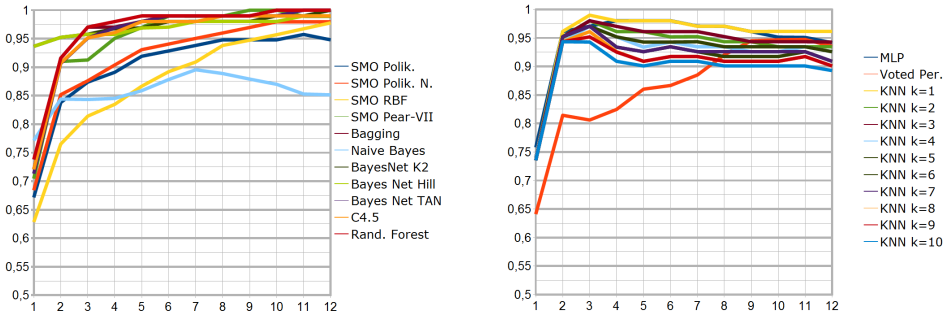


(a) (b)

Figura 6.12: Evolución de los índices de detección (TPR) obtenidos para la detección de muestras maliciosas según la longitud de las secuencias utilizadas para el etiquetado de tráfico genérico benigno o de *Command & Control* de las *botnets* analizadas.

paquetes considerados para cada representación. Como ocurría en la clasificación de tráfico por familias, dicha mejora es más acusada de $n = 1$ hasta $n = 5$ paquetes por representación, a partir de lo cual se produce un estancamiento general e incluso algún pequeño retroceso como ocurre con *SMO Pearson VII*. Nuevamente los clasificadores de *KNN* obtienen resultados muy similares con pocas o ninguna mejora a partir de $n = 4$. Como nota negativa, cabe destacar la escasa eficacia para las representaciones menores de los clasificadores *Voted Perceptron* y *Naïve Bayes* que obtienen los peores resultados de la muestra en todas las longitudes.

En la figura 6.12 se muestra la evolución de la tasa de verdaderos positivos (TPR) para la detección de tráfico de *Command & Control* en función del número de paquetes incluidos en la representación. Los resultados aquí son menos homogéneos que los obtenidos en la figura 6.11 poniendo de manifiesto que algunos clasificadores se comportan mejor para la identificación de muestras maliciosas que benignas o viceversa. Los mejores resultados son obtenidos tanto por *KNN* como por los árboles de decisión (tanto C4.5 como *Random Forest* obtienen valores muy elevados) siempre por encima de 0,975. Los peores resultados los vuelven a obtener *Naïve Bayes* y *Voted Perceptron* con diferencias variables en función de la longitud. Contrasta en este punto el empeoramiento de ambos —más acusado en el caso del clasificador bayesiano ingenuo— para secuencias de $n = 6$ y $n = 7$ paquetes, pese a que ambos recuperaron tasas más altas de detección para secuencias



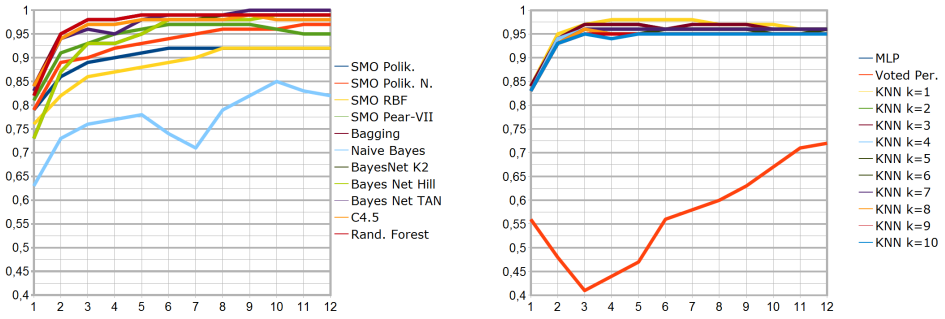
(a) (b)

Figura 6.13: Evolución del valor de exactitud (PPV) según la longitud de las secuencias utilizadas para el etiquetado de tráfico genérico benigno o de *Command & Control* de las *botnets* analizadas.

más largas. Mención especial merecen las redes bayesianas entrenadas con el algoritmo de búsqueda K2 cuyos malos resultados para secuencias cortas contrastan con la amplia mejora mostrada para secuencias más largas.

Nuevamente, los resultados de la figura 6.12 tienen que ser comparados con los que se muestran en la figura 6.13 para que puedan ser puestos en su contexto estos índices de detección. Las gráficas del PPV nos muestran la relación de verdaderos positivos con respecto a los falsos positivos encontrados. Los valores más elevados que obtienen algunos algoritmos —como *Naïve Bayes* y *Voted Perceptron*— que tenían los peores comportamientos en las métricas anteriores se muestran no tan malos como podía parecer en un primer momento, al obtener menos falsos positivos y, por tanto, una relación de $\frac{acierto}{fallo}$ más elevada cuando el clasificador estima que una muestra es un positivo. Los mejores resultados los obtienen una vez más los árboles de decisión liderados en este aspecto por *Random Forest*. En lo que a KNN se refiere, se aprecia una notable reducción del valor de los PPV para secuencias de $n > 3$ conforme se va incrementando la longitud de las representaciones.

En las figuras 6.14 se muestra la evolución de los valores de la *f-measure* para la detección genérica de tráfico de *Command & Control* en función del número de paquetes incluidos en la representación. En el caso de la *f-measure* se observa una peculiaridad más o menos compartida por todas las muestras: un estancamiento de los valores a partir de $n = 7$ con un ligero retroceso en algunos clasificadores como KNN o SMO con *kernel Pearson*



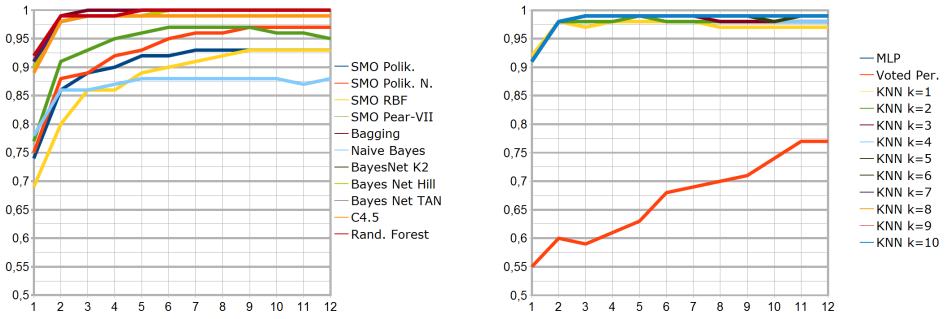
(a) (b)

Figura 6.14: Evolución del valor de la *f-measure* según la longitud de las secuencias utilizadas para el etiquetado de tráfico genérico benigno o de *Command & Control* de las *botnets* analizadas.

VII tras la notable mejoría experimentada por secuencias de tamaño medio. Los resultados más bajos los obtienen nuevamente *Voted Perceptron* y *Naïve Bayes*.

Por último, en las figuras 6.15 se muestra la evolución de los valores del área por debajo de la curva ROC para la detección de tráfico de *Command & Control* en función del número de paquetes incluidos en la representación. Estos valores vienen a representar la evolución de la bondad del conjunto de datos conforme se va ampliando la longitud de las secuencias de tráfico genérico de *botnet*.

Los resultados son especialmente positivos en el caso de los árboles de decisión, tanto *C4.5*, como *Random Forest* y el método *Bagging* empleando un árbol *REPTree* obtienen valores por encima de 0,975 en la práctica totalidad de los experimentos realizados. Asimismo, la mayor parte de las versiones de KNN obtiene unos resultados especialmente satisfactorios. En lo que se refiere a la parte baja de la gráfica, es nuevamente *Voted Perceptron* el que peores resultados obtiene. Sin embargo, el algoritmo *Naïve Bayes* obtiene, al menos para las secuencias más cortas, valores superiores a los de algunos de los algoritmos que se habían venido comportando mejor de acuerdo al resto de métricas utilizadas.



(a) (b)

Figura 6.15: Evolución del valor del área por debajo de la curva ROC (AUC) según la longitud de las secuencias utilizadas para el etiquetado de tráfico genérico benigno o de *Command & Control* de las *botnets* analizadas.

6.4 Implicaciones

Dado que la cantidad de paquetes generados por el tráfico de una conexión es muy amplia, se podría pensar en una clasificación teniendo en consideración la historia de los paquetes conectados a un mismo punto. Es decir, se trataría de emplear la clasificación de los paquetes como una medida de referencia para asignar un valor que nos permita estimar el nivel de confiabilidad de la sesión iniciada en función de cómo han sido clasificados los paquetes interceptados con un determinado origen y/o destino. Para ello, en las páginas que siguen se va a formular un caso práctico que ponga de relieve el interés de este enfoque empleando como métrica de referencia la exactitud.

Llegados a este punto, la utilización de la *exactitud* —en inglés, *accuracy*— en lugar de la precisión —*Positive Predictive Value* o PPV— está justificada. Se trata de términos que no deben ser confundidos ya que hacen referencia a medidas diferentes: la exactitud —*Acc.*— está relacionada con el grado de veracidad de una dispersión teniendo en cuenta todas las clasificaciones correctas realizadas [WAD+85], mientras que la precisión —PPV— compara solamente las clasificaciones positivas realizadas, sean o no verdaderas [VIM04]. Como se quiere evaluar la capacidad de los modelos para etiquetar correctamente paquetes y otorgar así un valor de confiabilidad a la conexión que se realiza, será la exactitud el parámetro que se utilizará en las comparaciones que se detallan en este apartado.

Para obtener dichos valores se ha calculado la probabilidad $P(n)$ de que

la mayoría $(50\,\% + 1)$ de los n $(\forall n \in \mathbb{N})$ secuencias de p paquetes $(\forall p \in \{1, 2, \cdots, 12\})$ tomadas como muestra procedentes de una misma conexión, hayan sido correctamente clasificadas por un clasificador C, siendo $P(C)$ la probabilidad de etiquetar correctamente una secuencia aislada empleando dicho clasificador.

Es decir, si a correspondiera a un etiquetado correcto y b a un etiquetado erróneo, para determinar la confiafibilidad de una conexión representada por $n = 3$ secuencias, las posibilidades que se pueden dar son (aaa), (aab, aba, baa), (abb, bab, bba) y (bbb). Como a efectos de estas estimaciones solamente nos interesa analizar aquellos casos en los que nuestros clasificadores han etiquetado correctamente al menos el $50\,\% + 1$ de las muestras, las combinaciones de secuencias etiquetadas con las que nos quedaremos son las de (aaa), (aab, aba, baa).

Entonces, la probabilidad $P(n)$ resultante es la suma de: 1) la probabilidad de que las n representaciones de p paquetes hayan sido correctamente clasificadas, 2) la probabilidad de que solamente una de las representaciones haya sido incorrectamente clasificada, 3) la probabilidad de que solamente dos hayan sido incorrectamente clasificadas, etc., y así sucesivamente hasta dar con todas las posibles combinaciones con al menos $n/2 + 1$ $(\forall (n/2) \in \mathbb{N})$ representaciones correctamente clasificadas, lo que, en combinatoria, se define en la ecuación 6.20:

$$\begin{aligned} P(n) &= \binom{n}{0} \cdot P(C)^n + \\ &+ \binom{n}{1} \cdot P(C)^{n-1} \cdot [1 - P(C)] + \\ &+ \binom{n}{2} \cdot P(C)^{n-2} \cdot [1 - P(C)]^2 + \\ &+ \ldots + \\ &+ \binom{n}{\frac{n}{2} - 1} \cdot P(C)^{n-(\frac{n}{2}-1)} \cdot [1 - P(C)]^{\frac{n}{2}} = \\ &= \sum_{k=0}^{\frac{n}{2}-1} \left\{ \binom{n}{k} \cdot P(C)^{n-k} \cdot [1 - P(C)]^k \right\} \end{aligned} \tag{6.20}$$

Las ecuaciones 6.21 y 6.22 son un ejemplo de la aplicación de estas fórmula

para observaciones formadas por $n = 3$ y $n = 5$ representaciones.

$$\begin{aligned} P(3) &= \sum_{k=0}^{1} \left\{ \binom{3}{k} \cdot P(C)^{3-k} \cdot [1 - P(C)]^k \right\} = \\ &= P(C)^3 + \\ &+ 3 \cdot P(C)^2 \cdot [1 - P(C)] \end{aligned} \tag{6.21}$$

$$\begin{aligned} P(5) &= \sum_{k=0}^{2} \left\{ \binom{5}{k} \cdot P(C)^{5-k} \cdot [1 - P(C)]^k \right\} = \\ &= P(C)^5 + \\ &+ 5 \cdot P(C)^4 \cdot [1 - P(C)] + \\ &+ 10 \cdot P(C)^3 \cdot (1 - P(C)^2) \end{aligned} \tag{6.22}$$

Como resultado, se obtiene la tabla 6.5, en la que se recogen las estimaciones de haber etiquetado correctamente la mayoría de las representaciones correspondientes a cada una de las diferentes conexiones. Los cálculos tratan de estimar las posibilidades de que para secuencias de $n = 3, 5, 7, 10$ representaciones al menos el $50\,\% + 1$ de estas hayan sido correctamente identificadas por cada uno de los clasificadores como provenientes de la versión utilizada de la *botnet* empleada en cada caso. Por poner el ejemplo del clasificador con mayor precisión de la prueba, *Random Forest*, dadas siete representaciones correspondientes a una misma conexión hay un 97,44 % de posibilidades de que al menos cuatro de ellas hayan sido clasificados correctamente. Dado que la precisión de los clasificadores es relativamente alta a la hora de etiquetar el tráfico como legítimo o malicioso, a mayor número de paquetes analizados por conexión, más legitimidad alcanzará la clasificación efectuada.

Estos cálculos vienen a corroborar la hipótesis de que la detección de tráfico malicioso no tiene por qué centrarse en la categorización de paquetes aislados, sino en la confiabilidad estimada de la conexión en su conjunto. Hay que tener en cuenta que es mucho más representativo el seguimiento del comportamiento de la red en función del tiempo que la toma en consideración de representaciones de éste en momentos dados. Siendo esto así, no hay que perder de vista que las bonanzas de este método podrían verse afectadas por el desarrollo de *botnets* capaces de generar ruido en forma de tráfico automatizado, creado con el objetivo práctico de intentar camuflar

Tabla 6.5: Estimación de posibilidades de que una secuencia de paquetes correspondientes a una conexión determinada haya sido correctamente clasificada.

Clasificador	$n = 1$	$n = 3$	$n = 5$	$n = 7$	$n = 10$
SMO Polik.	0,7430	0,8358	0,8890	0,9228	0,9774
SMO Polik. N.	0,7822	0,8784	0,9275	0,9554	0,9901
SMO RBF	0,7023	0,7869	0,8399	0,8769	0,9543
SMO Pear-VII	0,8191	0,9137	0,9557	0,9765	0,9962
Naive Bayes	0,6812	0,7599	0,8112	0,8483	0,9374
Bayes Net K2	0,7903	0,8865	0,9344	0,9608	0,9919
Bayes Net Hill	0,7892	0,8854	0,9334	0,9601	0,9916
Bayes Net TAN	0,8231	0,9172	0,9583	0,9782	0,9967
C4,5	0,8214	0,9157	0,9572	0,9775	0,9965
Rand. Forest	0,8248	0,9187	0,9594	0,9790	0,9968
Voted Per.	0,6616	0,7340	0,7826	0,8188	0,9181
MLP	0,8159	0,9108	0,9536	0,9750	0,9959
KNN k=1	0,8240	0,9180	0,9589	0,9786	0,9967
KNN k=2	0,8228	0,9169	0,9581	0,9781	0,9966
KNN k=3	0,8220	0,9162	0,9576	0,9778	0,9965
KNN k=4	0,8202	0,9146	0,9564	0,9770	0,9964
KNN k=5	0,8182	0,9129	0,9551	0,9761	0,9961
KNN k=6	0,8169	0,9117	0,9542	0,9754	0,9960
KNN k=7	0,8170	0,9118	0,9543	0,9755	0,9960
KNN k=8	0,8167	0,9115	0,9541	0,9754	0,9960
KNN k=9	0,8167	0,9115	0,9541	0,9754	0,9960
KNN k=10	0,8167	0,9115	0,9541	0,9754	0,9960

los mensajes de control entre una cantidad de muestras suficientemente grande.

6.5 Sumario

En este capítulo, se han presentado los resultados que validan los modelos de representación propuestos en los capítulos anteriores. En primer lugar, se ha procedido a definir el problema como un problema de clasificación supervisada. Para ello, los conjuntos de entrenamiento utilizados han sido adecuados etiquetando muestras de tráfico a sabiendas de la localización de los canales de *Command & Control*.

