Pontifícia Universidade Católica de Campinas

Centro de Ciências Exatas, Ambientais e de Tecnologia

Faculdade de Engenharia de Telecomunicações

Luiz Henrique Cunha

Análise da invasão em dispositivos IoT vulneráveis utilizando Honeypots

Campinas

2017

Luiz Henrique Cunha

Análise da invasão de dispositivos IoT vulneráveis utilizando Honeypots

Trabalho de conclusão de curso apresentado como exigência para obtenção do título de Engenheiro, ao programa de graduação em Engenharia de Telecomunicações do Centro de Ciências Exatas, Ambientais e de Tecnologia da Pontifícia Universidade Católica de Campinas.

Orientador: Prof. Me. André Luis Peternela

PUC-CAMPINAS

2017

Pontifícia Universidade Católica de Campinas

Centro de Ciências Exatas, Ambientais e de Tecnologia

Faculdade de Engenharia de Telecomunicações

Autor: CUNHA, Luiz Henrique de Oliveira.

Título: Análise da invasão de dispositivos IoT vulneráveis utilizando Honeypots

Trabalho de Conclusão de Curso

BANCA EXAMINADORA

_____

Orientador: Prof. Me. André Luis Peternela

_____

Examinador: Prof. Dr. Omar Carvalho Branquinho

Campinas, 12 de dezembro de 2017

Agradecimentos

Agradeço ao Prof. Me. André Luis Peternela pelo apoio e orientações durante todo o semestre.

Ao Eng.º Leonardo Suzan pela motivação, suporte e opiniões que tiveram grande valor para a consolidação deste trabalho.

À minha mãe Elizabeth, meu pai Luiz Fernando, meus irmãos Gustavo e Lucas por sempre me apoiarem nos momentos difíceis enfrentados longe de casa.

Ao meu avô Fernando, por sempre acreditar em mim e incentivar meus estudos.

RESUMO

A botnet Mirai, composta basicamente por dispositivos embarcados com busybox Linux, causou grande impacto na internet nos últimos dois anos, quando foi responsável pela indisponibilidade de serviços importantes na web, como Twitter, Dyn DNS, entre outros através de ataques DDoS massivos. Neste trabalho, é fornecida uma análise do período de um mês, no qual um Honeypot foi mantido em produção coletando uma variedade de informações. Realizando várias combinações desses dados coletados, pôde-se observar quais são as principais características de uma botnet, quais são os principais tipos de dispositivos alvos e como a uma botnet pode tirar proveito de vulnerabilidades encontradas na maioria dos dispositivos embarcados existentes nos dias de hoje.

Palavras-chave: IoT. Mirai. Botnet. Cyber segurança. Sistemas Distribuidos. Honeypot. Ataque DDoS.

ABSTRACT

The Mirai botnet, mainly composed by Linux busybox embedded devices, have caused major impact on the internet in the last two years, when it was responsible for the unavailability of some important services on the web, such as Twitter, Dyn DNS through massive profile DDoS attacks. In this paper, is provided an analysis of a 1 month cycle on which a honeypot was held in production colecting a variety of information. Performing various combinations of the colected data, we could observe which are the main characteristics of a botnet, what are the main types of targeted devices and how a botnet can take advantage of vulnerabilities found on the majority of the embedded devices prevailing nowadays.

Keywords: IoT. Mirai. Botnet. Cybersecurity. Honeypot. Embedded Systems. DDoS Attack.

SUMÁRIO

1. Introdução

O termo Internet das Coisas já não é novidade nos dias de hoje. O assunto atualmente é muito abordado em trabalhos acadêmicos e pesquisas de mercado. Pessoas estão sempre querendo discutir novas tecnologias e novos padrões envolvendo IoT afim de facilitar a interoperabilidade entre dispositivos e sistemas.

Para começarmos a discutir sobre esse tema, é necessário primeiro definir o que é IoT, tarefa que, segundo o próprio IEEE, não é tão simples de ser feita. Podemos perceber este fato logo de cara, quando descobrimos que o IEEE possui um documento de 86 páginas cuja finalidade, é definir o termo[1]. Consultando este documento, podemos definir IoT de maneira sucinta como sendo um domínio de aplicação que integra diferentes campos tecnológicos e sociais. Apesar da diversidade de pesquisa relacionada a IoT, sua definição ainda permanece imprecisa.

Mas afinal, se tantas pesquisas acadêmicas são feitas voltadas a IoT, principalmente na área da engenharia, qual é a motivação para este trabalho? Não estaria este assunto sendo exageradamente abordado?

Talvez o assunto esteja sim sendo discutido demasiadamente, principalmente na área da Engenharia de Telecomunicações e da Computação. Entretanto, ao longo dos anos na universidade acompanhando os trabalhos de colegas de curso e professores, estudantes de engenharia podem observar que o grande foco sempre foi a operação ou funcionamento dos sistemas propriamente ditos. E fora da academia não é diferente.

As grandes empresas do ramo, com o objetivo de lançar o seu *gadget* antes de seu concorrente, gastam muito recurso e dinheiro para garantir o funcionamento e operação dos seus produtos e aspectos como segurança, gerenciamento de atualizações, correção de bugs e gerenciamento de EOL são desvalorizados, ou às vezes, até inexistentes [2].

Tendo em vista este e outros fatores, que serão discutidos nos próximos capítulos, julga-se de extrema importância dedicar um trabalho acadêmico exclusivamente à segurança em IoT, assunto erroneamente negligenciado nos dias de hoje.

1.1. O Cenário IoT atual

No começo deste ano de 2017, a Gartner, empresa conceituada no ramo de pesquisa e consultoria em tecnologia da informação, previu[3] que até no fim do ano o número de *coisas* conectadas na internet será de 8,4 bilhões, 31% a mais que no fim de 2016

Regionalmente, China, América do Norte e Europa Ocidental estão liderando em relação ao número de dispositivos conectados, as 3 regiões juntas representam 67% da base de dispositivos instalados na internet[3].

Segundo a mesma pesquisa, temos os mesmos 8,4 bilhões de dispositivos, mas agora divididos entre as categorias Consumidor e Indústria, como podemos ver na Tabela 1.

Tabela 1

Categoria	**Nº de dispositivos** **(em milhões)**
Consumidor	5.244
Indústria	3.136
Total:	8.380

Fonte: [3]

Onde na categoria Consumidor se encaixam produtos como sistemas automotivos, smart TVs e set-top-boxes, enquanto na categoria Indústria são englobados produtos como medidores elétricos inteligentes e câmeras de segurança comerciais. Estes últimos que terão considerável participação nessa monografia.

Apesar de mencionado nas sessões anteriores que a preocupação com a segurança é mínima, ainda sim temos algumas empresas cujo foco é cybersegurança voltada a IoT. Se somando em um total de aproximadamente 150 fabricantes e provedores de soluções, nenhum destes apresentam uma solução fim-a-fim robusta. O provedor de serviços mais bem avaliado, segundo a Iot Analytics [4] contempla 12 dos 21 elementos julgados necessários pela mesma. Aproximadamente 2/3 de todas essas empresas estão sediadas nos E.U.A.

Na Tabela 2, temos os principais segmentos de IoT e quanto por cento dessas empresas possuem um produto dedicado a cada segmento.

Tabela 2

Industrial Manufatura	52%
Gov., Serviços Públicos, Defesa	47%
Saúde / Medicina	47%
Energia, petróleo e gás	32%
Carros conectados	30%
Varejo	29%
Transportes terrestres e aéreos	18%
Wearables	15%
Cidades Inteligentes	11%
Casas Inteligentes	10%
Logística	2%
Outros	48%

Fonte: [4]

1.2. As três categorias de desafios para Internet das Coisas

A Internet das Coisas apresenta uma enorme oportunidade para muitos stakeholders em todos os nichos de mercado e indústrias. Muitas empresas estão se organizando para se focar em IoT e na conectividade dos seus futuros produtos e serviços. Para que a indústria IoT prospere, é necessária a superação de três categorias de desafios: Tecnológicos, de Negócios e Sociais[5].

1.2.1. Desafios tecnológicos

Essa parte cobre todas as tecnologias necessárias para fazer com que os sistemas de IoT funcionem corretamente como uma solução autônoma ou parte de um sistema existente. Existem muitos desafios tecnológicos, sendo alguns deles, Segurança, Conectividade, Compatibilidade e Longevidade Padrões e Análise Inteligente[6].

Como segurança é um tópico chave nesta monografia, será discutida mais a frente com muito mais detalhe do que os outros desafios tecnológicos restantes.

- **Conectividade**

Conectar um número tão grande de dispositivos será um dos maiores desafios do futuro da IoT, e vai desafiar a estrutura dos modelos de comunicação atuais e suas respectivas tecnologias[7]. No momento, nós dependemos do paradigma servidor/cliente centralizado

para autenticar, autorizar e conectar diferentes nós em uma rede. O futuro da IoT dependerá muito da descentralização dessas redes.

- **Compatibilidade e Longevidade**

IoT está crescendo em várias direções, com várias tecnologias diferentes competindo para se tornarem padrão. Isso causará dificuldades e vai requerer o desenvolvimento de hardware e software extra para conectar dispositivos[7].

Algumas dessas tecnologias irão eventualmente se tornar obsoletas nos próximos anos, efetivamente tornando seus dispositivos inúteis. Isso é importante, desde que, em contraste com dispositivos computacionais genéricos, dispositivos IoT como câmeras e TVs tendem a permanecer em operação por muito mais tempo e devem ser capazes de funcionar mesmo que seus fabricantes estejam fora do mercado[7].

- **Padrões**

Padrões tecnológicos que incluem protocolos de rede e protocolos de comunicação são a soma de todas as atividades de manipulação, processamento e armazenamento de dados coletados por sensores[8].

- **Análise Inteligente**

O último estágio na implementação de IoT é a extração de *insights* através da análise de dados, onde tal análise é vetorizada por tecnologias cognitivas. Alguns desafios na adoção de análise inteligente são[5]:

- Análise imprecisa devido a falhas nos dados e/ou modelos.
- A habilidade dos sistemas legado em gerenciar dados em tempo real
- A habilidade dos sistemas legado em analisar dado desestruturado.

1.2.2. Desafios de negócios

Existe uma resistência para o investimento em qualquer negócio em IoT sempre que não for encontrado um modelo de negócio sólido. Tal modelo deve satisfazer todos os requisitos para todos os tipos de *e-commerce*, mercados verticais, mercados horizontais e mercado do consumidor. Mas tal categoria sempre foi vítima de um desleixo regulatório e legal[1].

Provedores de soluções fim-a-fim operando em industrias verticais e entregando serviços usando *analytics* em nuvem serão os mais prósperos em termos da monetização de uma grande porção do valor em IoT.

IoT pode ser dividida em três categorias, baseando-se na utilização e base de clientes[5]:

- IoT do consumidor: Inclui os dispositivos conectados, como carros inteligentes, *smartphones,* laptops e sistemas de entretenimento.
- IoT comercial: Inclui coisas como controle de estoque, rastreadores de dispositivos e dispositivos médicos conectados.
- IoT industrial: Cobre a gama de coisas como medidores elétricos conectados, sistemas de economia de água, robôs de manufatura entre outros.

1.2.3. Desafios sociais

Entender a IoT pela perspectiva dos consumidores e órgãos regulatórios não é uma tarefa fácil pelos seguintes motivos:

- O consumidor demanda mudanças constantemente.
- Novos usos para dispositivos, assim como novos dispositivos, crescem em velocidades muito altas.
- Criar e reintegrar funções mandatórias e capacidades é caro e consome tempo e recursos
- Falta do entendimento por parte dos consumidores das boas práticas para a segurança dos dispositivos IoT. Isso inclui por exemplo, a simples ação de mudar as senhas padrão dos dispositivos.

