Clasificación supervisada de paquetes procedentes de una *botnet* HTTP

Félix Brezo, José Gaviria de la Puerta, Xabier Ugarte-Pedrero, Igor Santos, Pablo G. Bringas
DeustoTech Computing
Universidad de Deusto
España
Email: {felix.brezo, jgaviria, xabier.ugarte, isantos, pablo.garcia.bringas}@deusto.es

David Barroso
Telefónica Digital
Email: barroso@tid.es

***Resumen*—Las posibilidades que ofrece la gestión de ingentes cantidades de equipos y/o redes está atrayendo a un cada vez mayor número de escritores de *malware*. En este documento, los autores proponemos una metodología de trabajo para la detección de tráfico malicioso procedente de las *botnets*, basándose en el análisis del flujo de paquetes que circulan por la red. Este objetivo se consigue con la parametrización de las características estáticas de esos paquetes para ser analizadas posteriormente con técnicas de aprendizaje supervisado enfocadas al etiquetado de tráfico, de modo que se permita hacer frente de una forma proactiva al gran volumen de información que manejan los filtros actuales.**

***Abstract*—The posibilities that the management of a vast amount of computers and/or networks offer, is attracting an increasing number of malware writers. In this document, the authors propose a methodology thought to detect malicious *botnet* traffic, based on the analysis of the packets flow that circulate in the network. This objective is achieved by means of the parametrization of the static characteristics of packets, which are lately analysed using supervised machine learning techniques focused on traffic labelling so as to face proactively to the huge volume of information nowadays filters work with.**

I. Introducción

Las *botnets* son una de las amenazas más recientes. Los masivos ataques distribuidos de denegación de servicio (DDoS, o *Distributed Denial of Service attacks* en inglés) recibidos por Estonia en 2007 [1] pusieron de manifiesto un cambio de paradigma en lo que a la protección de las infraestructuras críticas informáticas se refiere. Desde entonces, la capacidad de las *botnets* para paralizar servicios en la red, lo que incluye desde servicios gubernamentales a plataformas de pago [2] o entidades bancarias, ha quedado patente, y ya no sólo por sus efectos, sino también por la sencillez de ejecución de estos ataques. Así, empezando por la propia Organización del Tratado del Atlántico Norte (OTAN) y continuando con agencias policiales y mandos militares de los cuerpos de seguridad de países con presencia en los cinco continentes, son muchos estamentos gubernamentales los que se están preocupando cada vez más por hacer frente con solvencia a la amenaza de las ciberarmas [3], en el marco de las cuales podemos identificar a las *botnets*.

La multidisciplinaridad característica de las *botnets* y rápida curva de aprendizaje incluso para personas con conocimientos medios de informática, están dando lugar a un nuevo modelo que explota el concepto de *Hacking as a Service* o *HaaS* [4], por medio del cual un programador de *malware* diseña y codifica aplicaciones *ad-hoc* fáciles de usar y controlar por terceros. En este contexto, una vez dejados atrás en gran medida los canales de Command & Control basados en IRC[1] [5], una de las formas más habituales hoy en dia de presentar interfaces de control consiste en la utilización de servidores web a los que los nodos infectados se conectan para solicitar nuevos comandos. En nuestro caso la muestra de código empleada para la realización de los experimentos es una variación de la *botnet* educativa Flu[2], cuyo canal de *Command & Control* (C&C) está, precisamente, basado en HTTP y HTTPS.

Una vez estudiadas las técnicas utilizadas en el pasado, proponemos un nuevo método de clasificación supervisada de los canales de C&C de las botnet HTTP mediante el uso del tráfico de red.

En resumen las contribuciones derivadas en el trabajo son: (i) Uso de una muestra modificada de una botnet educativa para la generación del tráfico de red. (ii) Obtención de las características relevantes del tráfico de red, tanto procedente de una botnet como de tráfico legítimo. (iii) Clasificación mediante técnicas de aprendizaje supervisado extrayendo dichas características en tráfico legítimo y botnet.