En segundo lugar, se han definido los veintitrés algoritmos de aprendizaje automático utilizados como clasificadores: desde máquinas de soporte vectorial, hasta árboles de decisión, pasando por perceptrones, redes bayesianas o el análisis de los k vecinos más próximos en el espacion n-dimensional analizado. Seguidamente, se han analizado de antemano cuáles han sido las métricas a emplear para comparar los resultados obtenidos en los diferentes procesos experimentales eligiendo para ello algunas de las más utilizadas en el campo del *machine learning*. Además, teniendo en cuenta que los conjuntos de datos no estaban balanceados—es decir, que el número de muestras maliciosas y benignas no era el mismo para ambas clases dado que es más sencillo generar el tráfico legítimo— se ha procedido a definir la métrica de balanceo de datos utilizada: *Spread Subsample*, ajustando el número de muestras al número más bajo de muestras recogidas de una de las clases, garantizando así que el algoritmo era entrenado con un mismo número de representaciones.

En cuanto a los experimentos, se ha corroborado la hipótesis de que la forma en que las aplicaciones llevan a cabo la comunicación de *Command & Control* puede proveer de información valiosa a los algoritmos de aprendizaje automático. Tanto la representación que emplea paquetes individuales como la que emplea cadenas de ellos así lo indican en los experimentos realizados observándose en estas últimas una mejora sustancial.

Asimismo, las características propias del problema de clasificación de paquetes presentan una novedad adicional con respecto a los problemas de detección tradicionales: la aparición de falsos positivos no es tan relevante como ocurre en otros áreas de conocimiento dado que la escasa carga maliciosa de un paquete en sí misma no es suficiente como para suponer una amenaza real contra el nodo infectado.

«Vi Veri Vniversum Vivus Vici» («Por el poder de la verdad, mientras viva, habré conquistado el universo»).

V, personaje principal del cómic *V de Vendetta* de Alan Moore (1953 – ...)

Capítulo 7

Conclusiones

Uno de los principales problemas que trae consigo el aprendizaje automático supervisado en el campo de la clasificación del tráfico es la ingente cantidad de datos necesarios y las dificultades jurídico-legales asociadas a la extracción de las muestras. En esta línea, la necesidad legal (y moral) de tener el consentimiento tácito de los usuarios que van a generar las muestras de tráfico de red, nos obliga a asumir ciertos errores asociados a hábitos de navegación que en entornos reales podrían ser considerados normales —como visitas a páginas de contenido para adultos por ejemplo— pero cuya aparición en entornos de pruebas es escasa o nula por el mero hecho de ser consciente de estar trabajando en una red con tráfico monitorizado.

En este contexto, el objetivo de identificar una metodología de modelado del tráfico de control de una *botnet* ha sido satisfecho en los siguientes términos: en primer lugar, se han definido una serie de parámetros de clasificación que se han estimado concluyentes para la identificación del tráfico proveniente de una red concreta; en segundo lugar, se ha generado un modelo matemático sobre el que se han aplicado diferentes algoritmos de clasificación con resultados satisfactorios; y, por último, se ha realizado una estimación de las capacidades de estas técnicas para definir el nivel de confiabilidad de la conexión a la que pertenecen. Sin embargo, la principal virtud del enfoque aquí expuesto reside en que consigue clasificar el tráfico independientemente del contenido de la información que fluye por la red.

Para ello, emplea como métricas de detección valores relativos a la longitud de las cabeceras, la frecuencia de las conexiones o la periodicidad de los sondeos entre máquina infectada y máquina de control.

El resto de este capítulo final está estructurado como sigue. La sección 7.1 recoge un resumen de los temas desarrollados en esta tesis doctoral. La sección 7.2 resume los resultados obtenidos revisitando para ello la hipótesis de partida. La sección 7.3 define algunas de las principales potenciales aplicaciones de un enfoque de estas características. La sección 7.4 recoge algunas de las limitaciones a las que pueden tener que hacer frente los modelos propuestos. La sección 7.5 dimensiona las principales líneas de trabajo futuro amparadas en los planteamientos de este trabajo doctoral. Por último, la sección 7.6 pone fin a esta tesis doctoral con la extracción de las conclusiones generales de este trabajo.

7.1 Síntesis de la validación del sistema

Durante esta investigación doctoral se han llevado a cabo diferentes validaciones experimentales de los planteamientos iniciales del doctorando. A continuación, se van a enumerar cuáles son las conclusiones más relevantes a extraer de esta tesis.

Para validar la representación de las conexiones basada en los atributos atómicos de los paquetes, se utiliza un conjunto de datos de conexiones legítimas y maliciosas —procedentes de conexiones de canales de *Command & Control* de una serie de *botnets* dadas— extrayendo sus representaciones para secuencias de $n = 1$ hasta $n = 12$ paquetes. Posteriormente, dado que conocemos el tipo al que pertenecen, cada una de estas representaciones ha sido etiquetada y utilizada posteriormente para entrenar hasta veintitrés clasificadores diferentes.

En primer lugar, el conjunto de entrenamiento ha estado formado por representaciones compuestas por secuencias de paquetes de cada una de las *botnets* por separado que han sido intercaladas con representaciones de tráfico legítimo. Este experimento tenía como objetivo detectar aquellos paquetes procedentes de una misma familia de *botnets* lo cual se ha conseguido con porcentajes de precisión de en torno al 80 % para secuencias de $n = 1$ paquetes, alcanzando cotas de hasta el 95 % para algunas representaciones de más longitud.

Una vez identificado el tráfico malicioso procedente de cada una de las *botnets*, se ha querido ver si el modelo es capaz de detectar muestras gené-

ricas de tráfico. Pese a la capacidad del modelo para señalar qué conexiones son maliciosas con un alto nivel de efectividad, el objetivo de este último experimento era determinar hasta qué punto se puede identificar cada *botnet* diferenciándola de las demás. La relevancia de un enfoque de estas características, ponderada con las observaciones anteriores, es muy relevante de cara a, ya no solamente detectar nuevas infecciones, sino a tratar de ponderar el nivel de riesgo determinando el tipo de *malware* al que corresponde cada una de las muestras recogidas.

7.2 Resumen de los resultados

En esta sección se resumen las contribuciones de la tesis doctoral. Para ello, y como primer paso del proceso, es necesario repasar el contenido de la hipótesis fundamental de este trabajo:

«*El uso de características atómicas para detectar aquellas comunicaciones que compongan el proceso de envío y recepción de comandos maliciosos hacia o desde las víctimas infectadas por malware concebido para el despliegue de una botnet*».

En este sentido, los estudios realizados y resumidos en la sección 7.1 respaldan dicha hipótesis. La utilización de un modelo de representación de conexiones empleando los atributos pertenecientes a los paquetes que la conforman, ya propuesto por el autor [BGdlPUP+b], ha dado lugar a las siguientes contribuciones:

1. La definición de una metodología de trabajo para la obtención de muestras de tráfico.

2. La creación de un modelo de representación empleando los atributos de los paquetes pertenecientes a una conexión.

3. La mejora del modelo de representación anterior empleando secuencias de paquetes pertenecientes a una misma conexión.

4. La validación de estos modelos como elementos a considerar para la detección de patrones y el análisis de tráfico de red.

5. La utilización de dichos modelos para la detección de tráfico de *Command & Control* genérico.

7.3 Aplicaciones de la investigación

La utilización de técnicas de análisis de tráfico para la identificación de comportamientos automatizados ofensivos de carácter activo —como puede ser por ejemplo la ejecución de un ataque de denegación de servicio o la transferencia de ficheros— es una línea de acción bien documentada en el estado del arte de la detección de *botnets* y llevada a la práctica en forma de productos o servicios por empresas especializadas como Arbor Networks[1]. Sin embargo, estos enfoques más reactivos no son capaces de detectar otro tipo de comportamientos maliciosos vinculados a la sustracción de información o a la ejecución de actividades maliciosas en las máquinas infectadas de forma remota, una problemática que afecta cada vez con más frecuencia a empresas y corporaciones de todos los sectores económicos, así como a organismos de carácter público y administraciones.

Otros enfoques identificados en esta disertación emplean técnicas más vinculadas al análisis de los contenidos que fluyen por la red. Estas aproximaciones eran capaces de detectar en el pasado conexiones automatizadas en los primitivos canales de control de *botnets* implantados sobre plataformas IRC, pero la utilización de estas técnicas queda obsoleta a partir del momento en que los atacantes generalizan la utilización de conexiones cifradas para el envío y recepción de comandos. Esta realidad está ya más que presente como se ha visto en los casos de uso expuestos en este trabajo con algunas de las *botnets* HTTP que codifican sus órdenes a través de conexiones HTTPS. Sin embargo, el enfoque propuesto en esta tesis pretende dar una respuesta que no esté basada en el contenido de las comunicaciones si no, dejando de lado el análisis de los datos en sí mismos, que se centre en otros atributos relativos al direccionamiento de estas y a los tiempos con que los paquetes se envían y reciben.

En este sentido, la aplicación de los diferentes modelos de representación propuestos tienen una aplicación clara: la identificación de una infección por *malware* orientado al despliegue de una *botnet* empleando técnicas de análisis de tráfico. Estos métodos, complementarios a las técnicas de desinfección tradicionales, suponen una notable mejora de los sistemas de detección basados en firmas: primero, porque analizan el comportamiento del individuo en la red y, segundo, porque no son intrusivos con respecto a la privacidad de los usuarios al no estar basados en el contenido. Así, la

[1] Establecida en 2000, Arbor provee soluciones de red enfocadas a la mejora de su gestión y a la protección de las grandes redes corporativas. Página oficial: http://www.arbornetworks.com/es/

versatilidad de este enfoque nos permite utilizarlo como un indicador de la confiabilidad de una conexión determinada.

Desde un punto de vista empresarial, el impacto de un sistema de detección de canales de control vinculados a la detección de *malware* en general y a la construcción de *botnets* en particular aporta una interesante capa de protección adicional a los sistemas convencionales basados en *host*. A sabiendas de que la problemática del espionaje industrial, que explota las vulnerabilidades telemáticas presentes en las redes y equipos corporativos, es una amenaza cada vez más extendida, la identificación de filtraciones y de comportamientos maliciosos se torna fundamental para la mitigación de los efectos de actuaciones delictivas centradas en el ciberespionaje. En cualquier caso, la explotación económica de una solución de estas características puede enfocarse desde dos ámbitos: por un lado, compañías y/o instituciones con la intención de integrar herramientas similares para proteger el acceso a su propia red y, por otro, terceras empresas que ofrecieran el servicio como parte añadida de sus productos destinados a la detección de *malware* y la identificación proactiva de amenazas.

7.4 Limitaciones

La principal limitación de este enfoque es que, aunque sí ofrece pistas sobre cuál es la aplicación que lleva a cabo las conexiones no autorizadas, no provee al analista de seguridad de una respuesta definitiva que le indique cuál es el fichero infectado. Por ello, no se puede rechazar la necesidad de realizar un análisis forense más exhaustivo en todo aquel equipo sospechoso de haber sido infectado. Así, como ya se ha propuesto anteriormente, podrían existir dos enfoques diferentes capaces de distorsionar el modelo:

1. El camuflaje de las conexiones maliciosas mediante la generación de peticiones aparentemente benignas contra el servidor generando tráfico que cumpla con las características de estas es un primer problema. Teniendo en cuenta que el modelo confiará en la clase del juicio más mayoritario emitido, sería teóricamente posible que un *malware* generara artificialmente muestras benignas de forma que estas acapararan la mayoría de los paquetes de una conexión maliciosa.

2. El sistema es capaz de identificar la existencia de una infección en un equipo que se está intentando comunicar con su proveedor de órdenes, pero no lo hará si este se encuentra en estado de letargo y no

se producen intentos de conexión. Esto podría ser utilizado para espaciar en el tiempo las conexiones y realizarlas, a cada poco, contra servidores diferentes.

Además, si el operador de la *botnet* fuera capaz de transmitir los comandos desde localizaciones diferenciadas, se necesitará de una capa adicional de trabajo que permita asociar dichas comunicaciones.

7.5 Líneas futuras de trabajo

La principal línea futura de trabajo deberían enfocarse en minimizar la tasa de falsos positivos al máximo con el fin de etiquetar como tráfico corrupto aquellas muestras que con más seguridad han sido identificadas como maliciosas. Esto tendría dos consecuencias principales: 1), que la tasa de detección por paquete aislado descendería drásticamente —en la práctica, un descenso pronunciado del TPR— y 2), que los paquetes sospechosos serán maliciosos con mucha más seguridad —provocando un aumento considerable del PPV—. Si además tenemos en cuenta que la naturaleza semántica de las conexiones generadas en toda red de comunicación tiene sentido en un contexto de intercambio de información, podemos intuir que una gran cantidad de enfoques posteriores pueden ir centrados a un análisis más evolutivo del etiquetado del tráfico. Esto es, en lugar de pensar en paquetes independientes, hacerlo considerando las conexiones, definidas como sucesiones temporales de paquetes enviados y/o recibidos hacia/desde un mismo *host*.

Este tipo de aproximaciones podrían ser de especial interés si se tiene en cuenta que algunas de las representaciones son compartidas tanto por transmisiones legítimas como fraudulentas. Ser capaz de identificar aquellas representaciones de tráfico ambiguas y apartarlas de antemano del proceso de clasificación podría ser una oportunidad especialmente atractiva de cara a aumentar la distancia que separa a ambas clases. No hay que perder de vista que a la hora de etiquetar tráfico, la decisión que se toma con respecto a cada representación puede ser revertida si indicadores que nos vengan dados más adelante nos dan indicios de que nuestra clasificación original era errónea. Así, en este área, el investigador tiene, más elementos de juicio a la hora de tomar decisiones que, por ejemplo, en el caso de la detección de *malware*, en donde un solo falso negativo podría implicar la infección del equipo y una amenaza para el sistema al completo.

Al mismo tiempo, los aportes aquí descritos sugieren también una gran

variedad de líneas de trabajo futuro en campos relacionados con el modelado de tráfico. Hay que tener en cuenta que la propuesta de modelado aquí descrita se centra solamente en la detección del tráfico procedente de los canales de *Command & Control* de la *botnet*, es decir en la conexión establecida entre los nodos de la red y quien la gestiona.

De cualquier manera, el estudio aquí presentado hace frente a la detección de tráfico creado por *botnets* conocidas con cuyas muestras se han entrenado los clasificadores supervisados. Por esto mismo, y dada la gran cantidad de muestras maliciosas existentes, la aplicación de técnicas de aprendizaje de clase única que permitan la identificación de tráfico desconocido sería un gran avance de cara a anticipar la detección preventiva de nuevas muestras antes incluso de contar con muestras reales de tráfico maligno.

Asimismo, no es menos cierto que esta investigación tiene aplicaciones en dominios similares pero con enfoques diferentes. Aunque en este trabajo el objetivo era el de tratar de identificar similitudes entre el comportamiento del tráfico de *botnets*, este enfoque de clasificación de tráfico se puede extrapolar a ámbitos mucho más genéricos como la inteligencia de señales (SIGINT) o el *pentesting* para identificar la utilización de aplicaciones o servicios por canales cifrados mediante la escucha pasiva del flujo de paquetes de la red (de forma transparente al contenido). Del mismo modo, esta misma aproximación se podría utilizar para identificar hábitos de navegación cuando no se conozca el destino de la conexión, por ejemplo, cuando la navegación se está realizando a través de un *proxy* o de VPN.

Recuperando la casuística de la seguridad, un área que ha quedado sin tratar es el del estudio del tráfico saliente para la detección de comportamientos maliciosos más allá de la propia infección del equipo. Este ámbito conlleva un interés creciente para empresas e instituciones dada las posibilidades que ofrecería con vistas a hacer frente a algunas de las siguientes amenazas de una forma más efectiva: la detección de ataques DDoS, al análisis de *botnets* descentralizadas, al análisis del tráfico automatizado generado desde un nodo de la red, al estudio del tráfico saliente hacia plataformas que gestionan servicios de *pay-per-click* para cometer actos de *click-frauding* o al envío masivo de *spam* de forma similar a como se hacía en otra época en los canales IRC [GPZL08].

7.6 Consideraciones finales

Las *botnets* se han convertido ya en una de las amenazas más importantes del ciberespacio. La multidisciplinaridad de estas, la masificación de las infecciones y la gran cantidad de alternativas que tiene el botmaster para monetizar la infección son algunas de las características que hacen que este fenómeno sea de interés global. La principal novedad que presenta esta modalidad de crimen organizado es la sencillez con la que las nuevas variantes de *malware* son concebidas.

En esta tesis doctoral se propone un modelo de detección que utiliza atributos propios de los paquetes de las conexiones para identificar la existencia de conexiones maliciosas pertenecientes a canales de *Command & Control* de *botnets* como complemento de los enfoques tradicionales que detectan la infección a partir de la identificación de ejecutables en los equipos de usuario. Lo cierto es que la criptografía moderna, robusta y eficaz además de con múltiples aplicaciones como garantes últimos de la confidencialidad e integridad de las comunicaciones, es también el escudo perfecto para los ciberdelincuentes de modo que logran evadir los filtros de contenido en lo que a la transferencia de comandos se refiere.

