UNIVERSIDADE DO VALE DO TAQUARI - UNIVATES
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE ENGENHARIA DA COMPUTAÇÃO

ANÁLISE DE UM SERVIDOR DE BAIXO CUSTO PARA HOSPEDAR UM HONEYPOT UTILIZANDO RASPBERRY PI

Jonathan Felten Azevedo

Lajeado, novembro de 2019

Jonathan Felten Azevedo

ANÁLISE DE UM SERVIDOR DE BAIXO CUSTO PARA HOSPEDAR UM HONEYPOT UTILIZANDO RASPBERRY PI

Monografia desenvolvida na disciplina de Trabalho de Conclusão de Curso II, do curso de Engenharia da Computação, da Universidade do Vale do Taquari – Univates, como requisito para obtenção do título de Bacharel em Engenharia da Computação.

Orientador: Prof. Dr. Marcelo de Gomensoro Malheiros

Lajeado, novembro de 2019

RESUMO

Possuir segurança e confidencialidade de dados em dispositivos conectados à Internet é um grande desafio. Conforme a tecnologia vai evoluindo, novas formas de ataques cibernéticos surgem e, assim, geram um enorme perigo aos usuários e às corporações. Da mesma maneira, novas ferramentas para prevenção são criadas, como os *honeypots.* Estes ainda são pouco conhecidos, mas possuem um papel extremamente importante na identificação e no combate a ameaças. Este trabalho tem o objetivo de analisar um servidor de baixo custo para atuar como um *honeypot*. Para isso, um dispositivo Raspberry Pi foi configurado de várias maneiras, simulando uma máquina vulnerável com a finalidade de distrair o atacante e atuar como detector de ataques. Por meio de alguns cenários e ferramentas, foi possível realizar testes e analisar a viabilidade de utilizar o Raspberry Pi para essa função, em diferentes situações, de acordo com seu desempenho e baixo custo. Foi evidenciado que o Raspberry Pi consegue hospedar um *honeypot,* mas sua efetividade irá depender da eficiência do software utilizado e também do tamanho e complexidade da rede em que ficará esse dispositivo.

Palavras chave: Segurança de Redes. Vulnerabilidade. Honeypot. Raspberry Pi.

ABSTRACT

Having data security and confidentiality on devices connected to the Internet is a big challenge. As technology evolves, new forms of cyberattacks emerge and thus pose a huge danger to users and corporations. In the same time, new prevention tools are created, such as honeypots. Those are still poorly known, but play an extremely important role in identifying and fighting threats. This work aims to analyze a low-cost server to act as a honeypot. To do this, a Raspberry Pi device has been configured in many ways, simulating a vulnerable machine for the purpose of distracting the attacker and acting as an attack detector. Through some scenarios and tools, it was possible to perform tests and analyze the feasibility of using Raspberry Pi for this function, in different situations, according to its performance and low cost. It was evidenced that Raspberry Pi can host a honeypot, but its effectiveness will depend on the efficiency of the software used and also on the size and complexity of the network where this device will be placed.

Key words: Network Security. Vulnerability. Honeypot. Raspberry Pi.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ARM	Acorn RISC Machine
CERT.br	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
CIA	Confidentiality, Integrity and Availability
DDoS	Distributed Denial of Service
DMZ	Demilitarized Zone
DoS	Denial of Service
FTP	File Transfer Protocol
GHz	Gigahertz
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion Detection System
IIS	Internet Information Services
IoT	Internet of Things
IP	Internet Protocol
LAN	Local Area Network
MBR	Master Boot Record
NMAP	Network Mapper
OLT	Optical Line Terminal
ONU	Optical Network Unit
P2P	Peer-to-peer

PSAD	Port Scan Attack Detector
SD	Secure Digital
SQL	Structured Query Language
SSH	Secure Shell
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
VNC	Virtual Network Computing
VPN	Virtual Private Network
WAN	Wide Area Network
WOR	Windows On Raspberry
XMPP	Extensible Messaging and Presence Protocol

SUMÁRIO

1 INTRODUÇÃO

Atualmente, a maior parte das pessoas e empresas utiliza ao menos um dispositivo conectado à Internet para realizar tarefas ou buscar e enviar informações. Devido a esses fatores, a demanda por dispositivos e serviços que utilizem e trafeguem nessa rede vem aumentando exponencialmente nos últimos anos.

Com esse crescimento, os usuários desses recursos tendem a ficar vulneráveis a ataques, expondo informações básicas ou confidenciais para qualquer usuário que esteja na rede e que tenha um conhecimento mais avançado. Consequentemente, muitas empresas acabam gastando muitos recursos e tempo para combater esses possíveis atacantes. Ainda assim, podem continuar possuindo sistemas propensos a falhas, não dispondo de recursos para detectar intrusos e possíveis ataques.

Segundo Assunção (2002), a vulnerabilidade nos sistemas informatizados não é nenhuma novidade e, mesmo com o crescimento dessa tecnologia, muitas pessoas ainda possuem um grande medo do computador, devido à inexperiência com o mundo informatizado. Por consequência disso, os usuários leigos são as maiores vítimas dos ataques. Dessa maneira, criminosos conseguem roubar informações e podem, até mesmo, realizar ataques a bancos à distância, por meio de seus computadores.

Pelo simples motivo de estar conectado à Internet, um sistema está sujeito a ataques. O administrador deve estar ciente que estará correndo riscos, evitando tornar-se mais uma vítima, estando disposto a combatê-los com ferramentas

adequadas para a proteção. Muitos usuários e empresas não têm preocupação em relação a possíveis ataques, por acharem que não possuem nada de valor que atraia um invasor, ou por acharem que um ataque não pode acarretar grandes danos ou perdas em seus sistemas (OLIVEIRA, 2008).

Por esses crimes cibernéticos tornarem-se tão comuns hoje em dia, faz-se necessário o estudo mais aprofundado deste contexto. Este trabalho apresenta o que é a Segurança da Informação, abordando os riscos e ameaças que dispositivos informatizados correm atualmente. Também são apresentados os tipos de ataque e formas de prevenção dos mesmos, focando na implementação de um dispositivo de segurança do tipo *honeypot*, operando a partir de um sistema embarcado de baixo custo. Diante disso, faz-se a análise da sua viabilidade e eficiência como um sistema de detecção e aprendizado de intrusões e ataques.

1.1 Justificativa

Devido à grande quantidade de ataques cibernéticos que corporações e usuários sofrem. Buscou-se uma forma de conhecer melhor os ataques e seus praticantes. Dessa forma, com o uso de um *honeypot* é possível conhecer melhor o inimigo, analisando como ele age e assim mitigando os ataques e melhorando a segurança de uma rede.

1.2 Objetivos

O objetivo geral da pesquisa foi reduzir as incertezas e riscos de segurança em uma rede através do uso de um *honeypot*. Assim, foram analisadas suas vantagens e desvantagens, e como este pode ser efetivo para ajudar na segurança de uma rede através de seus registros.

Foram designados como objetivos secundários:

- Construir um referencial teórico sobre Segurança da Informação e sobre o tipo de ferramenta *honeypot;*

- Estudar trabalhos relacionados a Segurança de Redes, Segurança da Informação e sobre ataques cibernéticos;
- Buscar um *honeypot* que seja de fácil instalação e configuração para o administrador do sistema;
- Configurar uma solução de hardware e software modular capaz de atuar como servidor *honeypot*;
- Testar as funcionalidades de um *honeypot* dentro de um cenário empresarial;
- Analisar a viabilidade de um *honeypot* baseado em um sistema embarcado.

1.3 Estrutura do trabalho

O trabalho está estruturado em seis capítulos.

O primeiro capítulo faz uma breve introdução ao trabalho, discorrendo sobre os problemas dos ataques cibernéticos, integrando também a justificativa e objetivos do trabalho.

O segundo capítulo refere-se ao referencial teórico, o qual tem o intuito de explanar sobre Segurança de Redes, incluindo riscos, tipos de ataque e prevenções, servindo para fundamentar os objetivos propostos.

O terceiro capítulo detalha a metodologia utilizada na elaboração deste estudo, trazendo os métodos de pesquisa, seu objetivo e os procedimentos técnicos utilizados.

O quarto capítulo do estudo apresenta toda a parte de instalações e configurações dos sistemas operacionais e *honeypots*, além dos cenários e topologias utilizadas para os testes realizados neste trabalho.

O quinto capítulo contempla os experimentos realizados e os resultados obtidos, juntamente com sua análise e discussão.

O sexto capítulo aborda as conclusões observadas após a realização dos testes, além de trazer sugestões para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo é apresentada a revisão bibliográfica, com abordagem dos conteúdos que fundamentam o tema escolhido para o trabalho. Primeiramente, apresenta-se o que é Segurança da Informação e os conceitos relacionados, partindo para definições de ataques e prevenção dos mesmos. Então, são apresentados os conceitos e aplicações de um *honeypot* e de uma *honeynet,* além de descrever o Raspberry Pi como plataforma utilizada para um servidor de baixo custo.

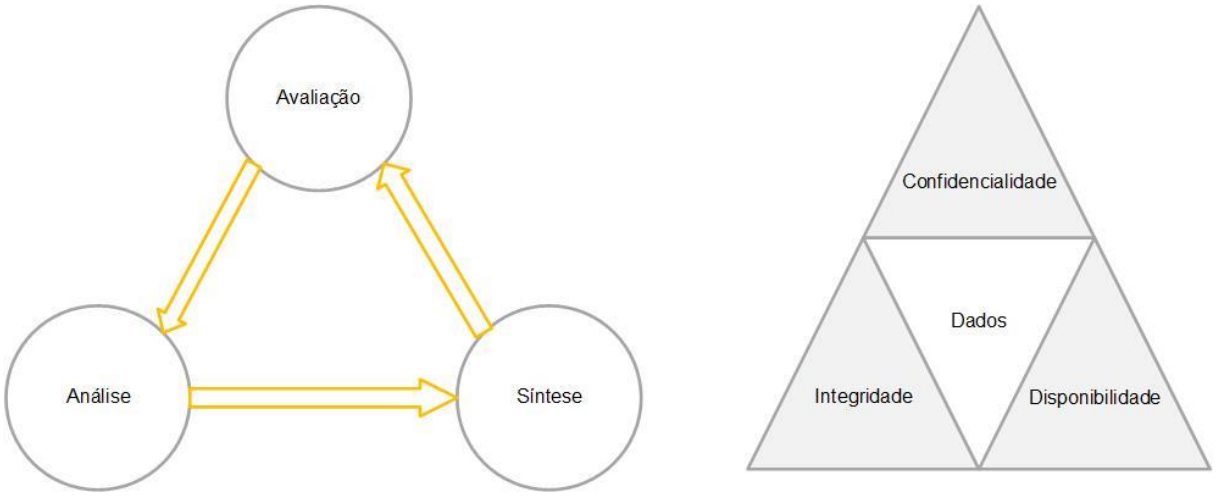
2.1 Segurança da Informação

A Segurança da Informação é um processo para proteger os dados de pessoas externas e internas à empresa, evitando que sejam utilizados para uso acidental quanto intencional (MORAES, 2010). Wadlow (2001) complementa que o processo da segurança, caso não seja aplicado, ou seja interrompido em algum momento, acarretará uma rede mais vulnerável, conforme surgem novas tecnologias e, por consequência disso, novas formas de ataques.

Wadlow (2001) diz que o processo para a Segurança da Informação é semelhante à tríade grega, como ilustra a Figura 1, no qual deve ser analisada a falha levando em consideração todos os aspectos da mesma. Em seguida, deve-se sintetizar uma solução para a possível falha a partir dessa análise e, por fim, a solução deverá ser avaliada e concluir quais aspectos atenderam, ou não, suas expectativas. Ou seja, o usuário aprende com seus erros. Após isso, a tríade se

repete. Stallings (2015) acrescenta que, visto pela Segurança da Informação, conseguimos analisar essa tríade de uma forma diferente, composta por três conceitos, sendo eles: confidencialidade, integridade e disponibilidade. Estes são chamados coletivamente de Tríade CIA, da sigla para Confidentiality, Integrity and Availability, em inglês.

Figura 1 - Exemplo da Tríade Grega e Tríade CIA.



Fonte: Adaptada pelo autor, com base em Wadlow (2001) e Stallings (2015).

A integridade garante que os dados estejam autênticos, ou seja, que não tiveram nenhuma alteração desde seu envio (origem) até o seu recebimento (destino), garantindo total consistência desses dados. A confidencialidade garante que os dados não possam ser acessados por usuários sem permissão, garantindo assim, seu total sigilo. Já a disponibilidade garante que os dados e o sistema estejam sempre ativos e disponíveis para suprir a necessidade de acesso ao usuário, quando for utilizá-lo (MORAES, 2010).

A tríade CIA está bem estabelecida, porém, existem mais dois conceitos muito importantes para a segurança: autenticidade e responsabilização. A autenticidade garante que tudo que trafega na rede seja genuíno, ou seja, garante que os usuários sejam realmente eles mesmos, impedindo que algum usuário tente transparecer como outro. Já a responsabilização garante que cada ação na rede tenha alguém como responsável, o que pode acontecer por meio de um histórico de alterações, mais conhecido como *log*. Sendo assim, o administrador do sistema fica ciente de cada alteração feita pelos usuários da rede. Caso algum usuário faça

alterações indevidas, ele será descoberto e terá que se responsabilizar pelo ato. Assim, o *log* pode ajudar a identificar quem realizou as intrusões e alterações não permitidas, e a partir dele, o administrador da rede pode visualizar as alterações indevidas realizadas pelo usuário e, assim, revertê-las para manter o sistema íntegro e autêntico (STALLINGS, 2015).

Uma empresa deve ter uma política de segurança, e é ela que irá estabelecer o seu comportamento perante a segurança de seus dados. Essa política determinará quais situações são e não são permitidas, e as ações que o administrador da rede tomará de acordo com cada situação. Com isso, torna-se possível que cada empresa mensure o nível de proteção de seus dados (TEIXEIRA JR. *et al*., 1999).

A segurança dos dados é uma batalha entre o atacante e o administrador do sistema. Nela, o *hacker* se beneficia com apenas uma falha do sistema e o administrador tem o trabalho de prever e analisar todas as falhas possíveis para que o *hacker* não consiga acesso à rede e a uma boa segurança (STALLINGS, 2015). Nakamura e Geus (2007) acrescentam que, caso o administrador se descuide de uma única falha, os esforços feitos para evitar as demais serão esforços em vão, correndo riscos de ataques ainda maiores, e transmitindo uma falsa sensação de segurança para o sistema e toda a rede.

Não é possível comprar um software ou dispositivo que torne sua rede ou sistema completamente seguro. Quando falamos de segurança, estamos falando de uma viagem na qual o destino é a própria segurança, no entanto, essa viagem não tem fim, o que pode ser feito é administrar um nível aceitável de risco (WADLOW, 2001).

Segundo Nakamura e Geus (2007), a vulnerabilidade da rede ou do sistema ocorre devido a falhas de projeto ou na implementação de um serviço, protocolo ou software mal configurado. Mas, ainda que corrigida a falha, *bugs* sempre estarão presentes nessas aplicações.

Com a vinda das novas tecnologias, é iminente uma nova quantidade de falhas e vulnerabilidades, e sucessivamente, novos ataques para essas novas falhas também virão (NAKAMURA; GEUS, 2007). Moraes (2010) complementa que não existe rede 100% segura, ou seja, qualquer acesso que o usuário fizer com à

Internet, tanto para enviar quanto para receber informações, poderá acarretar riscos. Além disso, o usuário em si pode ser uma vulnerabilidade, pois pode publicar senhas de acesso ou até mesmo passar informações sigilosas a terceiros em troca de algum benefício ou vingança.

Nakamura e Geus (2007) ressaltam que os crimes digitais estão aumentando e são cada vez mais organizados, porque os criminosos possuem novas técnicas para cobrirem seus rastros e impedir que revelem sua real identidade. Esses criminosos também possuem certo amparo da legislação, pois ela ainda é imatura para tratar de crimes digitais. Sendo assim, não há uma ação mais rígida para mitigar esses acontecimentos.

2.2 Problemas na segurança: ameaças, riscos e ataques

Os maiores riscos para a segurança de uma organização surgem pelo uso inadequado da Internet, que pode ser causado por um acesso a um *site* desconhecido, *download* de software pirata, ou ainda *download* de músicas, filmes e jogos. Esse tráfego tipicamente corresponde a mais de 70% do uso da rede. O uso inadequado, além de gerar maiores riscos a ataques, também gera improdutividade do usuário e problemas de lentidão, devido ao *link* estar sobrecarregado com dados que não interessam à empresa (MORAES, 2010).

A ameaça é representada por um usuário qualquer ou uma organização que queira infringir contra a segurança de um sistema. Quando o usuário consegue levar vantagem sobre uma falha, o mesmo estará praticando um ataque (TEIXEIRA JR. *et al.*, 1999). Wadlow (2001) complementa que o usuário, mesmo sem qualquer intenção, pode comprometer toda a segurança de uma rede, ainda que ela seja bem estruturada e tenha uma ótima política de segurança.

As ameaças podem ser classificadas em alguns tipos: ameaças intencionais, ameaças relacionadas aos equipamentos, ameaças relativas a um evento natural e ameaças não intencionais (MORAES, 2010).

As ameaças intencionais são feitas por agentes internos e externos à rede, os quais podem ser espiões industriais que são contratados por outra organização para

roubar dados e causar algum tipo de dano. Essas ameaças também podem ser causadas por um criminoso profissional que tem o intuito de ter algum ganho financeiro com o ataque. Além disso, ainda temos os funcionários mal-intencionados, chamados de *insiders* (MORAES, 2010).

As ameaças relacionadas a equipamentos ocorrem devido a falhas de hardware e até mesmo de software, seja por motivo de *bugs* ou por algum defeito no equipamento (MORAES, 2010).

As ameaças relativas a um evento natural englobam todos os equipamentos e instalações físicas da corporação, pois, estão propensos a desastres naturais como uma queda de luz decorrente de tempestades, inundação ou até mesmo a incêndios (MORAES, 2010).

Por último, as ameaças não intencionais são erros cometidos pelos usuários e administradores do sistema, geralmente por desconhecimento. Isso pode acarretar um sistema ou *firewall* mal configurado, ou descumprimento das regras de segurança, aumentando assim o risco de ameaças (MORAES, 2010).

Os *insiders* são o maior risco que pode haver nas empresas. Cerca de 33% dos ataques que ocorrem são causados por eles, pois os *insiders* estão presentes no local. Devido a isto, podem realizar até mesmo um ataque físico, desligando um servidor ou *firewall,* e causando uma grande vulnerabilidade ou indisponibilidade no sistema (NAKAMURA; GEUS, 2007).

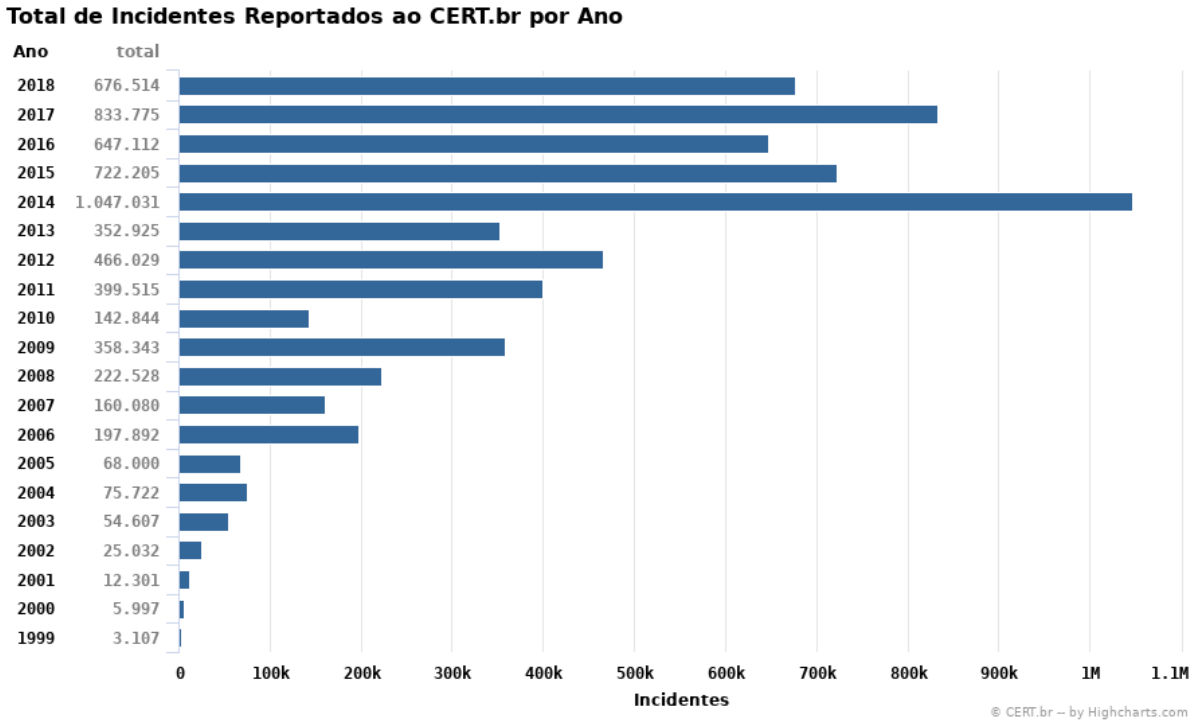
Os ataques contra a segurança das informações podem acarretar danos irreversíveis à grandes empresas, pois muitas vezes a empresa não consegue estimar o alto valor de suas informações, talvez pensando que apenas um *firewall* irá trazer segurança. Porém, ao sofrerem o ataque e perderem informações, ou terem as mesmas expostas, as perdas financeiras são incalculáveis, podendo até mesmo sujar a imagem dessa organização (MORAES, 2010).

Quando ouve-se falar em ataques, lembra-se dos *hackers*. Esse termo é normalmente utilizado de forma equivocada, pois *hacker*, no sentido tradicional, geralmente é uma pessoa que possui conhecimento em Informática e invade os sistemas como desafio, para melhorar e pôr em prática seus conhecimentos, sem

necessariamente ter a intenção de causar danos a empresas e usuários. Já o *cracker* é quem faz uma invasão com o intuito de causar danos, roubando informações e destruindo sistemas (NAKAMURA; GEUS, 2007). Moraes (2010) acrescenta que os Estados Unidos e o Canadá possuem mais de 200 milhões de usuários, e estima-se que 10% deles já tiveram interesse em realizar um ataque, ou tentaram utilizar alguma ferramenta para realizá-lo. Sendo assim, fica explícita a proporção de usuários que podem causar algum dano a outros usuários.

De acordo com o CERT.br (2019), o Brasil teve mais de um milhão de ataques registrados no ano de 2014, tendo uma queda de 31% em 2015. Mesmo assim, os dados mostram que a partir de 2014 o país possui uma média de mais de 750 mil incidentes por ano, como mostra a Figura 2.

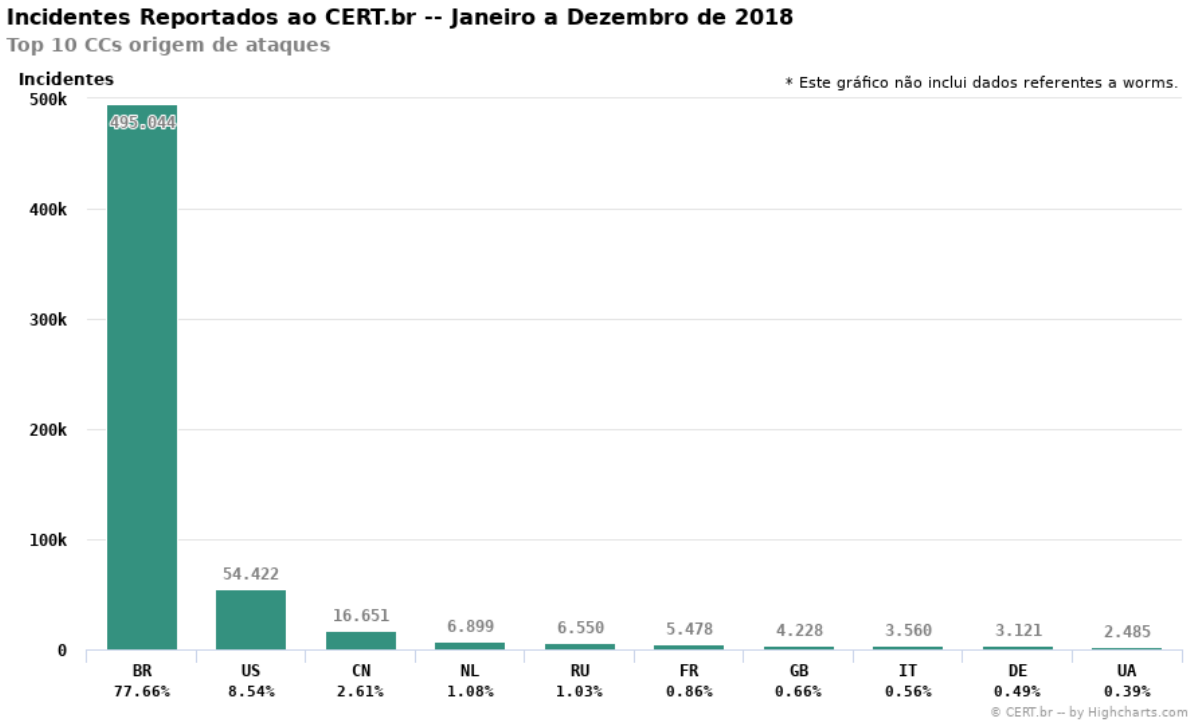
Figura 2 - Total de incidentes reportados ao instituto CERT.br de 1999 a 2018.



