A. Acien, A. Nieto, G. Fernandez, and J. Lopez, "Definición de procedimientos para fabricar honeypots IoT basados en criterios de búsqueda", *XV Reunión Española sobre Criptología y Seguridad de la Información*, 2018.
NICS Lab. Publications: https://www.nics.uma.es/publications

Definición de procedimientos para fabricar honeypots IoT basados en criterios de búsqueda

Antonio Acien
NICS Lab
Universidad de Málaga
acien@lcc.uma.es

Ana Nieto
NICS Lab
Universidad de Málaga
nieto@lcc.uma.es

Gerardo Fernandez
NICS Lab
Universidad de Málaga
gerardo@lcc.uma.es

Javier Lopez
NICS Lab
Universidad de Málaga
jlm@lcc.uma.es

***Resumen*—Con la revolución tecnológica que ha supuesto la Internet de las Cosas (Internet of Things, IoT) se han presentado escenarios donde la preocupación por la seguridad en dicho entorno es cada vez más relevante. Están comenzando a surgir vulnerabilidades en varios dispositivos, y los sistemas trampa son una excelente manera de lidiar con este problema. En este trabajo se analizan soluciones para honeypots en el entorno IoT (y en otros que se puedan adaptar) para sentar las bases de una metodología que permita el despliegue de honeypots IoT.**

***Index Terms*—IoT, honeypot, seguridad, botnet**

I. Introducción

La IoT está cada vez más presente en la vida diaria, afectando al sector público y empresarial, y a los usuarios que día a día delegan en sus objetos personales parte de su rutina. Las estimaciones de dispositivos conectados para 2020 varían entre diferentes estudios, pero en todo caso superan los veinte mil millones [1]. Cada vez se les da mayor poder y responsabilidad a estos dispositivos (vehículos autónomos, *smart locks* o cerraduras inteligentes), y los usuarios son más dependientes de ellos, haciendo que sean muy suculentos para los atacantes.

El panorama que presentan los dispositivos IoT es muy diverso, pues utilizan tecnologías muy distintas, y realizan funciones totalmente diferentes entre ellos. Cada uno supone una puerta de entrada distinta a ataques y amenazas, y teniendo en cuenta lo integrados que están en la sociedad actual, su seguridad debe ser un aspecto muy a tener en cuenta. A día de hoy, es imposible predecir con exactitud el efecto que tendría un ataque dirigido a la presente infraestructura de red.

Aunque la seguridad en dispositivos IoT se lleva considerando desde varios años atrás, debido a ataques a routers y dispositivos empotrados [2], fue la cantidad de *botnets* (redes de bots) que comenzaron a aparecer lo que hizo saltar las alarmas. Tras ciertos ataques por parte de botnets que usaban dispositivos IoT (Gafgyt, BrickerBot, Tsunami), fue a finales de 2016 cuando Mirai, el que más estragos causó, tuvo lugar, lanzando un ataque de denegación de servicio distribuido (DDoS) que dejó inoperativos sitios como Amazon, Github y Twitter[3]. A medida que la preocupación acerca de estos ataques ha ido creciendo, el presupuesto destinado a seguridad específica en IoT también lo ha ido haciendo, así como la cantidad de malware dirigido a esta plataforma[4].

Teniendo en cuenta el riesgo que supone que la IoT sea un escenario heterogéneo con el que es difícil lidiar, y que tenga un papel tan crucial en muchos casos, es necesario promover medidas para evitar ataques tan pronto como sea posible, entenderlos y analizarlos en un entorno seguro, ya que, como hemos visto, no es realista pensar que no van a ocurrir, o que las medidas presentes a día de hoy son suficientes. En otras palabras, se necesita atraer a los atacantes a sistemas trampa, de manera que obtengamos código que analizar con seguridad. Estos sistemas son los *honeypots* y *honeynets*.

Disponer de honeypots es especialmente útil a la hora del *logging* (registro) de eventos, ya que muchos ataques se esfuerzan en eliminar su rastro, y en el aspecto económico, ya que la emulación y simulación de dispositivos nos permite el despliegue de una gran cantidad de nodos sin un gran coste.

II. Trabajo relacionado

Hasta donde sabemos, no hay trabajos que presenten una metodología para desplegar honeypots destinados para el entorno IoT. Algunos sondeos han estudiado exhaustivamente los honeypots existentes, estableciendo clasificaciones y enfoques, y sí que se encuentran algunas soluciones enfocadas a IoT [5].

Estas soluciones no contemplan el análisis de los requisitos necesarios para desplegar los honeypots, y aquí es donde se centra este artículo: identificar correcta y claramente el conjunto de pasos y procedimientos a seguir para desplegar exitosamente honeypots en el entorno IoT. Este éxito se conseguirá si el sistema es lo suficientemente suculento para atraer atacantes de los que extraer información, y lo suficientemente realista como para no ser detectado como un sistema trampa.

