Flávio Túlio Côrtes dos Santos Machado

Implementação de *Port Knocking* com FWKnop

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Orientador
Prof. Sandro Melo

Lavras
Minas Gerais - Brasil
2007

Flávio Túlio Côrtes dos Santos Machado

Implementação de *Port Knocking* com FWKnop

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Aprovada em *22 de Setembro de 2007*

Prof. Heitor Augustus Xavier Costa

Prof. Samuel Pereira Dias

Prof. Sandro Melo
(Orientador)

Lavras
Minas Gerais - Brasil

Agradecimentos

Agradeço à minha esposa e filho pela compreensão, nem sempre fácil, das minhas ausências a comprimissos de família enquanto escrevia freneticamente a monografia ou quando passava horas na frente do computador escrevendo para os forums ou fazendo os trabalhos. É por vocês que eu me dediquei e de onde tirei forças para continuar.

Agradeço, também, aos professores do curso de Administração de Redes em Linux, que provam a cada dia que Software Livre não é apenas moda mas um modelo que merece estar presente e que se mostra cada vez mais importante na preservação da liberdade de uso e criação quando as outras alternativas levam sempre à dependência e limitação.

Ao professor João Pincovscy, que me auxiliou na revisão e sugeriu ótimas idéias para a estrutura do trabalho, obrigado. O trabalho ficou bem melhor por sua causa.

Em especial, agradeço ao professor Sandro Melo por sugerir este tema pelo qual me encantei e que espero que seja muito útil a todos. Sua oritenação foi crucial para facilitar o trabalho e permitir que eu atingisse as metas e sua paixão pelo tema Segurança foi uma fonte de inspiração. O seu convite para o capítulo do livro foi o maior elogio que um aluno poderia receber.

Agradeço, por fim, à professora Simone Pech, por entender que a mudança do tema da monografia era importante para mim.

Resumo

Esta monografia descreve a implementação do software FWKnop, utilizando a técnica de SPA - *Single Packet Authorization*. SPA é uma evolução da técnica de *Port Knocking* tradicional com envio de um único pacote e, geralmente, com uso de criptografia. O FWKnop implementa o SPA com uso de criptografia assimétrica e simétrica, grande flexibilidade de configuração, passagem de comandos, múltiplas configurações para um mesmo servidor e controle de segurança para múltiplos usuários. Este software se mostra bastante completo e deve fazer parte do arsenal de segurança de qualquer Administrador de Redes.

Esta monografia utiliza experimentos com duas topologias de rede distintas para demonostrar a capacidade de adaptação do FWKnop para atender às várias demandas de proteção de serviços que são necessárias ao Administrador de Redes na construção da segurança das redes administradas por eles. As topologias são *Multi-Homed* e *Screened Subnet*, sendo demonstradas várias formas de configuração do FWKnop durante os experimentos. Através dos exemplos apresentados é possível compreender o funcionamento do FWKnop e sua interação com o servidor onde é instalado, incluindo como configurar o firewall para possibilitar a sua atuação e como verificar se os comandos atuaram da forma desejada.

A todos os que me ensinaram e me ensinam, obrigado.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Port Knocking é uma técnica para proteger serviços em um servidor através de pacotes de rede enviados a ele em portas não abertas e sem resposta ao cliente mas que, seja pelo conteúdo do pacote ou pela seqüência destes pacotes, tragam a informação do serviço a ser acessado. A idéia principal é não permitir que qualquer um possa acessar os serviços disponibilizados no servidor, exceto os que conhecerem o método de autorização.

A origem do termo *Port Knocking* vem do inglês e pode ser traduzido como 'batendo na porta' em uma alusão ao método de obter acesso a reuniões em locais fechados onde a autorização para entrar era feita por sequências de batidas na porta e seus intervalos, pré-combinadas entre as pessoas convidadas. A analogia é apropriada ao contexto de acesso a portas de serviços em redes IP, onde a sequência de pacotes recebidos com destino a certas portas permite a autorização para acesso ao serviço protegido.

Além de proteger serviços, algumas implementações permitem executar quaisquer comandos programados, sendo útil para reiniciar serviços ou servidores, ativar alguma configuração específica ou executar qualquer outro comando que não necessite de interatividade.

1.1 Objetivo

O objetivo deste trabalho é descrever a implementação do FWKnop, um forma de PortKnocking baseada em envio de um único pacote, SPA - *Single Packet Authori-*

zation(BLACK HAT, 2005), contendo os parâmetros de acesso ao serviço dentro da área de dados do pacote, com criptografia e uma série de verificações de segurança visando impedir eventuais agressores de usar o pacote enviado uma nova vez, o que é conhecido com *Replay Attack*(PIVA, 2006).

1.2 Motivação

Port Knocking permite a implementação de mais um nível de segurança sobre os existentes, permitindo maior controle sobre o acesso aos serviços. A motivação principal dessta monografia é avaliar uma técnica capaz de mitigar problemas de segurança decorrentes da disponibilização de serviços para acesso remoto, impedindo ataques a vulnerabilidades do tipo '0-day', ou seja, vulnerabilidades para as quais ainda não há correções. O nível de segurança extra provido pela técnica, além dos que eventualmente existam, permite maximizar o controle de acesso e minimizar os efeitos dessas vulnerabilidades, confinando o uso dos serviços somente a usuários confiáveis.

1.3 Metodologia

Esta monografia descreve, além da base teórica, dois experimentos com topologias de segurança diferentes para observar a adequação do FWKnop em cada caso. Em ambos experimentos, são usadas redes isoladas sem interferência externa.

No primeiro experimento é utilizada uma rede com topologia *Multi-Homed*, aplicável à redes onde um único firewall é o separador entre diversas redes e a administração é feita tanto pela interface de rede externa quanto pela interna. O objetivo deste experimento é comprovar o funcionamento do FWKnop na topologia mais conhecida e utilizada.

No segundo experimento é utilizada uma topologia *Screened Subnet*. Esta topologia é mais complexa e mais segura mas exige mais detalhes de implementação. O objetivo deste experimento é comprovar a flexibilidade de configuração do FWKnop para se adaptar a um ambiente mais complexo.

Nos respectivos capítulos dos experimentos são descritos os detalhes das topologias de rede utilizadas e a base teórica que as sustenta.

1.4 Estrutura dos Capítulos

Este trabalho está dividido da seguinte forma:

- Algums Considerações sobre Segurança - Contém definições de alguns termos e conceitos de segurança que serão importantes no contexto desta monografia;
- *Port Knocking* Tradicional versus *Port Knocking* com SPA - Contém explicações mais direcionadas para a solução tratada neste trabalho;
- Implementação do FWKnop - Contém a descrição de como foi efetuada a instalação e a configuração do FWKnop, com a topologia utilizada, serviços disponíveis e preparação dos certificados para a criptografia assimétrica;
- FWKnop em uma topologia *Multi-Homed* - Descreve uma aplicação prática do FWKnop em uma topologia de rede do tipo *multi-homed* onde o firewall é o único separador entre as redes interna, externa e DMZ;
- FWKnop em uma topologia *Screened Subnet* - Descreve uma outra aplicação prática do FWKnop em uma topologia de rede do tipo *screened subnet* onde um *firewall* deverá fazer o papel de separador entre as redes interna e periférica e outro entre a rede periférica e externa;
- Conclusão, Contribuições e Trabalhos Futuros - Trata da avaliação dos resultados obtidos com o uso do FWKnop, contribuições feitas e outros trabalhos que podem ser feitos sobre o tema.

Capítulo 2

Algumas Considerações Sobre Segurança

Segurança total é um objetivo desejado, mas praticamente inalcançável. Algum tipo de compromisso entre usabilidade e segurança tem que ser feito. Um exemplo é quando surge a necessidade de acesso remoto a serviços. Parece óbvio que um servidor que não tenha serviços expostos a acessos remotos tenha menos oportunidades de ser atacado pois as tentativas de ataque dependem do acesso físico ao local do servidor. Ainda que não sejam expostos serviços, no caso de estar exposto a redes públicas, o simples fato de ter que responder ou processar volumes extremos de tentativas de conexão pode impedir qualquer processamento na máquina, ocorrendo negação de serviço. A simples conexão a redes externas pode tornar os servidores vulneráveis a ataques mesmo que nenhum acesso seja permitido.

O acesso remoto a um servidor pode ser questão imperativa de segurança pessoal, como no caso de computadores de controle de fábrica localizados em locais insalubres. Nos casos onde for possível isolar o acesso apenas à uma rede específica, o processo é mais simples, como neste caso da fábrica. Nos casos onde o servidor precisa estar conectado a uma rede pública, o processo é mais complexo pois proteções devem ser implementadas para impedir acessos indevidos.

O processo de adição de segurança depende de uma análise onde se pondere o nível de usabilidade necessário com o nível de segurança exigido. A adição de ferramentas desnecessárias age como mais um elemento a tornar vulnerável a solução pois um dos maiores problemas de segurança é quando as ferramentas de segurança apresentam problemas ou são alvos dos ataques. Para conseguir acesso

ao servidor, passa ser necessário burlar a segurança estabelecida para protegê-lo. Quando a própria ferramenta de proteção tem vulnerabilidades, sua proteção é inefetiva e pode ser ainda mais nociva que a sua inexistência pela transmissão da falsa sensação de segurança.

Esta monografia não irá tratar da teoria de segurança, mas de uma implementação de segurança que visa auxiliar na proteção em complemento a outras técnicas. Porém, se faz necessário definir alguns termos e técnicas que serão usados no texto e são indispensáveis para a sua compreensão.

2.1 Firewall

Firewall é um conceito de segurança que indica controle de fluxo. O termo remete aos fossos de óleo quente em volta do portão principal de alguns castelos da época medieval e objetivavam controlar o fluxo de entrada, barrando os inimigos do lado de fora e permitindo a entrada dos amigos quando a ponte elevadiça era baixada. O mesmo conceito pode ser implementado por software ou hardware dedicados que objetivam restringir o tráfego nas redes, deixando os tráfegos não autorizados do lado de fora e permitindo o tráfego autorizado adentrar na rede privativa. A principal implementação se dá por software devido ao alto grau de complexidade que estes *firewalls* precisam ter para identificar o tráfego de um agressor e o tráfego legítimo. Em (O'REILLY, 2000) são descritos vários tipos de firewalls e suas implementações.

Existem várias soluções disponíveis para os mais diversos sistemas operacionais. Nesta monografia, será utilizado o NetFilter (http://netfilter.org/), com sua contraparte Iptables. NetFilter é o software de firewall existente no *kernel* do sistemas operacional Linux desde a versão 2.4, extremamente flexível e expansível. O Iptables é a ferramenta de manipulação das regras do NetFilter, cuidando da inserção, remoção e listagem das regras de controle de fluxo. Em (GHEORGHE, 2006) são mostradas formas de configuração e conceitos de NetFilter usados nesta monografia.

Alguns termos importantes para esta monografia são:

- *Tables* são as tabelas que contém as *chains* e são basicamente três: `filter`, a tabela padrão; `nat`, onde são feitas as manipulações sobre a parte IP do pacote; e `mangle` onde são manipuladas partes do cabeçalho IP do pacote;

- *Chain* é um conjunto de regras que é analisado do início ao fim ou até que seja encontrada uma coincidência com o tráfego analisado. Algumas *chains* são fixas como `INPUT`, `OUTPUT` e `FORWARD`, da tabela `filter`, respectivamente sobre tráfego destinado ao *firewall*, tráfego iniciado pelo *firewall* e tráfego que atravessa o *firewall*. Existem *chains* em outras tabelas, como `PREROUTING` e `POSTROUTING`, da tabela `nat`, respectivamente contendo as regras que atuam na chegada do pacote na pilha IP, antes da decisão de roteamento e as regras que atuam após a decisão de roteamento, depois da análise das regras da tabela `filter`. Outras são criadas pelo usuário ou por algum software que manipule as regras, como é o caso do FWKnop;

- *Rules* são as regras de controle de fluxo que devem ser analisadas e efetivadas pelo *firewall*. Se o tráfego combinar com alguma regra inserida, há um *match* ou uma coincidência e a regra atuará naquela tráfego específico. Cada pacote deve percorrer as regras até que um *match* ocorra ou que a política padrão atue. A política padrão é uma definição que diz qual o comportamento do *firewall* se não ocorrer *match*. Uma boa política de segurança é descartar o tráfego que não atingiu qualquer regra.

