Estudio de características de comportamiento sincrónico en la red para la clasificación de Botnets

Sebastián García

23 de octubre de 2009

sgarcia@exa.unicen.edu.ar

Resumen

El estudio de las Botnets como amenazas a la seguridad ha llevado a encontrar diferentes formas para detectarlas y clasificarlas. La perspectiva del estudio de comportamiento desde la detección de las Botnets en general es la más adecuada para una solución escalable y completa. Esta generalidad se da por medio de ciertas características fundamentales de las mismas. Estas características muestran un patrón de sincronismo que ha sido estudiado en trabajos anteriores[3] y que ampliamos con el fin de mejorar la detección.

1. Introducción

Las técnicas de detección de Botnets mantienen una gran dependencia en las características seleccionadas para los algoritmos. Es así que muchas de las técnicas de detección solo se pueden aplicar a Botnets de cierto protocolo o incluso a una Botnet en particular. El dinamismo y variabilidad de las Botnets hace que tales técnicas necesiten ser actualizadas constantemente. Definir un algoritmo de detección que sea aplicable a la totalidad de las Botnets permitiría contar con una herramienta que perdure más en el tiempo y que no necesite actualizaciones cuando las Botnet cambien sus protocolos, puertos o métodos de cifrado.

Desde hace algunos años, las técnicas de detección de los Antivirus[6] se han volcado al análisis del comportamiento del malware porque ha sido la única forma de lograr características genéricas de detección.

El análisis de los paquetes directamente de la red es una forma de capturar datos de todas las Botnets y de todos los protocolos. Este estudio se centrará en el análisis de los datos capturados en las redes para que su aplicación pueda ser lo más genérica posible.

Diversos estudios han evaluado la posibilidad de trabajar con el sincronismo como una de las características más importantes de las Botnets a la hora

de analizar su comportamiento. El sincronismo permitiría detectar patrones ineludibles en el accionar de una Botnet, de tal forma que evitar tales comportamientos disminuiría considerablemente el rendimiento de la misma y ya no sería útil para el Botmaster o dueño de la Botnet.

En [2] se estudian patrones de sincronismo, pero solamente para ciertos momentos del ciclo de vida de algunas Botnets. Se hace hincapié en la naturaleza de los protocolos usados, más que en las propiedades intrínsecas.

Otros trabajos[5] se centran en las características y frecuencias temporales pero solo de ciertas aplicaciones ,una vez que estas han sido reconocidas. El mayor problema es que ciertas Botnets no utilizan un protocolo en particular para su canal de Comando y Control (de aquí en más C&C) y los diversos ataques tampoco tienen un protocolo definido. Un sistema que tenga como objetivo detectar a todas las Botnets debería ser lo más independiente del protocolo que sea posible.

Este trabajo se centra en reconocer las propiedades del sincronismo en las Botnets a fin de poder parametrizarlo y así encontrar las diferentes situaciones en el ciclo de vida de una Botnet donde es posible realizar esta detección.

La parametrización del sincronismo permite también estudiar este fenómeno desde la la visión del Bot y desde la visión de la Botnet. La visión de comportamiento global de toda la red es diferente de la visión localizada de sincronismo de un solo individuo.

Se definen varios conceptos novedosos, entre ellos la visión del sincronismo como la única característica general de las Botnets, la extracción de cuatro características para el análisis de sincronismo y la parametrización de tales características a fin de automatizar el análisis. También se realizaron diversas capturas de Botnets que sería posible en un futuro publicar como parte de un Dataset que permitiría la comparación de métodos de detección de Botnets.

Parte fundamental de los procesos de estudio son los datos utilizados como origen para realizar las detecciones. Es fundamental que los datos pertenezcan a Botnets reales funcionales. Es por esto que se dedicó gran parte del tiempo a configurar y diseñar los experimentos de captura de datos. Desde estas capturas se extrajeron los datos base que se utilizaron para verificar las hipótesis de sincronismo.

Este trabajo concluye con la extracción de los parámetros para detectar sincronismo en las Botnets y verificar su viabilidad. Posteriormente otros trabajos utilizarán estas características para realizar sus estudios de detección.