Sin embargo, los honeypots IoT existentes presentan unas características interesantes. Al ser un campo reciente, la cantidad de soluciones es limitada, pero propuestas como IoTPOT [6], SIPHON [7] o IoTCandyJar [8] ofrecen arquitecturas completas de despliegue, las cuales tienen particularidades como redirigir varios honeypots de baja interacción (más simples y menos costosos, pero también menos realistas) a uno o varios de alta, dando una apariencia más real de cara a los atacantes sin aumentar en gran medida el coste. Otros, en cambio, conectan los honeypots a servidores cloud para simular que están distribuidos por el mundo, o usan las respuestas a comandos de otros dispositivos IoT conectados a Internet para elaborar mensajes realistas. Hay algunos trabajos que se basan en estos, haciendo implementaciones de código abierto o mejoras en el soporte de protocolos [9].

Asimismo, hay otros honeypots que no cuentan con una arquitectura detrás que les dé soporte, pero que pueden ser útiles en el entorno IoT, ya que simulan dispositivos como cámaras [5] u otros que cuentan con vulnerabilidades concretas (routers vulnerables a Mirai [10] o que tienen el

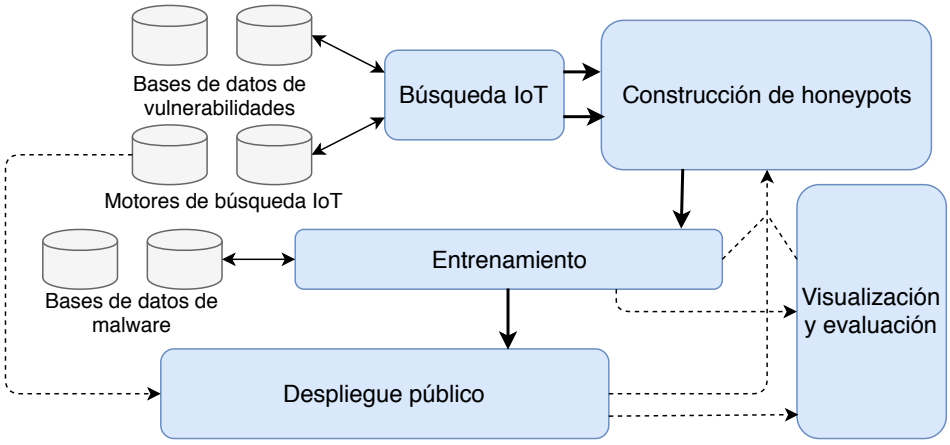

Figura 1. Metodología H-IoT

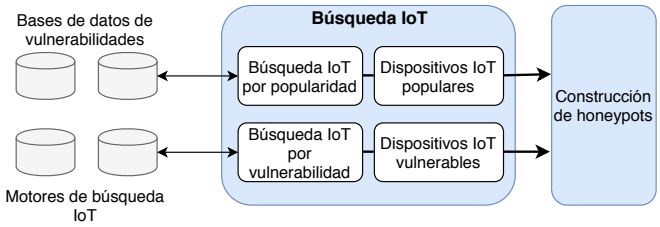

Figura 2. Fase de búsqueda IoT

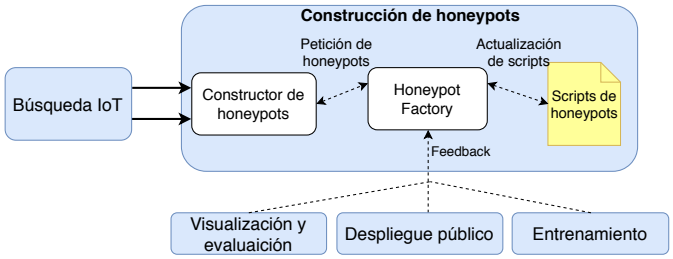

Figura 3. Fase de construcción de honeypots

protocolo TR-064 en un puerto abierto [11]), o bien se centran en ciertos protocolos, personales [12] o industriales [13].

III. Metodología

Aunque, como hemos visto, hay soluciones de honeypots IoT disponibles, éstas por sí solas no son suficientes. No basta con saber cómo se lleva a cabo el despliegue de una herramienta, sino que se necesita saber qué dispositivos simular de cara a los atacantes (cuáles pueden ser más atractivos, o cuáles pueden presentar un comportamiento más interesante), cómo evaluar su funcionamiento, y cómo sacar de esta evaluación información útil para futuras mejoras. A estos interrogantes pretende dar respuesta la metodología aquí propuesta, que va más allá de una simple arquitectura, abogando por cubrir la totalidad del proceso.

