Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

ALAN CRISTIAN CARDOSO

SEGURANÇA COMPUTACIONAL UTILIZANDO IPTABLES NO SISTEMA OPERACIONAL LINUX

Assis/SP

2016

ALAN CRISTIAN CARDOSO

SEGURANÇA COMPUTACIONAL UTILIZANDO IPTABLES NO SISTEMA OPERACIONAL LINUX

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação.

Orientando: Alan Cristian Cardoso

Orientador: Prof. Douglas Sanches Cunha

Assis

2016

FICHA CATALOGRÁFICA

CARDOSO, Alan Cristian

Segurança em Servidores Linux Utilizando Iptables / Alan Cristian Cardoso. Fundação Educacional do Município de Assis – FEMA – Assis, 2016.

62p.

Orientador: Prof. Douglas Sanches Cunha

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA

1.Iptables. **2**.Redes de Computadores **3.**Segurança de Redes

CDD: 001.6

SEGURANÇA COMPUTACIONAL UTILIZANDO IPTABLES NO SISTEMA OPERACIONAL LINUX

ALAN CRISTIAN CARDOSO

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientador: Prof. Douglas Sanches Cunha

Analisador: Prof. Diomara Martins Reigato Barros

Assis

2016

DEDICATÓRIA

Este trabalho é dedicado primeiramente a minha família que sempre acreditaram na minha capacidade, aos meus professores que sempre mostraram máxima seriedade, dedicação e empenho quanto ao ensino aos seus alunos, e aos meus amigos, frisando especialmente os que estiveram comigo neste curso.

AGRADECIMENTOS

Agradeço aos meus pais, **Cristiane Cardoso** e **Marcos Cardoso** por me apoiarem e me incentivarem em meus estudos e planos.

A minha namorada, **Josiane Viana**, estando sempre do meu lado, entendendo meu tempo escasso por conta dos afazeres do curso.

Aos meus **amigos e familiares**, em especial os meus colegas de classe em sua grande maioria sempre dispostos a ajudar e dividir seus conhecimentos e descobertas.

Ao meu professor **Douglas**, pela sua seriedade em seu trabalho e sua dedicação em ensinar e motivar seus alunos e por me orientar em meu trabalho me ajudando assim a alcançar todos os objetivos do mesmo.

E por fim, a todos os **professores** do curso, pessoas em quem me inspiro, e que possuem notável amor pelo trabalho que exercem, sempre ensinando com seriedade, dedicação, e muita preocupação com o aprendizado de todos seus alunos.

“Uma longa caminhada começa com o primeiro passo. ”

Lao-Tsé

RESUMO

A consistência do sistema operacional *Linux* acoplada a uma ferramenta de segurança e controle de tráfego de pacotes *Iptables*, faz dele um sistema estrategicamente seguro e confiável, evitando ataques e invasões com o intuito de acessar informações sigilosas. Neste contexto, e com intuito de proteger os dados contra ações nocivas, como furtos ou danos, faz-se necessário o uso deste recurso para manter seguro um dos maiores bens das organizações na atualidade, a informação.

A proposta deste trabalho é usar a ferramenta *Iptables* do sistema operacional *Linux* afim de inviabilizar ataques à um servidor contido neste sistema e proteger as informações guardadas nele, a pesquisa também propõe o desenvolvimento de um aplicativo, com a intenção de facilitar o gerenciamento, manutenção e manipulação das regras do *Iptables* para usuários da área de segurança e ou conhecimento na área de informática.

Palavras-chave: Segurança, Iptables, Firewall, Linux.

ABSTRACT

The consistency of the Linux operating system with your security tool and packets traffic control, it makes it a strategically secure and reliable system, avoiding attacks and invasions in ordem to access sensitive information. In this context, and with a view to protecting data against harmful actions, like theft or damage, it becomes necessary to use this feature to maintain secure one of the greatest assets of organizations today, the information.

The purpose of this research is to use the Iptables tool of the Linux operating system, aiming to derail attacks on a server contained in this system and protect the information stored on it, the research also proposes the development of an application, with the intention to facilitate the management and handling of Iptables rules for users of the safety área and or knowledge in computer science.

Keywords: Security in the Linux, Iptables, Firewall Linux.

Lista de Siglas e Abreviaturas

LAN: Local Area Network

MAN: Metropolitan Area Network

WAN: Wide Area Network

TCP / IP: Transition Control Protocol / Internet Protocol

OSI: Open Systems Interconnect

MAC: Media Access Control

NAT: Network Address Translation

SNAT: Source Network Address Translation

DNAT: Destination Network Address Translation

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 – INTRODUÇÃO

Alguns concordam que no mundo atual a tecnologia faz parte do cotidiano e mais do que facilitar nossa vida, ela tornou-se necessária para nossas tarefas, sejam no trabalho ou lazer. O mundo ao nosso redor, a cada dia mais passa a ser digital, fazer compras pela internet, efetuar pagamentos pela internet *banking*, requisitar e consultar documentos pelos sites governamentais, atividades que a pouco tempo atrás eram realizadas apenas presencialmente. Porém, deve-se atentar ao fato, de que apesar das grandes facilidades a tecnologia traz consigo, um problema pouco discutido entre a grande massa usuária destes serviços. A vulnerabilidade destes serviços e dos aparelhos usados para acessá-los, ocasionando crimes envolvendo *hackers* que invadem estes sistemas computacionais.

Dispositivos com acesso à internet, são comuns hoje em dia, e segundo Kurtz (2014), o Brasil é o país com maior número de ataques hacker a sites do governo de acordo com uma pesquisa divulgada pela empresa de segurança da informação Cyveillance.

De acordo com Bernardes (1999), a situação é mais complicada quando trata de grandes empresas com servidores que contém informações, que oferecem acessos a uma grande quantia de valores em patrimônios e dinheiro aplicado, assim como, órgãos e autarquias do governo que contém informações sigilosas, segredos de estado, dados processuais anônimos entre outros.

Essas organizações necessitam de uma proteção avançada, pois os criminosos interessados em praticar invasões e ataques aos seus sistemas, na grande maioria das situações, possuem alta capacitação nesse ramo da informática, conhecem profundamente como são feitas as blindagens a essas informações e, por consequência, sabem como invadir, quebrar essas proteções.

A informática evolui exponencialmente, isso faz com que a cada novo estágio tecnológico, ela precise de menos tempo para propor soluções. Entretanto, deve-se levar em conta, que os meios de defesa dessa tecnologia, não acompanham essa velocidade de desenvolvimento, tais como: aplicações móveis, novos dispositivos, novas plataformas, sistemas web, etc. Info (2010) apresentou um estudo, onde revela que os desenvolvedores, em muitos casos, acabam por relevar a segurança de sistemas. Mostra os problemas críticos em aparelhos com a tecnologia *bluetooth,*

falhas segurança, que permite aos criminosos obterem informações pessoais do usuário, redes *wireless*, onde pesquisadores das universidades de South Carolina e Rutgers University invadiram as redes sem fio de aparelhos que custam mais de 1.500 dólares usados em automóveis, mostrando assim a fragilidade desse sistema.
De acordo com Motta (2006) os atuais sistemas de defesa, utilizam políticas e mecanismos de segurança para evitar ataques cibernéticos. Uma política de segurança é uma diretriz de alto nível, onde uma empresa segue protocolos, regras para uso dos servidores, como por exemplo, para todo usuário do sistema, deverá usar uma senha de nível alto, não usar rede sem fio em dispositivos que tenham acesso ao servidor. São regras simples, porém deve-se atentar que de nada adianta blindar um servidor, caso os usuários não policiarem quanto aos cuidados que devem ter durante o acesso.
Melo e Geus (2010) apresentaram um modelo de defesa mais eficiente, onde o administrador não precisa acessar o servidor fisicamente, mas sim remotamente, esse modelo é importante se levarmos em consideração que se o servidor não precise de contato físico, a dificuldade quanto a invasões e descobertas de vulnerabilidades do sistema aumenta consideravelmente.
Outro modo de proteção são os Mecanismos de Segurança, que são ferramentas usadas para proteger o sistema, como *firewalls,* configuração das portas de entrada de rede e internet das máquinas, e o uso de um sistema operacional confiável nos servidores. Diante de tal cenário, identifica-se a necessidade de produzir ferramentas que auxiliem o uso das tecnologias que aumentem a defesa de sistemas.

