O USO DA FERRAMENTA DE HONEYPOT KIPPO PARA A SEGURANÇA DE REDES

Pablo Rezende SILVA
Centro de Ensino Superior de Juiz de Fora, Juiz de Fora, MG
Daves Márcio Silva MARTINS

Resumo: O objetivo do presente estudo é apresentar a coleta de informações de ataques a redes computacionais. Para simular a infraestrutura, foi utilizado um *Raspberry PI* para hospedar um *Honeypot* que sofrerá o ataque. Foram utilizados *pentest* direcionados ao *honeypot* para mostrar a exploração de uma senha fraca. Os resultados demonstram o potencial do uso dessa ferramenta para apoiar a segurança de redes computacionais.

Palavras-chave: *Honeypot*; *Raspberry PI*; Segurança Informação; *Firewall*.

1 INTRODUÇÃO

A informação é o bem que garante a continuidade dos negócios das organizações atuais, sendo um ativo muito importante para qualquer instituição e pode ser considerado o recurso patrimonial mais crítico, responsável pela viabilização dos negócios e maximizando o retorno dos investimentos (BRASIL, 2012).

Devido à relevância da informação para os negócios, também é grande a importância da segurança da mesma. De acordo com Coelho et al. (2014), é de responsabilidade da segurança, a proteção da informação, sendo determinante para assegurar a competitividade, a lucratividade, o atendimento aos requisitos legais e preservar a imagem da organização junto ao mercado.

Conforme Giavorato e Santos (2013), a segurança da informação não é um produto, é um processo e, por isso, há enorme dificuldade em se determinar o nível de segurança que um sistema possui. É necessário investigar os riscos, realizar testes, validar as políticas de segurança e tecnologias utilizadas com o objetivo de atender os preceitos de segurança da informação.

Configurar, adequadamente, a segurança de um ambiente computacional é uma tarefa árdua, principalmente se as vulnerabilidades do sistema não forem conhecidas pelo administrador e, por isso, deve-se implantar um processo de segurança da informação, verificar as falhas que o sistema possui para que essas sejam monitoradas. Um processo de gestão da segurança da informação deve cobrir o entendimento

e estabelecimento dos requisitos de segurança, implantação e controles de riscos, monitoramento e melhoria contínua (NBR ISO/IEC 27001, 2006).

Para combater os diversos tipos de ameaças a que os sistemas de informação estão expostos, é necessário analisar o comportamento do sistema quando o mesmo é alvo de ataques e conhecer os métodos utilizados pelos atacantes.

Uma questão deve ser levantada: como verificar se o sistema está protegido contra ataques? Para responder a essa questão e garantir os preceitos de segurança, são utilizadas ferramentas que criam um ambiente simulado com falhas de segurança propositais, permitindo o estudo das técnicas, ferramentas e métodos utilizados para a exploração das vulnerabilidades, para que sejam tomadas providências no ambiente real. Esses ambientes simulados são conhecidos como *honeypots* (Potes de Mel, que são iscas para invasores).

O uso de *honeypot* neste trabalho mostra suas contribuições, servindo de complemento à segurança existente, atraindo, tratando e monitorando ataques que exploram vulnerabilidades de acesso remoto. Será mostrada a infraestrutura necessária para a implantação de um *honeypot* e realizada a simulação de um ataque para se obter a senha de administrador do sistema Linux.

2 SEGURANÇA E PROBLEMAS DE SEGURANÇA DA INFORMAÇÃO

Para ter um nível de segurança computacional aceitável, minimizando e controlando os riscos, é necessário atender aos principais preceitos da segurança que são a confidencialidade, a integridade e a disponibilidade.

De acordo com Kurose e Ross (2010), confidencialidade é o conceito de comunicação segura e existe para garantir que as mensagens sejam compreendidas somente pelo remetente e pelo destinatário, sendo alcançada através de criptografia dos dados, combinada com a autenticação do ponto final, que é a confirmação de que as partes sejam realmente quem alegam ser. O mesmo autor define que a Integridade assegura que o conteúdo dos dados comunicados não tenha sido alterado durante a transmissão.

Já a disponibilidade é a garantia de que os dados estejam disponíveis a qualquer tempo e garante a prestação contínua do serviço (BRASIL, 2012).

São muitos os riscos a que um sistema computacional está exposto, pois eles possuem diversas vulnerabilidades que, se não forem tratadas e monitoradas, podem

ser exploradas por invasores. De acordo com Coelho et al. (2014), as vulnerabilidades são falhas que permitem o surgimento de deficiências na segurança e, por isso, é necessário que sejam utilizados controles de segurança por meio de políticas, processos, procedimentos, estruturas de *hardware* e *software* com constante monitoramento, auditorias e contínuo melhoramento.

Entre os tipos de vulnerabilidades mais comuns, podem ser citados: erros de configuração de *software*, engenharia social, falta de manutenção em *hardware*, uso de senhas fracas. Quando não tratadas, as vulnerabilidades tornam-se ameaças e podem ser exploradas, causando prejuízos.

3 ELEMENTOS DE SEGURANÇA

Os dados possuem valor pelo que representam para uma empresa e, para garantir sua segurança, alguns cuidados precisam ser tomados. Uma solução de segurança é alcançada pela combinação de diversas técnicas e ferramentas. A seguir, serão apresentados alguns itens essenciais que foram utilizados na infraestrutura adotada neste trabalho.

3.1 *Firewall*

O *Firewall* isola a rede interna da externa e realiza a segmentação e comunicação entre redes e sub-redes. De acordo com Kurose e Ross (2010), também atua controlando quais pacotes de dados são bloqueados ou descartados e seus destinos. O *firewall* é um componente de segurança computacional indispensável que evita principalmente invasões a uma rede ou máquina específica.

Os *firewalls* podem ter implantação em *hardware* ou em *software* e são classificados por categorias, quais sejam:

3.1.1 *Firewall* de filtragem de pacotes

Inspeciona os cabeçalhos dos pacotes de dados que passam pela rede e define se os mesmos serão aceitos ou descartados de acordo com as regras definidas. Essas regras podem ser baseadas no endereço de origem, de destino, porta utilizada para comunicação, tipo de serviços e etc.

3.1.2 *Firewall proxy*

Controla toda comunicação de dados da rede interna com a rede externa. Quando um *host* interno deseja comunicar com um externo, primeiro, deve estabelecer uma conexão com o *proxy* que, então, viabiliza a conexão com outra rede, por exemplo, a internet.

