HonIoT: Arquitetura de Honeynet com Controle de Propagação de Malwares para Dispositivos de IoT

Douglas B. de Godoy[1], Jaime F. Souza[1], Emerson R. A. Barea[2] e César A. C. Marcondes[3]

[1]Departamento de Computação (DC) - Universidade Federal de São Carlos (UFSCar)
[2]Departamento de Informática (DInf) - Instituto Federal do Tocantins (IFTO)
[3]Divisão de Ciência da Computação (IEC) - Instituto Tecnológico de Aeronáutica (ITA)

{douglas.godoy,jaime.souza}@ufscar.br, emerson.barea@ifto.edu.br, cmarcondes@ita.br

Resumo. *É indicada a utilização de honeypots e honeynets para o estudo aprofundado do comportamento dos malwares para dispositivos de IoT, porém, alguns cuidados devem ser tomados garantindo que esses ambientes sejam capazes de detalhar corretamente todo ciclo de infecção dos malwares e não se tornem um ponto de origem de ataques às outras redes. Nessa linha, este trabalho apresenta o HonIoT, uma honeynet que suporta a inclusão de honeypots reais, emuladas ou virtualizadas, criando uma internet emulada que corresponde à parte atacada da rede, possibilitando o monitoramento completo de todo ciclo de infecção e propagação do malware. Testes preliminares apontaram boa capacidade de escalabilidade do ambiente e correta execução do fluxo de ações da honeynet.*

Abstract. *It is indicated the use of honeypots and honeynets for IoT devices malware behavior study in-depth, but some attention should be taken to ensure these environments are able to accurately detail every malware infection cycle and do not become a attacker to other networks. In this line, this work presents HonIoT, a honeynet that supports inclusion of real, emulated or virtualized honeypots, creating an emulated internet corresponding to the attacked part of the network, allowing the complete monitoring of the whole cycle of malware infection and propagation. Preliminary tests pointed to good environment scalability and correct execution of honeynet's stock flow.*

1. Introdução

Nos últimos tempos foi perceptível o aumento da utilização de dispositivos de Internet das Coisas (*Internet of Things* - IoT) em áreas até então pouco exploradas. O desenvolvimento dessa estrutura tem sido bastante impulsionada principalmente pela evolução tecnológica de aparelhos de uso cotidiano, como telefones e TVs, porém, a IoT é mais ampla e atualmente está presente em diversas áreas, como indústrias, medicina, serviços, automobilismo, entre outras [Simpson et al. 2017] e [GARTNER 2017].

A ampliação da utilização de dispositivos de IoT veio acompanhada da necessidade de conectá-los às redes, privadas ou até mesmo pela internet, possibilitando sua utilização e gerenciamento remotos. Quanto a segurança, muitos desses dispositivos apresentam vulnerabilidades, que vão desde ausência de mecanismos de autenticação habilitados por padrão, passando por falhas em códigos, até problemas críticos de hardware [Dowling et al. 2017],

que vêm sendo exploradas por métodos diversos [SPAMHAUS 2018] e com efeitos cada vez mais abrangentes e críticos [Bhunia and Gurusamy 2017] [Marzano et al. 2018].

O cenário apresentado aponta claramente que o aumento da utilização de dispositivos de IoT e o risco associado a esse tipo de ambiente são problemas relevantes de pesquisa. Uma abordagem bastante utilizada para coleta e análise de malwares são os *honeypots* e *honeynets* ([Sochor and Zuzcak 2014], [Koniaris et al. 2014] e [Pathan 2014]), porém, esses mesmos ambientes podem não ser capazes de capturar corretamente todo ciclo de infecção de um *malware* ou até passar a ser um ponto de origem de ataque às outras redes caso não tomados alguns cuidados [Agnaou et al. 2016] e [Cabaj et al. 2017].

Baseado no exposto, esse trabalho propõe uma arquitetura de *honeynet* modular com controle de propagação de *malwares* para dispositivos de IoT. Entre os destaques e aspectos mais relevantes, essa *honeynet (i)* suporta a emulação de arquiteturas próprias de dispositivos de IoT com o uso de tecnologias de virtualização leve, lhe conferindo a possibilidade de ser livremente replicada em uma infraestrutura de computação genérica; *(ii)* emula a parte atacada da internet, portanto, não propagando ataques às outras redes enquanto fornece um ambiente propício para a completa infecção por parte do *malware*; e *(iii)* suporta o uso de equipamentos reais, emulados ou softwares de simulação como *honeypot*.

O restante desse trabalho está organizado da seguinte forma: a seção 2 apresenta os trabalhos relacionados ao tema principal da pesquisa. A seção 3 apresenta os requisitos a serem atendidos e os desafios impostos à proposta apresentada. Na seção 4 são apresentadas a arquitetura da *honeynet* e o detalhamento do protótipo desenvolvido. A seção 5 apresenta e comenta os resultados da avaliação do protótipo; e a seção 6 conclui e apresenta sugestões de trabalhos futuros.

2. Trabalhos Relacionados

São diversas as implementações de *honeypots* para dispositivos de IoT. Abaixo segue uma lista, não exautiva, de soluções divididas em algumas áreas de atuação.

O IoTPot [Pa et al. 2015] é um *honeypot* de baixa interatividade que emula *banners* reproduzidos artificialmente de diversos equipamentos de IoT utilizado para análise de tentativas de ataques por Telnet. Durante a fase de testes, os autores observaram mais de 76 mil tentativas de download de binários originados a partir de mais de 16 mil endereços IPs distintos. Adicionalmente, os autores fizeram o download de 43 exemplares de *malwares* para análise e constataram que eles estavam relacionados a 11 arquiteturas diferentes de CPU. Apesar dos resultados obtidos, os *banners* publicados pelo IoTPot são gerados manualmente, dificultando sua evolução e facilitando a identificação do *honeypot* pelo *malware*, já que não utiliza sistemas reais. Da mesma forma, ele não é capaz de detalhar o ciclo de infecção do *malware* apresentando as chamadas de rede e de sistemas nos alvos atacados.