2.2 Criptografia

Quando se discute segurança imediatamente é preciso discutir criptografia. Criptografia diz respeito ao processo de tornar ininteligível uma informação para quem não conhece os métodos e parâmetros para recuperar ou validar a informação. Seu uso advêm da idade antiga, desde os Egípcios com a linguagem dos hieroglifos conhecida apenas pelos seus escribas, passando pelos Romanos com a cifra de Caesar, pelo código da Alemanha durante a guerra até as implementações advindas dos meios eletrônicos, conforme explica (VAUDENAY, 2006).

As implementações modernas se diferenciam das anteriores por não se basearem em obscuridade e sim na matemática. Em especial, algoritmos capazes de alterar o tamanho original da mensagem ou introduzir outros elementos e ainda assim conseguir restaurar a mensagem são mais resistentes à criptoanálise[1]. A combinação de técnicas de compactação com criptografia produz resultados com alta resistência.

[1] Criptoanálise é um ramo da criptografia que estuda a identificação da técnica de cifragem utilizada e, se possível, dos parâmetros para obtenção da mensagem original. Também é utilizada para validar os algoritmos quanto à sua resistência a ataques. Vide (VAUDENAY, 2006)

Os algoritmos modernos baseados em chaves de cifragem e decifragem são os mais importantes para comunicação e, portanto, para uso de acesso remoto. Em especial, dois típo são importantes para essa monografia:

Chave Simétrica Algoritmos de chave simétrica baseiam-se no uso de uma mesma chave para cifrar e decifrar a mensagem. Um problema relacionado ao uso de uma única chave para todo o processo é a troca segura das chaves pois elas devem estar presentes nos dois lados da comunicação. A vantagem inerente a este tipo de algoritmo é sua menor utilização de recursos computacionais.

Chave Assimétrica Algortimos de chave assimétrica baseiam-se no uso de pares de chaves, uma pública e uma privada, onde a cifragem é feita com a chave pública e a decifragem com a privada. A vantagem técnica deste tipo de algoritmo é que não é necessário conhecer uma mesma chave e não há perigo em distribuir a chave pública pois apenas a chave privada é capaz de reverter o processo. A desvantagem é que o processo é computacionalmente mais intensivo e requer um volume de recursos significativo para seu uso.

As técnicas modernas combinam os dois métodos, minimizando as desvantagens de cada um. O início da comunicação é feito com chaves assimétricas pois a chave pública pode ser conhecida sem prejuízo. Uma vez que cada parte apresentou sua chave pública, é possível gerar uma chave aleatória para ser usada só naquela comunicação e por um certo prazo e trocar a chave entre as partes. Essa nova chave, trocada de forma segura, será usada de forma simétrica, economizando recursos computacionais. Um exemplo de uso dessa técnica é o uso de criptografia combinada com o protocolo HTTP (RFC2616, 2007), o que resulta no protocolo HTTPS (RFC2818, 2000).

Além de ofuscar mensagens, a criptografia por ser usada para validá-las, com uso de *hash*. *Hashs*, ou algoritmos de espalhamento, são algoritmos matemáticos cuja função é obter um único valor de tamanho fixo baseado em um conjunto de valores de tamanho fixo ou variável de forma unidirecional, ou seja, a partir do valor final não é possível conhecer o conjunto inicial. É bastante usado para gerar assinaturas sobre valores pois qualquer modificação nos valores originais gera um valor final diferente.(DCE, 2007)

Gerando *hashs* das mensagens e cifrando-os, é possível verificar que não houve alterações no texto pois é possível decifrar o *hash* original e compará-lo com o novo, gerado no conteúdo atual. Se forem iguais, a mensagem pode ser considerada válida. Os algoritmos mais conhecidos de *hash* são MD5, SHA1 e SHA256, descritos em (CRYPTO, 2001).

2.3 Vulnerabilidades

A segurança visa proteger os serviços através da mitigação de suas vulnerabilidades. Vulnerabilidade pode representar uma falha de arquitetura ou falha de implementação que leve a um comportamento de permitir mais acesso do que deveria ser dado ou a indisponibilidade do serviço. Em outros contextos, vulnerabilidade pode significar alguma falha que permite o acontecimento de alguma ação não desejada. Nesta monografia, apenas a primeira definição será usada.

A análise das vulnerabilidades dos serviços é essencial para o projeto da segurança necessárias a mitigá-las. O administrador do serviço deve identificá-las e aprofundar o conhecimento sobre elas de modo a saber como proteger os serviços. O agressor com certeza usará alguma técnica para identificar essas vulnerabilidades e obter conhecimento de como aproveitá-las para atacar os serviços.

Uma das formas de identificar as vulnerabilidade de serviços remotos é utilizar a técnica de *Port Scan*. *Port Scan* ou Varredura de Portas é a técnica de procurar por serviços disponíveis num servidor através da tentativa de abertura de portas dos serviços desejados, objetivando analisar quais os prováveis serviços disponibilizados pelo servidor. Com esta informação, é possível direcionar os esforços apenas aos serviços disponibilizados, o que aumenta a possibilidade de sucesso do ataque ou da proteção.

Uma forma mais sofisticada é combinar *Port Scan* com ferramentas capazes de testar por vulnerabilidades conhecidas, como é o caso da ferramenta Nessus (http://www.nessus.org/). Esta ferramenta contém uma base de dados de vulnerabilidades por tipo de serviço e protocolo utilizado e permite verificar se a versão instalada no servidor é suscetível a ataques que explorem cada uma das vulnerabilidades conhecidas. É uma ferramenta poderosa na mão do administrador ou do agressor.

Uma parte importante da preparação de um ataque ou da visualização do tráfego de rede para identificar um ataque é a atividade de *sniffing*. *Sniffers* são ferramentas capazes de observar o tráfego que ocorre na rede, mesmo que não seja destinado à máquina onde ele se encontra. O termo *sniffing* vem do verbo inglês *to sniff* e pode ser traduzido no contexto por bisbilhotar. Em geral é necessário observar a rede no modo chamado de promíscuo, onde o dispositivo de rede passa para a máquina todos os pacotes que ela recebe, independente de serem destinados ou não a ela. O comportamento normal é receber todos os pacotes e descartar os que não são destinados a si.

2.4 Ataques

Uma grande variedade de ataques pode ser desferida contra servidores remotos. Nesta seção, são comentados os ataques mais difundidos que tenham relação com o tema da monografia.

2.4.1 DOS - *Denial of Service*

Ataques do tipo DOS - *Denial of Service* ou Negação de Serviço, com suas variantes DDOS - *Distributed Denial of Service* e DRDOS - *Distributed Reflected Denial of Service*, descritos em http://www.rnp.br/newsgen/0003/ddos.html e http://www.grc.com/dos/drdos.htm, respectivamente, são formas de impedir o uso legítimo com o objetivo de prejudicar o funcionamento normal ou para obter dados do servidor por alguma falha através do envio de um número expressivo de pacotes com destino ao servidor, usando boa parte da banda de comunicação disponível ou mesmo forçando o servidor além de sua capacidade. Também em (O'REILLY, 2000) é descrito este tipo de ataque.

O resultado mais comum de um ataque DOS é a indisponibilidade temporária de um serviço, durante o período do ataque ou até que alguma forma de mitigá-lo seja implementada. Em alguns casos, o dano dura até que ações de recuperação do serviço ou do servidor sejam tomadas, como é caso de gerar um grande volume de tentativas com o único objetivo de encher os arquivos de *log*[2]. A partir da sobrecarga dos arquivos de *log*, o agressor pode conseguir realizar suas atividades de análise de vulnerabilidades ou propriamente os ataques, sem deixar rastros.

2.4.2 *Spoofing*

Spoofing, descrito em (SPOOFING, 2003), ocorre quando se altera a marcação de origem de um pacote, objetivando que a máquina de destino interprete a fonte de envio do pacote como sendo outra.

O objetivo do *spoofing* é enganar o servidor para parecer que uma outra fonte, que não o agressor, enviou o pacote. O ataque de DRDOS, por exemplo, se utiliza de *spoofing* para gerar pacotes que se originem de várias máquinas, mas cujo endereço de origem seja alterado para o do servidor a ser atacado. Quando as

[2] *Log* objetiva registrar informações de atividades. No caso de servidores, um sistema de *log* mantém registro das informações enviadas pelo sistema operacional ou pelos serviços

várias conexões iniciadas pelas máquinas agressoras em vários servidores tiverem seu tráfego de retorno direcionadas para o servidor atacado, um grande volume de tráfego indevido poderá causar problemas para ele.

Existem usos benéficos para a técnica de *spoofing* como o que ocorre no caso de NAT - *Network Address Translation* ou Tradução de Endereço de Rede, que é uma técnica usada para trocar a origem, destino ou ambos em um pacote IP. A técnica de NAT de origem é exatamente a mesma do *spoofing*.

É importante ressaltar que se todos os pontos de interconexão de rede impedirem a saída de tráfego originado de sua rede tenham no campo de origem um endereço que não pertence à rede, o nível de complexidade para efetuar algum ataque de *spoofing* seria muito maior, conforme relata (MEIJER, 2001).

2.4.3 *Replay Attack*

Replay Attack ou ataque de repetição é uma forma de tentar obter acesso ou induzir uma ação no servidor baseado no envio de um tráfego anterior capturado por um *sniffer*. O tráfego capturado pode ser alterado para parecer que foi enviado de uma outra fonte (*Spoofing*).

Este tipo de ataque é a principal preocupação quando o assunto é *Port Knocking*, pois pode representar um dos poucos ataques capazes de burlar sua proteção. Além de *Replay Attack*, apenas um ataque de DOS pode trazer algum efeito às soluções de *Port Knocking*.

Capítulo 3

Port Knocking Tradicional versus *Port Knocking* com SPA

Não sendo possível operar sob o manto da proteção total, os esforços devem ser feitos no sentido de prover a melhor segurança possível dentro do limite de riscos aceitável para a solução. Um serviço HTTP só é útil quando pode ser acessado remotamente mesmo quando é sabido que há ataques possíveis. A segurança deve atuar para impedir o sucesso dos ataques ou minimizar os seus efeitos de forma a torná-los pouco efetivos.

A implementação de segurança é realizada, normalmente, em camadas. Quanto maior o número de camadas e quanto maior seu potencial de resistir a ataques, maior o nível de segurança alcançado. Uma analogia possível é a forma de uma cebola, onde camadas sucessivas envolvem as camadas mais internas, como mostra a figura 3.1 demonstrando a analogia.

Como não existe autenticação nem criptografia diretamente no protocolo IPv4 padrão, técnicas alternativas para obter formas de autenticação e ocultamento não dependentes de modificações em aplicações começaram a ser elaboradas. Em (BARHAM, 2002), havia uma preocupação em criar novos protocolos e técnicas para ocultamento de serviços e permitir autenticação pelo conhecimento da forma de acessá-los.

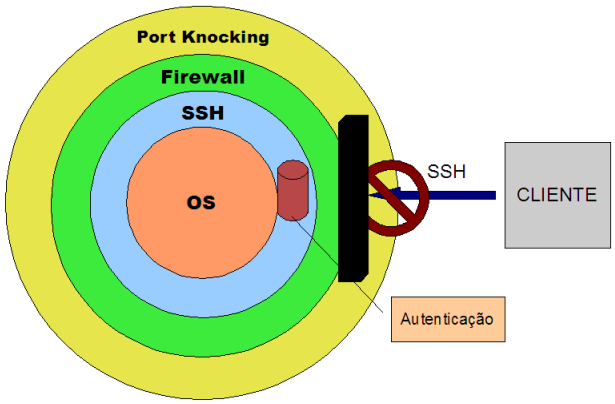

Figura 3.1: Camadas em um ambiente de negação por padrão

3.1 Implementações de Port Knocking

Existem várias implementações de PortKnocking, variando entre si em relação à simplicidade, funcionalidade e nível de segurança providos. Algumas dessas variantes serão mencionadas a seguir.