Este último tópico será muito importante mais a frente nesta monografia.

2. Segurança: Um desafio tecnológico

IoT já se tornou um tópico muito sério quando o assunto é segurança e isto está começando a chamar a atenção de empresas de tecnologia e órgãos governamentais ao redor do mundo[5].

O *hackeamento* de babás eletrônicas, *smart fridges*, termostatos, bombas de infusão de medicações, câmeras e até de carros inteligentes está se tornando um pesadelo causado pelo futuro da Internet das Coisas. O número de novos dispositivos conectados à rede cresce cada vez mais, consequentemente, aumentando também o número de vetores à serem utilizados pelos malfeitores afim de realizar novos ataques. Tudo isso devido ao fato de uma quantia considerável desses dispositivos apresentarem falhas de segurança[5].

O maior diferencial em segurança virá devido ao fato de que a IoT vai estar cada vez mais presente em nossas vidas. Preocupações não vão estar mais limitadas apenas à proteção de dados sensíveis. Nossas vidas e nossa saúde agora podem se tornar alvos de cyber ataques[9].

Existem muitas razões por trás da falta de preocupação com a segurança em IoT na atualidade. Pode-se dizer que uma das maiores razões é o fato da indústria estar no seu estado de "corrida para o ouro", onde cada fabricante está mais preocupado em lançar seu próximo *gadget* conectado antes de seu concorrente[2]. Dentro dessas circunstâncias, a funcionalidade e operação são o foco principal e a segurança fica em segundo plano, ou até terceiro ou quarto.

2.1. O Conceito de CID (Confidencialidade, Integridade e Disponibilidade)

O termo confidencialidade é familiar para a maioria das pessoas, não só àqueles da indústria de segurança. Esse termo está relacionado à garantia que uma informação não é divulgada a indivíduos, processos ou dispositivos não autorizados[10].

Garantir que terceiros não tenham acesso a informação sensível é uma tarefe complexa. Primeiro, a informação deve ter proteções capazes de prevenir alguns usuários de acessá-la. Segundo, limitações devem ser colocadas para restringir acesso à informação

apenas àqueles que tem a autorização de tê-la. Terceiro, um sistema de autenticação deve ser implementado para verificar a identidade daqueles que têm acesso aos dados[10].

Quando falamos de integridade no domínio da segurança da informação, geralmente nos referimos à integridade dos dados, ou a garantia de que dados armazenados são precisos e não contenham modificações não autorizadas.

Falhas de software e vulnerabilidades podem levar a perdas acidentais na integridade dos dados e podem abrir um sistema de modo que possa sofrer modificações não autorizadas. Programas tipicamente controlam rigorosamente quando um usuário tem acessos de leitura e escrita em determinados dados, mas uma vulnerabilidade no software pode tornar possível o contorno desse controle[10].

Em respeito à disponibilidade de sistemas de informação, precisamos compreender que tais sistemas devem estar acessíveis aos usuários os quais dependem dos serviços que esses sistemas provêm. Se um sistema cai, ou está respondendo muito vagarosamente, o mesmo não está provendo o serviço que ele deveria.

Ataques relacionados à disponibilidade são um pouco diferentes dos destinados à integridade e confidencialidade. O ataque mais conhecido deste tipo é o ataque de negação de serviço, conhecido como ataque DoS (*Denial of Service)*. Um DoS pode ocorrer de várias formas, mas todas elas interrompem um sistema de impossibilitam o acesso de usuários legítimos ao mesmo[10].

Entender cada um desses três conceitos e como eles se relacionam é fundamental para o desenvolvimento de um sistema seguro. A falha em qualquer um dos componentes pode resultar em um sistema comprometido. Como este trabalho é dedicado a segurança em sistemas IoT, é de suma importância detalhar esses conceitos de forma clara.

Para garantir a segurança de produtos e sistemas em respeito a esses 3 conceitos mencionados anteriormente, profissionais da área de TI fazem o uso de uma ferramenta, ou melhor, um conjunto de ferramentas chamado AAA, uma sigla para autenticação, autorização e auditoria[10].

2.1.1. Autenticação

Autenticação é importante para qualquer sistema que necessite de segurança, já que é o elemento chave para verificar a origem de uma mensagem ou se um indivíduo é realmente quem ele diz ser. Segundo o Comitê Nacional de Sistemas de Segurança dos E.U.A., autenticação é "uma medida de segurança desenvolvida para estabelecer a validação de uma transmissão, mensagem ou origem, ou um meio de verificar a autorização de um indivíduo de receber categorias específicas de informação"[11].

Na tabela 3, temos alguns fatores de autenticação, explicados em detalhes.

Tabela 3

FATOR	EXEMPLO
Algo que você sabe	Informação que o sistema assume que outros não sabem; essa informação deve ser secreta, como uma senha por exemplo
Algo que você tem	Alguma coisa que o usuário possui e que apenas ele tem. Tag RFID é um bom exemplo
Algo que você é	Uma impressão digital, escaneamento de retina e etc.

Fonte: [10]

2.1.2. Autorização

Enquanto autenticação está relacionada a verificar identidade, autorização foca em determinar o que um determinado usuário tem permissão de fazer.

Depois que um sistema de segurança autentica um usuário, esse sistema deve também decidir quais privilégios esse usuário tem, assim como quais ações esse usuário pode tomar. É importantíssimo lembrar que em sistemas de TI, autorização não se aplica apenas a usuários, mas também a programas ou processos[10].

2.1.3. Auditoria

A auditoria em segurança da informação tem o papel de assegurar a qualidade da informação e participar do processo de garantia quanto a possíveis e indesejáveis problemas de falha humana

De acordo com uma pesquisa feita pela VanDyke Sofrware Inc., 46% das empresas de TI que não possuem metodologias de auditoria passam por significantes problemas de segurança. Além do mais, 43% dos representantes das empresas participantes, estavam cientes que deveriam auditar suas redes mais frequentemente[12].

2.2. Criptografia

Na era da informação, quase todo tipo de comunicação está sujeita a algum tipo de intervenção, e como resposta a criptografia se aperfeiçoou rapidamente. Entender como criptografia funciona é importante para qualquer um que queira se assegurar que seus dados e comunicação estão protegidos de intrusos.

Em comunicação de dados, existem basicamente dois tipos de criptografia: a criptografia simétrica e a assimétrica, também conhecida como criptografia de chave pública. Não entraremos em muitos detalhes em relação a criptografia simétrica pois, apesar de ser rápida, é considerada vulnerável pois qualquer um que possuir a chave, pode decodificar a mensagem[10].

A criptografia de chave pública se difere devido ao fato de que o sistema utiliza duas chaves, que são correlacionadas uma a outra, uma dedicada a criptografar as mensagens e outra a decodificar as mesmas. Resumidamente, cada sistema utiliza uma chave para criptografar as mensagens, conhecida como chave pública e uma segunda chave, conhecida como chave privada, para decodificar as mensagens[10].

Um dos esquemas de criptografia assimétrica mais utilizado em sistemas de informação hoje em dia é o RSA[13]. O poder desse esquema se deve ao fato de que o mesmo utiliza grandes números primos para codificação e decodificação da comunicação. Não entraremos em detalhes a respeito da matemática por trás desse esquema, entretanto, um fato importante a se enfatizar é que, quanto maior for tamanho da chave utilizada (em bits), mais segura será a comunicação, pois o poder computacional disponível cresce cada vez mais, tornando ataques de força bruta cada vez mais poderosos. Segundo um estudo realizado[14], uma simples fórmula se apresentou eficiente quando o objetivo era gerar uma chave RSA com um número de bits suficientemente seguro. Tal fórmula pode ser observada abaixo.

$$(ano - 2000) * 32 + 512 = N$$

Onde, *ano* representa o ano em que se deseja gerar a chave e *N* representa o tamanho da chave gerada, em bits.

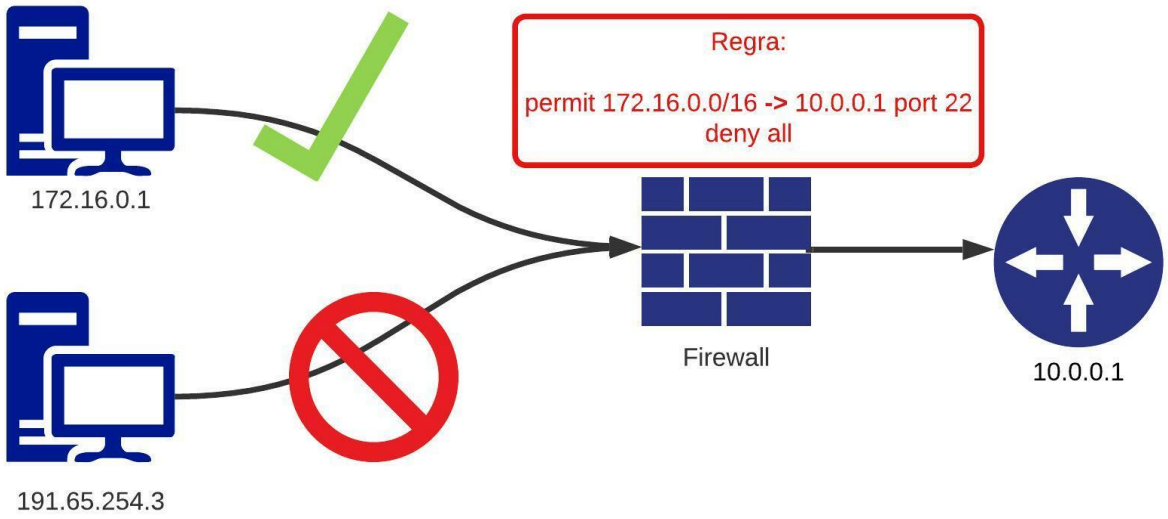
Até os dias de hoje, a maior chave RSA desvendada que se têm registro era composta por 768 bits. Para realizar tal tarefa foi necessária a utilização de 80 processadores trabalhando simultaneamente durante um período de aproximadamente 6 meses[15]

2.3. Firewalls

Os firewalls são dispositivos de rede ou software que separam uma rede confiável de uma rede não confiável, por exemplo, a Internet[10].

É importante discutir o conceito de firewall nesta monografia pois um dispositivo que está conectado à uma rede que esteja protegida por um firewall bem configurado tem muito menos chances de sofrer algum tipo de ataque. Isso pode ser mais simples do que parece. A seguir, temos a Figura 1, que demostra a lógica básica do funcionamento de um firewall.

Figura 1: Funcionamento de um firewall genérico



3. O Cyber Ataque

3.1. Os tipos comuns de Cyber Ataque

- **Malware**

Malware se refere a várias formas de software malicioso, como vírus e ransomware. Assim que um malware infecta uma máquina ou um dispositivo, ele pode causar ações como tomar controle da máquina, monitorar ações e registrar senhas, silenciosamente enviar dados confidenciais da máquina infectada ou da rede para a base de controle, entre outros[16].

- **Phishing**

Em um ataque de *phishing*, o atacante geralmente envia um email que parece ser de uma entidade confiável. O email parece legítimo e geralmente envolve certa urgência. Neste email conterá um arquivo anexado a ser aberto ou um link a ser clicado. Ao executar o arquivo baixado, o alvo instala o malware em sua máquina. No caso do link, o alvo é direcionado a um site que parece ser confiável e nesse site constam campos para login com credenciais, e, quando o alvo entra com suas credenciais, o site na verdade salva as credenciais para uso malicioso posterior[16].