Por todo ello, desde el punto de vista de la seguridad, se ha hecho necesario mejorar los sistemas de caracterización de las conexiones actuales mediante la definición de una metodología para el análisis de muestras de tráfico que permita estudiar estas de forma independiente a la naturaleza del contenido que circula por ellas. Aparentemente, un enfoque de estas características conseguirá sortear la problemática de la privacidad de las comunicaciones legítimas permitiendo un análisis del tráfico exhaustivo y riguroso. Sin embargo, no es menos cierto que estos enfoques extienden el significado de *privacidad* mucho más allá del acceso a los contenidos de un mensaje al permitir la identificación (aún sin detalles) de qué tipo de actividades se están realizando en los nodos monitorizados y con qué otras entidades se están llevando a cabo estas.

Hoy en día, hay quien cree haber encontrado una mayor protección en la maraña de nodos que componen la red y la imposibilidad de monitorizar todos ellos al mismo tiempo. A la vista de los acontecimientos, este doctorando no se atreve a afirmar tanto porque, aún siendo el objetivo último tan noble como dar caza a los tiranos, es muy peligroso confundir el fin —la libertad— con los medios de los que dispone la sociedad para darle alcance —la seguridad—. Benjamin Franklin, uno de los Padres Fundadores de los Estados Unidos, dijo hace ya más de dos siglos: «*They who can give*

up essential liberty to obtain a little temporary safety deserve neither liberty nor safety.».

Publicaciones y méritos

En este apartado se recogen aquellas publicaciones académicas y técnicas que han dado lugar al presente trabajo doctoral divididas en dos apartados: las relacionadas de forma directa con la temática principal de la tesis (6 artículos) y aquellas indirectamente relacionadas con el trabajo presentado pero que lo tocan transversalmente (22 documentos)[1].

Publicaciones directamente relacionadas

Este apartado incluye todas aquellas publicaciones que guardan una especial relación con los contenidos presentados en esta tesis doctoral por el contenido, la aplicación de sus metodologías o la base procedimental aplicada. En total, en este apartado se incluyen cinco comunicaciones a congresos y conferencias internacionales y un artículo de revista técnica que sustentan este trabajo.

Revistas y capítulos de libro

- Brezo, Félix; Gaviria de la Puerta, José; Ugarte-Pedrero, Xabier; Santos, Igor; Barroso, David y G. Bringas, Pablo (2013). *A Supervised Classification Approach for Detecting Packets Originated in a HTTP-based Botnet*. En *CLEI Electronic Journal*, v. 17, n. 3, paper 02, diciembre de 2013.

[1] No se incluyen en este capítulo publicaciones de carácter divulgativo.

Comunicaciones en conferencias internacionales y seminarios

- Brezo, Félix; Gaviria de la Puerta, José y G. Bringas, Pablo (2013). *Detecting Command & Control Channels of a Botnet using a N-packet-based Approach*. En *Third International Conference on Advances in Information Mining and Management* IMMM 2013 (ERA Rank[1]: –; EN Rank[2]: A), 17-22 de noviembre de 2013, Lisboa (Portugal).

- Brezo, Félix y G. Bringas, Pablo (2013). *Detección de tráfico de control de botnets mediante el modelado del flujo de paquetes*. En el *Doctoral Consortium* de *CAEPIA 2013*: *the XV Conference of the Spanish Association of Artificial Intelligence*, 17-20 de septiembre de 2013, Madrid (España).

- Brezo, Félix; Gaviria de la Puerta, José; Ugarte-Pedrero, Xabier; Santos, Igor; Barroso, David y G. Bringas, Pablo (2012). *Clasificación supervisada de paquetes procedentes de una botnet HTTP*. En *CLEI 2012* (ERA Rank: C; EN Rank: C), 1-5 de octubre de 2012, Medellín (Colombia).

- Brezo, Félix; Gaviria de la Puerta, José; Santos, Igor; Barroso, David y G. Bringas, Pablo (2012). *C&C Techniques in Botnet Development*. En *CISIS 2012* (ERA Rank: B; EN Rank: C).

- Brezo, Félix; Santos, Igor; G. Bringas, Pablo, Del Val; José Luis (2011). Challenges and Limitations in Current Botnet Detection. En FlexDBIST'11 organizado en DEXA 2011 (ERA Rank: B; EN Rank: B) Toulouse (Francia).

Publicaciones indirectamente relacionadas

Este apartado incluye aquellas publicaciones que recorren aspectos vinculados a los tratados en la tesis doctoral o que tocan áreas temáticas no directamente relacionadas con el propósito general de la tesis pero que, bien

[1]El ERA Conference Ranking, impulsado por el gobierno australiano, es el único reconocido por el Mº de Educacion. Se dejó de actualizar en 2010. Disponible en: http://core.edu.au/index.php/categories/conference%20rankings

[2]El sistema de evaluación de conferencias de la Universidad de Erlangen-Nurémberg, actualizado recientemente. Disponible en: http://www.wi2.uni-erlangen.de/_fileuploads/research/generic/ranking/index.html

conceptual o bien metodológicamente, lo soportan. En total, en este apartado se incluyen seis artículos de revista, once comunicaciones a congresos y conferencias internacionales y hasta otros cinco trabajos de carácter técnico en los que se derrollan modelos de representación, herramientas de inteligencia artificial, temáticas y metodologías también empleadas en esta tesis doctoral.

Revistas y capítulos de libro

- Rubio, Yaiza y Brezo, Félix (2013). *Nuevos escenarios para la comisión de delitos económicos en la red*. En *Cuadernos de Guardia Civil: Revista de seguridad pública*, número 43, pp. 93-114.

- Rubio, Yaiza y Brezo, Félix (2013). *Impacto de la evolución del sector aeroespacial*. En *Inteligencia y Seguridad: Revista de Análisis y Prospectiva*, número 14, diciembre de 2013 (*aceptado para su publicación*).

- Rubio, Yaiza y Brezo, Félix (2013). *La sobreexposición en la red de información geográfica acerca e instalaciones de interés para la Defensa*. En *Revista-IEEE*, n. 1, junio de 2013, Instituto Español de Estudios Estratégicos, Ministerio de Defensa.

- Santos, Igor; Brezo, Félix; Sanz, Borja; Laorden, Carlos y G. Bringas, Pablo (2013). *Opcode Sequences as Representation of Executables for Data-mining-based Unknown Malware Detection*. En *Information Sciences 2013*.

- Santos, Igor; Ugarte-Pedrero, Xabier; Brezo, Félix; G. Bringas, Pablo y Gómez-Hidalgo, José M. (2013). *NOA: An Information Retrieval Based Malware Detection System*. En *Computing And Informatics 2013*.

- Santos, Igor; Brezo, Félix; Ugarte-Pedrero, Xabier y G. Bringas, Pablo (2011). *Using Opcode Sequences in Single-Class Learning to Detect Unknown Malware*. En *IET Information Security 2011*.

Comunicaciones en conferencias internacionales y seminarios

- Brezo, Félix y Rubio Yaiza (2013). *Future Opportunities for Defense Industry on BRICKs*. En *Emerging Economies Conference '13* (EMECON'13), 25-28 de septiembre de 2013, Estambul (Turquía).

- Moya, Eva y Brezo, Félix (2013). *El futuro de la inteligencia económica*. En curso *Servicios de inteligencia y seguridad internacional (IV Edición)*, Universidad de Alicante (26 de marzo de 2013).

- Rubio, Yaiza y Brezo, Félix (2012). *Limitaciones de los motores cartográficos de búsqueda para el analista de inteligencia*. En *3rd International Intelligence Congress*, 12 de noviembre de 2012, Barcelona (España).

- Rubio, Yaiza; Brezo, Félix y Moya, Eva (2012). *Diseño de planes operativos de contrainteligencia en redes sociales en el ámbito empresarial*. En *3rd International Intelligence Congress*, 12 de noviembre de 2012, Barcelona (España).

- Brezo, Félix y G. Bringas, Pablo (2013). *Issues and Risks Associated with Cryptocurrencies such as Bitcoin*. En *Second International Conference on Social Eco-Informatics (SOTICS 2012)*, 21-26 de octubre de 2012, Venecia (Italia).

- Santos, Igor; Devesa, Jaime; Brezo, Félix; Nieves, Javier y G. Bringas, Pablo (2012). *OPEM: A Static-Dynamic Approach for Machine-learning-based Malware Detection*. En *CISIS 2012* (ERA Rank: B; EN Rank: C).

- Santos, Igor; Brezo, Félix; Sanz, Borja; Laorden, Carlos y G. Bringas, Pablo (2011). *Opcode-sequence-based Semi-supervised Unknown Malware Detection*. En *CISIS 2011* (ERA Rank: B; EN Rank: C).

- Brezo, Félix; Nieves, Javier; G. Bringas, Pablo y Salazar, Mikel (2011). *POMPEIA: Poker Opponent Classifier*. En *AAAI 2011* (ERA Rank: A; EN Rank: A), 6-9 de agosto de 2011, San Francisco, CA (EEUU).

- Nieves, Javier; Santos, Igor; Martín-Abreu, J. M.; G. Bringas, Pablo (2010). *Enhanced Foundry Production Control*. En *DEXA 2010* (ERA Rank: B; EN Rank: B), Bilbao (España).

- Gómez-Hidalgo, José M.; Martín-Abreu, J. M.; Nieves, Javier; Santos, Igor; Brezo, Félix y G. Bringas, Pablo (2010). *Data Leak Prevention through Named Entity Recognition*. En *PASWeb 2010* (ERA Rank: –; EN Rank: –).

- Santos, Igor; Brezo, Félix; Nieves, Javier; Penya, Yoseba K.; Sanz, Borja; Laorden, Carlos y G. Bringas, Pablo (2010). *Opcode-sequence-based Malware Detection*. En *ESSoS 2010* (ERA Rank: –; EN Rank: –).

Otras publicaciones y trabajos técnicos

- Rubio, Yaiza y Brezo, Félix (2013). *Futuras líneas de trabajo para la prevención de blanqueo de capitales en las plataformas de juego en línea*. IUISI (Centro de Análisis y Prospectiva, Guardia Civil) 2012: Doc ISIe nº 14/2013.

- Brezo, Félix (2012). *Aplicaciones ciberdelictivas de criptodivisas como Bitcoin*. IUISI (Centro de Análisis y Prospectiva, Guardia Civil) 2012: Doc ISIe nº 04/2012.

- Rubio, Yaiza y Brezo, Félix (2012). *La sobreexposición en la red de información geográfica acerca de instalaciones de interés para la Defensa*. Trabajo de Fin de Máster dirigido por Dr. Rubén Arcos. Calificación: 10, Cum Laude.

- Brezo, Félix y Gaviria de la Puerta, José (2012). *BRIANA: Botnet detection Relying on an Intelligent Analysis of Network Architectures*. Trabajo de Fin de Máster dirigido por David Barroso (Telefónica Digital). Calificación: 10.

- Brezo, Félix (2011). *Investigación sobre prevención de intrusiones y detección de malware en sistemas críticos industriales*. Proyecto de Fin de Carrera dirigido por Dr. Igor Santos (DeustoTech Computing). Calificación: 9,7.

7.7 Otros méritos científicos

En este apartado se recogen otros méritos científicos presentados por el doctorando en el período de preparación de la presente tesis doctoral.

Premios

- Accésit del VI Premio de investigación UD-Santander por el trabajo Opcode-sequence-based Malware Detection (2011). Junto a Igor Santos y Pablo G. Bringas.

Comités de revisión

- Revistas:

  - IET Information Security 2012. *Impact Factor*: 0,862.
  - Journal of Systems and Software (Elsevier Editorial System). *Impact Factor*: 0,836.

- Conferencias internacionales:
  - CISIS 2012, SS02 - Text Processing Applications to Secure and Intelligent Systems (ERA Rank: B; EN Rank: C).
  - CISIS 2013 (ERA Rank: B; EN Rank: C)
  - SOTICS 2013 (ERA Rank: –; EN Rank: A)

Presidencia de sesiones científicas

- SOTICS 2012, SOTICS 3 Mechanisms for social services (ERA Rank: –; EN Rank: A)
- SOTICS 2012, SOTICS 5 Mechanisms for social services (ERA Rank: –; EN Rank: A)

Bibliografía

[ACK+02] David P Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, y Dan Werthimer. SETI@ home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.

[Adl] Leonard Adleman. An abstract theory of computer viruses. En *Advances in Cryptology—CRYPTO'88, pages=354–374, year=1990, organization=Springer*.

[ASS] M. Asvial, D. Sirat, y B. Susatyo. Design and analysis of anti spamming SMS to prevent criminal deception and billing froud: Case Telkom Flexi. En *Management of Innovation and Technology, 2008. ICMIT 2008. 4th IEEE International Conference on, pages=928–933, year=2008, organization=IEEE*.

[AW99] S. Amari y S. Wu. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12(6):783–789, 1999.

[Bak11] J Bakker. CSO SECURITY AND RISK: DDoS attack forces Dutch bank offline, 2011.

[Bay63] T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society*, 53:370–418, 1763.

[BBR+] Leyla Bilge, Davide Balzarotti, William Robertson, Engin Kirda, y Christopher Kruegel. Disclosure: detecting botnet command and control servers through large-scale NetFlow

analysis. En *Proceedings of the 28th Annual Computer Security Applications Conference, pages=129–138, year=2012, organization=ACM*.

[BCJ+05] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, D. Watson, et al. The Internet Motion Sensor: A distributed blackhole monitoring system. En *Proceedings of the 12th ISOC Symposium on Network and Distributed Systems Security (SNDSS)*, páginas 167–179. Citeseer, 2005.

[BDM10] Arun Bakshi, Vikas Dixit, y Kaushal Mehta. Virus: A Menace for Information Security. *Global Journal of Enterprise Information System*, 2(1), 2010.

[BGB] Félix Brezo y Pablo G Bringas. Issues and Risks Associated with Cryptocurrencies Such as Bitcoin. En *SOTICS 2012, The Second International Conference on Social Eco-Informatics, pages=20–26, year=2012*.

[BGdlPB] Félix Brezo, José Gaviria de la Puerta, y Pablo G Bringas. Detecting Command and Control Channels of a Botnet Using a N-packet-based Approach. En *The Third International Conference on Advances in Information Mining and Management, IMMM 2013, year=2013, organization=IARIA*.

[BGdlPS+] Félix Brezo, José Gaviria de la Puerta, Igor Santos, David Barroso, y Pablo G. Bringas. Command and Control Techniques in Botnet Development . En *International Joint Conference CISIS'12-ICEUTE'12-SOCO'12,, year=2012, organization=IEEE*.

[BGdlPSB13] Félix Brezo, José Gaviria de la Puerta, Igor Santos, y Pablo G. Bringas. A Supervised Classification Approach for Detecting Packets Originated in a HTTP-based Botnet. *CLEI Electronic Journal*, 17, 2013.

[BGdlPUP+a] Félix Brezo, Jose Gaviria de la Puerta, Xabier Ugarte-Pedrero, Igor Santos, Pablo G Bringas, y David Barroso. Supervised classification of packets coming from a HTTP botnet. En *Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En, pages=1–8, year=2012, organization=IEEE*.

[BGdlPUP+b] Félix Brezo, José Gaviria de la Puerta, Xabier Ugarte-Pedrero, Igor Santos, Pablo G Bringas, y David Barroso. Supervised Classification of Packets Coming From a HTTP Botnet. En *Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En, pages=1–8, year=2012, organization=IEEE*.

[BHB+] Ulrich Bayer, Imam Habibi, Davide Balzarotti, Engin Kirda, y Christopher Kruegel. A view on current malware behaviors. En *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET), year=2009*.

[Bis06] C. M. Bishop. *Pattern recognition and machine learning*. Springer New York., 2006.

[BJZY12] Urvesh Bhowan, Mark Johnston, Mengjie Zhang, y Xin Yao. Evolving Diverse Ensembles using Genetic Programming for Classification with Unbalanced Data. 2012.

[BO02] Xabier Basogain Olabe. *Redes neuronales artificiales y sus aplicaciones*. Escuela Superior de Ingeniería de Bilbao, EHU , 2002.

[BOB+] Hamad Binsalleeh, Thomas Ormerod, Amine Boukhtouta, Prosenjit Sinha, Amr Youssef, Mourad Debbabi, y Lingyu Wang. On the analysis of the zeus botnet crimeware toolkit. En *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on, pages=31–38, year=2010, organization=IEEE*.

[BOB+10] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, y L. Wang. On the analysis of the zeus botnet crimeware toolkit. En *Eighth Annual International Conference on Privacy Security and Trust (PST)*. 2010.

[Bor07] C. Borghello. Botnets, redes organizadas para el crimen. 2007.

[BPH+] M. Balduzzi, C. Platzer, T. Holz, E. Kirda, D. Balzarotti, y C. Kruegel. Abusing social networks for automated user profiling. En *Recent Advances in Intrusion Detection, pages=422–441, year=2010, organization=Springer*.

[BPPS] P.G. Bringas, Y.K. Penya, S. Paraboschi, y P. Salvaneschi. Bayesian-Networks-Based Misuse and Anomaly Prevention System. En *Proceedings of the Tenth International Conference on Enterprise Information Systems (ICEIS 2008), pages=62–69*.

[Bra90] Anne W Branscomb. Rogue computer programs and computer rogues: Tailoring the punishment to fit the crime. *Rutgers Computer & Tech. LJ*, 16:1, 1990.