Fonte: CERT.br (2019).

CERT.br (2019) complementa que a maioria desses ataques registrados no Brasil têm origem no próprio país, que é seguido por Estados Unidos e China (juntos somam apenas 11.15% dos ataques). Ressalta-se que estes dados não incluem registros de ataque com *worms*. Estes dados são exibidos na Figura 3.

Figura 3 - Origem dos incidentes no ano de 2018.



Fonte: CERT.br (2019).

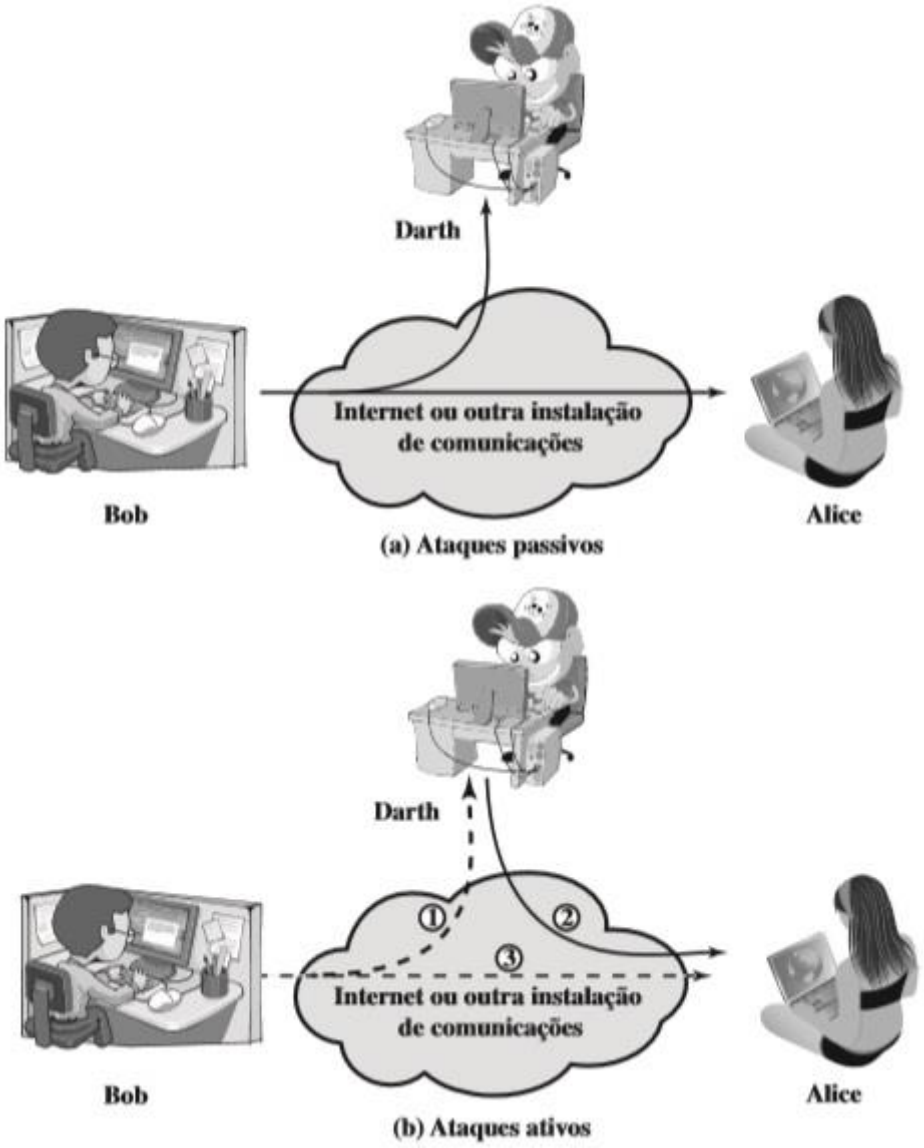
2.2.1 Tipos de ataques

Os ataques podem ser ativos e passivos. Os ataques ativos ocorrem quando os usuários realizam alguma alteração no sistema por meio de uma falha para serem favorecidos. Um exemplo desse ataque ativo é quando o usuário tenta acessar algum sistema com os dados de outra pessoa pelo método de tentativa ou erro. Entretanto, nos ataques passivos os atacantes apenas ficam à espreita e observando as informações trafegadas na rede. Para que esses ataques ocorram, geralmente o atacante instala alguma ferramenta maliciosa no computador da vítima, deste modo ele pode capturar as informações da vítima de forma passiva e realizar posteriormente um ataque ativo (TEIXEIRA JR. *et al*., 1999).

Stallings (2015) conta que é difícil detectar um ataque passivo, pois o atacante não altera os dados no meio da transmissão, mas fica apenas analisando-

os. Já no ativo, o atacante pode alterar a transmissão, enviando um novo fluxo de dados até o destinatário. Estes dois tipos de ataque são ilustrados na Figura 4.

Figura 4 - Diferença de ataque ativo e passivo.



Fonte: STALLINGS (2015).

Podemos classificar os ataques de acordo com a sua severidade, a qual é mensurada de acordo com o tempo empregado para recuperação e com o prejuízo que o ataque pode causar em um sistema. O grau de severidade pode variar de acordo com a organização, pois, para algumas empresas o grau de um ataque pode possuir baixa severidade, mas para outra esse ataque pode ser de alta severidade ou até mesmo crítico. Os de baixa severidade podem ser classificados como os ataques que não atrapalham o funcionamento da empresa, ou seja, mesmo que

tenham causado algum dano, ele será facilmente reparado por meio de um *backup* ou bloqueio de alguma porta de comunicação aberta (RAVANELLO *et al.*, 2004).

Os ataques de alta severidade implicam em um mal funcionamento do sistema da empresa, demandando custos e tempo de reparo aos danos causados pelo ataque. Esses danos podem ser a queda de servidores de arquivos, epidemias de vírus na rede ou impossibilidade de acesso à Internet, sendo necessária a reinstalação ou reconfiguração de sistemas e dispositivos de rede (RAVANELLO *et al.*, 2004).

Os ataques críticos ou incapacitantes, por sua vez, são aqueles que afetam de modo geral a empresa. Um exemplo que pode ser citado é a queda de um servidor que hospeda um banco de dados e o seu sistema para gestão de serviços, deixando a empresa incapacitada de realizar suas vendas, ou até mesmo ter os dados sigilosos de seus clientes expostos. Esses ataques são os que comprometem a visão da empresa e causam maiores prejuízos e tempo para reparo (RAVANELLO *et al*., 2004).

Os ataques podem ser subdivididos em automatizados e manuais. Os automáticos não dependem de ação humana para serem executados, dependendo apenas do uso de *scripts* ou de algum software já preparado. Entretanto, os manuais precisam de ação humana para sua execução, pois geralmente o ataque manual é feito por um usuário de maior conhecimento, que analisa com bastante cuidado a sua vítima antes de tomar qualquer atitude. Assim, ele pode cobrir os seus rastros e atacar diretamente a falha presente no sistema, a partir de uma ferramenta desenvolvida por ele mesmo. Essa ferramenta poderá ser disponibilizada na Internet para futuros ataques, tornando-os, desse modo, ataques automáticos (RAVANELLO *et al.*, 2004).

O vírus de computador é o ataque mais famoso e conhecido entre os usuários. Ele é um trecho de código de programação que geralmente é agregado a um software comum. Quando esse programa é executado, o vírus faz uma varredura no sistema e pode acabar infectando outros programas. Muitas vezes, pode abrir portas para que sejam feitas conexões não permitidas pelo usuário, mais conhecidas como *backdoors* (RAVANELLO *et al.*, 2004).

O ataque por vírus também pode afetar partições primárias do disco rígido como a MBR (Master Boot Record), sendo necessário executar uma reinstalação do sistema operacional (TEIXEIRA JR. *et al.*, 1999). Moraes (2010) menciona que o *backdoor* abre as portas para servir de acesso ao *cracker*, deixando que ele realize o acesso ao sistema que está vulnerável a qualquer momento, para assim, poder acessar os demais dispositivos da rede através dessa máquina vulnerável. Na maioria dos casos, as portas liberadas são de serviços HTTP (Hypertext Transfer Protocol) e FTP (File Transfer Protocol).

Outra forma de ataque são os vermes da Internet, também conhecidos como *worms*, Em geral, representam a maioria dos ataques à segurança dos computadores. Diferente do vírus, que se aloja em um programa, o *worm* é um programa completo. O primeiro *worm* foi desenvolvido por um estudante que descobriu dois *bugs* no sistema operacional Unix, fazendo com que seu código se replicasse automaticamente para explorar esses *bugs*, ganhando acesso a outras máquinas (TEIXEIRA JR. *et al.*, 1999).

Oliveira (2008) acrescenta que *worms* não necessitam de uma máquina para atuar como hospedeiro, pois eles trafegam diretamente na rede, comprometendo-a em decorrência de um grande fluxo e consumo de dados gerado na tentativa de ataque a outras máquinas.

Outro método muito utilizado para ataque são os *exploits*. Eles podem ser uma sequência de códigos, formando um *script* ou uma grande proporção de dados, que tira vantagem de falhas específicas de programas, serviços e sistemas operacionais. Permitem, assim, que o *cracker* consiga se beneficiar de algum recurso dessa aplicação. O SQL Injection pode ser considerado um *exploit*, pois explora vulnerabilidades em bancos de dados mal configurados. Assim, pode permitir a remoção ou modificação de tabelas, e até mesmo a extração dos dados contidos em um banco (OLIVEIRA, 2008).

A vulnerabilidade de serviços *web* é bastante explorada pelos *crackers*. Eles utilizam táticas de *buffer overflow* com as quais tentam gravar mais dados do que o permitido em um *buffer*, gerando assim um transbordamento da memória. Com isso conseguem fazer varreduras nessa memória ou até mesmo gerar um problema de

negação de serviço conhecido como DoS (Denial-of-Service), lotando assim os discos do servidor e fazendo com que o mesmo fique impossibilitado de realizar suas atividades. Servidores de banco de dados podem sofrer com ataques de SQL Injection no qual o *cracker* tenta passar uma instrução em SQL e, dessa maneira, pode derrubar a base de dados gerando um DoS, ou conseguindo extrair informações confidenciais. Além dos servidores de banco de dados, serviços como o IIS (Internet Information Services) da Microsoft e o Apache também são alvos desses *crackers*, pois possuem várias versões com falhas (MORAES, 2010).

Segundo Nakamura e Geus (2007), quem realiza um ataque de negação de serviço tipicamente não possui o intuito de roubar alguma informação ou realizar algum acesso especial, mas sim de impossibilitar que o usuário utilize aquele serviço. Após o ataque, o serviço ficará indisponível ao usuário.

Existem os ataques de negação de serviços distribuídos, conhecidos também como DDoS (Distributed Denial of Service), que têm o mesmo objetivo de um DoS, porém, afetando uma quantidade maior de máquinas. O *cracker* pode então comprometer e controlar várias máquinas na rede, fazendo com que mais de uma máquina ataque um único alvo (MORAES, 2010).

Esses ataques de DDoS são eficientes, sendo que a vítima fica indefesa e sem conseguir verificar a origem dos mesmos, devido aos níveis hierárquicos das máquinas atacantes. Esses níveis funcionam da seguinte maneira: o *cracker* invade uma máquina e a utiliza para atacar outra, e assim sucessivamente. Uma máquina já invadida é classificada como mestre, que passa instrução para outras máquinas que já foram invadidas e atuam como escravas. Assim, todos esses *hosts* infectados e controlados pelo *cracker* começam a realizar uma varredura de portas (*port scanning*), procurando por sistemas vulneráveis ou serviços que possuem alguma falha (NAKAMURA; GEUS, 2007).

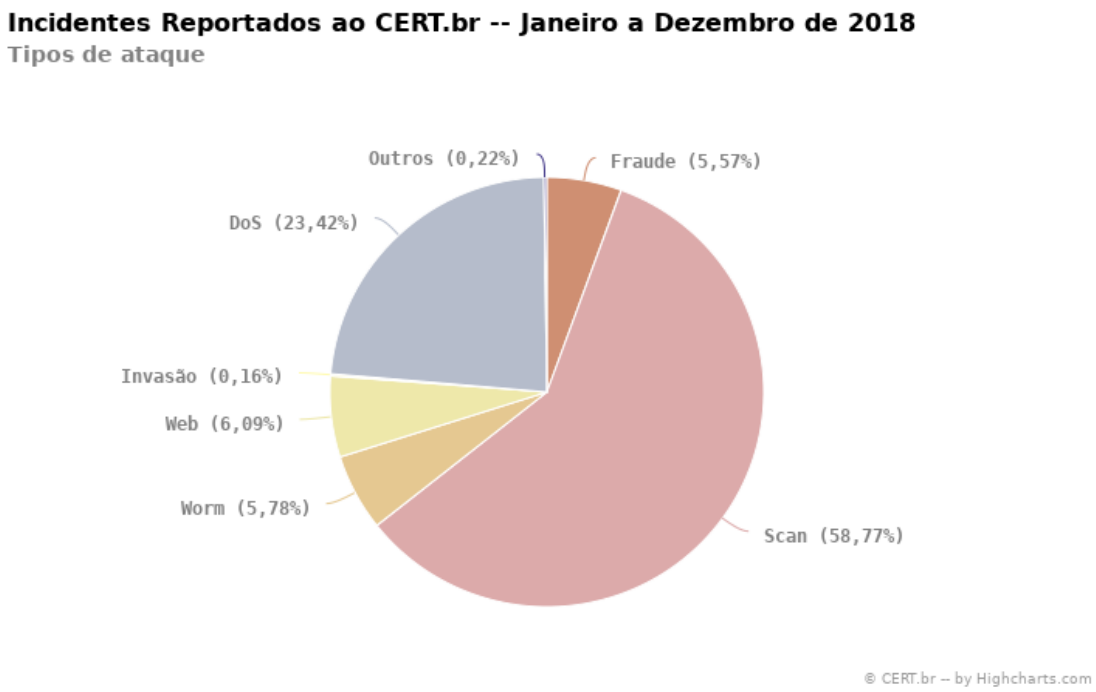
O *port scanning* ocorre através de ferramentas utilizadas para extrair informações de serviços ativos na rede e em outros computadores. Esses serviços são mapeados nas portas TCP (Transmission Control Protocol) e UDP (User Datagram Protocol), possibilitando a descoberta de serviços vulneráveis e portas abertas para alguma conexão específica. O programa mais utilizado para realizar o

port scanning é o nmap, que pode ser utilizado para realizar uma auditoria de *firewall*. Além dessa função, ele pode verificar se o sistema possui vulnerabilidades na pilha TCP/IP. Também pode mostrar todas as portas que estão abertas, com seus respectivos serviços, e até mesmo qual o sistema operacional do computador, sendo assim um prato cheio para os *crackers* (NAKAMURA; GEUS, 2007).

Moraes (2010) explica que o *cracker* verifica as versões dos sistemas operacionais e as versões dos serviços que estão em execução nas portas utilizando *port scanning*. Caso o sistema esteja desatualizado, o *cracker* poderá se favorecer de um *bug* ou *exploit* já revelado, permitindo que o atacante tenha privilégios de administrador do sistema.

Segundo CERT.br (2019), mais de 50% das notificações de ataques causados no ano de 2018 foram realizados a partir de *port scanner*, fazendo com que essa técnica seja crucial quando o *cracker* se prepara para atacar. Assim, a maioria dos ataques começa com uma varredura nas portas, seguido de um ataque DoS, como ilustra a Figura 5.

Figura 5 - Tipos de ataques reportados no ano de 2018.



Fonte: CERT.br (2019).

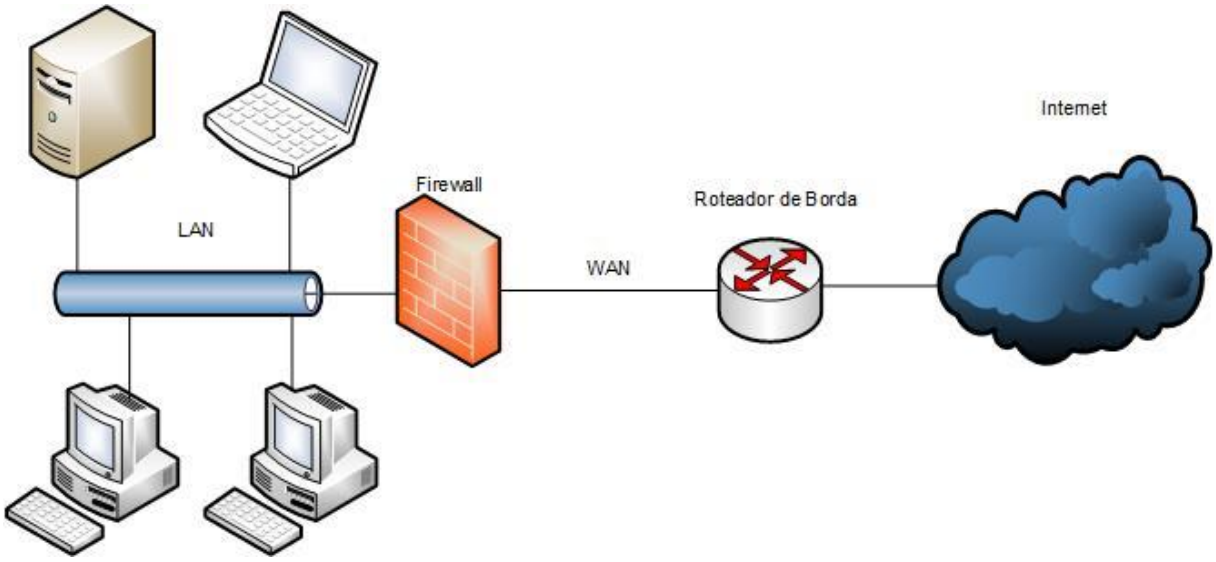
2.2.2 Prevenções e dispositivos de segurança

Qualquer recurso que tem o intuito de melhorar a segurança pode ser visto como uma ferramenta de segurança, porém, nem sempre este dará total garantia para a rede e seus dados. Entretanto, podem conseguir analisar e filtrar uma grande parte dos pacotes que trafegam na rede, além de analisar as vulnerabilidades e alertar os sistemas de possíveis arquivos e programas maliciosos (OLIVEIRA, 2008).

O *firewall* é um dispositivo que atua entre a rede pública e a rede privada, em que todo o tráfego que entra e sai da rede privada passa por ele. Desse modo, este pode ser considerado o único meio de bloqueio e defesa de qualquer ataque que venha da rede externa. O *firewall* pode realizar bloqueios e permissões, além de realizar um registro de tudo que passa sobre ele (MORAES, 2010).

Nakamura e Geus (2007) acrescentam que o *firewall* também pode ser definido como um sistema único ou um grupo de sistemas que dão um reforço na política de segurança e de acesso entre duas redes. Caso o *firewall* seja muito restritivo, muitos usuários podem procurar formas de contornar essa situação para realizar tarefas que realizavam normalmente antes da implementação das novas regras. Uma arquitetura de *firewall* é exibida na Figura 6.

Figura 6 - Arquitetura de um *firewall* na rede.



Fonte: Adaptada pelo autor, com base em Moraes (2010).

A principal função do *firewall* é o filtro de pacotes. Com essa função é possível realizar bloqueios de entrada e saída dos pacotes. Essas regras são definidas de acordo com a política de segurança de cada corporação, sendo possível limitar o acesso dos usuários e evitar que conexões indesejadas sejam feitas. O *firewall* não precisa necessariamente ser utilizado entre uma rede e outra, sendo possível utilizar um *firewall* pessoal em um computador qualquer ou até mesmo em um servidor por meio da ferramenta Iptables, que está presente no *kernel* do sistema operacional Linux (OLIVEIRA, 2008).

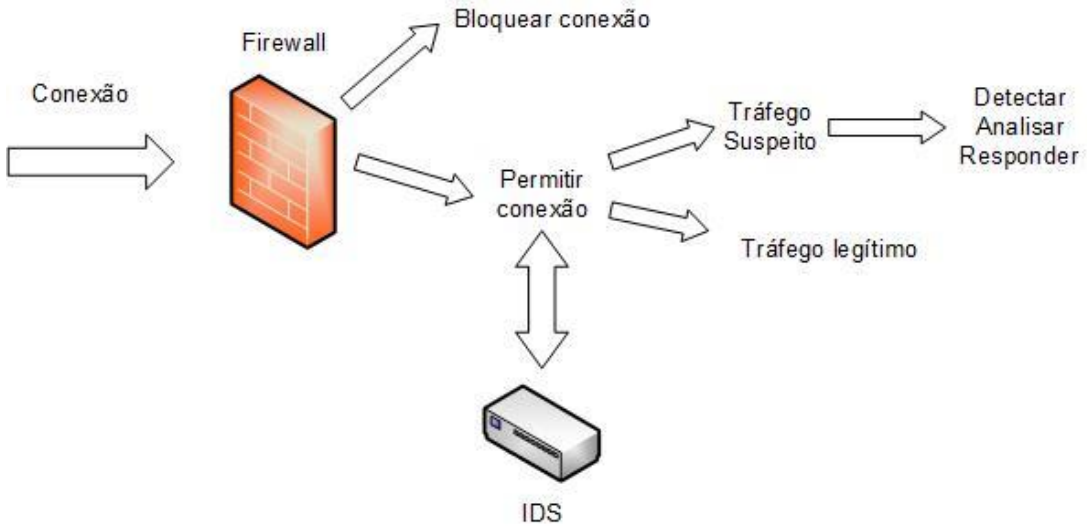
Nakamura e Geus (2007) acrescentam que o *firewall* pessoal está sendo cada vez mais utilizado devido às conexões Peer-to-Peer (P2P) e às redes privadas virtuais (VPN), as quais criam um túnel entre outros dispositivos externos. Sendo assim, o *firewall* que atua no limite da rede não é o suficiente para a segurança: caso um *host* seja invadido, o mesmo pode criar um acesso externo, e também um fluxo paralelo que evita que o fluxo de dados passe pelo *firewall*.

O filtro considera apenas os endereços IP e portas TCP e UDP, trabalhando com uma lista de controle de acesso que é verificada toda vez que um pacote é enviado. De acordo com o cabeçalho do pacote, o *firewall* consegue verificar se aquele endereço ou porta consta na lista e se pode encaminhar ou bloquear o pacote. Mesmo com um *firewall* bem configurado e uma boa política de segurança, não podemos garantir que a organização estará livre dos ataques, visto que o *firewall* poderá ficar suscetível a ataques de fragmentação, uma vez que o tráfego das redes não é totalmente isolado. As empresas tradicionalmente se preocupam somente em fechar as portas com um dispositivo *firewall,* desconhecendo os sistemas de detecção de intrusões (MORAES, 2010).

O sistema de detecção a intrusão (IDS) trabalha como se fosse uma câmera e um alarme, sinalizando alarmes ao detectar assinaturas de ataques ou de comportamentos estranhos na rede, como mostra a Figura 7. O IDS também pode alertar alguma falha devido a erros do sistema ou de usuários. Dessa maneira, o IDS alerta os administradores sobre algum comportamento estranho ou possíveis ataques que a rede possa estar sofrendo. Este também pode fazer com que a empresa melhore sua política de segurança, sinalizando possíveis ataques internos, seja por meio de funcionários, ou até mesmo por alguma máquina infectada. Assim

podemos dizer que o IDS coleta, analisa, armazena e responde às atividades suspeitas (NAKAMURA; GEUS, 2007).

Figura 7 - Arquitetura de um IDS na rede.



Fonte: Adaptada pelo autor, com base em Nakamura e Geus (2007).

Moraes (2010) diz que os alarmes dados por dispositivos IDS nem sempre são ataques, pois esses dispositivos detectam problemas ou anomalias. Quem deve determinar se essas anomalias são ataques é o administrador do sistema, porque é ele quem irá realizar a configuração e a determinação de todos os eventos. Assim, o IDS só irá notificar o administrador, e desse modo, recomenda-se ter um administrador de plantão, pois caso o IDS alerte sobre alguma anomalia, o mesmo deverá estar disponível para realizar uma intervenção manual e verificar se trata-se de um ataque. Oliveira (2008) complementa que o IDS é bem semelhante ao *firewall*, porém, ele só tem a função de alertar algum dado que já teve a sua passagem pelo *firewall*, mas que se trata de uma anomalia ou intrusão.

Os dispositivos IDS são fundamentais para auditorias e contra-ataques, porque é a partir deles que se tem a possibilidade de realizar uma análise sobre os ataques e, devido a esse conhecimento, melhorar o sistema de segurança da corporação. Se esses sistemas IDS podem ensinar conceitos e melhorar o aprendizado sobre ataques, os *honeypots* podem passar uma informação muito mais aprofundada, pois com eles é possível descobrir algumas informações de quem realizou o ataque (NAKAMURA; GEUS, 2007).

2.3 Honeypots e honeynets

O *honeypot* (pote de mel) é uma ferramenta destinada a interagir com um *cracker*, se passando por um equipamento legítimo da empresa, além de simular falhas e portas abertas para atrair os atacantes. Podemos considerar que o *honeypot* é uma armadilha, uma isca para os *crackers* e que, a partir dele, o administrador do sistema pode aprender muito mais sobre os ataques. O dispositivo permite capturar todo o fluxo de dados que passa sobre ele, sendo possível aprimorar a segurança da rede local e, ao mesmo tempo, adquirir maior conhecimento sobre ataques novos e já existentes (NAKAMURA; GEUS, 2007).