III-A. Búsqueda IoT

Los primeros desafíos abordados son hacer el honeypot atractivo para los atacantes, y hacer que el conjunto de honeypots desplegado sea representativo del contexto IoT. El primer punto tiene que hallar un compromiso en el realismo del sistema: si se trata de un sistema muy simple, puede ser obvio que es una trampa para el atacante. Sin embargo, si es un sistema complejo, puede perder interés como sistema vulnerable. El segundo desafío tiene la problemática de la heterogeneidad del entorno IoT, ya que los dispositivos son muy distintos entre sí y tienen comportamientos y respuestas muy diversos. Un honeypot que no satisfaga estas condiciones puede ser demasiado específico, o quedar rápidamente obsoleto.

Para cumplir estas condiciones se han usado varias herramientas. En primer lugar, se han utilizado motores de búsqueda especializados en IoT, tales como SHODAN, Censys o Wigle. Estos motores escanean la red y buscan dispositivos IoT, indexándolos para poder buscarlos en base a tags, localización, sus mensajes de bienvenida (*banners*) y otros criterios. Usándolos se han buscado marcas, modelos y tipos de dispositivos. También se ha hecho uso de portales de venta, tales como Amazon, u otros más especializados, como Network Webcams. Aquí se han observado los dispositivos más vendidos en distintas categorías (routers, drones, cámaras IP...), para ver qué modelos son más populares. Una vez se han obtenido los dispositivos populares, tanto en cantidad de unidades conectadas como en ventas, mediante los medios descritos, se procede a la búsqueda de los mismos en portales de vulnerabilidades. En esta ocasión se han usado Exploitee.rs, ExploitDB y CVE-Details, donde se detallan las vulnerabilidades conocidas para cada modelo y versión, su gravedad, y si hay maneras de explotarlas disponibles.

III-B. Constructor de honeypots

Este paso en la metodología se centra en la elaboración de honeypots usando la información recabada en la fase anterior: dispositivos IoT populares, y dispositivos IoT vulnerables, coincidiendo en ocasiones ambos aspectos. Aunque la implementación de honeypots puede depender de muchos factores (su tipo, su dependencia del hardware...), es importante mantener un repositorio de soluciones ya implementadas, para aumentar la eficiencia. Esto podría ahorrar mucho tiempo si, por ejemplo, se tuviera que implementar un honeypot de características similares a uno ya existente, pero cambiando ciertos valores o parámetros. Este escenario probablemente se dé, ya que los resultados de las fases posteriores, retroalimentarán a estas, para perfilar los honeypots ya existentes.