Outra implementação existente é o HoneyThing [Erdem et al. 2018]. *Honeypot* de baixa interatividade que emula o protocolo TR-069, utilizado para gerenciamento de equipamentos de clientes (*Customer-Premises equipament* - CPE), como modens e roteadores domésticos. Também possui o servidor Web RomPager 4.07[1] simulando algumas

[1]https://www.allegrosoft.com/embedded-web-server-ae

vulnerabilidades associadas ao protocolo TR-069 encontrados em modens. Apesar de ser um *honeypot* específico para CPEs, por ser de baixa interatividade, o HoneyThing apenas gera estatísticas de tentativas de acessos sem detalhar o processo de infecção do dispositivo, como também não fornece mecanismos de contenção em caso de comprometimento do *honeypot*.

Mais uma implementação, agora com foco em redes de sensores sem fio (*Wireless Sensor Networks* - WSN), é o ZigBee *honeypot* [Dowling et al. 2017], que reproduz um dispositivo com suporte a rede ZigBee para identificar ataques ao SSH. Os autores identificaram nos testes realizados que o tipo de ataque com maior ocorrência estava relacionado a DDoS. Assim como os outros *honeypots* apresentados, o ZigBee *honeypot* é de baixa interatividade, possuindo as mesmas limitações das soluções anteriores.

Por fim, outro exemplo de implementação é o HIoTPOT [Gandhi et al. 2018], que consiste em um *honeypot* de alta interatividade baseado no Raspberry PI conectado a duas redes, uma de dispositivos reais e a outra de dispositivos falsos. O HIoTPOT possui uma base de dados onde estão registrados todos usuários válidos e autenticados na rede. Quando um usuário não autenticado tenta acessar outro equipamento da rede, o *honeypot* automaticamente direciona seu tráfego para os dispositivos da rede falsa, mantendo segura a rede real enquanto possibilita a análise do processo de infecção do *malware*. Apesar de eficiente, o HIoTPOT tem custo elevado com a duplicação da estrutura da rede física, enquanto seu método de identificação de ataque é simples e passível de falha.

Como citado anteriormente, apesar de não exaustiva, a lista de *honeypots* apresentados indica que as soluções normalmente são baseados em serviços e protocolos específicos, reduzindo muito seu escopo e funcionalidades. Da mesma forma, não possuem mecanismos de contenção contra propagação de ataques caso o ambiente seja comprometido. Por outro lado, o exemplo de *honeypot* de alta interatividade apresentado é baseado em ambiente físico, elevando seu custo, além de possuir um mecanismo de contenção contra propagação de *malwares* muito simples e ineficiente.

3. Requisitos e Desafios

Ao propor o desenvolvimento de um *honeypot* de alta interatividade, ou *honeynet*, com controle de propagação de *malwares* para dispositivos de IoT emulados, alguns novos questionamentos surgem e levantam consigo requisitos ainda não contemplados.

Um *honeypot* de alta interatividade, assim como apresentado na Figura 1, tem como requisitos principais que as aplicações e serviços oferecidos por ele sejam reais, que possam ser comprometidos pelo *malware* e que o atacante tenha acesso irrestrito ao sistema operacional (S.O.) do dispositivo. Esses requisitos garantem que o *malware* não identifique que está em um *honeypot*, infectando completamente o ambiente e viabilizando a análise de seu funcionamento. Quanto aos desafios, são caracterizados pelo alto custo, dificuldade e complexidade de instalação e manutenção do ambiente, além do alto risco de comprometimento da estrutura e consequente propagação do *malware* utilizando o próprio *honeypot* como um ponto de ataque às outras redes [Hoepers et al. 2007].

Já os *honeypots* de baixa interatividade são reconhecidos por seu baixo custo de instalação, manutenção e risco de comprometimento, porém, como apresentado na seção 2, os resultados obtidos com sua utilização são limitados principalmente pelas deficiências

Características	*Honeypot* de baixa interatividade	*Honeypot* de alta interativ./*Honeynet*
Instalação	**fácil**	mais difícil
Manutenção	**fácil**	trabalhosa
Risco de comprometimento	**baixo**	alto
Obtenção de informações	limitada	**extensiva**
Necessidade de mecanismos de contenção	**não**	sim
Atacante tem acesso ao S.O. real	não (em teoria)	**sim**
Aplicações e serviços oferecidos	**emulados e reais**	
Atacante pode comprometer o *honeypot*	**não (em teoria)**	sim

Figura 1. Características desejáveis ao *honeynet* com controle de propgação de *malwares* para dispositivos de IoT. Fonte: adaptado [Hoepers et al. 2007].

de implementação ou ausência de recursos existentes nos sistemas reais, possibilitando serem descobertos pelos *malwares* mais avançados.

Baseado nas características apresentadas, identificamos que a proposta mais adequada para este trabalho é unir as características mais vantajosas dos dois tipos de *honeypots* (de baixa e alta interatividade), conforme destacado na Figura 1. Fazendo uso de sistemas reais de dispositivos de IoT em arquitetura correspondente emulada com o auxílio de softwares específicos, reduzindo o custo, simplificando a instalação e manutenção do ambiente; e emulando a parte atacada da rede com todas suas máquinas e serviços, impedindo a propagação do *malware* às outras redes enquanto cria um ambiente completo para infecção do *malware*, proporcionando meios para captura de informações para seu estudo.