Ravanello *et al*. (2004) complementa falando que a principal arma que podemos utilizar contra um *cracker* é conhecer seus passos e conseguir prevenir suas ações. Com o uso dessa ferramenta, conseguimos não apenas visualizar quando o dispositivo está sendo atacado, mas também podemos visualizar como ocorre esse ataque e como o sistema se comporta após o ocorrido.

O *honeypot* tem a intenção de transparecer para o *cracker* que navegará em um ambiente de produção real, pois o administrador pode implementar serviços falsos, deixando portas abertas e, além disso, simular servidores de banco de dados e sistemas operacionais. Isso fará com que o *cracker* pense que está sob controle da situação, mas o *honeypot* além de estar mapeando os passos desse *cracker*, também desviará o foco principal do atacante, atrasando assim ataques não efetivos (JUNIOR, 2006). Rodrigues *et al*. (2015) complementa dizendo que, caso o *cracker* tenha a sensação de estar navegando em um ambiente real, o *honeypot* estará bem estruturado.

Nakamura e Geus (2007) falam que uma das características do *honeypot* é que não existem alarmes falsos positivos iguais aos dispositivos IDS comuns, sendo que todo tráfego que passa por ele é real. Dessa forma, os ataques que levam tempo para serem efetivados podem ser detectados a partir de um registro de todas as ações que são realizadas contra esse sistema, levando assim a organização a sofrer drasticamente uma redução de ataques oportunos. Oliveira (2008) acrescenta que um *honeypot* possui baixo número de falsos positivos, porque este não possui nenhuma atividade produtiva ou autorizada. Então, cada pacote que trafega por ele

é considerado vírus, *port scan* ou ataque e, assim, os administradores do sistema conseguem detectá-los com maior facilidade.

As *honeynets* são redes repletas de *honeypots* junto a um *firewall*, preparadas para serem comprometidas, ou seja, são preparadas especificamente para serem atacadas. Mesmo assim, são ambientes controlados, onde não haverá nenhum risco para outros usuários e o *cracker* terá um alto nível de interação devido a sua construção ser somente com serviços reais. As *honeynets* trabalham como se fossem um grande *honeypot* de pesquisa, sendo que seu maior problema é sua construção, pois não existem soluções prontas. Então, devem ser construídas de acordo com a necessidade do administrador da rede e de sua aplicação (JUNIOR, 2006).

2.3.1 Classificações e níveis de envolvimento

Os *honeypots* podem ser classificados em *honeypots* de produção e de pesquisa. Os *honeypots* de produção possuem apenas o objetivo de proteger a organização em que foram instalados, detectando atividades suspeitas e possíveis anomalias na rede. Esses *honeypots* utilizam uma configuração semelhante à rede na qual foram instalados. Assim, cada ataque que for efetuado irá agregar conhecimento para o administrador a partir dos *logs* gerados e, desse modo, a rede ficará mais segura (OLIVEIRA, 2008).

Junior (2006) explica que esses *honeypots* de produção são baseados em sistemas operacionais virtuais e em serviços que são emulados para simular uma rede empresarial de verdade. Eles também são de baixa interatividade, porque o atacante que chegar a esse *honeypot* irá sofrer restrições nas suas ações, por se tratarem de serviços falsos. Ravanello *et al*. (2003) complementa que o principal encargo de um *honeypot* de produção é distrair ou dispersar o *cracker*, fazendo com que ele perca seu tempo e, assim, evitando que cause um dano maior à organização. Até o cracker realizar o seu ataque, o administrador do sistema já estará ciente que esse ataque irá ocorrer.

Um *honeypot* de pesquisa tem a intenção de servir como base de estudo para entender melhor a comunidade *cracker*, não possuindo a mesma função de um *honeypot* de produção atuando como IDS. Um *honeypot* de pesquisa possui a função de detectar quais ferramentas podem ser utilizadas para verificar a vulnerabilidade da rede e quais *exploits* podem ser utilizados para comprometer uma falha. Desse modo, não possuem nenhum valor para a organização por se tratarem de equipamentos específicos, focados nas ações do *cracker*, não estando focados para alertarem as intrusões (AZEVEDO, 2005).

Ravanello *et al*. (2003) acrescenta que os *honeypots* de pesquisa não emulam serviços, mas possuem um papel muito importante de capturar uma quantidade de informações mais detalhadas, sendo possível descobrir qual foi o tipo de ataque utilizado, qual sua origem, suas assinaturas e como o atacante conseguiu comprometer o sistema, indicando quais ferramentas este utilizou.

Os *honeypots* possuem três níveis de envolvimento conforme interagem com o atacante: nível alto, nível médio e nível baixo. Os de nível baixo são equivalentes a serviços emulados, onde o atacante possui uma interação bem limitada com o *honeypot*, gerando um registro bem enxuto, constando apenas as tentativas de conexão (AZEVEDO, 2005).

Nos *honeypots* de nível médio a interação entre o atacante e o sistema se torna maior, mas ainda não pode ser comparada com um sistema real, tendo uma interação limitada. É nessa fase em que se deve aumentar os cuidados de decidir quais *scripts* e ferramentas serão utilizados para interação do atacante, pois a qualquer momento ele pode perceber que está atacando um *honeypot*. Com essa interação média, mais dados e informações são obtidos sobre o ataque e sobre o atacante (OLIVEIRA, 2008).

Os *honeypots* de nível alto de interação possuem serviços reais, sem qualquer tipo de emulação. Como são servidores e serviços reais, possuem uma total interação com o atacante, passando assim uma sensação completamente real de estar acessando a rede principal da corporação. Além disso, é possível agregar maior quantidade de informações das ações e recursos utilizados pelo atacante. Porém, devido a esse elevado grau de interação, os riscos tendem a aumentar, pois

se tratam de máquinas reais e, caso o *cracker* consiga manipular esse *honeypot*, ele pode utilizar o mesmo para atacar outras máquinas na rede, trazendo assim um grande risco (AZEVEDO, 2005).

2.3.2 Localização e implementação

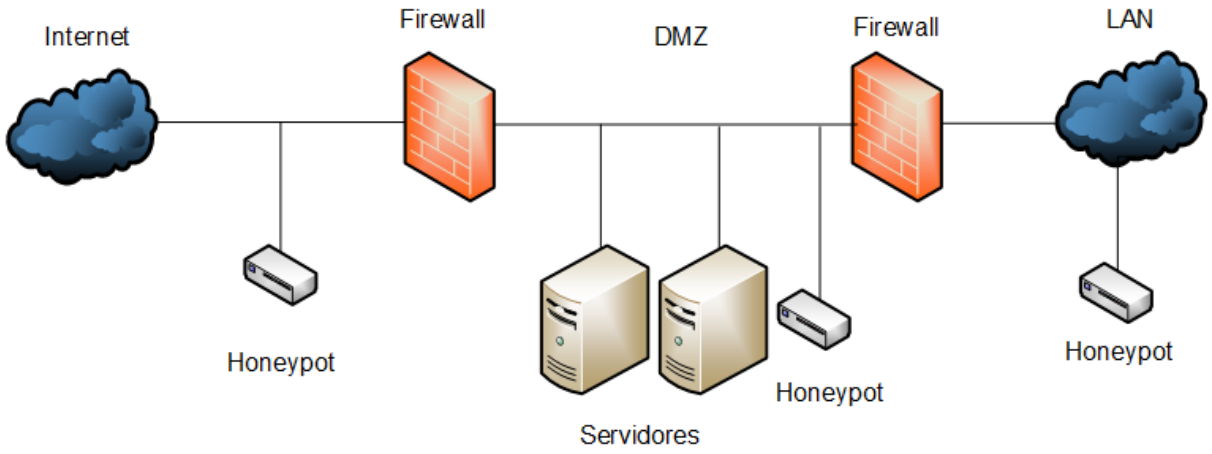
O *honeypot* pode ser implementado em três locais diferentes, considerando que a corporação possua uma DMZ (zona desmilitarizada), como mostra a Figura 8. Ele pode ser localizado antes do primeiro *firewall*, sendo exposto ao máximo, sem quaisquer métodos preventivos. Geralmente os *honeypots* de pesquisa são utilizados dessa maneira (ASSUNÇÃO, 2009).

Oliveira (2008) complementa que o *firewall* não gerará nenhum registro e o *honeypot* poderá ser totalmente comprometido, colocando em risco outras redes, pois será gerado um tráfego muito alto no *link* de entrada e que poderá atrair vários atacantes e ainda ser uma grande barreira para a velocidade da rede. A vantagem de utilizar o *honeypot* antes do *firewall* é que não será necessária a configuração do *firewall* ou de qualquer outro dispositivo IDS, pois não conseguirão detectar esses ataques.

Na DMZ, o *honeypot* fica na mesma camada dos servidores principais da rede, não ficando tão exposto quanto na primeira situação antes do *firewall*. Assim, o *honeypot* ainda estará suscetível a ataques, mas não atrairá novos atacantes à rede. Esse é o melhor local para posicionar o *honeypot*. Caso o *cracker* faça uma varredura na rede e encontre os servidores, ele detectará esse “servidor” vulnerável e irá realizar o ataque diretamente ao *honeypot* localizado na DMZ (ASSUNÇÃO, 2009).

Os *honeypots* que ficam atrás do *firewall* (dentro da LAN) são utilizados para detectar invasões que possam surgir na rede interna, ou para detectar falhas no próprio *firewall*. Caso esse *honeypot* seja invadido, o atacante terá acesso à rede interna, sendo assim, poderá atacar outras máquinas a partir desse *honeypot*. Entretanto, os dispositivos de IDS e *firewall* irão gerar um *log* do ocorrido (OLIVEIRA, 2008).

Figura 8 - Localizações dos *honeypots* em uma rede.



Fonte: Adaptada pelo autor, com base em Assunção (2009).

2.3.3 Vantagens e desvantagens

A principal vantagem do *honeypot* é que ele é um recurso sem valor de produção, com o qual só há interação quando ocorre um ataque. Assim, ele aprimora as detecções ao reduzir o número de falsos positivos, agilizando o trabalho do administrador do sistema. Além disso, mostra ao administrador todos os procedimentos do ataque, sendo possível visualizar a identidade do atacante, quais ferramentas e falhas ele utilizou e também a motivação do ataque. Um ponto forte é que o *honeypot* não faz parte do sistema de produção da empresa, desse modo, ele pode ser inserido ou removido a qualquer momento para analisar os dados ou até mesmo se for comprometido (RAVANELLO *et al*., 2004).

A maior desvantagem de um *honeypot* é o seu campo de visão, visto que só é possível visualizar um ataque quando o mesmo é direcionado a ele. Caso o atacante descubra os *honeypots* da rede, ele pode atacar as outras máquinas e assim, nenhum dado será capturado pelo *honeypot*. Outra desvantagem é que um *honeypot* deverá ser bem configurado e estruturado para que não seja detectado com facilidade. Da mesma maneira, há o risco de alguém conseguir invadir o *honeypot* e a partir dele atacar os outros dispositivos presentes (AZEVEDO, 2005).

2.4 Trabalhos relacionados

O CERT (Centro de Estudos para Resposta e Tratamento de Incidentes em Computadores) possui um projeto junto ao honeyTARG Honeynet Project, que consiste em *honeypots* distribuídos, sendo uma rede distribuída por 28 cidades do Brasil, que possuem um *honeypot* de baixa interatividade. O projeto possui a intenção de aumentar a capacidade de detecção de ataques e suas tendências. Nesse projeto, são cobertos uma grande quantidade de endereços IPv4 da Internet do Brasil, utilizando como *honeypot* o HoneyD, que é um *honeypot* de código aberto. Este pode ser facilmente instalado em qualquer dispositivo que possua o sistema GNU/Linux. O projeto também tem o intuito de mapear diariamente *scans* de portas UDP e TCP que foram direcionados aos *honeypots,* disponibilizando dados atualizados para os grupos de tratamentos de incidentes, para que possam analisar a origem dos ataques e suas novas tendências.

De acordo com esse projeto, foi possível verificar que no ano de 2018 entre as portas mais atacadas estava a porta 23 do serviço TELNET, que não possui criptografia e teve o tráfego de 2,2 TB de dados de ataque. Em seguida, observou-se a porta 22 do serviço SSH (Secure Shell), que teve o tráfego de 205,2 GB. O dado mais impressionante foi o da porta 8291, que pertence ao serviço Winbox da Mikrotik, que teve uma versão com falhas e que obrigou o fabricante a recomendar a atualização do sistema para os usuários desse serviço (CERT.br, 2019). Os dados citados estão listados no Quadro 1.

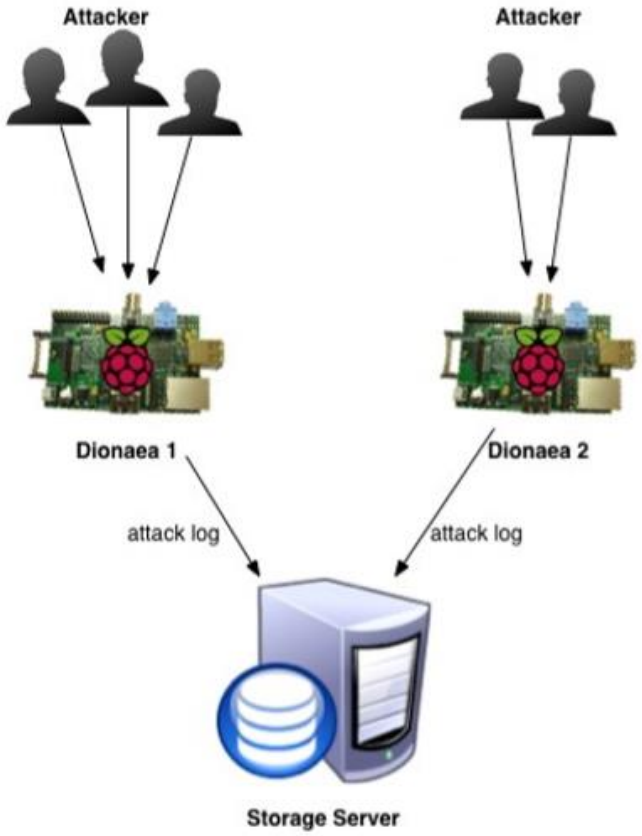
Quadro 1 - Portas TCP mais atacadas no ano de 2018.

20 Portas TCP que mais sofreram varreduras em 2018 Ordenadas por número de pacotes			
Porta TCP	**Bytes**	**Pacotes**	**Flows**
23	2.2 TB	46.4 G	740.4 M
22	205.2 GB	2.3 G	399.6 M
445	67.2 GB	1.2 G	829.1 M
80	35.2 GB	680.1 M	515.4 M
110	8.3 GB	189.2 M	35.4 M
1433	6.4 GB	165.9 M	138.0 M
8080	8.0 GB	157.2 M	106.4 M
81	3.4 GB	90.7 M	83.9 M
5555	2.7 GB	69.8 M	60.0 M
3389	2.6 GB	65.3 M	50.9 M
8545	2.3 GB	61.3 M	59.5 M
8291	2.2 GB	56.8 M	56.3 M
7547	2.0 GB	52.0 M	51.7 M
443	2.2 GB	45.8 M	33.3 M
2323	1.7 GB	45.6 M	45.3 M
5900	1.8 GB	43.2 M	18.6 M
8000	1.7 GB	27.5 M	16.2 M
82	995.0 MB	25.8 M	19.6 M
8088	972.4 MB	25.1 M	19.0 M
21	1.2 GB	24.4 M	14.4 M

Fonte: CERT.br (2019).

Outro trabalho relacionado foi efetuado por um grupo de estudantes da Swiss German University, que desenvolveram um *honeypot* distribuído, utilizando o Dionaea e um Raspberry Pi. Segundo Lim *et al*. (2014), eles utilizaram dois Raspberry Pi, cada um hospedando um *honeypot* Dionaea. Esse *honeypot* possui uma baixa interação e emula alguns serviços bem conhecidos como HTTP, TFTP (Trivial File Transfer Protocol), FTP e MySQL, assim, todo *log* gerado pelos dispositivos era enviado via protocolo XMPP (Extensible Messaging and Presence Protocol) para um servidor. A configuração usada neste trabalho é mostrada na Figura 9.

Figura 9 - Estrutura do *honeypot* distribuído com Dionaea.



Fonte: LIM *et al*. (2014).

Para realizar os testes, foi simulada uma rede com 50 usuários virtuais, incrementando mais 25 usuários que supostamente estariam realizando um ataque. Desse modo, foram obtidas em média 6535 conexões (tentativas de ataque) em uma hora, ou seja, cerca de duas conexões por segundo, tendo um pico de 400 conexões por segundo na simulação. Mostrou-se que o Raspberry Pi tem bastante capacidade para lidar com uma grande escala de ataques e concluindo-se que a implementação dos *honeypots* distribuídos com Raspberry Pi funcionou como o esperado. Segundo os autores, este demonstrou uma total eficiência do hardware em comparação com o seu custo, podendo substituir um computador comum (LIM *et al*., 2014).

3 PROCEDIMENTOS METODOLÓGICOS

Apresentam-se neste capítulo os métodos utilizados no desenvolvimento do trabalho, tendo como embasamento os conhecimentos teóricos. A metodologia está subdividida em três grupos: método de pesquisa, objetivos de pesquisa e procedimentos técnicos.

3.1 Métodos de pesquisa

O método de pesquisa empregado neste trabalho foi o experimental.

Prodanov e Freitas (2013), destacam que a Pesquisa Experimental é mais complexa pois possui o intuito de explicar, registrar, analisar os fenômenos estudados tendo como uma preocupação central, identificar os fatores determinantes, explicando a razão, o motivo e o porquê de todas as coisas estudadas na pesquisa, e justamente por esse motivo esse método de pesquisa está mais sujeito a erros devido a sua complexidade, além de ocorrer mudanças nas variáveis dos resultados já obtidos.

3.2 Objetivos de pesquisa

A metodologia de pesquisa adotada pelo trabalho quanto ao objetivo geral foi a metodologia descritiva e exploratória.

Segundo Prodanov e Freitas (2013), pesquisa descritiva é quando um pesquisador apenas requer saber os dados de uma população, estabelecimento ou um fenômeno, registrando e descrevendo os fatos observados sem quaisquer alterações, envolvendo o uso de coleta de dados padronizados. Em geral, é um levantamento de dados que visa apenas analisar, demonstrar ou explicar algum fenômeno ou situação ocorrida.

Para Gil (2012), a pesquisa descritiva tem como objetivo a descrição de características de algum local, população ou fenômeno. Nesse aspecto podem ser realizados vários estudos, mas a característica deles está relacionada com a coleta de dados através de técnicas padronizadas. São geralmente as pesquisas que são contratadas por algumas empresas comerciais ou então partidos políticos.

A pesquisa exploratória se dá quando a pesquisa está em sua fase preliminar e, tem o intuito de proporcionar maiores informações sobre o assunto tratado. Além disso, esse modelo de pesquisa proporciona um planejamento flexível, permitindo o estudo do tema retratado sob diversos aspectos e visões (PRODANOV; FREITAS, 2013).

3.3 Procedimentos técnicos

A metodologia de pesquisa adotada no trabalho, quanto aos procedimentos técnicos, foi a pesquisa bibliográfica.

Segundo Gil (2012), a pesquisa bibliográfica é feita a partir de um material já elaborado, principalmente livros e artigos, sendo algumas pesquisas desenvolvidas apenas a partir de fontes bibliográficas. A vantagem desse tipo de pesquisa é o fato de permitir uma cobertura muito mais ampla do assunto pesquisado. Prodanov e Freitas (2013) acrescentam que a pesquisa bibliográfica ocorre quando ela é elaborada a partir de um material que já foi publicado. Geralmente é extraída de

livros, revistas, artigos, jornais e internet, possuindo o objetivo de colocar o pesquisador em contato direto a todo o conteúdo pesquisado sobre o respectivo assunto abordado na pesquisa. Nesse método de pesquisa é importante que o pesquisador verifique a confiabilidade de suas fontes.

4 DISPOSITIVOS E CONFIGURAÇÕES

Neste capítulo serão descritas as etapas que foram seguidas para que o dispositivo principal do trabalho pudesse ser testado em configurações e cenários diferentes. Dessa forma, serão abordados o dispositivo principal que foi utilizado, as etapas de instalação do sistema operacional e as configurações das ferramentas utilizadas para transformar o dispositivo em um *honeypot*, além da metodologia de testes, cenário e topologia em que foram executados.

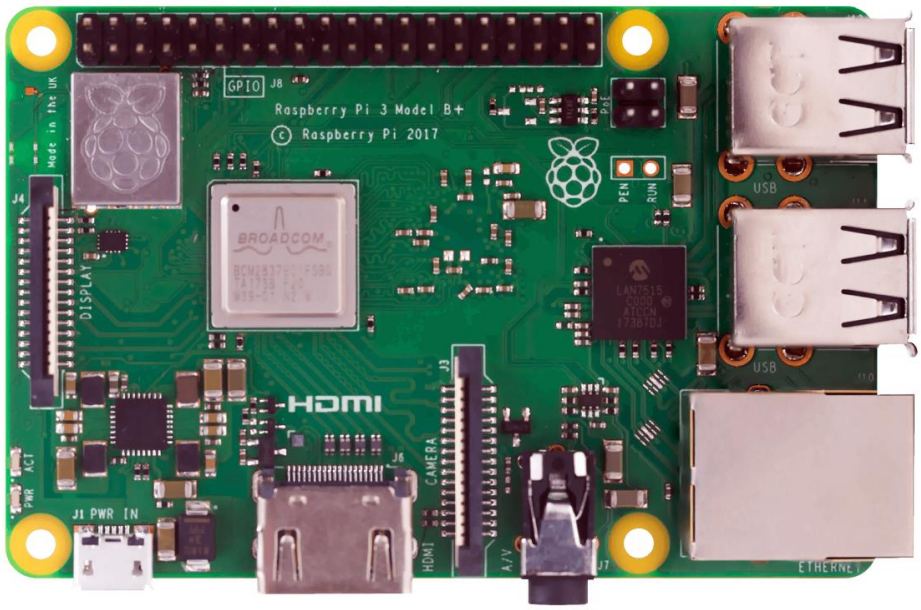
4.1 Hardware

Para hospedar o *honeypot* foi utilizado um microcomputador de placa única, produzido pela Raspberry Pi Foundation. O modelo que foi utilizado é o modelo 3 B+, que possui um desempenho bem superior aos seus antecessores.

Segundo a Raspberry Pi Foundation (2019), o modelo 3 B+ possui um processador que é um *chip* da marca Broadcom, modelo BCM2837B0, que possui quatro núcleos construídos com a arquitetura ARM, sendo do modelo Cortex-A53 (ARMv8) 64-bit SoC, que opera com uma frequência de 1,4 GHz.

Esse modelo, ilustrado na Figura 10, possui uma memória RAM de 1 GB, além de ter quatro *interfaces* USB (do inglês Universal Serial Bus), uma interface Gigabit Ethernet, *interface* Wi-Fi 2.4 GHz / 5 GHz, *interface* Bluetooth e saídas de áudio e vídeo.

Figura 10 - Raspberry Pi 3 modelo B+.

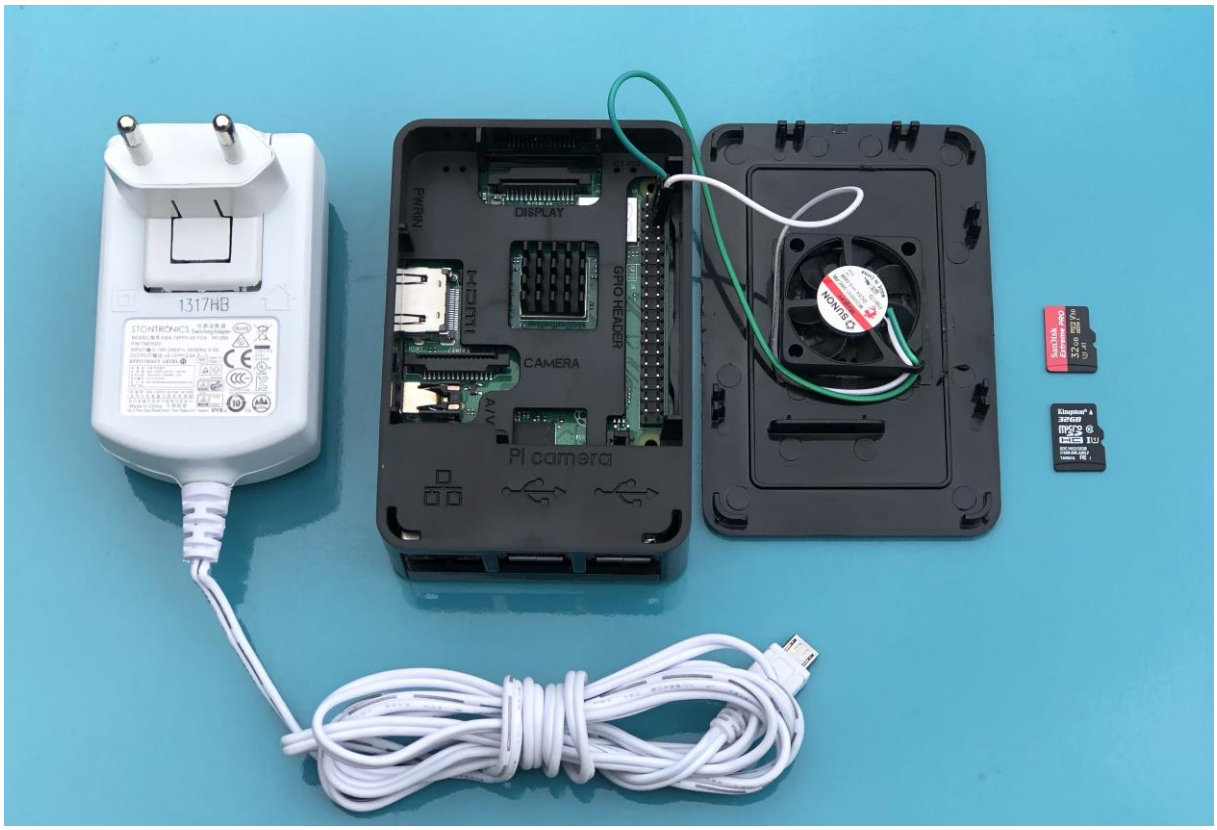


Fonte: Raspberry Pi Foundation (2019).