[Bre94] L. Breiman. Bagging predictors: Technical Report No. 421. Informe técnico, Berkeley, CA, itd: University of California-Department of Statistics, 1994.

[Bre96a] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[Bre96b] L. Breiman. Out-of-bag estimation. Informe técnico, Citeseer, 1996.

[Bre01a] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[Bre01b] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. ISSN 0885-6125. 10.1023/A:1010933404324.

[Bre12] Félix Brezo. Aplicaciones ciberdelictivas de criptodivisas como Bitcoin. Informe técnico, IUISI, UNED, 2012.

[Bri] P.G. Bringas. *Entorno de seguridad inteligente para la detección y prevención de intrusiones basado en la detección unificada de patrones y anomalías*. Tesis Doctoral, University of Deusto.

[Bri01] Ted Bridis. Federal bureau of investigation (fbi) develops eavesdropping tools, 2001.

[BS] A. Bloxham y S. Swinford. WikiLeaks cyberwar: hackers planning revenge attack on Amazon.

[BSBdV] Félix Brezo, Igor Santos, Pablo G Bringas, y José Luis del Val. Challenges and Limitations in Current Botnet Detection. En *Database and Expert Systems Applications (DEXA), 2011 22nd International Workshop on, pages=95–101, year=2011, organization=IEEE*.

[BSBK] L. Bilge, T. Strufe, D. Balzarotti, y E. Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. En *Proceedings of the 18th international conference on World wide web, pages=551–560, year=2009, organization=ACM*.

[Cal13] Joaquín Calderón. El Mando Conjunto de Ciberdefensa de las Fuerzas Armadas podrá planear ataques y no supondrá más gasto. *Atenea Digital*, 2013.

[Cas] C. Castillo. SEXY VIEW: EL INICIO DE LAS BOTNETS PARA DISPOSITIVOS MÓVILES.

[Cas56] Jorge Castel. *España y el Tratado de Münster, 1644-1648*. Artes Gráficas Martos, 1956.

[ČC09] Lenka Černá y Milan Chytrỳ. Supervised classification of plant communities with artificial neural networks. *Journal of Vegetation Science*, 16(4):407–414, 2009.

[CdSI10] Andrés Castrillejo, Natalia da Silva, y Gabriel Illanes. Consistencia de Random Forests y otros clasificadores promediados, 2010.

[CFSW] E. Chin, A.P. Felt, V. Sekar, y D. Wagner. Measuring user confidence in smartphone security and privacy. En *Proceedings of the Eighth Symposium on Usable Privacy and Security, pages=1, year=2012, organization=ACM*.

[CG11] Juan Antonio Calles y Pablo Gonzàlez. Troyano Flu b0.4 Windows. Manual de Usuario. 2011.

[CGH96] Enrique Castillo, José M. Gutiérrez, y Ali S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer, New York, NY, USA, 1996. ISBN 0387948589.

[CH91] Gregory F. Cooper y Edward Herskovits. A bayesian method for constructing bayesian belief networks from databases. En *Proceedings of the 7^{th} conference on Uncertainty in artificial intelligence*. 1991.

[CHC+10] Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, y Chih-Jen Lin. Training and testing low-degree polynomial data mappings via linear SVM. *The Journal of Machine Learning Research*, 11:1471–1490, 2010.

[Che03] Thomas M Chen. Trends in viruses and worms. *The Internet Protocol Journal*, 6(3):23–33, 2003.

[CJM] Evan Cooke, Farnam Jahanian, y Danny McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. En *Proceedings of the USENIX SRUTI Workshop*, *pages=39–44*, *year=2005*.

[CL12] H. Choi y H. Lee. Identifying botnets by capturing group activities in DNS traffic. *Computer Networks*, 56(1):20–33, 2012.

[Cla13] Jon Clay. Botnets Are Everywhere – See How They Spread in the Trend Micro Global Botnet Map. Informe técnico, Trend Micro, 2013.

[CLK] H. Choi, H. Lee, y H. Kim. BotGAD: detecting botnets by capturing group activities in network traffic. En *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE, COMSWARE, volume=9, pages=2, year=2009*.

[CLT13] A. Colubi, A. Lubiano, y P. Terán. *Estadística Administrativa I: Medidas de dispersión*. GAP-Oviedo, 2013.

[Coh87] Fred Cohen. Computer viruses: theory and experiments. *Computers & security*, 6(1):22–35, 1987.

[Coh89] Fred Cohen. Models of practical defenses against computer viruses. *Computers & Security*, 8(2):149–160, 1989.

[Coh91] Fred Cohen. A cost analysis of typical computer viruses and defenses. *Computers & Security*, 10(3):239–250, 1991.

[Coh94] F. Cohen. *A short course on computer viruses*. John Wiley & Sons, Inc. New York, NY, USA, 1994.

[Col] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. En *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, pages=1–8, year=2002, organization=Association for Computational Linguistics*.

[Cor10] Luis Corrons. Mariposa botnet, 2010.

[Cra10] Tim Cranton. Cracking Down on Botnets. 2010.

[CX12] Silvio Cesare y Yang Xiang. Software Similarity Searching and Classification. *Software Similarity and Classification*, páginas 71–75, 2012.

[dCdFD+] J.P.C.L. da Costa, E.P. de Freitas, B.M. David, D. Amaral, y RT de Sousa Jr. Improved Blind Automatic Malicious Activity Detection in Honeypot Data. En *The International Conference on Forensic Computer Science (ICoFCS), year=2012*.

[Dew84] Alexander K Dewdney. In the game called Core War hostile programs engage in a battle of bits. *Scientific American*, 250(5):15–19, 1984.

[DF03] S. Dudoit y J. Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19(9):1090–1099, 2003.

[DGP08] L. M. De Campos, J. A. Gámez Martín, y J. M. Puerta. Learning bayesian networks by ant colony optimisation: searching in two different spaces. *Mathware & Soft Computing*, 9(3):251–268, 2008.

[Dun09] K. Dunham. *Mobile malware attacks and defense*. Syngress Publishing, 2009.

[DWF03] Christian Dewes, Arne Wichmann, y Anja Feldmann. An analysis of internet chat systems. En *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (ICM)*, páginas 51–64. ACM, New York, NY, USA, 2003. ISBN 1-58113-773-7.

[Ell99] J.H. Elliott. Europa después de la Paz de Westfalia. *Pedralbes: revista d'història moderna*, (19):131–146, 1999.

[Eur08] Europol. Training course on investigating malware and botner-driven crime, 2008.

[FFC+] A.P. Felt, M. Finifter, E. Chin, S. Hanna, y D. Wagner. A survey of mobile malware in the wild. En *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, pages=3–14, year=2011, organization=ACM.*

[FGG97] Nir Friedman, Dan Geiger, y Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2):131–163, 1997.

[FH52] E. Fix y J. L. Hodges. Discriminatory analysis: Nonparametric discrimination: Small sample performance. *Technical Report Project 21-49-004, Report Number 11*, 1952.

[FLJ+] Chris Fleizach, Michael Liljenstam, Per Johansson, Geoffrey M Voelker, y Andras Mehes. Can you infect me now?: malware propagation in mobile phone networks. En *Proceedings of the 2007 ACM workshop on Recurring malcode, pages=61–68, year=2007, organization=ACM.*

[Flo13] Michelle FlorCruz. Chinese Netizens Respond to NSA PRISM Data Mining Scandal. *International Business Times. Retrieved June*, 13, 2013.

[Föl73] Hans Föllmer. On entropy and information gain in random fields. *Probability Theory and Related Fields*, 26(3):207–217, 1973.

[for11] Fortinet threat landscape research shows two new malware variants targeting facebook users, 2011.

[Fra10] Antonia Fraser. *The gunpowder plot: Terror and faith in 1605*. Hachette UK, 2010.

[FRBH] Nicholas Fraser, Richard Raines, Rusty Baldwin, y Kenneth Hopkinson. Mitigating Distributed Denial of Service Attacks in an Anonymous Routing Environment: Client Puzzles and Tor. En *Proceedings of the International Conference on*

i-Warfare and Security 2006, *pages=85*, *year=2006*, *organization=Academic Conferences Limited*.

[FS99] Yoav Freund y Robert E Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

[fSEC+94] International Organization for Standardization/International Electrotechnical Commission et al. Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model. *ISO/IEC*, páginas 7498–1, 1994.

[Gar] S. R. Garner. Weka: The Waikato environment for knowledge analysis. En *Proceedings of the New Zealand Computer Science Research Students Conference*, *pages=57–64*, *year=1995*.

[GC] Sarah Gordon y David Chess. Where there's Smoke, there's Mirrors: the Truth about Trojan Horses on the Internet. En *Virus Bulletin International Conference Proceedings*, *year=1998*.

[GD98] MW Gardner y SR Dorling. Artificial neural networks (the multilayer perceptron)–a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[GGP+97] Dan Geiger, Moises Goldszmidt, G. Provan, P. Langley, y P. Smyth. Bayesian network classifiers. En *Machine Learning*, páginas 131–163. 1997.

[GH07] J. Goebel y T. Holz. Rishi: Identify bot contaminated hosts by irc nickname evaluation. En *Proceedings of the USENIX Workshop on Hot Topics in Understanding Botnets (HotBots)*. 2007.

[GM13] Glenn Greenwald y Ewen MacAskill. NSA Prism program taps in to user data of Apple, Google and others. *The Guardian*, 7(6), 2013.

[Gol06] Hilary Goldstein. V for Vendetta: Comic vs. Film. *IGN*, March 2006.

[Gos11] Alexander Gostev. Kaspersky Security Bulletin. Malware Evolution 2010. Informe técnico, Karspersky Labs, February 2011.

[Got07] Greg Goth. The politics of DDoS attacks. *Distributed Systems Online, IEEE*, 8(8):3–3, 2007.

[GPJ04] José Ramón García y Alberto Pérez Jover. Mydoom: El gusano del día del juicio final. *PC world*, (207):108–126, 2004.

[GPZL08] Gu Guofei, Roberto Perdisci, Junjie Zhang, y Wenke Lee. BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection. En *SS'08 Proceedings of the 17th conference on Security symposium*, páginas 139–154. USENIX Association, 2008.

[GS+66] D.M. Green, J.A. Swets, et al. *Signal detection theory and psychophysics*, tomo 1974. Wiley New York, 1966.

[GSN+07] J.B. Grizzard, V. Sharma, C. Nunnery, BB Kang, y D. Dagon. Peer-to-peer botnets: Overview and case study. En *Proceedings of the First USENIX Workshop on Hot Topics in Understanding Botnets*. 2007.

[GZL08] G. Gu, J. Zhang, y W. Lee. BotSniffer: Detecting botnet command and control channels in network traffic. En *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*. Citeseer, 2008.

[Har12] Jeffrey A Hart. Information and communications technologies and power. *Cyberspaces and Global Affairs*, página 203, 2012.

[HKNT] Markus Huber, Stewart Kowalski, Marcus Nohlberg, y Simon Tjoa. Towards automating social engineering using social networking sites. En *Computational Science and Engineering, 2009. CSE'09. International Conference on, volume=3, pages=117–124, year=2009, organization=IEEE*.

[Hun08] Philip Hunter. PayPal, FBI and others wage war on Botnet armies. Can they succeed? *Computer Fraud & Security*, 2008(5):13–15, 2008.

[Hyp06] Mikko Hypponen. Malware goes mobile. *Scientific American*, 295(5):70–77, 2006.

[IA] Rafiqul Islam y Irfan Altas. A comparative study of malware family classification. En *Information and Communications Security, pages=488–496, year=2012, publisher=Springer*.

[IB02] John Ioannidis y Steven Michael Bellovin. Implementing pushback: Router-based defense against DDoS attacks. 2002.

[IH05] Nicholas Ianelli y Aaron Hackworth. Botnets as a vehicle for online crime. *CERT Coordination Center*, páginas 1–28, 2005.

[Inf11] InfoSecurity. Anonymus hacking group uses IRC channles to co-ordinate DDoS attacks. 2011.

[Int12] Estudios Internet. Estudio sobre hábitos en Internet e identidad digital de las marcas. Informe técnico, Factoría Interactiva con la colaboaración del Instituto de Economía Digital ICEMD-ESIC, 2012.

[IT10] Kaspersky IT. Computer threats. En *Technology Day 2010, Costa Rica*. Eset, 2010.

[ITBV12] Rafiqul Islam, Ronghua Tian, Lynn M Batten, y Steve Versteeg. Classification of Malware Based on Integrated Static and Dynamic Features. *Journal of Network and Computer Applications*, 2012.

[Jon13] Steven L Jones. The Passion of Bradley Manning: The Story behind the Wikileaks Whistle-Blower. *Journal of Military Ethics*, 12(2), 2013.

[JWCY07] Liangxiao Jiang, Dianhong Wang, Zhihua Cai, y Xuesong Yan. Survey of improving naive bayes for classification. En Reda Alhajj, Hong Gao, Xue Li, Jianzhong Li, y Osmar Zaïane, editores, *Advanced Data Mining and Applications*, tomo 4632 de *Lecture Notes in Computer Science*, páginas 134–145. Springer Berlin / Heidelberg, 2007. ISBN 978-3-540-73870-1. 10.1007/978-3-540-73871-8_14.

[Kas12] Kaspersky Lab. How Kaspersky Lab and CrowdStrike Dismantled the Second Hlux/Kelihos Botnet: Success Story. 2012.

[Kee02] Rich Kee. Evolution of the Computer Virus, 2002.

[Kle96] E. M. Kleinberg. An overtraining-resistant stochastic modeling method for pattern recognition. *The annals of statistics*, 24(6):2319–2349, 1996.

[KMXS10] Erhan Kartaltepe, Jose Morales, Shouhuai Xu, y Ravi Sandhu. Social network-based botnet command-and-control: Emerging threats and countermeasures. En Jianying Zhou y Moti Yung, editores, *Applied Cryptography and Network Security*, tomo 6123 de *Lecture Notes in Computer Science*, páginas 511–528. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-13707-5. 10.1007/978-3-642-13708-2_30.

[Kot] S. B. Kotsiantis. Supervised Machine Learning: A Review of Classification Techniques. En *Proceeding of the Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies, pages=3–24, year=2007,*.

[KRH07] A. Karasaridis, B. Rexroad, y D. Hoeflin. Wide-scale botnet detection and characterization. En *Proceedings of the USENIX Workshop on Hot Topics in Understanding Botnets (HotBots)*. 2007.

[KSCW97] Jeffrey O Kephart, Gregory B Sorkin, David M Chess, y Steve R White. Fighting computer viruses. *Scientific American*, 277(5):56–61, 1997.

[KSS] Hahnsang Kim, Joshua Smith, y Kang G Shin. Detecting energy-greedy anomalies and mobile malware variants. En *Proceedings of the 6th international conference on Mobile systems, applications, and services, pages=239–252, year=2008, organization=ACM.*

[KWC93] Jeffrey O Kephart, Steve R White, y David M Chess. Computers and epidemiology. *Spectrum, IEEE*, 30(5):20–26, 1993.

[Kya] Othmar Kyas. Mobile WiMAX for networks with enhanced security and reliability requirements. En *Military Communications Conference, 2007. MILCOM 2007. IEEE, pages=1–4, year=2007, organization=IEEE*.

[Lan86] Christopher G Langton. Studying artificial life with cellular automata. *Physica D: Nonlinear Phenomena*, 22(1):120–149, 1986.

[Law08] G. Lawton. Is it finally time to worry about mobile malware? *Computer*, 41(5):12–14, 2008.

[Lea05] N. Leavitt. Mobile phones: the next frontier for hackers? *Computer*, 38(4):20–23, 2005.

[les] The new front line: Estonia under cyberassault, author=Lesk, Michael, journal=Security & Privacy, IEEE, volume=5, number=4, pages=76–79, year=2007, publisher=IEEE.

[Lew98] D. D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. *Lecture Notes in Computer Science*, 1398:4–18, 1998.

[LH06] X. Lou y K. Hwang. Prevention of index-poisoning DDoS attacks in peer-to-peer file-sharing networks. *submitted to IEEE Trans. on Multimedia, Special Issue on Content Storage and Delivery in P2P Networks*, 2006.

[Lil] K. Lillington. Time to talk: Anonymus speaks outs.

[LJVR07] J.M. Lobo, A. Jiménez-Valverde, y R. Real. AUC: a misleading measure of the performance of predictive distribution models. *Global Ecology and Biogeography*, 17(2):145–151, 2007.

[LLBG11] Fernando Lera-López, Margarita Billon, y María Gil. Determinants of internet use in spain. *Economics of Innovation and New Technology*, 20(2):127–152, 2011. doi:10.1080/10438590903378017.

[LLLF] Weimin Luo, Jingbo Liu, Jing Liu, y Chengyu Fan. An analysis of security in social networks. En *Dependable, Autonomic and Secure Computing, 2009. DASC'09. Eighth IEEE International Conference on, pages=648–651, year=2009, organization=IEEE*.

[LNR06] J. Liang, N. Naoumov, y K.W. Ross. The index poisoning attack in p2p file sharing systems. En *IEEE INFOCOM*, tomo 6. Citeseer, 2006.

[LRS] E. Lamma, F. Riguzzi, y S. Storari. Improving the K2 Algorithm Using Association Rule Parameters. En *Proceedings of the 2004 Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU), pages=1667–1674, year=2004*.