- **Ataque de injeção SQL**

SQL, do inglês *Structured Query Language*, ou Linguagem de Consulta Estruturada é uma linguagem de programação usada para comunicação com base de dados. Muitos servidores que armazenam dados críticos para websites e serviços usam SQL para gerenciar os dados em suas bases de dados[16].

Um ataque de injeção SQL almeja esse tipo de servidores, usando código malicioso para fazer com que o servidor exponha informações que normalmente não deveria. Isso pode ser muito sério quando o servidor armazena informação privada de clientes como cartão de crédito, usuários e senhas, entre outros.

Resumidamente, esse tipo de ataque funciona quando se explora qualquer uma das vulnerabilidades conhecidas do SQL que permitem que o servidor SQL execute código malicioso. Por exemplo, se um servidor web é vulnerável a esse tipo de ataque, é possível que

o atacante vá ao campo de pesquisa do site e digite um código que pode forçar o serviço SQL do servidor a retornar todos os usuários e senhas armazenados no site[16].

- **Ataque Man-in-the-middle**

O conceito por trás do ataque man-in-the-middle é bastante simples. Em sua forma mais simples, o ataque apenas necessita que o atacante se coloque entre as duas partes que estão se comunicando na rede e que seja capaz de interceptar as mensagens sendo trocadas entre as duas partes e se passar por uma dessas partes[16].

3.2. Força bruta

Um ataque de força bruta é um método de tentativa e erro utilizado para a obtenção de informação, como senhas de usuários ou número PIN. Em um ataque de força bruta, é utilizado software para a geração de um grande número de tentativas consecutivas afim de acertar as credenciais que um sistema utiliza[17].

Devido ao fato de que atualmente existem ferramentas poderosas que permitem o uso de força bruta para cyber ataques, concluímos que um sistema com autenticação baseada em senha será tão seguro quanto a senha com a qual o mesmo conta[10].

Quando falamos de sistemas com autenticação por senhas, não é incomum que esses sistemas armazenem as senhas em uma forma criptografada ou em forma de *hash*, pois caso um atacante comprometa a base de dados de usuários e senhas, o mesmo não conseguirá descobrir as senhas imediatamente sem antes recuperá-las usando a função inversa do sistema de criptografia.

Para facilitar o entendimento, uma *hash* é basicamente uma função que recebe objetos como entrada e retorna como saída uma string ou um número. Geralmente essas strings de saída são relativamente pequenas[46].

No caso onde o sistema armazena a senha como uma *hash* da string de texto, não existirá uma função inversa. Isso limitará as possibilidades de um hacker, restando apenas a possibilidade de ataques de força bruta.

Usuários de sistemas geralmente utilizam as senhas mais simples possíveis, pois assim serão mais fáceis de serem lembradas[10]. Dessa maneira, optam pela escolha de uma senha relacionada a um fato fácil de ser lembrado e que cumpra com os requisitos do sistema.

Baseado nesse fato, foram criadas as chamadas *dictionaries* ou *word lists* que nada mais são que listas gigantescas em formato de texto compostas por strings representando as senhas mais comumente utilizadas e suas variações[10].

O ataque de força bruta utiliza sistematicamente essas *word lists* como base para tentar passar pela autenticação de um sistema. O processo do ataque pode levar bastante tempo, e esse tempo depende basicamente de 2 fatores: o tamanho do conjunto de caracteres permitidos e o tamanho máximo permitido para a senha[10].

Quando falamos de dispositivos IoT, as coisas se tornam mais simples e isso se deve ao fato de que os fabricantes desses dispositivos tais como câmeras IP, DVRs, roteadores, entre outros, utilizam como parâmetros de autenticação nomes de usuário e senhas predefinidos, também chamados de *factory default*, ou padrões de fábrica[47].

Isso torna possível que as *word lists* usadas para ataques direcionados a dispositivos IoT seja absurdamente curta[10].

Podemos observar na Tabela 4 a demonstração de uma simples *word list* contendo algumas combinações de usuários e senhas padrões mais utilizados em dispositivos IoT.

Tabela 4

admin	admin
admin	1234
admin	(sem senha)
admin	password
ubnt	ubnt
root	root
root	admin
root	888888
root	default

root	(sem senha)
default	default

Fonte: [10]

Este capítulo é extremamente importante para esta monografia, visto que o experimento, apresentado posteriormente no capítulo 6, se baseia em sistemas vulneráveis que utilizam esse tipo de credencial de acesso.

3.2.1. DoS

Um ataque DoS, ou Denial of Service, é um evento que ocorre quando um atacante realiza alguma ação que impede que usuários legítimos de acessarem determinados sistemas, dispositivos ou recursos de rede[18].

Nesse tipo de ataque, tipicamente ocorre um *flood* em servidores ou redes com tráfego que acaba causando a saturação dos recursos e consequentemente tornando o acesso legitimo difícil ou impossível.

Enquanto um ataque que causa um *crash* em um servidor pode ser resolvido apenas realizando um *reboot* no mesmo, ataques de *flood* podem ser mais difíceis de serem tratados[18].

Segundo a US-CERT, alguns indicativos de que um ataque DoS pode estar ocorrendo são:

- Degradação na performance da rede.
- Incapacidade de acessar certo website
- Volumes altos de e-mails de Spam.

Existem alguns tipos de ataque de negação de serviço. O Ataque DDoS, ou Ataque de Negação de Serviço Distribuído consiste no ataque em que múltiplos sistemas e dispositivos comprometidos atacam um determinado alvo, como um website ou um recurso de rede, causando a indisponibilidade do mesmo. Este ataque em especifico tem um papel importante nesta monografia, pois a infecção dos dispositivos discutida aqui tem como motivação a criação de uma botnet que será posteriormente utilizada para realizar um ataque DDoS[24].

Outro ataque DoS comum é o ataque DoS amplificado por DNS, onde o atacante gera requisições DNS que parecem ter sido originadas na rede da vítima e envia para servidores DNS mal configurados gerenciados por terceiros. A amplificação ocorre quando o servidor DNS responde para as requisições falsas. As respostas acabam gerando um volume de dados muito maior do que em respostas ordinárias causando como resultado a negação do serviço[19].

3.3. O que motiva um cyber ataque?

O maior motivador para os cyber ataques hoje em dia é, compreensivelmente, o dinheiro. Segundo o site securityintelligence.com o cybercrime se tornará um problema no valor de US$2,1 trilhões em 2019[20].

Porém, dinheiro não é o único motivador quando se trata de cyber ataques. Organizações que possuem sistemas de controle industriais, companhias de energia, indústrias químicas, podem ser alvos de atacantes motivados por sabotagem. Esses cybercriminosos tomam suas atitudes baseando se em crenças ideológicas, patrióticas ou políticas.

Segundo uma pesquisa[21] da Palo Alto Networks, 67% dos hackers do Reino Unido admitiram que dinheiro é seu maior incentivo pelo crime, entretanto, a mesma pesquisa revelou que a média de dinheiro que um cyber criminoso ganha por ano, é em torno de 20 mil libras, valor que não é tão expressivo assim, considerando que um profissional de cybersegurança pode ganhar em média 4 vezes mais. Além disso, 57% dos participantes disseram que leva menos de 24 horas para um criminoso experiente planejar e executar um ataque contra uma organização com uma típica estrutura de segurança em TI, e que, se o tempo do ataque tiver de aumentar para dois dias, 60% dos participantes disseram que moveriam suas atenções para outro alvo. Isso mostra que esses criminosos são oportunistas e tem como alvo empresas com estrutura de segurança frágil.

3.4. Alguns tipos de malware

De acordo com uma investigação da Verizon[22], as nove categorias de incidentes de cybersegurança mais recorrentes são: ataques em aplicação web, intrusão em ponto de venda (PoS), mau uso de privilégios, erros diversos, perda de hardware, *crimeware*, fraudes de cartão de crédito(Skimmers), cyber espionagem e ataques DoS.

O relatório analisou 64199 incidentes, os quais 2260 envolveram violação de dados. O ganho financeiro foi o maior motivador para os ataques, com 89% dos incidentes tendo motivos de espionagem ou financeiro. Na vasta maioria desses ataques, malware estava envolvido.

A seguir iremos discutir um pouco sobre os principais tipos de malware.

- **Virus e Worms**

Vírus consistem em programas maliciosos desenvolvidos para infectar programas legítimos. Assim que uma pessoa instala e executa o programa infectado, o vírus se ativa e se espalha para outros programas instalados no computador antes de tomar alguma outra ação maliciosa como deletar arquivos críticos dentro do SO. Similarmente, um Worm é um programa que é capaz de se transmitir através de uma rede diretamente. Diferentemente do Vírus, um worm não precisa estar atrelado a um outro programa existente[23].

- **Trojan**

Similar ao mito grego, os Trojans se apresentam inofensivos, são utilizados para persuadir as vítimas a instalá-los em seus computadores. Quando instalados, ativam outros softwares atrelados a ele que podem introduzir os chamados *backdoors* permitindo a acesso não autorizado de terceiros na máquina da vítima[23].

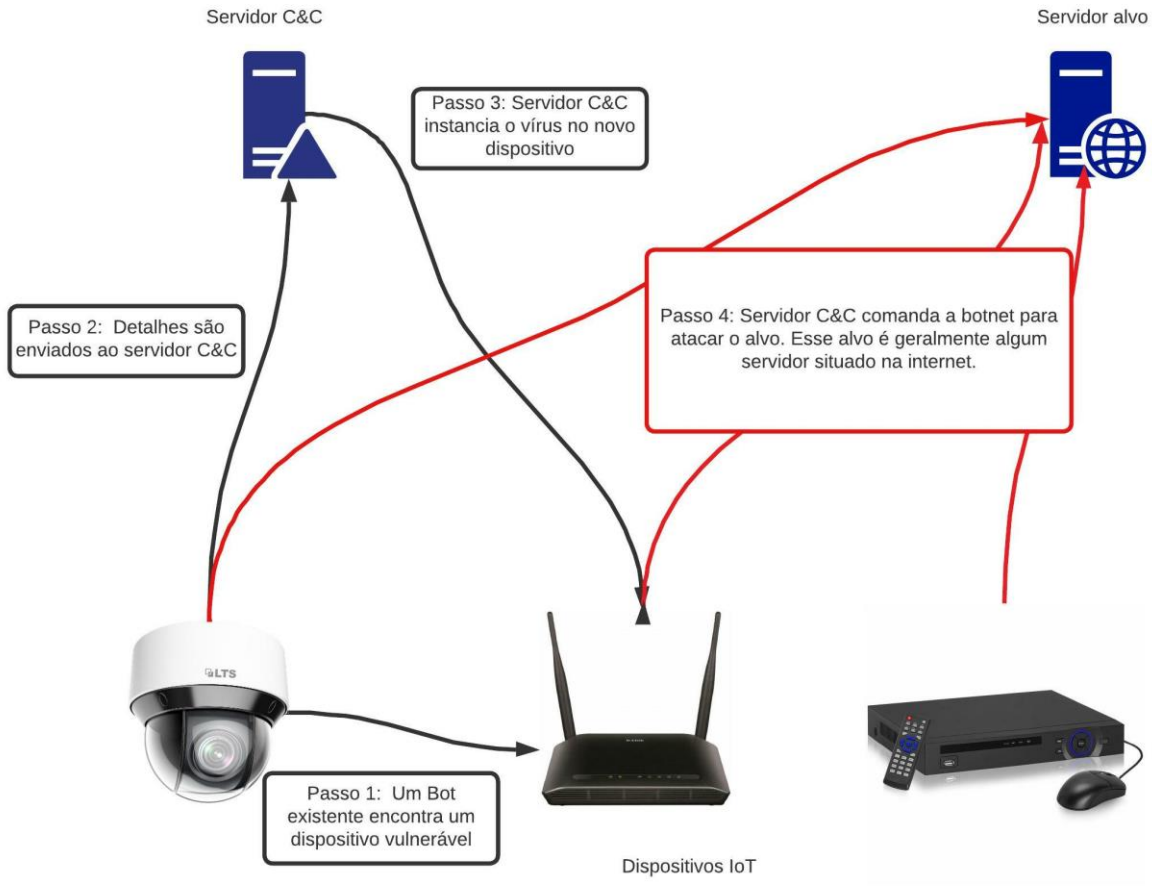
- **Botnets**

Botnets ou bots, tem uma arquitetura de ataque baseada em *reports* para a central. Diferentemente dos malwares comuns, esse tipo de malware não toma nenhuma ação maliciosa após infectar um disposotivo. Invés disso, apenas espera receber comandos de um servidor de comando e controle (C2) controlado pelo atacante[23].