La estructura de este documento está organizada como sigue. La sección II detalla el trabajo realizado por otros autores para la modelización de tráfico. La sección III concreta el alcance de esta investigación describiendo los procesos de captura de datos. La sección IV define las técnicas y métodos empleados para la definición de las capturas. La sección V aglutina el conjunto de experimentos realizados y los resultados obtenidos tras su ejecución. La sección VI avanza algunas de las próximas líneas de trabajo en el área del análisis de tráfico procedente de las *botnets*. Por último, la sección VII recoge las conclusiones extraídas durante la realización de este trabajo.

[1]IRC: protocolo de comunicación en tiempo real basado en texto, que permite debates entre dos o más personas

[2]Flu es un proyecto con fines educativos que pretende dar conocer al público en general los peligros que las *botnets* atañen. La página oficial del proyecto es: http://www.flu-project.com/

II. Trabajo relacionado en la modelización del tráfico de *BOTNET*

Existen diferentes enfoques para la detección de tráfico malicioso procedente de las *botnets*. Algunos de ellos están centrados en la detección de anomalías una vez asumido un flujo de tráfico normal [6], [7]. La detección de anomalías se basa en la definición de un comportamiento estándar de la red frente a determinados parámetros como las características de las conexiones de red, el uso de la CPU o las modificaciones de los sistemas de ficheros. La principal ventaja de este enfoque es que suelen ser eficientes para la detección de nuevos ataques ya que lo habitual es que las infecciones, cualquiera que sea su naturaleza, generen alteraciones en la actividad de la red [8]. Sin embargo, la detección de anomalías lleva consigo un problema: que en el proceso de modelización del tráfico *normal* se filtre tráfico puramente malicioso que distorsione la representación real de la actividad verdaderamente confiable, haciéndose pasar para el detector como un proceso más de los ejecutados confiablemente.

El resto de técnicas de detección se pueden agrupar en función del elemento que monitorizan, en dos tipos: centrados en el análisis de la red (*network-based*) o centrados en el tráfico saliente del nodo (*host-based*). En esta línea, BotSniffer [9] explotaba las diferencias espacio-temporales de las comunicaciones que aparecían en las comunicaciones de las *botnets* contraponiéndolas con las que tienen lugar con el tráfico legitimo. Lo que inspiraba ese modelo era que cada *bot* de una misma red iba a recibir o pornerse en contacto con el servidor a intervalos y frecuencias suficientemente similares como para identificar el tráfico en el seno de los canales IRC. Es éste el enfoque que se quiere aplicar, aunque con otros parámetros, para las redes controladas a través de HTTP. Por otra parte tenemos a Dietrich et ál. [10], crearon una aplicación que haciendo uso del análisis de la red, podían detectar los canales de C&C de una botnet. Para ello recogían los diferentes parámetros del tráfico de red y los diferentes protocolos. Una vez obtenidos estos haciendo uso de una media vinculación de la agrupación jerárquica de las etiquetas y los flujos de C&C podían reconocer el 88% de los canales de control de las botnet entre los 20 más recientes.

Otros autores como Karasiridis et ál. [11] se han centrado en realizar análisis de tráfico de forma pasiva a nivel de la capa de transporte con el fin de no depender de aquella información manejada directamente por la capa de aplicación independientemente de la naturaleza de los datos en sí, mostrándose también claramente efectivos en las antiguas redes basadas en IRC. También hay que contar con las estrategias que tratan de identificar el canal de comunicación para anular así la posibilidad de que los nodos reciban los comandos del *botmaster* o administrador de la *botnet* cuando no sea posible paliar los efectos de la infección en los propios nodos de la red. Es precisamente esto lo que ocurria en marzo de 2012 con Hlux/Kelihos[3] [12]. Esto es debido a la proliferación de técnicas conocidas como *fast flux* que ocultan el origen del canal en *botnets* que implementan la arquitectura *peer-to-peer* [13].

En nuestro caso, la idea es evitar la conexión a un nodo centralizado cuya detección pudiera comprometer el funcionamiento de la totalidad de la red. Para ello se ha diseñado una topología de red que, transparentemente, va rotando el punto en el que las órdenes se van publicando entre la colección de servidores comprometidos disponibles, aprovechándose para ello de que no se hace referencia al C&C por medio de direcciones estáticas, sino a través de nombres de dominio.