3.1.1 Módulo *recent* do *NetFilter*

O NetFilter (WELTE, 2000) é um *firewall* do tipo *statefull* incorporado ao *kernel* do Linux. Um dos seu módulos, `recent`, permite uma implementação básica de *Port Knocking* com uma ou múltiplas portas.

A facilidade em obter a sequência de ativação é a principal limitação deste método. Cada conexão estabelecida na porta do serviço será precedida de uma sequência de pacotes para as portas monitoradas pelo `recent`. Como a seqüência não muda e não há criptografia, a análise de tráfego será bastante efetiva em determinar a forma de acesso ao serviço e permitir um *Replay Attack* [1]

[1] Conforme descrição no capítulo 2, seção 2.4.3

3.1.2 Knockd

A solução baseada no *software* Knockd, implementado em (KRZYWINSKI, 2003), permite uma configuração mais simples, sem ter a preocupação em fazer a seqüência de `recent` usada pelo NetFilter. Além disso, não só a abertura de regras é possível, a execução de comandos pré-programados, associados a seqüências específicas, também pode ser efetivada.

Assim como o `recent`, não há criptografia e há pouca ou nenhuma variação de portas, o que pode permitir a descoberta da sequência. Além disso, tanto no caso do `recent` quanto do Knockd, o envio de sequências de pacotes para várias portas dentro de um curto espaço de tempo pode disparar alarmes dos sistemas de detecção de intrusos e provocar um falso positivo ou mesmo ter o tráfego barrado.

3.1.3 Tumbler

Uma evolução significativa na técnica de *Port Knocking* é o uso de um único pacote para prover a autorização, o que passou a ser conhecido como SPA - *Single Packet Authorization*. Uma das implementações existentes é o Tumbler, descrito no artigo (GRAHAM-CUMMING, 2004). Assim como o Knockd, é possível executar comandos e não apenas permitir acesso a serviços.

A vantagem do Tumbler em relação às implementações anteriores é o uso de criptografia, na forma de verificação de um *hash* do conteúdo da área de dados do pacote recebido. Não é feita a decifração do conteúdo, apenas é verificada a validade do *hash*. A ação a ser executada está pré-progamada e vinculada à cada porta monitorada.

Várias proteções são usadas para impedir *Replay Attacks*, através do uso de criptografia e impedimento do recebimento de mais de um pedido de autorização dentro do mesmo minuto. Como a menor unidade de tempo usada na geração do *hash* é um minuto, não é possível reutilizar o mesmo pacote em *Replay Attack*

A desvantagem do Tumbler reside na dependência de senhas para validar os *hashs*. Como a senha tem que estar pré-cadastrada no servidor, é necessário trocá-la constantemente e lembrar de atualizar os respectivos arquivos, o que pode se tornar um problema adminsitrativo no caso de uso em vários servidores.

3.1.4 FWKnop

A implementação mais completa existente da técnica de SPA é o FWKnop - *Firewall Knock Operator*, descrita inicialmente em (RASH, 2004) e posteriormente em (RASH, 2006). Similar ao Tumbler, apenas um pacote é necessário para ativar a autorização. Diferentemente do Tumbler, o pacote é cifrado com chaves assimétricas e é decrifrado para verificação. É possível o uso de outras técnicas, mas o uso de criptografia assimétrica é o grande diferencial do FWKnop. Assim como o Knockd e o Tumbler, é possível executar comandos e não apenas permitir acesso a serviços.

Assim como o Tumbler, a menor unidade de tempo é um minuto e não é possível o uso do mesmo pacote dentro do mesmo minuto, impedindo o *Replay Attack*. Além dessa proteção, a análise compara o IP do pacote com o IP contido dentro da área cifrada, impedindo o uso de *Spoofing*.

3.1.5 PortKnockO

Existe um módulo criado como uma extensão do NetFilter chamado PortKnockO (http://portknocko.berlios.de/br/index.html), com sua respectiva contraparte no programa iptables. Esse módulo permite uma variante do método tradicional do módulo `recent`, onde há os dois modos: *Port Knocking* tradicional, com sequência de portas, limitado ao protocolo TCP; SPA, com assinatura feita por sha256. O limitador dessa técnica é, como no módulo `recent`, apenas regras de firewall serem criadas e a chave ser simétrica e permanente, ou seja, uma senha ainda deve ser mantida dos dois lados, diferentemente do modo assimétrico do FWKnop.

3.2 Análise do Funcionamento de *Port Knocking*

Muito se discute se as técnicas e Port Knocking não são apenas uma forma de Segurança pela Obscuridade como discutido em (SCHNEIER, 2002), mas é mais correto afirmar que se trata de uma camada extra à segurança existente, uma forma de *Concealment* [2], como confirma (SEBASTIEN, 2006). Porém, é importante que ela não seja a única forma de segurança.

[2] *Concealment*, ou ocultamento, descreve a técnica de manter os serviços fora da visibilidade normal de forma a que somente quem conhece a forma específica possa acessar estes serviços.

A idéia do Port Knocking é estabelecer esta camada sobre o firewall e controlar a existência da regra que permite acesso ao serviço, como o SSH na figura 3.1. A existência ou não da regra depende diretamente da atuação de um *knock*.

Uma vez que seja enviado o *knock*, o comportamente é como o demonstrado a seguir, onde a regra permitindo o acesso ao serviço foi criada após a identificação da ação desejada pelo *knock*, como mostra a figura 3.2.

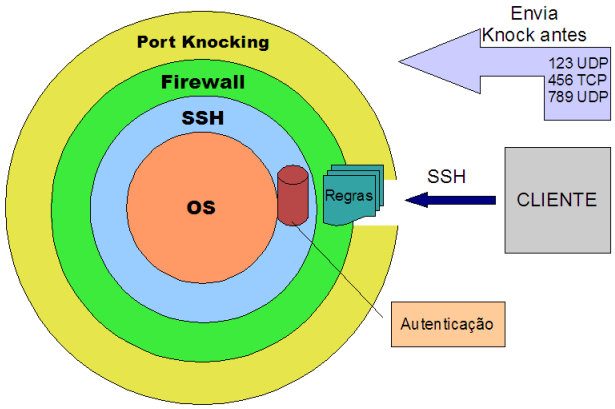

Figura 3.2: Após a autorização, uma camada de proteção aberta

Para cada serviço que seja protegido pela técnica de *Port Knocking*, será necessário um ciclo de envio do *knock* seguido do acesso ao serviço.

As soluções mencionadas no início deste capítulo mostram um panorama das variantes de implementação de *Port Knocking* existentes. É possível classificá-las em duas categorias: envio de sequências, o modo mais tradicional de *Port Knocking*; e envio de único pacote, o método mais avançado.

3.2.1 *Port Knocking*

As soluções da categoria de envio de sequências sofrem de alguns males críticos, a saber:

- São mais vulneráveis a *Replay Attack*[3] pois a sequência é chave para autorização;
- Raramente implementam criptografia, o que torna a segurança mais fraca;
- Demandam mais tempo para permitir o acesso pois há necessidade de esperar a sequência completa;
- Tem problemas quando a sequência de pacotes chega fora de ordem;
- Devem ter os comandos pré-progamados, pois a seqüência atua como identificador do comando a ser executado;
- É possível implementar comunicação codificada na própria sequência, mas o número de pacotes seria muito elevado e certamente se pareceria com um *port scan*[4].

A maior vulnerabilidade da técnica de *Port Knocking* diz respeito a *Replay Attacks*. Como a autorização para acesso aos serviços passa a depender da camada do *Port Knocking*, passa a ser indispensável para um agressor conseguir alguma forma de obter o acesso, em geral por meio da repetição do tráfego previamente usado de modo a tentar obter este acesso. Como analisa (SEBASTIEN, 2006), a forma segura de implementar técnicas de *Port Knocking* deve envolver criptografia e verificação de autenticidade do pacote utilizado.

Uma vantagem natural das soluções baseadas em *Port Knocking* é serem geralmente portáveis entre sistemas operacionais diferentes e terem implementação simples, pois o envio de sequências de pacotes é facil de ser programado. Especialmente no caso dos clientes que fazem envio dos knocks, esta é uma vantagem importante.

3.2.2 *Single Packet Authorization*

Nesta categoria, onde se usa um único pacote, há vantagens inerentes:

[3] vide o capítulo 2, seção 2.4.3
[4] vide o capítulo 2, seção 2.3

- Não há sensibilidade a reordenação de pacotes, pois um único pacote é usado;
- É mais curto o tempo entre o envio do pacote e a liberação de acesso ao recurso, pois basta verificar um único pacote;
- Como é usada a área de dados ao invés do cabeçalho IP, um volume maior de informações pode trafegar, incluindo autenticações, informações de data e hora, qual usuário está solicitando a autorização de acesso, entre outras;
- Com um volume maior de informações, é possível usar criptografia do conteúdo, tornando mais seguro ainda o processo de autorização.

As duas implementações mais conhecidas nesta categoria são Tumbler e FWKnop. Embora tenham características similares como o uso de criptografia, inclusão de informações de IP de origem e data e hora, a implementação do FWKnop é mais completa e segura, pois:

- Usa-se criptografia com chaves assimétricas, garantindo a individualidade dos usuários e a rastreabilidade de suas ações, bem como um nível de segurança maior;
- É possível passar comandos e parâmetros dentro do pacote, permitindo uma flexibilidade sem precedentes em comparação com as outras ferramentas, inclusive o Tumbler, que exigem a pré-programação das ações;
- A adição de um bloco de conteúdo randômico gerado para cada pacote permite garantir que a criptografia de cada pacote seja única e possa ser armazenada em cache por um certo período impedindo o reuso do pacote em um *Replay Attack* pois cada pacote é comparado ao cache antes de ser permitido.

O software FWKnop permite o uso do melhor das outras soluções, sem suas limitações. Duas de suas características são especialmente importantes: criptografia assimétrica e possibilidade de execução de comandos não-programados. A segunda característica é difícil de ser implementada nas outras soluções apresentadas, pois geraria um volume muito grande de pacotes ou traria riscos de segurança para permitir seu uso. O Tumbler não permite o uso dessa característica, pois só autentica o conteúdo do pacote e o pacote é assinado e verificado criptograficamente mas não é decodificado, impedindo a passagem de informações do cliente para o servidor, necessárias para entregar o comando e seus argumentos para execução.

Capítulo 4

FWKnop

Neste capítulo, são descritas a instalação e a configuração básica do FWKnop, mostrando algumas de suas características.

4.1 Obtenção e Instalação do FWKnop

O *site* do FWKnop é o http://www.cipherdyne.org/fwknop. Documentação e pacotes de instalação podem ser obtidos a partir dele, bem como referências a artigos, apresentações e tutoriais. Existem vários artigos, como (RASH, 2007b) e (RASH, 2007a), que descrevem o SPA e a utilização do FWKnop.

No *site* estão disponibilizados os fontes *tarball*[1], RPMs[2] com binários para x86 e x86_64[3], bem como Source RPMs[4].

A distribuição utilizada para os testes é Fedora (http://fedoraproject.org/), versão 7, com atualizações. O projeto Fedora é a versão comunitária da distribuição

[1] *tarball* é um nome genérico para um conjunto de arquivos agregados com o uso do *tar* e compactados com *gzip* ou *bzip2*, por exemplo

[2] RPM ou RedHat Package Manager é um gerenciador de pacotes que controla a instalação de softwares em distribuições Linux RedHat ou derivadas. O termo é costumeiramente usado para indicar um arquivo com extensão `rpm` que será tratado pelo software RPM.