2. Orígenes de Datos

La problemática más importante en la obtención de datos de Botnets es que no se encuentran fácilmente de forma pública repositorios que puedan utilizarse como base de trabajo compartida. Esto es así, en parte, porque el acceso a datos de malware puede potencialmente ser perjudicial o ser mal utilizado. Incluso obteniendo estos datos, existirían dos problemas a solucionar: por un lado obtener datos que sean de una Botnet y no de un Bot aislado y por otro que los

datos sean solamente del accionar de una Botnet y no haya tráfico normal mezclado, osea su verificación Esto sería de gran importancia para el aprendizaje del comportamiento.

A fin de recolectar datos propios, se diseñaron una serie de experimentos de acuerdo a las necesidades planteadas. Los datos requeridos deberán ser de dos tipos. Primeramente se necesitan datos etiquetados o "ground truth data" a fin de poder analizar el comportamiento de los Bots y de las Botnets. Seguidamente se necesitan datos con tráfico de Botnets mezclados con tráfico normal. Es importante notar que la captura de tráfico normal lleva implícita la definición de qué es *normal* y consecuentemente cómo se verifica que el tráfico capturado cumple con las condiciones de normalidad.

Para la captura de datos se creó una máquina virtual con el sistema operativo Microsoft Windows XP donde se ejecutaban binarios de diferentes malware disponibles públicamente[8]. Se estudio este sistema operativo por ser el más atacado por malware actualmente.

Se utilizó virtualbox 3.0.x como sistema de virtualización sobre Linux Kubuntu 9.10, y máquinas virtuales (VM de acá en adelante se utilizará por ser el término establecido comúnmente) en modo Bridge

El proceso de captura siguió la siguiente metodología:

1. Descarga de un binario infectado
2. Verificación de qué malware es y cómo es detectado mediante su envío a http://www.virustotal.com
3. Almacenamiento del binario reconocido y de sus características conocidas
4. Encendido de la máquina virtual sin infectar
5. Verificación de la dirección IP de la VM y del acceso a Internet
6. Usando el Explorador de Archivos, copia del binario mediante carpetas compartidas
7. Cierre del Explorador de Archivos
8. Ejecución del sniffer desde el Linux para capturar solo la IP de la VM
9. Estabilización del tráfico dejando pasar un minuto de captura sin infección
10. Ejecución del binario malware con un Explorador de Archivos en la VM
11. Captura de la actividad si la hubiere mientras dure. No más de un día dependiendo de las acciones
12. Estabilización del tráfico si no hay actividad dejando pasar un minuto y apagado del sniffer
13. Apagado de la VM sin guardar el estado actual y vuelta al estado anterior de la VM sin infectar

14. Almacenamiento y registro de la captura en un directorio separado con el binario y una explicación de lo probado incluyendo detalles del entorno

Se realizaron 18 experimentos pero solo en ocho se logró infectar verdaderamente el equipo virtual con una Botnet y capturar los datos. Esto nos indicó la dificultad de una captura de datos reales. También se realizaron capturas de redes en su utilización normal para un posterior análisis. Dentro de esta misma línea se capturó el tráfico normal que produce la VM Windows cuando no está infectado mientras realiza operaciones de booteo y rebooteo para poder posteriormente extraer estos patrones y no confundirlos con acciones de la Botnet. Aparte de las capturas de tráfico infectado de laboratorio y de las capturas de tráfico normal verificado, se realizaron capturas en redes que contenían al menos un equipo infectado junto con tráfico normal. Estas últimas capturas se reservarán para los procesos de verificación de los algoritmos. Esta etapa nos permitió obtener archivos de capturas de actividad de Bots y Botnets en formato pcap. En resumen se obtuvieron aproximadamente 811MB de datos de seis Botnets diferentes.

Este proceso de captura de datos llevó un tiempo considerable dado que se realizaron varias veces todos los experimentos, probando de infectar equipos en distintas redes, e incluso con distintas versiones del sistema operativo Windows.

3. Parametrización del sincronismo

En todo trabajo de extracción de características es recomendado[4] tener un conocimiento del problema a priori a fin de mejorar las probabilidades de detección. Es importante en esta selección mantener claro el objetivo final de clasificar Botnets mediante el estudio de su comportamiento en la red e independencia del protocolo.