3.5. Arquitetura da operação de uma botnet

Em uma arquitetura genérica de uma botnet existem 2 componentes principais: o bot propriamente dito e a central de comando e controle (C&C). O bot, também chamado de vírus, contém os vetores de ataque. O bot Mirai, que será o mais estudado nesta monografia, possui por volta de dez vetores que podem ser utilizados e um processo de escaneamento que constantemente procura por outros dispositivos a serem comprometidos[24]. A C&C é um componente separado que controla os dispositivos comprometidos (bots), enviando instruções para os mesmos para eventualmente lançarem os ataques contra as vítimas. Na Figura 2, temos o funcionamento de uma botnet ilustrado de uma maneira simplificada.

Figura 2: Diagrama resumido do funcionamento de uma botnet



O processo de escaneamento acontece continuamente em cada bot, utilizando o protocolo telnet (na porta 23 ou 2323) ou o protocolo ssh (na porta 22 ou 2222) para tentar o login em endereços IP aleatórios. Essa tentativa de login se baseia em uma lista de várias combinações de usuário e senhas padrão, como discutido no capítulo 3.1.1. Quando o login é bem-sucedido, a identidade do novo Bot e suas credenciais são enviadas para o servidor C&C.

O servidor C&C suporta uma simples interface de linha de comando que permite o atacante a especificar um vetor de ataque, o endereço IP da vítima e duração do ataque. O C&C também espera que os bots existentes enviem endereços IP de novos dispositivos descobertos e suas credenciais, o qual utiliza dessas informações para copiar o código malicioso nesses novos dispositivos para criar novos bots.

3.5.1. O Código Mirai

O malware Mirai, provavelmente é o malware mais bem conhecido no gênero de malwares formadores de botnets. Isso porque, o mesmo fora usado para ataques DDoS de altíssima escala nos últimos anos, mostrando ao mundo volumes de ataque sem precedentes[25].

O código-fonte Mirai foi divulgado em fóruns de hackers e depois depositado no GitHub[26]. O que torna mais fácil a análise de seu comportamento por parte de acadêmicos e entusiastas do ramo.

Este malware foi desenvolvido para suportar diferentes arquiteturas de CPU (x86, ARM, Sparc, PowerPC, etc) para cobrir as várias CPUs implementadas em dispositivos IoT[24]. A imagem do malware em si é pequena, e o mesmo implementa algumas técnicas para permanecer não descoberto e para dificultar o entendimento de seus mecanismos internos através de engenharia reversa.

Uma vez que o malware é carregado na memória do dispositivo, ele deleta a si mesmo do disco desse mesmo dispositivo. A partir disso, o Mirai permanecerá ativo até que o dispositivo, que agora é um bot, seja reiniciado[27].

Imediatamente depois do *reboot* o dispositivo está livre do malware, entretanto, em questão de minutos o dispositivo pode ser novamente descoberto e infectado.

Os vetores de ataque são altamente configuráveis a partir do C&C[24], porém por padrão o Mirai tende a manter aleatórias as variáveis como número de porta e número de sequência nos pacotes de ataque. Na tabela 5 podemos observar os 10 tipos possíveis de vetores de ataque utilizados pelo Mirai.

Um dos blocos do código, consiste em uma lista de endereços IP os quais o malware não deve se interagir. O real motivo dele evitar essas redes não é conhecido, mas acredita-se que no geral pode ser devido à questões de segurança e sigilo. Além disso, ele também evita interação com endereços IP internos.

Tabela 5: Endereços IP evitados pelo Mirai

127.0.0.0/8	Loopback
0.0.0.0/8	Endereço inválido
3.0.0.0/8	General Electric
15.0.0.0/7	Hewlett-Packard
56.0.0.0/8	Serviço Posta dos E.U.A
10.0.0.0/8	Rede interna
192.168.0.0/16	Rede interna
172.16.0.0/14	Rede interna
100.64.0.0/10	Reservado IANA
169.254.0.0/16	Reservado IANA
198.18.0.0/15	Uso especial IANA
224.*.*.*+	Multicast

Fonte: Código fonte do Mirai

Estudos anteriores, observaram que a partir do momento que um novo dispositivo vulnerável é descoberto, o servidor C&C acessa esse dispositivo e faz uma checagem utilizando o comando "/bin/busybox ECCHI" e espera como retorno do sistema "ECCHI: applet not found". Isso indica que esse dispositivo ainda não tem o malware instalado[28].

Depois dessa verificação, o C&C checa a arquitetura do processador do alvo. Se a mesma for conhecida, o C&C carrega um binário compatível com a arquitetura da CPU do alvo. Isso é feito por um dos três métodos: wget, tftp ou transferência direta utilizando comandos echo através do protocolo Telnet.

Após carregado o *payload*, o servidor C&C executa o malware, espera pelo malware entregar como saída a string "listening tun0" e então disconecta[28].

Observações feitas anteriormente também constataram que Mirai tenta terminar qualquer processo relacionado as portas 22, 23 e 80 e então, se associar a essas mesmas portas. Isso é feito para prevenir outros ataques de malwares concorrentes de tomar o acesso desse dispositivo[28].

4. Caracterizando um dispositivo/sistema vulnerável

Por mais que esta monografia se trate de ataques em dispositivos IoT vulneráveis, quando fala-se desse assunto, deve-se lembrar que, dispositivos IoT são dispositivos que se enquadram numa vasta categoria. Observando o código-fonte do Mirai, depara-se com uma wordlist que contempla usuários e senhas padrão de diversos dispositivos e fabricantes. A maioria desses dispositivos consiste em: DVRs, câmeras IPs, roteadores, dispositivos de storage, e etc. Não demora muito para perceber que o foco desse malware são dispositivos com sistema embarcado baseado em Linux.

Tabela 6: Dispositivos de rede e suas respectivas senha parão

Senha	**Dispositivo**
123456	Câmera IP ACTi
888888	DVR Dahua
ubnt	Roteador Ubiquiti
00000000	Impressora Panasonic
realtek	Roteador Realtek
ikwb	Câmera IP Toshiba

Fonte: [29]

Na Tabela 6 podemos observar algumas senhas padrão utilizadas no código-fonte do Mirai e seus respectivos dispositivos.

Um recente estudo realizado pela Universidade de Michigan, em parceria com grandes empresas de tecnologia[29] obteve uma relação dos dispositivos mais infectados na atualidade, separados por tipo de dispositivo. Esses dados foram factíveis de serem obtidos

graças a ferramentas que realizam técnicas chamadas *fingerprinting*[30]. O resultado desse estudo pode ser visto na Tabela 7.

Tabela 7: Tipos de dispositivos mais infectados

CWMP (28.30%)		**Telnet (26.44%)**		**HTTPS (19.13%)**		**FTP (17.82%)**		**SSH (8.31%)**	
Router	4.7%	Router	17.4%	Camera/DVR	36.8%	Router	49.5%	Router	4.0%
		Camera/DVR	9.4%	Router	6.3%	Storage	1.0%	Storage	0.2%
				Storage	0.2%	Camera/DVR	0.4%	Firewall	0.2%
				Firewall	0.1%	Media	0.1%	Security	0.1%
Other	0.0%	Other	0.1%	Other	0.2%	Other	0.0%	Other	0.0%
Unknown	95.3%	Unknown	73.1%	Unknown	56.4%	Unknown	49.0%	Unknown	95.6%

Fonte: [29]

Mas afinal, o que caracteriza um dispositivo vulnerável?

Em suma, são dispositivos com sistema embarcado baseado em Linux (busybox Linux), que possuem algumas nuances[31], como:

- **Serviços SSH e TELNET**

Os serviços SSH e Telnet rodam por padrão, nas portas 22 e 23 respectivamente. Ambos são protocolos de rede utilizados para realizar acesso remoto em hosts conectados na internet[32]. Esses serviços são vastamente utilizados na comunicação entre dispositivos de rede. Como na comunicação de um computador desktop com um modem, por exemplo. Geralmente esses serviços são utilizados para realizar comunicação de gerência e administração de dispositivos.

- **Usuários e senhas padrão**

Como já foi mencionado anteriormente, usuário e senha padrão ainda continuam sendo um dos maiores causadores de problemas de segurança em dispositivos de rede. Tudo isso ocorre porque, quando um administrador de rede ou um consumidor final adquire um novo dispositivo de rede, ele dificilmente irá trocar o nome de usuário e senha.

Isso torna o número de dispositivos conectados na internet com usuário padrão extremamente alto.

- **Poucas atualizações de firmware**

Quando falamos de atualização de firmware em dispositivos IoT, falamos a partir de duas perspectivas. A primeira perspectiva é a do consumidor, onde se parte da premissa de que o mesmo deve realizar atualizações de firmware em seus dispositivos sempre que estiverem disponíveis. Isso se deve ao fato de que, quando um fabricante lança uma nova atualização, além de correção de bugs e aprimoramentos, geralmente se encontram também correção de vulnerabilidades e falhas de segurança.

Agora, partindo do suposto de que um determinado consumidor assuma as boas práticas de atualização constante, temos um outro possível problema, que inclusive ocorre bastante nos dias de hoje.

Os fabricantes estão tão preocupados em lançar seus novos dispositivos inovadores que deixam de dar a devida atenção à correção das vulnerabilidades dos dispositivos atualmente em produção. E menos ainda quando o dispositivo já saiu de linha e se encontra em fase *End of Life* (EOL). Ou seja, a falta de uma gerência de EOL sólida também é um problema comum hoje em dia[2].

Quando falamos sobre atualização de software e segurança, é interessante lembrar o caso do Windows XP.

De acordo com um estudo realizado pela Net Applications[33] o Windows XP é o terceiro sistema operacional mais utilizado na atualidade, com uma contribuição de 7,04% do Market Share. Curiosamente, as atualizações de segurança e suporte da Microsoft para esse Sistema Operacional pararam de ser fornecidas em 8 de abril de 2014[34]. Não é difícil imaginar o mundo de oportunidades para malfeitores explorarem vulnerabilidades do sistema, sem a preocupação de que as vulnerabilidades possam ser corrigidas.