1.1 OBJETIVOS

Este estudo tem como intuito fazer o uso da internet, em seus variados fins, segura para usuários comuns da internet e também para grandes servidores que contenham informações sigilosas, evitando invasões, ataques e crimes cometidos por *hackers*. Para este fim, será usado o *firewall* do sistema operacional Linux, o *Netfilter*, utilizando a ferramenta *Iptables* para sua manipulação, e também tem como objetivo a implementação de um *software* que auxilia a criação dos *scripts,* como regras dessa ferramenta de um modo mais fácil, porém mantendo a eficiência do modo

tradicional.

Ademais a pesquisa tem como finalidade destacar que o desenvolvimento das proteções na informática é tão importante quanto o desenvolvimento tecnológico e que ambos necessitam evoluir em igual escala.

1.2 JUSTIFICATIVAS

Pode-se afirmar que, quanto mais informações no servidor e mais sigilosas elas são, torna-se mais necessário a proteção contra as vulnerabilidades. Uma pessoa que acha vulnerabilidades no sistema de uma empresa, por exemplo, consegue assim invadi-lo e causar prejuízos financeiros a esta empresa, tanto diretamente transferindo valores de contas bancárias, quanto indiretamente, roubando projetos e cadastros do bando de dados da mesma, uma organização sem seu banco de dados torna-se praticamente incapaz de prestar seus serviços.

O Globo (2014), mostra um estudo do CEIE, Centro de Estudos Internacionais e Estratégicos nos EUA, onde é apontado que o prejuízo causado por operações de *hackers* no mundo todo é de pelo menos 300 bilhões de dólares, podendo chegar a 1 trilhão. Segundo essa mesma pesquisa, é possível encontrar na internet, em camadas escondidas para usuários comuns, kits prontos sendo vendidos, com pacotes de vírus e softwares de invasão. O site também informa de outra pesquisa da RAND Corporation, onde é observado que 80% desses *hackers* trabalham em grupo, formando organizações para cometerem este tipo de crime, somando em torno de 80 mil integrantes.

1.3 MOTIVAÇÃO

A ideia para este trabalho surgiu pelo interesse de trabalhar como perito na área de segurança computacional na polícia federal. Com alto nível de concorrência para os cargos, é preciso aprofundar os conhecimentos relacionados a redes de computadores, servidores, vulnerabilidades, ataques a sistemas, blindagem de

sistemas operacionais, proteção servidores entre outros temas, unindo assim a necessidade de se fazer um trabalho de conclusão de curso e o desejo de ingressar neste cargo.

1.4 PÚBLICO ALVO

Usuários da internet que executam tarefas que necessitam de proteção como transferências bancárias, operações com documentos pessoais, mesmo com pouco conhecimento em informática, poderão tirar proveito do sistema de proteção, assim como empresas que utilizem servidores com informações que precisem ser protegidas.

1.5 PERSPECTIVA DE CONTRIBUIÇÃO

Este trabalho tem a possibilidade de contribuir na área da informática e tecnologia dando segurança aos sistemas operacionais e consequentemente aos usuários dos mesmos. Há ainda o intuito de disponibilizar a aplicação que será desenvolvida para a comunidade de *software* livre, através de um servidor para *softwares* chamado *Git – Hub*, podendo assim ser otimizado e melhorado por programadores participantes da comunidade.

1.6 ESTRUTURA DO TRABALHO

No primeiro capítulo, é apresentado uma explicação abrangente sobre o tema do trabalho, contendo introdução ao assunto, o estado da arte do tema, mostrando também importância dessa pesquisa. No capítulo seguinte, é apresentado alguns conceitos sobre redes de computadores, será mostrado a base do trabalho, apresentando o que especificamente são redes computacionais, e por que precisam ser protegidas. No terceiro capítulo comenta-se sobre a segurança nas redes, uma explicação sobre segurança de redes em geral, como ela é desempenhada em variados sistemas operacionais, explicando termos e dissertando sobre noções

importantes sobre o assunto. No quarto capítulo sobre, *Iptables,* é apresentado o sistema de segurança do sistema operacional Linux, que é o assunto principal do trabalho, mostrando suas funções, características, vantagens em relação a outros sistemas operacionais, e também mostrando técnicas para configurá-lo. No penúltimo capítulo é mostrado a estrutura e as características da aplicação que será desenvolvida para inserir regras na ferramenta *Iptables*, visando facilitar as configurações de segurança do sistema operacional Linux. No último capítulo são apresentados os resultados e a eficiência tanto dos *scripts* criados pelo modo tradicional, linha de comando, quanto dos *scripts* originados da aplicação, além da comparação entre ambos.

2 – REDES DE COMPUTADORES

Neste capítulo é apresentado o conceito sobre redes de computadores, suas características, para se entender o que será protegido utilizando os métodos de segurança apresentados no decorrer do trabalho.

Uma rede de computadores, como o nome sugere, é a interligação de dois ou mais computadores, com o objetivo de trocar informações entre si usando uma dentre várias infraestruturas para este fim, rede via cabo, par de cobre e *wireless* são exemplos. Chama-se cada computador contido na rede de *host,* e para que cada *host* se comunique entre si com segurança e qualidade usamos regras, mais conhecidas como protocolos (Tanenbaum, 2003).

Segundo Tanenbaum (2003), os protocolos de rede são essenciais para o funcionamento eficiente da mesma, são eles que garantem que os computadores irão se comunicar com um padrão de informações, sem conflito entre eles, chama-se essas informações de pacotes. Para tal finalidade são necessários vários tipos de protocolos, com características parciais, e cada qual realizando diferentes funções. Eles são divididos em camadas, que por sua vez também executam diferentes trabalhos.

Nos próximos tópicos serão apresentados os tipos de rede e suas características de acordo com Tanenbaum (2003).

2.1 REDE LOCAL (LAN)

LAN é um termo que significa rede local (*Local Area Network*), é caracterizado por cobrir uma área de espaço reduzido, como uma rede de computadores comunicando-se entre si em empresas, escolas ou casas por exemplo. Além de computadores essas redes também são conhecidas por constituírem aparelhos como impressoras que podem ser acessados por todos os computadores, desde que autorizados pelo administrador da rede. Uma rede LAN não necessita ter necessariamente internet, uma simples estrutura onde tem-se por exemplo um roteador, e dois dispositivos ligados a este roteador se comunicando, caracteriza-se uma LAN.