Podemos clasificar el estudio de los comportamientos en dos grupos. El primero corresponde al comportamiento del Bot como unidad de trabajo autónoma y el segundo al comportamiento de la red Botnet como agrupación coordinada y síncrona. Sería posible entonces encontrar patrones de comportamiento que no son visibles al estudiar un Bot de forma individual[3].

Para definir correctamente las diferentes características que sería posible evaluar se diagramó en el Cuadro 1 el ciclo de vida tentativo y general de una Botnet. Algunas etapas del ciclo de vida son opcionales según la Botnet y otras son ineludibles.

Cada una de estas fases puede verse desde el punto de vista del Bot o de la Botnet y cada una puede generar distintas características a utilizarse para detectar comportamientos. Por ejemplo, analizando el comportamiento de una Botnet, durante las fases de ejecución de órdenes sería posible detectar el *sincronismo* de todos los Bots involucrados al intentar cumplir esas acciones. Desde el punto de vista del Bot, en esa misma fase, sería posible detectar cuando un equipo *envía Spam*. La misma acción genera dos características diferentes.

Este trabajo se centra en los patrones sincrónicos de la Botnet, pero invariablemente será capaz también de ver los comportamientos de los Bots.

1. Escaneo del equipo víctima (opcional ya que puede infectarse directamente desde la Web)
2. Infección (ineludible)
3. Conexión al servidor de comando y control para reportarse (acción periódica e ineludible)
4. Envío de datos sobre el Bot (opcional pero sucede en la mayoría de los casos)
5. Recepción de órdenes (ineludible)
6. Ejecución de órdenes
    - Descarga de software
    - Envío de información privada de la víctima
    - Ataques DDOS
    - Envío de Spam
    - Permitir la interacción con el Bot manual
    - Infectar otros equipos (ineludible por algunos Bots)

Cuadro 1: Diagrama del ciclo de vida tentativo de una Botnet

Para encontrar las características que mejor detectan el comportamiento, diversos trabajos en el área[[2]] definen tres situaciones que son ineludibles de los Bots y de las Botnets:

- Un Bot se comunica con su servidor de comando y control y si es funcional recibe órdenes[1]
- Una Botnet realiza sus acciones de forma síncrona
- Un Bot realiza acciones maliciosas.

La comunicación con el servidor de C&C es parte de las fases del ciclo de vida vistas con anterioridad, al igual que las acciones maliciosas. El sincronismo de las acciones es lo único que no se encuentra en el ciclo de vida propuesto ya que es una consecuencia de las necesidades de la Botnet. Este sincronismo es la característica principal que se intentará detectar como parte del comportamiento.

El estudio del sincronismo se fundamenta en la utilidad más importante de las Botnets. Desde que el modelo de infección pasó de equipos sueltos a equipos en red, la ventaja más importante es la unión de las tareas. Esto le proporciona al Botmaster un recurso del que normalmente no dispondría. Para utilizar esta unión de equipos, los mismos deben actuar en forma coordinada y simultánea. Cada conexión desde un Bot hacia la red se ve reflejada como una cantidad de paquetes que se envían y reciben, por lo tanto solamente estudiando los flujos de paquetes es posible trabajar con el sincronismo.

Se denomina flujo TCP[2] o circuito virtual TCP a los paquetes que comparten un par de sockets, osea una dirección IP origen, un puerto origen, una dirección IP destino y un puerto destino. Estos paquetes pueden separarse fácilmente del resto ya que cada flujo es único en un intervalo de tiempo considerable. Este intervalo de tiempo está dado por lo que tarda el sistema operativo origen en reusar el puerto origen del flujo. Normalmente esto no sucede en varios días de operación, aunque veremos más adelante que en el caso de un escaneo de puertos puede volver a utilizarse el mismo puerto origen luego de apenas unos minutos.

Luego de separar los flujos de forma unívoca utilizando el par de sockets, se selección los datos de las cabeceras TCP/IP en cada flujo que son útiles para la detección del sincronismo. Primeramente necesitamos poder distinguir el destino y origen de los flujos TCP y esto lo conseguimos con las direcciones IP origen y destino y el puerto destino. En nuestro caso, el puerto origen no es utilizado para la detección ya que no es seleccionado por la Botnet sino que es automáticamente asignado por el sistema operativo. De esta forma su variación no está relacionada directamente con el comportamiento.