3.1.3 Arquitetura de *firewall*

Para a implantação de um *firewall*, existem alguns modelos de arquitetura que podem ser aplicados de maneira isolada ou combinada como o de DMZ (Zona Desmilitarizada) que, segundo Nakamura e Geus (2007), é uma área localizada entre a rede interna e a externa. O uso de DMZ visa diminuir os danos que possam ocorrer por uma violação de segurança, pois os recursos estão isolados.

Outras arquiteturas de *firewall* são utilizadas para unir redes. Nessas arquiteturas, o *firewall* possui duas ou mais interfaces de rede (ligadas a redes distintas), e, ao receber um pacote, determina se o mesmo será descartado ou redirecionado a outro *host*, realizando o redirecionamento e a comunicação entre redes.

3.1.4 Tabelas de *firewall*

Os *firewalls* devem ser configurados com as regras das operações que devem ser executadas. Nas tabelas, estão as cadeias (*chains*) que determinam condições para a aplicação das regras (Hertzog e Mas, 2013).

A tabela Filtro é responsável pelas regras de filtragem dos pacotes, podendo aceitar, recusar ou ignorar os pacotes de dados. As cadeias dessa tabela são *INPUT* (aplica a filtragem dos pacotes que chegam ao *firewall*), *OUTPUT* (aplica a filtragem dos pacotes enviados pelo *firewall*), e *FORWARD* (preocupa-se com os pacotes que atravessam o *firewall*). A tabela NAT (*Network Address Translation*, tradução de endereço de rede) tem a responsabilidade de traduzir os endereços de origem, o destino e as portas de comunicação. Também realiza o redirecionamento dos pacotes para aplicar as regras e implementa três cadeias: *PREROUTING* (na entrada dos pacotes), *POSTROUTING* (para a saída) e *OUTPUT* (aplica as regras em pacotes gerados pelo *firewall*). A tabela *Mangle* trata alterações nos pacotes, modificando seus cabeçalhos e utiliza as cadeias *PREROUTING*, *OUTPUT*, *INPUT*, *FORWARD* e *POSTROUTING*.

3.2 IDS, Sistemas de detecção de Intrusão

Um sistema IDS pode ser utilizado de duas maneiras. A primeira baseia-se em Rede (NIDS), que realiza a captura e analisa os cabeçalhos dos datagramas da rede, e os compara aos registros de uma base de dados de assinaturas de ataques e anomalias para, assim, detectar atividades maliciosas suspeitas. Na rede, próximos a sistemas que precisam ser monitorados, devem ser instalados sensores de maneira escondida (*Stealth*). Outra forma de IDS é a baseado em *host* (HIDS), que analisa *logs* de eventos de auditoria locais de um *host* para detectar as anomalias e indícios de intrusão. O sistema computacional gera registros de acessos, alterações de arquivos do sistema, privilégios de usuários, processos, programas em execução, desempenho de *hardware*, integridade de arquivos, atividades específicas do sistema e diversos outros, e esse material é utilizado para determinar alterações no comportamento da máquina e de aplicações que indiquem uma violação.

Um IDS pode detectar invasões em tempo real e ser configurado para notificar a atividade ou tomar alguma ação reativa. Os IDSs não reconhecem se o ataque foi bem sucedido, apenas apontam alterações que indicam que um ataque foi iniciado, o que gera diversos “falso-positivos”, o que pode tornar a atividade de monitoramento muito complexa.

3.3 *Honeypot*

Honeypots são sistemas e dispositivos que criam ambientes computacionais falsos, semelhantes aos reais, e tem por objetivo atrair possíveis ataques para que se possa monitorar e estudá-los.

De acordo com Marcelo e Pitanga (2003), o valor de um *honeypot* está justamente na sua capacidade de ser atacado de maneira monitorada, não sendo usados em concorrência a ambientes de produção. O objetivo comum aos diversos tipos de *honeypot* é o de enganar os atacantes.

O *honeypot* executa a coleta de dados somente quando há interação sua única função é de atrair e monitorar ataques e, por isso, qualquer interação com o mesmo é considerada não autorizada, assim, não é gerado um número alto de sinais de alertas falsos. De acordo com Pauli (2014), os *honeypots* são procedimentos e ferramentas táticas que fornecem o primeiro movimento de vantagem para a segurança, podendo obter sucesso contra um invasor que irá buscar uma forma de burlar os IDS.

Organizações como o *The Honeynet Project*[1] e o CERT[2] utilizam esses recursos para fins de pesquisa, coletando informações e características de tipos de ataques utilizados, ferramentas, métodos e estratégias de invasão, dados geográficos, repositórios de *malwares* e outras informações relevantes. Os dados obtidos por um *honeypot* de pesquisa são analisados e contribuem para a indústria de sistemas de segurança como antivírus e detectores de intrusão com dados de assinaturas e padrões que alimentam as bases de dados dessas aplicações.

Os *honeypots* também são utilizados como elemento de segurança corporativa, utilizado para detecção ativa, análise e prevenção de ataques aos sistemas por receberem os atacantes nesse ambiente falso, impedindo que os mesmos atinjam o ambiente real e permitindo a tomada de medidas reativas para preservar a segurança da rede computacional.

Apesar das contribuições do *honeypots* para a segurança, existe a discussão das falhas que ele pode gerar na rede, caso a mesma não seja preparada adequadamente para o seu uso. De acordo com Marcelo (2005), é necessário o isolamento entre a rede de produção e a *honeynet* (rede de *honeypots*), em conjunto com o monitoramento e medidas de segurança preventivas e reativas a incidentes. É necessária uma análise dos riscos e a implantação de políticas de monitoramento e uso de ferramentas de segurança. Dessa forma, o *honeypot* funcionará como um complemento da segurança, não uma brecha.

Os *honeypots* podem ter três níveis de interação com o atacante. O baixo nível oferece atividades limitadas, dando respostas falsas para confundir aquele que interage com o sistema, sendo bastante útil contra ataques de *bots* (sistemas que automatizam tentativas de invasão), oferece um risco mínimo para a segurança da rede, pois funciona de forma emulada, não dando ao atacante acesso ao sistema.

Os de alta interação são sistemas operacionais com aplicações reais e falhas propositais. Nesses, a interação do invasor é de fato com o sistema, mas os serviços que são oferecidos não são os de produção. Devido a esse nível de interação, o *honeypot* permite coletar os dados sobre o invasor em grande quantidade (AZEVEDO, 2005) e possui uma implementação bastante complexa. De acordo com Marcelo

[1] Disponível em https://www.honeynet.org/

[2] Disponível em http://www.cert.org/

(2005), o risco que envolve esse tipo de *honeypot* é também sua principal vantagem já que, de fato, atrai o invasor.