[3] x86 é um termo que descreve a arquitetura de processadores baseadas nas especificações da Intel e implementadas pela Intel, AMD e outras. x86_64 se refere à extensão de 64bits para a arquitetura x86, desenvolvida pela AMD.

[4] Source RPMS indicam arquivos RPM contendo códigos-fonte e descrição da compilação que, uma vez compilados e preparados, geram o arquivo RPM para instalação

RedHat, que tem foco corporativo, onde as novidades são testadas no Fedora e, após maturação, passam a ser incorporadas na RedHat.

Durante a instalação, o pacote RPM binário para x86 apresentou problemas devido à dependência do libpcap na versão 0.9.4, enquanto a versão instalada era a 0.9.5. Para resolver o problema, foi usado o arquivo Source RPM que foi recompilado com o comando `rpmbuild -rebuild fwknop-1.8.1-1.src.rpm`, obtendo o novo RPM binário para instalação, que foi usado com sucesso. Para fazer a compilação, é necessário ter instalados os pacotes `gcc`, `make`, `lipcap`, `libpcap-devel`, `Perl Net::PCap` e, no caso tratado nesta monografia, `gpg` e `gpg-devel`. Para distribuições que não usam RPM, o pacote fonte tem um instalador em linguagem Perl que realiza os passos para preparar e instalar os programas, mas as dependências deverão ser satisfeitas antes de executar o instalador. Este modo de instalação consiste basicamente em executar o comando `./install.pl` no diretório onde tiver sido descompactado o *tarball*.

Após a instalação, será criado um diretório `/etc/fwknop` contendo os arquivos de configuração do fwknop, em especial o `fwknop.conf` e o `access.conf`. Os binários do servidor `fwknopd` e o cliente `fwknop` serão instalados também, bem como os respectivos arquivos para o serviço de inicialização.

É importante ressaltar que o servidor só pode ser instalado em sistema operacional Linux com NetFilter ou BSD com IPFW a partir da versão 1.8.0. O cliente pode ser instalado em FreBSD, OSX, Linux e Windows, este último a partir da versão 1.8.0.

4.2 Configuração

Antes de qualquer outra ação, a preparação das regras de iptables é essencial. Para que seja possível obter a proteção de abertura seletiva de portas que o FWKnop pode prover, é necessário atribuir regras restritivas para os tráfegos protegidos, seja por regras explícitas seja por definição de políticas padrão, como exemplificado na figura 4.1.

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
```

Figura 4.1: Políticas padrão

Após definidas as políticas e demais regras necessárias ao servidor, como as regras que permitem a saída e entrada de tráfego `ESTABLISHED`[5], o FWKnop inserirá uma nova *chain*[6] que conterá exclusivamente as regras do FWKnop, facilitando a inclusão e exclusão de regras pelo `fwknopd`. O nome dessa *chain* é configurável no arquivo `fwknop.conf`. Quando ocorre a ativação por algum pacote SPA que envolva a criação de regras, estas são criadas na *chain* do fwknop e, posteriormente, removidas por *timeout* ou envio de outro pacote SPA.

Todos os parâmteros de configuração do daemon `fwknopd`, exceto as configurações de serviços, estão contidos no arquivo `/etc/fwknop/fwknop.conf`. Alguns parâmetros só são especificados neste arquivo, outros parâmetros especificam os *defaults* que podem ser sobrescritos na configuração de cada serviço no arquivo `/etc/fwknop/access.conf`.

Em especial, os parâmetros indicam qual o ambiente a ser usado, como qual o software de firewall, qual a interface, protocolo e porta a serem monitorados, bem como qual o método usado, qual o *hostname* a ser usado nos pacotes, uso do endereçamento IP do cliente dentro do pacote ou não, entre vários outros. Esse deve ser o primeiro arquivo a ser ajustado, antes de definir os serviços. As diretivas mais importantes do arquivo `fwknop.conf` são:

- `EMAIL_ADDRESSES` - Especifica o *email* do administrador ou responsável pelo fwknop e será usado para enviar alertas de tentativas de *replay* de pacotes ou falhas de autenticação;
- `HOSTNAME` - Especifica o nome a ser usado como *hostname* da máquina;
- `FIREWALL_TYPE` - Especifica que tipo de firewall deve ser usado, por *default* é `iptables`, pode ser ipfw, se o sistemas operacional for BSD;
- `AUTH_MODE` - Especifica o modo de captura dos pacotes para análise, por *default* `PCAP`;
- `PCAP_INTF` - Especifica a interface onde deve ser feita captura dos pacotes, por *default* `eth0`;
- `ENABLE_PCAP_PROMISC` - Especifica se a captura deve ser feita em modo promíscuo, *default* `Y`;
- `PCAP_FILTER` - Especifica que protocolo e porta devem ser buscados no tráfego capturado, por *default* `udp port 62201`;

[5] Conforme capítulo 2, seção 2.1.
[6] Conforme capítulo 2, seção 2.1.

- `ENABLE_SPA_PACKET_AGING` - Especifica se deve ser verificado o tempo do pacote (exige sincronia dos relógios), por *default* `Y`;
- `MAX_SPA_PACKET_AGE` - Especfifica o tempo máximo pelo qual o pacote ainda será aceito, por aem default `120`;
- `ENABLE_MD5_PERSISTENCE` - Especifica se deve manter os últimos *hashs* MD5 para verificação de *replay*, por *default* `Y`;
- `MAX_HOPS` - Especifica quantos *hops*[7] no máximo o cliente pode estar de distânca, por *default* `20`;
- `ENABLE_TCP_SERVER` - Especifica se deve ser habilitado o servidor TCP [8], por *default* `N`;
- `TCPSERV_PORT` - Especifica, se estiver habilitado o servidor TCP, qual porta deve ser usada. Por *default* `62201`;
- `SYSLOG_DAEMON` - Especifica qual o *daemon* de *log* utilizado, por *default* `syslogd`;

Uma vez definidos os parâmetros de operação, atividade que provavelmente será feita uma única vez por servidor, se faz necessário definir os serviços a serem monitorados.

O FWKnop traz por padrão uma configuração simples para proteção de um serviço de SSH. Para esta configuração, basta alterar os valores que vêm no arquivo `/etc/fwknop/access.conf`, em especial `KEY` e as variáveis de suporte a GPG, se forem utilizados. É possível usar a forma tradicional de *Port Knocking*. Não será tratada nesta monografia outra forma que não o modo SPA.

Para situações simples, a configuração *default* é totalmente funcional e capaz de proteger o serviço SSH completamente. Apenas após e envio do pacote específico será aberta a regra de permissão de acesso à porta do SSH configurada no arquivo, 22 no caso *default*. O acesso só será permitido pelo tempo estabelecido em `FW_ACCESS_TIMEOUT`, 30 segundos na configuração *default*.

Pode-se notar que a configuração dos casos com chaves simétricas é composta de poucas diretivas e, por *default*, vem pré-configurado na instalação, com exceção

[7] Um *hop* ou salto acontece quando uma decisão de roteamento é tomado em algum ponto do tráfego, significando que algum elemento roteou o pacote entre redes. O número de hops indica por quantas redes o pacote passou.

[8] Embora o padrão seja o uso de UDP, para o caso de combinar o fwknop com a rede TOR(http://tor.eff.org/), é necessário o uso de TCP, como exemplifica (DEFCON, 2006)

da DIRETIVA `KEY` que deve ser alterada. Para uma configuração de acesso a um serviço de única porta, como SSH, basta especificar um bloco semelhante ao da figura 4.2.

```
SOURCE: ANY;
OPEN_PORTS: tcp/22;
FW_ACCESS_TIMEOUT: 30;
KEY: m1nh4ch4v3;
```

Figura 4.2: Configuração para SSH, modo SPA, chave simétrica

Para cada serviço, deve ser definido um bloco iniciando com `SOURCE` e, pelo menos até a versão 1.8.1, é obrigatória a presença da diretiva `OPEN_PORTS`. Cada serviço deve ter uma chave própria, pois o fwknopd fará a associação do serviço (composto da porta e protocolo) a cada bloco e usará a respectiva chave para a criptografia. Quando for executado o envio do pacote pelo cliente, a porta e o protocolo de destino indicarão qual bloco de configurações procurar e o bloco cifrado será decifrado no servidor com a chave associada, que deve ser igual à usada pelo cliente ou não será aceita a mensagem SPA. Nenhuma parte da área de dados do pacote trafegará em claro, pois será usada criptografia Rijndael[9].

Após a configuração do serviço e a inicialização do fwknopd, tentativas de acesso ao serviço SSH serão bloqueadas até que seja enviado o pacote SPA com a especificação correta da chave, no caso `m1nh4ch4v3`. Uma vez autorizado o acesso, será criada uma regra permitindo a conexão ao SSH a partir do computador cujo IP de origem tenha sido designado no interior da mensagem SPA. Após o número de segundos estabelecidos pelo parâmetro `FW_ACCESS_TIMEOUT`, a regra será removida e somente será permitida um novo envio de pacote SPA no minuto seguinte ao do pacote anterior.

O nível de segurança dessa configuração é bastante elevado pois a criptografia, mesmo simétrica, envolve todo o conteúdo do pacote. A desvantagem desta técnica é a necessidade de troca da chave constantemente para diminuir a chance de quebra. A troca tem que ser feita no servidor e no cliente, o que requer bastante atenção para evitar perda de conectividade durante a troca, o que poderia ocasionar que a troca não ocorra em um dos lados ou alguma solução de troca das chaves de

[9] O algoritmo criptográfico Rijndael (http://www.csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf) é considerado bastante seguro e foi escolhido para se tornar o AES - Advanced Encryption Standard, padrão americano de criptografia, adotado também internacionalmente pela sua qualidade e resistência a quebra

forma automática e constante, que deve funcionar em ambos os lados. Em ambos os casos, é imprescindível fazer a troca de chaves de forma segura e confiável.

A segunda forma de configuração envolve o uso de criptografia assimétrica com o uso do GPG - GNU Privacy Guard (http://www.gnupg.org), uma implementação livre do protocolo OpenPGP, especificado em (RFC2240, 1998). Especificamente, é usado o algoritmo ElGamal[10] de até 2048 bits, o que torna a solução extremamente resistente a ataques de força bruta. Para uso do gpg, é necessária a versão `0.9.6` ou superior do FWKnop, sendo que a última versão disponível e utilizada para a preparação deste trabalho era a 1.8.1.

A grande vantagem do uso de criptografia assimétrica é não existir troca das senhas e é mais completa a identificação do cliente, pois a sua chave pública será cadastrada no servidor e vice-versa, sendo necessário assinar as chaves GPG do cliente com a chave privada do servidor. Como a assinatura da chave do cliente é um processo que tem que ser feito no local onde se encontre a chave privada, é certo que quem assinou conhece o receptor da chave cliente.

Como o fator tempo é importante, pois faz parte do conteúdo do pacote, torna-se necessário manter os relógios dos clientes e do servidor em sincronia, assim como do servidor com alguma fonte confiável na Internet. O mais importante, porém, é a sincronia entre clientes e servidor. Além disso, o *log* do sistema será mais útil se estiver sincronizado por retratar mais fielmente a sequência de ações ocorridas em cada lado. Isto é importante para detectar erros e tentativas de ataques. Por *default* o `fwknopd` aceita um pacote com diferença de até 120 segundos, seja por demora de entrega seja por pequenas diferenças nos relógios.

Um bloco de configuração para o modo GPG que permita a execução de comandos é semelhante ao da figura 4.3.

Existem alguns detalhes importantes a serem observados na figura 4.3, a saber:

- Embora não sejam usadas portas, pois o comando não envolverá regras de *firewall*, o *parser* exige a presença de `OPEN_PORTS`. Segundo o autor do FWKnop, a próxima versão não necessitará dessa diretiva, em situações semelhantes;

[10] Algoritmo ElGamal é assimétrico, implementa o uso de chaves públicas e privadas e é considerado bastante seguro, conforme (CRYPTO, 2001)