Seguidamente para detectar el sincronismo solo necesitamos conocer la fecha y hora exacta de comienzo del flujo.

Es posible trabajar entonces solamente con los siguientes datos de cada flujo TCP:

[1] La recepción de órdenes ineludible como contrapartida del pedido de órdenes no aparece en ningún trabajo, por lo tanto sería algo 'novedoso' a estudiar dado que más adelante se muestra que es importante.

[2] http://en.wikipedia.org/wiki/Flow_%28computer_networking%29

- Dirección IP origen
- Dirección IP destino
- Puerto destino del flujo TCP (el puerto de la IP destino)
- Tiempo de comienzo[3]

Estos cuatro datos pueden generalizarse relacionándolos con cuatro variables que identifican diferentes situaciones de sincronismo en una red. Al tener varios flujos TCP simultáneos en un tráfico característico, podemos hablar de una variable denominada *sip* que identifique a la cantidad de direcciones IP origen únicas en un intervalo de tiempo de un segundo. Asimismo denominaremos *dport* a la cantidad de puertos destino diferentes a los que esas *sip* direcciones IP se están conectando (de todas las IP en el grupo *sip*), y denominaremos *dip* a la cantidad de direcciones IP destino a las que esas *sip* direcciones IP se están conectando, finalmente llamaremos t al intervalo de tiempo durante el cual continúa el sincronismo de todas las *sip* direcciones IP.

En un principio no podemos diferenciar las conexiones entrantes de las salientes de una red interna, ya que no tenemos datos sobre que direcciones utilizaría esa red interna. A todos los efectos, las redes vistas en los análisis tienen el mismo grado de privacidad.

Utilizando estos cuatro valores, se definió en el Cuadro 2 un conjunto de situaciones sincrónicas detectables.

Los valores del Cuadro 2 asignados como 'grande' y 'chico' dependerán en realidad de dos factores. En primera instancia de la cantidad de equipos en la red que se está monitoreando y seguidamente de la cantidad de equipos cuyo comportamiento se considere *normal*. Por ejemplo si en una red de 1000 equipos, en un momento dado es *normal* que en promedio 200 estén conectándose al puerto 80 de www.google.com, entonces el valor de *sip* 'grande' para esta regla en esa red debería ser quizá mayor a 200.

En general se puede considerar que en relación a las *sip* y *dip* un valor chico oscila entre 3 y 10 y un valor grande en más de 100.

En referencia a los tiempos, se tomará como instantáneo hasta un segundo de tiempo, dados las infecciones vistas en las capturas realizadas.

Dado que se puede parametrizar estas cuatro variables, es posible generar la combinación de valores a fin de estudiar todas las situaciones de sincronismo posibles. La variable *sip* tiene tres valores (grande, chico y 1), la variable *dip* tiene tres valores (grande, chico y 1), la variable *dport* tiene tres valores (grande, chico y 1) y la variable t tiene tres valores (grande, chico e instantáneo). En total hay 81 combinaciones diferentes de situaciones a analizar. Muchas de estas situaciones son del comportamiento normal de una red, otras son sospechosas de pertenecer a una Botnet y algunas están directamente relacionadas con una Botnet.

[3]Es necesario definir la precisión que se utilizará

Id	Detección	*sip*	*dip*	*dport*	*t*
1	Botnet contactando a un servidor de C&C	grande	1	1	Instantáneo (<1seg)
2	Escaneo de equipos para infectar	grande	grande	1	chico
3	Escaneo de equipos para infectar	grande	grande	chico	chico
4	Sincronismo desde el server de C&C hacia los Bots	1	grande	1	Instantáneo (<1seg)
5	Escaneo de puertos tradicional y no de un Bot	1	chico(any)	grande (>1000)	chico(<30seg)
6	Respuestas a un escaneo de puertos	chico (>10)	1	chico (=sip)	Instantáneo (~1seg)
7	Ataque de DDoS	chico (>10)	chico	1	chico(> 60seg)
8	Envío de spam	1	grande	1 (puertos 25s)	grande
9	Botnet escaneando de forma distribuida un equipo	grande	1	grande	chico(>60seg)
10	Botnet contactando a un servidor de C&C	grande	chico	1	instantáneo

Cuadro 2: Parámetros probables para las cuatro características

Sería posible ahora aplicar algún algoritmo de verificación para obtener cuales de estas nuevas características identifican mejor a una Botnet. Este es un nuevo campo que sería conveniente estudiar detenidamente.