Juntamente com a placa, foi utilizada uma fonte de energia oficial da Raspberry Pi Foundation. Com ela foi possível suprir toda a demanda de energia exigida pela placa, já que a fonte consegue entregar uma tensão de 5,1 volts a uma corrente de 2,5 amperes, evitando que o microcomputador tenha qualquer problema de travamento devido à má alimentação. Além disso, foi feita uma melhoria para o resfriamento do hardware: o Raspberry Pi foi inserido dentro de um *case*, juntamente com um dissipador em seu processador e uma ventoinha para evitar quaisquer problemas de superaquecimento ao longo dos testes.

Para armazenar os dados e os sistemas operacionais que foram instalados, foram utilizados dois cartões de memória classe 10, com a capacidade de 32 GB de armazenamento, sendo que um cartão possui uma capacidade de leitura e escrita superior ao outro. Nesse caso, foi possível analisar o uso de leitura de disco dos sistemas operacionais, além de otimizar os testes com dois sistemas distintos prontos para o uso.

Figura 11 – Raspberry Pi instalado na case, juntamente com a fonte e cartões SD.



Fonte: Do autor (2019).

4.2 Preparação dos sistemas operacionais

Para que fosse possível realizar os testes e analisar o comportamento da placa, foram preparados e instalados os seguintes sistemas operacionais: Raspbian, Ubuntu Mate e Windows 10.

Inicialmente, o processo para preparar os sistemas operacionais foi bem complexo devido à falta de conhecimento das ferramentas que poderiam ser utilizadas para executar esse procedimento. Mas, após alguns testes de pacotes de instalação, foi possível encontrar ferramentas que proporcionam uma instalação bem simplificada para qualquer usuário que tenha um conhecimento entre básico e intermediário em Informática. Deste modo, serão listados a seguir os métodos de instalação mais simplificados para cada sistema.

4.2.1 Raspbian

O primeiro sistema operacional foi o Raspbian, que foi escolhido por ser o sistema operacional original do Raspberry Pi, e também por ser uma extensão do Debian. Ele possui um desempenho superior aos outros sistemas instalados nessa plataforma, consumindo poucos recursos do hardware e tendo um excelente rendimento.

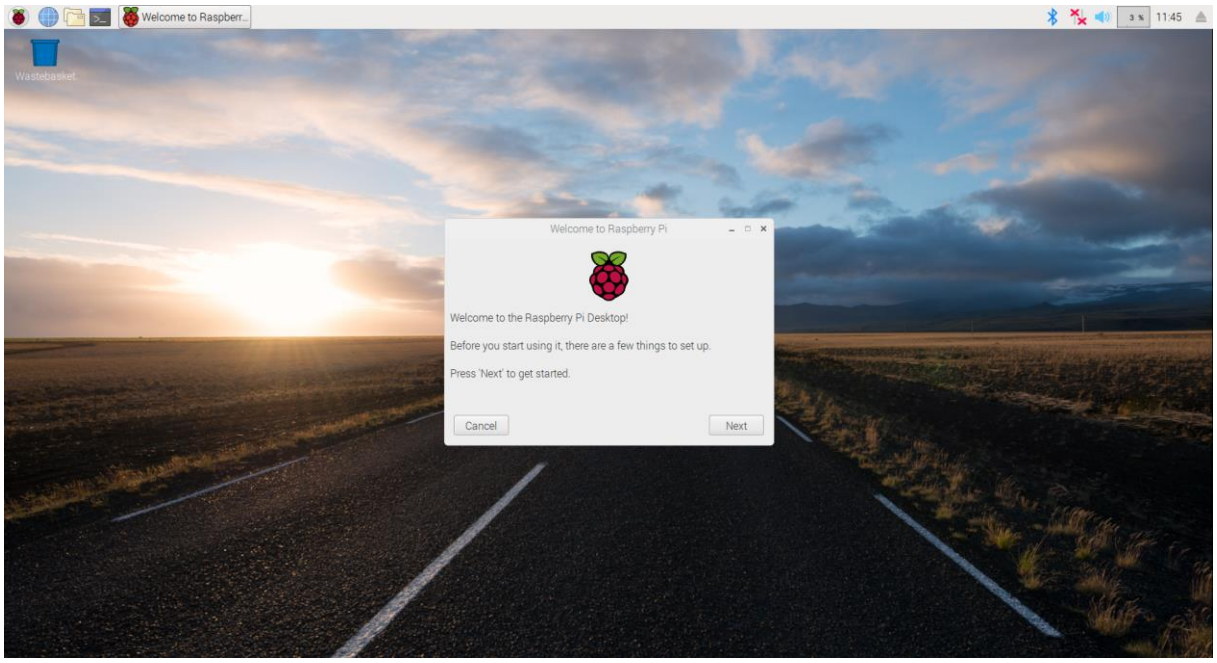
Segundo a Raspbian (2019), o sistema Raspbian é gratuito, baseado em Debian e otimizado para o hardware do Raspberry Pi. Nele são encontrados mais de 35 mil pacotes, não sendo considerado um sistema puro, mas sim, um sistema com muitas funcionalidades e com uma instalação simples.

Para iniciar a instalação do sistema operacional original do Raspberry, foi necessário utilizar as ferramentas SD Card Formatter e balenaEtcher. O SD Card Formatter serve para preparar o cartão de memória para receber quaisquer tipos de arquivos, enquanto o balenaEtcher é uma ferramenta que tem o intuito de gravar arquivos de imagem das extensões **.iso** e **.img** em cartões de memória SD (Secure Digital) e *pendrives* inicializáveis.

Após a conclusão da gravação dos arquivos de imagem no cartão de memória utilizando o software balenaEtcher, o Raspbian já está pronto para ser utilizado, não necessitando de nenhuma instalação adicional. Serão necessárias apenas algumas configurações básicas como selecionar o país, o *layout* de teclado, a data, a hora e a seleção da rede.

O Raspbian possui uma interface bem simplificada para o usuário, além de conter todas as funcionalidades de um sistema Debian. Este recebe atualizações periódicas, feitas oficialmente pela Raspberry Pi Foundation. A tela inicial do Raspbian pode ser visualizada na Figura 12.

Figura 12 – Interface do sistema operacional Raspbian.



Fonte: Do autor (2019).

4.2.2 Ubuntu MATE

O Ubuntu MATE 18.04 foi o segundo sistema operacional e o mais utilizado para a realização dos testes. A escolha desse sistema foi baseada na facilidade para instalação e utilização da plataforma Docker, tendo em vista que o Ubuntu contém muitos materiais e fóruns na Internet, que abordam o assunto e auxiliam o usuário na configuração e instalação de novas aplicações. Em comparação ao Raspbian, o Ubuntu MATE utiliza uma quantidade maior de recursos de hardware do Raspberry Pi. Além disso, necessita de um cartão de memória superior, que tenha uma maior velocidade de leitura e escrita, para que o usuário consiga utilizar o sistema com uma boa fluidez, sem que ocorram muitos travamentos.

Segundo o Ubuntu MATE Team (2019), o sistema Ubuntu MATE é estável e integra o *kernel* do Ubuntu com a *interface* MATE Desktop, que possui uma *interface* diferenciada e simplificada, atraindo mais usuários para utilizarem o sistema.

Para realizar a instalação do Ubuntu MATE também é necessário o SD Card Formatter, mas, neste caso, em substituição ao balenaEtcher, foi utilizado o software

Win32 Disk Imager para gravar a imagem do sistema no cartão de memória. Após realizada a gravação, o sistema ainda não está pronto, e devem ser feitas as configurações iniciais como: nome de usuário, seleção de rede, senha, idioma e *layout* do teclado. Desta maneira, o sistema operacional irá concluir a sua instalação e estará pronto para o uso, incluindo várias ferramentas básicas pré-instaladas como navegador, editor de texto e LibreOffice, melhorando muito a interação com o usuário, mas ao mesmo tempo reduzindo o desempenho do Raspberry Pi (requerendo um cartão de memória mais rápido).

Figura 13 – *Interface* do sistema operacional Ubuntu MATE.

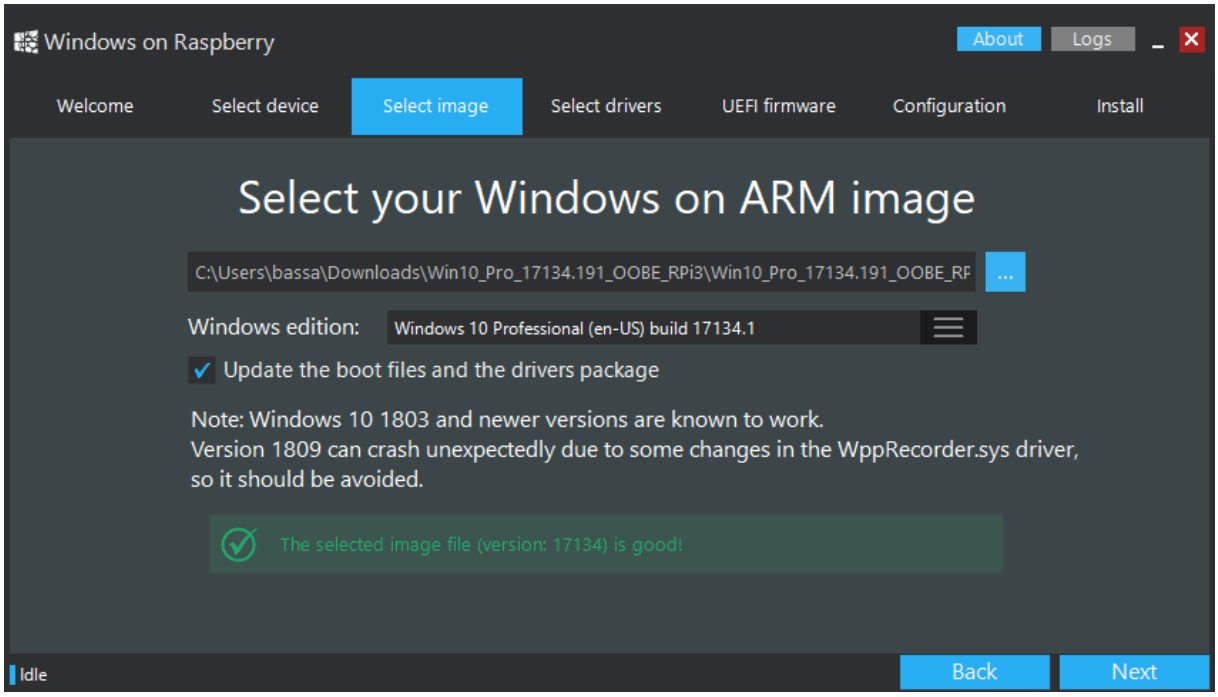
Fonte: Do autor (2019).

4.2.3 Windows 10

A escolha do sistema operacional Windows foi devido aos requisitos do Valhala Honeypot, que só tem compatibilidade com esse sistema. Dessa forma, foi feita a instalação de um Windows compatível com o Raspberry Pi, sendo este o Windows 10 IoT Core, porém, depois verificou-se que essa versão em particular não tinha compatibilidade com o Valhala Honeypot.

Então, a única forma foi achar uma adaptação desse sistema Windows para Raspberry Pi e, após muita busca, foram encontradas algumas soluções de emulação a partir da ferramenta QEMU, que possibilitou a instalação do Windows. Essa opção, porém, se mostrou inviável, tendo em vista a dificuldade da instalação e o uso excessivo de processamento do Raspberry Pi. A outra solução, e a melhor descoberta, foi o WOR (Windows on Raspberry) Project, que é um projeto com o intuito de instalar o Windows no Raspberry Pi de uma forma mais fácil e eficiente.

Figura 14 – Interface do WOR para gravar a imagem do sistema no cartão SD.



Fonte: Do autor (2019).

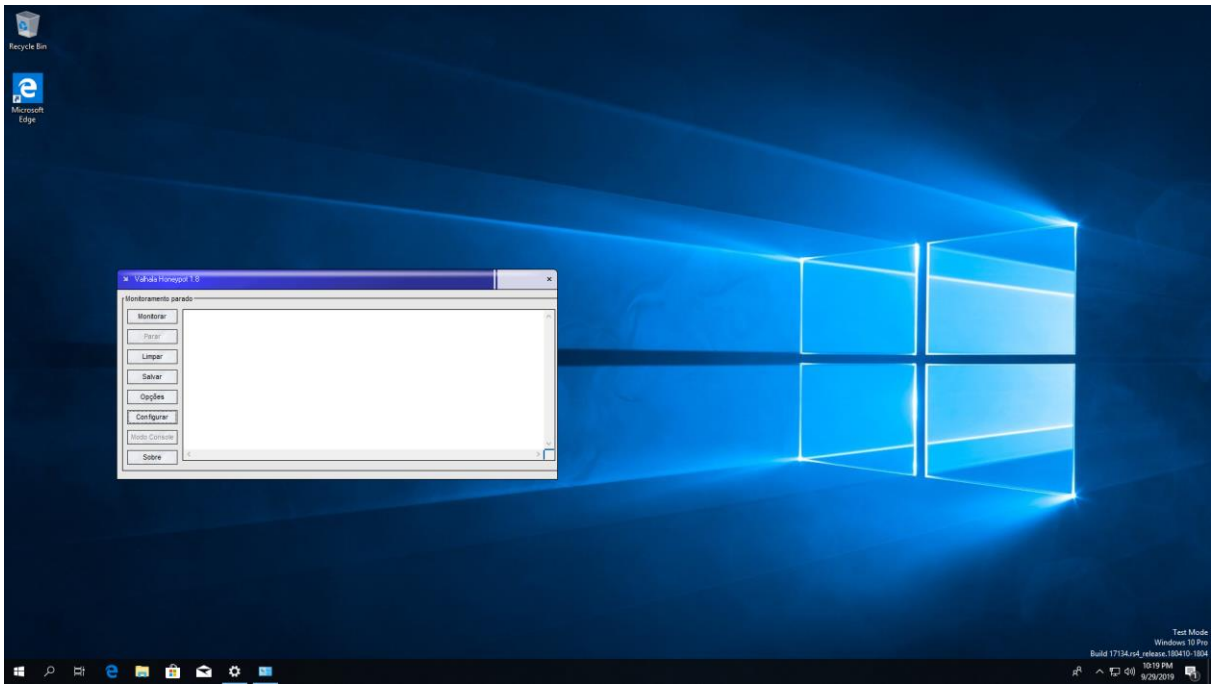
Instalar o Windows no Raspberry Pi por meio do WOR Project é uma tarefa mais simples, semelhante à instalação do Raspbian e do Ubuntu MATE. Entretanto, o Windows requer alguns passos adicionais na instalação, como ter a imagem de uma versão específica do Windows 10 ARM compatível com o WOR Project (no caso, foi utilizada a versão PRO, build 17134.1). Precisa também de um pacote de *drivers* que é disponibilizado pelo próprio projeto, além de um pacote de *firmwares* com várias versões da UEFI BIOS, que serve para transferir a BIOS para o cartão e para que o sistema possa ser iniciado.

Após a conclusão da instalação da imagem do Windows ARM no sistema, foi necessário acessar a BIOS e liberar toda a frequência de processamento do

dispositivo, além de concluir a instalação do Windows, selecionando o nome do usuário, senha e seleções básicas como uso de assistente virtual, localização e outros recursos padrão do Windows.

Destaca-se no Windows 10 ARM que, além de possuir todos os recursos do sistema operacional da Microsoft, ele também requer um cartão de memória com uma velocidade de leitura maior. Caso utilize um cartão classe 4 ou classe 10 simples, o usuário não terá uma boa fluidez ao utilizar o sistema.

Figura 15 – Interface do Windows 10 ARM com o Valhala Honeypot.



Fonte: Do autor (2019).

4.3 Preparação e configuração dos *honeypots*

Após a instalação dos sistemas operacionais, foi a vez de instalar as ferramentas e *honeypots* que foram utilizados para a realização dos testes.

O HoneyPi foi instalado no Raspbian, a plataforma Docker foi instalada no Ubuntu MATE, juntamente com o HoneyD e o Windows foi utilizado apenas com o Valhala Honeypot.

4.3.1 HoneyPi

No sistema operacional Raspbian foi feita a instalação do HoneyPi, que será o *honeypot* utilizado no primeiro cenário. O HoneyPi trabalha com um detector de *scan* de portas, o PSAD (Port Scan Attack Detector), que utiliza o iptables para escutar as portas e verificar tentativas de conexão a elas. Ele também consegue realizar notificações por *e-mail.* Além disso, o HoneyPi possui um *script* simples em Python, que é capaz de abrir algumas portas falsas, com o intuito de atrair os *crackers*, fazendo com que eles “mordam a isca”. As portas falsas abertas por padrão são as portas 21, 23 e 5900 dos respectivos serviços FTP, Telnet e VNC (Virtual Networking Computing).

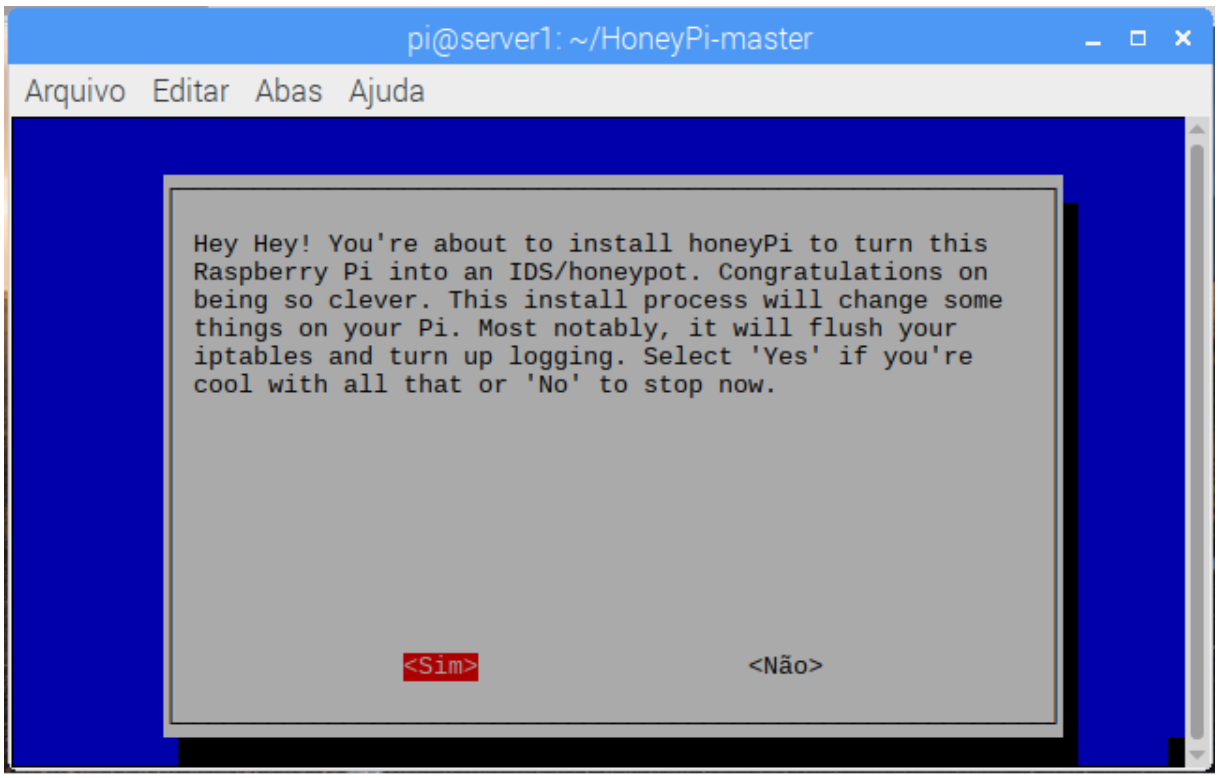
O HoneyPi é um software de fácil utilização e possui uma fácil configuração padrão, porém, mesmo configurando-o da forma correta, alguns problemas podem ocorrer, deixando o administrador sem aviso por meio de *logs* ou *e-mail*. Caso o usuário queira maiores funcionalidades do HoneyPi, ele terá que buscar nos diretórios do Linux, pois os arquivos ficam espalhados, e não em um diretório específico, dificultando assim a configuração e adição de melhorias para o *honeypot*. Para instalar o HoneyPi, devemos utilizar os seguintes comandos:

- # wget https://github.com/mattymcfatty/HoneyPi/archive/master.zip: baixa o HoneyPi pelos repositórios do Github;
- # unzip master.zip: extrai o arquivo master baixado anteriormente;
- # cd HoneyPi-master: vai para o diretório de extração dos arquivos;
- # chmod +x *.sh: dá a permissão para executar todos os arquivos que terminam com .**sh** e que estejam no mesmo diretório;
- # sudo ./honeyPiInstaller.sh: executa a instalação do HoneyPi.

Após isso, inicia-se sua instalação. Dessa forma, aparece a configuração do HoneyPi no terminal do sistema, bastando seguir os passos de instalação: alterar a senha de administrador raiz (do inglês *root*), além de realizar as configurações para

receber os alarmes e endereço IP do dispositivo. O processo é ilustrado na Figura 16.

Figura 16 - Interface de configuração do HoneyPi.

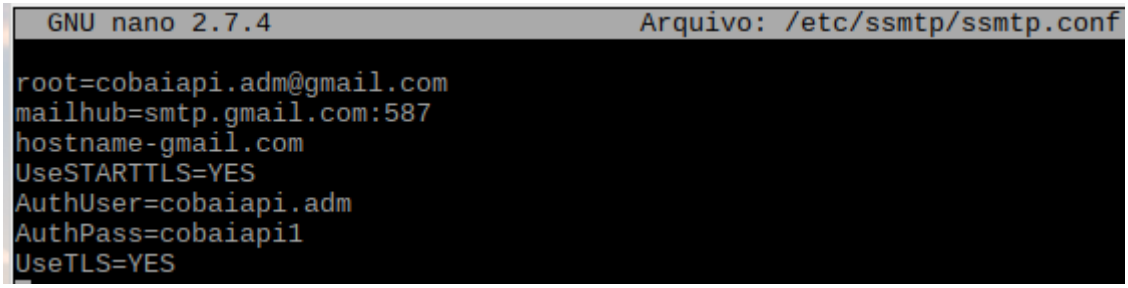


Fonte: Do autor (2019).

Para recepção dos alarmes, foi feita a configuração para recebê-los por *e-mail*, ou seja, caso alguma pessoa tente fazer uma varredura no *honeypot*, ele avisará o administrador do sistema via *e-mail,* informando qual o IP do atacante e quando o ataque ocorreu, além de mostrar o seu nível de risco.

Caso seja necessário realizar alguma alteração na configuração de *e-mail,* a mesma se encontra no diretório “/etc/ssmtp/ssmtp.conf”. Na Figura 17 podemos verificar a configuração realizada para receber os alertas por *e-mail*.

Figura 17 – Configuração de alertas por *e-mail* do HoneyPi.



Fonte: Do autor (2019).

O HoneyPi também possui a função de alerta via *script*, ou via *led*. A opção via *script* permite que o administrador do sistema crie um *script* que o alerte de alguma maneira personalizada, enquanto o alerta via *led* é o mais simples, apenas ligando um *led* que esteja conectado ao Raspberry Pi. Após realizar a configuração de alerta, basta reiniciar o Raspberry Pi e o HoneyPi estará pronto para ser utilizado.

4.3.2 Docker

Juntamente ao Ubuntu MATE, foi instalada a ferramenta Docker. Com essa ferramenta é possível virtualizar vários serviços distintos, sendo uma excelente opção para desenvolver um *honeypot* que tenha ótima segurança, sem correr um grande risco de ser invadido. Essa segurança é causada pelo fato do Docker isolar seus serviços da máquina em que está sendo hospedado, criando uma *interface* de rede virtual e, dessa forma, os serviços só conseguem ser acessados se houver um redirecionamento de IP, ou seja, feito uma *bridge* entre a placa de rede do servidor e o Docker.

O Docker é uma ferramenta que cria *containers*, os quais são unidades padrões de software que empacotam o código e podem ser utilizados em outras plataformas. Em outras palavras, dentro de um *container* pode-se executar apenas uma aplicação ou serviço, sem ter a necessidade de uma máquina virtual completa. O Docker virtualiza apenas a parte lógica, e não o hardware da máquina, sendo por isso mais flexível e mais eficiente (DOCKER INC, 2019).

Para realizar a instalação do Docker no Ubuntu MATE, deve-se estar em modo administrador e utilizar os seguintes comandos:

- # apt-get install curl: instala o Curl para fazer o download do Docker;
- # curl -sSL https://get.docker.com | sh: faz o download do Docker e sua instalação;
- # sudo usermod -aG docker ubuntu: adiciona o usuário do sistema ao grupo do Docker;
- # sudo systemctl enable docker: habilita o Docker para iniciar juntamente com o sistema;
- # sudo reboot -h now: reinicia o Raspberry Pi para que a instalação seja concluída.