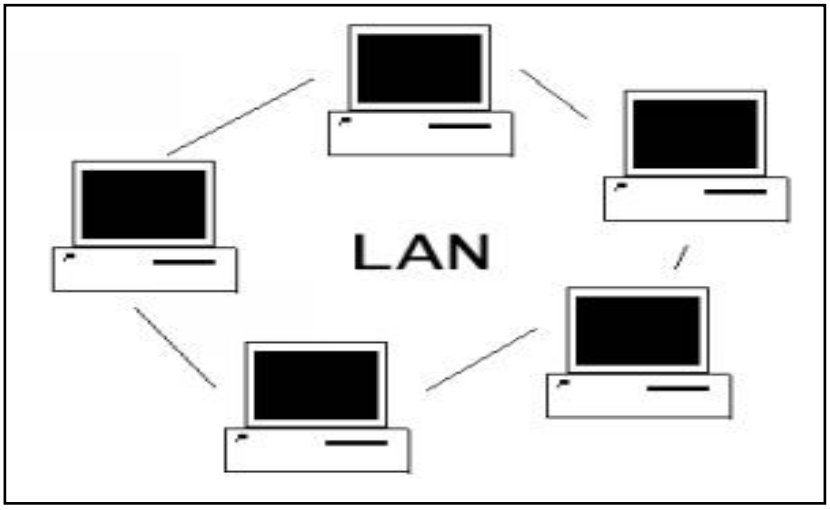

Figura 1 – Rede Local (LAN). Fonte: Site “A Informática”. < https://carlos1990.wordpress.com/category/pmmri/>

2.2 REDE MAN

Significa *Metropolitan Area Network* ou traduzido à risca, rede de área metropolitana. É uma rede que abrange um espaço maior que a já citada LAN, geralmente é resultado de várias redes locais interligadas, cobrindo frequentemente uma cidade ou um campus.

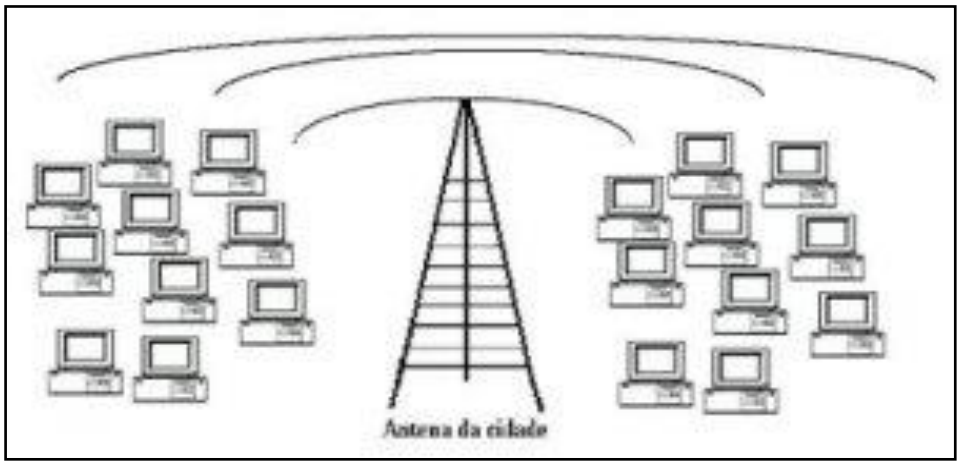

Figura 2 – Rede Metropolitana (MAN). Fonte: Site “A Informática”. < https://carlos1990.wordpress.com/category/pmmri/>

2.3 REDE WAN

Wide Area Network, é a rede usada para interligar redes longínquas, dispersa por um grande espaço geográfico, com porte e infraestrutura maiores e mais robustas que as anteriores.

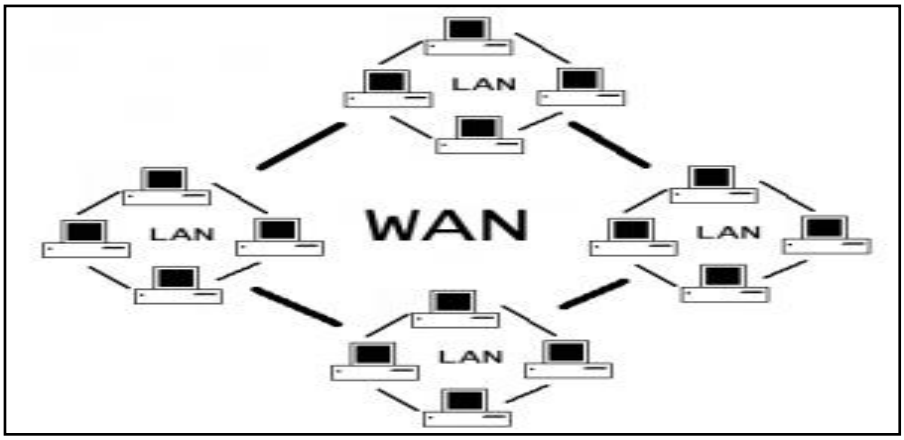

Figura 3 – Rede Larga (WAN). Fonte: Site “A Informática”. < https://carlos1990.wordpress.com/category/pmmri/>

2.4 TOPOLOGIA DE REDES

Topologia de redes é o modo como os *hosts* trocam pacotes um com o outro, e a estrutura utilizada para que este objetivo seja alcançado. São usualmente utilizados dois tipos de comunicação, o modo Ponto a Ponto e o modo Multiponto.

Ponto a Ponto é a basicamente a infraestrutura que possibilita comunicação entre dois dispositivos da rede, diretamente e sem a viabilidade de um terceiro dispositivo trocar informações com os dois primeiros (Tanenbaum 2003).

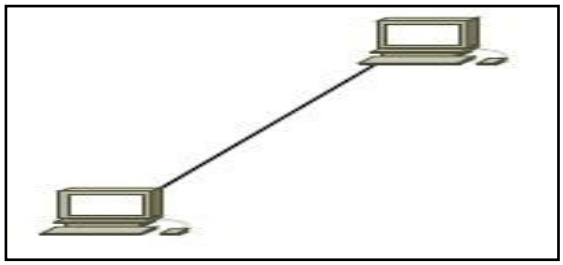

Figura 4 – Topologia Ponto a Ponto. Fonte Site: “Tecinformática”. < http://tecinformaticadecomputadores.blogspot.com.br/2015/04/introducao-1_11.html>

Na rede tipo Multiponto usa-se apenas uma forma de comunicação para vários *hosts* ao mesmo tempo, não há inevitabilidade de uma linha direta para troca de pacotes entre os dispositivos da rede, e os mesmos conseguem receber e enviar informações mais de um *host* ao mesmo tempo (Tanenbaum 2003).