Os *honeypots* de média interação são ambientes falsos que oferecem ao invasor a sensação de estar interagindo com um sistema real. De acordo com Marcelo (2005), o atacante, ao invadir esse tipo de sistema, fica preso num ambiente controlado sem contato com o real. Assim, tal nível de interação exige as mesmas preocupações de segurança que os de alta interação.

3.3.1 Ferramentas de Honeypot

Para este trabalho foi utilizada uma ferramenta de *honeypot* chamada Kippo [3] que é um *honeypot* de média interação e simula um serviço SSH (protocolo de rede de conexão segura). Foi escolhida por ser uma ferramenta específica para este protocolo, diferente de diversas outras que são mais abrangentes, e por isto tem maior complexidade de configuração, manutenção e extração de seus dados.

As principais alternativas de ferramentas de *honeypot* são: Honeyd[4], que e um *honeypot* de baixa interação que pode simular diversos sistemas operacionais, Nepentes[5] que provê alta interação, sendo muito útil para a coleta de *malwares,* que são sistemas maliciosos usados por invasores, Dionea[6] que é um *honeypot* de alta interação, muito usado para detectar ataques que utilizam Shellcode, linguagem de Shell UNIX/Linux) e dar suporte ao protocolo IPV6, entre outras ferramentas.

4 *SECURE SHELL* (SSH)

SSH ou *Secure Shell* é um protocolo de rede que permite a conexão segura entre dois computadores remotamente. De acordo com Perboni (2013), o SSH é uma ferramenta de acesso remoto com foco em segurança que utiliza criptografia para assegurar que somente o cliente autorizado ira acessar o servidor Linux ou Unix. Tendo sido realizada a conexão SSH, é possível executar comandos de forma segura, realizar transferência de arquivos e utilizar aplicações do sistema remoto.

---

[3] Disponível em https://code.google.com/p/kippo/

[4] Disponível em http://www.honeyd.org/

[5] Disponível em http://nepenthes.carnivore.it/

[6] Disponível em http://dionaea.carnivore.it/

A comunicação via SSH utiliza a porta número 22 por padrão, mas essa configuração pode ser modificada de acordo com as necessidades, para permitir maior segurança de acordo com Perboni (2013).

A alteração da porta padrão de conexão dificulta a ação de programas de enumeração (descoberta de dados) que vasculhem as redes, procurando acesso por essa porta. Outra forma de aprimorar a segurança dos sistemas que possuem um servidor SSH é determinar uma lista de *hosts* e usuários que terão autorização para a conexão.

Apesar de o SSH permitir o acesso remoto seguro, é constante que as más práticas de segurança dos usuários e administradores de rede possibilitem que pessoas mal intencionadas utilizem do SSH para obter controle sobre o sistema. Os principais motivos para isso são: manter configurações *default* (padrão) do serviço, manter usuários padrão como “adm”, “admin”, “administrador” e uso de senhas fracas.

5. ARQUITETURA NECESSÁRIA PARA A IMPLANTAÇÃO DE UM *HONEYPOT*

Nesse tópico, serão abordados aspectos da infraestrutura necessária para um *honeypot* de média interação em uma rede, com equipamentos e sistemas necessários para seu uso de forma segura e funcional.

5.1 Máquinas virtuais e *Raspberry PI* para a infraestrutura de rede

A virtualização de servidores simplifica a complexidade e consiste em utilizar sistemas que são criados de forma lógica e substituem máquinas físicas.

De acordo com Wallen (2013), alguns dos principais benefícios da virtualização são a facilidade de implantação, a manutenção e o *backup* (arquivamento de dados para recuperação), oferecendo flexibilidade para os planos de recuperação de desastres. O autor cita, também, como vantagem, a redução de custos devido à redução e melhor aproveitamento do *hardware* envolvidos.

A virtualização é usada ainda para a implantação de elementos de segurança, mas o uso de sistemas dedicados a essa finalidade tem os benefícios de evitar o comprometimento de outros serviços, devido ao isolamento dos recursos e a facilitação da manutenção.

Possuir um sistema dedicado somente para a segurança envolve um aumento significativo nos custos com equipamentos. Além da virtualização, há a possibilidade de se dedicar outros dispositivos para essa finalidade. Neste trabalho, o *Raspberry PI*, figura 1, é utilizado, sendo dedicado a um sistema de *honeypot*.

FIGURA 1
Raspberry PI

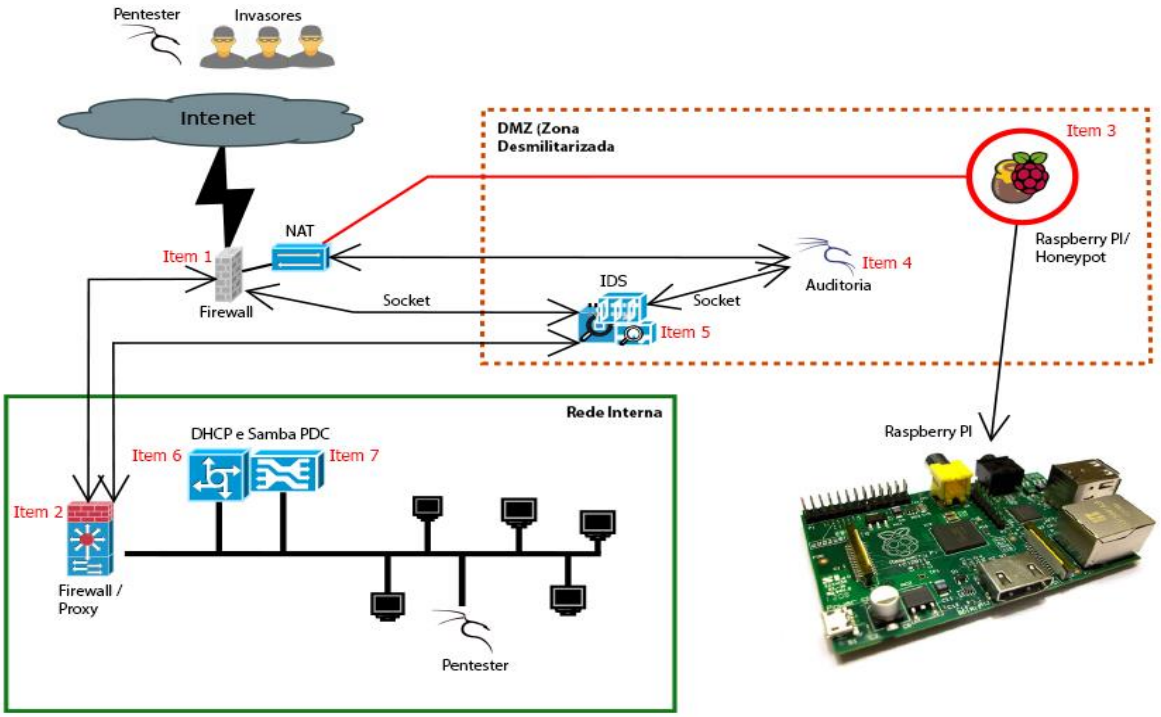
Esse dispositivo, é um pequeno computador do tamanho aproximado de um cartão de crédito que permite a interação com um sistema computacional (RASPBERRY PI FOUNDATION).