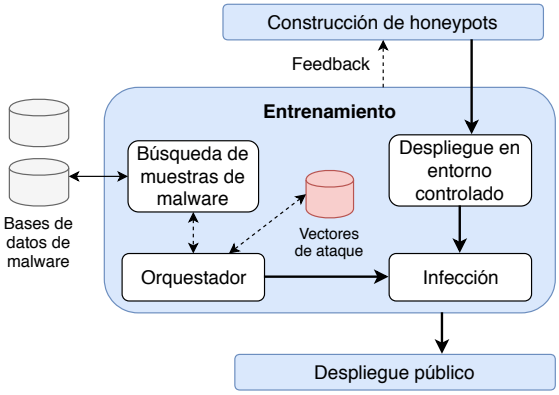

Figura 4. Fase de entrenamiento

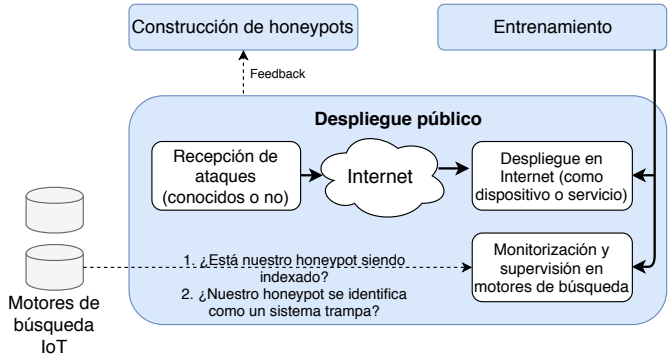

Figura 5. Fase de despliegue público

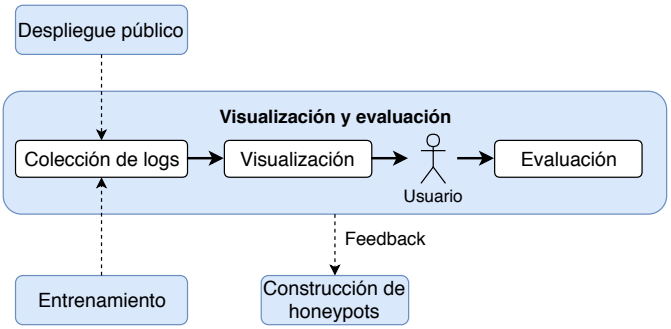

Figura 6. Fase de validación y evaluación

Para clasificar y catalogar estas soluciones, se recomienda usar una Honeypot Factory (Fábrica de honeypots) durante esta fase. Este componente se usará para controlar el acceso y las modificaciones a los archivos que describan la configuración de los honeypots. A su vez, el componente recibirá feedback para clasificar mejor los honeypots y mejorar el Honeypot Builder. Además, el despliegue de los honeypots con esta fábrica, será mucho más eficiente, ya que permitirá desplegar un nuevo honeypot en caso de que uno de los que estén operando se corrompa de alguna manera. En este trabajo, la infección de los sistemas se lleva a cabo de forma manual, por lo que tanto esta fase como la anterior no se ponen en funcionamiento. Sin embargo, son necesarias para que al desarrollar honeypots en el futuro siguiendo esta metodología, éstos cubran todo el entorno IoT y sean útiles.

III-C. Entrenamiento

Durante esta fase, se lleva a cabo el despliegue de los honeypots en un entorno controlado para probarlo. Para ello, el honeypot se infecta usando malware conocido. En algunos casos, infectar un dispositivo no es en absoluto trivial. Uno de los propósitos de este paso en la metodología es evaluar la viabilidad de un honeypot para ser infectado. Hay que destacar que, en algunos casos, es deseable que el honeypot sea lo más vulnerable posible (por ejemplo, para recabar información sobre ataques automatizados u oportunistas), pero en otros debería ser un desafío para el atacante, ya que el honeypot si no presenta unos niveles de seguridad mínimos, las sospechas sobre estar atacando a un sistema trampa crecen.

III-D. Despliegue público

El feedback más útil al sistema implementado mediante esta metodología se obtiene de esta fase. Durante estos pasos, el honeypot se despliega expuesto directamente a internet, disponible para los atacantes. Aunque pueda parecer paradójico, ya que conectar dispositivos IoT directamente a Internet los expone a muchos riesgos, es justo esto lo que motiva este trabajo. Uno de los problemas a los que se puede enfrentar el honeypot desplegado es pasar desapercibido frente a los atacantes, debido a la alta densidad de aparatos IoT disponibles. Y aún peor, si el honeypot es identificado como tal por el atacante, éste podría difundir la información a otros atacantes. El trabajo realizado en las fases previas trata de prevenir esto. Hay que tener en cuenta que se necesita promocionar el sistema trampa para hacer que los motores de búsqueda IoT indexen los honeypots desplegados. Estos sistemas deben ser comprobados a fin de detectar si se ha llevado a cabo correctamente esta indexación, por lo que se requiere una monitorización constante durante esta fase. Por último, los sistemas deben ser realistas. La complejidad de este punto es crucial, puesto que si, por ejemplo, se despliega un honeypot de un dispositivo industrial en un sistema SCADA, para imitar correctamente su funcionamiento, habría que implementar intercambios de comandos con controladores que, en muchas ocasiones, siguen protocolos propietarios, lo cual sería muy costoso de llevar a cabo a la perfección.

III-E. Validación y evaluación

Esta fase es paralela a las de entrenamiento y despliegue público. Sin embargo, los resultados de ambas fases tienen que estar claramente separados y clasificados. Esto nos facilitará saber si los resultados esperados antes de realizar el despliegue se corresponden con los reales obtenidos más tarde. En particular, esta fase está pensada para hacer que la solución sea usable por un administrador o investigador que use el sistema y pueda observar los resultados de manera clara y sencilla. Se ha de enfatizar que los logs pueden ser recogidos en diferentes formatos y esto puede representar un problema a la hora de la visualización, por lo que la traducción entre diferentes formatos se tiene en cuenta en esta fase, para que las herramientas puedan procesarlos correctamente. La metodología también considera el feedback que provee el usuario para mejorar y configurar las plataformas de los honeypots, asegurando así su mantenimiento.