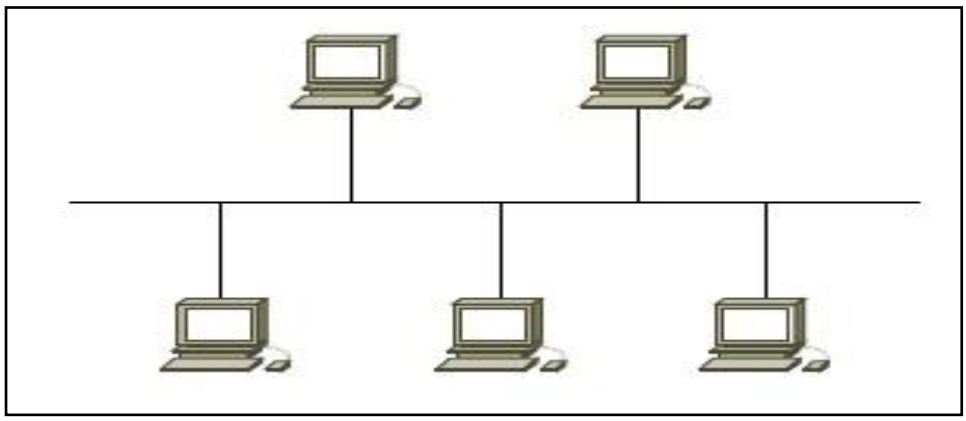

Figura 5 – Topologia Multiponto. Fonte Site: "Tecinformática". < http://tecinformaticadecomputadores.blogspot.com.br/2015/04/introducao-1_11.html>

2.5 PROTOCOLO TCP/IP (Transition Control Protocol / Internet Protocol)

Tanenbaum (2003), diz que a arquitetura TCP/IP é um agregado de protocolos responsáveis pela comunicação entre os *hosts*, é o TCP/IP quem soluciona os conflitos de compatibilidades entre os dispositivos de uma rede, tanto na internet quanto na intranet.

É esse conjunto de protocolos que faz a separação das funções para o sistema de comunicação em camadas, sendo divididas em: Camada de aplicação, Camada de Transporte, Camada de Inter – Rede, e por último a Camada de Rede. A seguir será especificado cada uma delas segundo Tanenbaum (2003):

Camada de Aplicação: Na Camada de Aplicação há um conjunto de protocolos de comunicabilidade, é ela quem faz a comunicação imediata entre a aplicação e as camadas de baixo nível.

Camada de Transporte: Como o próprio nome já diz, esta camada possui os protocolos para transição de pacotes, como o TCP (*Transmission Control Protocol*), e o UDP (*User Datagram Protocol*), vale ressaltar que está camada não é responsável pelos elementos contidos no meio do caminho do transporte, cuidando apenas da origem e do destino do pacote.

Camada de Inter – Redes / Internet: A Camada de Inter – Redes tem a tarefa de autorizar que os dispositivos enviem pacotes para variadas redes e assegurar que eles trafeguem independentemente até sua meta. É possível que os pacotes

cheguem em uma sequência desigual da qual foi enviado, porém as camadas superiores reordenarão eles.

Camada de Rede: Esta camada é incumbida de enviar datagramas (mensagens enviadas sem conexão e sem confirmação), criadas pela Camada de Inter – Redes, ela contém um mapeamento de endereço de recognição no grau físico das redes, quando se fala de redes *ethernet* cada estágio contém um endereço nomeado de MAC, (*Media Access Control*).

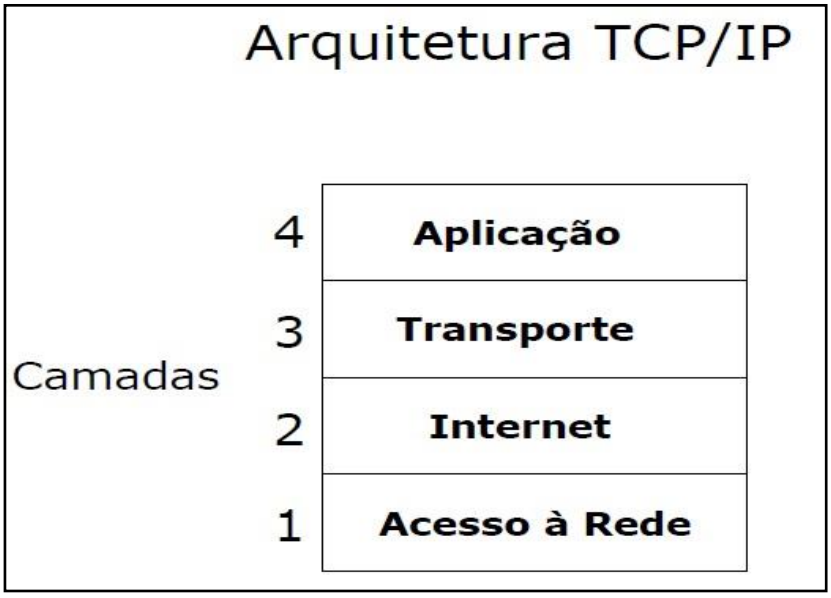

Figura 6 – Camadas TCP/IP. Fonte Site: "Tecinformática". < http://tecinformaticadecomputadores.blogspot.com.br/2015/04/introducao-1_11.html>

2.6 MODELO OSI (Open Systems Interconnect)

Segundo Tanenbaum (2003), comenta que assim como o TCP/IP, o Modelo OSI é um conjunto de protocolos de rede, separados em sete camadas: 7 – Aplicação, 6 – Apresentação, 5 – Sessão, 4 – Transporte, 3 – Camada de Rede, 2 – Link de Dados, 1 – Camada Física. A seguir será especificado cada uma delas:

7. Aplicação – Nesta camada é onde a aplicação requisita os arquivos para o sistema operacional, sem se importar com o que é necessário para obtê – lo.

6. Apresentação – A camada seis somente é requisitada quando necessário, funciona como uma camada extra, executando processos aditivos.

5. Sessão – Quando recebe a requisição da camada de aplicação, o sistema abre uma sessão, que é iniciada quando recebe a solicitação e encerrada somente na hora em que o problema for decidido.

4. Transporte – É na camada de transporte onde o protocolo TCP/IP age, juntamente com o sistema operacional para controlar a troca de pacotes, o sistema identifica qual é o endereço de IP do site, qual dos protocolos ele irá usar, entre outros dados importantes para assim enviar o pedido ao servidor que mantém o site, requisitando os dados que integram a página junto com o endereço de destino dos pacotes.

3. Camada de Rede – Nesta camada acontece o endereçamento de IP, a solicitação é convertida em um pacote de dados, e endereçada ao endereço IP do servidor do site.

2. Link de Dados – No Link de Dados se encontram os *switches* e as placas de rede, que conseguem ler frames e endereços MAC, executando o trabalho de transportar *frames* somente para o dispositivo correto. Em outras palavras eles encaminham os *frames* para a porta certa.

1. Camada Física – Na camada física trabalham os *hubs*, eles não reconhecem pacotes nem endereços de rede, apenas recebem dados binários em uma porta e os reenviam para as outras. O *hub* atua assim como um centralizador e repetidor, apenas transmitindo os dados.

Figura 7 - Camadas Modelo OSI. Fonte Site: "Tecinformática". < http://tecinformaticadecomputadores.blogspot.com.br/2015/04/introducao-1_11.html>

2.7 ROTEAMENTO

Segundo Tanenbaum (2003), roteamento das informações trocadas pela rede é uma das mais importantes tarefas executadas pela camada de rede, a função de um roteador é escolher em qual rota a informação será enviada. O pacote de dados virá

por alguma porta do roteador, que por sua vez reenviará esta informação para uma porta diferente, empregando uma lista de rotas. Existem dois tipos de listas de rotas, um deles é a tabela estática, que é definida manualmente, e a outra é a tabela dinâmica, que usa um protocolo que monta esta tabela de forma dinâmica.