5. Ferramentas para mitigação de atividades maliciosas

Nesta seção, avalia-se técnicas genéricas para detecção disponíveis para operadores de redes e ISPs. Não há o foco em técnicas que requerem instalação no usuário final porque geralmente os ISPs não tem acesso aos mesmos.

5.1. Netflow

O Netflow, um protocolo originalmente desenvolvido pela Cisco, pode ser usado para coletar métrica de tráfego em um roteador ou switch. O Netflow se tornou um termo genérico na indústria para se referir à análise de tráfego. Hoje em dia, muitos fabricantes de hardware têm suporte a Netflow.

Esse protocolo é capaz de caracterizar fluxo de tráfego até a camada 4 da pilha TCP/IP. Administradores de rede utilizam esse protocolo para compreender padrões de tráfego e monitorar utilização de banda. Ele também é utilizado para localizar malfeitores na rede e para algumas outras finalidades[28].

Como falado anteriormente, Netflow caracteriza fluxo de tráfego, que são então agregados em entradas nos pacotes NetFlow. Um *flow* é uma sequência unidirecional de pacotes que compartilham propriedades em comum. A exata definição de um *flow* se encontra na RFC3954 e 6437[35].

As propriedades comuns mencionadas nas RFCs dependem do protocolo de exportação do *flow* específico e a versão. As propriedades incluem no mínimo: Interface de origem, endereços IP de origem e destino, protocolo camada 4, número da porta de origem e destino.

NetFlow é uma ferramenta bastante útil para detectar atividade em certas fases do ciclo de vida de um malware, dependendo de certos fatores.

Quando está em operação em uma rede na qual existem dispositivos infectados, essa ferramenta pode ser útil para capturar comportamento de escaneamento, pois vários pacotes são gerados nesse evento.

5.2. Captura e análise de pacotes

A captura e análise de pacotes, também conhecida como *packet sniffing*, descreve o processo da captura e intepretação dos dados que fluem através de uma rede.

Algumas ferramentas bem conhecidas para essa finalidade são o tcpdump e o Wireshark. Eles permitem ao usuário escolher múltiplas interfaces de rede a serem monitoradas. As interfaces escolhidas então, são colocadas em modo promíscuo, no qual passa todos os frames recebidos para a CPU, ao invés de apenas os frames destinados a ela[28].

Em comparação com o NetFlow, mais dados do cabeçalho podem ser analisados. Além disso, dados de NetFlow são apenas baseados na informação no cabeçalho, e não no payload. Com captura e análise de pacotes é possível inspecionar o payload também.

A maioria das ferramentas podem ser utilizadas para análise de dados em tempo real ou de dados que foram previamente capturados e salvos em um arquivo no formato .pcap. Os dados então são dissecados e filtrados usando uma combinação de seletores especificados pelo usuário para que o mesmo possa analisar apenas a informação necessária.

A captura e análise de pacotes é eficiente para detectar todas as propriedades baseadas em rede de um malware, mas isso vem a um certo custo. Para redes grandes com grande quantidade de banda, muitos recursos são necessários para inspecionar e armazenar todos os pacotes. Cuidado também deve ser tomado para evitar problemas de privacidade relacionados a inspeção detalhada de pacotes.

Tudo isso torna a captura de pacotes especialmente útil para análise de tráfego em tempo real em hosts específicos.

5.3. Honeypot

Honeypots são sistemas desenvolvidos para encorajar atacantes a realizarem tentativas de ganhar acesso a esses mesmos sistemas. Para isso, esses sistemas simulam algum tipo de vulnerabilidade, como por exemplo senha fraca ou vulnerabilidade de software e eles geralmente são desenvolvidos para parecerem servidores importantes, para assim atrair mais atenção dos malfeitores. O principal objetivo de um Honeypot é analisar as técnicas utilizadas pelos hackers.

Os Honeypots são frequentemente classificados pelos seus níveis de interação, que podem variar de baixo (sem interação, apenas log), médio (alguma interação, mas nenhum

código arbitrário é executado) e alto, onde o Honeypot é basicamente um ambiente enjaulado capaz de oferecer quase todas as features de um dispositivo real. O risco de segurança aumenta com o nível de interação[36].

Nos experimentos deste trabalho, será utilizado o Honeypot Cowrie[37].

O Cowrie é um Honeypot de média interação com suporte a Telnet e SSH desenvolvido em Python. Ele é baseado no Honeypot Kippo mas com algumas funções extras.

6. Experimento: Análise da arquitetura de um ataque de malware em dispositivos IoT

De todas as técnicas de análise e mitigação de ataques maliciosos disponíveis para Sistemas Autônomos e ISPs, faremos uma análise focada na utilização de Honeypot.

Esta ferramenta foi escolhida devida a sua relativa facilidade de implementação e sua vasta gama de recursos disponíveis para realizar a coleta e análise de dados relacionados aos ataques de bots.

O Honeypot utilizado foi o Cowrie, um Honeypot de média interação desenvolvido por Michel Oosterhof[37]. Este Honeypot foi escolhido devido a sua capacidade detalhada de log e sua capacidade de personalização de acordo com as necessidades.

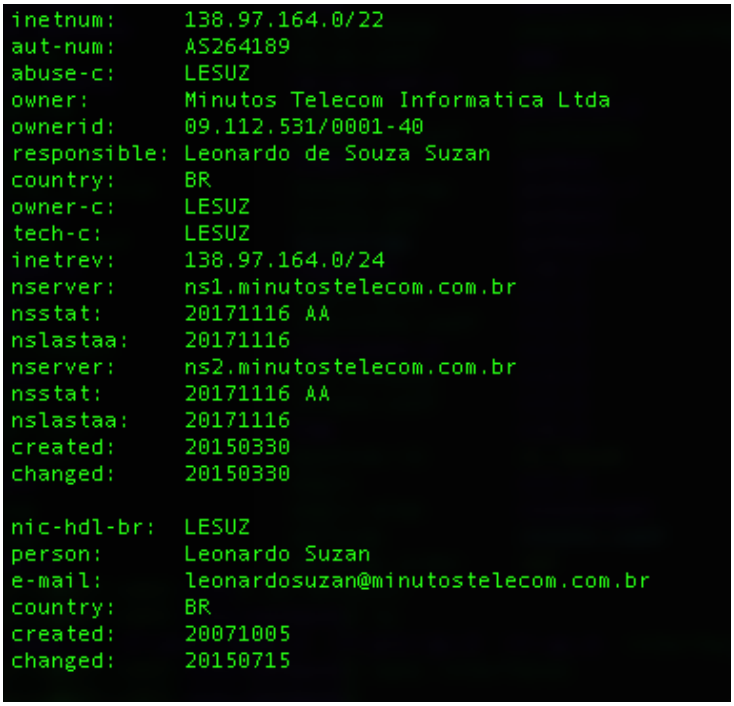
Alguns recursos interessantes encontrados no Honeypot utilizado são:

- Filesystem falso com a habilidade de adicionar e remover arquivos
- Possibilidade de adicionar arquivos falsos para que então o invasor possa usar *cat* em arquivos como */etc/passwd.*
- Log das sessões armazenados em texto e no format *.json.*
- Armazenamento dos arquivos baixados com *wget/curl* para inspeção posterior.

Além disso, se trata de um Honeypot de interação média, o que torna a interação com os bots menos arriscada, já que nenhum binário pode ser executado no servidor. Tudo isso precisa ser avaliado pois o servidor foi instanciado em uma rede em produção pertencente a um provedor de internet de médio porte (Sistema Autônomo), situado na cidade de

Campinas/SP. Qualquer código malicioso executado de forma não desejada no servidor poderia comprometer toda a rede.

Figura 3: Informações sobre o Sistema Autônomo



De acordo com a Figura 3, podemos notar que o Sistema Autônomo possui uma rede 138.97.164.0/22, totalizando um total de 1024 IPs, entretanto, o Honeypot foi implementado em uma subrede, 138.97.164.0/27 (máscara de rede 255.255.255.224).

6.1. Metodologia

Neste subcapitulo, iremos apresentar as metodologias adotadas para a configuraçãodo Honeypot propriamente dito, utilizado na coleta de dados importantes a serem analisados posteriormente, assim como a configuração da plataforma que será utilizada para a mineração e análise desses dados.

6.1.1. Configuração do Honeypot

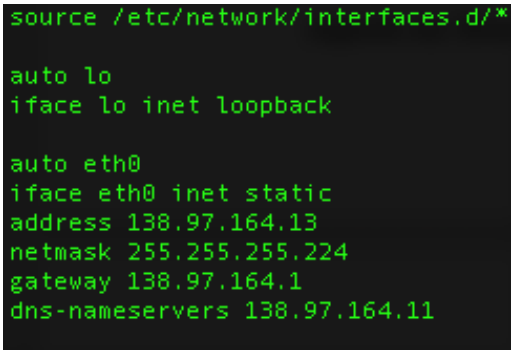
Utilizou-se o software Citrix para a realização da instância da máquina virtual que hospedaria o servidor do Honeypot.

Em relação à configuração da máquina, foram alocados: 60GB de espaço em disco disponível e 1GB de memória RAM. Esse espaço em disco foi escolhido devido a expectativa de um grande volume de dados em forma de log que seria gravado no disco.

Como o Cowrie só é compatível com Linux, foi necessária a escolha de uma distribuição Linux. Neste caso foi escolhido o Ubuntu 16, por ser um sistema robusto, completo, seguro e que possui muito material de apoio e suporte em fóruns na interntet.

Após a instalação do Ubuntu 16 na máquina virtual, o próximo passo foi garantir a conectividade de rede da máquina. Isso pode ser feito editando o arquivo de configuração de interface de rede localizado em */etc/network*, da maneira que se mostra na Figura 4.

Figura 4: Arquivo de configuração de interface de rede



Depois que a nova configuração é aplicada e salva no arquivo, é necessário que seja realizado um reset na rede na máquina. Isso pode ser realizado com os comandos:

$ sudo ifdown –a

$ sudo ifup –a

Feito isso, a máquina deve agora possuir conexão com a internet. Isso pode ser testado com o simples comando *$ ping 8.8.8.8.*

Se houver conexão, será obtido como resposta a latência dos pacotes entre a máquina e o servidor DNS Google (8.8.8.8).

Foi feita também a configuração de servidor NTP para a máquina. Isso foi necessário pois era esperada no fim do experimento a análise de dados em forma de log do honeypot, portanto, é importante que o log esteja com o horário e data corretos.

Feito isso, agora é necessária a criação de um usuário na máquina destinado a realizar as operações do serviço Cowrie. Isso pode ser feito da seguinte maneira:

$ sudo adduser cowrie

$ usermod –aG sudo cowrie

Desta maneira, criamos o usuário cowrie e então adicionamos o mesmo no grupo sudo.

Agora que o usuário já foi criado e a máquina possui acesso a internet, o próximo passo é instalar o Honeypot Cowrie. Antes de tudo, deve se instalar o suporte necessário para o Python.