[LS04] Mei-Ling Liu y José María Peña Sánchez. *Computación distribuida: Fundamentos y aplicaciones*. Pearson Educación, 2004.

[LT11] C.X.F.B.Y. Lihua y L.X.Z. Tianning. Andbot: towards advanced mobile botnets. 2011.

[LWLS06] Carl Livadas, Robert Walsh, David Lapsley, y W. Timothy Strayer. Using machine learning techniques to identify botnet traffic. En *In 2nd IEEE LCN Workshop on Network Security (WoNS'2006*, páginas 967–974. 2006.

[Ly08] H. Liang-you. On the Fraud by Mobile Phone Short Message and Its Countermeasures. *Journal of Chongqing University of Posts and Telecommunications (Social Science Edition)*, 6:010, 2008.

[Mar08] John Markoff. Before the gunfire, cyberattacks. *New York Times*, 12:27–28, 2008.

[MD11] Steve Mansfield-Devine. Anonymous: serious threat or mere annoyance? *Network Security*, 2011(1):4–10, 2011.

[Mej13] Carlos Mejia. Malware Attacks Targeting Hugo Chavez's Death. Informe técnico, Symantec, 2013.

[MHM] Ryan McDonald, Keith Hall, y Gideon Mann. Distributed training strategies for the structured perceptron. En *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages=456–464, year=2010, organization=Association for Computational Linguistics*.

[MMDC13] Geoff McDonald, Liam O Murchu, Stephen Doherty, y Eric Chien. Stuxnet 0.5: the missing line. Informe técnico, Symantec, Security Response, 2013.

[Nak09] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. http://www.bitcoin.org, 2009.

[Nav07] Miguel Ángel Esteban Navarro. *Glosario de inteligencia*. Ministerio de Defensa, 2007.

[Naz07] Jose Nazario. Estonian DDoS Attacks-A summary to date, 2007.

[Naz09a] Jose Nazario. Politically Motivated Denial of Service Attacks. *The Virtual Battlefield: Perspectives on Cyber Warfare*, páginas 163–181, 2009.

[Naz09b] Jose Nazario. Twitter-based Botnet Command Channel. 2009.

[Neg05] Michael Negnevitsky. *Artificial intelligence: a guide to intelligent systems*. Pearson Education, 2005.

[New07] BBC News. Fbi tries to fight zombie hordes, 2007.

[NMH+10] Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, y Nikita Borisov. BotGrep: Finding P2P Bots with Structured Graph Analysis. 2010.

[NR09] Naoum Naoumov y Keith Ross. Exploiting P2P Systems for DDoS Attacks. 2009.

[Off07] FBI National Press Office. Over 1 Million Potential Victims of Botnet Cyber Crime, 2007.

[OWD+] Thomas Ormerod, Lingyu Wang, Mourad Debbabi, Amr Youssef, Hamad Binsalleeh, Amine Boukhtouta, y Prosenjit Sinh. Defaming botnet toolkits: A bottom-up approach to mitigating the threat. En *eCrime Researchers Summit (eCrime), year=2010,*.

[Pag12] Francois Paget. Police Ransomware' Preys on Guilty Consciences. Informe técnico, McAfee, 2012.

[Paq00] Jeremy Paquette. A history of viruses. *SecurityFocus, January*, 16:2004, 2000.

[Par05] Jussi Parikka. The universal viral machine: Bits, parasites and the media ecology of network culture. *CTheory*, 12(15):2005, 2005.

[Pau10] D Pauli. PayPal hit by DDoS attack after dropping Wikileaks, 2010.

[Pem05] Matthew Pemble. Evolutionary trends in bank customer-targeted malware. *Network Security*, 2005(10):4–7, 2005.

[Per] Paul Perkinson. Lessons learned from an active cyber defense deployment pilot program. En *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2012, pages=1–15, year=2012, organization=IEEE*.

[Pla99] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. *Advances in Kernel Methods-Support Vector Learning*, 208, 1999.

[Pou08] Kevin Poulsen. Valve Tried to Trick Half Life 2 Hacker Into Fake Job Interview. Informe técnico, Wired, 2008.

[Pow07] D.M.W. Powers. Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation. *School of Informatics and Engineering, Flinders University, Adelaide, Australia, Tech. Rep. SIE-07-001*, 2007.

[qui] *C4. 5: programs for machine learning, author=Quinlan, J.R., year=1993, publisher=Morgan kaufmann*.

[Qui86] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[Qui93] J. R. Quinlan. *C4. 5 programs for machine learning*. Morgan Kaufmann Publishers, 1993.

[Qui98] Miguel Quintanilla. Técnica y cultura. *Teorema*, 17(3):49–69, 1998.

[Rac04] S. Racine. Analysis of internet relay chat usage by ddos zombies. *Master's thesis, Swiss Federal Institute of Technology Zurich*, 2004.

[Rag] Steve Ragan. Overview: Inside the Zeus Trojan's source code.

[Rai12] Costin Raiu. Cyber-threat evolution: the past year. *Computer Fraud & Security*, 2012(3):5–8, 2012.

[Ram13] Ignacio Ramonet. ¡ Todos fichados! *Le Monde diplomatique en español*, (213):1–2, 2013.

[RB13] Yaiza Rubio y Félix Brezo. Futuras líneas de trabajo para la prevención de blanqueo de capitales en las plataformas de juego en línea. Informe técnico, IUISI, UNED, 2013.

[RDP04] S. Racine, T. Dubendorfer, y B. Plattner. Analysis of Internet Relay Chat Usage by DDoS Zombies. 2004.

[Rho00] Keith A Rhodes. Information Security:Ï LOVE YOUÇomputer Virus Emphasizes Critical Need for Agency and Governmentwide Improvements. Informe técnico, DTIC Document, 2000.

[RN03] S. J. Russell y Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[Roj10] Elisabeth Rojas. Malware: terremoto en Haití. Informe técnico, MuyComputerPRO, 2010.

[ROL+] M. Riccardi, D. Oro, J. Luna, M. Cremonini, y M. Vilanova. A framework for financial botnet analysis. En *eCrime Researchers Summit (eCrime), year=2010*,.

[Sal94] Steven L. Salzberg. C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16:235–240, 1994. ISSN 0885-6125. 10.1007/BF00993309.

[Sau11] Ester Romero Saugar. Anonymous¿ Por qué protestan? 2011.

[SBN+10a] Igor Santos, Félix Brezo, Javier Nieves, Yoseba Penya, Borja Sanz, Carlos Laorden, y Pablo Bringas. Idea: Opcode-sequence-based malware detection. En *Engineering Secure Software and Systems*, tomo 5965 de *LNCS*, páginas 35–43. 2010. 10.1007/978-3-642-11747-3_3.

[SBN+10b] Igor Santos, Félix Brezo, Javier Nieves, Yoseba Penya, Borja Sanz, Carlos Laorden, y Pablo Bringas. Idea: Opcode-Sequence-Based Malware Detection. En *Engineering Secure Software and Systems*, tomo 5965 de *LNCS*, páginas 35–43. 2010. 10.1007/978-3-642-11747-3_3.

[SBS+] Igor Santos, Félix Brezo, Borja Sanz, Carlos Laorden, y Pablo García Bringas.

[SBUPB13] Igor Santos, Félix Brezo, Xabier Ugarte-Pedrero, y Pablo G. Bringas. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Information Sciences*, 231:64 – 82, 2013. ISSN 0020-0255. doi: 10.1016/j.ins.2011.08.020.

[SDB+] Igor Santos, Jaime Devesa, Félix Brezo, Javier Nieves, y Pablo Garcia Bringas. OPEM: A Static-Dynamic Approach for Machine-Learning-Based Malware Detection. En *International Joint Conference CISIS'12-ICEUTE´ 12-SOCO´ 12 Special Sessions, pages=271–280, organization=Springer*.

[SdlPSPL+12] Igor Santos, Jorge de-la Peña-Sordo, Iker Pastor-López, Patxi Galán-García, y Pablo G Bringas. Automatic categorisation of comments in social news websites. *Expert Systems with Applications*, 2012.

[Sei] J. Seiiler. Entrance of Wikileaks Into Fourth Estate Creates Perils, Opportunities.

[Sel] Larry Seltzer. Zeus Source Code Released.

[SG09] A.K. Seewald y W.N. Gansterer. On the detection and identification of botnets. *Computers & Security*, 2009.

[SH12] Michael Sikorski y Andrew Honig. *Practical Malware Analysis*. No Starch Press, 2012.

[Sha51] C. E. Shannon. Prediction and Entropy of Printed English. *Bell Systems Technical Journal*, 30:50–64, 1951.

[SHS] Kyong Jin Shim, Kuo-Wei Hsu, y Jaideep Srivastava. Modeling Player Performance in Massively Multiplayer Online Role-Playing Games: The Effects of Diversity in Mentoring Network. En *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on, pages=438–442, year=2011, organization=IEEE.*

[SKK08] G. Starnberger, C. Kruegel, y E. Kirda. Overbot: a botnet protocol based on Kademlia. En *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, página 13. ACM, 2008.

[SKM09] Y. Singh, A. Kaur, y R. Malhotra. Comparative analysis of regression and machine learning methods for predicting fault proneness models. *International Journal of Computer Applications in Technology*, 35(2):183–193, 2009.

[SKV] G. Stringhini, C. Kruegel, y G. Vigna. Detecting spammers on social networks. En *Proceedings of the 26th Annual Computer Security Applications Conference, pages=1–9, year=2010, organization=ACM.*

[SLTW04] V. Svetnik, A. Liaw, C. Tong, y T. Wang. Application of Breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules. *Multiple Classifier Systems*, páginas 334–343, 2004.

[Spa] K.A. Spackman. Signal detection theory: Valuable tools for evaluating inductive learning. En *Proceedings of the sixth international workshop on Machine learning, pages=160–163, year=1989, organization=Morgan Kaufmann Publishers Inc.*

[Spa90] Eugene H Spafford. Computer Viruses–A Form of Artificial Life? 1990.

[SPDB09] I. Santos, Y.K. Penya, J. Devesa, y P.G. Bringas. N-Grams-based file signatures for malware detection. En *Proceedings of the 11th International Conference on Enterprise Information Systems (ICEIS), Volume AIDSS*, páginas 317–320. 2009.

[Spi03] L. Spitzner. The honeynet project: Trapping the hackers. *IEEE Security & Privacy*, 1(2):15–23, 2003.

[SPW02] Stuart Staniford, Vern Parxson, y Nicholas Weaver. How to own the internet in your spare time. En *Proceedings of the 11th USENIX Security Symposium*. 2002.

[SS12] Cyber Intelligence Section y Criminal Intelligence Section. Bitcoin Virtual Currency: Intelligence Unique Features Present Distinct Challenges for Deterring Illicit Activity (Unclassified). Informe técnico, FBI: Directorate of Intelligence, 2012.

[SSL$^+$] Igor Santos, Borja Sanz, Carlos Laorden, Félix Brezo, y Pablo G. Bringas.

[Sta04] William Stallings. *Comunicaciones y Redes de Computadores*. Pearson, Prentice Hall, 20004.

[SW49] C. E. Shannon y W. Weaver. The mathematical theory of communication. 1949.

[sym09] State of phising: A monthly report (report 25, novemeber 2009), 2009.

[Szo05] Peter Szor. *The art of computer virus research and defense*. Addison-Wesley Professional, 2005.

[TT12] Alex Teichman y Sebastian Thrun. Tracking-based semi-supervised learning. *The International Journal of Robotics Research*, 31(7):804–818, 2012.

[UMB06] B. Üstün, W.J. Melssen, y L.M.C. Buydens. Facilitating the application of Support Vector Regression by using a universal Pearson VII function based kernel. *Chemometrics and Intelligent Laboratory Systems*, 81(1):29–40, 2006.

[UPSB+] Xabier Ugarte-Pedrero, Igor Santos, Pablo G Bringas, Mikel Gastesi, y José Miguel Esparza. Semi-supervised learning for packed executable detection. En *Network and System Security (NSS), 2011 5th International Conference on, pages=342–346, year=2011, organization=IEEE*.

[UPSB11] Xabier Ugarte-Pedrero, Igor Santos, y Pablo Bringas. Structural feature based anomaly detection for packed executable identification. *Computational Intelligence in Security for Information Systems*, páginas 230–237, 2011.

[VAJ07] R. Vogt, J. Aycock, y M. Jacobson. Army of botnets. En *Proceedings of the 2007 Network and Distr. System Sec. Symposium (NDSS 2007)*, páginas 111–123. Citeseer, 2007.

[Vij07] Jaikumar Vijayan. Unix Admin Pleads Guilty to Planting Logic Bomb. *PCWorld, September*, 2007.

[VIM04] ISO VIM. International vocabulary of basic and general terms in metrology (VIM). *International Organization*, 2004:09–14, 2004.

[VN49] John Von Neumann. Theory and organization of complicated automata. *Burks (1966)*, páginas 29–87, 1949.

[VSB] R. Villamarín-Salomón y J.C. Brustoloni. Bayesian bot detection based on DNS traffic similarity. En *Proceedings of the 2009 ACM symposium on Applied Computing, pages=2035–2041, year=2009, organization=ACM*.

[WAD+85] C.J. Willmott, S.G. Ackleson, R.E. Davis, J.J. Feddema, K.M. Klink, D.R. Legates, J. O'Donnell, y C.M. Rowe. Statistics for the evaluation and comparison of models. *Journal of geophysical Research*, 90(C5):8995–9005, 1985.

[Weh07] S. Wehner. Analyzing worms and network traffic using compression. *Journal of Computer Sec.*, páginas 303–320, 2007.

[Wil08] Clay Wilson. Botnets, cybercrime, and cyberterrorism: Vulnerabilities and policy issues for congress. DTIC Document, 2008.

[WWACZ07] Ping Wang, Lei Wu, Baber Aslam, y Cliff C. Zou. An advanced hybrid peer-to-peer botnet. En *USENIX Workshop on Hot Topics in Understanding Botnets (HotBots'07), 2007*. 2007.

[WWACZ09] Ping Wang, Lei Wu, Baber Aslam, y Cliff C. Zou. A systematic study on peer-to-peer botnets. En *Proceedings of 18th Internatonal Conference on Computer Communications and Networks, 2009. ICCCN 2009.* 2009.

[WY11] A. Wilson y M. York. Perceived or Real Risks Using Smartphones. *ASSOCIATION OF BUSINESS INFORMATION SYSTEMS*, página 79, 2011.

[XYA+08] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, y I. Osipkov. Spamming botnets: Signatures and characteristics. *ACM SIGCOMM Computer Communication Review*, 38(4):171–182, 2008.

[Yf06] C. Yong-feng. Causes and Countermeasures of Fraud Cases by SMS. *Journal of Beijing People's Police College*, 2:017, 2006.

[YZA08] Wei Yan, Zheng Zhang, y Nirwan Ansari. Revealing packed malware. *Security & Privacy, IEEE*, 6(5):65–69, 2008.

[ZHH+] Jianwei Zhuge, Thorsten Holz, Xinhui Han, Jinpeng Guo, y Wei Zou. Characterizing the IRC-based Botnet Phenomenon. En *Reihe Informatik, year=2007, publisher=Pace University, White Plains, NY, url =http://honeyblog.org/junkyard/reports/botnet-china-TR.pdf,*.

[Zim80] Hubert Zimmermann. OSI reference model–The ISO model of architecture for open systems interconnection. *Communications, IEEE Transactions on*, 28(4):425–432, 1980.

[ZSH] Y. Zeng, K.G. Shin, y X. Hu. Design of SMS commanded-and-controlled and P2P-structured mobile botnets. En *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks, pages=137–148, year=2012, organization=ACM.*

[ZWHZ] H. Zhang, X. Wen, P. He, y W. Zheng. Dealing with Telephone Fraud Using CAPTCHA. En *Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on, pages=1096–1099, year=2009, organization=IEEE*.

[ZZ12] Yanfei Zhong y Liangpei Zhang. An adaptive artificial immune network for supervised classification of multi-/hyperspectral remote sensing imagery. *Geoscience and Remote Sensing, IEEE Transactions on*, 50(3):894–909, 2012.

Apéndices

Apéndice

A

Grado de similitud de los paquetes relativos a las muestras de tráfico utilizadas

La redacción de esta tesis doctoral ha requerido la obtención de una gran cantidad de muestras de tráfico tanto procedente de *botnets* como de tráfico legítimo. Sin embargo, las dificultades para la obtención de dicha información han sido constantes. Por este motivo, se recoge en este apartado algunas de las métricas empleadas para evaluar la heterogeneidad de las muestras y para disponer de una métrica de comparación entre la complejidad de estas.

Este anexo está estructurado como sigue. La sección A.1 explica la base teórica aplicada para comparar las representaciones de paquetes. La sección A.2 establece las métricas de comparación establecidas en forma de función de distribución además de proponer un escalar normalizado que permita la comparación de la complejidad de las conexiones. Por último, la sección A.3 recoge los elementos más relevantes de este anexo.