Por ser um dispositivo pequeno fisicamente, ter baixo consumo de energia e baixo custo, passa a ser uma boa opção para ser dedicado a sistemas de segurança e por estes motivos foi utilizado neste trabalho ao invés de outas alternativas, como máquinas virtuais e físicas.

5.2 Arquitetura do ambiente

Neste trabalho, foi utilizado um dispositivo *Raspeberry PI* para abrigar um *honeypot* Kippo e receber as tentativas de acesso SSH não autorizadas na rede, ficando tal dispositivo isolado do restante da infraestrutura e monitorado. A Figura 2 ilustra os equipamentos necessários para a implantação dessa arquitetura.

FIGURA 2
Infraestrutura da rede



Aprimeira linha de segurança do modelo de arquitetura de rede proposto possui um *Firewall* de Segurança Perimetral, item 1, que define qual o destino de uma requisição da rede externa, implementando algumas regras de segurança essenciais para evitar ataques à rede. Já o segundo *firewall*, item 2, que é um *proxy*, possui uma política mais restritiva de acesso do lado interno ao externo (*internet*), além de complementar o *firewall* perimetral quanto à segurança no sentido externo interno.

O *Firewall* de Segurança Perimetral realiza a filtragem de pacotes ao receber a conexão da rede externa, realizando a filtragem do tráfego, analisando os cabeçalhos dos pacotes, descartando as requisições que não atendem à política de segurança e permitindo o acesso a serviços SSH através da porta 22. As requisições são encaminhadas para uma máquina *Raspberry PI*, item 3, que se encontra na DMZ (Zona Desmilitarizada). É, nessa máquina, que está instalado o *Honeypot*. Os demais pacotes, que são filtrados pelo *firewall*, são roteados, através de outra interface de rede, para o *proxy*, permitindo que os mesmos cheguem aos seus destinos.

O acesso via SSH ao *firewall*, sem ser redirecionado para o *honeypot*, só é possível partindo de um IP da rede interna, conhecido e permitido, através de uma porta diferente da 22 definida para isso.

No primeiro *firewall*, a tabela *Mangle* usando *Prerouting* faz a alteração nos cabeçalhos dos pacotes que serão direcionados para o *honeypot*, modificando-os conforme o TOS de acesso remoto (TOS, *Type of Service*, tipo de serviço), alterando a prioridade dos pacotes e o TTL (*Time to Live*, Tempo de Vida para o protocolo IPV4, ou *Hop Limit*, Limite de Encaminhamento para o protocolo IPV6). TTL ou *Hop Limit* é um campo do cabeçalho do datagrama que é decrementado a cada salto de roteamento (cada transmissão da informação), indicando o máximo de roteadores pelos quais o datagrama pode passar até o valor chegar a 0 e ser descartado (EQUIPE IPV6.BR, 2012). Com essa alteração, é possível determinar que, caso o pacote de dados do acesso SSH seja roteado, por qualquer motivo, para um destino diferente do *honeypot*, este seja descartado, impedindo que o invasor acesse outros pontos da rede.

Para que os pacotes da conexão SSH sejam redirecionados para o *honeypot*, é utilizada a tabela NAT, com a *chain Pre routing Destination* NAT (DNAT). As alterações nos pacotes com o novo destino são feitas antes do roteamento. Assim o *firewall* perimetral trata o destino do NAT, fazendo a tradução de endereço e permitindo o acesso SSH para o *honeypot*.

O *Raspberry PI* que hospeda *honeypot* é configurado na DMZ, localizada entre a rede interna e a externa, protegida pelo *proxy* e o *firewall* de segurança perimetral. Complementando a política de monitoramento do acesso ao *honeypot* pelo invasor, devem ser implementados recursos secundários de segurança como IDS de *host* (HIDS) e de rede (NIDS). Um sensor de IDS deve ser instalado no *firewall* perimetral para assim notificar os acessos que serão direcionados para o *honeypot* e as ameaças em potencial. Outro deve ser instalado na máquina *Raspberry PI* e na máquina responsável por receber os *logs*.

Na DMZ, há um *host* que possui a responsabilidade de coletar dados de auditoria, item 4. Essa máquina recebe os registros de *log* do *firewall* perimetral, dos *hosts* que implementam IDS, sistema de resposta a incidentes, e do honeypot. Toda comunicação com essa máquina deve ser realizada através de um *socket* (interface de comunicação através de uma rede) dedicado à comunicação segura.

Na detecção dos ataques de SSH que são o alvo deste estudo, o IDS deve levar em consideração, na análise dos cabeçalhos dos pacotes, o TOS de SSH, o TTL e endereços de origem e destino, tratando como intrusão pacotes do serviço SSH com TTL alto.

As máquinas mais expostas da rede ou que possuam serviços mais críticos, como as máquinas físicas que hospedam as virtuais da infraestrutura da rede, devem ter um monitoramento constante, usando NIDS. Dessa forma, o sistema monitorado gera registros de acessos, de alterações e integridade de arquivos, de processos, de desempenho e de diversos outros parâmetros com a finalidade de detectar alterações que possam dar indícios de violações. Assim, é garantida a integridade das máquinas e permite a tomada de ações reativas para casos de percebidas invasões.

Complementando a estrutura da rede, internamente deve haver um servidor DHCP, item 5, para fazer a distribuição de endereços IP da rede interna, garantindo assim que todos os *hosts* tenham um IP válido para a rede interna e um servidor Samba, item 6, configurado para gerenciar o domínio local através do serviço PDC.

O *Firewall proxy* deve estar configurado com regras de controle de acesso, aceitando conexões somente de *hosts* e usuário configurados e autenticados para isso.

6 FALHAS DE SEGURANÇA E TESTES DE INVASÃO

Para verificar os mecanismos de segurança e identificar as falhas, os sistemas devem ser testados através de *pentests*, ou testes de penetração. Esse processo pode apontar diversas falhas de segurança, como vulnerabilidades ocasionadas pelo uso de senhas fracas.