$ sudo apt-get install git python-virtualenv libssl-dev libffi-dev build-essential libpython-dev python2.7-minimal authbind

Então, deve-se realizar o clone do código-fonte e outros arquivos do Cowrie que se encontram em um repositório do GitHub:

```
$ git clone http://github.com/micheloosterhof/cowrie
Cloning into 'cowrie'...
remote: Counting objects: 2965, done.
remote: Compressing objects: 100% (1025/1025), done.
remote: Total 2965 (delta 1908), reused 2962 (delta 1905), pack-reused 0
Receiving objects: 100% (2965/2965), 3.41 MiB | 2.57 MiB/s, done.
Resolving deltas: 100% (1908/1908), done.
Checking connectivity... done.
```

Fonte: [37]

O próximo passo é criar o ambiente virtual para o Python:

```
$ pwd
/home/cowrie/cowrie
$ virtualenv cowrie-env
New python executable in ./cowrie/cowrie-env/bin/python
Installing setuptools, pip, wheel...done.
```

Fonte: [37]

Feito isso, deve-se ativar o ambiente virtual e instalar os pacotes que estão especificados no arquivo *requirements.txt*:

```
$ source cowrie-env/bin/activate

(cowrie-env) $ pip install --upgrade pip

(cowrie-env) $ pip install --upgrade -r requirements.txt
```

Fonte: [37]

A configuração para o Cowrie é armazenada no arquivo cowrie.cfg. Para executar o serviço com a configuração padrão, não é necessário realizar nenhuma mudança no arquivo. Para trocar o hostname falso do servidor por exemplo, basta realizar a seguinte alteração:

[honeypot]

sensor_name = Honeypot

hostname = ACTi

No caso deste experimento, o hostname escolhido realmente foi ACTi devido à reputação de dispositivos vulneráveis que o fabricante possui[38].

Com todos esses passos feitos, agora temos o serviço cowrie instalado, porém com todas as configurações em padrão. Não faz sentido executar o serviço sem antes realizar as personalizações necessárias. Por isso, nessa próxima sessão expomos as alterações que foram realizadas para tornarem o experimento mais genuíno.

Começaremos expondo as mudanças que foram necessárias a respeito ao acesso ao serviço. Tanto o acesso legítimo de administrador quanto o acesso dos bots e invasores que eram esperados.

Como o acesso telnet e ssh acontecem por padrão nas portas 23 e 22 respectivamente, foi preciso elaborar uma maneira de que o administrador pudesse acessar o servidor remotamente através do protocolo ssh, mas em uma porta diferente da 22, pois o serviço cowrie estaria escutando nessa porta, a espera de um acesso malicioso.

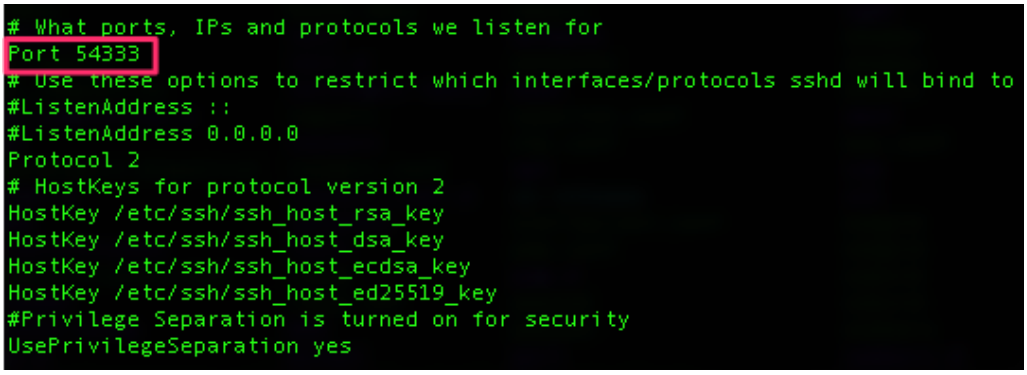
Para configurar o serviço a ouvir nessas portas e ao mesmo tempo possibilitar o acesso legitimo ao servidor, foram realizadas as seguintes modificações.

Primeiro, deve-se abrir o arquivo */etc/ssh/sshd_config* com um editor de texto e editar a porta que o serviço utiliza para um outro número que não seja 22. No caso foi escolhido o

valor 54333 por ser um número alto e que raramente é utilizado por scanners[39]. Dessa maneira garantimos mais segurança.

A Figura 5 apresenta o arquivo de configuração do *deamon* SSH. A seção destacada mostra o parâmetro que define qual porta será utilizada pelo serviço.

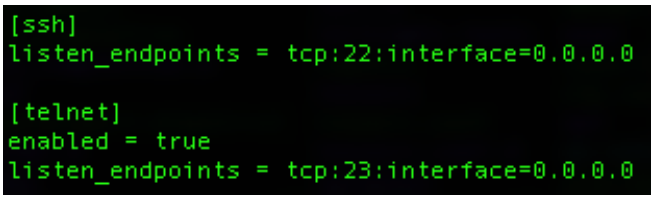
Figura 5: Arquivo de configuração do sshd.



Feito isso, basta reiniciar o serviço ssh com o comando *$ servisse ssh restart* , para que então, o mesmo comece a utilizar a porta configurada.

O próximo passo é configurar as portas utilizadas pelo serviço cowrie. Para isso, deve-se alterar o arquivo de configuração mencionado anteriormente como mostra a Figura 6.

Figura 6: Arquivo de configuração do cowrie.



Por último, devemos atribuir ao cowrie as portas mencionadas, através do comando authbind[40]. Isso pode ser feito realizando os seguintes comandos.

$ touch /etc/authbind/byport/22
$ chown cowrie /etc/authbind/byport/22
$ chmod 777 /etc/authbind/byport/22

O mesmo deve ser realizado para a porta 23.

Como medida de segurança, também foi julgada necessária a criação de uma regra de firewall no servidor Ubuntu para que apenas tentativas de acesso na porta 54333 vindas de um servidor VPN do Sistema Autônomo fossem permitidas.

Isso pode ser feito da seguinte maneira:

$ iptables -A INPUT -p tcp -s vpn.minutostelecom.com.br --dport 54333 -j ACCEPT

$ iptables -A INPUT -p tcp -s 0.0.0.0/0 --dport 54333 -j DROP

Feito isso, qualquer pacote que chegasse no servidor com a porta de destino 54333 seria descartado, a não ser que fosse originado na VPN do Sistema Autônomo.

Para verificar se a regra foi aplicada, basta observar o output do comando

$ iptables –L que lista as regras que estão em produção no sistema. O resultado deve ser como na Figura 7.

Figura 7: Regrais aplicadas no iptables.

```
root@HPT-LUIZ:/home/cowrie# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  vpn.minutostelecom.com.br  anywhere            tcp dpt:54333
DROP       tcp  --  anywhere             anywhere            tcp dpt:54333
```

Para testar se a regra é efetiva, basta tentar o acesso SSH a partir de uma rede que não seja a configurada como permitida.

Em relação ao arquivo de configuração, basicamente foram feitas poucas modificações, e a maioria dos parâmetros foram mantidos no padrão. Os parâmetros que foram modificados foram: *hostname*, *listen_endpoints* para telnet e SSH, como pode ser observado na Figura 8.

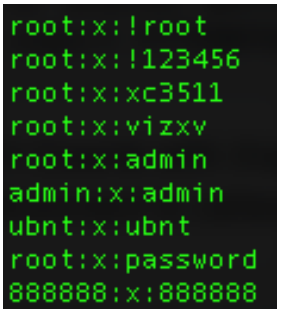
Figura 8: Modificações feitas na configuração do cowrie.

```
honeypot]
sensor_name = Honeypot
hostname = ACTi
interactive_timeout = 300

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

[telnet]
enabled = true
listen_endpoints = tcp:23:interface=0.0.0.0
```

Além do arquivo de configuração, este honeypot vem equipado com um arquivo que permite que o administrador escolha quais combinações de usuário e senha devem permitir que o invasor tenha o falso acesso ao servidor e quais não.

Este arquivo é chamado *userdb.txt* e foi editado como na Figura 9.

Figura 9: Regras de credenciais do cowrie.



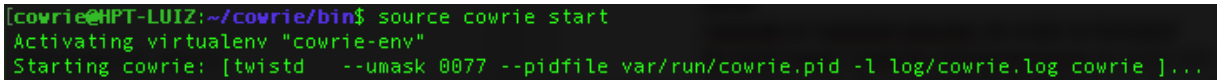
Com essa configuração e sintaxe, o servidor entende que as combinações de usuário e senha *root* e *password* respectivamente permitem o acesso ao invasor, ao passo de que *root* e *123456* não.

A escolha destas combinações de credenciais foi feita baseando-se na *wordlist* utilizada no código-fonte do bot Mirai. Foram escolhidas algumas credenciais a serem permitidas e outras negadas, para que houvesse um certo número de tentativas por parte dos bots e então nos fosse fornecido um volume mais consistente de informações relacionadas a credenciais.

Com a configuração básica realizada, o próximo passo é executar o serviço cowrie para que então a etapa de testes do honeypot se inicie.

O serviço do honeypot deve ser iniciado executando o comando *$ source cowrie start*. O output do comando deve ser como na Figura 10.

Figura 10: Output do comando *source cowrie start*



Para checar se o serviço está em execução, deve se executar o comando

$ source cowrie status e deve se obter no output o número do PID do processo.

Uma vez que o serviço foi iniciado, o próximo passo é realizar os testes iniciais. Basicamente, os testes realizados foram relacionados à segurança do sistema.

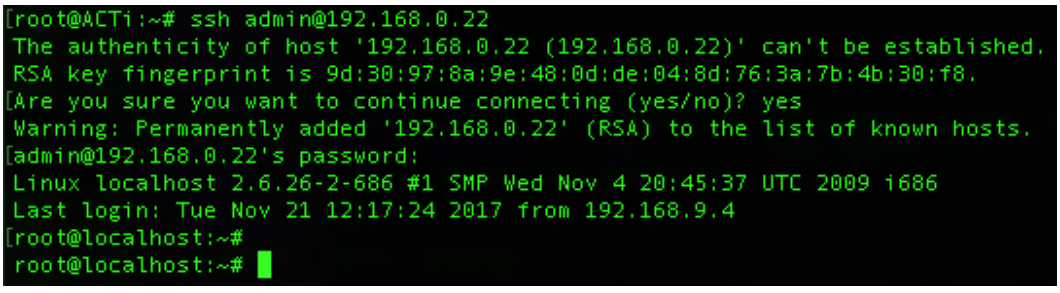
- **Teste 1:** Verificar se realmente um script não pode ser executado dentro do honeypot.

A metodologia adotada para fazer este teste foi realizar o acesso ao honeypot utilizando uma das credenciais permitidas e em seguida realizar o download e tentativa de execução de um arquivo através do utilitário *wget.* Quando esta tentativa foi realizada, observou-se que o download foi feito e o arquivo baixado foi copiado para a pasta de downloads do Honeypot, entretanto o arquivo baixado, que no caso era um script que removeria uma pasta do filesystem, não fora executado.

- **Teste 2:** Verificar se, uma vez que o invasor teve acesso ao Honeypot, ele pode ter acesso à rede interna a partir do servidor.

Para isso, foi realizado um acesso ao Honeypot simulando um invasor e, após o acesso bem-sucedido, foi feita a tentativa de acesso a um host existente na rede interna usando credenciais erradas, e supreendentemente, o Honeypot retorna como output na shell, mensagens de acesso bem-sucedido, mesmo que o acesso não tenha efetivamente ocorrido. A Figura 11 mostra esse evento.

Figura 11: Simulação de acesso à host da rede interna



Feita essa comprovação, foi eliminada a necessidade de criação de uma regra de firewall bloqueando pacotes com origem no Honeypot e destino à rede interna.

Após realizados os testes de segurança, o servidor pôde então ser executado e colocado efetivamente em produção. O servidor permaneceu ativo em produção durante o período de 30 dias corridos, entre os dias 13 de agosto e 13 de setembro de 2017.