Este enfoque independiente del tráfico es el que se pretende desarrollar para *botnets* basadas en HTTP y/o HTTPS, utilizando técnicas de aprendizaje supervisado para el etiquetado de las conexiones en base a parámetros atómicos de las conexiones, pero siempre anonimizando en su totalidad el origen y destino de las conexiones.

III. Alcance de la experimentación

En este contexto, el objetivo del presente estudio consiste en la captura de información relativa a los tamaños de las cabeceras, la frecuencia de las conexiones, etc., con el fin de ser capaces de etiquetar el tráfico procedente de la *botnet*, tanto legítimo como de botnet, en base a esas características propias de los paquetes procedentes de sus canales de C&C, implementando una filosofía ya empleada en el pasado para la detección de intrusiones o la detección de tráfico ilegítimo [14]. Una vez obtenido todo el tráfico, el objetivo principal de la investigación es la capacidad, mediante técnicas de aprendizaje supervisado, de detectar con el menor número de falsos positivos y falsos negativos, los paquetes que proceden de un canal de C&C. Adicionalmente, el lector tiene que tener en cuenta las dificultades que entraña un estudio de estas características desde el punto de vista de la privacidad y de la legalidad, dado que se hace necesario la lectura e interpretación de datos de comunicaciones reales para evaluar las aptitudes del método. Si, además, se añade el hecho de que muchas de estas redes están estableciendo comunicaciones a través de protocolos cifrados como HTTPS que ocultan el contenido ante técnicas basadas en el análisis del contenido [15], el enfoque que aquí se expone cobra un nuevo interés ya que analiza el propio flujo de información en la red y no los contenidos en bruto de la comunicación

En las subsecciones que siguen se define la metodología empleada para la modelización del tráfico de C&C así como las características concretas de la *botnet* utilizada para exponer esta prueba de concepto, que explota las posibilidades de la herramienta de la Universidad de Waikato Weka[4] para emplear diversos algoritmos de clasificación.

[3]Kelihos es una *botnet* originalmente descubierta en 2010 que trata orientada principalmente al robo de *bitcoins* y al envío de correo basura cuyo canal de C&C fue desactivado en marzo de 2012 por investigadores de Kaspersky Labs.

[4]*Weka: Data Mining Software* es una colección de algoritmos de aprendizaje automático para la automatización de tareas de minería de datos: http://www.cs.waikato.ac.nz/ml/weka/

A. Entorno de pruebas

La filosofía de trabajo propuesta en el articulo está basada en un IDS[5] basado en *host*. Es decir, el punto de escucha de tráfico esta situado en los nodos infectados de la red. Así, para el entorno de pruebas se han definido dos tipos de máquinas:

- **La máquina de control**. 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. La instalación de la interfaz de C&C se ha realizado 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 aquéllos perdidos, pudieran introducir sesgos en la clasificación que aparecerían si la instalación se hubiera llevado a cabo en un entorno de laboratorio.
- **Las máquinas infectadas**. 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. 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.

B. Recopilación de información

Para la recolección de información relativa al tráfico generado se ha optado por colocar un programa de *sniffing* como Wireshark[6] en las máquinas infectadas, con el fin de capturar la información enviada y recibida por cada uno de los nodos infectados con Flu. La infección de estas máquinas se ha producido de forma manual, generando externamente un ejecutable cuyo canal de C&C apunta directamente al servidor web sobre el que estará montada toda la estructura de comunicación. En esta línea, las capturas de Wireshark permiten realizar un análisis paquete a paquete del tráfico enviado y recibido en cada uno de los nodos.

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 [7] 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. Esta imposición, implica 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. Sin embargo, los autores consideran que el error introducido es aceptable ya que fueron instados a consultar páginas de uso cotidiano para un ciudadano promedio [16] 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.

TABLA I
10 ATRIBUTOS MÁS RELEVANTES PARA LA CLASIFICACIÓN EMPLEANDO *information gain*.