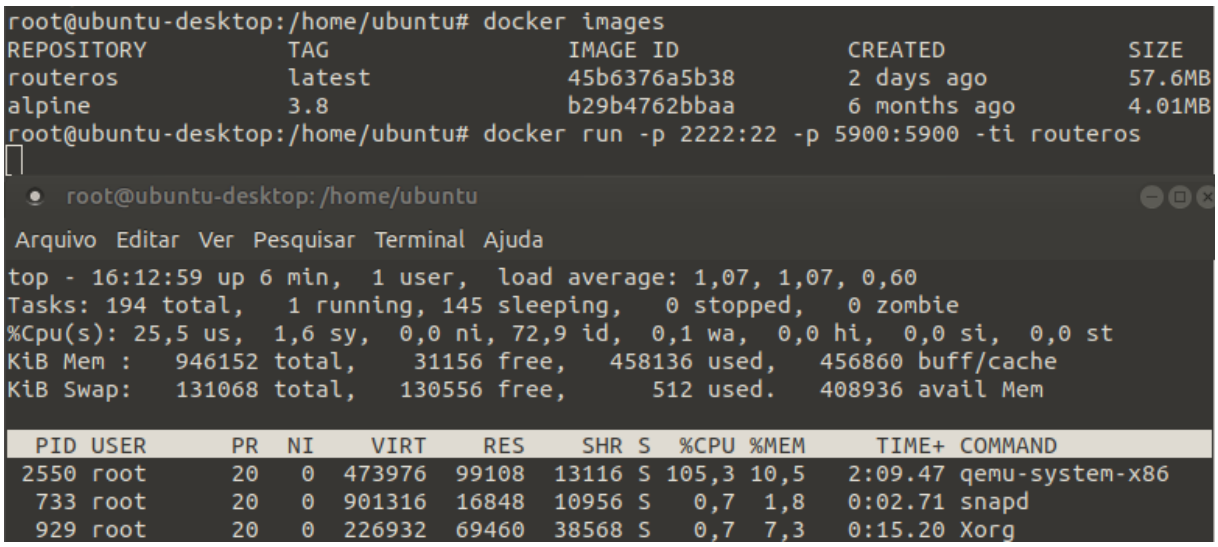
Após o reboot, basta aplicar o comando “docker ps” e verificar se o Docker está instalado, assim, procuram-se os *containers* que deseja-se executar na plataforma. Foram baixados alguns *containers* como RouterOS, Winbox, SSH Honeypot, Telnet Logger e Fake SSH.

Para aprender a utilizar o Docker e instalar os *containers*, utilizou-se uma máquina virtual com o sistema Lubuntu instalado, que permitiu verificar se os *containers* estavam de fato em funcionamento, antes de passá-los para o Raspberry Pi. Com isso, foi possível visualizar se o problema estava em algum *container* ou na compatibilidade com o Raspberry Pi.

Junto ao Docker foi instalado o sistema RouterOS da Mikrotik, com a intenção de desenvolver uma rede totalmente falsa, sendo uma *honeynet* através de um Docker, com o sistema de um roteador. A instalação do RouterOS primeiramente foi executada na máquina virtual, e apresentou um excelente funcionamento (ainda que um pouco lento para executar), sendo possível realizar o acesso ao sistema via *web*. Porém, ao instalar o *container* do RouterOS no Raspberry Pi, o mesmo não pôde suportar a quantidade de recursos exigidos pelo sistema RouterOS em cima de um *container*, com o consumo de processamento atingindo 100%. Caso alguém tente acessar as configurações do roteador instalado no *container*, o sistema trava e encerra o processo. Em virtude disso, testou-se o acesso por meio do Winbox, criando outro *container* e tentando acessar o RouterOS por meio de outro Docker. O

problema persistiu e, tendo isso em vista, decidiu-se realizar a busca de novos *containers* que atuam como *honeypots*.

Figura 18 – Consumo de processamento do RouterOS no Raspberry Pi.



Fonte: Do autor (2019).

Após a tentativa de executar o RouterOS, foi instalado o Fake SSH. Esse *container* tem o intuito de atuar como um serviço falso de SSH, atrasando o *cracker* e fazendo-o pensar que pode acessar a máquina com o serviço aberto. Ao instalar o Fake SSH, ele foi executado, porém, não foi possível deixar a porta visível a outras pessoas na rede, ou seja, esse *container* não funcionou como esperado.

Então, após a falha apresentada pelo *container* anterior, foi instalado o SSH Honeypot, que faz o mesmo serviço que o Fake SSH (simular um serviço SSH falso). Para realizar a instalação desse *container* deve-se alterar a porta do serviço de SSH padrão do Raspberry Pi. Para isso, o arquivo no endereço "# /etc/ssh/sshd_config" deve ser acessado e o valor de "Port 22" deve ser alterado para "Port 222". Assim, basta reiniciar o serviço com o comando "# sudo /etc/init.d/ssh restart" e executar a instalação do *container* com os comandos a seguir:

- # git clone https://github.com/random-robbie/docker-ssh-honey: faz o download do *container* no site GitHub;

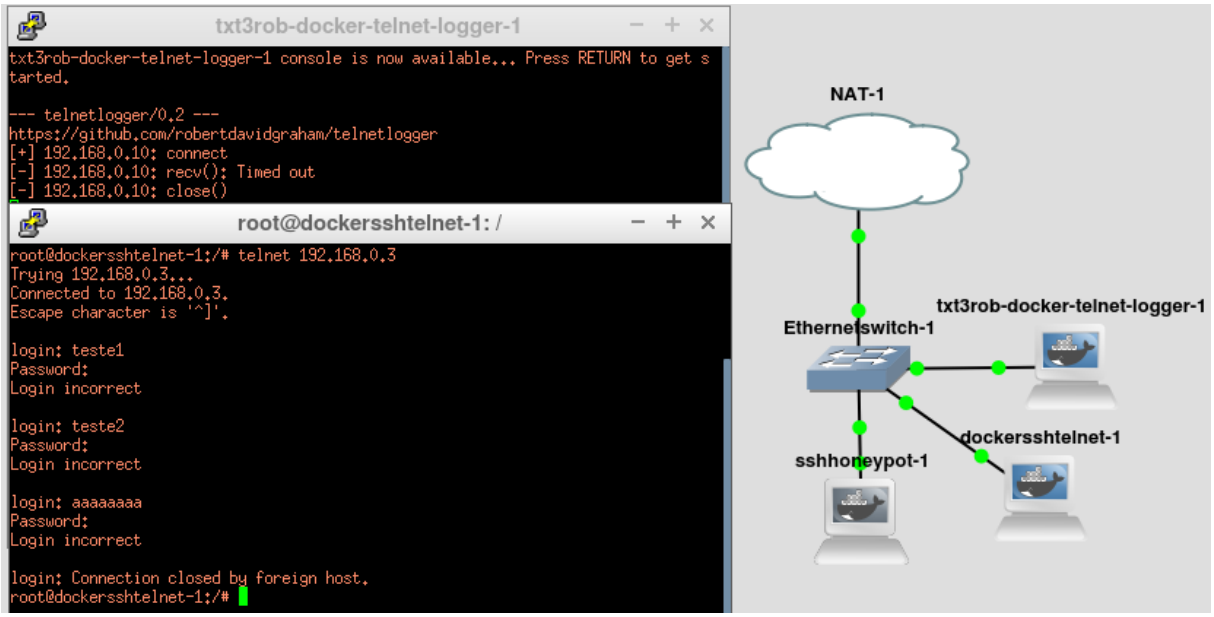
- # cd docker-ssh-honey/: entra na pasta que foi baixada para o Raspberry Pi;

- # docker build . -t local:ssh-honepot: dá um *build* no *container*, fazendo com que ele vire uma imagem e possa ser executado pelo Docker.

Após essa instalação, basta utilizar o comando “# docker run -itd --name ssh-honeypot -p 22:22 local:ssh-honepot” e começar a utilizar o *honeypot* de SSH.

Para deixar o *honeypot* em Docker ainda mais completo, foi baixado e instalado na máquina virtual o *container* Telnet-logger, que funciona de uma forma bem semelhante ao SSH Honeypot, porém, simula um serviço de Telnet. O serviço foi testado na máquina virtual utilizando o GNS3, simulando uma topologia e realizando alguns acessos por meio de outro *host*.

Figura 19 – Telnet-logger sendo executado juntamente ao GNS3 na máquina virtual.



Fonte: Do autor (2019).

O Telnet-logger funcionou na máquina virtual perfeitamente, sendo possível visualizar os *logs* do endereço IP de quem está tentando realizar o acesso e o usuário e senha que foram inseridos. Podemos visualizar o descrito na Figura 20, porém, ao instalar esse *container* no Raspberry Pi, o mesmo não funcionou, informando a mensagem de erro “standard_init_linux.go:211: exec user process caused exec format error". Ou seja, esse *container* é compatível apenas com

arquitetura x86/64 e não consegue ser executado em ARM, que é a arquitetura utilizada pelo Raspberry Pi.

Figura 20 – Log de saída do Telnet-logger.

```
lubuntu@lubuntu-VirtualBox:~$ docker exec -i -t eb6478a48e13 /bin/ash
/telnetlogger # cat passwds.txt
teste1 testesenha1
teste2 testesenha2
aaaaaaaa bbbbbbb
/telnetlogger # cat ips.txt
192.168.0.10
192.168.0.10
192.168.0.10
/telnetlogger #
```

Fonte: Do autor (2019).

Vários *containers* foram testados, porém, são escassos os que possuem o propósito de atuar como um *honeypot* e que funcionam na arquitetura ARM, por isso, o *container* que apresentou o melhor funcionamento e que foi utilizado para os testes foi o SSH Honeypot.

4.3.3 HoneyD

Juntamente ao Ubuntu MATE, foi instalado o HoneyD, um *honeypot* que é utilizado no honeyTARG Honeynet Project e possibilita a criação de novas máquinas virtuais na rede, com serviços falsos através de *scripts* e *banners* personalizados. Sua finalidade é mascarar essas máquinas virtuais e transformá-las em qualquer tipo de sistema operacional e hardware.

O HoneyD possui uma funcionalidade bem semelhante ao Dionaea Honeypot, porém, não foi viável realizar os testes com o Dionaea devido à nova versão necessitar de virtualização de outras máquinas. O Raspberry Pi não consegue lidar muito bem com virtualizações reais devido ao seu hardware e, por isso, optou-se pela utilização do HoneyD, que consegue virtualizar as máquinas de uma forma eficiente, sem consumir muitos recursos do Raspberry Pi.

A instalação e configuração do HoneyD é bastante complexa, pois ele necessita da instalação de várias dependências e, necessita de *scripts* para que os

serviços funcionem. Para instalar o HoneyD foram utilizados os comandos a seguir (em modo administrador):

- # apt-get update
- # apt-get install libdnet make libpng-dev git
- # apt-get install libevent-dev libdumbnet-dev libpcap-dev libpcre3-dev libedit-dev bison flex libtool automake
- # cd /usr/src/
- # git clone https://github.com/DataSoft/Honeyd.git
- # cd Honeyd
- # ./autogen.sh
- # ./configure
- # make
- # make install

Após a instalação do HoneyD, também foi instalado o FARPD (Fake ARP Daemon), que será o responsável por deixar as máquinas visíveis na rede para os outros usuários e atacantes. O FARPD escuta a interface de rede e redireciona todas as solicitações de ARP para o Daemon do HoneyD. Ele pode ser instalado e executado com os comandos listados abaixo:

- # apt-get install farpd: instala o FARPD;
- # farpd -d -i wlan0 192.168.0.7-192.168.0.100: executa o FARPD como um processo não Daemon, que fica escutando as requisições de ARP na interface wlan0 e redireciona para o range de IP 192.168.0.7-192.168.0.100 do HoneyD.

Dessa forma, o HoneyD está quase pronto para ser utilizado, sendo o próximo passo realizar a configuração das máquinas virtuais. A configuração das máquinas é feita de acordo com a Figura 21.

Figura 21 – Configuração das máquinas a serem virtualizadas pelo HoneyD.

```
create windows
set windows personality "OpenBSD 4.0 (x86)"
add windows tcp port 80 "scripts/web.sh"
add windows udp port 53 open
set windows ethernet "dell"
bind 192.168.0.108 windows

create windows2
set windows2 personality "Microsoft Windows XP Professional SP3"
add windows2 tcp port 80 "scripts/web.sh"
add windows2 udp port 53 open
set windows2 ethernet "realtek"
bind 192.168.0.109 windows2
```

Fonte: Do autor (2019).

Na configuração acima, é possível visualizar que foram criadas duas máquinas virtuais com as portas 80 e 53 abertas e com suas identidades alteradas para OpenBSD e Microsoft Windows XP. Também foi alterado o fabricante das *interfaces* de rede para Dell e Realtek. Tendo feito essas alterações, bastou utilizar o comando descrito a seguir para executar o HoneyD:

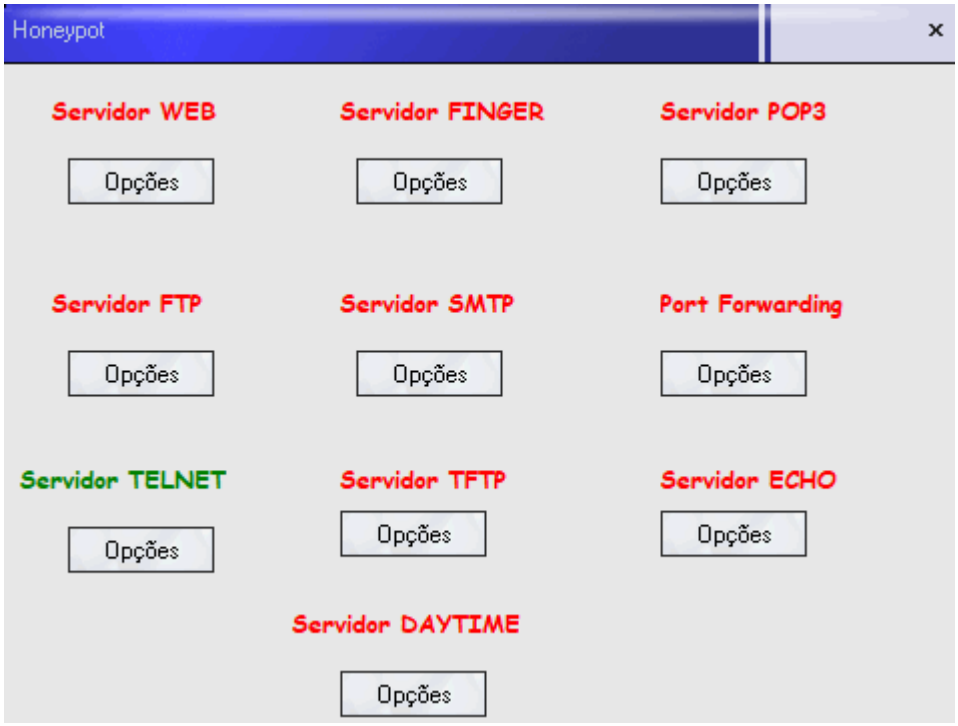
- # honeyd -d -f /usr/src/Honeyd/configuracao -i wlan0: executa o HoneyD em primeiro plano, utilizando o arquivo de configuração criado na Figura 21, tendo como saída e entrada de dados à interface wlan0.

4.3.4 Valhala Honeypot

O Valhala Honeypot não possui instalação, é apenas um programa executável, mas em contrapartida é o *honeypot* mais fácil de se utilizar e o mais completo. Sua única grande desvantagem é a compatibilidade restrita ao Windows, e devido a isso pode ser atacado facilmente.

O Valhala possui muitas funcionalidades, como: servidor WEB, FINGER, POP3, FTP, SMTP, TELNET, TFTP, ECHO, DAYTIME e redirecionamento de portas. Além disso, o administrador do sistema pode utilizar mais de uma função e monitorar todos os *logs* pela tela principal do Valhala Honeypot. O administrador também pode ser alertado via *e-mail* ou configurar para que os *logs* sejam enviados para algum servidor remoto.

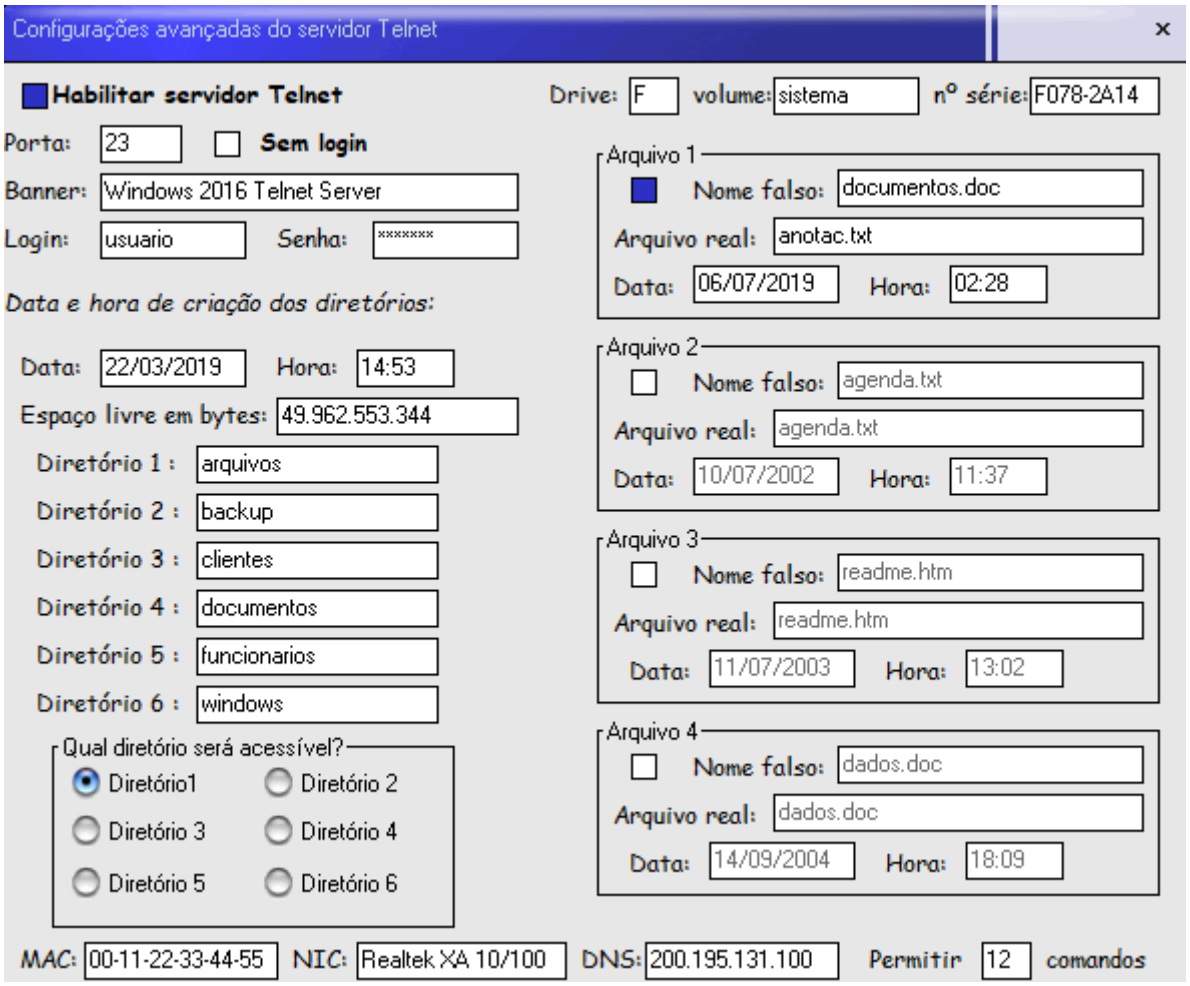
Figura 22 – *Interface* e funcionalidades do Valhala Honeypot.



Fonte: Do autor (2019).

A opção de redirecionamento de portas ou *port forwarding* é bem interessante, pois, o *honeypot* redireciona a porta 80 do servidor para um *site* qualquer, ou seja, é possível redirecionar para uma página de *login* em HTTP e, com isso, é possível capturar o nome e senha de quem tentar acessar a página pela porta 80 do próprio servidor. O Valhala também possui um servidor falso de Telnet com várias configurações, possibilitando ao administrador do sistema alterar o *banner* do servidor de Telnet. Caso alguém consiga acessar o servidor Telnet falso, é possível simular diretórios e arquivos, para que o *cracker* pense que está acessando um servidor real.

Figura 23 – Configurações do servidor Telnet falso.



Fonte: Do autor (2019).

As funções de servidores FTP e TFTP também podem ser utilizadas como servidores reais, bastando utilizar a opção de “serviço real” ou “acesso total”, a depender do serviço que será escolhido. Porém, devem ser tomados alguns cuidados, pois, o intuito do *honeypot* é distrair o atacante e capturar suas informações e, com esses serviços de modo real, o usuário ou administrador do sistema pode estar correndo riscos de ser atacado.

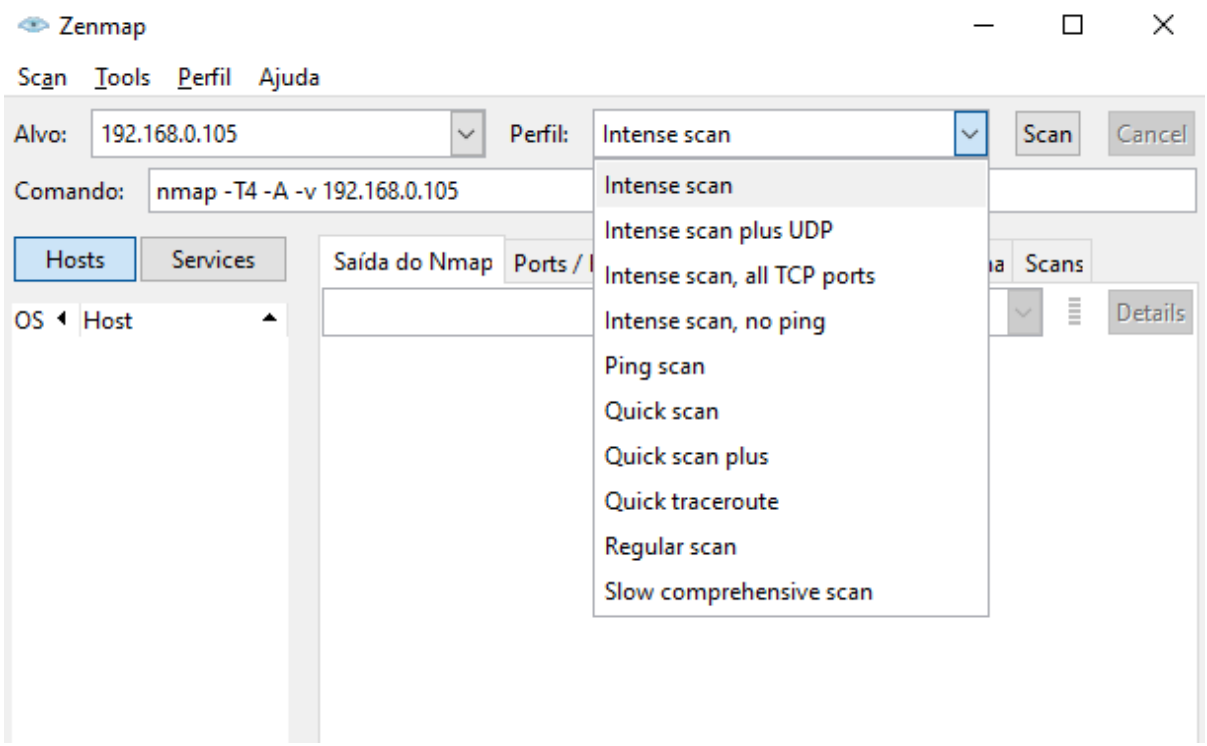
4.4 Ferramentas utilizadas para testar os *honeypots*

Para realizar os ataques e os testes, foram utilizadas algumas ferramentas em um ambiente Linux e em um ambiente Windows. Essas ferramentas foram

essenciais para cumprimento dos requisitos e análise da funcionalidade dos *honeypots*.

Para realizar o *scan* de portas, foram utilizados os softwares nmap e o Advanced Port Scanner no sistema operacional Windows, visando analisar se existe algum dispositivo *online* na faixa de IPs selecionados ou em um IP específico. Além disso, podemos analisar todas as portas que estão sendo utilizadas em cada dispositivo varrido, juntamente com o serviço que está sendo executado em cada porta, informando a versão e se essa porta está aberta. Ambos os programas não informam apenas as portas, mas também conseguem extrair informações dos dispositivos, tais como endereço MAC da *interface* de rede, o sistema operacional que está em execução no dispositivo, a quantidade de pulos (saltos) que são feitos para chegar ao dispositivo varrido através da origem (atacante), nome do dispositivo e fabricante da *interface* de rede.

Figura 24 – Interface do nmap (Zenmap) para Windows.