3.1. Justificación de la selección de características

Como justificación o verificación general de que estas características son posibles de obtener y sirven al proceso de detección por comportamiento, se hicieron algunos estudios preliminares. Se obtuvieron dos tipos de datos, por un lado algunas muestras de paquetes de Botnets reales en acción en una red Universitaria con quince equipos infectados y por otro lado diversas infecciones en un solo equipo en una VM. Cada una de las capturas fue procesada a fin de visualizar los procesos sincrónicos.

En la parte superior de las figuras se muestran los valores para los ejes X e Y. El eje X siempre se corresponde con la variable t, osea el tiempo en que comienza un flujo de datos. En la Figura 1 se puede ver un estudio con el algoritmo de clustering EM[1] (Expectation-Maximization),. El algoritmo detectó ciertos patrones temporales que pudrían revelar sincronismos. La zona marcada como 1b se corresponde con al menos nueve direcciones IP destino recibiendo datos al mismo momento. El hecho de que sean horizontales indica que la misma dirección IP destino recibió paquetes por un tiempo considerable de forma continuada (las direcciones IP origen y puerto destino pueden variar). El hecho de que verticalmente se vean muchas líneas horizontales significa que de for-

Figura 1: Algoritmo EM sobre datos de Botnet capturada

ma sincrónica muchos destinos diferentes recibieron una conexión por el mismo intervalo de tiempo. La anomalía se ve reflejada en la cantidad de conexiones instantáneas sincrónicas.

En la zona 1a se ve el mismo patrón para dos equipos diferentes.

En la zona 2 se puede ver como al menos once equipos diferentes reciben muy pocos paquetes (quizá uno o dos) exactamente al mismo tiempo, repitiéndose luego la misma situación.

Esta forma de comportarse nos muestra que sería posible trabajar con los sincronismos en el estudio de las Botnets.

Otro experimento se desarrolló con una Botnet sobre la que teníamos control, en la que se logró infectar satisfactoriamente al menos diez equipos y se los manipuló para que realizaran un ataque de Negación de Servicio Distribuido (DDoS) contra otro equipo en Internet.

Si bien este trabajo no detectará patrones de tiempos regulares no sincrónicos (detección del Polling de los Bots), se puede ver que es factible tal detección si se buscan conexiones que se repiten a intervalos de tiempos regulares para la misma IP destino. Se denomina Polling a la acción periódica a través de la cual un Bot se pone en contacto rutinariamente con su servidor de C&C. En la Figura 1 podemos ver en la zonas 3a, 3b y 3c como se detecta este polling. Se