```
SOURCE: ANY;
OPEN_PORTS: tcp/22;
PERMIT_CLIENT_PORTS: N;
DISABLE_FW_ACCESS: Y;
ENABLE_CMD_EXEC: Y;
DATA_COLLECT_MODE: PCAP;
GPG_REMOTE_ID: 1234ABCD;
GPG_DECRYPT_ID: ABCD1234;
GPG_DECRYPT_PW: M1nhaS3nhAGpG;
GPG_HOME_DIR: /root/.gnupg;
FW_ACCESS_TIMEOUT: 60;
```

Figura 4.3: Configuração para execução de comandos, modo SPA, chave assimétrica

- `PERMIT_CLIENT_PORTS` é usado para avisar ao *daemon* `fwknopd` que não deve interpretar alguma lista de portas a serem abertas, pois não serão usadas regras de *firewall*, por causa da diretiva a seguir;

- `DISABLE_FW_ACCESS` é essencial, justamente para não implantar regra para a porta especificada por `OPEN_PORTS`;

- `ENABLE_CMD_EXEC` é a diretiva que permite execução de comandos no servidor;

- Se for necessário reduzir o escopo dos comandos que podem ser executados, pode ser usada a diretiva `CMD_REGEX` que tem como parâmetro uma expressão regular que será comparada ao comando enviado para permitir ou não sua execução;

- As diretivas iniciadas por GPG descrevem os parâmetros de uso dos certificados GPG a serem usados na criptografia assimétrica.

Algumas diretivas não se aplicam ao modo SPA e, portanto, não são tratadas por esta monografia. Algumas diretivas que ainda não foram mencionadas, mas que estão disponíveis em modo SPA, são:

- `REQUIRE_SOURCE_ADDRESS` - O uso desta diretiva implica que é necessário o envio do IP de origem dentro do pacote, o que causa problemas com *NAT*. O IP externo do pacote será comparado com o IP interno e obrigatoriamente devem ser iguais para que o pacote seja aceito. Para resolver

o problema, é preciso conhecer o IP externo após o *NAT* ou usar a opção -R (ou -w) para obtê-lo através de uma chamada ao *site* WhatsMyIP (http://www.whatismyip.com/);

- `KNOCK_INTERVAL` - Esta diretiva permite estabelecer o mínimo intervalo de tempo dentro do qual tentativas de *knock* serão rejeitadas. Por *default* o intervalo é de 60 segundos, o que significa que novas tentativas de *knock* dentro de um minuto não serão aceitas.

Pode-se observar a flexibilidade de configuração de serviços no FWKnop. Múltiplos blocos de configuração podem ser usados no mesmo arquivo, cada um com suas especificidades como: permitir executar ou não comandos, usar criptografia simétrica ou assimétrica, que portas abrir com regras de firewall, etc. Nos próximos capítulos, são mostradas duas situações distintas: no capítulo 5 será mostrada a forma de configuração para acesso ao próprio firewall que estará em uma topologia de *Dual-Homed Firewall*; no capítulo 6 será mostrada uma configuração mais complexa em uma topologia *Screened Subnet*

4.3 Pacote SPA

Um pacote SPA é composto pelos campos descritos na figura 4.4, nesta ordem, separados pelo caracter `':'`.

```
random number (16 bytes)
username
timestamp
software version
mode (command mode (0) or access mode (1))
if command mode => command to execute
  else   access mode  => IP,proto,port
MD5 sum
```

Figura 4.4: Campos do pacote SPA do FWKnop

O número randômico no início é usado para gerar *hashs* únicos e, por consequência, quando cifrado, o pacote também será único. O parâmetro username é enviado, mas nem sempre é usado pelo servidor, dependendo da configuração da diretiva `REQUIRE_USERNAME`.

O *timestamp* contém a hora de preparação do pacote com resolução de um minuto. Por isso, é necessário descartar qualquer outra tentativa de conexão dentro do intervalo de um minuto, através de `KNOCK_INTERVAL`, para impedir um *Replay Attack*.

Em *software version*, é apresentada a versão do software, usada apenas para informação.

Mode permite especificar se será enviado um comando ou um pedido de abertura de porta, especificados logo a seguir.

Por fim, um *hash* MD5 será calculado a partir dos campos anteriores. Após este processo será efetuada a criptografia, seja ela simétrica, com o algoritmo Rijndael, seja ela assimétrica, com o algoritmo do GPG escolhido, sendo o mais robusto ElGamal.

Capítulo 5

FWKnop em uma topologia *Multi-Homed*

Neste capítulo é demonstrada uma configuração do FWKnop para uma topologia *Multi-Homed*, conforme descrição em (O'REILLY, 2000) sobe o título de arquitetura *Dual-Homed*. Um esquema da topologia utilizada é mostrado na figura 5.1.

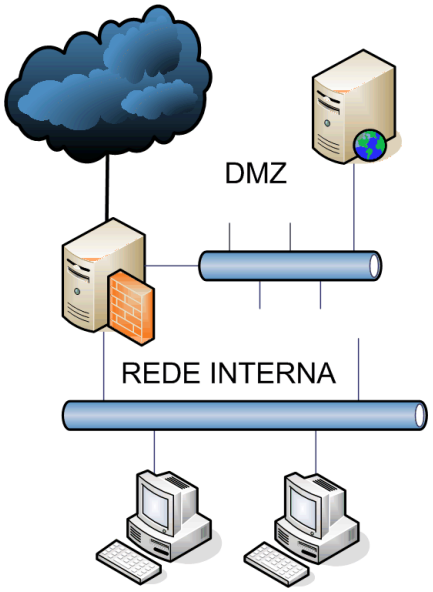

Figura 5.1: Topologia *Multi-homed*

O objetivo do FWKnop é proteger o acesso ao serviço SSH no *firewall* e o acesso ao SSH do servidor Web. Fora o SSH, nenhum acesso será permitido ao *firewall* pela rede externa. Para a rede interna, é permitido o SSH sem uso de FWKnop.

Para efeito de mostrar as diversas formas de configuração possíveis, no firewall, será usada a autenticação com uso de senhas e no capítulo 6 será mostrada a configuração com GPG.

O servidor na DMZ só disponibiliza acesso a HTTP, HTTPS e SSH. Para a rede externa, apenas HTTP e HTTPS estão disponíveis para acesso. O acesso SSH a partir da rede externa é permitido com uso de FWKnop no firewall. Em todos os casos, as regras de firewall usarão NAT para permitir acesso ao servidor Web interno. No caso do SSH neste servidor, a porta usada será 2222 para diferenciar da porta usada no SSH do *firewall*.

5.1 Firewall

A configuração do FWKnop no firewall envolve os serguintes passos:

- Configurar o arquivo `/etc/fwknop/fwknop.conf`;
- Configurar o arquivo `/etc/fwknop/access.conf`;
- Preparar as regras de firewall no arquivo `/etc/rc.firewall`.

No arquivo `fwknop.conf`, são alteradas algumas diretivas, conforme a listagem da figura 5.2.

As demais diretivas deverão permanecer com os valores em que se apresentam no arquivo `/etc/fwknop.conf` que é instalado inicialmente.

O arquivo `/etc/access.conf` deverá ter a configuração demonstrada na figura 5.3.

A partir dessas duas configurações, é possível obter acesso ao SSH tanto no *firewall* quanto no servidor *Web*, cada um por uma das sessões iniciadas por `SOURCE` acima.

O arquivo `/etc/rc.firewall` usado no *firewall* tem o conteúdo descrito na figura 8.1

```
EMAIL_ADDRESSES             root@localhost;
HOSTNAME                    fw;
AUTH_MODE                   PCAP;
PCAP_INTF                   eth0;
PCAP_FILTER                 udp port 62201;
ENABLE_SPA_PACKET_AGING     Y;
MAX_SPA_PACKET_AGE          120;
ENABLE_MD5_PERSISTENCE      Y;
REQUIRE_SOURCE_ADDRESS      N;
FLUSH_IPT_AT_INIT           Y;
ENABLE_TCP_SERVER           N;
```

Figura 5.2: Diretivas do arquivo `fwknop.conf`

```
SOURCE: ANY;
OPEN_PORTS: tcp/22;
KEY: m1nhach4v3;
FW_ACCESS_TIMEOUT: 30;

SOURCE: ANY;
OPEN_PORTS: tcp/2222;
DATA_COLLECT_MODE: PCAP;
DISABLE_FW_ACCES: Y;
ENABLE_CMD_EXEC;
KEY: myencryptkey;
FW_ACCESS_TIMEOUT: 30;
```

Figura 5.3: Diretivas do arquivo `access.conf`

Observa-se na figura 8.1 que não existem regras para a porta 2222 usada pelo SSH do servidor *Web* e, pela política padrão `DROP`, a tentativa de conexão será descartada. Não há regras específicas para descarte de varreduras nem outras técnicas comuns em `iptables`, pois não é o objetivo deste trabalho.

5.2 Acesso SSH ao firewall

Para obter acesso ao SSH, é necessário o envio de um pacote SPA para liberação de acesso. Sem este pacote, não há regras que permitam o uso do SSH (porta 22) no *firewall*. Se for tentado o comando `ssh 192.168.8.129` a partir do cliente, este não surtirá qualquer efeito e terminará por *timeout*, conforme mostra o trecho descrito pela figura 5.4.

```
fmachado@VM1:~$ ssh 192.168.8.129
ssh: connect to host 192.168.8.129 port 22: Connection timed out
```

Figura 5.4: Tentativa de conexão do cliente antes do envio do SPA

O pacote SPA deve ser enviado e, imediatamente, pode ser iniciada a sessão SSH, conforme mostrado pela figura 8.2.

O acesso SSH foi permitido após o envio do pacote SPA com a senha correta. No lado servidor, o que ocorre é mostrado na figura 5.5.

```
Aug 23 00:35:53 localhost fwknopd: received valid Rijndael
  encrypted packet from: 192.168.8.128, remote user: fmachado
Aug 23 00:35:53 localhost fwknopd: adding iptables FWKNOP_INPUT
  ACCEPT rule for 192.168.8.128 -> tcp/22 (30 seconds)
Aug 23 00:36:24 localhost knoptm: removed iptables FWKNOP_INPUT
  ACCEPT rule for 192.168.8.128 -> tcp/22, 30 second timeout
  exceeded
```

Figura 5.5: Registro do envio do SPA, criação e remoção da regra no servidor

A regra de firewall mostrada na figura 5.6 é criada e permanece por 30 segundos.

O comportamento é o esperado e combina com o que é mostrado no servidor. Tentativas de conexão após 30 segundos provocam *timeout* como ocorre antes do envio do SPA.

```
Chain FWKNOP_INPUT (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.8.128       0.0.0.0/0           tcp dpt:22
```

Figura 5.6: Regra de firewall criada no servidor

5.3 Acesso SSH ao servidor Web

Para testar o acesso ao servidor *Web*, é necessária uma outra estratégia. Embora o FWKnop permita trocar as regras de `INPUT` para `FORWARD`, não é possível usar as duas formas ao mesmo tempo. Por isso, deve ser usada a estratégia de envio de comandos explícitos ao FWKnop, o que implica no uso da segunda parte da configuração existente no `access.conf`. Isso é feito no cliente usando a opção `-Server-cmd`, conforme demonstrado na figura 8.3.

No lado servidor, o efeito desta mensagem SPA é o mostrado pela figura 5.7

```
Aug 23 01:21:28 localhost fwknopd: received valid Rijndael
 encrypted packet from: 192.168.8.128, remote user: fmachado
Aug 23 01:21:28 localhost fwknopd: executing command ",;iptables -
  t nat -A PREROUTING -p tcp --dport 2222
 -d 192.168.8.129 --j DNAT --to 192.168.4.146:22;/sbin/iptables -A
    FORWARD -p tcp --dport 22 -d 192.168.4.126 -j ACCEPT for
   192.168.8.128
```

Figura 5.7: Registro do envio do SPA e execução dos comandos

As regras foram criadas conforme o envio, como pode ser visto na figura 5.8.