A.1 Uso de métricas de similitud para representaciones vectoriales

En este apartado se describen matemáticamente las métricas de correlación utilizadas para ponderar la similitud que existe entre dos muestras. Aunque existen otras métricas —como el coeficiente de correlación de Pearson— la elegida ha sido la similitud del coseno del ángulo que forman las representaciones vectoriales de dos muestras en un espacio n-dimensional de n atributos.

A.1.1 Similitud del coseno: base teórica

En trigonometría, la similitud del coseno —en inglés, *cosine similarity*— es una medida de similitud entre dos vectores basada en la medición del coseno del ángulo que forman en el espacio en que se representan. La teoría dice que el rango de valores que se puede obtener a la hora de calcular el coseno de un ángulo θ ($\cos\theta$) es de entre -1 y 1. El valor absoluto de la función coseno será igual a 1 cuando el ángulo formado por dos vectores sea 0º, es decir, cuando ambos apuntan en la misma dirección (un signo positivo indicará que ambos apuntan, no solo en la misma dirección sino que también en el mismo sentido, mientras que un signo negativo indicará que ambos apuntan en sentidos opuestos en la misma dirección), y teniendo un valor de 0 cuando los vectores sean perpendiculares, es decir, cuando el ángulo formado por estos sea de 90ºal apuntar en direcciones completamente diferentes. A modo de guía, la figura A.1 se pueden ver algunos ejemplos de los signos que obtendrán los senos y cosenos del ángulo que forman diferentes vectores de los cuatro cuadrantes de un espacio bidimensional con el eje de abscisas.

Matemáticamente, el coseno de dos vectores puede ser obtenido a partir de la fórmula euclidiana del producto escalar. En matemáticas, el producto escalar, también conocido como producto interno o interior (en inglés, *dot product*), es una operación definida sobre dos vectores de un espacio euclídeo cuyo resultado es un número o escalar. Esta operación permite explorar algunos conceptos de la geometría euclidiana tradicional como las longitudes, los ángulos o la ortogonalidad en dos y tres dimensiones, aunque también puede definirse en espacios euclídeos de dimensiones mayores a tres, tal y como se va a aplicar en las páginas que siguen.

Sean $\vec{a}$ y $vecb$ dos vectores de n dimensiones, el producto escalar de $\vec{a}$ y

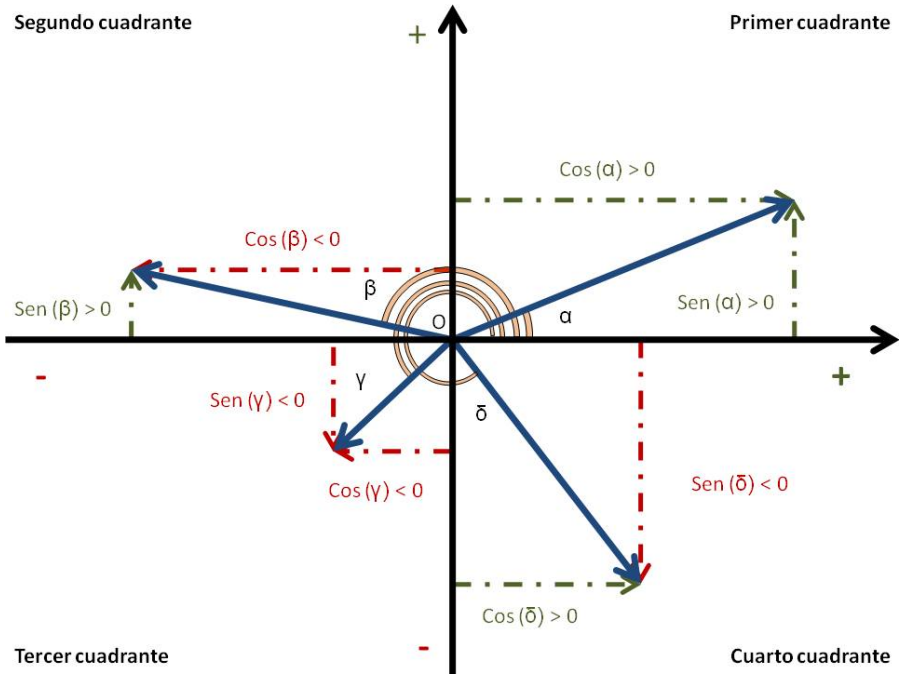


Figura A.1: Determinación del signo del seno y del coseno del ángulo que forman diferentes vectores con el eje de abscisas (o eje X).

$\vec{b}$, expresado $\vec{a}\cdot\vec{b}$, se define como el producto de sus módulos por el coseno del ángulo que forman estos dos vectores:

$$\vec{a}\cdot\vec{b} = |\vec{a}||\vec{b}|\cos\theta \tag{A.1}$$

De dicha ecuación se puede despejar el $\cos\theta$ para obtener:

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|} \tag{A.2}$$

Teniendo en cuenta que la expresión analítica del producto escalar entre dos vectores n-dimensionales se puede expresar como un producto matri-

cial:

$$\begin{aligned}\vec{a}\cdot\vec{b} &= \begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_n \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix} = \\ &= a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3 + \cdots + a_n \cdot b_n = \\ &= \sum_{i=1}^{i=n} (a_i \cdot b_i)\end{aligned} \tag{A.3}$$

Si además sabemos que el módulo de un vector $\vec{v}$ se calcula como la raíz cuadrada de la suma de los cuadrados de sus n componentes:

$$\vec{v} = \sqrt{\sum_{i=1}^{i=n} (v_i)^2} \tag{A.4}$$

Por tanto, dados dos vectores de atributos $\vec{a}$ y $\vec{b}$, podremos definir la similitud del coseno entre el ángulo que forman como:

$$\cos\theta = \frac{\sum_{i=1}^{i=n} (a_i \cdot b_i)}{\sqrt{\sum_{i=1}^{i=n} (a_i)^2} \cdot \sqrt{\sum_{i=1}^{i=n} (b_i)^2}} \tag{A.5}$$

Habitualmente, en las técnicas utilizadas en el análisis de texto, normalmente los vectores de atributos $\vec{a}$ y $\vec{b}$ son, por lo general, los vectores frecuencia de aparición de cada término dentro de los documentos. Así, geométricamente, la similitud del coseno puede ser vista como un método de normalización de la longitud del documento durante la comparación en un espacio n-dimensional, donde cada uno de los términos encontrados corresponde a una dimensión diferente, siendo la frecuencia del mismo la posición relativa ocupada en esa dirección.

De esta manera, un resultado de $\cos\theta = -1$ indicaría que los vectores se encuentran en la misma dirección pero orientados en sentidos completamente opuestos, mientras que un valor resultante de 0 indica, por lo general, la total independencia de los mismos, entendiendo como tal la ortogonalidad de los vectores. En los casos de recuperación de información, el rango de similitud del coseno de dos documentos estará siempre comprendido entre $[0,1]$. Esto se deriva del hecho de que las frecuencias de

Tabla A.1: Ejemplos de los valores que se obtendrían a la hora de calcular la similitud del coseno de los vectores representados. Se entiende que el valor del ángulo α es de aproximadamente 30º.

Vector 1 ($\vec{v_1}$)	**Vector 2** ($\vec{v_2}$)	$cos(\vec{v_1}, \vec{v_2})$
Eje OX	Eje OX	1
Eje OY	Eje OY	1
Eje OX	Eje OY	0
$\vec{s}$	$\vec{t}$	1
$\vec{s}$	$-\vec{t}$	1
$\vec{s}$	$\vec{v}$	≈ 1

aparición de los términos encontrados no son nunca negativas, de lo que, geométricamente, se concluye también que el ángulo entre dos vectores de frecuencia no puede ser mayor de 90. En la tabla A.1, se pueden ver algunos ejemplos de los valores que obtendrán los vectores que se representan en la figura A.2.

De esta manera, mediante el cálculo del coseno del ángulo de dos vectores, se puede determinar si dos vectores apuntan en aproximadamente la misma dirección. Esta técnica es a menudo utilizada para comparar los documentos en la minería de texto (o *text mining*[1] en inglés), además de ser utilizada para medir la cohesión dentro de las agrupaciones en el ámbito de la minería de datos. Se trata de una métrica que también ha sido utilizada por el autor en el marco de otras investigaciones relacionadas con la detección de *malware* [SBN+10b, SSL+, SDB+, SBUPB13, SBS+]. En dichos trabajos, se generaban como modelos de representación de las aplicaciones vectores que agrupaban la frecuencia de aparición de cada secuencia de *opcodes*[2] en un ejecutable dado.

[1] *Text mining* es la voz inglesa más ampliamente utilizada para definir el concepto de la minería de datos aplicada a los textos. Por tanto, se refiere al proceso de obtener información de alta calidad a partir del texto derivada de la detección de patrones y/o tendencias a través, por ejemplo, de herramientas de aprendizaje estadístico. Entre las tareas típicas de la minería de texto se incluyen: la categorización y agrupación del texto, así como la la producción de taxonomías.

[2] Los *opcodes* son los códigos operacionales de las instrucciones de bajo nivel de un ejecutable ya desensamblado.

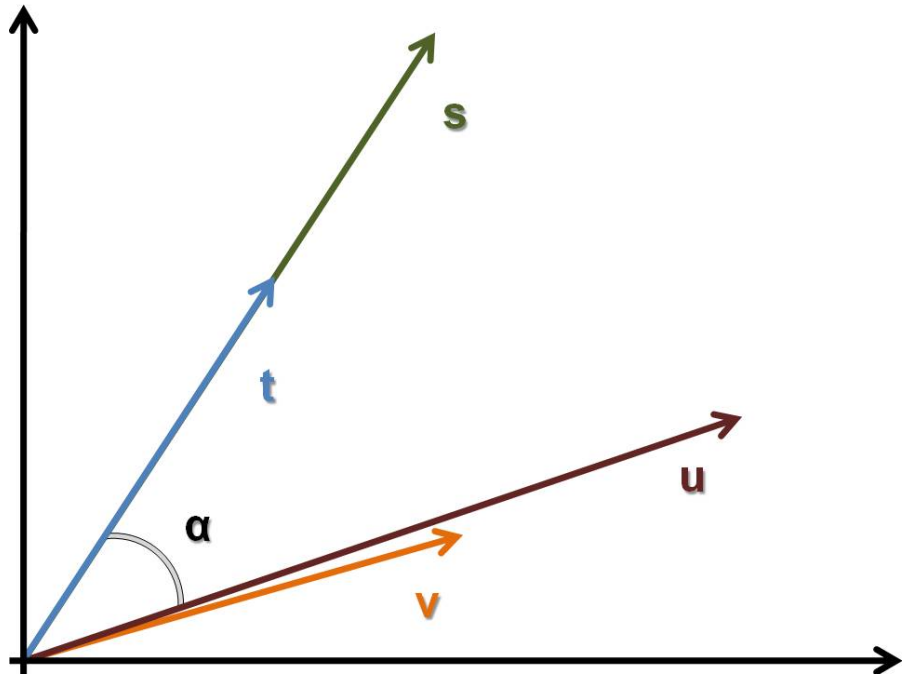


Figura A.2: Representación en un espacio bidimensional de cuatro vectores $(\vec{s}, \vec{t}, \vec{u}, \vec{v})$ presentes en el primer cuadrante.

A.2 Adaptación de las métricas al proceso de modelado

Hay que señalar que el dominio de las similitudes obtenidas es $[0, 1]$. Esto es debido a que los vectores utilizados en esta investigación no presentan en ningún caso valores negativos. Por ejemplo, suponiendo que las muestras de tráfico pudieran ser representadas a partir de vectores bidimensionales, estos ocuparían un espacio perteneciente al primer cuadrante tal y como se muestra en la figura A.2.

Sin embargo, en dicha figura se muestran únicamente representaciones bidimensionales. En el caso que nos ocupa, cada una de las representaciones utilizadas tendrá tantas dimensiones como atributos hayan sido utilizados para modelarla, lo que hace imposible la representación gráfica de los vectores, a pesar de que el proceso de comparación es idéntico.

Tabla A.2: Matriz de resultados de la comparación de n muestras de tráfico.

Muestras	M_1	M_2	M_3	…	M_{n-1}	M_n
Muestra 1	1,00	CS_{12}	CS_{13}	⋯	$CS_{1(n-1)}$	CS_{1n}
Muestra 2		1,00	CS_{23}	⋯	$CS_{2(n-1)}$	CS_{2n}
Muestra 3			1,00	⋯	$CS_{3(n-1)}$	CS_{3n}
…	⋮	⋮	⋮	1,00	⋮	⋮
Muestra $n-1$				⋯	1,00	$CS_{(n-1)n}$
Muestra n				⋯		1,00

A.2.1 Ejecución de las comparaciones

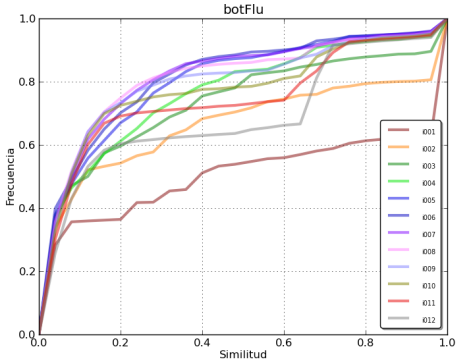
Una vez construidas todas las representaciones de una misma familia, estas se dispondrán en forma de una matriz simétrica de n filas por n columnas. El valor CS_{ij} de cada celda de dicha matriz corresponderá con el valor de la similitud del coseno entre la muestra de la fila i y la muestra de la columna j. Por esta razón, la diagonal principal estará compuesta por unos en su totalidad dado que las muestras comparadas son idénticas. En la tabla A.2 se muestra un ejemplo de los valores que contendrá dicha tabla. Por tanto, para un conjunto de n muestras de una misma familia, se ejecutarán $\sum_{i=1}^{n} i$ comparaciones, siendo al menos n valores iguales a 1,00 al estarse comparando las muestras consigo mismas.

A.2.2 Representación de la complejidad del tráfico de una familia

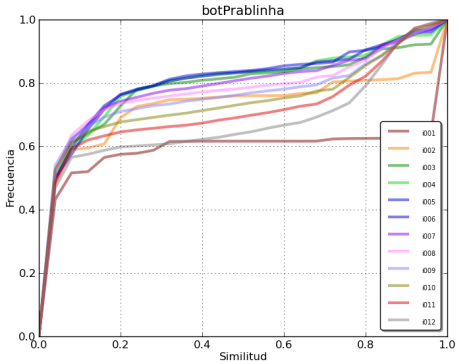
Las matrices generadas en el apartado anterior pueden ser empleadas para representar la complejidad del tráfico de una familia de *malware*. A continuación, se van a mostrar los resultados empleando una función de distribución acumulativa. En estadística, estas funciones representan la probabilidad de que una variable sujeta a cierta ley de distribución de probabilidad se encuentre en la zona de valores inferiores a x.

De la observación de estas gráficas se pueden extraer dos conclusiones principales. La primera es que a mayor número de muestras consideradas en la representación, más dispersión existe entre estas. La segunda sugiere una menor variedad de funcionalidades en la *botnet* Warbot, dado que existen una gran cantidad de representaciones concentradas en la parte final del espectro.

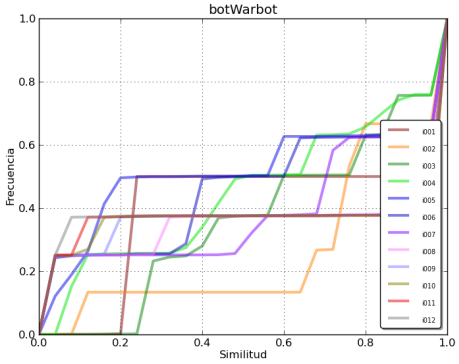
En base a los datos obtenidos en este apartado, podría utilizarse este



(a)



(b)



(c)

Figura A.3: Funciones de distribución de las muestras de tráfico de *Command & Control* de las *botnets* analizadas en esta tesis doctoral: Flu, subfigura A.3(a); Prablinha, subfigura A.3(b); y Warbot, subfigura A.3(c).

enfoque para comparar la complejidad de las interacciones que tiene con la red una aplicación dada. Para ello, podría emplearse, además de la función de distribución acumulada, un escalar con los valores medios obtenidos en la familiad complementados con la dispersión de estos, lo que permitiría representarlos en base a un parámetro normalizado de complejidad de las conexiones.