Atributo	Relevancia
tcp.len	0,031446
tcp.reassembled_in	0,017189
tcp.flags	0,014811
tcp.hdr_len	0,007136
tcp.flags.push	0,005424
http.request	0,005346
tcp.flags.syn	0,004180
http.response.code	0,003739
http.response	0,003552
http.content_length	0,002799

C. Filtrado de protocolos y anonimización del tráfico

Anonimizamos las conexiones al máximo para evitar que se identifique la conexión por parámetros estáticos como la dirección IP, la dirección MAC o los puertos ya que el objetivo consiste en clasificar las conexiones usando otros atributos. Adicionalmente, por las características concretas de las conexiones procedentes de Flu se ha omitido toda referencia a aquellos protocolos no relacionados con la *botnet* en sí como ARP, SIP o STP; para centrarse únicamente en aquellos que sí que utiliza, por construcción, para comunicarse: HTTP, HTTPS, UDP y TCP. De estos atributos, los 10 que se recogen en la tabla I, son los considerados más relevantes para la clasificación utilizando como principal evaluador la divergencia de Kullback-Leibler [17], implementada en Weka bajo *InfoGain*[8].

Al mismo tiempo, se ha considerado definir la duración y la interrelación entre las sesiones como un factor clave para enlazarlas. Puesto que en toda comunicación se generan multitud de peticiones contra un *host* y no una única petición, se ha decidido definir un atributo adicional que interrelacione los paquetes: el tiempo θ. Se ha definido el tiempo θ como

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

[6]Wireshark es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones, para desarrollo de software y protocolos, y como una herramienta didáctica con fines educativos.

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

[8]Implementación en Weka de la entropía relativa (KLIC por sus siglas en inglés) también conocida como *normalized information gain*

el intervalo de tiempo que ha transcurrido entre la recepción de un paquete y la recepción del anterior paquete procedente de esa misma conexión. Este atributo nos permite establecer una relación acerca de la frecuencia con que se establecen las conexiones a los diferentes nodos.

IV. Aprendizaje automático y clasificación supervisada de paquetes

Los autores presentan en este artículo un modelo experimental que hace uso de las técnicas de modelado descritas anteriormente. El objetivo es generar una cantidad de tráfico tal que puede ser empleada por algunos de los clasificadores implementados en Weka para la clasificación de tráfico.

De igual manera a como ya hicieran en el pasado Livadas et ál. [18] para detectar *botnets* controladas por IRC, se ha planteado el problema como un problema de clasificación supervisada. En estos problemas, se estudia un fenómeno representado por un vector X en R^d que puede ser clasificado de K maneras de acuerdo a un vector Y de *etiquetas*.

Con tal fin, se dispone de $D_n = \{(X_i, Y_i)\}_{i=1}^n$ llamado *conjunto de entrenamiento*, 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, en el caso que nos ocupa, estaremos hablando de un paquete X_i definido por una serie de atributos que lo representan, siendo Y_i la categoría asignada a dicho paquete según las estimaciones del clasificador.

A. Algoritmos de clasificación

En esta experimentación, se ha 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 [19]. Los utilizados para las pruebas de la sección V han sido los siguientes:

- ***Random Forest***. Se trata de un clasificador de agregación desarrollado por Leo Breiman [20] y que consta de un abanico de árboles de decisión (o *decision trees*) de modo que se mejora la precisión en la clasificación ya que en la construcción de cada clasificador individual se introduce un componente estocástico, bien en la partición del espacio (la propia construcción de los árboles) o bien en la muestra de entrenamiento.
- **J48**. J48 es una implementación de código abierto del algoritmo C4.5 [21] para Weka. C4.5 crea árboles de decisión dado una cantidad de información de entrenamiento usando el concepto de entropía de la información [22]. Los datos de entrenamiento son un grupo $S = s_1, s_2, ..., s_n$ de ejemplos $s_1 = x_1, x_2, ..., x_m$ ya clasificados en los que $x_1, x_2, ..., x_m$ representan los atributos o características del ejemplo. 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*).
- ***Bayes Net***. Las redes bayesianas 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.
- ***Naive Bayes***. Se trata de 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 [23], de ahí las reducciones que simplifican la muestra y que le dan el apelativo de *Bayes ingenuo*.
- ***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. 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 [24]. Un MLP consta de una red neuronal artificial formada por capas de nodos en un gráfico 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.
- ***K-Nearest Neighbor* (KNN)**. El algoritmo KNN es uno de los algoritmos de clasificación más simples de todos 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 $k = 1, 2, 3, 4, 5$ a fin de determinar si la toma en consideración de más vecinos mejora significativamente los resultados.
- ***Sequential Minimal Optimization* (SMO)**. SMO, inventado por John Platt [25], es un algoritmo iterativo para la solución de los problemas de optimización que surgen con el entrenamiento de los *Support Vector Machines* o máquinas de soporte vectorial. SMO parte el problema en una serie de subproblemas más pequeños que son solucionados analíticamente con posterioridad.