```
Chain FORWARD (policy DROP)
target     prot opt source          destination
ACCEPT     tcp  --  0.0.0.0/0       192.168.8.129   tcp dpt:22

[root@localhost ~]# iptables -t nat -nL
Chain PREROUTING (policy ACCEPT)
target     prot opt source          destination
DNAT       tcp  --  0.0.0.0/0       192.168.8.129   tcp dpt:2222 to
  :192.168.4.146:22
```

Figura 5.8: Regras de firewall criadas no servidor

Durante o processo de testes, foi identificada uma provável falha no FWKnop, evidenciada no trecho `... command ,;iptables ...` onde aparece um caracter ',' a mais. Esse erro apareceu insistentemente nas tentativas, mesmo com variação de comandos. Por isso, o uso do caracter ';' no início da linha que isola a virgula e, mesmo ainda existindo um erro de execução relacionado a essa vírgula, o resto dos comandos executam corretamente, como mostrado na figura 5.8.

Após a execução do envio do SPA e a criação das regras no *firewall*, o acesso ao servidor web através do redirecionamento é permitido. É possível substituir as duas regras por uma, (com o uso de `REDIRECT` ao invés de `DNAT` mas é parte do objetivo mostrar a execução de mais de um comando enviado pelo cliente.

Uma diferença importante para a criação de regras automáticas, como foi usado para o acesso SSH ao *firewall*, é a regra não ser automaticamente removida, sendo necessário o envio de comando explícito de remoção com uso de `-D` ao invés de `-A`.

Capítulo 6

FWKnop em uma topologia *Screened Subnet*

Neste capítulo é demonstrada uma configuração do FWKnop para uma topologia *Screened Subnet*, conforme descrita (O'REILLY, 2000) sobe o título de mesmo nome. Um esquema da topologia utilizada é mostrado na figura 6.1. Não serão mostradas as configurações do servidor de *firewall* interno por não usarem FWKnop.

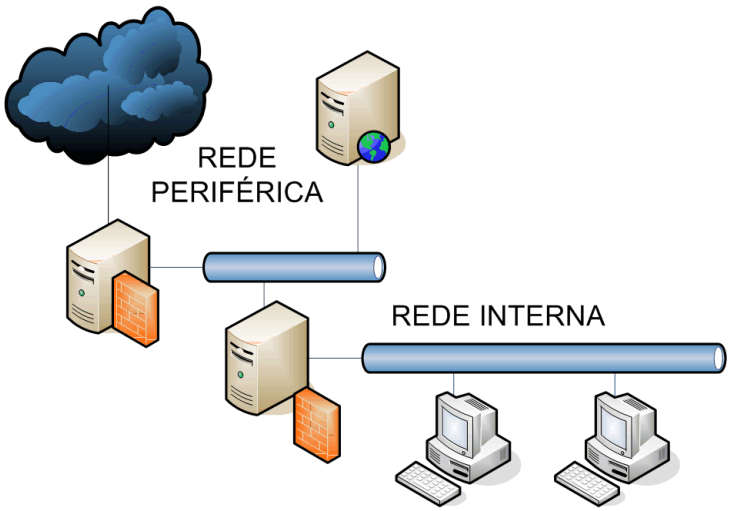

Figura 6.1: Topologia *Screened Subnet*

O objetivo para este cenário é não ter portas abertas diretamente no firewall externo com destino ao próprio firewall, ou seja, não será possível administrá-lo diretamente através da interface externa. Apenas duas regras serão permitidas na interface externa quando for recebido o pacote SPA correto:

- Tráfego UDP para a porta 62202 do Servidor Bastião;
- Tráfego de SSH, na porta 22, com destino ao Servidor Bastião.

6.1 Firewall Externo

Para implementar esta configuração, altera-se o parâmetro no arquivo `fwknop.conf` que regula a criação de regras de *firewall* do `NetFilter`. A figura 6.2 mostra como foi configurada a opção. A diretiva `IPT_AUTO_CHAIN1` pode ser usada para criar regras de INPUT (com destino ao *firewall*), por *default*, mas foi alterada para criar regras FORWARD (que atravessam o *firewall*). É possível utilizar a forma de permissão de envio de comandos descrita no capítulo 5, com envio de comandos completos para o *firewall*, mas pode ser utilizada a forma descrita neste capítulo.

```
IPT_AUTO_CHAIN1    ACCEPT, both, filter, FORWARD, 1, FWKNOP_FWD, 1;
```

Figura 6.2: Configuração do fwknop.conf para criar regras de FORWARD

Para complementar a configuração, será necessário criar as regras de *firewall* que impeçam o tráfego por *default* e as regras que fazem o NAT para o Servidor Bastião, conforme mostrado na figura 9.1. Observe-se que as regras de NAT não permitem o tráfego, apenas alteram o destino do pacote, redirecionando o tráfego para o Servidor Bastião, embora originalmente tenha sido enviado para o *firewall* externo. Ao atingirem a *chain* `FORWARD` não será encontrada uma regra de permissão da passagem do tráfego e, por consequência, o tráfego será descartado.

O resultado é demonstrado na figura 9.2, onde fica claro que não existem regras para aceitar o tráfego das portas udp/62202 e tcp/22 pela interface externa. Somente existe uma regra que permite o tráfego para as conexões estabelecidas e registradas na tabela de estado. As regras de permissão de passagem do tráfego serão inseridas pelo FWKnop após o recebimento do pacote SPA correto. A regra

de acesso ao SSH, porta 22, pela interface interna está presente para permitir que uma vez estabelecido o tráfego SSH para o Servidor Bastião, inicie a partir deste último uma sessão SSH para o *firewall* com objetivo de administrá-lo remotamente por via indireta.

Para aumentar ainda mais o nível de segurança da solução, será utilizada a configuração baseada em chaves assimétricas com uso do GPG. Por não ser necessário o compartilhamento de chaves a solução não sofre do problema de manutenção e distribuição das senhas. Para preparar as chaves e registrá-las no cliente e no servidor, os passos usados foram os descritos na figura 6.3 onde são mostrados os passos de geração e importação de chaves.

A última alteração antes de reiniciar o serviço do FWKnop para entrar em funcionamento é preparar o arquivo `access.conf`, conforme o conteúdo da figura 6.4. Note-se que é necessário fornecer a senha privada no arquivo, o que implica em não utilizar uma chave GPG existente, mas criar uma nova exclusivamente para o FWKnop. De forma similar, a chave do cliente também deve ser exclusiva para acessar o FWKnop.

Ao final dessas configurações, o servidor FWKnop estará pronto para monitorar o tráfego e tratar da abertura das regras quando o SPA criptografado pela chave pública do servidor no cliente for verificado no servidor.

A geração de um pacote SPA pelo cliente se dá como mostra a figura 9.3. Nela mostra-se o uso da opção `-Spoof-src`, que objetiva inserir no pacote o IP de origem como sendo o especificado pelo comando. Seu objetivo é permitir o envio de pacotes SPA mesmo a partir de redes que implementem NAT de origem. As outras soluções possíveis para o problema do NAT são não usar o campo de IP dentro do pacote SPA, alterando a diretiva `REQUIRE_SOURCE_ADDRESS` para N no arquivo `fwknop.conf` ou usar a opção -w na linha de comando do cliente, o que gera uma chamada ao *site* www.whatsmyip.com e o endereço retornado seria utilizado como endereço de origem.

6.2 Firewall Interno

O efeito no *firewall* do envio do SPA descrito na figura 9.3 pode ser verificado na figura 9.4, onde foram inseridas as regras de FORWARD na *chain* `FWKNOP_FWD`, conforme configuração efetuada no arquivo `fwknop.conf`. Algumas linhas foram omitidas e são as mesmas descritas na figura 9.2.

```
root@VM1:~# gpg --import server.asc
gpg: key 2EAF7BFC: public key "Servidor FWKnop (Servidor FWKnop) <
  fwknop@servidor>" imported
root@VM1:~# gpg --edit-key 2EAF7BFC
pub  1024D/2EAF7BFC  created: 2007-08-29  expires: never
  usage: SC
                     trust: unknown       validity: unknown
sub  2048g/2180D480  created: 2007-08-29  expires: never
  usage: E
[ unknown] (1). Servidor FWKnop (Servidor FWKnop) <fwknop@servidor
  >

Command> sign

pub  1024D/2EAF7BFC  created: 2007-08-29  expires: never
  usage: SC
                     trust: unknown       validity: unknown
 Primary key fingerprint: 732F 887D BB58 4F11 7F80  8002 635B EE54
    2EAF 7BFC

     Servidor FWKnop (Servidor FWKnop) <fwknop@servidor>

Are you sure that you want to sign this key with your
key "Cliente (Cliente) <cliente@cliente>" (3C80FD7F)

Really sign? (y/N) y

You need a passphrase to unlock the secret key for
user: "Cliente (Cliente) <cliente@cliente>"
1024-bit DSA key, ID 3C80FD7F, created 2007-08-29

Command> lsign
"Servidor FWKnop (Servidor FWKnop) <fwknop@servidor>" was already
  signed by key 3C80FD7F
Nothing to sign with key 3C80FD7F

[root@localhost ~]# gpg --import cliente.asc
gpg: key 91AF8A14: public key "Cliente FWKnop (Cliente FWKnop) <
  cliente@fwknop>" imported
[root@localhost ~]# gpg --edit-key 91AF8A14
Command> sign
1024-bit DSA key, ID 2EAF7BFC, created 2007-08-29
```

Figura 6.3: Geração de chaves assimétricas e importacao de chaves

```
SOURCE: ANY;
OPEN_PORTS: tcp/22,udp/62202;
FW_ACCESS_TIMEOUT: 60;
GPG_HOME_DIR: /root/.gnupg;
GPG_DECRYPT_ID: 2EAF7BFC;
GPG_DECRYPT_PW: S3rv1doRfwkNOP;
GPG_REMOTE_ID: 91AF8A14;
```

Figura 6.4: Arquivo access.conf

A combinação das regras de NAT existentes com a criação das regras de FORWARD pelo FWKnop permite o acesso do cliente ao Servidor Bastião, primeiro para enviar o SPA para ele na porta 62202 e, após autorizado, o início da sessão SSH na porta 22.

6.3 Servidor Bastião

Em conjunto com as regras preparadas no *firewall*, são necessárias configurações no Servidor Bastião para permitir que o SPA enviado para a porta 62202 do *firewall* externo e redirecionado para o Servidor Bastião seja analisado pelo FWKNop deste servidor. A porta 62202 foi utilizada para não conflitar com a porta 62201 usada pelo FWKNop do *firewall* externo.

No arquivo `fwknop.conf`, são alteradas algumas diretivas, conforme a listagem da figura 6.5.

```
EMAIL_ADDRESSES             root@localhost;
HOSTNAME                    fw;
AUTH_MODE                   PCAP;
PCAP_INTF                   eth0;
PCAP_FILTER                 udp port 62202;
ENABLE_SPA_PACKET_AGING     Y;
MAX_SPA_PACKET_AGE          120;
ENABLE_MD5_PERSISTENCE      Y;
REQUIRE_SOURCE_ADDRESS      N;
FLUSH_IPT_AT_INIT           Y;
ENABLE_TCP_SERVER           N;
```

Figura 6.5: Diretivas do arquivo `fwknop.conf`

O arquivo `access.conf` contém as diretivas descritas pela figura 6.6. Na figura 9.1, está descrito o arquivo `rc.firewall` que contém as regras de *firewall* necessárias para proteger o servidor. Note-se que existe uma regra permitindo o estabelecimento de sessão SSH iniciada no Servidor Bastião com destino ao *firewall* externo.