6.1.2. Configuração da plataforma Elastic

O Elasticsearch é uma ferramenta para mineração e análise de dados capaz de resolver diversos tipos de casos. Essa plataforma faz o uso de um estilo arquitetônico chamado *REST*, do inglês *Representational State Transfer*, que parte da filosofia de que aplicações web devem usar o protocolo HTTP, como foi originalmente visionado[45]. Nessa arquitetura, consultas são utilizadas com o comando GET e requisições PUT, POST e DELETE devem ser usadas para mutação, criação e deletar respectivamente.

Essa plataforma é composta por três módulos interacionados entre si. Esses módulos são: Logstash, Elasticsearch e Kibana.

O primeiro, Logstash, é um *pipeline* de código aberto utilizado para o processamento de dados que pode ingerir esses dados de múltiplas fontes simultaneamente, transformar esses dados e então enviar para o Elasticsearch[41].

O Elasticsearch então, recebe esses dados e organiza em forma de índices, que podem ser personalizados de uma maneira que o usuário julgue necessário, para que então, finalmente esses dados possam ser visualizados através do Kibana, que é o *front-end* da plataforma.

Para simular um ambiente de sistemas distribuídos, a plataforma foi instalada em uma máquina virtual que não era a utilizada pelo Honeypot.

Essa máquina foi dimensionada com 20GB de espaço em disco e 4GB de memória RAM, visto que a plataforma faz o uso do Java versão 8 ou superior e possui processamento gráfico.

A migração dos dados coletados no honeypot para o servidor Elasticsearch foram feitas através do comando *scp (secure copy)*.

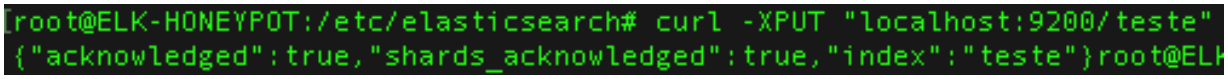
Após feita a instância da máquina virtual, que nesse caso também foi Ubuntu 16, foi feita a instalação dos serviços Elasticsearch versão 5.6.4. Esta etapa de instalação não será mostrada em detalhes por ser um processo de instalação como qualquer outro em sistemas baseados em Debian. De qualquer maneira, existem alguns guias na internet que podem ser usados[42]

Para a configuração, alguns detalhes precisam ser levados em conta. Primeiramente, foi importante levar em consideração que os 3 serviços estariam sendo executados na mesma máquina.

Primeiramente, foi editado o arquivo de configuração *elasticsearch.yml* localizado em */etc/elasticsearch/* para que o serviço fosse executado na porta 9200, no endereço 127.0.0.1 (localhost).

A primeira coisa necessária é a criação de um índice Elasticsearch para onde os dados possam ser armazenados. Para teste, foi realizada a criação de um índice chamado "teste" utilizando o seguinte comando.

Figura 12: Output do comando XPUT



Logo em seguida podemos ver o output "*acknowledged: true"* o que indica que o índice foi criado com sucesso. A Figura 12 demonstra esse evento.

Para a criação de um índice a ser usado para armazenar os dados do honeypot, foi utilizado um script que pode ser encontrado em um repositório do GitHub[43].

Depois de executar esse script como usuário root, o próximo passo é fazer com que todos os dados sejam indexados nesse índice do Elasticsearch. Isso pode ser feito utilizando outro script[44] e então utilizando o comando a seguir.

$ for JSON in /home/mnt/JSON/.*; do ./bulk_index.sh $JSON; done*

Onde */home/mnt/JSON/* estão localizados os logs em formato *json* e o script utilizado tem o nome *bulk_index.sh*. É importante lembrar que o comando deve ser executado no diretório onde o script se encontra.

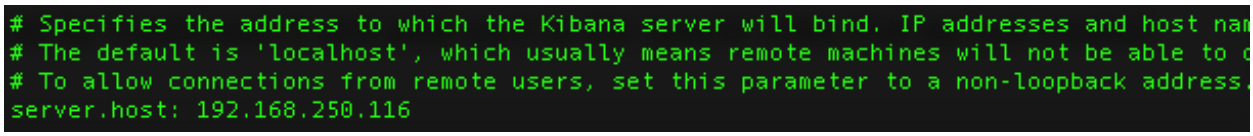
Após executado esse comando, foi necessária a espera de aproximadamente 15 minutos, até que todos os 1,2GB de dados fossem indexados no Elasticsearch.

O próximo passo é configurar o Kibana para que então esses dados possam ser visualizados. Isso pode ser feito editando o arquivo *kibana.yml* localizado em */etc/kibana/*.

O único parâmetro modificado foi *server.host*. Esse parâmetro indica o endereço do host que estará executando o serviço. Esse parâmetro vem como padrão o endereço de

localhost, mas como a máquina virtual não possui interface gráfica, ela precisaria ser acessada via HTTP por uma outra máquina. Visto isso, alimentamos esse parâmetro com o endereço IP que a máquina está utilizando no momento, como podemos ver na Figura 13.

Figura 13: Arquivo de configuração do Kibana



O serviço Kibana é executado por padrão na porta 5601.

Feito isso, o próximo passo é realizar um acesso HTTP via browser através de outra máquina que esteja na mesma rede para que então, o Kibana possa finalmente ser acessado em seus recursos gráficos.

6.2. Resultados

A análise dos logs do honeypot assim como os arquivos coletados, permite que seja obtido um grande volume de informações valiosas relacionadas aos bots/malwares. Dependendo do comportamento do malware, é possível recolher informações como:

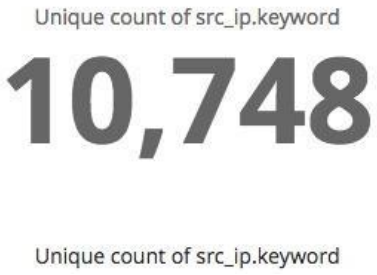
- Código do malware propriamente dito
- Tipo de dispositivo alvo, baseado nas combinações de credenciais
- Informação de localização (endereços IP, hostnames, números de porta) das centrais de comando, invasores, alvos, etc.

Durante o período de produção do honeypot, foram coletados o total de:

- 1,2GB em forma de logs em arquivos de texto e *.json.*
- 31MB em forma de arquivos baixados pelos bots (scripts, códigos executáveis, binários, etc)

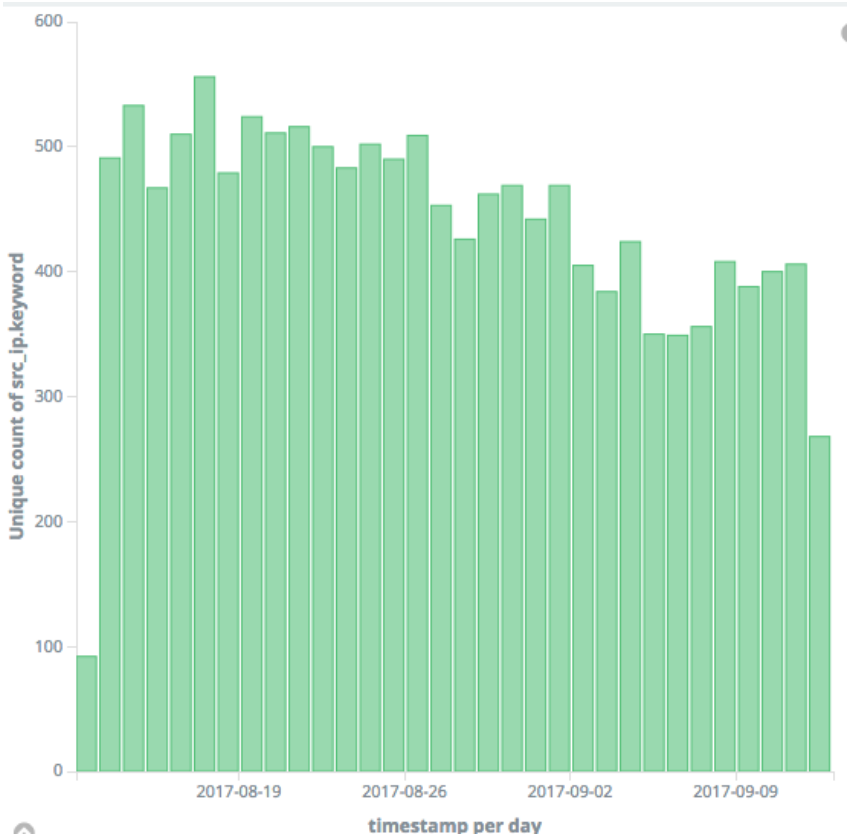
Ao longo do período que o honeypot ficou ativo, o mesmo recebeu contato de 10.748 endereços IP únicos, como pode-se observar na Figura 14. Considerando que alguns provedores fornecem serviço DHCP, o número de hosts provavelmente foi menor, visto que o mesmo host pode ter entrado em contato mais de uma vez utilizando endereços distintos.

Figura 14: Números de IPs únicos que interagiram com o Honeypot



A seguir, no Gráfico 1, pode-se visualizar essas interações de IPs únicos distribuídas ao longo do período ativo.

Gráfico 1: Interação de IPs únicos, distribuídos em 1 mês.



A seguir, na Tabela 8, temos os 8 usuários e senhas mais utilizados pelos bots, afim de realizar a autenticação por força bruta.

Tabela 8: Usuários mais utilizados.

username.keyword: Descending	Count
root	117,420
admin	17,174
guest	1,166
support	1,067
ubnt	1,011
user	697
service	639
usuario	556

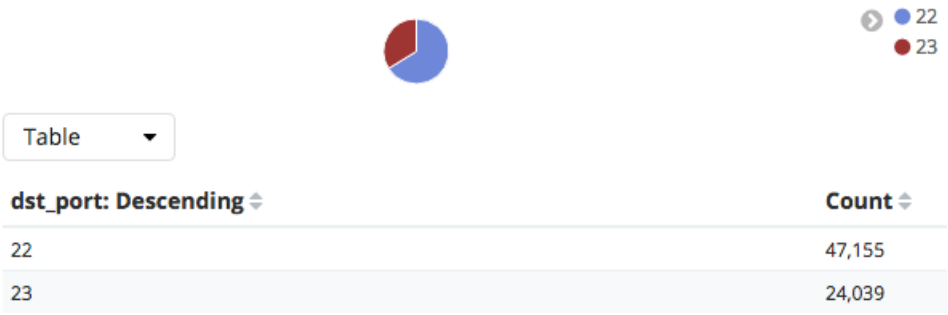
Pôde-se também, obter a relação das senhas mais utilizadas para acesso ao longo do período. Essa relação é apresentada na Tabela 9.

Tabela 9: Senhas mais utilizadas

admin	6,831
xc3511	3,752
password	2,691
1234	2,400
12345	2,013
	1,965
aquario	1,780
123456	1,553

Como mencionado anteriormente, foram configuradas apenas duas portas abertas para simular os serviços Telnet e SSH. No Gráfico 2, vemos a distribuição da utilização desses protocolos para o acesso ao Honeypot por parte dos bots.