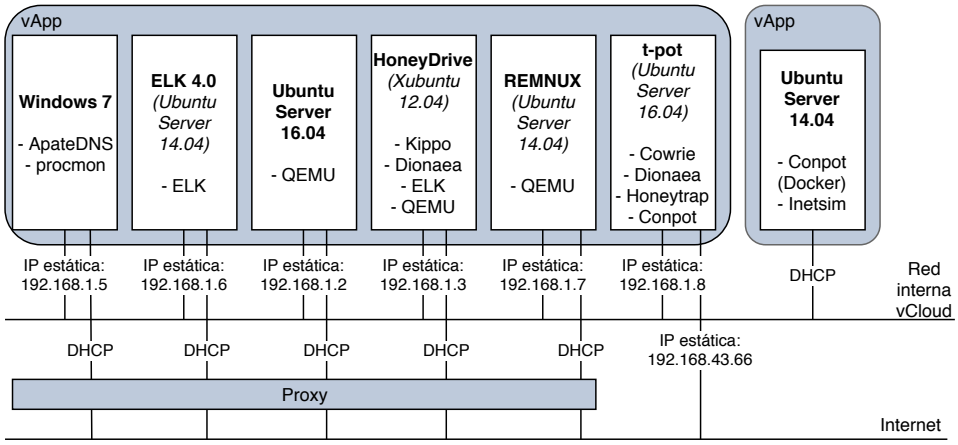

Figura 7. Entorno de despliegue vCloud

IV. Resultados preliminares

En octubre de 2016, un malware específicamente enfocado a dispositivos IoT llamado Mirai causó un ataque distribuido de denegación de servicio contra el proveedor de DNS estadounidense Dyn, haciendo que miles de dispositivos (como routers, DVRs, cámaras IP, etc.) que conservaban las credenciales de acceso por defecto del fabricante formasen parte de una botnet a merced de servidores de *command and control*, que modificaron su comportamiento. La magnitud de este ataque no tenía precedentes y afectó a servicios muy populares. Recientemente, el malware ha dado el salto a plataformas Windows, aunque comportándose de forma distinta. Anteriormente, sólo los dispositivos IoT que fueran visibles de forma pública (expuestos con una IP, como muchas cámaras de seguridad o routers de empresas, por ejemplo) eran susceptibles a Mirai. Ahora, si el malware entra en un equipo Windows, los dispositivos que se encuentren en su red local, incluso tras un firewall, se pueden ver comprometidos [14]. Dada la novedad y repercusión de este ataque, y su aspecto desafiante, se ha usado en la prueba de concepto presentada en este artículo, la cual muestra una infección con Mirai en un equipo Windows dentro de un entorno controlado de pruebas. Se han desplegado otros honeypots en este mismo entorno para llevar a cabo pruebas que quedan fuera del enfoque de este trabajo. Las muestras de malware han sido obtenidas de VirusTotal y Malwr, y el procedimiento seguido para desplegar la infección se ha llevado a cabo según el proceso detallado en SecureList, y se puede ver en Figura9.

IV-A. Entorno

La prueba de concepto se lleva a cabo en el entorno vCloud, el cual permite virtualización, definición de redes y la captura de *snapshots* para restaurar las máquinas a un estado anterior. Además, es un sistema que se puede aislar, de manera que no permite que los ataques desplegados en el mismo se extiendan fuera de él.

Los honeypots desplegados y sus conexiones se muestran en Figura7. Se puede ver cómo se conectan a una red interna que no tiene acceso a Internet. En esta configuración, hay una máquina dedicada específicamente a la centralización de *logs* (registros) para la fase de evaluación y validación, la cual se ha implementado usando el stack ELK, compuesto