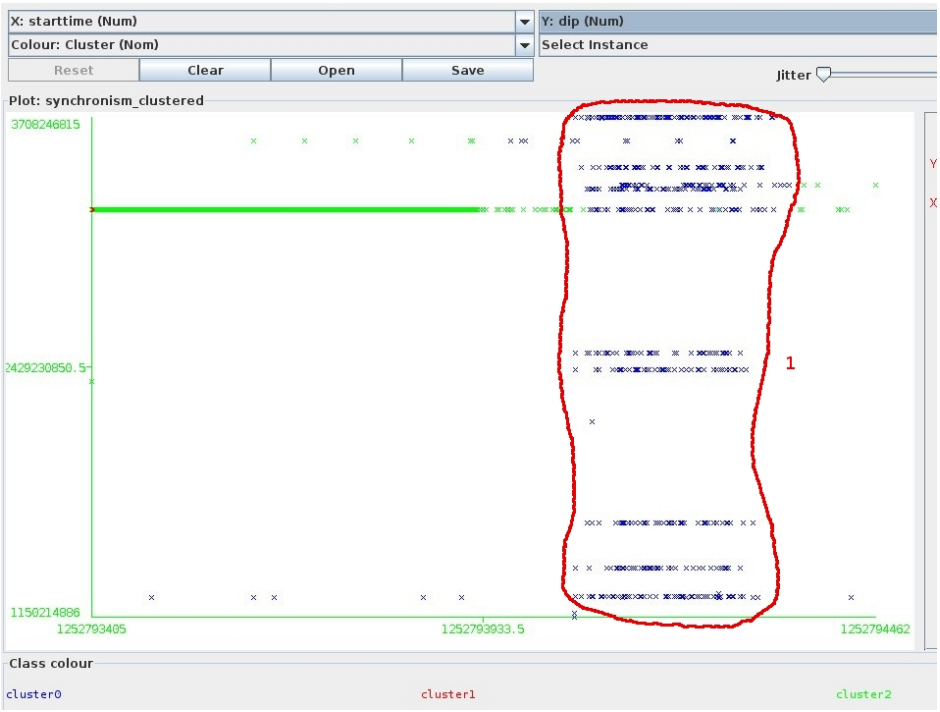

Figura 2: Algoritmo EM sobre datos de Botnet controlada

aprecia que la zona 3c está ligeramente desplazada hacia la derecha respecto de la zona 3b, mostrando que el patrón es de tiempos regulares, pero no sincrónicos entre sí.

El patrón de la Figura 2 está identificando a este ataque de DDoS junto con otros ataques de otra Botnet. En este caso es más notoria la anomalía del sincronismo dado que la red estaba casi sin actividad normal exceptuando las Botnets.

La verificación del patrón de respuestas a un escaneo de puertos como parte de una infección se puede ver en la captura del Cuadro 3, donde vemos que el equipo 192.168.3.104 realizó un escaneo de puertos como parte de una infección y los equipos escaneados le responden en un solo segundo. Este patrón corresponde a la conexión de una cantidad 'grande' de *sip* o IP origen hacia una sola *dip* en un tiempo instantáneo.

En el caso de un Bot escaneando toda una red para continuar infectándose podemos ver claramente los patrones si analizamos los puertos destinos, como en la Figura3. Aquí vemos que las líneas oblicuas corresponden a las respuestas de los equipos escaneados hacia el Bot diciéndole que los puertos están cerrados. Nótese la importancia de la selección correcta de coordenadas, ya que los mismos

Id	Sip	Dip	Dport	Time
287	192.168.3.104	192.168.142.191	445	22:44:08
288	192.168.2.218	192.168.3.104	1073	22:44:20
289	192.168.40.152	192.168.3.104	1076	22:44:20
290	192.168.78.85	192.168.3.104	1079	22:44:20
291	192.168.198.3	192.168.3.104	1075	22:44:20
292	192.168.160.69	192.168.3.104	1072	22:44:20
293	192.168.236.192	192.168.3.104	1078	22:44:20
294	192.168.18.127	192.168.3.104	1081	22:44:20
295	192.168.138.44	192.168.3.104	1077	22:44:20
296	192.168.100.110	192.168.3.104	1074	22:44:20
297	192.168.176.234	192.168.3.104	1080	22:44:20
298	192.168.3.104	192.168.53.24	445	22:44:29

Cuadro 3: Detalle de respuestas a un escaneo de puertos

datos analizados desde las IP destino, como se muestra en la Figura4 no nos dan una pauta clara de este comportamiento. Si podemos ver que el algoritmo de clustering EM es capaz de separar satisfactoriamente los dos tipos de tráfico, por un lado las peticiones del Bot a la red marcadas en rojo en el cluster1 y las respuestas de los puertos cerrados de toda la red al Bot en azul en el cluster0.

4. Pre-procesamiento de los datos

Para poder analizar el funcionamiento del sincronismo como característica de detección es necesario pasar de las capturas de paquetes en la red en formato pcap a los formatos utilizados por el weka[7]. El protocolo TCP/IP permite identificar a todos los paquetes utilizados en el mismo flujo, de forma que el primer paso en el procesamiento de los datos es la separación de todos los flujos de datos en la captura. Esta extracción es automática y existen diversas herramientas que permiten realizarla, ya que consiste en unir todos los paquetes que comparten el par de sockets. Una herramienta muy utilizada en este trabajo es tcptrace[4], que permite obtener una salida con todos los flujos (o sesiones para el tcptrace) y sus características estadísticas más importantes.