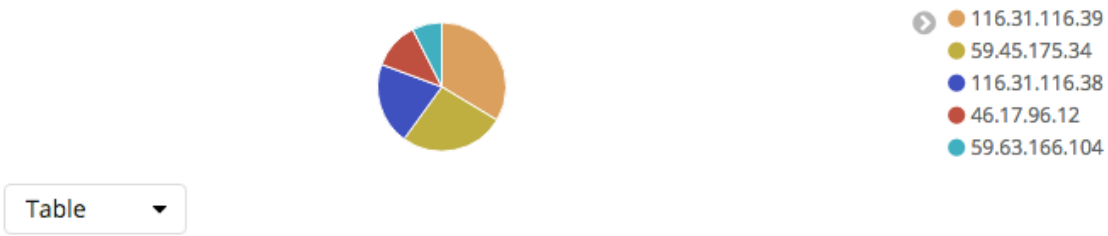
Gráfico 2: Protocolos de acesso mais utilizados



dst_port: Descending	Count
22	47,155
23	24,039

No Gráfico 3, temos os endereços de IP que mais realizaram conexões com o Honeypot. A primeira intuição, é considerar que esses IPs são das centrais de Controle e Comando (C2), entretanto é difícil de tirar esse tipo de conclusão pois essas centrais fazem uso de técnicas de mascaramento de IP, como *proxies.*

Gráfico 3: Endereços IP que realizaram mais interações



src_ip.keyword: Descending	Count
116.31.116.39	9,481
59.45.175.34	7,435
116.31.116.38	5,828
46.17.96.12	3,341
59.63.166.104	2,113

Na Figura 12, temos o número mínimo de IPs únicos infectados pelo malware da família MIRAI. Essa conclusão foi possível realizando um filtro em todas as interações de IPs únicos em que algum momento o comando “/bin/busybox MIRAI” foi realizado.

Figura 12 : Números de IPs únicos infectados pelo malware MIRAI

O termo “número mínimo” foi utilizado pois outros comandos podem indicar infecção de outro tipo de variante da família de malwares MIRAI, como por exemplo o comando “/bin/busybox ECCHI”[28], que no caso do experimento ocorreu a partir de 25 IPs únicos.

Durante o período de produção do Honeypot foram coletados um total de 75 tipos de arquivos que foram baixados de 160 endereços IP diferentes pelos malwares, como mostra a Figura 15. Esses arquivos foram baixados por meio de protocolos como FTP, TFTP, wget, entre outros e são compostos basicamente de arquivos binários ou scripts.

Figura 15: Números de IPs únicos dos quais foram baixados arquivos

Na Tabela 10, pode-se observar os endereços IP dos quais mais arquivos maliciosos foram baixados.

Tabela 10: IPs que forneceram mais arquivos únicos

src_ip.keyword: Descending	Unique count of outfile.keyword
222.186.51.138	11
195.22.127.83	9
178.62.236.49	6
118.104.18.47	5
118.99.228.13	5

Vista a alta capacidade de filtragem de dados da plataforma Kibana, existem inúmeras maneiras de retirar informações relevantes a respeito do experimento. Nesta análise foram apenas apresentados os dados julgados mais relevantes para a compreensão dos principais eventos que ocorreram durante o período de exposição do Honeypot na internet.

6.3. Discussão

O malware Mirai e suas variantes vem cada vez mais trazendo ao foco os desafios tecnológicos e regulatórios quando se diz a respeito de dispositivos IoT gerenciados por consumidores. Em contraste com sistemas desktop ou mobile, onde um pequeno número de fabricantes conscientes com a segurança controla as partes mais sensíveis do software (Windows, iOS, Android), dispositivos IoT são muito mais heterogêneos e, em uma perspectiva de segurança, geralmente negligenciados.

Estas botnets demonstraram que até ataques de força bruta pouco sofisticados podem comprometer milhares de dispositivos conectados na internet. Para mitigar essas ameaças, a segurança em IoT deve deixar de se basear no modelo que utiliza portas SSH e Telnet abertas por padrão e adotar boas práticas de segurança mais robustas. Dispositivos devem considerar configurações de rede padrão que limitam o acesso remoto aos mesmos apenas à rede interna ou a endereços específicos.

Em uma perspectiva regulatória, certificações devem ajudar os consumidores à fazerem escolhas mais seguras ao mesmo tempo que pressionar os fabricantes a produzirem produtos mais seguros.

7. Conclusão

As botnets voltadas à IoT, compostas primariamente de dispositivos com Linux embarcado, estão provocando grande impacto na internet ao longo dos últimos dois anos, quando acabaram comprometendo vários alvos de alto perfil com os maiores ataques DDoS registrados até então. Neste trabalho, é fornecida uma análise compreensiva das características de ataque e propagação da botnet Mirai e suas variantes e quais são seus dispositivos alvo. Após serem feitas todas as análises, percebemos que por mais que os dispositivos IoT apresentem muitos desafios de segurança únicos, a evolução da botnet Mirai foi baseada principalmente na ausência das boas práticas de segurança, o que acabou resultando em um ambiente frágil e passível de abuso. Enquanto o domínio IoT continua a se expandir e evoluir, espera-se que estes ataques ocorridos nos últimos tempos sirvam como um alerta para que os *stakeholders* industriais, acadêmicos e governamentais se preocupem mais com a segurança e a privacidade deste novo mundo onde todas as coisas estarão conectadas.

Referências

[1] IEEE. Towards a definition of the Internet of Things. https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf

[2] Garry Sims. IoT Security – What you need to know. https://www.androidauthority.com/iot-security-gary-explains-727977/

[3] Gartner. Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017. https://www.gartner.com/newsroom/id/3598917

[4] Janina Bartje. The top 10 IoT application areas based on real IoT projects.

https://iot-analytics.com/top-10-iot-project-application-areas-q3-2016/

[5] Ahmed Banafa. Three Major Challenges Facing IoT. https://iot.ieee.org/newsletter/march-2017/three-major-challenges-facing-iot

[6] Ahmed Banafa. IoT implementation and Challenges. https://www.linkedin.com/pulse/iot-implementation-challenges-ahmed-banafa?trk=mp-author-card

[7] Robbie Mitchell. 5 challenges of the Internet of Things. https://blog.apnic.net/2015/10/20/5-challenges-of-the-internet-of-things/

[8] Ben Dickson. 4 Major Technical Challenges Facing IoT Developers. https://www.sitepoint.com/4-major-technical-challenges-facing-iot-developers/

[9] Myrsini Athinaiou. Why has healthcare become such a target for cyber-attackers? http://theconversation.com/why-has-healthcare-become-such-a-target-for-cyber-attackers-80656

[10] James Graham, Richard Howard, Ryan Olson. Cyber Security Essentials, 2011.

[11] Hamid R. Nemati. Information security and ethics: concepts, methodologies, tools, and applications: concepts, methodologies, tools, and applications, 2007.

[12] VanDyke Software Inc. Press Release. https://www.vandyke.com/aboutus/news/pressreleases/company/it_survey09-09a.html

[13] Contel Bradford. 5 Common Encryption Algorithms and the Unbreakables of the Future. https://www.storagecraft.com/blog/5-common-encryption-algorithms/

[14] Arjen K. Lenstra, James P. Hughes. Ron was wrong, Whit is right. https://eprint.iacr.org/2012/064.pdf

[15] Thorsten Kleinjung, Kazumaro Aoki. Factorization of a 768-bit RSA modulus. https://eprint.iacr.org/2010/006.pdf

[16] Rapid7. Common Types of Cybersecurity Attacks. https://www.rapid7.com/fundamentals/types-of-attacks/

[17] Techopedia. Brute Force Attack. https://www.techopedia.com/definition/18091/brute-force-attack

[18] Margaret Rouse. Denial-of-service attack. http://searchsecurity.techtarget.com/definition/denial-of-service

[19] Incapsula. DNS Amplification. https://www.incapsula.com/ddos/attack-glossary/dns-amplification.html

[20] Lyndon Sutherland. Know Your Enemy: Understanding the Motivation Behind Cyberattacks. https://securityintelligence.com/know-your-enemy-understanding-the-motivation-behind-cyberattacks/

[21] Michael Hill. What Motivates Cyber-criminals and Who Are They Targeting? https://www.infosecurity-magazine.com/blogs/what-motivates-cybercriminals/

[22] Verizon. 2017 Data Breach Investigations Report. https://www.ictsecuritymagazine.com/wp-content/uploads/2017-Data-Breach-Investigations-Report.pdf

[23] AlienVault. Common Types of Malware. https://www.alienvault.com/blogs/security-essentials/comm3on-types-of-malware-2016-update

[24] Corero. Mirai Botnet DDoS Attack Type. https://www.corero.com/resources/ddos-attack-types/mirai-botnet-ddos-attack.html

[25] The Guardian. DDoS attack that disrupted internet was largest of its kind in history, experts say. https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet

[26] Brian Krebs. Who is Anna-Senpai, the Mirai Worm Author? https://krebsonsecurity.com/2017/01/who-is-anna-senpai-the-mirai-worm-author/

[27] USAT. Update: Sierra Wireless Technical Bulletin: Mirai Malware. http://usatcorp.com/update-sierra-wireless-technical-bulletin-mirai-malware/

[28] Ivo Van der Elzen, Jeroen Van Heugten. Techniques for detecting compromised IoT devices, 2017.

[29] Manos Antonakakis, Tim April, et al. Understanding The Mirai Botnet, 2017.

[30] Zakir Durumeric, Michael Bailey, J. Alex Halderman. An Internet-Wide View of Internet-Wide Scanning, 2014.

[31] Kirill Shipulin. Practical ways to misuse a router. http://blog.ptsecurity.com/2017/06/practical-ways-to-misuse-router.html

[32] Margaret Rouse. Telnet. http://searchnetworking.techtarget.com/definition/Telnet

[33] NetMarketshare. Desktop Operating System Market Share. https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0

[34] Microsoft. Support for Windows XP ended. https://www.microsoft.com/en-us/windowsforbusiness/end-of-xp-support

[35] IETF. Cisco Systems NetFlow Services Export Version 9, RFC 3954. https://www.ietf.org/rfc/rfc3954.txt

[36] Iyatiti Mokube, Michele Adams. Honeypots: Concepts, Approaches, and Challenges. http://ai2-s2-pdfs.s3.amazonaws.com/88d5/f45a8fc8c6947cc02e8fb8b9cc3a53227ee3.pdf

[37] Michel Oosterhof. Cowrie SSH/Telnet Honeypot. https://github.com/micheloosterhof/cowrie

[38] Universidade Carnegie Mellon. Vulnerability Note VU#355151. https://www.kb.cert.org/vuls/id/355151

[39] Nmap. Nmap-services. https://svn.nmap.org/nmap/nmap-services.

[40] Sehque. HOW TO CONFIGURE AND DEPLOY A COWRIE SSH HONEYPOT FOR BEGINNERS. https://sehque.wordpress.com/2015/07/23/how-to-configure-and-deploy-a-cowrie-ssh-honeypot-for-beginners/

[41] Elastic. LOGSTASH. https://www.elastic.co/products/logstash

[42] Execute Malware Blog. Honeypot Visualization Revisited. http://executemalware.com/?p=355#comment-634

[43] Executemalware. Honeypot-Visualizations. https://github.com/executemalware/Honeypot-Visualizations/blob/master/create-cowrie-mappings.sh

[44] Executemalware. Honeypot-Visualizations. https://github.com/executemalware/Honeypot-Visualizations/blob/master/bulk_index.sh

[45] David M. Geary. Core Javaserver Faces, 3ª Ed. 2004.

[46] Stack Exchange. What exactly (and precisely) is hash. https://cs.stackexchange.com/questions/55471/what-exactly-and-precisely-is-hash/55472

[47] Dan Goodin. Leak of >1,700 valid passwords could make the IoT mess much worse. https://arstechnica.com/information-technology/2017/08/leak-of-1700-valid-passwords-could-make-the-iot-mess-much-worse/