4. Materiais e Métodos

Nesta seção são detalhados os elementos formadores da arquitetura da *honeynet* (seção 4.1), seguido da apresentação da estrutura física utilizada no desenvolvimento do protótipo HonIoT (seção 4.2) e do detalhamento do fluxo de ações e códigos do controlador do protótipo (seção 4.3).

4.1. Arquitetura

Para atingir os objetivos propostos, atendendo aos requisitos apresentados e solucionando os desafios impostos, foi definida uma arquitetura modularizada para *honeynet* de dispositivos de IoT. Essa arquitetura é responsável por suportar a análise de todo ciclo de infecção e propagação de *malwares* para IoT, com estrutura de *honeypot* de alta interatividade com controle de propagação, expondo totalmente um sistema de IoT aos ataques provindos da internet, enquanto monitora todo acesso e fluxo de dados num ambiente controlado, criando a representação de parte da internet quando o *honeypot* passa a ser um ponto de origem e disseminação de ataques.

Para facilitar a compreensão de sua estrutura e o funcionamento de cada componente, a arquitetura foi dividida em quatro módulos principais, conforme apresentado na Figura 2. Abaixo estão descritas as funcionalidades dos elementos pertencentes a cada módulo e a relação entre eles:

Dispositivos de IoT: corresponde aos dispositivos destinados a sofrerem ataques na *honeynet*. Suporta a utilização de *honeypots* de baixa interatividade, com reprodução de serviços ou sistemas de IoT virtualizados em *hardware* genérico ou com a utilização de plataformas e sistemas construídos especialmente para capturar informações de acessos maliciosos; ou alta interatividade, suportando a utilização de sistemas e dispositivos de IoT emulados em plataformas específicas ou reais conectados à *honeynet*.

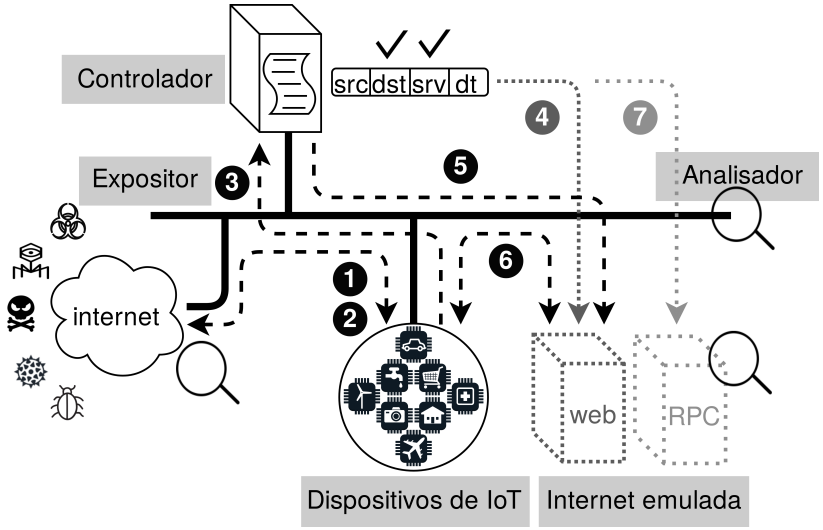


Figura 2. HonIoT: arquitetura da *honeynet* com controle de propagação de *malwares* para dispositivos IoT.

Outra função desse módulo é garantir que o *malware* não identifique que está em um *honeypot*, possibilitando a infecção completa do ambiente. Alguns *malwares* utilizam-se de artifícios variados para identificar se o ambiente é um *honeypot*, limintando ou até mesmo anulando seu comportamento quando a identificação é positiva. Dessa forma, é aconselhável utilizar plataformas e sistemas que reproduzam fielmente a original, com o auxílio de emulação da arquitetura correta e configurações adicionais necessárias, como a implementação de *banners* que correspondam aos dos serviços reais e a ocultação de parâmetros e informações do sistema base de emulação do ambiente.

Expositor: esse módulo corresponde a toda infraestrutura de rede necessária para exposição completa dos dispositivos de IoT (*honeypot*). Sua função é possibilitar que *malwares* provindos da internet acessem o *honeypot* para infectá-lo completamente. Essa fase corresponde aos Fluxos 1 e 2 apresentados na Figura 2 (fluxos identificados pelas bolas pretas com os números 1 e 2). No Fluxo 1, o *honeypot* é acessado livremente por qualquer *malware* com origem na internet, sem restrição alguma, possibilitando sua exposição completa e, consequentemente, aumentando a probabilidade de captura de informações de *malwares* diversos. Já o Fluxo 2 permite que o *honeypot* acesse os servidores de comando e controle (C&C) dos *malwares* distribuídos na internet, e vice-versa, a fim de completar o ciclo de infecção do *malware*.

Toda a exposição representada pelos Fluxos 1 e 2 na arquitetura são gerenciadas pelo $Controlador$ da *honeynet*, módulo detalhado em seguida.

Controlador: esse módulo é considerado o mais importante da *honeynet*, pois, é ele que garante a inteligência da rede atuando como um gerenciador de tráfego pró-

ativo, durante a infecção do *honeypot*; e também reativo ao comportamento do *malware*. Inicialmente sua função é permitir o fluxo de dados entre os *malwares* provindos da internet e o *honeypot*, como apresentado anteriormente no módulo $Expositor$, onde toda comunicação é baseada em regras de controle de tráfego e definição de rotas que permitem o encaminhamento dos dados entre os dispositivos envolvidos na comunicação.