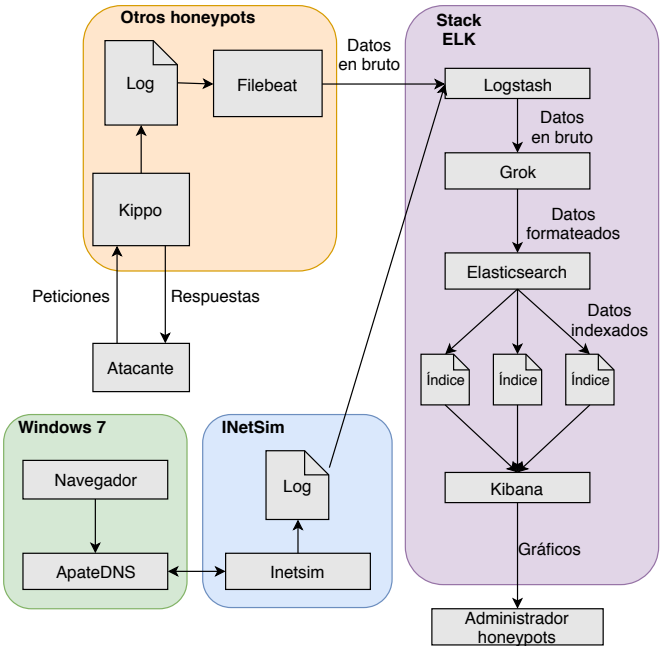

Figura 8. Integración de ELK con los honeypots

por ElasticSearch, Logstash y Kibana. Se han escogido estas aplicaciones por su sencillez de uso, la amplia documentación y la disponibilidad de plugins que aumentan su funcionalidad. La interacción entre estos sistemas es bastante directa: Logstash recibe los registros de las demás máquinas, ElasticSearch los indexa, y Kibana recoge los datos y los presenta de una manera gráfica que resulta sencilla y comprensible. Algunos de los plugins usados han sido Filebeat, para enviar los logs de los honeypots, y Grok, para el *parsing* de los eventos recibidos en los registros. El flujo de la información de los honeypots a la máquina de registro se detalla en Figura8.

Una vez los honeypots estén correctamente conectados a la máquina de centralización de logs, para poder almacenarlos y obtener información de ellos correctamente, y el entorno esté configurado para evitar que el ataque se propague, se puede llevar a cabo la infección.

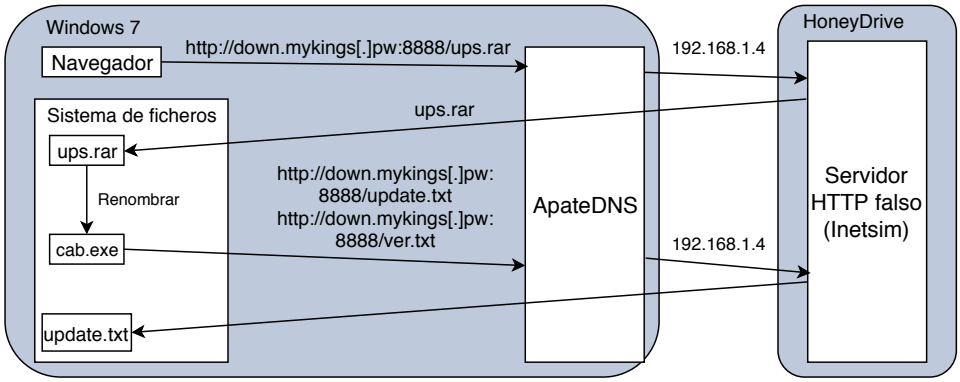

Figura 9. Proceso de infección con Mirai

IV-B. Infección

Dado que el primer paso de la infección es descargar un ejecutable malicioso, y las máquinas no tienen conexión a Internet, tiene que disponerse un servidor HTTP falso. Éste se ha situado en otra máquina del entorno con la herramienta INetSim. Los archivos que se necesitan para la infección se alojan en él, y las peticiones a las URLs donde se alojan los ejecutables de Mirai se redirigen a este servidor mediante ApateDNS. Por tanto, la máquina Windows descarga los archivos como si contara con una conexión.

Una vez se obtiene el primer archivo, se renombra y ejecuta, tal y como se detalla en la guía de infección. Si la ejecución se lleva a cabo con privilegios de administrador, ésta cambia las direcciones DNS primaria y secundaria de la máquina a 114.114.114.114 y 8.8.8.8, respectivamente. En cambio, si se ejecuta como un usuario sin estos permisos, no realiza ningún cambio.

Tras esto, el proceso intenta conectarse de nuevo con URLs del mismo dominio donde se alojaba el ejecutable para descargar los archivos *ver.txt* y *update.txt*, estando disponible el primero en el las bases de datos con muestras de malware a las que teníamos acceso, pero no el segundo. La salida de consola de esta ejecución es "*DNS set ok. ver different web:1.0.0.8 local:, needs update...*", refiriéndose la última parte a que falta el archivo *update.txt*, por lo cual la ejecución se detiene.

Si el ejecutable intenta descargar los archivos sin la herramienta ApateDNS redirigiendo las peticiones y sin conexión a Internet, el mensaje que se muestra es "*get file list failed, exit*", justo antes de detenerse.

Todos los eventos en este proceso se registran en *logs* que se envían al *stack* ELK. Los mensajes se envían mediante el plugin Filebeat, se procesan mediante Grok, se indexan y visualizan. Los mensajes de este proceso se separan del resto con lo que Kibana denomina *patrones*, unos prefijos que ayudan a diferenciar el origen de los *logs*. Además, el tráfico generado en las peticiones HTTP a los servidores que alojan los archivos de Mirai y el intercambio de los mismos se ha registrado mediante la herramienta de captura de paquetes Wireshark.