De cada archivo de captura pcap se genera un archivo tcptrace. Para extraer de cada flujo, de este archivo tcptrace, las cuatro características se desarrolló un pequeño programa en python. La salida del mismo es un archivo en formato arff listo para ser analizado con el weka4. En este archivo se puede ver que las direcciones IP están expresadas en formato decimal en lugar del clásico formato con puntos. Esto se debe a que es más fácil para el WEKA calcular distancias entre direcciones IP si estas se pueden restar numéricamente. También vemos que el timestamp está expresado en el formato UNIX time[5], de forma

[4] http://jarok.cs.ohiou.edu/software/tcptrace/
[5] http://en.wikipedia.org/wiki/Unix_time

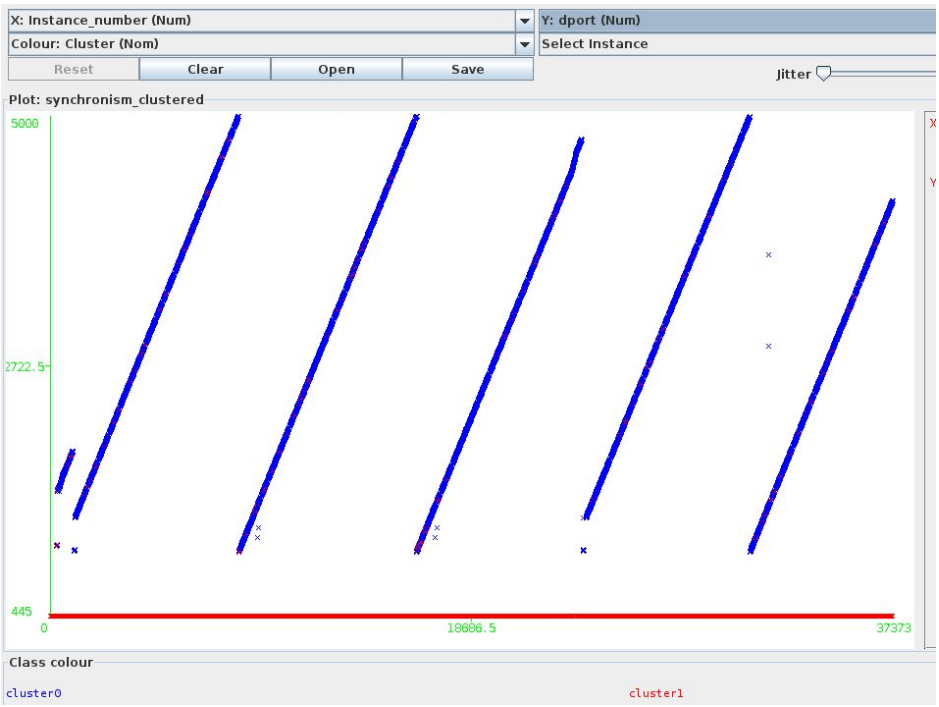

Figura 3: Patrón de respuestas a un escaneo de puertos normal

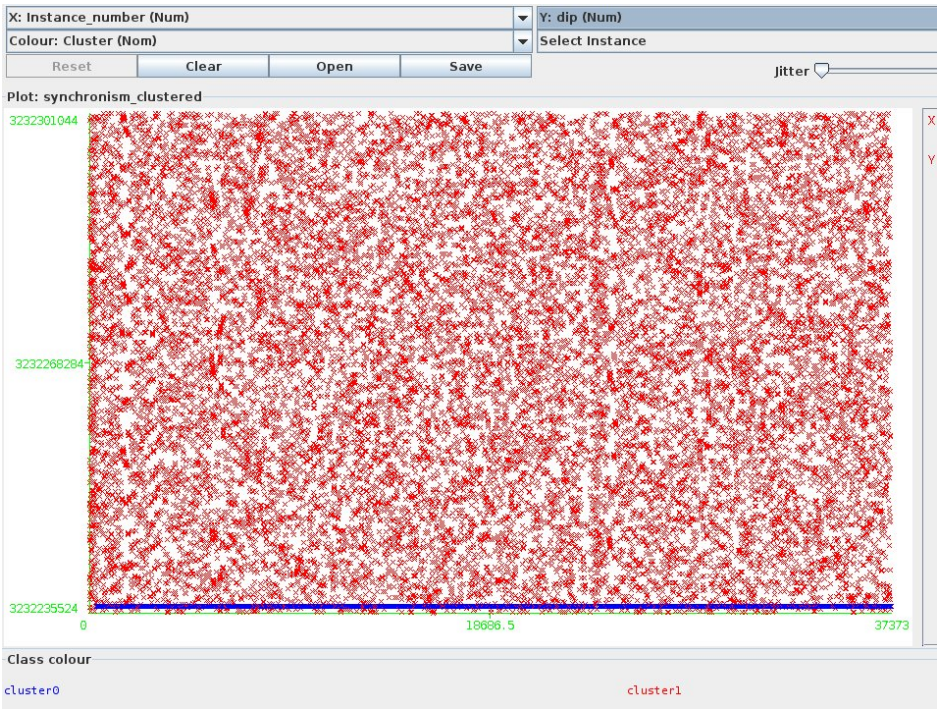

Figura 4: Patrón de respuestas a un escaneo de puertos de un Bot

@relation synchronism

@attribute sip real
@attribute dip real
@attribute dport real
@attribute starttime real

@data
3232236392 3232276549 445 1252373661
3232236392 3232236250 445 1252373661
3232236392 3232261230 445 1252373661
3232236392 3232286211 445 1252373661

Cuadro 4: Archivo arff generado para el weka

que también el cálculo de las distancias entre dos fechas es más fácil al poder solamente restar los números.

5. Trabajo Futuro

Hay varios aspectos de los análisis que necesitan refinarse. Por un lado los datos capturados necesitan ser verificados ya que se encontró que varias de las capturas contenían otras Botnets funcionales que no fueron explicitamente generadas.

Sería posible también en un futuro detectar comportamiento por un patrón repetitivo de conexiones no sincrónicas cuando los Bots mantienen una comunicación periódica con el servidor de C&C. Esto nos marcaría las conexiones de polling a los servidores, Al no ser sincrónica, está conexión se dejará para un futuro análisis.