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 V.

B. 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 del experimento realizado, la cantidad de paquetes correspondientes al 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. Por ello, se ha optado por utilizar técnicas de *resampling* que balanceen la cantidad de paquetes de cada clase a emplear por el algoritmo. En el conjunto de datos finalmente utilizado se han empleado 5.127 paquetes correspondientes a información con destino u origen en el servidor controlador de la *botnet* y una muestra de 5.130 paquetes correspondientes a tráfico inocuo.

V. Validación experimental

Una vez definidas las pautas de modelización en las secciones anteriores, se ha procedido a la recabación de tráfico de red con la herramienta ya mencionada Wireshark. Las muestras obtenidas han correspondido a sesiones de tráfico extraídas de equipos utilizados por alumnos del Máster Universitario de Seguridad de la Información de la Universidad de Deusto, que voluntariamente, se han ofrecido a dejar monitorizar su tráfico saliente en equipos infectados con la versión modificada de Flu dedicada a este experimento.

El objetivo residía en que los usuarios legítimos llevaran a cabo sesiones de una hora de navegación normal durante una clase: acceso a buscadores, consultas a enciclopedias *online*, navegación por redes sociales como Facebook, Twitter, LinkedIn o Tuenti, consultas de periódicos digitales, acceso a servidores web de correo electrónico, etc. Mientras tanto, desde un equipo diferente, un administrador iba actualizando el fichero de comandos y órdenes a ejecutar por parte de los nodos de la *botnet* infectados. Algunas de las acciones realizadas incluían la creación de cuentas de usuarios administradores, la obtención de capturas de pantalla, el envío de los ficheros de *keylogging*, la ejecución remota de programas instalados en el equipo, etc.

Con todos estos datos, el tamaño de cada una de las capturas generadas ha sido de entre 0,9GB y 1,5GB en el formato de captura .cap. Posteriormente, para facilitar el parseo de los paquetes, se ha exportado dichas sesiones al formato .xml, lo cual facilitaba ya no solamente el proceso de recabación de la información sino la identificación de los parámetros susceptibles de introducir un conocimiento sesgado en las muestras, como la información relativa a los puertos, direcciones IP u otros protocolos.

Así, en esta sección detallaremos los resultados obtenidos a la hora de evaluar los distintos elementos analizados como predictores de tráfico malicioso. Para cada uno de ellos, se realizará un experimento con el fin de demostrar su validez como predictores utilizando los diferentes clasificadores detallados en la sección IV.

Esta evaluación se realizará en base a cuatro parámetros:

- *True Positive Ratio* (TPR), el cual se calcula dividiendo el número de paquetes de muestra legítimos correctamente clasificados (TP) entre el total de muestras extraídas ($TP + FN$).

$$TPR = \frac{TP}{(TP + FN)} \tag{1}$$

TABLA II
Desempeño de los clasificadores a la hora de categorizar correctamente los paquetes capturados.