6.1 Falhas de segurança de senhas SSH e ataque de *Brute Force*

Um tipo comum de ataque que explora vulnerabilidades de senha utilizando o protocolo SSH é o de força bruta *(bruteforce*) que, de acordo com Luchi (2013), é o tipo mais comum de ataque para explorar essas vulnerabilidades. Nesse ataque, é realizado um processo de enumeração que, de acordo com Giavorato e Santos (2013), é a fase de um ataque que permite obter nomes de máquinas, usuários, serviços e versões. Após esse processo, o atacante usa uma *wordlist* (lista de palavras), que é utilizada em testes repetitivos de combinações de usuários e senhas usando ferramentas específicas para isso (Martinelo e Bellezi, 2014).

6.2 *Pentest* como auditoria da segurança de sistemas

Pentest ou *Penetration Test* é um método para testar e descobrir vulnerabilidades em uma rede ou em sistemas, que utiliza métodos de avaliação de segurança, aplicando simulações de ataques para a verificação de falhas e, assim, realizar ajustes nos mecanismos e políticas de segurança (GIAVORATO e SANTOS, 2013).

De acordo com Giavorato e Santos (2013), os *pentests* crescem de importância devido ao aumento do número de empresas e instituições que, para aumentar a produtividade, disponibilizar serviços ou reduzir custos e tempo de implantação, instalam servidores e sistemas computacionais sem a preocupação com segurança.

Ainda de acordo com os referidos autores, em um processo de *pentest*, o responsável deve agir e pensar como um invasor e utilizar das suas mesmas técnicas e ferramentas. Os testes podem ser realizados com diversos níveis de conhecimento em relação ao sistema auditado, sendo mais comum a prática dos testes *Black Box*, ou caixa preta, caracterizados pelo desconhecimento prévio do sistema alvo e que, por isso, é o mais verossímil.

Ao final dos *pentests*, o auditor gera relatórios com detalhes dos testes realizados, resultados e vulnerabilidades encontradas para determinar as medidas para combater ou, pelo menos, amenizar as falhas encontradas.

7 FUNCIONAMENTO DO *HONEYPOT* KIPPO E RESULTADOS DE UM ATAQUE SSH

Foi necessário para o desenvolvimento deste trabalho a instalação do sistema operacional Raspbian[7] no Raspberry PI, após sendo instalado o *honeypot* Kippo.

Para o funcionamento do Kippo faz-se necessário instalar o Subversion, usado para acessar e baixar o Kippo do repositório, o Python-Twisted ferramenta que monitora o tráfego de rede usando o protocolo SSH, o Python-MySQLdb, que permite conectar no banco MySQL, o MySQL-Server, servidor de banco de dados e o Apache 2, servidor web.

Para o correto funcionamento da ferramenta de *honeypot* Kippo, algumas configurações podem ser realizadas no arquivo de configuração kippo.cfg: como definição

[7] Distribuição Linux baseada no Debian específica para *Raspberry PI*, disponível em http://www.raspbian.org/

de endereço IP em que a ferramenta irá aguardar conexão, porta que utilizará, diretório que registrará *logs*, diretório que armazenará os arquivos baixados pelo atacante na máquina, IP falso que o *honeypot* apresentará, entre outras coisas.

As configurações da senha e do usuário de que o Kippo irá aceitar a conexão SSH são realizadas no arquivo localizado no diretório do Kippo data/ no arquivo userdb.txt.

Quando um atacante obtém acesso ao terminal oferecido pelo Kippo, o intruso irá realizar alguns comandos Linux para usufruir do acesso que obteve. A ferramenta Kippo vem preparada para responder a diversos desses comandos Linux, mas as definições *default* (padrão) podem ser facilmente identificadas por um atacante experiente. Além disso, a ferramenta cobre a resposta de poucos comandos e arquivos de configuração.

Por isso, o Kippo permite a personalização dessas respostas e a criação de novas. Essas configurações devem ser feitas nos diretórios do Kippo *honeyfs*/ e textcmds/.

Uma máquina Backtrack 5R3 foi utilizada para realizar o *pentest*, simulando ser um atacante. A figura 3 mostra que é realizada uma varredura com a ferrramenta NMAP para verificar serviços ativos em um *host*, que é o *honeypot*. Conforme a figura, o serviço de SSH está funcionando na porta 22 com o estado de “open”.

FIGURA 3
Descoberta de informações pelo atacante

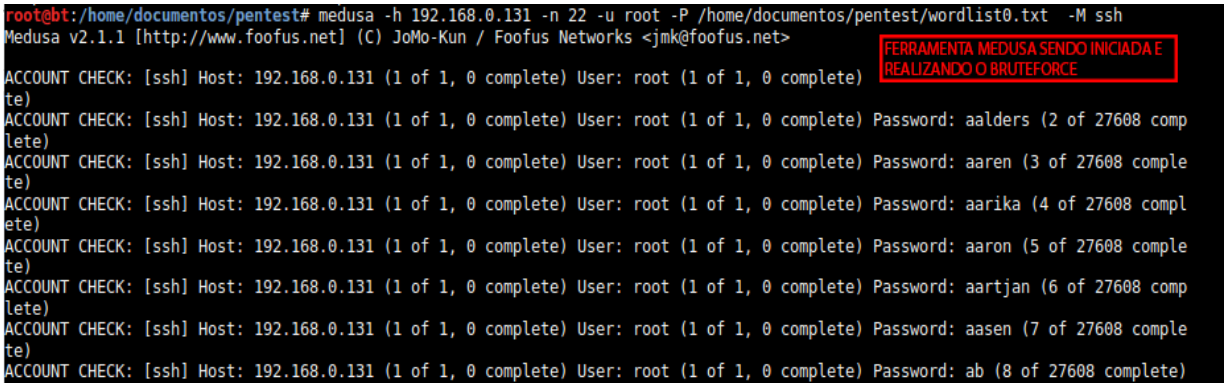
Após obter informações de seu alvo, o invasor executa o ataque de *bruteforce*, utilizando a ferramenta Medusa[8] (ferramenta que tenta obter acesso a serviços remotos, como SSH, FTP, HTTP e outros), que inicia as tentativas de descobrir a senha

[8] Disponível em http://h.foofus.net/?p=586

para o usuário *root*, utilizando uma lista wordlist.txt de senhas, até a lista ser toda percorrida ou obter sucesso como mostrado na figura 4.