Un aspecto técnico a tener en cuenta es que las capturas realizadas parecen tener una precisión temporal de segundos solamente. Esto hace que todo lo que suceda dentro de ese segundo es virtualmente instantáneo y sincrónico. Habría que estudiar la precisión.

El trabajo futuro más importante es el estudio de las 81 combinaciones de las variables a fin de conocer todas las situaciones posibles donde se pueden detectar sincronismos. Una vez identificadas estas situaciones es preciso evaluar cuales de ellas se comportan mejor en sistemas de clasificación.

6. Conclusiones

En base a los experimentos realizados y a las previsualizaciones de los análisis creemos que existe evidencia que apoya la utilización de la característica del sincronismo para detección de Botnets mediante el estudio de los paquetes en la red. Se observó que si bien es necesario un estudio más detallado de los momentos

exactos donde se producen los sincronismos, es posible obtener reglas generales que nos permiten alertas tempranas sobre comportamientos sospechos.

References

[1] Frank Dellaert College and Frank Dellaert. The expectation maximization algorithm. Technical report, 2002.

[2] G. Gu, J. Zhang, and W. Lee. BotSniffer: Detecting Botnet Command and Control Channels in Network traffic. In *Proc. 15th Network and Distributed System Security Symposium (NDSS), San Diego, CA, February*, 2008.

[3] Guofei Gu. *Correlation-based Botnet Detection in Enterprise Networks*. PhD thesis, August 2008.

[4] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, March 2003.

[5] Wei Lu, Mahbod Tavallaee, Goaletsa Rammidi, and Ali A. Ghorbani. Botcop: An online botnet traffic classifier. pages 70–77, May 2009.

[6] C. Nachenberg. Behavior blocking: the next step in anti-virus protection, 2002.

[7] I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann Pub, 2005.

[8] V. Yegneswaran, H. Saidi, P. Porras, M. Sharif, W.S. Mark, and V. President. Eureka: A Framework for Enabling Static Analysis on Malware. Technical report, Georgia Institute of Technology, April 2008.