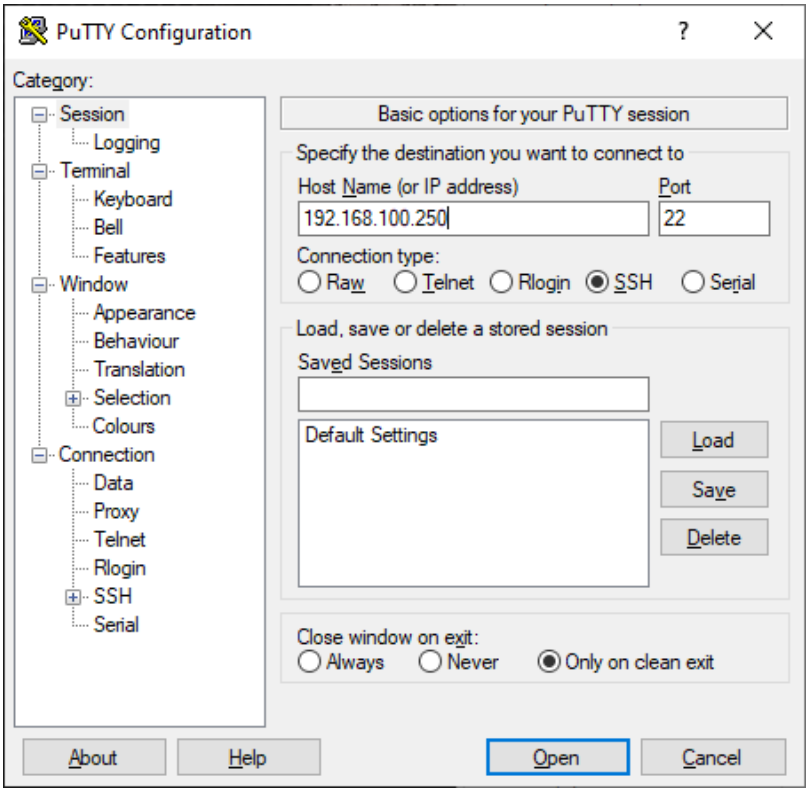
Fonte: Do autor (2019).

O primeiro passo de um atacante ao querer acessar um sistema é feito pelo nmap e, após isso, ele busca falhas nas versões dos serviços listados ou tenta acessar um serviço como SSH e Telnet pelo método de força bruta.

Como método de força bruta, foi utilizado o software Hydra em uma máquina virtual com sistema Linux Lubuntu. O Hydra é compatível com os seguintes serviços e protocolos: VNC, CVS, SNMP, FTP, LDAP3, LDAP2, MSSQL, PostgreSQL, SSH, HTTP/HTTPS e Telnet. Dessa forma, o Hydra permite que sejam feitos vários acessos simultâneos através de uma *wordlist*, ou seja, com o Hydra é possível forçar múltiplos acessos com nomes de usuários e senhas aleatórias, fazendo com que o acesso seja feito por força bruta, ou que o dispositivo seja derrubado, sendo este um ataque de DoS.

Para realizar os acessos básicos e testar a iteratividade dos *honeypot*s, além de sua funcionalidade, foram utilizadas as ferramentas Putty e Filezilla FTP Client. Ambas as ferramentas permitem acesso aos respectivos serviços. O Putty é um cliente de acesso via Serial, SSH, Telnet, RAW e Rlogin.

Figura 25 – *Interface* de configuração da ferramenta Putty.



Fonte: Do autor (2019).

Sua *interface* é de fácil configuração, bastando selecionar o tipo de serviço e definir a sua porta, ou velocidade, no caso de algum acesso via comunicação serial. O acesso pode ser feito de forma simplificada e sem precisar instalar qualquer tipo

de cliente para esses serviços no Windows, pois o Putty é apenas um programa executável, dispensando sua instalação.

4.5 Cenários e topologias utilizadas

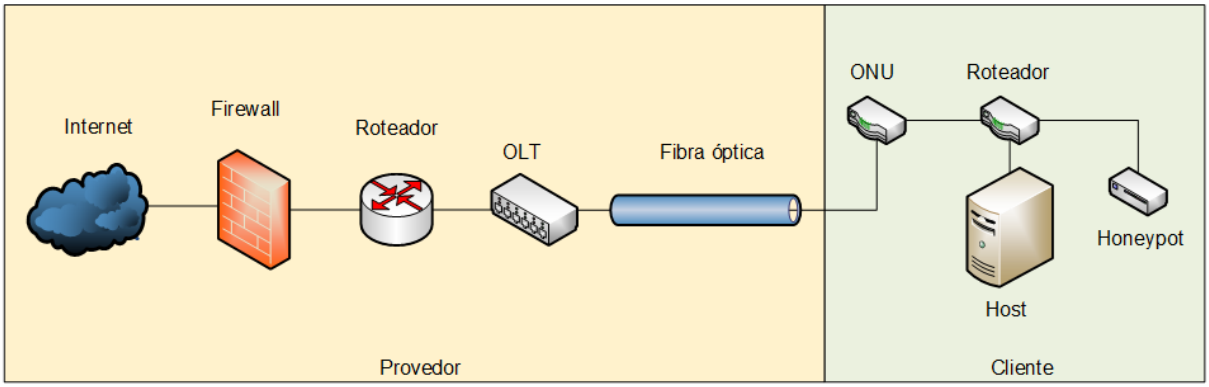
Para a execução dos testes, o *honeypot* foi colocado em três cenários, cada um possuindo uma topologia e níveis de segurança totalmente distintos. Os cenários utilizados foram escolhidos devido à aplicação e ao modo em que o *honeypot* pode ser utilizado. Sendo assim, o *honeypot* ficou dentro de uma rede LAN residencial e de uma rede empresarial e, por fim, em uma rede com IP público sem qualquer tipo de *firewall.*

O primeiro cenário utilizado foi o cenário residencial, que permite que sejam feitos os testes básicos, com o acesso de poucos dispositivos ao *honeypot*, sendo assim, não é um cenário viável para realizar um teste final, mas em contrapartida, é um cenário excelente para a configuração e preparação do *honeypot*.

O cenário residencial possui uma topologia muito “simples”, tendo em vista que se trata de uma ONU (Optical Network Unit) que recebe o *link* de fibra óptica do provedor de Internet. Essa ONU transmite o sinal para um roteador e esse roteador é o responsável por interligar o *honeypot* aos outros dispositivos. Porém, atrás dessa ONU há uma OLT (Optical Line Terminal), que fica no provedor de Internet e é responsável por transmitir o sinal para toda rede óptica passiva.

Juntamente com a OLT, o provedor possui diversos roteadores e *firewalls* que bloqueiam a conexão do usuário final, fazendo com que a rede do usuário final possua um nível de segurança bem elevado, pois atua com um IP privado do próprio provedor, não possuindo acesso direto à Internet.

Figura 26 – Rede residencial utilizada para teste do *honeypot*

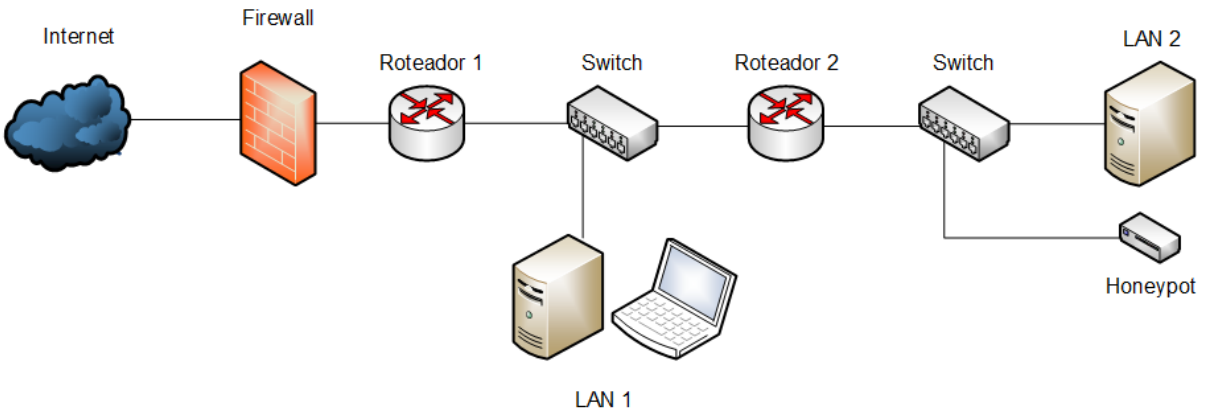


Fonte: Do autor (2019).

Nesse cenário é impossível mensurar a viabilidade do *honeypot*, tendo em vista que é muito difícil que um *cracker* consiga visualizar o *honeypot* sem que antes tenha acesso à toda rede do provedor de serviços de Internet. A única maneira de ser atacado, nesse caso, seria uma rede *wireless* com uma criptografia ou senha frágil, assim, o atacante acessaria a rede e, após estar dentro da mesma, poderia executar os ataques.

Como segundo cenário, foi escolhida uma rede empresarial, que possui IP Público com saída diretamente para a Internet, além de VPNs entre as filiais, *firewall* e uma rede separada para testes de equipamentos de telecomunicações.

Figura 27 – Rede empresarial utilizada para teste do *honeypot.*



Fonte: Do autor (2019).

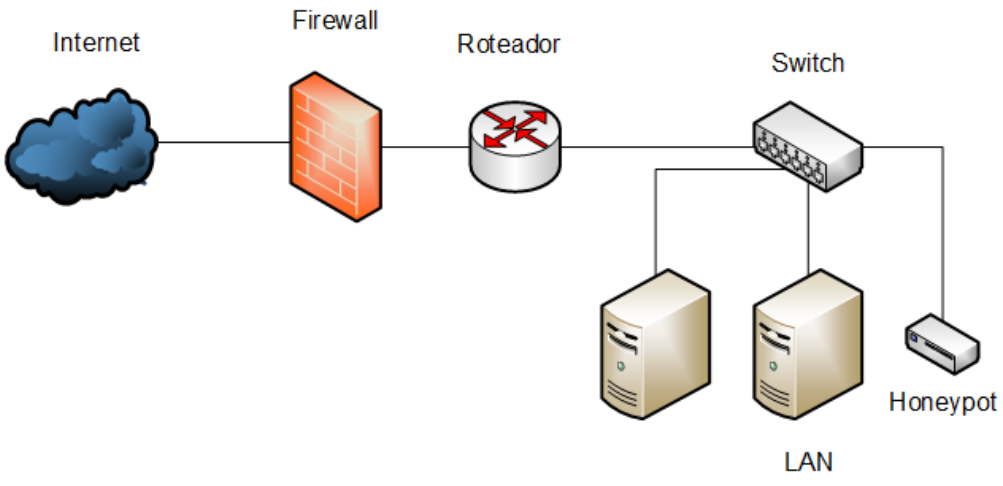
A conexão do *honeypot* foi feita na rede de testes, em um *switch* instalado após o roteador. Com essa conexão foi possível fazer um redirecionamento de IP,

sendo que os dispositivos da LAN 1 conseguiram ter acesso à LAN 2 e ao *honeypot*. Foi possível, a partir disso, simular o uso do *honeypot* em uma rede empresarial, com maior chance de ser atacada por terceiros ou por eventuais funcionários mal-intencionados, conhecidos como *insiders.*

O último cenário simulado foi na rede da UNIVATES, na sala 404 do prédio 11, com um IP público e acesso externo totalmente livre, possibilitando analisar e verificar algum tipo de ataque ou acessos indesejados. A sala é composta por um *rack* que contém um roteador com a chegada do *link* e um *switch* que distribui o acesso à Internet para os computadores do laboratório. A faixa de IP utilizada nessa sala é a mesmo utilizada no *datacenter* experimental da sala 415, também do prédio 11, onde há uma *blade* PowerEdge M520 com 8 lâminas que está disponível para os alunos realizarem experimentos e que já foi alvo de ataques externos anteriormente.

Esse cenário é o ideal para verificar a segurança do *honeypot* e analisar a tentativa de ataques reais. Para realizar o teste nesse cenário foi necessário utilizar o *honeypot* com maior nível de segurança, para não comprometer outros dispositivos na rede da instituição.

Figura 28 – Topologia do cenário campus utilizada para teste do *honeypot*.



Fonte: Do autor (2019).

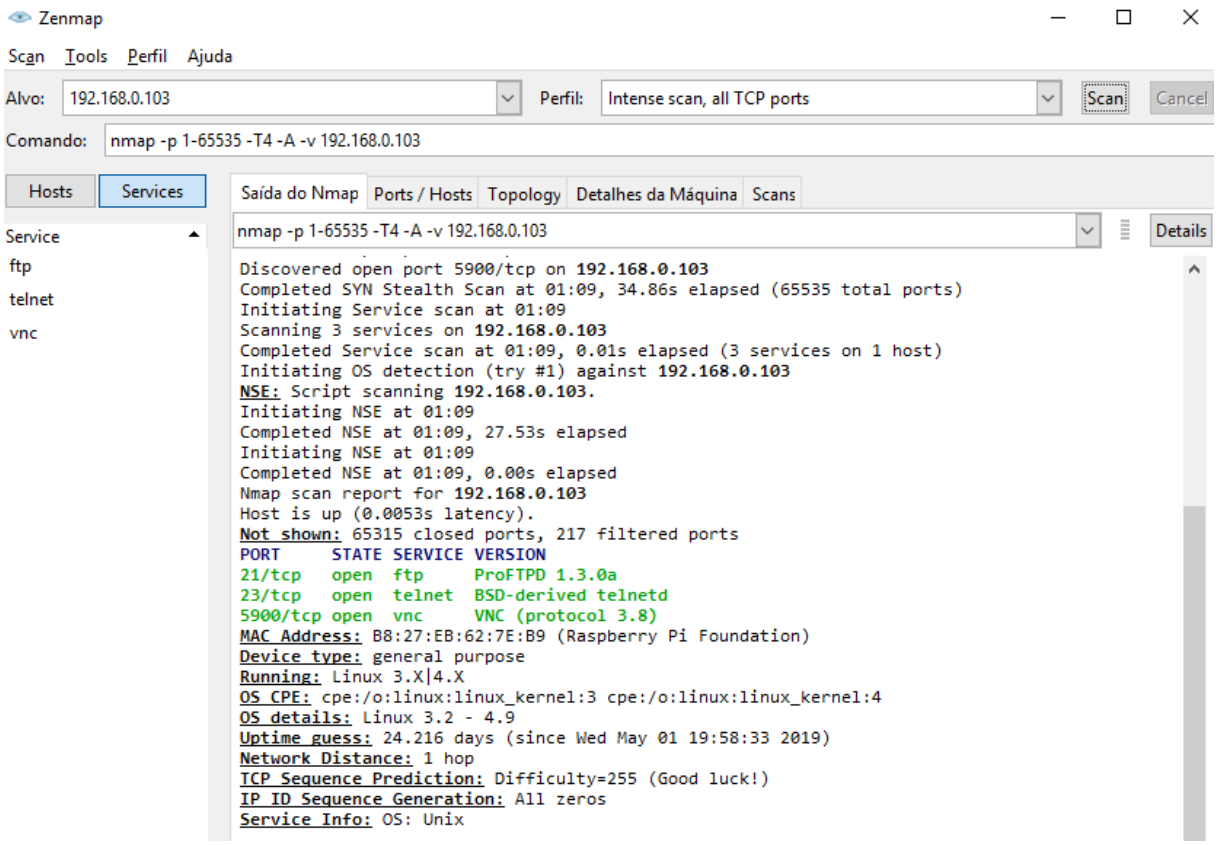
5 TESTES E RESULTADOS

Neste capítulo são demonstrados os testes realizados com os diferentes tipos de *honeypots,* nos diferentes cenários e em cada topologia. Também serão analisados os resultados, além da viabilidade em utilizar o Raspberry Pi para essa função, a facilidade na utilização do *honeypot* para fazer os testes, os níveis de interação, a melhor aplicação de cada *honeypot* e qual possui maior eficiência e segurança.

5.1 Primeiro Experimento – Cenário residencial com Raspbian e HoneyPi

O primeiro teste realizado foi um *port scan*, utilizando a ferramenta nmap no endereço IP do *honeypot* (192.168.0.103). O nmap conseguiu trazer dados de quais portas estavam abertas e quais serviços estavam sendo executados. Também conseguiu verificar qual o sistema operacional estava rodando nesse “servidor”, além de identificar por métodos de *fingerprinting* qual é o fabricante da *interface* utilizada. O resultado é mostrado na Figura 29.

Figura 29 - Resultados da execução do nmap no HoneyPi.



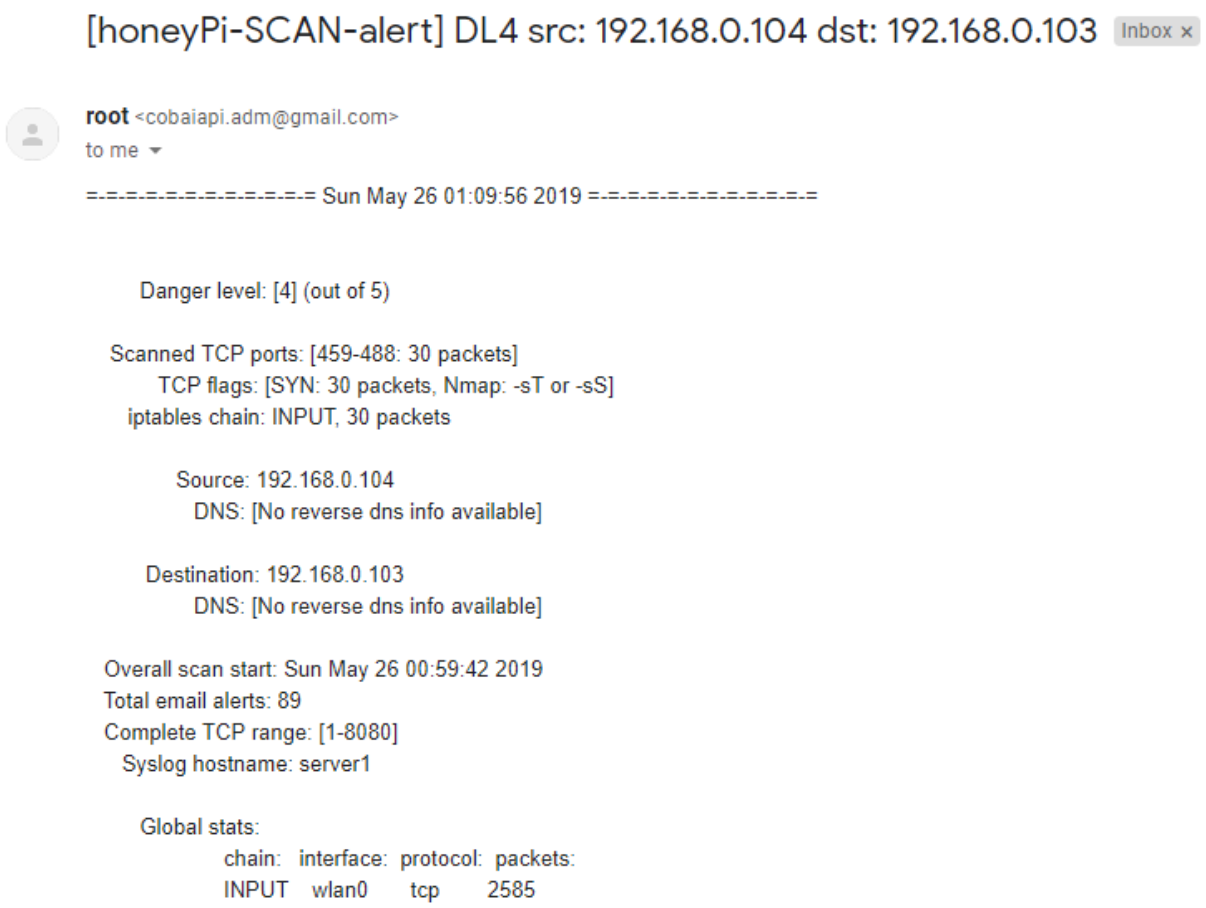
Fonte: Do autor (2019).

O segundo teste foi apenas um reforço deste primeiro, no qual foi utilizada uma outra ferramenta de *scan* chamada Advanced Port Scanner. Com ela obteve-se o mesmo resultado do teste anterior, e a partir dela também foi possível descobrir que o “servidor” se tratava de um Raspberry Pi.

Esse envio de informações do fabricante da placa de rede pode ser mascarado por meio de alterações na pilha TCP/IP do Raspberry Pi. Após essa detecção, foram testados alguns métodos como IPlog, Stealth Patch e IP Personality, que foram métodos eficientes para mascarar o sistema operacional, mas o fabricante da placa ainda aparecia visível. Levando em consideração que o intuito do presente trabalho é verificar a viabilidade do Raspberry Pi como um *honeypot,* também levando-se em consideração o seu desempenho, a pilha TCP foi mantida como original para que não houvesse nenhum problema no desempenho da placa devido a conexões da *interface* de rede.

Ambos os testes se mostraram muito efetivos, visto que o *honeypot* de fato avisou o administrador do sistema da ação tomada pelo *cracker*, enviando um *e-mail* e alertando sobre o ocorrido, como mostra a Figura 30.

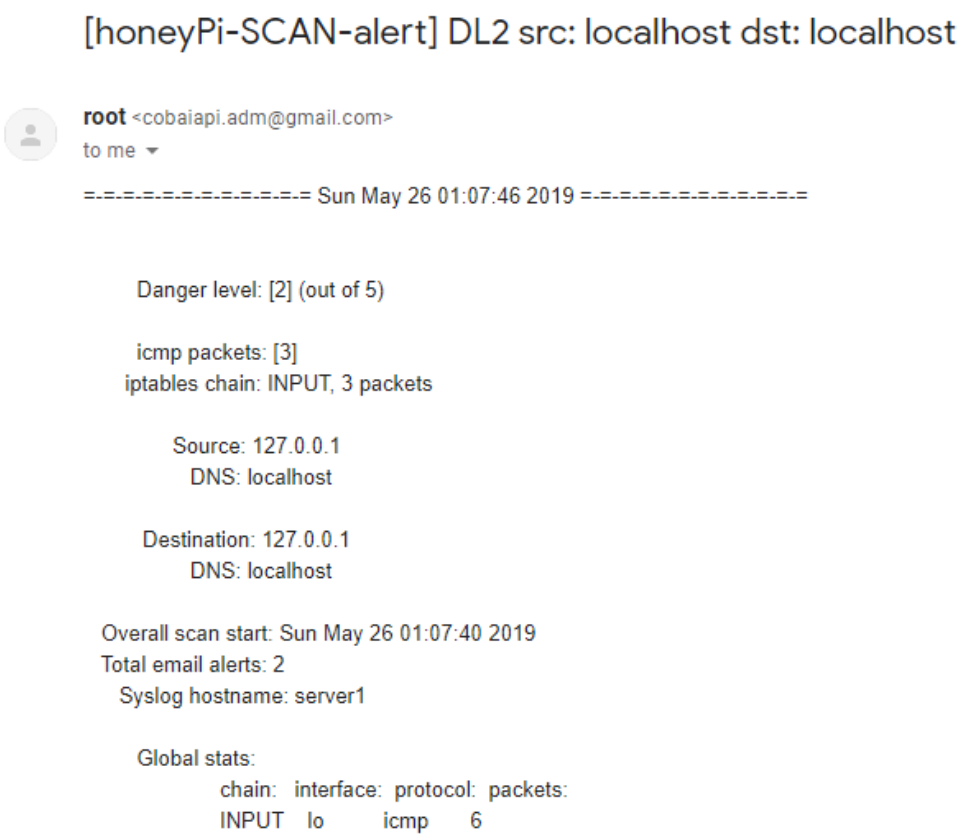
Figura 30 - Mensagem de aviso do HoneyPi para o administrador do sistema.



Fonte: Do autor (2019).

O último teste realizado foi tentar acessar o “servidor” via Telnet. Dessa maneira, o *cracker* logo iria perceber que se tratava de um *honeypot*, pois o serviço não solicitou nenhuma autenticação como de costume e apenas apresentou a falha para se conectar. Dessa vez, o *honeypot* se mostrou menos efetivo, porque conseguiu detectar o ataque, mas sua origem constava como o próprio usuário local, ou seja, não conseguindo localizar o endereço IP do atacante, conforme apresentado na Figura 31.

Figura 31 – Mensagem de aviso após o acesso Telnet, com o IP não reconhecido.



Fonte: Do autor (2019).

5.1.1 Análise dos resultados do Primeiro Experimento

Devido aos testes básicos realizados no Primeiro Experimento e seus resultados, foi inviável avançar com o HoneyPi em cenários mais complexos e perigosos devido à sua simplicidade de funcionamento e limitação.

Com os testes foi possível analisar que o HoneyPi é um *honeypot* de baixa interatividade, que apenas simula serviços e portas abertas. Porém, caso o *cracker* tente realizar algum ataque acessando esses serviços, facilmente ele detectará que se trata de um *honeypot*. Outro ponto negativo do HoneyPi é a sua configuração de alertas. Essa configuração é um processo complicado para um usuário leigo, que necessitaria ter algum conhecimento no sistema Linux e, mesmo assim, caso queira realizar alguma alteração nas configurações, precisaria buscar os arquivos que estão dispersos em todo o sistema, já que o HoneyPi não cria apenas um diretório com toda a configuração.

O HoneyPi possui um ponto muito forte que é o seu IDS, que pode ser excelente para utilizar junto a servidores dentro de uma DMZ, ou até mesmo, dentro da rede LAN empresarial. Se o *cracker* realizar um *port scanner* em uma determinada faixa de IPs que passe por esse *honeypot,* o mesmo consegue avisar o administrador do sistema com extrema eficiência.

Para essa função, o Raspberry Pi teve um excelente comportamento e, seu hardware realizou a atividade sem apresentar problemas com uso de memória e processamento. Além disso, pode ser utilizado um cartão de memória classe 4 ou 10 com uma velocidade de leitura e escrita baixa, o que não influenciará no desempenho final do *honeypot.*

Uma sugestão de melhoria a se fazer com o HoneyPi seria a inserção de *scripts* para melhorar a interatividade com indivíduos que tentarem realizar um acesso ao sistema, ou até mesmo, desabilitar essa função de portas e utilizá-lo como um dispositivo IDS, o qual irá apenas alertar os *port scanners* ao administrador do sistema e utilizá-lo juntamente com outro *honeypot.*

5.2 Segundo Experimento – Cenário residencial com Windows e Valhala Honeypot

No Segundo Experimento, foi utilizada a mesma topologia do experimento anterior, porém, com um sistema operacional e um *honeypot* diferente.