```
SOURCE: ANY;
OPEN_PORTS: tcp/22;
KEY: CH4v3d0B4stiao;
FW_ACCESS_TIMEOUT: 30;
```

Figura 6.6: Arquivo access.conf do Servidor Bastião

Testes realizados no *firewall* externo mostram a criação das regras FORWARD para o Servidor Bastião. Uma vez ativadas as configurações do Servidor Bastião, é possível estabelecer o contato com o seu SSH, conforme figura 9.5. Note-se o uso da opção `-Server-port` com o parâmetro `62202` para atingir o FWKnop do Servidor Bastião.

Capítulo 7

Conclusão

Esta monografia procurou mostrar os benefícios da utilização do FWKnop para implementar mais uma camada de segurança para proteger os servidores de rede. A flexibilidade de configuração e o nível de segurança atingido demonstram que o FWKnop pode ser mais uma ferramenta a disposição dos administradores de rede na proteção do serviços disponibilizados pelos servidors.

Os objetivos foram todos alcançados, sejam eles:

- O FWKnop foi instalado com sucesso e as várias configurações testadas atingiram os objetivos desejados;
- Os experimentos fora bem sucedidos e atingiram os objetivos desejados nas duas topologias utilizadas, servindo para comprovar a flexibilidade e facilidade de configuração, controle de acesso aos serviços e segurança do tráfego do FWKnop.

Como mencionado no capítulo 2, não existe segurança total ou perfeita sem impedir completamente a usabilidade. O que deve ser buscado é o correto balanço entre segurança e acesso. O que a ferramenta FWKnop permite é aumentar o nível de segurança até de serviços com pouca ou nenhuma segurança própria, possibilitando a disponibilização de mais acesso de maneira muito controlada.

Como toda ferramenta, existem limitações no FWKnop que afetam em parte sua aplicabilidade, como não ser possível estabelecer regras do tipo FORWARD com endereços enviados no pacote. O FWKnop apenas consegue criar regras permitindo todo o tráfego FORWARD com restrição da origem mas não do destino.

Há ainda alguns erros de implementação mas que não são impeditivos para seu uso e o FWKnop está em constante evolução.

Em segurança não existem soluções que atendam a todos os casos e o FWKnop não é exceção à regra. Seu uso não é adequado a situações onde o tráfego seja efêmero, devido à necessidade de envio de pacote para cada abertura de sessão, nem para serviços que tenham que ser acessados por muitos usuários, devido à limitação de uma chave simétrica ou assimétrica associada a cada serviço, pois a distribuição de uma mesma senha ou chave para muitos usuários fragiliza a segurança. Também não é adequado o uso de chaves simétricas para acessos muito frequentes, pois permite o acúmulo de captura de tráfego por um agressor para efeito de tentar ataques de força bruta na decifração do tráfego para obtenção da chave simétrica, comprometendo a segurança.

A limitação apresentada não é séria o suficiente para diminuir o potencial do FWKnop. Sua implementação é simples, mas efetiva; flexível, mas controlada; leve, mas poderosa.

A motivação desta monografia era avaliar o uso da ferramenta FWKnop e sua efetividade em aumentar a proteção existente e pode-se concluir que ela atende plenamente a estes objetivos. Como qualquer ferramente, pode ser aperfeiçoada e é isso que propõe a seção Trabalhos Futuros.

7.1 Contribuições

O autor desta monografia foi convidado a escrever um capítulo sobre FWKnop para o livro em desenvolvimento pelo orientador, professor Sandro Melo. Este trabalho se dará após a conclusão desta monografia.

Após os experimentos realizados, o autor desta monografia entrou em contato com o autor do FWKnop e informou a ocorrência do erro de acréscimo do caracter ',' mencionado no capítulo 5. Após análise das configurações anexadas e da captura da saída dos comandos, foi identificado um erro de análise de comandos enviados pelo cliente que contivessem endereços IP. O erro foi corrigido na versão 1.8.2, disponibilizada após a conclusão dos experimentos descritos nesta monografia.

Por esta contribuição, o autor desta monografia passou a figuras no arquivo `CREDITS` da versão 1.8.2, conforme anexo.

7.2 Trabalhos Futuros

Esta monografia se dedicou a avaliar o estado atual do FWKnop, mas existem alguns pontos onde é possível melhorá-la. Futuros trabalhos sobre o tema podem se dedicar a avaliar soluções para as limitações do FWKnop, em especial ao uso de FORWARD com o endereço do destino incorporado ao conjunto de dados enviados no pacote SPA. Uma opção é extender o número de campos do pacote SPA e incluindo um campo como `DESTINATION_IP_FWD` para os casos de uso do FWKnop para permitir acesso através do *firewall* e não com destino a ele. Em contato com o autor do FWKnop, ele menciona que está estudando uma solução para este problema.

Uma monografia de uso prático e que pode solucionar muitos problemas é usar o FWKnop para proteger acessos SNMP a servidores. Como SNMP, nas versões anteriores à v3, é bastante inseguro, o FWKnop pode prover uma camada de isolamento e permitir acesso somente vindo do servidor de gerência com abertura seletiva via pacote SPA.

Capítulo 8

Apêndice - Topologia *Multi-Homed*

```
#!/bin/sh
IPT=/sbin/iptables
IP_FW=192.168.8.129
IP_WEB=192.198.4.126

$IPT -F
$IPT -X
$IPT -P INPUT DROP
$IPT -P FORWARD DROP
$IPT -P OUTPUT DROP

$IPT -A INPUT -i lo -j ACCEPT
$IPT -A OUTPUT -o lo -j ACCEPT
$IPT -A OUTPUT -p icmp -m state --state NEW,ESTABLISHED,RELATED -j
   ACCEPT
$IPT -A INPUT -p icmp -m state --state ESTABLISHED,RELATED -j
  ACCEPT

$IPT -A INPUT -m state --state ESTABLISHED -j ACCEPT
$IPT -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
$IPT -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

$IPT -t nat -A PREROUTING -d $IP_FW --dport 80 -j DNAT --to
  $IP_WEB
$IPT -t nat -A PREROUTING -d $IP_FW --dport 443 -j DNAT --to
  $IP_WEB
```

Figura 8.1: Arquivo de configuração das regras de firewall

```
fmachado@VM1:~$ fwknop -k 192.168.8.129 -A "tcp/22" -s
[+] Starting fwknop client.
[+] Enter an encryption key. This key must match a key in the file
    /etc/fwknop/access.conf on the remote system.

Encryption Key:

[+] Building encrypted single-packet authorization (SPA) message
  ...
[+] Packet fields:

        Random data: 5059916184554811
        Username:    fmachado
        Timestamp:   1187840153
        Version:     1.8.1
        Action:      1 (access mode)
        Access:      0.0.0.0,tcp/22
        MD5 sum:     StLGrjnPoo+2eMTu+BL8uQ

[+] Sending 150 byte message to 192.168.8.129 over udp/62201...
fmachado@VM1:~$ ssh 192.168.8.129
The authenticity of host '192.168.8.129 (192.168.8.129)' can't be
  established.
RSA key fingerprint is 58:81:cd:72:40:57:3c:93:37:6e:7e:34:bd:a8:
  be:86.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.8.129' (RSA) to the list of
  known hosts.
fmachado@192.168.8.129's password:
[fmachado@localhost ~]$ exit
Connection to 192.168.8.129 closed.
fmachado@VM1:~$
```

Figura 8.2: Acesso permitido ao cliente após o SPA

```
root@VM1:/usr/local/src/fwknop-1.8.1# fwknop --Server-cmd ';
  iptables -t nat -A PREROUTING -p tcp --dport 2222 -d
  192.168.8.129 --j DNAT --to 192.168.4.146:22;/sbin/iptables -A
  FORWARD -p tcp --dport 22 -d 192.168.4.126 -j ACCEPT' -k
  192.168.8.129
[+] Starting fwknop client.
[+] Enter an encryption key. This key must match a key in the file
    /etc/fwknop/access.conf on the remote system.

Encryption Key:

[+] Building encrypted single-packet authorization (SPA) message
  ...
[+] Packet fields:

        Random data: 6943434705763198
        Username:    fmachado
        Timestamp:   1187842403
        Version:     1.8.1
        Action:      0 (command mode)
        Cmd:         ;;iptables -t nat -A PREROUTING -p tcp --
          dport 2222 -d 192.168.8.129 --j DNAT --to
          192.168.4.146:22;/sbin/iptables -A FORWARD -p tcp --
          dport 22 -d 192.168.4.126 -j ACCEPT
        MD5 sum:     lVDkbmysJAhslgyrOLMbeA

[+] Sending 256 byte message to 192.168.8.129 over udp/62201...
```

Figura 8.3: Envio do SPA para o servidor com regras para acesso ao servidor Web

Capítulo 9

Apêndice - Topologia *Screened Subnet*

```
#!/bin/sh
# Arquivo /etc/rc.firewall, executado a cada boot

IPT=/sbin/iptables
IP_FW=192.168.8.129
IP_BASTIAO=192.168.4.127
IF_EXTERNA=eth0
IF_INTERNA=eth1

$IPT -F
$IPT -X
$IPT -P INPUT DROP
$IPT -P FORWARD DROP
$IPT -P OUTPUT DROP

$IPT -A INPUT -i lo -j ACCEPT
$IPT -A OUTPUT -o lo -j ACCEPT
$IPT -A OUTPUT -p icmp -m state --state NEW,ESTABLISHED,RELATED -j
   ACCEPT
$IPT -A INPUT -p icmp -m state --state ESTABLISHED,RELATED -j
  ACCEPT

$IPT -A INPUT -m state --state ESTABLISHED -j ACCEPT
$IPT -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
$IPT -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

$IPT -t nat -A PREROUTING -p udp --dport 62202 -d $IP_FW -j DNAT
  --to $IP_BASTIAO
$IPT -t nat -A PREROUTING -p tcp --dport 22 -d $IP_FW -j DNAT --to
   $IP_BASTIAO
$IPT -A INPUT -p tcp --dport 22 -i $IF_INTERNA -j ACCEPT
```

Figura 9.1: Arquivo rc.firewall contendo as regras iniciais do firewall

```
[root@localhost etc]# iptables -nL
Chain INPUT (policy DROP)
target   prot opt source          destination
ACCEPT   0    --  0.0.0.0/0       0.0.0.0/0
ACCEPT   icmp --  0.0.0.0/0       0.0.0.0/0       state RELATED,
  ESTABLISHED
ACCEPT   0    --  0.0.0.0/0       0.0.0.0/0       state
  ESTABLISHED
ACCEPT   0    --  0.0.0.0/0       0.0.0.0/0       state RELATED,
  ESTABLISHED
ACCEPT   tcp  --  0.0.0.0/0       192.168.4.129   tcp dpt:22

Chain FORWARD (policy DROP)
target   prot opt source          destination
ACCEPT   0    --  0.0.0.0/0       0.0.0.0/0       state RELATED,
  ESTABLISHED

Chain OUTPUT (policy DROP)
target   prot opt source          destination
ACCEPT   0    --  0.0.0.0/0       0.0.0.0/0
ACCEPT   icmp --  0.0.0.0/0       0.0.0.0/0       state NEW,
  RELATED,ESTABLISHED
ACCEPT   0    --  0.0.0.0/0       0.0.0.0/0       state NEW,
  RELATED,ESTABLISHED

[root@localhost etc]# iptables -t nat -nL
Chain PREROUTING (policy ACCEPT)
target   prot opt source          destination
DNAT     udp  --  0.0.0.0/0       192.168.8.129   udp dpt:62202 to
  :192.168.4.127
DNAT     tcp  --  0.0.0.0/0       192.168.8.129   tcp dpt:22 to
  :192.168.4.127

Chain POSTROUTING (policy ACCEPT)
target   prot opt source          destination

Chain OUTPUT (policy ACCEPT)
target   prot opt source          destination
```

Figura 9.2: Listagem das regras aplicadas

```
root@VM1:~# fwknop -A 'tcp/22,udp/62202' --gpg-recip 2EAF7BFC --
  gpg-sign 91AF8A14 --Spoof-src 192.168.8.128 -k 192.168.8.129 -s
[+] Starting fwknop client.
[+] Enter the GnuPG password for signing key: 91AF8A14

GnuPG signing password:

[+] Building encrypted single-packet authorization (SPA) message
  ...
[+] Packet fields:

        Random data: 6153318370497824
        Username:    root
        Timestamp:   1188355691
        Version:     1.8.1
        Action:      1 (access mode)
        Access:      0.0.0.0,tcp/22,udp/62202
        MD5 sum:     MAMs8270VxlxndmlwiVXGw

[+] Sending 1019 byte message to 192.168.8.129 over udp/62201
    (spoofed src ip: 192.168.8.128).
```

Figura 9.3: Envio do SPA pelo cliente para o Firewall

```
[root@localhost ~]# tail -n 3 /var/log/messages
Aug 28 23:48:39 localhost fwknopd: received valid GnuPG encrypted
  packet (signed with required key ID: "91AF8A14") from:
  192.168.8.128, remote user: root
Aug 28 23:48:39 localhost fwknopd: adding iptables FWKNOP_FWD
  ACCEPT rule for 192.168.8.128 -> udp/62202 (60 seconds)
Aug 28 23:48:40 localhost fwknopd: adding iptables FWKNOP_FWD
  ACCEPT rule for 192.168.8.128 -> tcp/22 (60 seconds)
[root@localhost ~]# iptables -nL
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
FWKNOP_FWD  0    --  0.0.0.0/0            0.0.0.0/0

Chain FWKNOP_FWD (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.8.128        0.0.0.0/0           tcp
  dpt:22
ACCEPT     udp  --  192.168.8.128        0.0.0.0/0           udp
  dpt:62202
```

Figura 9.4: Resultado do SPA no Firewall

```
root@VM2:~# fwknop --Server-port 62202 -A "tcp/22" -k
  192.168.8.129 -s
[+] Starting fwknop client.
[+] Enter an encryption key. This key must match a key in the file
    /etc/fwknop/access.conf on the remote system.