V. Discusión y trabajo futuro

Hay muchos más casos cubiertos por la metodología de los que se muestran en la prueba de concepto de este trabajo. Un paso fundamental sería la definición de honeypots IoT basada en los criterios discutidos en este trabajo, ya que la decisión de desplegar un honeypot para Mirai fue basada más en su reciente impacto que en los resultados obtenidos al seguir la metodología, pero definir un conjunto de honeypots IoT que desplegar para capturar ataques desconocidos es un objetivo a conseguir en el trabajo futuro. Aunque ya se está trabajando en este aspecto, no podemos dar detalles del proceso hasta que se obtengan datos sustanciales, ya que los resultados podrían verse afectados si se comparte información específica del mismo.

La fase de entrenamiento es esencial antes de que se considere siquiera el despliegue público de los honeypots, por lo que es en lo que se ha centrado este trabajo. Sin embargo, algunas cuestiones que no han sido consideradas en el trabajo pero merece la pena mencionar son las siguientes.

- **Infección desde una botnet Mirai.** En lugar de descargar el malware y realizar la infección manualmente, sería interesante provocar que la máquina se infecte desde una botnet de Mirai ya existente. Este enfoque permitiría ver cómo la máquina es indexada por los mecanismos de monitorización de dispositivos con Mirai. No sería sencillo, puesto que significaría dejar la máquina expuesta al exterior y contribuiría a posibles propagaciones.
- **Infección a red de dispositivos IoT desde Windows.** Crear una honeynet para este caso específico, en el que el equipo Windows esté conectado con dispositivos IoT y ver cómo se infectan.
- **Emulación de dispositivos IoT con QEMU.** Mediante una plataforma de emulación, y disponiendo de firmware de los dispositivos a tratar, podrían crearse máquinas que simularan ser dispositivos IoT vulnerables a modo de honeypot, pero con una interacción completa. Este procedimiento es complejo, porque en muchas ocasiones el firmware presenta una fuerte dependencia de la plataforma hardware a la que se destina (arquitecturas de los procesadores, sistemas de archivos...).
- **Conexión con ELK para que muestre resultados del ataque y sistema de inteligencia.** Consiste en hacer que desde HoneyDrive se envíen las acciones a la máquina ELK, y que el sistema de inteligencia se dé cuenta basándose en el comportamiento (peticiones, mensajes, timing) de que se trata de Mirai. Ya que este malware en concreto se destruye automáticamente si el equipo

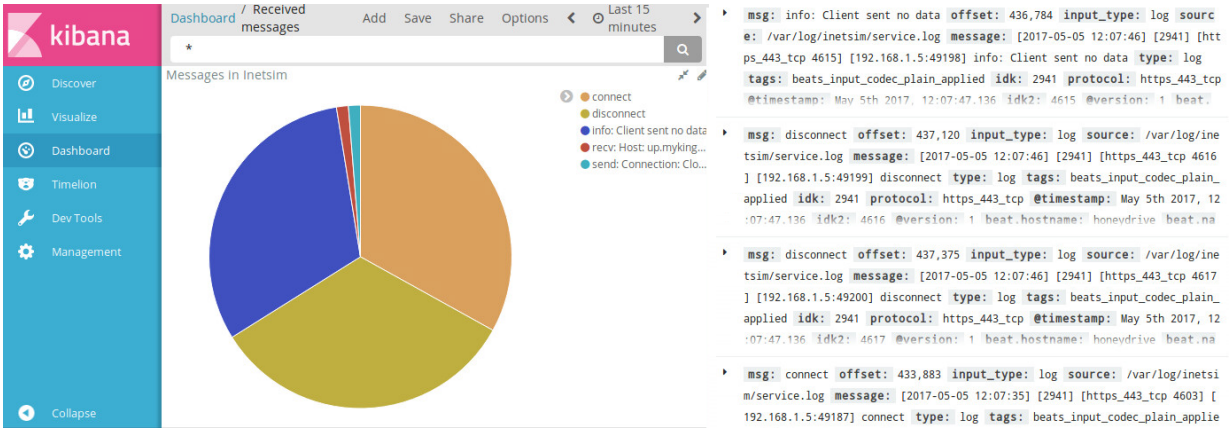

Figura 10. Recopilación y visualización de mensajes de logs en Kibana

se queda sin conexión tras la infección, decidir aislar automáticamente el equipo Windows 7 (infectado) hasta que se copie el código de Mirai, las evidencias y aquellos ficheros relevantes para el análisis del malware.

- **Desplegar "infección" de Hajime o dejarse infectar por una botnet.** Como se ha descrito anteriormente, Hajime se trata de una botnet que basa su funcionamiento en el mismo patrón que Mirai: entra a los dispositivos que conserven las credenciales por defecto del fabricante mediante telnet, usando prácticamente la misma lista de usuarios y contraseñas. Sin embargo, una vez entra, bloquea el acceso desde varios puertos, para evitar ataques vía telnet o TR-064. Al contrario que Mirai, no tiene servidores de *command and control*, sino que se basa en un esquema *peer-to-peer*, lo que hace más difícil detener su avance, y no tiene (por el momento) capacidad de efectuar ataques de DDoS [15].