A partir da infecção completa do *honeypot*, todo tráfego originado por ele deve ser interceptado pelo $Expositor$ e direcionado ao $Controlador$ (Fluxo 3), pois, a partir desse momento, os novos fluxos podem estar relacionados aos procedimentos de propagação e disseminação do *malware*, que devem ser evitados para não tornar a *honeynet* num ponto de origem de ataques. Ao mesmo tempo, a análise dessas chamadas de rede são material fundamental para o estudo e entendimento do comportamento e funcionamento do *malware*.

Com o tráfego interceptado, o $Controlador$ identifica e extrai os parâmetros de endereçamento de rede e do serviço solicitados das máquinas envolvidas na comunicação - endereço IP de destino, protocolo e portas de serviço utilizadas. Com posse dessas informações, o $Controlador$ tem a função de criar as máquinas da $Internet\ emulada$ sob demanda, criando automaticamente máquinas clone configuradas com os parâmetros de rede e de serviços previamente extraídos pelo $Controlador$ (Fluxo 4).

A partir da criação da máquina clone, o controlador cria as regras de roteamento e direcionamento de tráfego, instalando-as no $Expositor$ para então encaminhar os dados da solicitação de comunicação diretamente à máquina clone na $Internet\ emulada$ (Fluxo 5). A partir desse momento, toda nova comunicação entre o *honeypot* e a máquina clone é feita diretamente através do ambiente $Expositor$ (Fluxo 6), dessa forma, nenhum tráfego originado no *honeypot* correspondente a uma tentativa de propagação do *malware*, ou qualquer outra ação dele, será propagada para outras redes. Esse procedimento impede que a *honeynet* seja um ponto de ataque e viabiliza o monitoramento do tráfego de rede originado pelo *malware*, possibilitando sua análise detalhada. A cada solicitação de comunicação iniciada pelo *honeypot* a um novo destino, protocolo ou serviço, esse processo é repetido (Fluxo 7).

Analisador: esse módulo é responsável por capturar dados que servirão como base de informações para a análise de todo processo de infecção, comprometimento e propagação do *malware*. As ferramentas pertencentes a esse módulo estão distribuídas em quatro áreas específicas da arquitetura:

- Internet: aqui o analisador monitora todas as tentativas de acesso ao *honeypot*, incluindo tempo, volumetria de tráfego, origem dos ataques e quais mecanismos são utilizados.
- Dispositivos de IoT: o monitoramento interno nos dispositivos de IoT provê informações que podem ser utilizadas para análise das ações do *malware* nesses dispositivos, através de monitoramento da utilização de memória, processos criados, chamadas de sistema ou outros parâmetros que se fizerem necessários. Vale ressaltar que o monitoramento nesse ponto dependente do tipo de *honeypot* utilizado, pois, devem ser respeitadas restrições de acesso a determinados parâmetros por questão de licenças ou limitações técnicas em equipamentos ou softwares proprietários.
- Rede da Internet emulada: monitora todo tráfego de rede passante entre o *honeypot*

e todos elementos da $Internet\ emulada$, assim como realizado no link de Internet. Monitora o fluxo de dados correspondente à propagação do *malware* a partir do comprometimento do *honeypot*.

- Máquinas da Internet emulada: aqui o analisador monitora as ações executadas pelo *malware* internamente nas máquinas da $Internet\ emulada$, como apresentado no Dispositivo de IoT. O objetivo é gerar informações que sirvam como base para análise do comportamento do malware nas máquinas atacadas durante sua propagação.

Com a utilização de todas informações obtidas no módulo $Analisador$, é possível realizar um estudo aprofundado de todo ciclo de infecção de um malware.

4.2. HonIoT

Para validar a arquitetura da *honeynet* com controle de propagação de *malwares* para dispositivos de IoT proposta neste trabalho e apresentada na subseção 4.1, foi desenvolvido o protótipo HonIoT (*Honeynet* IoT), cuja finalidade é simular parte do processo de infecção sofrido por um *honeypot* de alta interatividade a partir de *malwares* originados na internet, possibilitando o estudo dos mecanismos de infecção e propagação.

O HonIoT é um protótipo que une um conjunto de aplicações de rede e emulação leve pré-existentes, a elementos desenvolvidos especialmente para esse fim. De modo geral, baseia-se na plataforma de emulação Qemu versão 2.11.1 como base para o $Dispositivo\ de\ IoT$ (*honeypot*); o OVS versão 2.9.2 e Wireshark versão 2.6.5 na estrutura do $Expositor$; o LXD versão 3.0.3 como base para as máquinas da $Internet\ emulada$; e o Ryu versão 4.24 como controlador para SDN do OVS.

A estrutura física do HonIoT é composta por dois servidores físicos, cada um contendo um processador de 3.50GHz e 8 núcleos, 16GB de memória RAM e múltiplas interfaces de rede de 1Gbps cada. Além de um computador com processador de 3GHz e 4 núcleos, 2 GB de memória RAM e duas interfaces de rede de 1Gbps cada. As máquinas foram conectadas fisicamente conforme apresentado na Figura 3. Em todas as máquinas foi instalado o S.O. Ubuntu 16.04.4 LTS Server sem serviços adicionais, propiciando um ambiente livre de interferências indesejáveis.