Clasificador	FPR	TPR	Precisión (%)	AURC
Random Forest	0,01	0,64	81,47%	0,92
J48	0,02	0,64	81,07%	0,91
Naive Bayes	0,45	0,92	73,41%	0,82
Bayes Net	0,40	0,96	77,65%	0,91
MLP	0,14	0,72	78,61%	0,90
1NN	0,21	0,80	79,79%	0,80
2NN	0,02	0,64	80,90%	0,91
3NN	0,02	0,63	80,74%	0,91
4NN	0,02	0,64	80,64%	0,91
5NN	0,02	0,63	80,68%	0,91
SMO	0,44	0,92	73,88%	0,74

- *False Positive Ratio* (FPR), el cual se calcula dividiendo el número de muestras de paquetes correspondientes a tráfico de *botnet* cuya clasificación se erró (FP) entre el número total de muestras ($FP + TN$).

$$FPR = \frac{FP}{(FP + TN)} \tag{2}$$

- *Precisión* (P), que se calcula dividiendo el total de aciertos entre el número total de instancias del conjunto de datos.

$$P = \frac{FP}{(FP + TN)} \tag{3}$$

- *Area Under ROC Curve* ($AURC$) [19], la cual establece la relación entre los falsos negativos y los falsos positivos. La curva ROC se suele usar para generar estadísticas que representan el rendimiento o la efectividad, en un sentido más amplio del clasificador.

Siendo así, los resultados obtenidos se visualizan en la tabla II. Los mejores resultados han sido obtenidos para el clasificador Random Forest, con una precisión de clasificación del 81,47% y un valor del área por debajo de la curva ROC de 0,92. Por contra, se ha observado que los clasificadores que peor han funcionado han sido Naive Bayes (con una precisión total del 73,41% y un valor de $AURC$ de 0,82) y SMO (con una precisión total del 73,88% y un valor de $AURC$ de 0,74). Como nota adicional, cabe reseñar que no se han apreciado mejoras significativas en cuanto a la efectividad del algoritmo KNN para valores de k mayores, manteniéndose homogéneos la mayor parte de los parámetros analizados.

Dado que la cantidad de paquetes generados por el tráfico de una conexión suele ser muy amplia, se podría pensar en una clasificación teniendo en consideración la historia de los paquetes conectados a un mismo punto. Es decir, emplear la clasificación de los paquetes como una medida de referencia para asignar una valoración del 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 obtener dichos valores se ha calculado la probabilidad $P(n)$ de que la mayoría ($50\% + 1$) de los n ($\forall n \in \mathbb{N}$) últimos paquetes analizados de una conexión, hayan sido correctamente clasificados por un clasificador C, siendo $P(C)$

la probabilidad de etiquetar correctamente un paquete aislado empleando dicho clasificador.

Es decir, si a corresponde a un etiquetado correcto y b a un etiquetado erróneo, para secuencias de una longitud de $n = 3$ paquetes, 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 paquetes etiquetados 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 los n paquetes hayan sido correctamente clasificados, 2) la probabilidad de que sólo 1 de los paquetes haya sido incorrectamente clasificado, 3) la probabilidad de que sólo 2 hayan sido incorrectamente clasificados, etc., y así sucesivamente hasta dar con todas las posibles combinaciones con al menos $n/2 + 1$ ($\forall(n/2) \in \mathbb{N}$) paquetes correctamente clasificados, lo que, en combinatoria, se define en la ecuación 4:

$$\begin{aligned} P(n) &= \binom{n}{0} * P(C)^n + \\ &+ \binom{n}{1} * P(C)^{n-1} * [1 - P(C)] + \\ &+ \binom{n}{2} * P(C)^{n-2} * [1 - P(C)]^2 + \\ &+ \ ... + \\ &+ \binom{n}{\frac{n}{2} - 1} * P(C)^{n-(\frac{n}{2}-1)} * [1 - P(C)]^{\frac{n}{2}} = \\ &= \sum_{k=0}^{\frac{n}{2}-1} \left\{ \binom{n}{k} * P(C)^{n-k} * [1 - P(C)]^k \right\} \end{aligned} \quad (4)$$

Las ecuaciones 5 y 6 son un ejemplo de la aplicación de estas fórmula para secuencias de $n = 3$ y $n = 5$ paquetes.