O roteador utiliza um algoritmo de roteamento para escolher as rotas, que realiza cálculos para decidir qual é o caminho mais indicado para encaminhar o pacote de dados. Esses cálculos se baseiam em um padrão de medida chamo Métrica, que aplica parâmetros para delimitar qual a melhor rota, entre eles: a extensão do trajeto, a confiabilidade da rota, o atraso, a carga, os custos da transição, e a largura da banda (Tanenbaum, 2003).

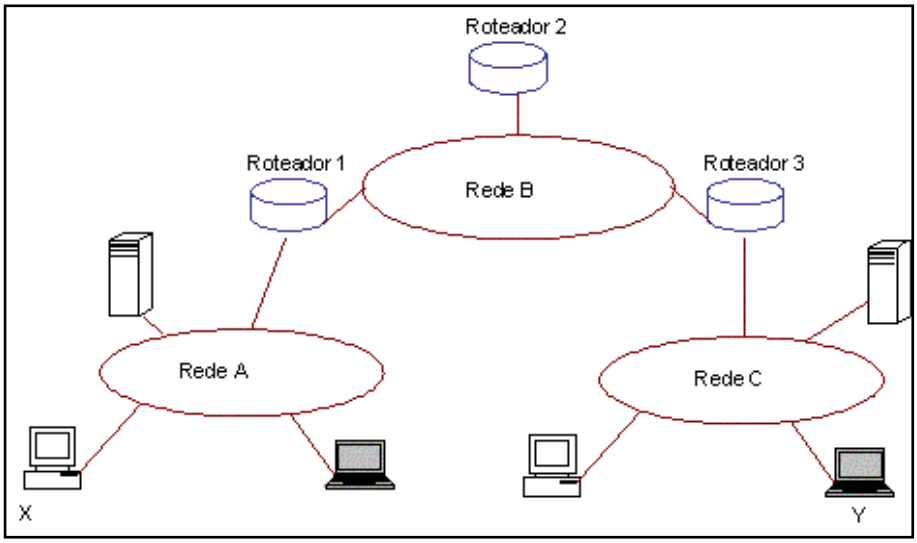

Figura 8 – Exemplo de Roteamento. Fonte: Site “Teleco”. < http://www.teleco.com.br/tutoriais/tutorialmpls/pagina_1.asp>

2.7.1 Roteamento Dinâmico

Tanenbaum (2003), informa que no caso de haver duas ou mais rotas possíveis para o mesmo destino, usa-se o roteamento dinâmico, que forma uma lista de rotas automaticamente com base na transição de pacotes de dados dos protocolos de roteamento.

2.7.2 Roteamento Estático

Opostamente ao dinâmico, o roteamento estático as listas de rotas são formadas de modo manual, atribuído a redes com uma menor quantidade de roteadores. Essas rotas predefinidas tem a possibilidade, ou não, de serem mostradas a outros *hosts*,

posto que a não autorização dessa informação é uma qualidade do roteamento estático, pois aumenta a segurança da rede. Este tipo de roteamento também tem a vantagem de minimizar o excesso de processamento produzido pela transição de informações de roteamento na rede (Tanenbaum, 2003).

Neste capítulo foi possível rever alguns conceitos importantes sobre os tipos de redes de computadores e o alcance que elas proporcionam no momento de conexão com outros computadores. O protocolo utilizado para comunicação de dados, os Ips e as tabelas de roteamento, que são necessárias para a construção de regras de iptables.

3 – SEGURANÇA DE REDES

Neste capítulo são apresentados os meios de defesa utilizados para a segurança de redes de computadores, características, vantagens, desvantagens, e vulnerabilidades desses métodos.

O conceito de defesa em tecnologia da informação significa diminuir a possibilidade de se explorar as fraquezas e minimizar essas próprias fraquezas de um sistema computacional. De acordo com Bernardes (1999), a segurança está associada à exigência de proteção contra o acesso proposital ou não, de dados confidenciais por elementos não permitidos, e o uso não autorizado do dispositivo da rede.

3.1 INTRUSOS (HACKERS)

O termo *hacker* nem sempre foi associado a um indivíduo mal-intencionado que promove ataques à sistemas computacionais, nos anos 70 e 80, essa definição era usada para pessoas com grande conhecimento na área de tecnologia da informação, em especial sistemas operacionais e programas. Depois de algum tempo esse conceito continua sendo coligado à essas pessoas com muito conhecimento em informática, porém usando essas informações para praticar atos ilícitos como invasões em servidores e ataques à sistemas computacionais (Bernardes, 1999).

Bernardes (1999), diz que além do termo *hacker* ainda existe outra definição semelhante chamada *cracker*, assim como o primeiro, o *cracker* possui grande conhecimento na área de tecnologia de informação, a diferença é que um *hacker* geralmente tem apenas o propósito de invadir sem causar danos ao sistema, ele somente deseja provar que consegue efetuar tal ato, onde quanto mais complicado seja adentrar esse sistema, mais ele se empenha, já um *cracker* tem o intuito de causar danos ao servidor, e consequentemente aos seus proprietários.

Segundo Bernardes (1999), a maior responsabilidade pelos inúmeros sistemas com baixa proteção, é porque a internet não foi projetada, mais especificamente o protocolo TCP/IP, para os seus diversos usos atuais. As primeiras intenções é que a internet fosse utilizada somente por militares e instituições de pesquisas, como sabe-se, a rede mundial de computadores cresceu de forma exponencial, com inúmeras aplicações, comércio eletrônico, e muitas outras características e serviços disponíveis nela, e isso tudo não foi previsto no projeto e no foco inicial. Desse modo, os protocolos, serviços, sistemas operacionais, e as aplicações utilizadas na internet não foram criados com os necessários cuidados em se tratando de proteção.

3.2 PROTEÇÃO COMUNICACIONAL

De acordo com Bernardes (1999), quando se fala em segurança em comunicações, deve-se entender a proteção aos pacotes de dados que trafegam na rede, Belovins (1989, citado por Bernardes, 1999), fala que como o conjunto de regras do TCP/IP é característico por ser vulnerável, variados recursos precisam ser postos em prática agrupados com o propósito de se maximizar a proteção original do protocolo. Os pacotes de dados que transitam na rede pelo TCP/IP são abertos, ou seja, uma pessoa com uma interface de rede adaptada para obter as informações desses dados e acesso ao meio físico consegue ter entrada aos pacotes que estiverem trafegando, este método é nomeado de *Sniffing*.

Existem duas categorias de intrusos, uma delas é chamado de passivo, o qual é denominado aquele intruso que somente supervisiona o trafego da rede, com intuito de obter dados confidenciais, a segunda classificação de intruso é intitulada de intruso ativo, que por sua vez trabalha alterando o contento dos pacotes de dados. Deste modo são necessários alguns atributos para se conseguir se comunicar de forma segura. Bernardes (1999) informa que esses requisitos são:

Confidencialidade: Somente os indivíduos incluídos deverão ter acesso às informações que transitam na rede, intruso nenhum pode conseguir acesso à esses dados.

Integridade: O mesmo dado que é enviado de sua origem, deve ser o que é recebido em seu destino, o pacote não pode ser adulterado, nem por algum tipo de intruso, nem por algum tipo de falha na rede.

Autenticidade: É necessário que todos as partes que estão se comunicando devam ter meios de, reciprocamente, se identificar, sabendo assim, com quem estão trocando informações.

Garantindo esses três requisitos em conjunto, aumenta-se consideravelmente o nível de segurança na comunicação de informações computacionais.