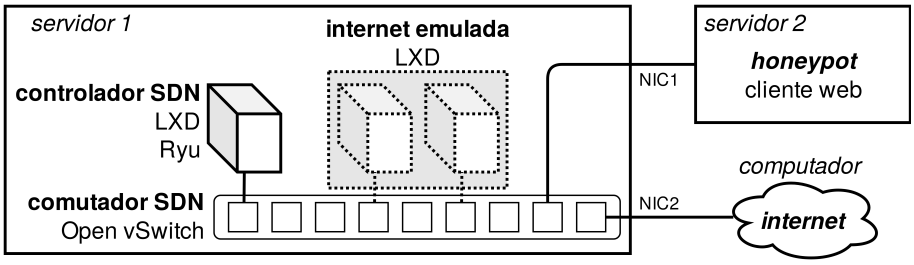


Figura 3. Topologia para o protótipo *HonIoT*.

Conforme apresentado na Figura 3, o $servidor\ 1$ hospeda todo ambiente de controle e emulação responsáveis por reproduzir a porção da rede atacada. Tal ambiente é composto por $controlador\ SDN$ (*container* LXD com Ryu instalado), $comutador\ SDN$ (OVS) e $internet\ emulada$ (*containers* LXD criados sob demanda). O $servidor\ 2$ possui o S.O. Linux instalado contendo softwares de servidor e cliente Web instalados sobre

uma arquitetura ARM emulada (Qemu) simulando um *honeypot*. Já o computador que representa a $internet$ possui apenas o S.O. Linux instalado diretamente em seu hardware.

O $servidor\ 1$ possui uma interface de rede conecta diretamente ao $servidor\ 2$. Essa interface também está conectada logicamente ao $comutador\ SDN$, possibilitando a comunicação entre o ambiente virtualizado da *honeynet* e o *honeypot* emulado. O $servidor\ 1$ também possui uma interface de rede conectada fisicamente ao computador $internet$ e logicamente ao $comutador\ SDN$, possibilitando a interligação de todos elementos virtuais e físicos no protótipo.

4.3. Fluxo de Ações

Quanto ao fluxo das ações executadas para validação do protótipo, inicialmente foram geradas requisições de rede com origem na $internet$ e destinadas ao servidor Web do $honeypot$, simulando a infecção por um *malware*. Essas chamadas são encaminhadas sem restrição no $comutador\ SDN$, ao mesmo tempo em que o tráfego de rede é monitorado ao passar pelo comutador. Essa etapa da validação do protótipo corresponde ao Fluxo 1 da Figura 2.

Para isso, assim que percebe o fluxo de dados, o $comutadorSDN$ intercepta e direciona os pacotes ao $controladorSDN$, que inspeciona o parâmetro correspondente ao tipo de protocolo da camada 4, identificando se o segmento é TCP ($bits == 2$ indica que o segmento possui a flag SYN utilizada). A partir dai, o $controladorSDN$ extrai os endereços IP de origem e destino do datagrama identificando se a origem está na $internet$ (src_port) e o destino é o *honeypot* ($HONEYPOT\_IP$). Caso a comunicação atenda essa premissa, o $controladorSDN$ cria a regra que permite a comunicação entre a máquina da internet e o *honeypot* ($match1$) e a instala no $comutador\ SDN$ (add_flow) (Figura 4).

```
###
# comunicação internet <-> honeypot
if dst_ip == self.HONEYPOT_IP and \
((protocol == 'TCP' and bits == 2) or protocol != 'TCP'):
    ...
    match1 = msg.datapath.ofproto_parser.OFPMatch(eth_type=2048, \
        ip_proto=protocolN, udp_src=src_port, udp_dst=dst_port, \
        ipv4_src=src_ip, ipv4_dst=dst_ip)
    ...
    action1 = [msg.datapath.ofproto_parser.OFPActionOutput(out_port1)]
    ...
    self.add_flow(msg.datapath, 1, match1, action1)
```

Figura 4. Código Python do controlador que permite máquinas da internet acessarem livremente o *honeypot*.

Seguindo esse processo, quando o *honeypot* inicia a comunicação com um servidor de C&C, o tráfego também deve ser permitido, conforme descrito na seção 4.1, pois essa comunicação é considerada requisito para que o *malware* continue o processo de infecção. Durante esse processo, o $controladorSDN$ verifica se o endereço IP de destino contido no datagrama interceptado (dst_ip) existe na *white_list* (ip_list) recuperada em [Emerging Threats 2017]. Essa filtragem por endereços da *white_list* permite que o

controlador identifique quais endereços IP são de servidores de C&C, permitindo que o $controladorSDN$ crie regras liberando o tráfego para esses servidores (Figura 5).

```
###
# comunicação honeypot <-> C&C
def is_attack(self, dst_ip):
    if self.command_ip_list.in_list(dst_ip):
        return False
    else:
        return True
```

Figura 5. Código Python do controlador que permite a comunicação do *honeypot* com servidores de C&C.

Todo tráfego gerado no *honeypot* com destino a qualquer endereço IP não constante na *white list* pode representar uma tentativa de propagação do *malware* que deve ser impedida pelo HonIoT. Para isso, o $comutador\ SDN$ intercepta o fluxo de dados e o encaminha ao $controlador\ SDN$ que, por sua vez, extrai as informações de endereçamento das máquinas, protocolos e serviços contidos no fluxo, e acessa o gerenciador de *containers* localizado no *servidor 1* através de uma conexão SSH, solicitando a criação de um novo *container* LXD a partir de modelos pré-estabelecidos com base no serviço desejado ($DEFAULT\_CONTAINER\_TEMPLATE$) (Figura 6). A partir desse momento, o gerenciador de *containers* configura o *container* recém criado com os parâmetros de rede identificados anteriormente.

```
###
# honeypot atacando outras máquinas
    def create_container(self, msg, src_ethernet, ip_version, src_ip, \
                         dst_ip, container_name):
        template = self.DEFAULT_CONTAINER_TEMPLATE
        ...
        ssh = SSH()
        cmd = 'lxc launch %s %s' % (template, container_name)
        ssh.exec_cmd(cmd)
        ...
        cmd = 'lxc exec %s -- cat /sys/class/net/eth0/address' \
            % (container_name)
        new_container_mac = ssh.exec_cmd(cmd).strip()
```

Figura 6. Exemplo de código do controlador que cria *containers* na Internet emulada.