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

$$\begin{aligned} P(5) &= \sum_{k=0}^{2} \left\{ \binom{5}{k} * P(C)^{5-k} * [1 - P(C)]^k \right\} = \\ &= P(C)^5 + \\ &+ 5 * P(C)^4 * [1 - P(C)] + \\ &+ 10 * P(C)^3 * (1 - P(C)^2) \end{aligned} \quad (6)$$

Como resultado, se obtiene la tabla III, en la que se recogen las estimaciones de haber etiquetado correctamente la mayoría de los paquetes correspondientes a cada una de las diferentes conexiones. Los cálculos tratan de estimar las posibilidades de que para secuencias de $n = 3, 5, 7, 10$ al menos el $50\%+1$ de los paquetes hayan sido correctamente identificados por cada uno de los clasificadores como provenientes de la versión utilizada de la *botnet* Flu. Por poner el ejemplo del clasificador con mayor precisión de la prueba, Random Forest, dados 7 paquetes correspondientes a una misma conexión hay un 97,44% de posibilidades de que al menos 4 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.

TABLA III
ESTIMACIÓN DE POSIBILIDADES DE QUE UNA SECUENCIA DE PAQUETES CORRESPONDIENTES A UNA CONEXIÓN DETERMINADA HAYA SIDO CORRECTAMENTE CLASIFICADA.

Clasificador	1	3	5	7	10
Random Forest	0,8147	0,9097	0,9527	0,9744	0,9957
J48	0,8107	0,9061	0,9500	0,9724	0,9952
Naive Bayes	0,7341	0,8255	0,8790	0,9138	0,9733
Bayes Net	0,7765	0,8725	0,9224	0,9513	0,9887
MLP	0,7861	0,8823	0,9308	0,9581	0,9910
1NN	0,7979	0,8940	0,9405	0,9654	0,9933
2NN	0,8090	0,9045	0,9488	0,9716	0,9950
3NN	0,8074	0,9030	0,9476	0,9707	0,9948
4NN	0,8064	0,9021	0,9469	0,9702	0,9946
5NN	0,8068	0,9024	0,9472	0,9704	0,9947
SMO	0,7388	0,8310	0,8843	0,9186	0,9755

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, es cierto 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 camuflar los mensajes de control en sí mismos.

VI. LÍNEAS DE TRABAJO FUTURO

Las líneas futuras de trabajo están centradas en minimizar la tasa de falsos positivos al máximo con el fin de sólo 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, y 2), que los paquetes sospechosos serán maliciosos con mucha más seguridad. 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, pensando, en lugar de en paquetes independientes, en conexiones, definidas como sucesiones temporales de paquetes enviados y/o recibidos hacia/desde un mismo *host*.

Al mismo tiempo, los aportes aquí descritos sugieren también una gran variedad de líneas de trabajo futuro en

campos relacionados con la modelización 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 maneja. En esta línea, 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 las instituciones dada las posibilidades que ofrecería de cara a hacer frente a las siguientes amenazas: la detección de detección de ataques DDoS[9], 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 [26].

De cualquier manera, los resultados ya comentados consiguen altos índices de detección para el tráfico proveniente de una *botnet* con un canal de *Command & Control* basado en HTTP y HTTPS muy concreto. 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 por ellas incluyendo muestras maliciosas de familias de *malware* lo más heterogéneas posibles.

VII. 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 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 es que pretende clasificar el tráfico independientemente del contenido de la información que fluye por la red, empleando 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.

[9] *Distributed Denial of Service* o ataques distribuidos de denegación de servicio.

Agradecimientos

Los autores de este artículo quieren agradecer a los alumnos del Máster Universitario de Seguridad de la Información de la Universidad de Deusto su ofrecimiento y colaboración para la extracción de las muestras de tráfico necesarias para la realización de esta investigación.

Referencias

[1] Blank, Stephen: Web war i: Is europe's first information war a new kind of war? Comparative Strategy (2008) **27** 227

[2] Hunter, P: Paypal, fbi and others wage war on botnet armies. can they succeed? Computer Fraud Security (2008) 13–15

[3] Libicki, Martin C.: Cyberdeterrence and Cyberwar. RAND, Santa Monica, CA (2009)

[4] Ollmann, Gunter: Hacking as aservice. Computer Fraud & Security (2008) **2008** 12–15

[5] Han, Kyoung-Soo and Im, Eul Gyu: In . Volume. 120 of Lecture Notes in Electrical Engineering. Springer Netherlands 589–593 10.1007/978-94-007-2911-7_56.