3.3 CRIPTOGRAFIA

De acordo com Tanembaum (1997, citado por Bernardes, 1999), o melhor método de proteção usada nas comunicações é a criptografia, ela foi criada com intuito de dar segurança às mensagens enviadas por meio de um método de comunicação não seguro, onde não há como garantir que um intruso esteja impedido de ter acesso às informações, e adultera-las.

A técnica de criptografia consiste basicamente em transformar o texto original da mensagem, em outro texto, substituindo cada caractere por outro correspondente, o mesmo algoritmo que faz essa troca deve ter a capacidade de fazer o inverso, ou seja, descriptografar o texto final, destrocando os caracteres dele pelos verdadeiros, dando forma assim ao texto original. Tomamos por exemplo um método de criptografia em que se substitui uma letra pela letra posterior a ela no alfabeto, seguindo este método a palavra SEGURANÇA daria origem a palavra criptografada TFHVSBODB, assim se um intruso interceptasse uma mensagem com essa palavra, apenas teria esse “amontoado” de letras como informação, este é um método básico e simples, que não é usado nos dias de hoje e serve apenas como base para sistemas muito mais complexos de criptografia (Tanenbaum, 2003).

Usando este método é possível se assegurar a confidencialidade das informações que serão mandadas para a rede através de uma conexão, porém é somente isso que ela pode garantir, caso algum intruso intercepte a mensagem, ele tem toda a possibilidade de alterar o conteúdo da mensagem, o mesmo ainda pode usar um ataque chamado *man – in the – middle* onde um terceiro indivíduo se coloca entre as duas pessoas que estão trocando mensagens, e age de modo com que essas duas pessoas pensem estarem falando entre si, quando na verdade o intruso é quem responde as duas (Bernardes, 1999).

Figura 9 – Criptografia: Codificação usando chave simétrica. Fonte livro **“**Computer Networks - Vol: 4” (2003).

Figura 10 – Criptografia: Decodificação usando chave simétrica. Fonte livro **“**Computer Networks - Vol: 4” (2003).

3.4 PERIGOS E ATAQUES À REDE

Bernardes (1999), chama a possibilidade de uma invasão à segurança de um sistema computacional de ameaça, das quais as mais comuns são: apagamento de dados e informações, adulteração dos dados, apropriação não autorizada ou a perda de informações, exibição não autorizada de informações particulares e a paralização de serviços.

As ameaças computacionais podem ser categorizadas como intencionais ou acidentais, onde tanto uma quanto a outra podem ser classificadas como ativas e passivas. Uma ameaça intencional é aquela que não tem como motivo falhas no sistema, defeitos nos equipamentos, se concluída a ação dessa ameaça ela pode ser chama de ataque, e como o próprio nome já diz, o causador da ação teve a intenção premeditada de atacar o sistema. As ameaças passivas decorrem de motivos não intencionais, desde erros humano, até defeitos no equipamento e falhas no sistema (Bernardes, 1999).

Segundo Bernardes (1999), uma ameaça passiva, seja ela intencional ou acidental, é aquela que, depois de concretizada, não altera o conteúdo dos dados e das informações, e uma ameaça ativa, tanto a intencional como a acidental é aquela que após sua conclusão, altera o teor dos documentos digitais contidos no ambiente onde ela foi realizada.

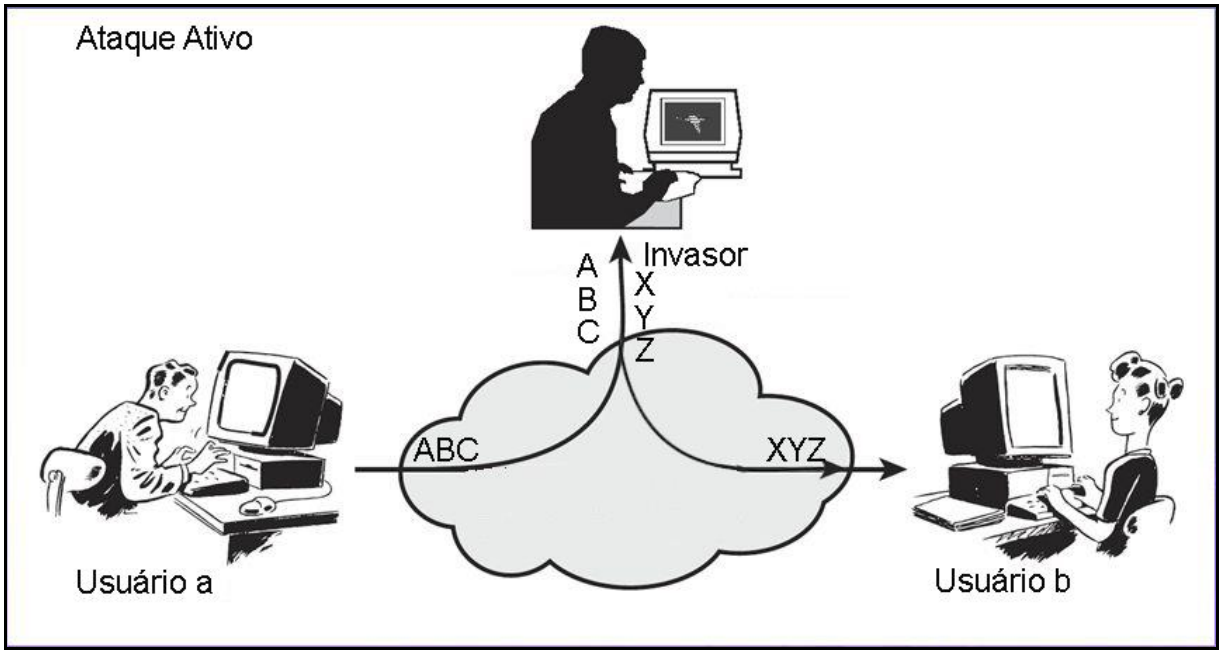

Figura 11 – Exemplo de Ataque Ativo. Fonte: Site “Doc Player”. <http://docplayer.com.br/10451297-Criptografia-e-seguranca-em-rede-capitulo-1-de-william-stallings.html>