Feito isso, o $controladorSDN$ recupera a informação de endereço de camada 2 (*Media Access Control* - MAC) atribuído automaticamente ao *container* recém gerado pelo serviço LXD ($new\_container\_mac$), cria e instala uma regra ($action1$ e $action2$ - Figura 7) no $comutador\ SDN$ permitindo o tráfego entre o *honeypot* e o *container* com base nesse endereço, possibilitando assim o encaminhamento de tráfego diretamente entre essas máquinas.

```
action1 = [msg.datapath.ofproto_parser.\
          OFPActionSetField(eth_dst=new_container_mac), \
          msg.datapath.ofproto_parser.OFPActionOutput(out_port1)]
action2 = [msg.datapath.ofproto_parser.OFPActionOutput(out_port2)]
```

Figura 7. Código Python do controlador que cria e instala a regra para comunicação com o *container* da internet emulada.

Todo esse processo é repetido sempre que o *honeypot* gera fluxos de dados para novos destinos, protocolos ou serviços, impedindo que o *honeypot* propague o *malware*, além de viabilizar a análise de todo esse processo com o monitoramento dos fluxos de dados na rede e as ações executadas pelo *malware* nas máquinas da $internet\ emulada$.

5. Testes de Validação e Discussão dos Resultados

Para avaliar o completo funcionamento do protótipo HonIoT, dividimos os testes em 4 etapas principais que estão detalhadas a seguir:

Experimentos de Invasão ao honeypot: esse teste analisou o correto funcionamento do processo em que máquinas distintas da $internet$ acessam serviços disponíveis no *honeypot*, simulando uma tentativa de iniciar a infecção. Para isso, foram geradas 100 requisições ao serviço Web partindo de IPs gerados aleatoriamente na máquina da $internet$ destinadas ao *honeypot*. Nesse teste, 100% das requisições foram devidamente tratadas pelo $comutador\ SDN$ e $controlador\ SDN$, seguindo o procedimento apresentado na seção 4.3, Figura 4, com todas solicitações de acesso devidamente encaminhadas ao *honeypot*, conforme demonstraram os rastros de execução. Essa validação foi considerada satisfatória.

Experimentos de Comunicação *Honeypot* e Servidores de C&C: em seguida, um segundo experimento foi o de acompanhar o correto funcionamento da comunicação livre do *honeypot* com servidores de C&C a qualquer momento. Esse acesso deve ser permitido quando o endereço IP de destino do datagrama emitido pelo *honeypot* constar na *white list* de servidores de C&C. Ao invés de testar com um *malware* imprevisível, foram feitos testes de conexão iniciadas a partir do *servidor 2* (*honeypot*), para isso, foram gerados requisições Web originadas no *honeypot* e destinadas aos endereços IP dos servidores constantes na *white list* de servidores de C&C. Ao final do teste, observou-se que todas requisições foram devidamente tratadas pelo $comutador\ SDN$ e $controlador\ SDN$, e encaminhadas para a saída de $internet$ corretamente. Os rastros de execução demonstraram que esse teste de validação foi considerado satisfatório.

Experimentos de Criação das máquinas na Internet emulada: esse terceiro experimento teve o propósito de verificar se o $controlador\ SDN$ solicita corretamente a criação de um ou mais *containers* ao *Servidor 1* após receber datagramas IP com parâmetros de rede "não constantes" em regras já instaladas no $comutador\ SDN$; e se os *containers* são criados e inicializados corretamente para encaminhamento do tráfego partindo do *honeypot* em tempo hábil.

Nessa etapa, o interesse era em dois cenários possíveis. O primeiro verificou o comportamento do ambiente, na forma dessa implementação, quando solicitadas chamadas de grupos de *containers* de forma sequencial, onde um *container* posterior somente é solicitado quando o anterior já está totalmente funcional. A ideia era estudar o comportamento

de um *malware* que interage e analisa o ambiente em que está antes de tomar a próxima decisão.

O segundo cenário analisou o comportamento do ambiente quando grupos de *containers* são solicitados em rajadas pelo *honeypot*, simulando uma situação de ocorrência de *scan* de rede. O *scan* de rede pode ser intenso e requerer a criação de dezenas ou centenas de *containers* em cada execução. Esse cenário permitiu verificar e analisar a escalabilidade e estabilidade do protótipo nesse tipo de situação.

Os resultados demonstrados na Figura 8 apresentam a utilização de memória (em MB) agrupada por conjuntos de *containers* criados sequencialmente, ou seja, no cenário 1. Os dados mostram que a utilização de memória está diretamente relacionada ao número de *containers* criados, porém, seu crescimento não é linearmente perfeito, partindo de aproximados 70 MB para criação de 1 *container*, passando por aproximados 463 MB de memória consumida para um grupo de 50 *containers*, chegando a até 925 MB de memória consumida para grupos de 100 *containers*, sempre com 100% de sucesso na criação dos *containers*. Os resultados indicam uma sobrecarga de memória ao serem criados mais containers. Não foi investigado a fundo o porque desse comportamento, mas pode ser em função de replicação de dados do sistema operacional.