Una propuesta de indicador de simplicidad $S_{n=k}$ de representaciones de longitud k de una conexión C de la que se disponen j muestras vectoriales se puede representar como sigue:

$$S_{n=k} = (\frac{\sum_{i=1}^{j} CS_{ij}}{j^2})^{1-\sigma} \tag{A.6}$$

Es decir, la media de los valores obtenidos en las j^2 comparaciones ejecutadas entre las j representaciones del conjunto de datos elevado a la desviación típica de estos valores. Este indicador devolverá valores más próximos a uno cuanto más similares sean entre sí las muestras, pero como el objetivo es obtener un indicador de complejidad $C_{n=k}$ calcularemos el inverso de $S_{n=k}$:

$$C_{n=k} = 1 - S_{n=k} \tag{A.7}$$

O lo que es lo mismo:

$$C_{n=k} = 1 - (\frac{\sum_{i=1}^{j} CS_{ij}}{j^2})^{1-\sigma} \tag{A.8}$$

Como se puede ver en la figura A.4, en general la complejidad de la *botnet* Warbot es sensiblemente más baja que las obtenidas para las *botnets* de Flu y Prablinha. Este hecho sugiere una realidad que confirman los experimentos de esta tesis doctoral: los clasificadores encontrarán con más facilidad el tráfico de control procedente de dicha *botnet* que las de Flu o Prablinha dado que los paquetes de esta son más similares entre sí que los observados en estas otras familias. Como dato particular, cabe destacar una leve reducción de la complejidad de Flu y Prablinha (ligeramente más acusada en el caso de la primera *botnet*). Este hecho puede estar relacionado con la reducción de muestras de longitudes mayores observadas en el conjunto de datos por las características ya definidas en el capítulo experimental. Una ampliación del número de muestras de estas longitudes en el conjunto de datos de entrenamiento estabilizaría dichos valores además de dotar a los clasificadores de capacidades de detección adicionales.

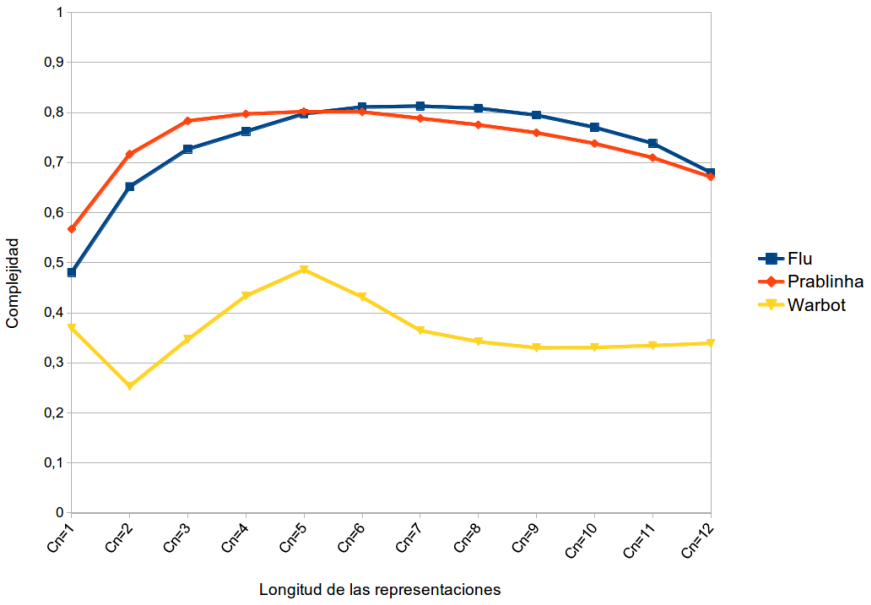


Figura A.4: Representación del grado de complejidad de las representaciones de las *botnets* Flu, Prablinha y Warbot.

A.3 Sumario

En vista la diversidad de las muestras de tráfico generadas, en este anexo se ha optado por establecer métricas de comparación de los diferenetes conjuntos de datos empleados en esta tesis doctoral. Para ello, el doctorando ha adaptado las métricas de similitud empleadas en el pasado en experimentos relacionados con la detección de *malware* con representaciones de códigos operacionales [SDB+, SBN+10a] para adecuarlas a las características propias del caso de estudio.

Así, se ha conseguido un doble objetivo. Por un lado, establecer una función de distribución que permita establecer *thresholds* variables en función de las representaciones empleadas a partir de la función de distribución de similitud de una familia. Por otro, se ha propuesto la adopción de un indicador que permita la comparación de la complejidad de varias conexiones —y de sus comunicaciones— de acuerdo a la similitud de las representaciones observadas con lo que ello puede suponer a la hora de facilitar las tareas de categorización de tráfico.

Apéndice

B

Fichas técnicas de las muestras de *malware* empleadas

En este apéndice se profundizará en las características propias de cada una de las *botnets* empleadas para el desarrollo de esta tesis doctoral que no han sido ya recogidas a lo largo del manuscripto pero que se consideran de interés. Cada uno de los apartados contiene información relativa al despliegue de los servidores de *Command & Control*, el lenguaje o lenguajes de programación utilizados o algunas de sus funcionalidades así como algunas capturas de pantalla relativas a sus interfaces de usuario. Por tanto, en la sección B.1 se analiza la *botnet* Flu. En la sección B.2 se estudia el funcionamiento básico y algunas características relevantes de la *botnet* Prablinha. Por último, en la sección B.3 se recogen las características más relevantes de la *botnet* Warbot.

B.1 Flu

Flu[1] es un troyano diseñado para permitir la construcción de una *botnet*. Ha sido diseñado con una arquitectura de cliente servidor, donde el servidor es

[1] www.-flu-project.com

un pequeño ejecutable programado en C# que tratará de infectar cualquier equipo con un sistema operativo de Windows para hacerse con el control de la máquina que lo hospeda.

B.1.1 Características

El cliente está desarrollado en PHP y se ejecuta en un servidor web Apache y tiene como objetivo suminsitrar las órdenes y comandos a todas las máquinas infectadas por Flu en Internet.

La información referente al estado de la infección puede ser accedida en cualquier momento a través de una interfaz gráfica en HTML y PHP utilizando cualquier navegador convencional con *javascript* activado. El objetivo de dicho proyecto es permitir que aquellos usuarios que quieran aprender a manejar y desarrollar herramientas de control remota se puedan beneficiar de los conocimientos ya desarrollados por terceros. Adicionalmente, satisfacen otro objetivo paralelo: despertar la concienciación entre los usuarios de los peligros del *malware* demostrando la facilidad con la que se puede desarrollar herramientas con una carga maliciosa con una gran capacidad de daño, a través de la publicación de una aplicación que implementa capacidades de *malware* real.

Para satisfacer estos objetivos educativos, el código fuente del proyecto está convenientemente comentado y compartimentado. Al mismo tiempo, existe también una documentación pública accesible desde la propia página web oficial del proyecto de cara a facilitar las tareas de despliegue de la herramienta incluso por aquellos usuarios menos expertos. La apariencia final del panel de control de la versión de Flu modificada por el equipo de trabajo del S3lab se muestra en la figura B.1. Las características básicas del despliegue de Flu se encuentran recogidas en un documento hecho público por Juan Antonio Calles y Pablo González [CG11] en la *Flu Trojan's User Guide* con explicaciones y gráficos sobre cómo proceder en cada paso.

- **Conexión a la base de datos**. La estructura de Base de Datos de Flu en la versión utilizada no estaba concebida como para ser empleada como parte de una aplicación real: apenas contaba con un par de tablas, una donde se listaban los usuarios de la *botnet* —con nombres de usuario y contraseñas almacenadas en texto plano— y una segunda en la que se almacenaban los detalles de las máquinas conectadas a la propia red.

- **Canales de *Command & Control***. La publicación de nuevos coman-

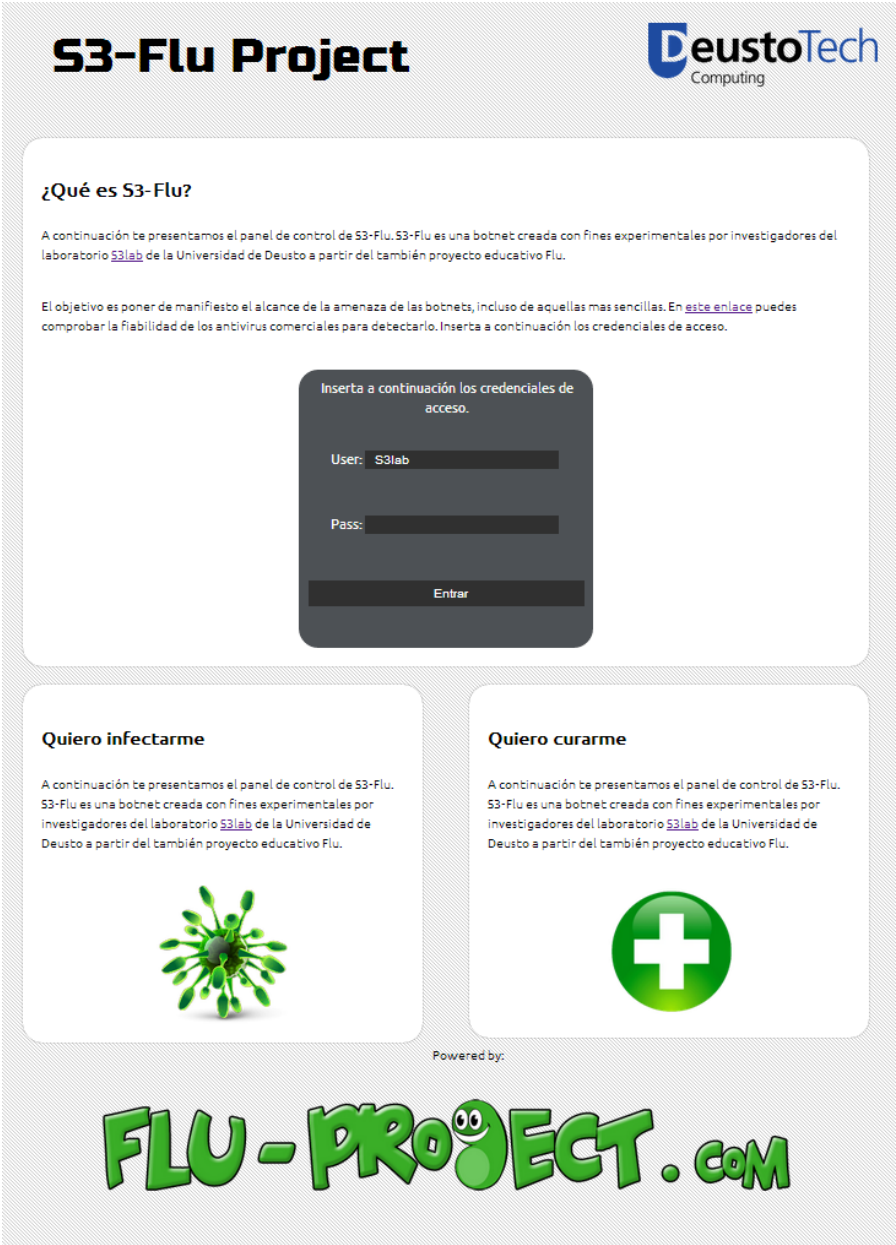


Figura B.1: Captura de pantalla de la pantalla de *login* de Flu. Es una versión modificada del trabajo ya realizado por Flu Project y adaptada para los diferentes experimentos realizados por el equipo del S3lab.

dos en Flu se puede realizar empleando un navegador web a través de una sencilla interfaz web que permite al *botmaster* conectarse y gestionar la misma. Por un lado, la víctima infectada solicita un fichero *.xml* que contendrá el listado completo de comandos e instrucciones a ejecutar y que está almacenado en el servidor web. Una ve que la víctima haya ejecutado los comandos, devolverá las pertinentes respuestas al servidor de control de la *botnet*. En este caso, toda la información enviada o recibida es cifrada por defecto empleando el algoritmo de cifrado simétrico AES con una clave de 128 bits. En la interfaz web, el *botmaster* será capaz de monitorizar el funcionamiento de todos y cada uno de los nodos infectados además de tener la posibilidad de lanzar, en cada uno de ellos, una *shell* de comandos desde la que dar instrucciones más personalizados a dicha máquina. En otra pestaña, las órdenes grupales pueden ser transmitidas seleccionando estas entre ataques ya codificados por defecto o mediante el lanzamiento de comandos en la consola de las víctimas. Cabe destacar en este punto que, de tener instalada la Powershell, también se permitirá la ejecución de este tipo de comandos.

- **Creación del programa *zombie***. La generación de *bots* tiene lugar por medio de un motor de generación automática que recibe como parámetro la URL donde estará localizado el servidor de la *botnet* junto con la ruta especifica hacia el fichero *.xml* en donde se publicarán los comandos.

Aunque Flu b0.4 está concebido como un proyecto eminentemente educativo, incluye algunas características interesantes incluso para escritores de *malware* actuales. Entre ellas, podemos encontrar las siguientes: ejecución remota de comandos individuales o en grupo, obtención de información y de capturas de pantalla, privilegios para crear cuentas de usuarios como administrador, sustracción de ficheros (aunque por limitaciones de diseño, solamente hasta 3,5MB), desactivación del *firewall* de Windows o funcionalidades de *keylogger* entre otras. Adicionalmente, también existe una versión más compleja de Flu orientada a la lucha contra la pederastia y conocida como Flu-AD, abreviando *Flu-Anti Depredadores*. El prototipo, presentado en la No cON Name de 2011 implementaba capacidades de *rootkit*, captura de sonido y webcam, funciones de *crypter* y mutación para evitar la detección por parte de los motores de antivirus y diferentes utilidades para recuperar nombres de usuario y contraseñas de las cuentas de los supuestos pedófilos.

B.1.2 Otras consideraciones y ficha técnica

A pesar de todas las características anteriormente mencionadas, un aislamiento adecuado de las conexiones realizadas por los nodos infectados podría derivar en una rápida identificación del servidor en el que el servidor de la *botnet* está instalado. Además, dadas las limitaciones de seguridad de la base de datos de algunas de las versiones, debería ser relativamente sencillo tomar control de la *botnet* accediendo a la tabla de usuarios de la base de datos para añadir o modificar manualmente los usuarios con privilegios para manipularla o modificar las instancias de los *bots* conectados y poder así monitorizarla o incluso ponerla fuera de servicio. Al mismo tiempo, conviene puntualizar que, siendo una plataforma creada con fines educativos, los propios desarrolladores de Flu Project proveen de una herramienta de desinfección que elimina cualquier traza de las infecciones por defecto. De cualquier manera, es cierto que dichas aplicaciones solo serán de utilidad si el ejecutable principal mantiene el nombre de *flu.exe*, dándoles poca utilidad en un entorno real.

B.2 Prablinha

Prablinha[1] se puede definir como otro troyano diseñado para permitir la construcción de una *botnet* y hecho *open-source* en diciembre de 2010 también con fines educativos para mostrar lo fácil que es realizar ataques distribuidos de denegación de servicio. Ha sido igualmente diseñada con una arquitectura cliente-servidor en donde los *zombies* han sido codificados en C# y son generados por un constructor también diseñado en C#. En cuanto a las herramientas de gestión, estas están desarrolladas en PHP y se pueden ejecutar en un servidor web Apache. El sistema ha sido concebido para ejecutar ataques de denegación de servicio distribuida HTTP, UDP y TCP, así como para proveer al atacante de una *shell* remota para la ejecución de ficheros y aplicaciones en el ordenador infectado sin el permiso de su legítimo propietario.

B.2.1 Características

En lo que respecta a la accesibilidad del código, tanto los *zombies* como el generador están escritos en C#. Aunque el código no esta completamente comentado, el proyecto está diseñado de una forma estructurada que va en

[1] http://www.indetectables.net/foro/viewtopic.php?t=29086

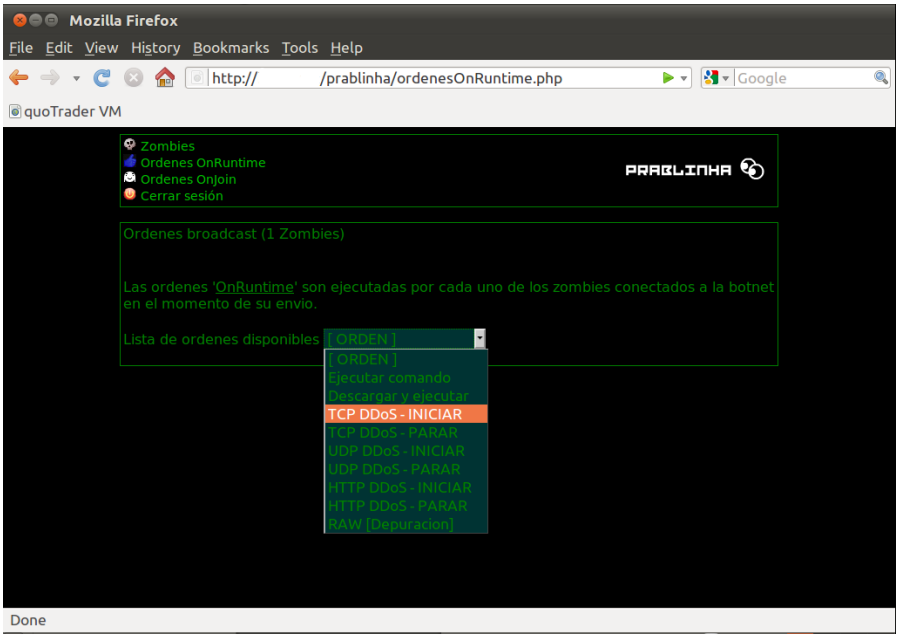


Figura B.2: Captura de pantalla del panel de control de Warbot. Desde aquí se puede proceder a la ejecución de los diferentes ataques de denegación de servicio implementados por Prablinha así como acceso a *shells* remotas. También se pueden monitorizar otras tareas de interés para el administrador, como los detalles del sistema, aspectos relativos a la geolocalización u órdenes emitidas al inicio de la infección o en cualquier momento.

consonancia con el tamaño de la solución. Existen dos proyectos diferentes dentro de esta: el primero de ellos hace referencia al funcionamiento de los canales de C&C mientras que el segundo se centra en la definición de los ataques. La configuración se puede realizar manualmente a través de la modificación del fichero *Config.cs* en el proyecto *CnC*.