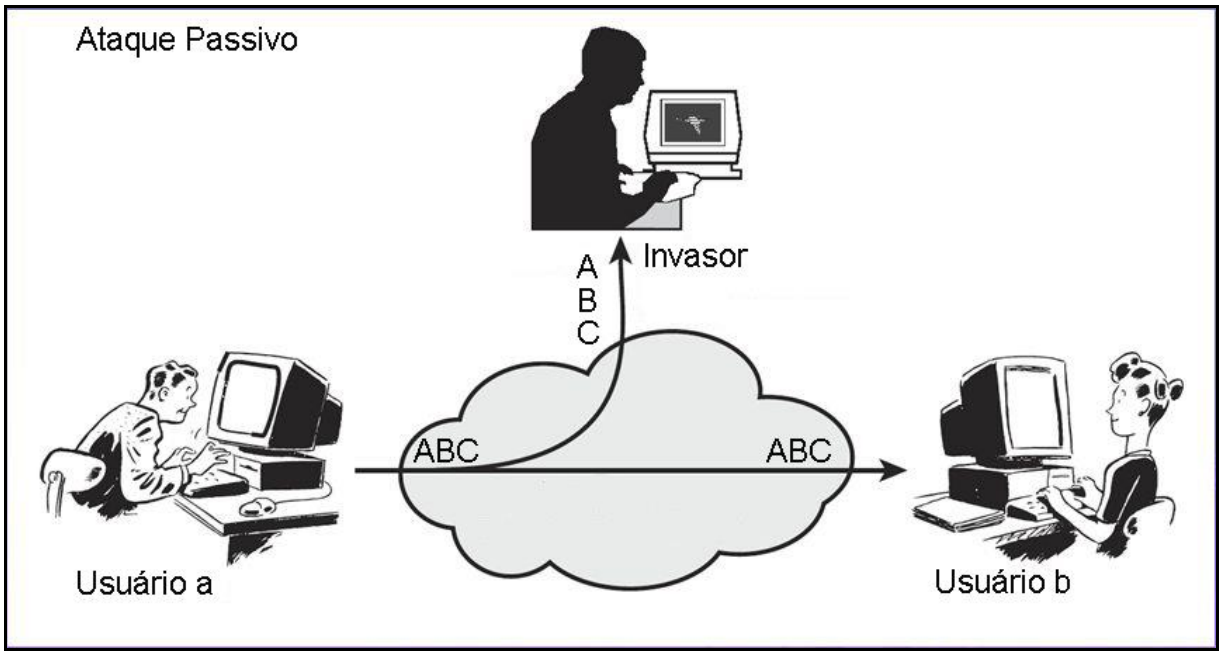

Figura 12 – Exemplo de Ataque Passivo. Fonte: Site “DocPlayer”. <http://docplayer.com.br/10451297-Criptografia-e-seguranca-em-rede-capitulo-1-de-william-stallings.html>

Os ataques mais comuns em sistemas computacionais pela rede de acordo com Bernardes (1999), são:

Alteração: O teor dos documentos digitais é modificado, sem autorização e sem que o sistema consiga perceber a alteração.

Ataques Interiores: É quando pessoas de dentro daquele próprio ambiente computacional, agem de modo ilícito, executando tarefas não autorizadas.

Replay: Acontece quando um pacote de dados é interceptado, e após algum tempo enviado ao seu destinatário original, produzindo efeitos não permitidos.

Personificação: Um terceiro identifica-se como um usuário para usufruir de seus privilégios no sistema.

Ataques Exteriores: Ocorre quando pessoas conseguem por uma conexão externa invadir o sistema.

Interrupção do Serviço: Chama-se de interrupção do serviço quando um elemento do sistema não faz suas tarefas de modo correto, ou impede que outros elementos façam suas tarefas eficientemente.

Armadilha: Acontece quando um elemento do sistema é alterado com intuito de agir sem autorização em resposta a uma ação comandada, ou a um seguimento de eventos antes determinados.

3.5 FIREWALLS

Firewall é um elemento de proteção muito usado para maximizar a segurança em redes, dificultando ataques exteriores ao sistema, ele é uma barreira segura, que impede o acesso a defeitos de proteção do sistema (Neto, 2004).

Segundo Neto (2004), um programa *firewall* não pode corrigir falhas e vulnerabilidades em protocolos, porém umas de suas várias utilidades é delimitar o uso da internet em uma rede, ou seja, o administrador não precisa que a rede tenha acesso a toda internet, ele tem a possibilidade quando usa um *firewall*, de apenas autorizar o acesso a seu sistema, redes e host de sua confiança.

O *firewall* pode agir por um método chamado de “ponto de indução”, onde sendo o único *host* diretamente ligado à internet, tem a possibilidade de um modo seguro, enviar serviços de interconectividade para sua LAN. Bernardes (1999), fala que existem duas abordagens básicas para a configuração de um *firewall*, onde a primeira seria o conceito de se autorizar tudo o que não é expressamente proibido, neste método, o administrador de rede, deve antecipar todos os tipos de tarefas que os usuários ou intrusos possam executar fora da política de segurança e se precaver com proteções contra elas. A outra abordagem é que tudo o que não for expressamente autorizado é bloqueado, neste caso o administrador deve configurar o *firewall* para bloquear todos os serviços, e ir liberando um a um, posteriormente à um minucioso estudo de riscos e necessidade dos mesmos. Geralmente a primeira opção é mais usada em entidades de estudo e pesquisar, onde o acesso à internet necessita ser mais amplo, já o segundo método é mais utilizado por instituições com maior rigidez e regras bem definidas, uma vez que o administrador sabe exatamente cada serviço que a mesma necessita.

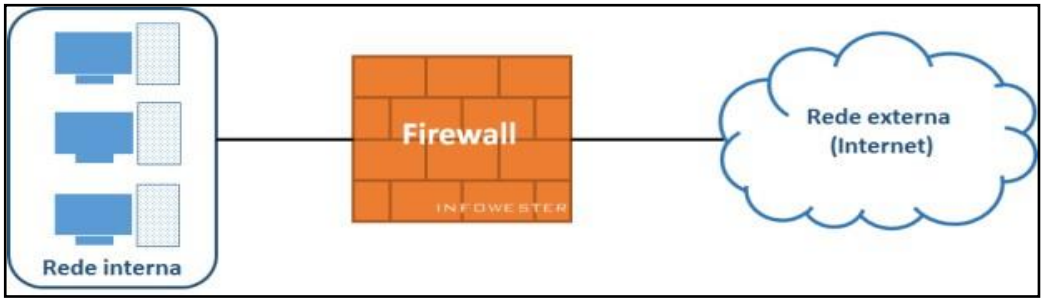

Figura 13 – *Firewall*. Fonte: Artigo “Segurança Computacional” (1999).

De modo geral um *firewall* de acordo com Neto (2004), é constituído basicamente pelos dois seguintes elementos:

Filtros (*Screens*): Esses componentes bloqueia a transmissão de algumas categorias de trafego.

Componente *Gateway*: Este elemento é uma máquina, ou um agregado delas, conectadas por um segmento de rede, que oferece serviços de retransmissão.

É usado um filtro entre o *gateway* e a rede externa chamado filtro de saída e um filtro entre o *gateway* e a rede interna nomeado de filtro interno, o primeiro funciona como um protetor para intrusos e ameaças externas, já o filtro interno é o responsável pela proteção da rede interna, na situação onde um ataque consegue comprometer o correto funcionamento do *gateway*. Simplificando, os dois filtros trabalhando sozinhos ou em conjunto, tem o objetivo de proteger a rede interna de possíveis ataques, ameaças e intrusos da rede externa (Neto, 2004).

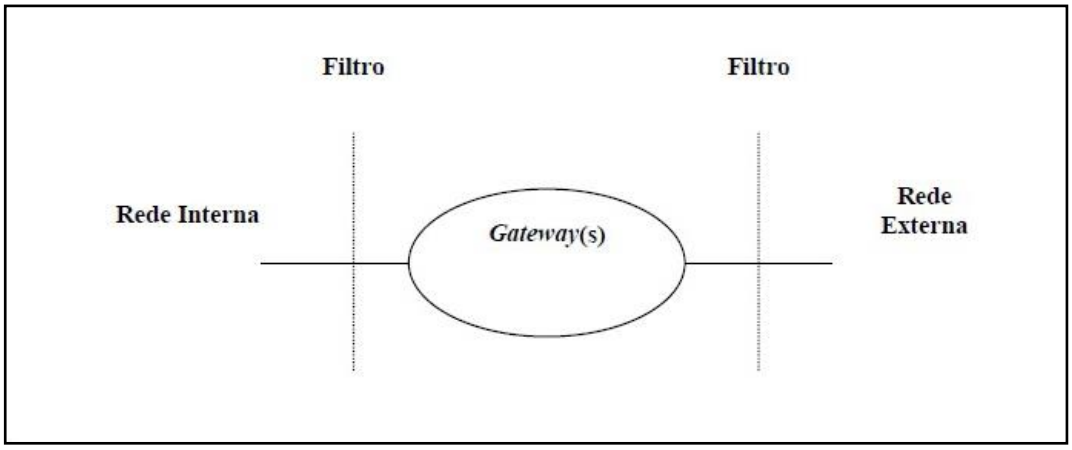

Figura 14 – Componentes de um *Firewall*. Fonte: Artigo “Segurança Computacional” (1999).