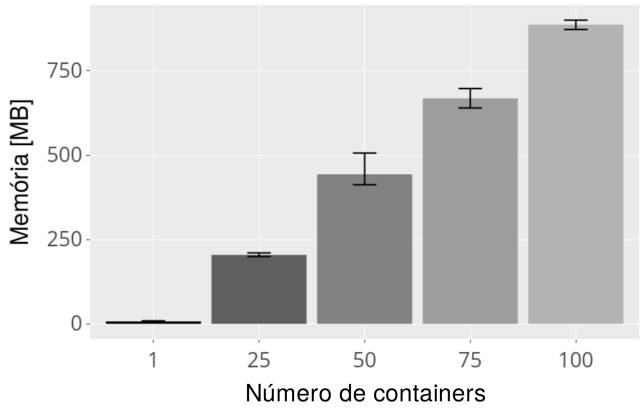


Figura 8. Memória (em MB) consumida para criação dos *containers* correspondentes às máquinas da *Internet emulada*.

Apesar desse comportamento de sobrecarga adicional por *container* adicional, o protótipo comprova que a utilização de *containers* LXD na $Internet\ emulada$ é promissora quando considerado o consumo total de memória versus o número de máquinas emuladas. Ou seja, 100 *containers* em uma máquina servidora pode ser considerado baixo uso de recursos.

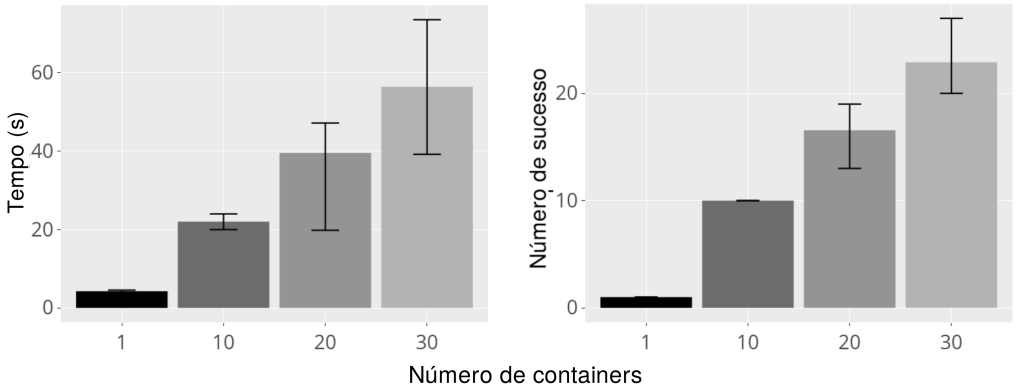
Outro resultado importante identificado é o comportamento padrão quanto ao tempo de criação dos *containers*, que foi de 4.1 segundos em média por *container*. Esse tempo tem o revés de ser elevado para tratar o encaminhamento de um pacote, podendo gerar desconfiança em um *malware* mais sofisticado sobre a possibilidade de ele estar em um *honeypot*.

Com relação ao cenário 2, onde é analisado o comportamento do protótipo frente a situações de rajadas de pacotes, foram verificados o tempo de criação para os grupos

de *containers* e a quantidade de sucesso nesse procedimento. Como não existe tempo mínimo entre as chegadas de pacotes, o $controlador\ SDN$ recebe múltiplas requisições simultâneas por meio de eventos em *threads*, e repassa para o gerenciador de *containers* as solicitações de criação no melhor esforço da tecnologia utilizada no protótipo.

Os dados da Figura 9a apresentam que, apesar das solicitações terem sido geradas em rajadas a partir do *honeypot*, onde verificou-se um tempo médio de apenas 0,6 segundos para o encaminhamento de 100% das solicitações, houve um aumento considerável de tempo para criação dos *containers* na $Internet\ emulada$ do *servidor 1*, chegando a aproximadamente 53 segundos em média para grupos de 30 *containers* solicitados. Esse comportamento ocorre devido os recursos de criação de *containers* serem compartilhados e a contenção ser grande.

Além dos tempos de instanciação serem elevados, outro dado importante é a grande variação do tempo obtido entre os testes, que demonstrou instabilidade do ambiente do protótipo nesse caso.



(a) Tempo de criação dos *containers*

(b) Número de *containers* criados com sucesso

Figura 9. Tempo (segundos) consumidos para criação de grupos de *containers* solicitados em rajadas (a) e quantidade de *containers* criados com sucesso por grupos de *containers* solicitados em rajadas (b).

Para analisar essa questão de instabilidade e a incapacidade de dar vazão rápida para um número grande de instanciação de *containers*, uma observação importante é o aumento expressivo da taxa de insucesso com o aumento da quantidade de *containers* solicitados nas rajadas. Conforme apresentado na Figura 9b, a taxa de insucesso é de aproximadamente 10% a partir da solicitação de criação de 20 *containers*, aumentando consideravelmente para até 20% de falhas para grupos de 30 *containers* solicitados.

Análises preliminares dos resultados apontam que as tecnologias utilizadas no protótipo possuem deficiência na forma como o controlador conecta no *servidor 1* para solicitar a criação dos *containers*, utilizando conexões SSH através do Python Paramiko. Os rastros da execução apontaram erros no estabelecimento de conexões entre essas duas máquinas quando as solicitações chegam em rajadas. Uma possível alternativa é a utilização de sockets Unix ou outro tipo de comunicação de rede menos complexa que uma conexão SSH. Uma outra opção é a pré-instanciação de centenas de *containers* no inicio da criação do ambiente experimental, cabendo ao controlador somente a tarefa de

configurá-los com os parâmetros de rede extraídos previamente durante a interceptação dos fluxos de dados.