FIGURA 4

Execução do *bruteforce*



Esse processo de descoberta e tentativas de violação por *bruteforce* foi executado de maneira simultânea por cinco máquinas atacantes. O sistema Kippo, através da ferramenta Python-Twisted, mostra, na tela e em tempo real, toda a operação dos atacantes (figura 5), com IP, serviço, ferramentas utilizadas, usuário e senhas utilizadas, estas informações são gravadas em banco de dados e em arquivos de *log*, kippo.log (figura 6, mostrando a identificação da ferramenta usada pelo atacante e quando o mesmo obtém sucesso).

FIGURA 5

Tela do Python-Twisted sofrendo tentativas de invasão

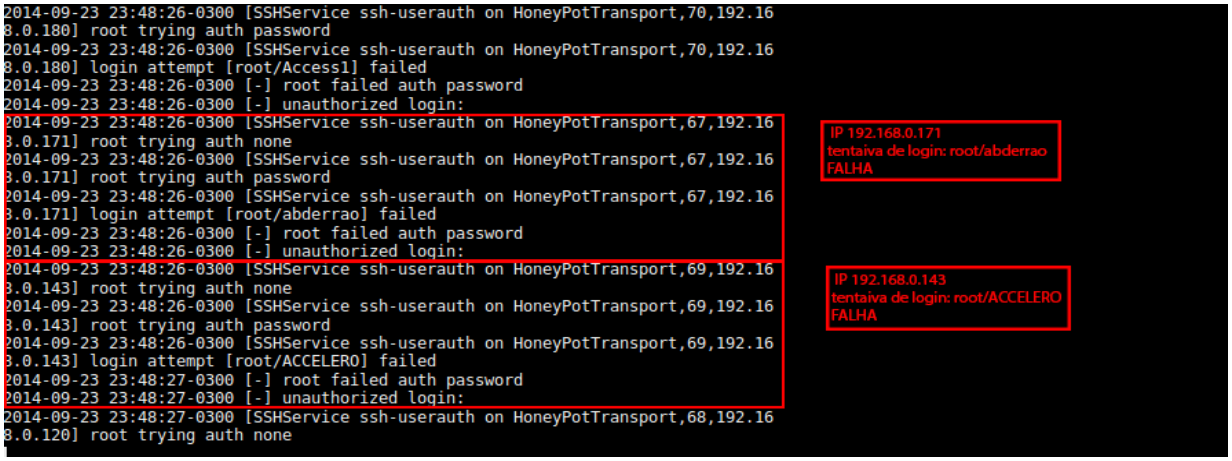
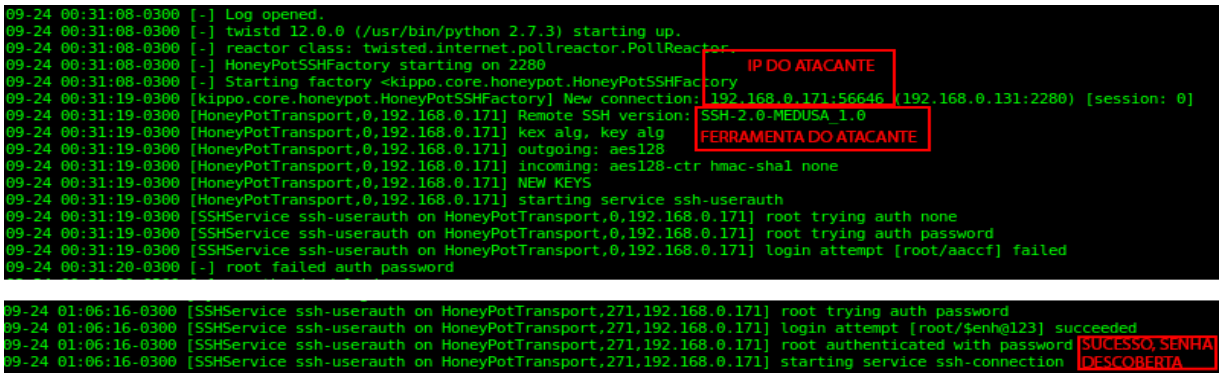


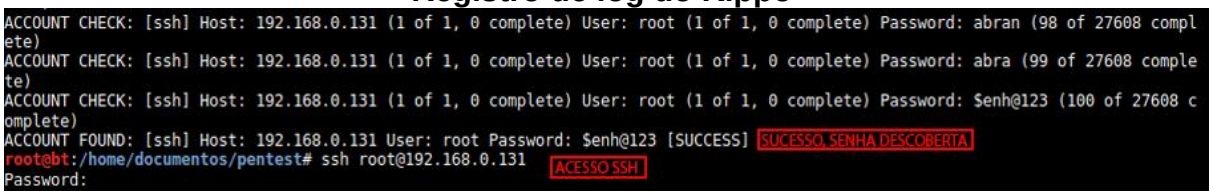
FIGURA 6
Registro de *log* do Kippo




```
09-24 00:31:08-0300 [-] Log opened.
09-24 00:31:08-0300 [-] twistd 12.0.0 (/usr/bin/python 2.7.3) starting up.
09-24 00:31:08-0300 [-] reactor class: twisted.internet.pollreactor.PollReactor.
09-24 00:31:08-0300 [-] HoneyPotSSHFactory starting on 2280
09-24 00:31:08-0300 [-] Starting factory <kippo.core.honeypot.HoneyPotSSHFactory
09-24 00:31:19-0300 [kippo.core.honeypot.HoneyPotSSHFactory] New connection: 192.168.0.171:56646 (192.168.0.131:2280) [session: 0]
09-24 00:31:19-0300 [HoneyPotTransport,0,192.168.0.171] Remote SSH version: SSH-2.0-MEDUSA_1.0
09-24 00:31:19-0300 [HoneyPotTransport,0,192.168.0.171] kex alg, key alg
09-24 00:31:19-0300 [HoneyPotTransport,0,192.168.0.171] outgoing: aes128
09-24 00:31:19-0300 [HoneyPotTransport,0,192.168.0.171] incoming: aes128-ctr hmac-sha1 none
09-24 00:31:19-0300 [HoneyPotTransport,0,192.168.0.171] NEW KEYS
09-24 00:31:19-0300 [HoneyPotTransport,0,192.168.0.171] starting service ssh-userauth
09-24 00:31:19-0300 [SSHService ssh-userauth on HoneyPotTransport,0,192.168.0.171] root trying auth none
09-24 00:31:19-0300 [SSHService ssh-userauth on HoneyPotTransport,0,192.168.0.171] root trying auth password
09-24 00:31:19-0300 [SSHService ssh-userauth on HoneyPotTransport,0,192.168.0.171] login attempt [root/aaccf] failed
09-24 00:31:20-0300 [-] root failed auth password
09-24 01:06:16-0300 [SSHService ssh-userauth on HoneyPotTransport,271,192.168.0.171] root trying auth password
09-24 01:06:16-0300 [SSHService ssh-userauth on HoneyPotTransport,271,192.168.0.171] login attempt [root/$enh@123] succeeded
09-24 01:06:16-0300 [SSHService ssh-userauth on HoneyPotTransport,271,192.168.0.171] root authenticated with password
09-24 01:06:16-0300 [SSHService ssh-userauth on HoneyPotTransport,271,192.168.0.171] starting service ssh-connection
```