[6] Binkley, James R. and Singh, Suresh: An Algorithm for Anomaly-based Botnet Detection . In: SRUTI '06: 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet, Portland State University Portland, OR, USA, USENIX Association (2006)

[7] N. Davis: Botnet detection using correlated anomalies. Master's thesis, Technical University of Denmark, DTU Informatics, E-mail: reception@imm.dtu.dk, Asmussens Alle, Building 305, DK-2800 Kgs. Lyngby, Denmark (2012) Supervised by Professor Robin Sharp, ris@imm.dtu.dk, DTU Informatics.

[8] Fedynyshyn, Gregory and Chuah, Mooi Choo and Tan, Gang: Detection and Classification of Different Botnet C&C Channels. In: ATC'11 Proceedings of the 8th international conference on Autonomic and trusted computing, Lehigh University. Bethlehem, PA 18015, USA, Springer-Verlag 2228–242

[9] Gu, Guofei and Zhang, Junjei and Lee, Wenke: BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. In: Network and Distributed System Security (2007). (2007)

[10] Dietrich, Christian J. and Rossow, Christian and Pohlmann, Norbert: CoCoSpot: Clustering and recognizing botnet command and control channels using traffic analysis. Computer Networks (2012)

[11] Karasaridis, A. and Rexroad, B. and Hoeflin, D.: Wide-scale botnet detection and characterization. In: Proceedings of the USENIX Workshop on Hot Topics in Understanding Botnets (HotBots). (2007)

[12] Ortloff, Stefan: Botnet shutdown success story - again: Disabling the new hlux/kelihos botnet. (Computer Fraud Security)

[13] Banks, Greg and Fattori, Aristide and Kemmerer, Richard and Kruegel, Christopher and Vigna, Giovanni: In . Volume. 6739 of Lecture Notes in Computer Science. Springer Berlin Heidelberg 184–193 10.1007/978-3-642-22424-9_11.

[14] Pablo G. Bringas: Entorno de seguridad inteligente para la detección y prevención de intrusiones de red basado en la detección unificada de patrones y anomalías. PhD thesis, Universidad de Deusto (2007)

[15] Rieck, Konrad and Holz, Thorsten and Willems, Carsteb and Duessel, Patrick and Laskov, Pavel: Learning and Classification of Malware Behavior. In: n Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA), Springer-Verlag 232–249

[16] Lera-López, Fernando and Billon, Margarita and Gil, María: Determinants of internet use in spain. Economics of Innovation and New Technology **20** 127–152

[17] Kullback, S. and Leibler, R. A.: On Information and Sufficiency. Volume. 22., USENIX Association 79–86

[18] Carl Livadas and Robert Walsh and David Lapsley and W. Timothy Strayer: Using machine learning techniques to identify botnet traffic. In: In 2nd IEEE LCN Workshop on Network Security (WoNS'2006. 967–974

[19] Singh, Y. and Kaur, A. and Malhotra, R.: Comparative analysis of regression and machine learning methods for predicting fault proneness models. International Journal of Computer Applications in Technology (2009) **35** 183–193

[20] Breiman, Leo: Random forests. Machine Learning **45** 5–32 10.1023/A:1010933404324.

[21] Quinlan, J.R.: C4. 5: programs for machine learning. Morgan kaufmann (1993)

[22] Salzberg, Steven L.: C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. Machine Learning **16** 235–240 10.1007/BF00993309.

[23] Jiang, Liangxiao and Wang, Dianhong and Cai, Zhihua and Yan, Xuesong: In . Volume. 4632 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg 134–145 10.1007/978-3-540-73871-8_14.

[24] Basogain Olabe, Xabier: Redes neuronales artificiales y sus aplicaciones. (Escuela Superior de Ingeniería de Bilbao, EHU)

[25] John C. Platt: Sequential minimal optimization: A fast algorithm for training support vector machines (1998)

[26] Guofei, Gu and Perdisci, Roberto and Zhang, Junjie and Lee, Wenke: BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: SS'08 Proceedings of the 17th conference on Security symposium, USENIX Association 139–154