Comunicação com as máquinas da Internet emulada: o último experimento de validação verificou o funcionamento do encaminhamento do tráfego partindo do *honeypot* e destinado aos *containers* da $Internet\ emulada$, ao invés de direcioná-lo para saída pelo link de internet real da *honeynet*. Nessa etapa, basicamente após a criação dos *containers* na $Internet\ emulada$, foi observado a criação as regras de encaminhamento no $comutador\ SDN$, atualizando a tabela de fluxos e, a partir dai, todo tráfego partindo do *honeynet* foi encaminhando corretamente, sem atrasos relevantes ou quebra de conexão TCP para as partes correspondentes. Os rastros de execução do controlador e dos *containers* envolvidos demonstraram que esse teste de validação foi considerado satisfatório.

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou o HonIoT, uma *honeynet* para dispositivos de IoT que suporta *honeypots* baseados em equipamentos físicos e reais, simulados ou emulados. Garante a exposição completa do *honeypot* aos *malwares* disponíveis na internet, enquanto garante o controle de propagação do *malware* criando uma internet emulada e protegida, além de propiciar um ambiente completo de monitoramento das ações do *malware* através do monitoramento de suas ações no *honeypot*, na rede e nas máquinas emuladas.

Testes preliminares demonstraram bom nível de escalabilidade da *honeynet* quanto ao tempo de resposta às solicitações de criação de máquinas na rede emulada e quantidade suportada, bem como o correto funcionamento do fluxo de ações do protótipo proposto.

Como trabalhos futuros, considera-se relevante investigar maneiras mais sofisticadas para identificar quando o *malware* que infectou o *honeypot* pretende acessar servidores de C&C ou realizar um ataque, testar outras técnicas de conexão entre o $controlador\ SDN$ e a máquina hospedeira da $Internet\ emulada$ para melhorar a taxa de sucesso da criação de *containers* em situação de rajadas de fluxos de rede. Outra necessidade é testar o HonIoT em ambiente real para monitorar seu comportamento em produção.

Agradecimentos

Este trabalho teve apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001 e do Instituto Federal do Tocantins (IFTO).

Referências

Agnaou, A., Kalam, A. A. E., Ouahman, A. A., and Montfort, M. D. (2016). Reduce positive and negative falses from attacks collected from the deployment of distributed honeypot network. *CoRR*, abs/1611.03252.

Bhunia, S. S. and Gurusamy, M. (2017). Dynamic attack detection and mitigation in iot using sdn. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–6.

Cabaj, K., Gawkowski, P., Grochowski, K., Nowikowski, A., and Żórawski, P. (2017). The impact of malware evolution on the analysis methods and infrastructure. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 549–553.

Dowling, S., Schukat, M., and Melvin, H. (2017). Data-centric framework for adaptive smart city honeynets. In *2017 Smart City Symposium Prague (SCSP)*, pages 1–7.

Dowling, S., Schukat, M., and Melvin, H. (2017). A zigbee honeypot to assess iot cyberattack behaviour. In *2017 28th Irish Signals and Systems Conference (ISSC)*, pages 1–6.

Emerging Threats (2017). Emerging threats botnet command and control drop rules. `http://rules.emergingthreats.net/open/suricata/rules/botcc.rules`. Acessado em: 18/12/2017.

Erdem, O., Pektas, A., and Kara, M. (2018). Honeything: A new honeypot design for cpe devices. *KSII Transactions on Internet and Information Systems*, 12:4512–4526.

Gandhi, U. D., Kumar, P. M., Varatharajan, R., Manogaran, G., Sundarasekar, R., and Kadu, S. (2018). Hiotpot: Surveillance on iot devices against recent threats. *Wireless Personal Communications*, 103(2):1179–1194.

GARTNER (2017). 6.4 billion connected "things"will be in use in 2016. `https://www.gartner.com/newsroom/id/3165317`. Acessado em: 13/04/2017.

Hoepers, C., Steding-Jessen, K., and Chaves, M. H. P. C. (2007). Honeypots e honeynets: Definições e aplicações. `https://www.cert.br/docs/whitepapers/honeypots-honeynets/`. Acessado em: 18/12/2018.

Koniaris, I., Papadimitriou, G., Nicopolitidis, P., and Obaidat, M. (2014). Honeypots deployment for the analysis and visualization of malware activity and malicious connections. In *2014 IEEE International Conference on Communications (ICC)*, pages 1819–1824.

Marzano, A., Alexander, D., Fazzion, E., Fonseca, O., Cunha, Italo e Hoepers, C., Steding-Jessen, K., e Chaves, M. H. P. C., Guedes, D., and Meira Jr., W. (2018). Monitoramento e caracterização de botnets bashlite em dispositivos iot. *Simpósio Brasileiro de Redes de Computadores (SBRC)*, 36.

Pa, Y. M. P., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T., and Rossow, C. (2015). Iotpot: Analysing the rise of iot compromises. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, Washington, D.C. USENIX Association.

Pathan, A.-S. K. (2014). *The State of the Art in Intrusion Prevention and Detection*. Auerbach Publications, Boston, MA, USA.

Simpson, A. K., Roesner, F., and Kohno, T. (2017). Securing vulnerable home iot devices with an in-hub security manager. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*, pages 551–556. IEEE.

Sochor, T. and Zuzcak, M. (2014). Study of internet threats and attack methods using honeypots and honeynets. In Kwiecień, A., Gaj, P., and Stera, P., editors, *Computer Networks*, pages 118–127, Cham. Springer International Publishing.

SPAMHAUS (2018). Spamhaus botnet threat report 2017. `https://www.spamhaus.org/news/article/772/spamhaus-botnet-threat-report-2017`. Acessado em: 05/03/2018.