El despliegue básico de la *botnet* con fines experimentales se puede resumir en la instalación y ejecución de los *frameworks* en los nodos a infectar y la preparación de los servidores PHP y MySQL. Estas últimas tareas se pueden dejar atadas con aplicaciones como HeidiSQL o PHPMyAdmin. La apariencia final del panel de control se muestra en la figura B.2.

Podemos decir que Prablinha es un troyano concebido en tres capas:

1. El troyano ejecutable ha sido codificado en C# requiere de la instalación del *framework* de .NET en las máquinas infectadas.

2. El conjunto de ficheros .php que, instalados en un servidor web, proveerán de la interfaz de usuario al *botmaster*.

3. La base de datos MySQL instalada en dicho servidor en donde se almacena la información relativa a las máquinas infectadas.

Se trata de un *malware* que permite la configuración de *zombies* para la ejecución de los siguientes ataques: TCP, UDP y HTTP **DDoS**, geolocalización de las víctimas identificando su dirección IP, ejecución remota de comandos a través del navegador web y descarga y ejecución de ficheros remotos.

- **Conexión a la base de datos**. La estructura de la base de datos de esta versión abierta de Prablinha, tal y como ocurría con Flu, no es tampoco profesional. Incluye cinco tablas: una para los usuarios de la *botnet* en la que los nombres de usuarios y las contraseñas son almacenadas en texto plano; otra para el almacenamiento de los detalles de cada una de las máquinas infectadas; una para almacenar las respuestas de las víctimas y dos para describir el listado de acciones a ejecutar por los *zombies*, existiendo dos posibilidades: *on join*, para configurar las primeras acciones que llevará a cabo un *bot* en el momento de darse de alta como nodo infectado y *on runtime* para definir las actuaciones que tendrán que ejecutar los nodos en ese momento. Otro aspecto a señalar es la forma en que los usuarios se pueden conectar a la base de datos. Tanto el nombre de usuario como la contraseña —que no es la misma que se ha de utilizar para loguearse está codificada tal cual en la página PHP del servidor, con lo que si no se desarrollan políticas adecuadas de control de acceso y gestión de privilegios un usuario externo podría terminar accediendo a todo el contenido de la base de datos con permisos de administración. Por otro lado, la instalación de las tablas consiste en la ejecución de un fichero *db.sql* que creará las estructuras necesarias en la base de datos. Nótese también que las credenciales de acceso por defecto son *thesurthesur*.

- **Canales de *Command & Control***. Sobre su canal de Command & Control, hay que mencionar aunque la versión más reciente utiliza un navegador web como gestor de los comandos, versiones anteriores utilizaban clientes IRC para las comunicaciones. Las versiones actuales son capaces de llevar a cabo estas modificaciones empleando los protocolosHTTP y HTTPS a través de canales cifrados (SSL).

- **Creación del programa *zombie***. Para crear y configurar el *zombie* que sea finalmente distribuido y ejecute la infección en sí misma, el generador deberá cargar lo que se conoce previamente como el ADN del

código del *bot*. En el caso de Prablinha, esta utilidad de generación mostrará una pantalla que permitirá al usuario elegir algunas opciones de configuración del *bot*:

- *Regedit Name*. Nombre de la clave de registro usada para la ejecución del *zombie* tras cada reinicio.
- *Install Name*. Nombre usado para almacenar la copia del programa en el disco duro.
- *Loader Name*. Nombre usado para almacenar la copia del lanzador del *malware* en el disco duro. Este lanzador sera el encargado de relanzar a ejecución el programa en el caso de que por cualquier eventualidad haya sido cerrado.
- *URL del panel de C&C*. En esta URL estarán situadas las herramientas de administración de la *botnet*. Como ya se ha dicho, tanto los protocolos HTTP como HTTPS pueden ser utilizados para evitar la detección de la red.

En este punto hay que matizar que es importante no introducir como nombres de los ficheros ninguno que corresponda con algún proceso ya existente en el sistema. Por último, tras completar estos pasos, el *bot* será generado y quedará listo para ser desplegado bajo el nombre de *ZombieX.exe* (en donde *X* corresponde al número de la variante generada).

En este punto, cabe destacar que Prablinha almacenada cifradas las cadenas de texto sobre entradas del registro o nombres de los lanzadores que incluye en los ejecutables finales. Esto se realiza a través de las llamadas a dos métodos públicos de la clase *Algorithm* definida en el *Encryption.cs* que encapsulan el procedimiento de cifrado, tal y como se muestra en la figura B.3.

Este procedimiento de cifrado simétrico no es un estándar de cifrado como podría ser AES, utilizado ampliamente incluso para cifrado de carácter militar [Kya], si no que se trata de una sencilla implementación de un cifrado básico de sustitución y transposición del que se obtendrá un criptograma de idéntica longitud al texto en claro original. En la figura B.4 se puede ver la sencilla implementación de este algoritmo.

Sabiendo el funcionamiento del mismo, es trivial anticipar cuáles serán los valores que tomarán algunos de los parámetros que serán almacenados en el fichero final. En la tabla B.1 se incluyen los criptogramas que se

```
1          public static string Encrypt(string str)
2          {
3              return Algorithm(str, str.Length);
4            }
5
6          public static string Decrypt(string str)
7            {
8              return Algorithm(str, -1 * str.Length);
9          }
```

Figura B.3: Llamadas públicas a los métodos de cifrado de Prablinha.

```
1        private string Algorithm(string str, int offset)
2          {
3            string charset = "a0b1c2d3e4f5g6h7i8j9k:l/m.n_o|p
                 qrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
4
5              string tmp = string.Empty;
6
7              for (int i = 0; i < str.Length; i++)
8              {
9                 int pos = charset.IndexOf(str[i].ToString());
10
11                if (pos == -1)
12                {
13                    tmp += str[i].ToString();
14                    continue;
15                }
16                offset = offset % charset.Length;
17                int entryPoint = pos + offset;
18                if (entryPoint <= 0)
19                    entryPoint = entryPoint + charset.Length;
20                if (entryPoint >= charset.Length)
21                    entryPoint = entryPoint - charset.Length;
22
23                tmp += charset[entryPoint].ToString();
24             }
25             return tmp;
26           }
```

Figura B.4: Algoritmo de cifrado de cadenas de texto de Prablinha.

Tabla B.1: Criptogramas obtenidos tras ser cifrados con el algoritmo por defecto de Prablinha.

Campo	Texto plano	Texto cifrado
Reg. Key Name	prablinha	xA45 .t4
Install Name	prablinha.exe	BE67t|x_6w:K:
Loader Namer	svchos.exe	CFhmwCtjHj
Control Panel	http://127.0.0.1	zUUPGIIpqAKoKoKp
	/prablinha/	IPS_|HBLz_I

podrán encontrar en los ficheros maliciosos, pudiendo ser utilizados estos como parámetros de un sistema de detección rudimentario de ficheros infectados por Prablinha.

B.2.2 Otras consideraciones

En cuanto a la actualización del proyecto, está concebido para ser modificado en pequeños pasos y ofrecer los cambios a la comunidad que los gestiona sin que sea necesario que los usuarios estén permanetemente actualizando sus muestras. Cuenta con un pequeño manual en cinco páginas en el que se explica cómo desplegar la búsqueda, escaso para conocer mayores detalles pero suficiente para entender su funcionamiento básico. Su desarrollo quedó paralizado a partir de la tercera semana de noviembre de 2011 y la URL en la que se ofrecían los nuevos cambios resultaría desde entonces inaccesible[1], aunque previamente el autor, que firmaba como *TheSur* hizo el proyecto público en el foro ya mencionado.

B.3 Warbot

La *botnet* Warbot es una *botnet* relativamente conocida en la red y aparecida en 2010. Una primera búsqueda en servicios como Google o Bing proveerá al lector de una primera aproximación a la gran cantidad de contenidos multimedia que se han escrito sobre ella incluyendo videotutoriales en Youtube sobre cómo desplegarla y algunas funciones básicas de administración.

[1] http://itsm3.com/Aplicaciones/prablinha/

B.3.1 Características

La principal característica de Warbot es la ejecución de ataques de denegación de servicio distribuida TCP, UDP y HTTP.

- **Canales de *Command & Control***. El canal de Command & Control es una sencilla interfaz en tonos oscuros con un menú superior desde el que llevar a cabo las diferentes tareas para las que ha sido programada: la consola, un lanzador de los programas, una pestaña de estadísticas y tres ventanas para la ejecución de los ataques de denegación de servicio ya mencionados. Implementa una filosofía *push* y la apariencia final del panel de control se muestra en la figura B.5.

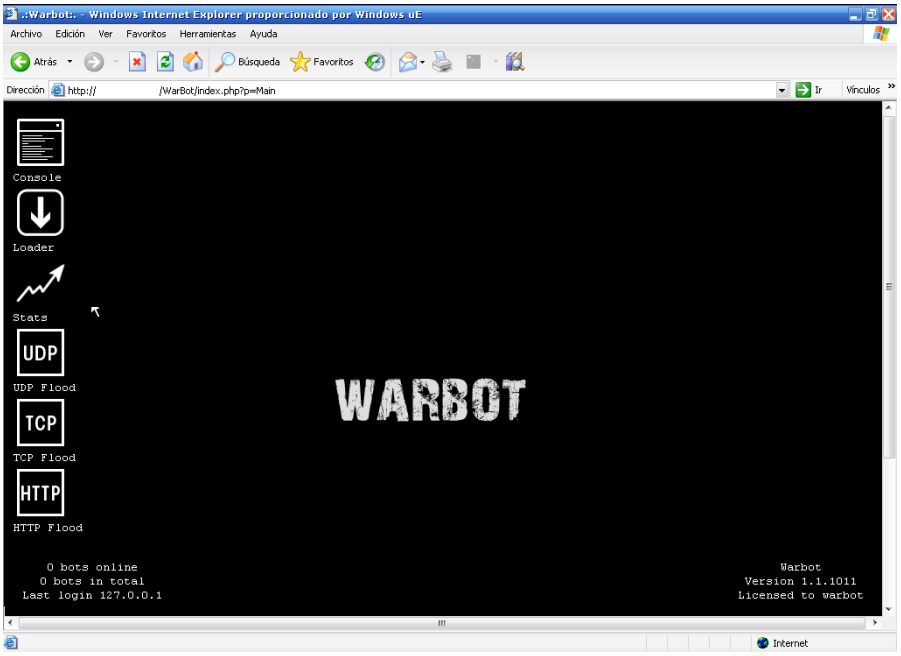


Figura B.5: Captura de pantalla del panel de control de Warbot. Desde aquí se puede proceder a la ejecución de los diferentes ataques de denegación de servicio implementados por Warbot así como acceso a *shells* remotas. También se pueden monitorizar otras tareas de interés para el administrador, como los detalles del sistema.

- **Creación del programa *zombie***. La creación del programa troyanizado se realiza a través de un sencillo *builder* que presenta una sencilla interfaz de consola que solicita los siguientes datos: la URL del servidor HTTP donde se aloja el panel de control, información relativa al

User Agent que enviará el *software* en las peticiones HTTP que realice, la frecuencia de actualización, una cadena de texto aleatoria como contraseña y dos rutas de instalación. Tras introducir esta información, se creará un binario con el que realizar las infecciones.

B.3.2 **Otras consideraciones**

En algunos círculos *underground*[1] se ha presentado esta *botnet* como «*de comportamiento inestable*» a partir de los 400 nodos infectados. En resumidas cuentas, se trata de una *botnet* de escasa complejidad y con no demasiadas funcionalidades, pero utilizada en entornos *underground* reales.

[1] Por entornos *underground* se entienden aquellos foros en los que se detalla la utilización de este tipo de herramientas. El caso de www.anarchyforums.net es un ejemplo.

Índice alfabético

ACK, 54
adware, 32
AES, 192, 196
Anonymous, 38, 43
applet, 55
aprendizaje automático
    supervisado, 90–100
árbol de decisión, 94, 95
Arbor Networks, 140
ARPANET, 23
ataque Sybil, 59
Axel Gembe, 48

base de datos, 78
Bitcoin, 8, 39, 40
blacklisting, 60
bosque aleatorio, *véase* Random Forest

C4.5, 95–96
CaaS, 9
    Crime as a Service, 9
ciberdefensa, 34
ciberespionaje, 39
clasificador bayesiano ingenuo, *véase* Naïve Bayes
click-frauding, 30, 40
clustering, 94
Command & Control, 54, 78
complejidad de Kolgomorov, 57
computación distribuido, 40
computación GPU, 41
cookies, 51
cookies, 32
CVV, 38

darknet, 53, 84
DDoS, 37, 38, 47, 56
detección, 53
DeustoTech Computing, 190
distancia euclidiana, 102
DNS, 56
DoS
    Denial of Service, 37

eMule, 44
espionaje económico, 39
Europol, 34
exploit, 27

Facebook, 50
fast-flux, 46
FBI, 34
fichero
    .arff, 73, 82, 84
    .au3, 50
    .com, 23
    .cpp, 48

.cs, 194, 196
.csv, 73
.exe, 23
.pcap, 81, 83
.pdf, 37
.pdml, 82, 83
.php, 194
.sql, 195
.xml, 79, 83, 192
firewall, 31
firma, 55
Free Software Foundation, 48
FTP, 31, 48, 51

GNU GPL, 48
Google AdSense, 40

HaaS, 9
Hacking as a Service, 9
Half Life, 48
hash, 59
herramientas
shell, 192
Airsnort, 57
Apache, 190, 193
BotSniffer, 57
Hide Window, 47
mIRC, 47
PHPMyAdmin, 194
Powershell, 192
WEKA, 81
Weka, 101, 108, 111
Wireshark, 79, 81, 83
hiperplano, 91
hoax, 28, 32
honeypot, 53, 59
Hotmail, 36
HTML injection, 51
HTTP, 43, 46, 57
HTTPS, 43

IDS, 79
in-the-cloud, 25
ingeniería social, 42, 47
IPv6, 51
IRC, 34, 43, 45, 47–49, 57
ISO, 64
ISP, 50

K-Nearest Neighbour (KNN), 101
K2, 99–100
Kaspersky, 36
Kazaa, 44
keylogger, 51
keylogger, 38, 192

lenguajes (programación)
AutoIt v3, 50
BASIC, 50
C++, 48, 51
C#, 190, 193
ensamblador, 48
HTML, 190
javascript, 190
MySQL, 194
PHP, 51, 78, 190, 193

malware
muestras
Warbot, 74
Zbot, 40
malware
Flu, 111
muestras
Agobot, 48, 57
AIDS-Trojan, 27
Bagle, 24
Blaster, 24
Conficker, 26
Creeper, 23

Elk Cloner, 23
Flu, 51, 81, 187, 189
Gaobot, 48
GTbot, 47
Happy, 24
Hlux/Kelihos, 38
ILoveYou, 24
IMMONIA, 50
Jerusalem, 23
Mariposa Botnet, 37
Mydoom, 24
Netsky, 24
Prablinha, 51, 79, 81, 187, 193, 198
Rbot, 48
Reaper, 23
Sasser, 24
SDBot, 36
SDbot, 48, 57
SpyBot, 57
Stormnet, 44, 49
Stuxnet, 26
Trojan-Ransom, 26
Trojan.Peacomm, 42, 44, 49
Waledac, 36
Warbot, 81, 187, 198
Zeus, 26, 40, 51
tipos
bombas lógicas, 28, 33
botnets, 34, 37, 53
bots, 28
gusano, 33
gusanos, 28, 57
spyware, 28, 31
troyanos, 28, 29
virus, 28
McAfee, 27
minería de texto, 183
Ministerio de Defensa, 39
Multilayer Perceptron, 103
MUSI, 80

Naïve Bayes, 99

P2P, 34, 41, 42, 48, 49
Pastebin, 43, 50
pay-per-click, 40
payload, 28, 66, 75
polimorfismo, 55
POP3, 51
proxy, 30, 86
pull, 45
push, 46

RAM, 29
Random Forest, 96, 97
Redes bayesianas, 97–100
redes sociales
Twitter, 43, 50, 52

S3lab, 190
SMS, 52, 53
SMTP, 34
spam, 34, 58
spammers, 34
SpamHaus, 35
Spread Subsample, 108
SSL, 195
Support Vector Machines (SVM), 91–93
funciones de kernel, 92–93
Symantec, 35

TCP, 31
timestamp
UNIX *timestamp*, 83
timestamp, 68, 83
Tor, 84, 86
Tree Augmented Naïve (TAN), 100
trigger, 33

Valve, 48
vida artificial, 23
Voted Perceptron, 103

Wikileaks, 43

XOR, 26
XSS, 51

Youtube, 198
Yung-Hsun Lin, 33

Esta tesis doctoral se terminó de escribir
en Madrid el 29 de agosto de 2013.