O primeiro teste com o Valhala Honeypot e o Windows também foi feito por um *port scan,* em que o *honeypot* no momento estava com as portas 22, 23 e 80 abertas, rodando os seus respectivos serviços como FTP, Telnet e HTTP. Dessa forma, ao fazer o *scan* das portas, foi possível verificar que o Valhala Honeypot além de liberar as portas escolhidas, também simulou algumas portas abertas como *backdoors* de *trojans* do tipo Netbus e Subseven, entre outras. O resultado pode ser visualizado na Figura 32.

Figura 32 – Resultados da execução do nmap no Valhala Honeypot.

```
PORT      STATE SERVICE       VERSION
21/tcp    open  ftp
| fingerprint-strings:
|   GenericLines:
|     220 220 Wu-ftpd 1.7.0
|     command not understood.
|     command not understood.
|   Help, SMBProgNeg:
|     421 Too many users connected.
|   NULL:
|     220 220 Wu-ftpd 1.7.0
|   SSLSessionReq:
|     220 220 Wu-ftpd 1.7.0
|_    command not understood.
23/tcp    open  backdoor      OptixPro backdoor (**BACKDOOR**)
80/tcp    open  http          Apache httpd
|_http-favicon: Unknown favicon MD5: 1E387F492989A173263C68F8E6BA89C2
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
1243/tcp  open  tcpwrapped
2583/tcp  open  tcpwrapped
5742/tcp  open  tcpwrapped
7680/tcp  open  pando-pub?
12345/tcp open  tcpwrapped
20034/tcp open  tcpwrapped
31785/tcp open  tcpwrapped
49666/tcp open  msrpc         Microsoft Windows RPC
54321/tcp open  tcpwrapped
```

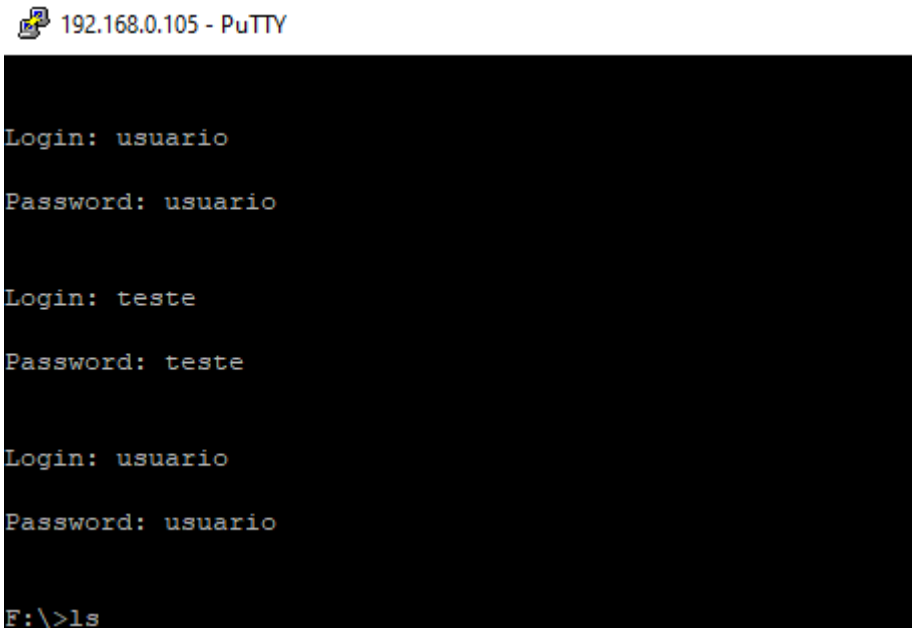
Fonte: Do autor (2019).

Após o primeiro teste com o nmap, foi executado novamente o Advanced Port Scanner, relatando o mesmo que o nmap. Porém, em ambos os testes não foi possível visualizar o sistema operacional que estava sendo executado, sendo apresentada apenas uma tabela com a probabilidade de qual sistema operacional aquele dispositivo poderia estar executando. Também, não foi possível visualizar o fabricante da *interface* de rede, mas foi possível visualizar o nome do dispositivo (que aparece nos dispositivos que possuem o sistema operacional da Microsoft).

Outro detalhe do Valhala Honeypot é que este não mostra o *banner* de serviço nas portas, aparecendo como um *backdoor*, sem a descrição de “Windows 2016 Telnet Server” conforme configurado anteriormente. Em contrapartida, ao executar o teste de acesso ao Telnet com a ferramenta Putty, o Valhala se saiu muito bem. A ferramenta consegue simular totalmente um acesso Telnet, sendo possível colocar um nome de usuário e senha, permitindo que o atacante consiga

“invadir” o sistema, tendo acesso a pastas e arquivos fictícios criados pelo administrador do dispositivo.

Figura 33 – Acesso via Telnet no Valhala Honeypot.

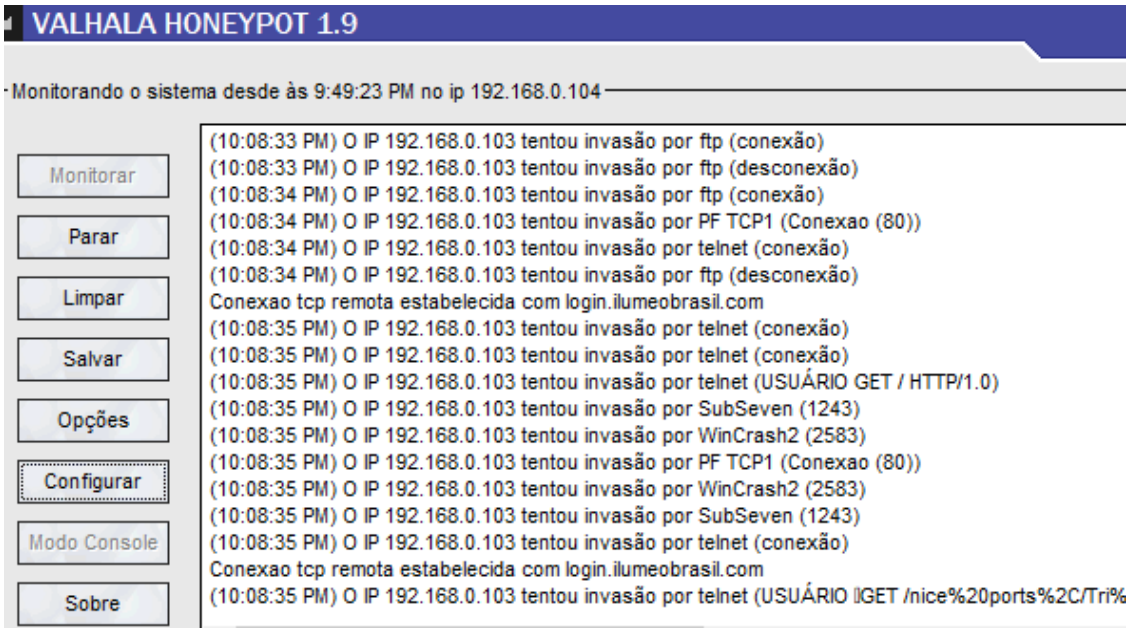


Fonte: Do autor (2019).

Outro teste realizado foi o acesso via FTP. O Valhala Honeypot possui um modo falso e um modo verdadeiro de servidor FTP e ambos funcionam. No modo falso, acontece o mesmo ocorrido no acesso Telnet, quando o usuário fica limitado a uma parte do sistema com arquivos falsos e não consegue sair ou navegar por outras pastas do sistema. Para realizar o acesso ao *honeypot* neste teste, foi utilizado o FileZilla FTP Client.

Porém, ao realizar esses acessos, foi possível verificar que o aviso por *e-mail* do Valhala Honeypot não funcionou, mesmo com testes realizados nas versões 1.8 e 1.9 do aplicativo. Ambas as versões não obtiveram nenhum sucesso em relação aos *logs* dos acessos via *e-mail.* A única maneira de visualizar os *logs* do Valhala Honeypot é por sua tela principal, que consegue listar todas as informações de tentativas de *scan* e acessos. O *log* do Valhala é mostrado na Figura 34.

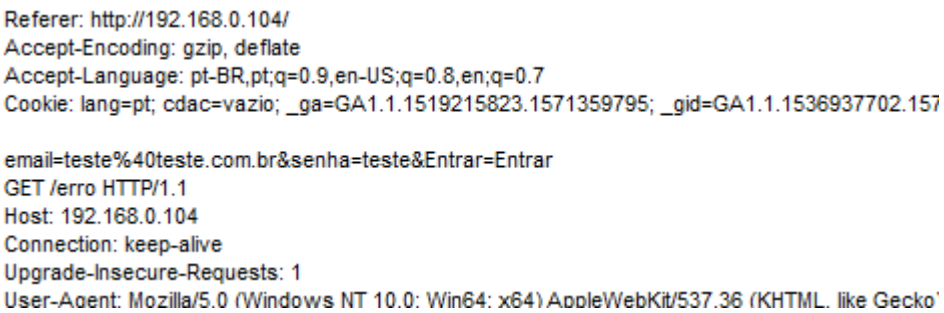
Figura 34 – Logs na *interface* principal do Valhala Honeypot.



Fonte: Do autor (2019).

O último teste realizado no Valhala Honeypot foi o teste de redirecionamento de porta, ou seja, a porta 80 do Raspberry Pi foi redirecionada para um *site*, fazendo com que toda vez que o invasor tentasse realizar um acesso pela *interface web*, fosse redirecionado a um *site* predefinido pelo administrador do sistema. Para o teste, foi utilizado um site com *login* no protocolo HTTP, fazendo com que os dados do atacante fossem capturados e informados no *log* do Valhala, conforme podemos visualizar na Figura 35. O mesmo teste foi feito em uma página HTTPS, porém, como se trata de um protocolo seguro não foi possível visualizar os usuários e senhas enviados para o servidor.

Figura 35 – *Log* de acesso HTTP com usuário e senha.



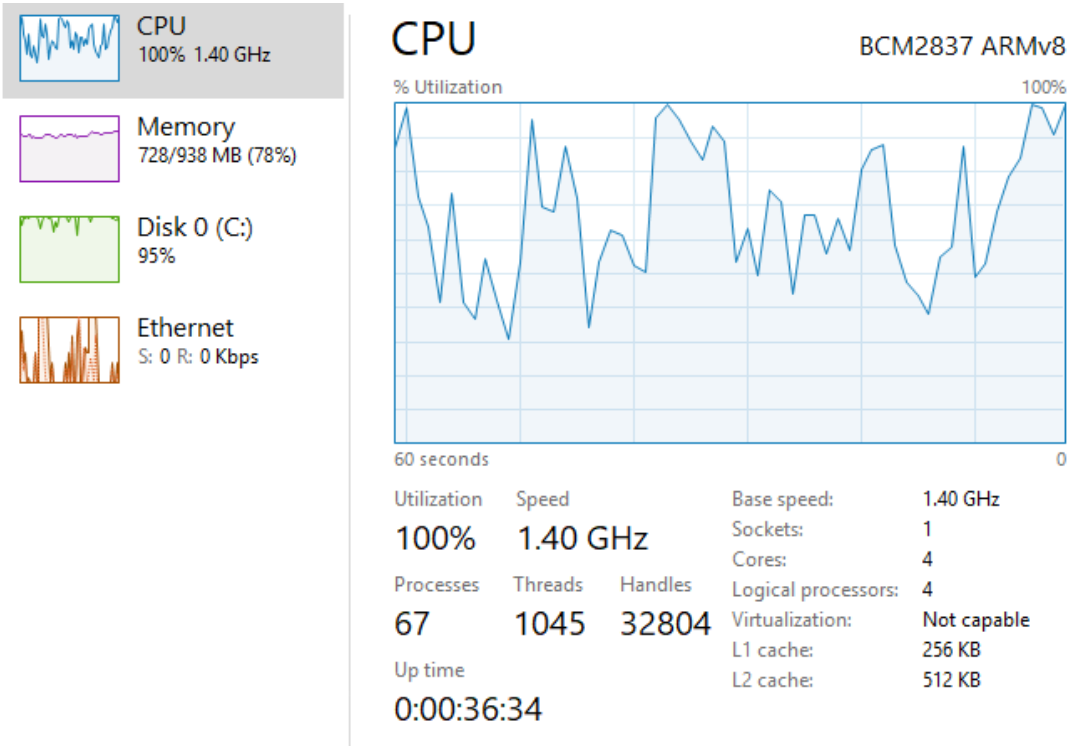
Fonte: Do autor (2019).

5.2.1 Análise dos resultados do Segundo Experimento

Com os testes foi possível analisar que o Valhala Honeypot possui um nível de interatividade com o atacante muito maior que o HoneyPi, além disso, mostrou-se muito mais completo, contando com vários serviços e até mesmo com a possibilidade de utilizá-los como falsos ou reais. Mesmo o Valhala sendo mais completo, a sua opção de enviar avisos por *e-mail* não funcionou, tornando-se um ponto negativo para o *honeypot.*

Outro ponto negativo e crucial, para não se utilizar o Valhala Honeypot em outros cenários, foi o consumo de hardware, que ocasionou, em todos os testes realizados, o uso de disco (cartão SD) do Raspberry Pi em 100% na maior parte do tempo. Isso ocorreu mesmo com a utilização de um cartão de memória Sandisk Extreme Pro com velocidade de leitura e escrita nominal de 95 MB/s, além do elevado uso de memória RAM e o excessivo uso de processamento. Com esse uso elevado, foi possível detectar que o Raspberry Pi 3 B+ consegue usufruir de uma velocidade de leitura e escrita do cartão de até 22 MB/s (Figura 36).

Figura 36 – Consumo de hardware com os testes do Valhala Honeypot.



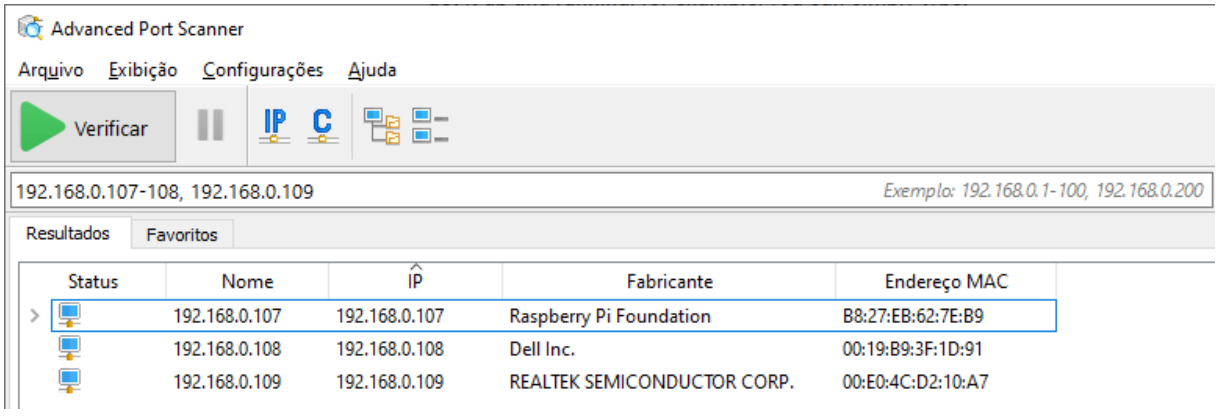
Fonte: Do autor (2019).

Mesmo com esses pontos negativos, o Valhala Honeypot possui outros dois pontos fortes, sua instalação e configuração. Como se trata apenas de um executável e possui uma interface no padrão Windows, qualquer usuário com um conhecimento intermediário em Informática consegue utilizá-lo. Além disso, o Valhala teria uma boa aplicação em dispositivos de uma pequena rede empresarial, para verificar se essa rede está de fato segura. Porém, em hipótese alguma é recomendado o uso do Valhala em dispositivos que não estejam protegidos por bom *firewall*, já que o Windows possui várias falhas e não é um bom sistema para hospedar um *honeypot* que atue antes do *firewall.*

5.3 Terceiro Experimento – Cenário residencial com Ubuntu Mate e HoneyD

O primeiro teste neste cenário foi feito com o Advanced Port Scanner, sendo que com ele foi possível verificar que o HoneyD consegue simular as máquinas virtuais como se fossem máquinas físicas reais. O IP do Raspberry é o 192.168.0.107, mas o HoneyD, juntamente com o FARPD, consegue simular os dispositivos com endereços IP terminados em 108 e 109, conforme podemos visualizar na Figura 37.

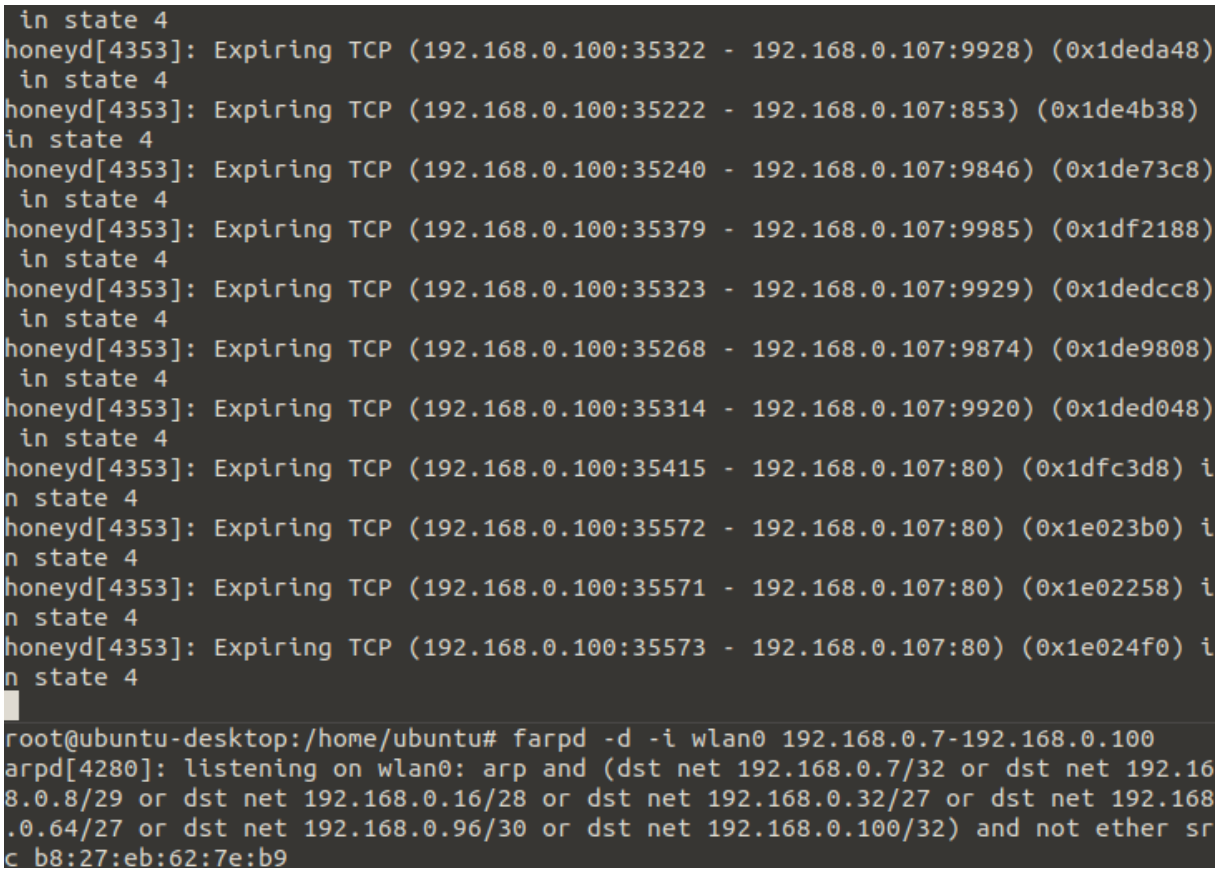
Figura 37 – *Port scan* realizado na faixa de endereço IP do *honeypot.*



Fonte: Do autor (2019).

O segundo teste realizado foi com o nmap e, o resultado não foi diferente. Neste caso, podemos verificar que o nmap também foi enganado e descobriu os dispositivos falsos com *interfaces* de rede Dell e Realtek. Na Figura 38 podemos visualizar o *log* do *port scan* no HoneyD.

Figura 38 – *Log* do HoneyD junto com o FARPD em execução na interface wlan0.



Fonte: Do autor (2019).

Infelizmente, os dois e únicos testes realizados foram o *port scan* com o Advanced Port Scanner e nmap. O motivo de não realizar mais testes foi devido ao fato de que não funcionaram os *scripts* de acesso *web*, SSH e Telnet com o HoneyD.

5.3.1 Análise dos resultados do Terceiro Experimento

Com os testes de *port scan*, foi possível analisar que o HoneyD possui um excelente funcionamento no quesito de criar máquinas falsas, porém, não foi possível adicionar serviços ou abrir as portas dessas máquinas. O mais interessante é que pelo HoneyD o administrador do sistema consegue enganar o *fingerprinting* das ferramentas de *port scan.* Desse modo, é possível colocar qualquer nome de um fabricante nas *interfaces* de rede e, dessa forma, gerar um endereço MAC aleatório de acordo com o nome inserido.

O HoneyD é um *honeypot* interessante, tanto que é utilizado no projeto HoneyTARG de *honeynets* distribuídas. Porém, possui uma configuração e instalação complicada para um usuário leigo. Além do mais, o usuário é quem deve desenvolver seu próprio *script* para liberar e disponibilizar as portas e seus serviços para os invasores. Como o HoneyD se trata de um *honeypot* de 2007, é bem provável que ele tenha total funcionalidade de recursos para uma determinada versão do Linux.

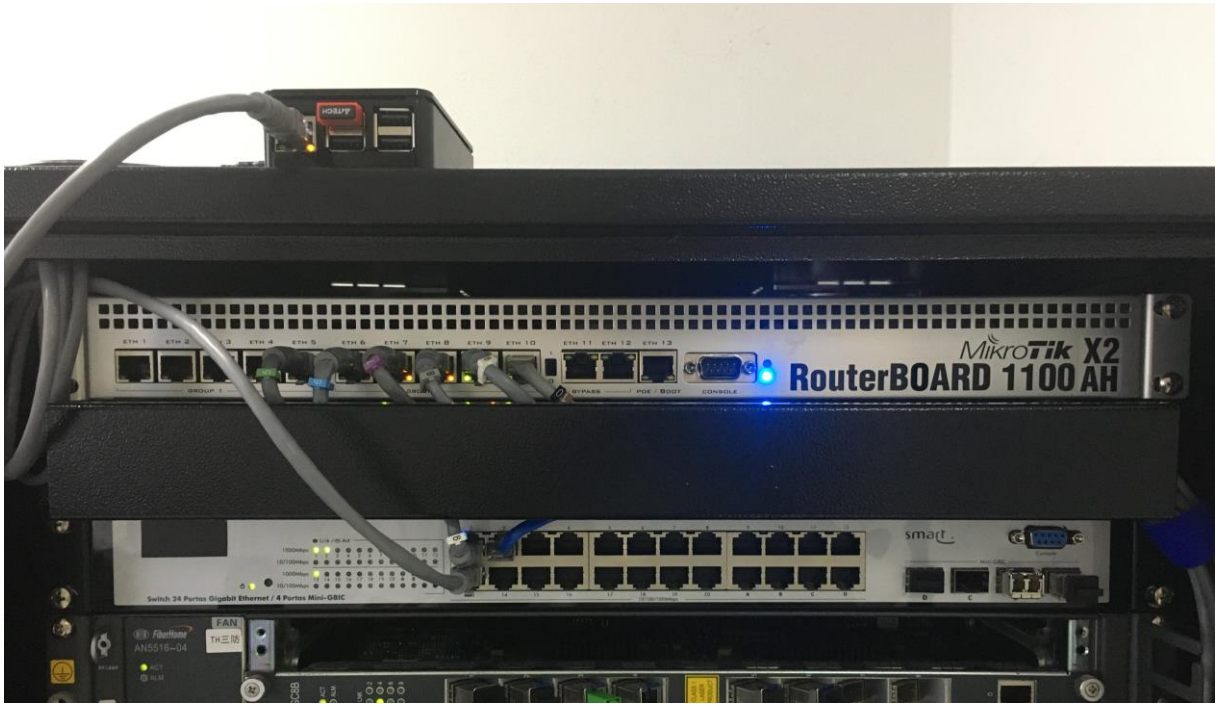
Como o HoneyD é utilizado em um grande projeto, não se pode negar que ele é um excelente virtualizador de máquinas falsas, porém, nos testes realizados não foi possível considerá-lo um bom *honeypot,* devido à sua falta de interação com o invasor.

5.4 Quarto Experimento – Cenário empresarial com Ubuntu Mate, Docker e SSH Honeypot

O Quarto Experimento foi realizado em um cenário empresarial com uma topologia bem distinta, contendo mais de 60 usuários, com VPNs, mais de um roteador, *firewalls* e *switches.* Esse experimento foi realizado com o *container* SSH Honeypot na plataforma Docker.

Os testes iniciais foram de *port scanner,* feitos para garantir que apenas o serviço de SSH Honeypot estivesse ativo no dispositivo. Desta maneira foi possível perceber que o nmap e o Advanced Port Scanner reconheceram que a *interface* de rede era de um Raspberry Pi. Ambos os programas também reconheceram a *interface* de rede quando foi utilizado o Raspbian.

Figura 39 – Raspberry Pi pronto para os testes no cenário empresarial.