VI. Conclusiones

A lo largo del trabajo, se ha detallado cómo el escenario IoT no sólo es novedoso y desafiante, sino una realidad palpable. La mayoría de las personas que utilizan internet en su vida diaria contarán con uno o varios dispositivos IoT, e interactuarán con una infinidad a lo largo del día, muchas veces sin ser siquiera conscientes de ello. Los dispositivos IoT no han sido diseñados con las suficientes medidas de seguridad, y se les han dado roles muy importantes, tales como preservar nuestra privacidad o seguridad. La situación resultante es un aparato muy vulnerable llevando a cabo tareas muy sensibles. La dificultad para obtener muestras de malware procedentes de ataques IoT, junto con la posibilidad de simular o emular plataformas sin contar con dispositivos reales, la capa de aislamiento que proveen, el registro de eventos (dado que los ataques IoT tienden a borrar su rastro) y otras particularidades de este escenario, hacen que los honeypots sean la herramienta perfecta para analizar el entorno de seguridad IoT. Aunque las herramientas de honeypots IoT están disponibles, se debe conocer antes qué honeypots se deben desplegar, cómo se puede atraer a los atacantes, y cómo se pueden mejorar los sistemas en base a la información obtenida, por lo que es necesaria una metodología que tenga en cuenta todas las partes de este proceso. Cada fase se detalla meticulosamente en la metodología propuesta, y tienen relevancia justificada a lo largo del proceso, como se demuestra en la prueba de concepto llevada a cabo. En esta prueba se despliega uno de los ataques IoT más devastadores en un entorno controlado, mostrando los datos obtenidos al realizarse dicho proceso, el cual se ha llevado a cabo siguiendo esta metodología.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Economía y Competitividad mediante los proyectos IoTest (TIN2015-72634-EXP) y SMOG (TIN2016-79095-C2-1-R). El segundo autor ha sido financiado por INCIBE a través del programa de ayudas para la excelencia de los equipos de investigación avanzada en ciberseguridad.

Referencias

[1] T. Danova, "Morgan stanley: 75 billion devices will be connected to the internet of things by 2020," *Business Insider*, vol. 2, 2013.
[2] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, no. 9, pp. 51–58, 2011.
[3] G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the iot zombie armies," in *Military Communications Conference (MILCOM), MILCOM 2017-2017 IEEE*. IEEE, 2017, pp. 267–272.
[4] B. Insider, "This one chart explains why cybersecurity is so important," *Retrieved August*, vol. 16, p. 2016, 2016.
[5] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A survey on honeypot software and data analysis," *arXiv preprint arXiv:1608.06249*, 2016.
[6] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "Iotpot: analysing the rise of iot compromises," *EMU*, vol. 9, p. 1, 2015.
[7] J. D. Guarnizo, A. Tambe, S. S. Bhunia, M. Ochoa, N. O. Tippenhauer, A. Shabtai, and Y. Elovici, "Siphon: Towards scalable high-interaction physical honeypots," in *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*. ACM, 2017, pp. 57–68.
[8] T. Luo, Z. Xu, X. Jin, Y. Jia, and X. Ouyang, "Iotcandyjar: Towards an intelligent-interaction honeypot for iot devices," *Black Hat*, 2017.
[9] P. Krishnaprasad, "Capturing attacks on iot devices with a multi-purpose iot honeypot," Ph.D. dissertation, PhD thesis, INDIAN INSTITUTE OF TECHNOLOGY KANPUR, 2017.
[10] A. Radice, "Playing with a mirai honeypot: Mtpot," 2017.
[11] M. Wang, J. Santillan, and F. Kuipers, "Thingpot: an interactive internet-of-things honeypot," 2017.
[12] S. Dowling, M. Schukat, and H. Melvin, "A zigbee honeypot to assess iot cyberattack behaviour," in *Signals and Systems Conference (ISSC), 2017 28th Irish*. IEEE, 2017, pp. 1–6.
[13] A. Jicha, M. Patton, and H. Chen, "Scada honeypots: An in-depth analysis of conpot," in *Intelligence and Security Informatics (ISI), 2016 IEEE Conference on*. IEEE, 2016, pp. 196–198.
[14] K. Lab, "A windows-based spreader for mirai malware has been discovered," 2017.
[15] S. Edwards and I. Profetis, "Hajime: Analysis of a decentralized internet worm for iot devices," *Rapidity Networks*, vol. 16, 2016.