Encryption Key:

[+] Building encrypted single-packet authorization (SPA) message
  ...
[+] Packet fields:

        Random data: 8803647494214658
        Username:    fmachado
        Timestamp:   1188795820
        Version:     1.8.1
        Action:      1 (access mode)
        Access:      0.0.0.0,tcp/22
        MD5 sum:     agJA2whj1+zRAXaCXFVi7Q

[+] Sending 150 byte message to 192.168.8.129 over udp/62202...

root@VM2:~# ssh fmachado@192.168.8.129
password:
```

Figura 9.5: Arquivo access.conf do Servidor Bastião

Capítulo 10

Anexo - Arquivo CREDITS

```
David Jacobs
    - Suggested IP/network lists in SOURCE definitions
    - Wording fixes in fwknop(8) manpage.

Brian Snipes
    - Wrote a graphical front-end for fwknop called "fwknopFE":
      http://www.snipes.org/index.php?page=fwknopFE
    - Found bug with legacy fingerprinting file "posf".

Joel Loudermilk
    - Submitted patch to optionally disable email alerting.  The
      end result
      was the addition of the REPORT_METHOD keyword in fwknop.conf
        .

Blair Zajac
    - Submitted patch to not install perl modules in /usr/lib/
      fwknop/ that
      are already installed in the system perl lib tree.
    - Submitted patch to use getpwuid() instead of just getlogin()
      .
    - Submitted patch to fix bug in install.pl and how the ~/lib
      directory
      is created in client install mode.
    - Found bug with perl module file paths and naming convention
      (this bug
      resulted in some modules being needlessly installed).
    - Suggested that fwknop handle rotated log files (even pcap
      logs get
      rotated on some systems).
```

```
    - Suggested that modules only required in server mode are not
      use at
      runtime when running fwknop in client mode.
    - Suggested -O optimization in Makefile.
    - Found bug where log rotation detection would break under the
       size
      change detection method.  The result was the inode check in
        0.9.6.
    - Found bug where some Linux distributions have /var/run as
      type tmpfs,
      and this caused fwknopd to die because it couldn't write to
        its PID
      file.
    - Suggested command path update code in install.pl so that the
       user does
      not always have to edit the fwknop.conf and knopwatchd.conf
        files if
      the system does not have commands in the default locations.

Will McCracken
    - Reported bug on OS X where getlogin() does not return the
      correct
      data.  This permitted fwknop to be updated to fall back to
        ENV{'USER'}
      var.

Omar A. Herrera
    - Submitted a patch to fix a timeout bug in knoptm that caused
       newly
      created rules to be deleted too quickly.

Werner Wiethege
    - Submitted a patch to fix a bug in knoptm where inappropriate
       hash
      keys were being deleted and so previous timeouts would apply
         to the
      current interval.

Ronald Bister
    - Submitted a fix for not being able to parse ifconfig output
      correctly
      when languages besides English are used.

Hank Leininger
    - Suggested privilege separation to minimize code that
      executes as root.
    - Suggested NULL password GPG keys.
    - Suggested integration with ssh-agent and gpg-agent.
```

Dwayne Rightler
    - Submitted patch to fix bug where whatismyip.com altered
      their return
      data format and this broke the -w command line switch.

Sebastien Jeanquier
    - Contributed more rigorous regular expression for matching an
       IP address.
    - Suggested allowing symmetric keys to exceed 256 bits.
    - Suggested using Crypt::Random for random number generation.
    - Suggested the integration of time synchronization as an
      additional
      measure for the fwknopd daemon to validate incoming SPA
        packets (this
      will probably be enabled by default).
    - Suggested a new method of interacting with iptables to
      redirect
      connections to one port to another port on the same system.
    - Suggested making the --Spoof-user argument useable by non-
      root users.
    - Suggested the ability to randomize a spoofed IP address.
    - Suggested building in compatibility with external IP
      resolution sites
      such as http://www.whatismyip.com/
    - Provided a Mac OS X system to develop fwknop support for OS
      X. Many
      thanks!
    - Helped with the testing process for fwknop-1.8.2 and OS X
      support.

Mate Wierdl
    - Contributed patch (originally for the psad project) for
      building the
      RPM on x86_64 platforms.

Raul Siles
    - Bug report to allow OPEN_PORTS to be omitted in access.conf
      in favor of
      having only PERMIT_CLIENT_PORTS enabled.

Leland Weathers
    - Submitted patch to allow the GPG_REMOTE_ID variable to have
      the value
      "ANY" to allow arbitrary gpg signing keys to match the
        SOURCE block.

Juuso Alasuutar

- Suggested that the install.pl script offer a mode where the user is not
  prompted at all in order to make it easier to integrate fwknop with
  the Source Mage Linux distribution. The result is the --Defaults option
  to the install.pl script.
- Suggested the ability to use gpg keys without passwords.

Neal Baer
- Tested the fwknop-1.8 release for Windows systems (running Cygwin).
- Tested the cd_rpmbuilder script on SuSE systems.

Graham Clark
- Suggested man page documentation bug fixes.

Roy Segovia
- Submitted patch to fix print statement bug in command mode where the
  command was inappropriately prepended with the source IP address.
- Reported bug with running fwknop under Cygwin on Windows 2003 Server,
  which reports 'Gygwin' under the 'uname -o' output.

Mark Van De Vyver
- Reported a bug where the iptables command path was not being properly
  discovered if it did not reside at the default location specified in the
  fwknop.conf file.
- Submitted various documentation issues with the fwknop man pages. The -D
  option in fwknop-1.8.2 resulted from this feedback.
- Reported a bug where the .xsession.errors file would fill with output
  logged by fwknop when null passwords were read from stdin.

Flavio Machado
- Reported command mode bug where the source IP address is not properly
  communicated to the SPA server.

Referências Bibliográficas

BARHAM, P. e. a. Techniques for lightweight concealment and authentication in ip networks. *Intel Research Berkeley*, july 2002. Disponível em: <http://www-.intel-research.net/Publications/Berkeley/012720031106_111.pdf>.

BLACK HAT. *SPA: Single Packet Authorization - MadHat 2005 - Presentations*. Black Hat, 2005. Disponível em: <http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-madhat.pdf>.

CRYPTO. *Handbook of Applied Cryptography*. 5. ed. CRC Press, 2001. Disponível em: <http://www.cacr.math.uwaterloo.ca/hac/about/chap8.pdf>.

DCE. *Fundamentos de Hashing*. Instituto de Ciências Matemáticas de São Carlos, 2007. Disponível em: <http://www.icmc.usp.br/~sce183/hash.html>.

DEFCON. *Service Cloaking and Anonymous Access; Combining Tor with Single Packet Authorization (SPA)*. DEFCON, 2006. Disponível em: <http://www.cipherdyne.org/fwknop/docs/talks/dc14_fwknop_slides.pdf>.

GHEORGHE. *Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT, and L7-filter*. 5. ed. Packt Publishing, 2006. Disponível em: <http://www.packtpub.com/linux-firewalls/book>.

GRAHAM-CUMMING, J. Practical secure port knocking. *Dr. Dobb's Journal*, november 2004. Disponível em: <http://www.ddj.com/184405890>.

KRZYWINSKI, M. Port knocking - a stealthy system for network authentication across closed ports. *Port Knocking*, 2003. Disponível em: <http://www-.portknocking.org/>.

MEIJER. *Why spoofing is the number one security problem on the Internet, and how we should fight it*. Rob J. Meijer, 2001. Disponível em: <http://www.xs4all-.nl/~rmeijer/spoofing% -.html>.

O'REILLY. *Building Internet Firewalls*. 2. ed. O'Reilly, 2000. Disponível em: <http://www.oreilly.com/catalog/fire2/>.

PIVA. *Uma técnica informal de prevenção de ataques baseada em estruturas de mensagens*. Unicamp, 2006. Disponível em: <https://www.lca.ic.unicamp.br-/modules/wfdownloads/visit.php?cid=3&lid=4>.

RASH, M. Combining port knocking with os fingerprinting. *The USENIX Magazine*, v. 29, n. 6, p. 19–25, december 2004. Disponível em: <http://www-.usenix.org/publications/login/2004-12/pdfs/fwknop.pdf>.

RASH, M. Combining port knocking with os fingerprinting. *The USENIX Magazine*, v. 31, n. 1, p. 63–69, february 2006. Disponível em: <http://www-.usenix.org/publications/login/2006-02/pdfs/rash.pdf>.

RASH, M. Protecting ssh servers with single packet authorization. *Linux Journal*, n. 157, may 2007. Disponível em: <http://www.linuxjournal.com/article/9621>.

RASH, M. Single packet authorization. *Linux Journal*, n. 156, april 2007. Disponível em: <http://www.linuxjournal.com/article/9565>.

RFC2240. *RFC2240 - OpenPGP Message Format*. IETF - Network Working Group, 1998. Disponível em: <http://www.ietf.org/rfc/rfc2440.txt>.

RFC2616. *RFC2616bis - Protocolo HTTP, com revisões*. W3C - World Wide Web Consortium, 2007. Disponível em: <http://www.w3.org/Protocols/HTTP/1.1-/rfc2616bis/draft-lafon-rfc2616bis-03.txt>.

RFC2818. *RFC2818 - HTTP Over TLS*. IETF, 2000. Disponível em: <http://www.ietf.org/rfc/rfc2818.txt>.

SCHNEIER, B. Secrecy, security, and obscurity. *Crypto-Gram Newsletter*, may 2002. Disponível em: <http://www.schneier.com/crypto-gram-0205.html>.

SEBASTIEN. *An Analysis of Port Knocking and Single Packet Authorization - MSc Thesis*. Royal Holloway - Univeristy of London, 2006. Disponível em: <http://web.mac.com/s.j/>.

SPOOFING. *IP Spoofing: An Introduction*. SecurityFocus, 2003. Disponível em: <http://www.securityfocus.com/infocus/1674>.

VAUDENAY. *A CLASSICAL INTRODUCTION TO CRYPTOGRAPHY - Applications for Communications Security*. 5. ed. Springer Science+Business, 2006. Disponível em: <http://www.cacr.math.uwaterloo.ca/hac/about/chap8-.pdf>.

WELTE, H. Netfilter - firewalling, nat and packet mangling for linux. *NetFilter/iptables*, 2000. Disponível em: <http://www.netfilter.org/>.