Quando o atacante obtém sucesso, a ferramenta Medusa exibe o usuário e senha utilizados com sucesso e o atacante realiza o acesso ao sistema com o usuário *root (*usuário com poder de administrador no sistema Unix ou Linux*)*, figura 7. Logado no sistema, o invasor executa uma série de comandos no termimal Linux e o *honneypot* responde conforme configurações realizadas.

FIGURA 7
Registro de log do Kippo

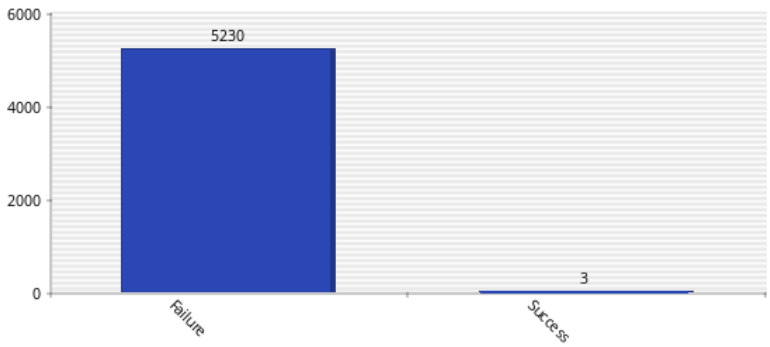



```
ACCOUNT CHECK: [ssh] Host: 192.168.0.131 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: abran (98 of 27608 compl
ete)
ACCOUNT CHECK: [ssh] Host: 192.168.0.131 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: abra (99 of 27608 comple
te)
ACCOUNT CHECK: [ssh] Host: 192.168.0.131 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: $enh@123 (100 of 27608 c
omplete)
ACCOUNT FOUND: [ssh] Host: 192.168.0.131 User: root Password: $enh@123 [SUCCESS]
root@bt:/home/documentos/pentest# ssh root@192.168.0.131
Password:
```

Através da ferramenta Kippograph, o Kippo gera as informações de forma gráfica para facilitar a análise.

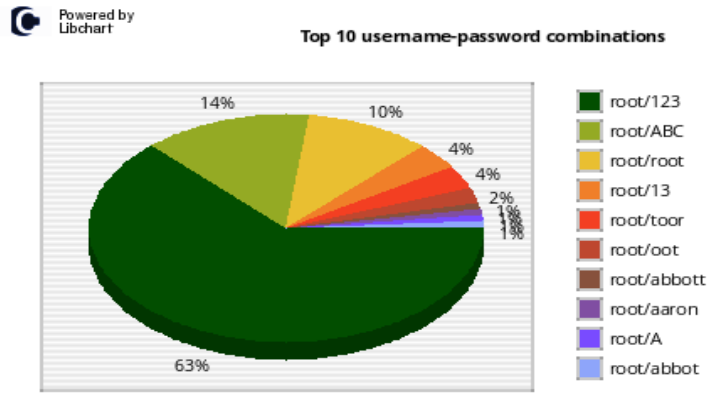
A figura 8 apresenta o total de 5233 tentativas de acesso via SSH ao *honeypot*, sendo que 5230 foram falhas e três bem sucedida. Essa quantidade de falhas é devido às tentativas do uso de *bruteforce*.

FIGURA 8
Quantidade de falhas e sucessos de acesso SSH



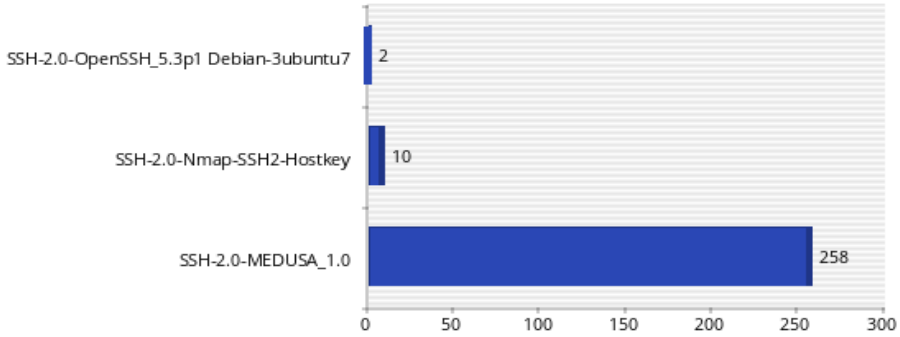
A figura 9 apresenta as principais combinações de usuário e senha utilizadas para as tentativas de acesso SSH.

FIGURA 9
Principais combinações de usuário e senha



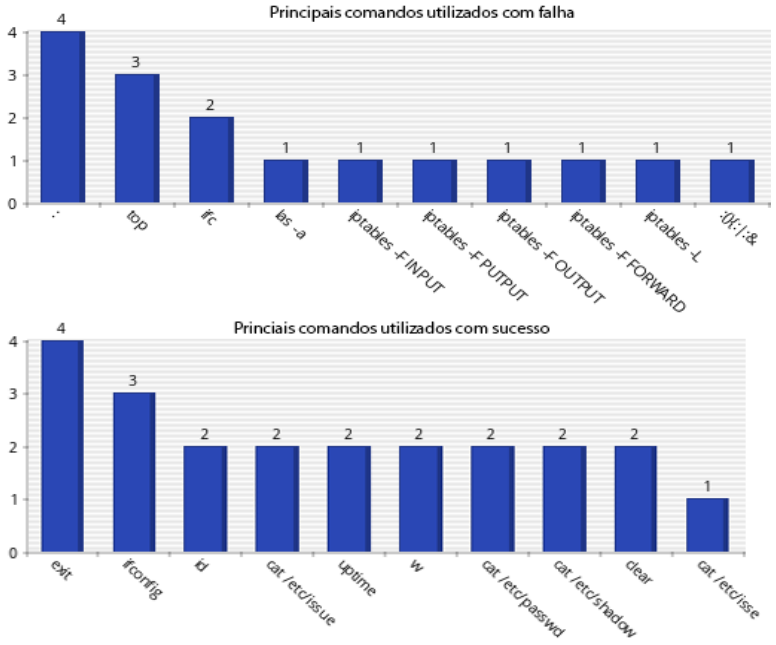
A figura 10 mostra as ferramentas SSH utilizadas, sendo o maior número de uso o da ferramenta Medusa, que é utilizada para *bruteforce*. O segundo número é o da ferramenta NMAP, utilizada para descoberta de informações do *host* alvo, e, por fim, a ferramenta OpenSSH, que é utilizada para estabelecer a conexão SSH.