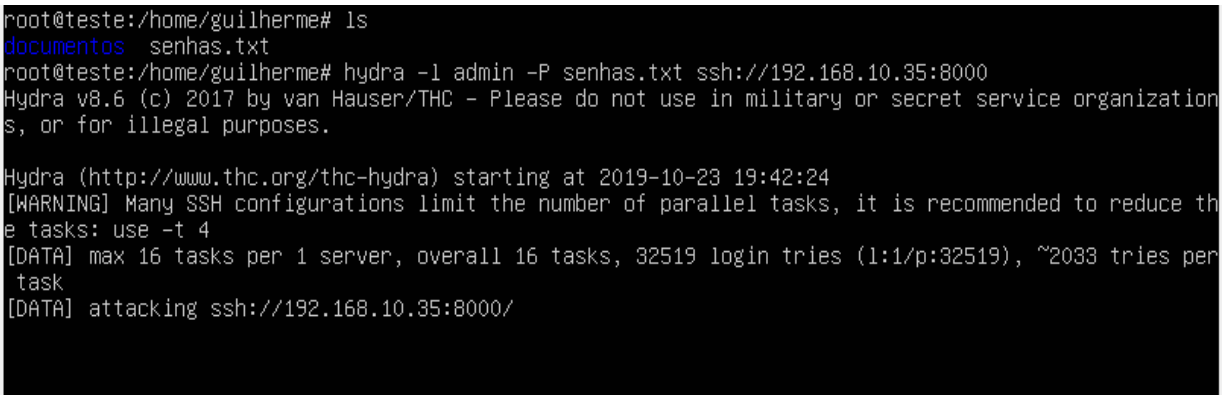
Fonte: Do autor (2019).

Após realizar um teste básico de acesso ao SSH Honeypot, foi feito um ataque em massa utilizando a ferramenta Hydra (Figura 40), com a qual é possível utilizar *wordlists.* Ou seja, o Hydra utiliza uma lista de palavras para abrir várias sessões de SSH e forçar um ataque por meio de força bruta, testando todas as possibilidades de usuário e senha.

Para executar esse teste no Hydra, foram utilizados dois computadores que estavam conectados à rede empresarial, em que cada computador utilizou uma *wordlist* diferente. Uma das *wordlists* continha 129.329 palavras em português e a outra, 319.378 palavras em inglês, juntas somando 448.707 palavras. Para realizar o ataque por força bruta foi utilizado o comando a seguir:

- hydra -l admin -P senhas.txt ssh://192.168.10.35:8000: esse comando executa o Hydra para acessar o SSH no IP 192.168.10.35:8000 com o usuário admin e com todas as senhas contidas no arquivo senhas.txt.

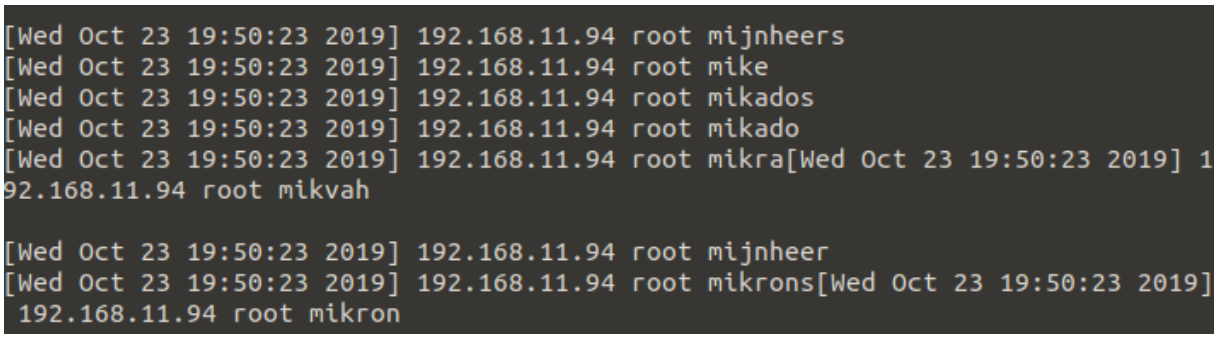
Figura 40 – Ataque de força bruta sendo realizado com o Hydra.



Fonte: Do autor (2019).

Com esse ataque por meio de dois computadores, foi possível gerar uma média de 60.000 tentativas de acesso por minuto, sendo 30.000 por computador que estava tentando acessar. Nesse teste, foi possível analisar o consumo de processamento do Raspberry Pi quando estava recebendo mais de 1000 requisições por segundo.

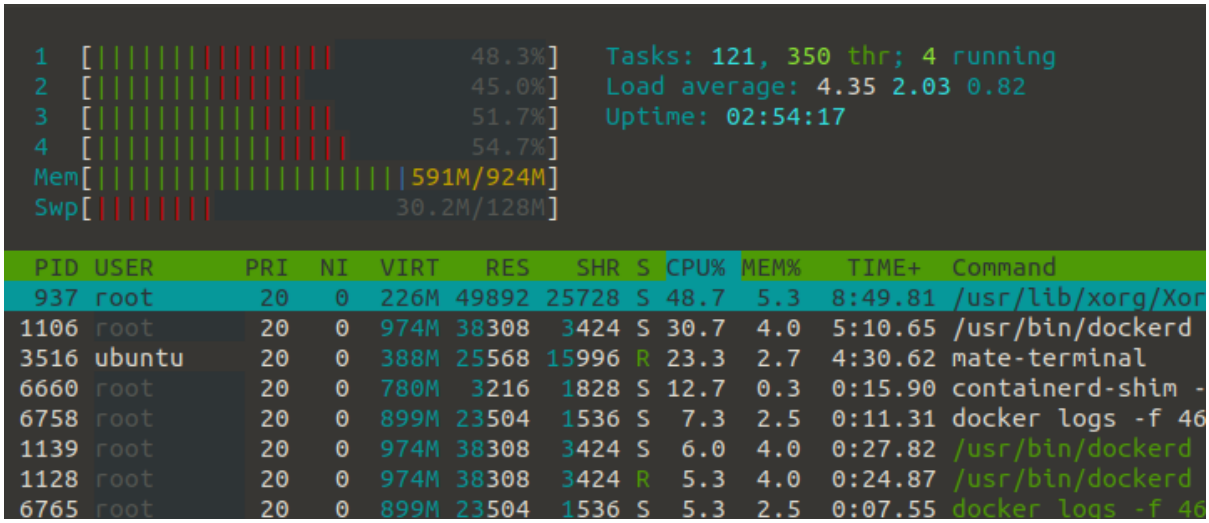
Figura 41 – *Log* de saída do SSH Honeypot sendo atacado pelo Hydra.



Fonte: Do autor (2019).

O uso de processamento do Raspberry Pi teve um pico de 80% em um de seus núcleos, mas, quando o computador que possuía a menor lista terminou de realizar os acessos, o Raspberry Pi manteve-se na faixa de 50% com 500 tentativas de acesso por minuto, conforme podemos visualizar na Figura 42.

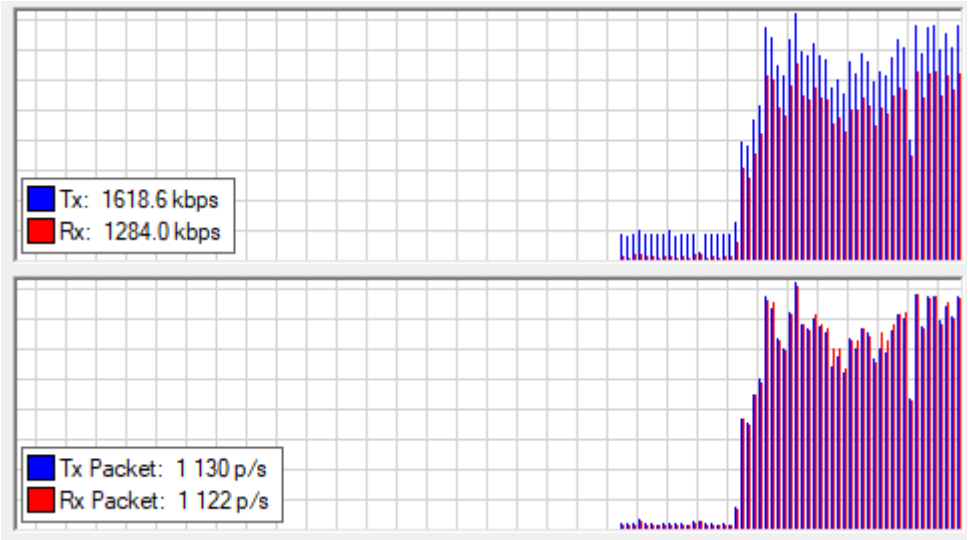
Figura 42 – Consumo de processamento do Raspberry Pi com cerca de 500 acessos por segundo.



Fonte: Do autor (2019).

Durante esse teste, também podemos acompanhar a quantidade de tráfego gerado pelo Hydra para o Raspberry Pi. Esse tráfego foi acompanhado pela RouterBoard RB1100 AHX2 que estava interligando os dois computadores ao Raspberry Pi. Como podemos analisar na Figura 43, o Raspberry estava recebendo cerca de 1618,6 Kbps (Kilobits por segundo) de tráfego com as 1000 tentativas de acesso por segundo do Hydra.

Figura 43 – Tráfego enviado do Hydra para o Raspberry Pi.



Fonte: Do autor (2019).

5.4.1 Análise dos resultados do Quarto Experimento

Com esse teste, foi possível analisar que o SSH Honeypot é uma excelente ferramenta em Docker, a qual consome poucos recursos de hardware e não deixa nenhuma vulnerabilidade para o servidor em que está hospedado. O Docker possui uma isolação do servidor, quando o SSH Honeypot apenas captura os dados de quem está tentando realizar o acesso, não deixando que o invasor se conecte pelo SSH.

Dessa forma, pode-se visualizar a robustez do Raspberry Pi para essa função, já que suportou as 60 mil tentativas de acesso por minuto sem problemas, não ultrapassando 80% do uso de processamento. Com isso, é possível notar que o Raspberry Pi possui um grande potencial para atuar como um *honeypot.*

Além de ser o menos vulnerável, o SSH Honeypot foi de fácil instalação e configuração, sendo assim de grande viabilidade para ser utilizado em qualquer cenário. Pode ainda ser instalado e configurado facilmente por qualquer administrador do sistema.

5.5 Quinto Experimento – Cenário campus com Ubuntu Mate, Docker e SSH Honeypot

O quinto experimento é bem semelhante ao anterior, porém, nesse caso, o Raspberry Pi foi colocado em uma rede ainda mais vulnerável. A rede utilizada foi a da UNIVATES, na sala 404 do prédio 11, com IP público.

Nesse teste, o Raspberry Pi ficou com o IP público “177.44.248.233” e, dessa forma, foi utilizado o nmap para verificar como o Raspberry Pi estava sendo visualizado pelos “invasores”. Podemos ver o resultado do nmap na Figura 44.

Figura 44 – Dados obtidos pelo nmap após realizar um scan no Raspberry Pi.

```
Initiating Service scan at 20:37
Scanning 1 service on 177-44-248-233.univates.br (177.44.248.233)
Completed Service scan at 20:37, 0.01s elapsed (1 service on 1 host)
Initiating OS detection (try #1) against 177-44-248-233.univates.br (177.44.248.233)
NSE: Script scanning 177.44.248.233.
Initiating NSE at 20:37
Completed NSE at 20:37, 0.28s elapsed
Initiating NSE at 20:37
Completed NSE at 20:37, 0.00s elapsed
Initiating NSE at 20:37
Completed NSE at 20:37, 0.00s elapsed
Nmap scan report for 177-44-248-233.univates.br (177.44.248.233)
Host is up (0.0020s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 5.9p1 Debian 5ubuntu1.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_  2048 ab:ec:f0:65:de:31:4c:11:8d:d4:ec:1f:9b:f3:30:c8 (RSA)
MAC Address: B8:27:EB:37:2B:EC (Raspberry Pi Foundation)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Uptime guess: 48.375 days (since Wed Sep 11 11:37:24 2019)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=263 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Fonte: Do autor (2019).

Após o *scan*, o IP do Raspberry foi informado para usuários que estavam fora da rede da UNIVATES e, eles tentaram realizar o acesso via SSH. O SSH Honeypot ficou em testes por cerca de 4 horas e, os únicos IPs registrados foram aqueles que tentaram realizar os acessos com o intuito de testar a ferramenta. Dessa forma, não detectou-se nenhuma tentativa de acesso suspeita.

5.5.1 Análise dos resultados do Quinto Experimento

Com esse experimento foi possível analisar que o SSH Honeypot está preparado para qualquer rede, pois, não oferece riscos e vulnerabilidades para o servidor em que está hospedado e, consegue ser facilmente instalado e executado. Além disso, consome poucos recursos de hardware em um cenário real, ficando nesse teste com o uso de processamento do Raspberry Pi abaixo de 15%. Podemos levar em consideração o teste anterior que chegou em 80% do uso de processamento com cerca de 60 mil tentativas de acesso por minuto como comparação. Isso só pode ocorrer no cenário real e, caso alguém deseje realizar um

ataque de DoS, enviando várias solicitações por segundo e fazendo o dispositivo travar.

Como o Raspberry Pi se trata de um *honeypot,* caso fosse derrubado não iria influenciar no resto da rede. Cumpre assim o seu papel, de desviar o foco dos ataques e fazer com que os invasores percam tempo tentando invadir algo falso. Conforme dito no quarto experimento, o SSH Honeypot, juntamente com o Docker, está pronto para qualquer tipo de topologia e cenário. Com certeza, foi a melhor ferramenta testada e que teve melhor compatibilidade com o Raspberry Pi, consumindo pouco processamento e deixando sobras para que o seu processamento fosse elevado devido à quantidade de acessos.

6 CONSIDERAÇÕES FINAIS

O presente trabalho teve como proposta configurar e analisar um *honeypot* hospedado em um servidor de baixo custo, tendo como hardware um Raspberry Pi e dispondo de ferramentas como Ubuntu Mate, Raspbian, Windows 10, Docker, Valhala Honeypot, HoneyD e HoneyPi. Com essas ferramentas e, apesar das dificuldades, foi possível atingir os objetivos especificados no início da pesquisa.

A instalação dos sistemas operacionais foi um grande desafio, primeiramente por desconhecer dos métodos mais simplificados de instalação e por ter que determinar qual sistema que iria ser utilizado com determinada ferramenta. Essa tarefa foi bastante complexa também devido ao fato de utilizar nos primeiros testes uma fonte de baixa corrente e um cartão de memória com baixa velocidade de leitura. Após ter alguns problemas com o desempenho da Raspberry Pi, foi providenciado uma fonte original que conseguiu fornecer toda a demanda de energia necessária para a placa, além do cartão de memória Sandisk Extreme PRO classe 10 que foi fundamental para instalação de todos os sistemas operacionais e para que fosse possível desfrutar de todo o desempenho do Raspberry Pi que mesmo assim foi pouco para algumas ferramentas.

A instalação dos *honeypots* foi outro grande desafio, devido ao fato de que os *honeypots* não são muito conhecidos e possuem poucas instruções acerca de sua instalação e funcionamento. Além disso, esse processo também foi complexo devido às limitações de hardware do Raspberry Pi, que não era compatível com alguns

softwares devido à sua arquitetura ARM ou, por não suportar toda a carga exigida pela aplicação.

Mesmo com essas limitações, foi possível configurar *honeypots* funcionais que possuem um importante papel na Segurança de Redes. Cada *honeypot* tem seus pontos positivos e negativos, tanto na instalação, quanto na funcionalidade.

Foi possível concluir que o HoneyPi é um excelente *honeypot* se for utilizado dentro de uma DMZ, que o HoneyD é uma excelente ferramenta para simular máquinas falsas, que o Valhala Honeypot é um *honeypot* completo e com um uso simplificado, dispensando instalações, porém consome muitos recursos. E que a ferramenta Docker foi crucial para cumprir os objetivos desse trabalho, com ele foi possível testar containers em uma máquina virtual, simular uma *honeynet* no GNS3 e após esses testes, foi utilizado com o SSH Honeypot no Raspberry Pi.

Dessa forma, pode-se dizer que um dos objetivos do trabalho, de buscar um *honeypot* de fácil instalação e configuração, foi atingido. Em particular, o SSH Honeypot em Docker possui uma instalação e execução bem simplificada e uma excelente segurança para atuar como um *honeypot* em diferentes topologias.

Através dos testes foi possível evidenciar a melhor configuração a ser utilizada com o Raspberry Pi. Ele pode executar o SSH Honeypot, simulando o serviço de SSH falso. Ao mesmo tempo, permite simular máquinas com o HoneyD, com a finalidade de apenas criar uma rede ao redor do *honeypot,* atuando como *honeynet.* Isso faz com que o sistema consiga capturar *logs* de *port scans* com um alcance muito maior do que apenas o endereço de IP do Raspberry Pi.

Como o Raspberry Pi é um servidor de baixo custo, foi possível analisar que ele se saiu muito bem para atuar como *honeypot* em pequenas e médias redes. Com ele foi possível implementar soluções que conseguem analisar a segurança de uma rede e fazer com que o administrador do sistema conheça melhor o inimigo, ou seja, o *cracker*.

Esse microcomputador juntamente com sua fonte e caixa teve um custo total de R$ 270. Soma-se a este os custos de cartões de memória, de R$ 70 para o Sandisk Extreme Pro, que apresenta maior velocidade, e R$ 40 para outro cartão de

classe 10, que dispõe de menor velocidade. Pode-se concluir que um sistema com custo inferior a R$ 350 conseguiu simular máquinas falsas, serviços vulneráveis e prover um IDS com um consumo de energia máximo de 12,75 W (potência de saída da fonte). Se comparado a um servidor ou computador básico, o Raspberry Pi é extremamente barato e eficiente como solução de *honeypot*, desde que sejam instaladas ferramentas otimizadas, compatíveis com a sua arquitetura e para uma rede que não seja tão complexa. Lembrando que o *honeypot* não deve ser utilizado como uma única ferramenta de segurança em uma rede, pois ele não trará nenhuma segurança efetiva, ele é apenas um complemento para a segurança de toda a infraestrutura, avisando o administrador dos incidentes e sendo uma distração para os *crackers*.

Entretanto, o Raspberry Pi não se mostrou efetivo para atuar em grandes redes como um único *honeypot.* Caso a rede seja muito vulnerável a ataques ou tenha muitos *hosts,* será necessário utilizar uma quantidade maior desses dispositivos ou, utilizar algum hardware mais robusto, já que, caso o Raspberry Pi possua uma quantidade de acessos maior do que 1.000 conexões SSH por segundo, ou esteja com algum outro recurso que tenha um alto consumo de hardware*,* é bem provável que atinja o uso máximo de processamento e memória, sofrendo um ataque de DoS por consequência.

Como contribuição, este trabalho apresenta passo-a-passo a instalação dos sistemas operacionais no Raspberry Pi, além de apresentar a instalação e configuração de *honeypots*, além de analisar possíveis locais e topologias em que podem ser utilizados.

Outra contribuição, a mais importante e principal deste trabalho, é a apresentação dos *honeypots,* que não são muito conhecidos. Ainda que seja incomum ouvir-se falar deles ou utilizá-los, estes são extremamente eficientes para a análise de vulnerabilidades em uma rede ou, até mesmo, para evitar ou atrasar os ataques dos invasores aos servidores ou a rede principal.

Para trabalhos futuros, sugere-se o desenvolvimento de algum *script* para utilização no HoneyD, com o intuito de liberar as portas como o SSH Honeypot, transformando um único Raspberry Pi em uma *honeynet* com serviços distintos e

interativos. Porém, seria necessário realizar uma análise no uso de processamento do Raspberry Pi, para verificar a viabilidade dessa *honeynet*. Caso o Raspberry Pi não consiga suprir a capacidade de processamento, seria interessante aplicar mais de um Raspberry Pi como um *cluster*, somando o poder de processamento e, talvez transformá-lo em um sistema de alta disponibilidade, com um custo reduzido.

Por fim, outra sugestão seria comparar o desempenho de um *honeypot* com Raspberry Pi a outras soluções no mercado, como o Orange Pi, Banana Pi, entre outros sistemas embarcados.

REFERÊNCIAS

ASSUNÇÃO, Marcos Flávio Araújo. **Guia do hacker brasileiro**. 1. ed. Florianópolis: Visual Books, 2002.

ASSUNÇÃO, Marcos Flávio Araújo. **Honeypots e Honeynets**. 1. ed. Florianópolis: Visual Books, 2009.

AZEVEDO, Tiago Souza. **Honeypots: A segurança através do disfarce**. 2005. 7p. Universidade Federal do Rio de Janeiro, Rio de Janeiro, Rio de Janeiro, 2005. Disponível em: <http://www.ravel.ufrj.br/sites/ravel.ufrj.br/files/publicacoes/honeynet.pdf>. Acesso em: 19 mai. 2019.

CERT.br. **Estatísticas Mantidas pelo CERT.br**. Disponível em: <https://www.cert.br/stats/>. Acesso em: 4 mai. 2019.

CERT.br. **honeyTARG Honeynet Project**. Disponível em: <https://honeytarg.cert.br/>. Acesso em: 4 mai. 2019.

DOCKER INC. **What is a container?** Disponível em: < https://www.docker.com/resources/what-container>

GIL, Antonio C. **Métodos e Técnicas de Pesquisa Social**. 6. ed. São Paulo: Editora Atlas, 2012.

JUNIOR, José de Sousa Martins. **Honeypot: Conceitos gerais e implementação de um Honeytoken**. 2006. 32p. Monografia de fim de curso (Bacharel em Engenharia da Computação), Universidade São Francisco, São Paulo, Itatiba, 2006. E-book. Disponível em: <https://docplayer.com.br/2982356-Honeypot-conceitos-gerais-e-implementacao-de-um-honeytoken.html>. Acesso em: 11 mai. 2019.

LIM, Charles; MARCELLO, Mario; JAPAR, Andrew; TOMMY, Joshua; KHO, I Eng. Development of Distributed Honeypot Using

Raspberry Pi. In: INTERNATIONAL CONFERENCE ON INFORMATION, COMMUNICATION TECHNOLOGY AND SYSTEM, 2014, Tangeran, Indonésia. **Proceedings**… Surabaya, Indonésia: IEEE, 2014. Disponível em: <https://www.researchgate.net/publication/271825880_Development_of_Distributed_Honeypot_Using_Raspberry_Pi >. Acesso em: 25 mai. 2019.

MARCONI, Marina de A; LAKATOS Eva M. **Fundamentos de Metodologia Científica**. 5. ed. São Paulo: Editora Atlas S.A, 2003.

MORAES, Alexandre Fernandes de. **Segurança em redes: fundamentos**. 1. ed. São Paulo: Editora Érica, 2010.

NAKAMURA, Emilio Tissato; GEUS, Paulo Lício de. **Segurança de redes em ambientes cooperativos**. 1. ed. São Paulo: Novatec Editora, 2007.

OLIVEIRA, Frederico Santos de. **Honeypot: Um Ambiente para Análise de Intrusão**. 2008. 62p. Monografia de fim de curso (Bacharel em Ciência da Computação), Universidade Federal de Lavras, Minas Gerais, Lavras, 2008. E-book. Disponível em: <http://repositorio.ufla.br/jspui/bitstream/1/5091/1/MONOGRAFIA_Honeypot_um_ambiente_para_analise_de_intrusao.pdf>. Acesso em: 18 mai. 2019.

PRODANOV, Cleber C; DE FREITAS, Ernani C. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico.** 2. ed. Novo Hamburgo: Editora Feevale, 2013.

RASPBERRY PI FOUNDATION. **Raspberry Pi 3 Model B+**. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>. Acesso em: 25 mai. 2019.

RASPBIAN. **Welcome to Raspbian**. Disponível em: < https://www.raspbian.org/>. Acesso em: 31 out. 2019.

RAVANELLO, Anderson Luiz; HIJAZI, Houssan Ali; MAZZORANA, Sidney Miguel **Honeypots e Aspectos Legais**. 2004. 85p. Dissertação (Pós-Graduação em Informática Aplicada), Pontifícia Universidade Católica do Paraná, Paraná, Curitiba, 2004. E-book. Disponível em: <https://docplayer.com.br/510632-Honeypots-e-aspectos-legais.html>. Acesso em: 12 mai. 2019.

RODRIGUES, Fabrício dos Santos; SOUZA, Thiago Rafael de; DINIZ, Cristiano Antônio. Avaliação da Ferramenta de Segurança da Informação Honeypot. **Revista Pensar Tecnologia**, v. 4, n. 1, Sem I. 2015. Disponível em: <http://revistapensar.com.br/tecnologia/pasta_upload/artigos/a91.pdf >. Acesso em: 19 mai. 2019.

STALLINGS, William. **Criptografia e segurança de redes: princípios e práticas**. 6. ed. São Paulo: Pearson Education do Brasil, 2015.

TEIXEIRA JR, José Helvécio; SUAVÉ, Jacques Philipe; MOURA, José Antão Beltrão; TEIXEIRA, Suzana de Queiroz Ramos. **Redes de computadores: Serviços, Administração e Segurança**. 1. ed. São Paulo: Makron Books, 1999.

UBUNTU MATE TEAM. **About**. Disponível em: <https://ubuntu-mate.org/about/>. Acesso em: 1 nov. 2019.

WADLOW, Thomas A. **Segurança de redes: projeto e gerenciamento de redes seguras**. 1. ed. Rio de Janeiro: Editora Campus, 2001.

UNIVATES

R. Avelino Talini, 171 | Bairro Universitário | Lajeado | RS | Brasil
CEP 95914.014 | Cx. Postal 155 | Fone: (51) 3714.7000
www.univates.br | 0800 7 07 08 09