FIGURA 10
Ferramentas clientes SSH utilizados pelos atacantes



A figura 11 mostra os principais comandos utilizados com sucesso e falha no terminal. Os sucessos mostram os comandos para descoberta de informação do sistema e permitem o estudo do comportamento de invasores, enquanto que as falhas mostram comandos que o administrador do *honeypot* precisa configurar para simular um comportamento mais próximo ao de uma máquina real.

FIGURA 11
Comandos utilizados com sucesso e falha



Além dos gráficos apresentados, o Kippo Grapf gera outros menos relevantes para este estudo, como os que apontam dados geográficos de origem dos ataques, IPs dos hosts atacantes, ataques por dia e por semana, e dados de atividades detectadas por terem sido realizadas por humanos e por máquinas robôs automatizadas para executar as tentativas de acesso.

8 CONSIDERAÇÕES FINAIS

A ferramenta adotada para este trabalho, Kippo, assim como outras ferramentas de segurança de redes não possui a instalação e configuração triviais, pois, exigem conhecimentos multidisciplinares e planejamento por parte do responsável. Dentre os conhecimentos necessários e adquiridos na execução deste trabalho destacam-se: conhecimento em infraestrutura de rede, *firewall*, sistemas operacionais, protocolos e camadas de rede, roteamento de pacotes, tipos de ataques e suas prevenções.

Com base no ambiente utilizado e no experimento de invasão, foi possível obter diversas informações sobre o ataque, todos os eventos e dados foram registrados, permitindo assim uma análise mais detalhada e o mais importante, o invasor não teve acesso ao ambiente real. Com essas informações e experiências é possível a adoção preventiva de medidas de segurança.

Como trabalhos futuros podem ser realizados estudos comparativos entre tipos *e* ferramentas de *honeypot.* Outra contribuição importante a este trabalho seria a adoção desse ambiente em uma rede que esteja em produção.

REFERÊNCIAS

AZEVEDO, Thiago Souza. **Honeypots - A segurança através do disfarce**. 2005. Disponível em: <http://gris.dcc.ufrj.br/documentos/artigos/honeypots-a-seguranca-atraves-do-disfarce/view> Acesso em: 27 nov. 2014.

BRASIL, Tribunal de Contas da União. **Boas práticas em segurança da informação**. Brasília: TCU, Secretaria de Fiscalização de Tecnologia da Informação, 2012. 108 p.

COELHO, Flávia Estélia Silva; ARAÚJO, Luiz Geraldo Segadas de; BEZERRA, Edson Kowask. **Gestão da segurança da informação: NBR 270001 e NBR 27002**. Rio de Janeiro: RNP/ESR, 2014. 198 p.

FILIPPETTI, Marco Aurélio. **CCNA 4.1: Guia Completo de Estudo**. Florianópolis: Visual Books, 2008. 478 p.

GIAVAROTO, Sílvio César Roxo; SANTOS, Gerson Raimundo dos. **Backtrack Linux:** Auditoria e Teste de invasão em redes de computadores. 1. ed. Rio de Janeiro: Editora Ciência Moderna Ltda, 2013.

HERTZOG, Raphaël; MAS, Roland. **The debian administrator's handbook**. 1. ed. La Talaudière, França: Freexian SARL, 2012.468 p.

EQUIPE IPV6.BR. **Cabeçalho**. 2012. Disponível em: <http://ipv6.br/entenda/cabecalho/>. Acesso em: 27 nov. 2014.

KUROSE, James F; ROOS, Keith W. **Redes de computadores e a Internet: uma abordagem top-down**. 5. ed. São Paulo: Addison Wesley, 2010. 614 p.

LUCHI, Deivid. **Ataques brute force com o Hydra**. 2013. Disponível em: <http://www.brutalsecurity.com.br/2013/05/ataques-brute-force-com-o-hydra.html>. Acesso em: 27 nov. 2014.

MARCELO, Antonio; PITANGA, Marcos. **Honeypots - a Arte de Iludir Hackers**. 1. ed. Rio de Janeiro: BRASPORT, 2003. 98 p.

MARCELO, Antônio. **Honeypots no Linux**. Linux Magazine. São Paulo. v. 11, p. 68-73, 2005.

MARTINELO, Clériston Aparecido Gomes; BELLEZI, Marcos Augusto. **Análise de Vulnerabilidades com OpenVAS e Nessus**. T.I.S., São Carlos, v. 3, n. 1, p. 34-44, 2014.

NAKAMURA, Emilio Tissato; GEUS, Paulo Lício de. **Segurança de Redes em AmbientesCooperativos**. São Paulo: Novatec, 2007. 488 p.

NBR ISO/IEC 27001; **Tecnologia da Informação. Sistema de Gestão da Segurança da Informação**. Rio de Janeiro: ABNT, 2006.

PAULI, Darren. **Security chap writes recipe for Raspberry Pi honeypot network**: Cunning security plan: dangle £28 ARM boxes and watch crooks take the bait. 2014. Disponível em: <http://www.theregister.co.uk/2014/08/01/bust_comment_crew_with_this_armada_of_raspberry_pi_honeypots/>. Acesso em: 27 nov. 2014.

PERBONI, Marcos. **SSH: Protocolo seguro para acesso remoto**. 2013. Disponível em: <http://marcosvperboni.wordpress.com/2013/02/15/ssh-protocolo-seguro-para-acesso-remoto/>. Acesso em: 27 nov. 2014.

RASPBERRY PI FOUNDATION. **What is a raspberry?** Disponível em: <http://www.raspberrypi.org/help/what-is-a-raspberry-pi/>. Acesso em: 27 nov. 2014.

WALLEN, Jack. 10 **benefits of virtualization in the data center**. 2013. Disponível em: <http://www.techrepublic.com/blog/10-things/10-benefits-of-virtualization-in-the-data-center/>. Acesso em: 27 nov. 2014.