De acordo com Neto (2004), um *firewall* não é efetivo quando fala-se em vírus, visto que este tipo de problema geralmente é ativo já na rede interna do computador, geralmente por trocas de e-mails contaminados por usuários de sua própria lista de contatos, também por download de arquivos como vídeos e músicas por compartilhadores de arquivos, ou até por mídias exteriores como pen-drives, CDs ROM, discos rígidos externos entre outros dispositivos de memória de terceiros que possam conter os mais diversos tipos de vírus. Visto que todas essas ações são permitidas pelo próprio usuário, um *firewall* não tem como inibir este tipo de ameaça, para este fim é indicado um *software* antivírus, desenvolvido exatamente para este fim. Em relação a este tipo de problema, o máximo que um *firewall* conseguirá fazer

será é não deixar que sua rede seja monitorada por vírus do tipo *trojan*, evitando que este tipo de *software* mal intencionado troque dados não autorizados pelo usuário com outros dispositivos da rede externa, para isso é necessário configurar um bloqueio para as tentativas de conexão adventos da internet para os *hosts* de sua rede, sob uma determinada porta ou mesmo um *host*, monitorando os segmentos que transitam procurando palavras chaves contidas em pacotes não permitidas.

Tendo isso em mente subentendesse que um *firewall* tem a função de defender ameaças externas, dentre vários tipos de métodos ele executa esta tarefa especificando os tipos de protocolos e serviços que serão autorizados tanto na rede interna quanto na rede externa, fazendo compartilhamentos de acesso na internet para a parte interna do sistema não autorizando que se comuniquem de modo direto, monitorando conexões, impedindo acessos não autorizados à sites e *hosts* não permitidos, e principalmente monitorando os dados usados por serviços não confiáveis da internet (Neto, 2004).

3.5.1 Histórico dos *Firewalls*

Neto (2004), afirma que um *firewall* é um software que detém a autonomia permitida pelo próprio sistema com intuito de pré-determinar e disciplinar todos o tipo de trânsito que está situado entre o *firewall* e outros dispositivos da rede e mesmo entre outras redes. Um bom *firewall* é praticamente o meio mais seguro de enviar e receber serviços de interconectividade a dispositivos e redes.

O primeiro *firewall* do mundo, segundo Neto (2004), desenvolvido coma função de filtrar todos os dados enviados e recebidos por uma rede corporativa, manipulando-os de modo a obedecer as regras pré-determinadas foi criado pela empresa Bell nos anos 80, à pedido da empresa de telecomunicações AT&T. Desde então mesmo com o crescimento exponencial da área de tecnologia da informação, este tipo e *software* continua usando os mesmos conceitos do primeiro desenvolvido pela empresa Bell, obviamente com vários aprimoramentos, alterações e adições de novas funções, visto que nos dias de hoje a principal função de um *firewall* é a filtragem de pacotes de dados advindos de uma rede, porém essa não é mais sua única utilidade, tanto que podemos subdividi-los em três tipos de classificações: *firewall* filtro de pacotes, *firewall* NAT e *firewall* híbrido. Neto (2004), fala que as classificações e as devidas funções e diferenças de um *firewall* são:

***Firewall* Filtro de Pacotes:** Esta categoria de *firewall* tem a função de filtrar todo tipo de trânsito que tem conexão com o *host* ou a rede que este *firewall* protege, assim como os dados enviados por ele ou sua rede. Ele realiza este filtro a partir das regras pré-determinadas pelo seu administrador.

Este tipo de *firewall* é responsável por analisar cabeçalhos (*headers*) de dados durante sua transição, tomando assim a decisão do destino que aquele pacote de dados irá tomar, mediante regras previamente impostas pelo administrador da rede o *firewall* decide se a informação será parada e ignorada, ou se ele pode prosseguir com seu trajeto. Pode-se dizer que este é tipo de *firewall* mais usado nos sistemas, pois abrir mão do mesmo é permitir que pacotes de dados entrem e saem livremente de sua rede sem nenhum tipo de verificação.

***Firewall* NAT:** Este *firewall* possui a capacidade de manipular a rota padrão de pacotes de dados que passam pelo *kernel* do *host Firewall*, adicionando diversas funcionalidade ao mesmo, como a de alterar o endereço de origem do pacote chamado de *SNAT* e também o endereço de destino do pacote nomeado de *DNAT*, assim como a funcionalidade de executar as tarefas de um *proxy*, talvez não com a mesma rapidez, porém com a máxima proteção.

***Firewall* Híbrido:** Um *firewall* híbrido utiliza-se tanto das tarefas de filtragem do *firewall* filtro de pacotes quanto as utilidades do *firewall* NAT, sendo, portanto, uma junção dos dois primeiros e não apenas uma utilidade isolada.

3.5.2 *Firewall Linux*

No sistema operacional Linux as utilidades do *firewall* são acopladas à própria arquitetura do *kernel*, segundo Neto (2004), isto é, sem dúvidas, uma grande vantagem em relação aos seus concorrentes.

Os pacotes que transitam pelo *host* são processados pelo kernel de qualquer sistema operacional, o que diferencia o Linux dos demais, é que ele agrega por meio de um programa do sistema chamado de *Netfilter*, utilidades de monitoramento do tráfego interior em termos de *firewall* (Neto, 2004).

De acordo com Neto (2004), o *kernel* tem a tarefa de núcleo o sistema, sendo assim ele deve monitorar todos as informações que transitam em sua estrutura, tudo o que é lido e executado, escrito ou apagado, redirecionado ou encaminhado, em outras palavras, tudo o que ocorre em seu sistema deve ser autorizado pelo *kernel* do sistema operacional. Levando em consideração que estas tarefas exigem do *kernel* um grande tempo de trabalho do *kernel*, e alguns sistemas operacionais tem dificuldades em operar suas tarefas mais básicas como monitorar seu processos e serviços, é possível entender a razão pela qual nem todo sistema operacional consegue também monitorar o fluxo de dados de sua estrutura, buscando alternativas secundárias para esta tarefa, acoplando em seu sistema ambientes de monitoramento de fluxo de informações. Já o sistema operacional Linux tem a capacidade de executar todas essas tarefas sem dificuldades, de forma eficiente e tornando muito mais seguro.

Para que o *kernel* do Linux tenha a capacidade de monitorar seu próprio trafego de dados em seu interior lhe foi acoplado uma aplicação chamada de *Netfilter*, desenvolvido por Marc Boucher, James Morris, Harald Welte e Rusty Russel, esta ferramenta é um agregado de elementos de fluxo de informações acopladas ao *kernel* do Linux e separadas em listas (Neto, 2004).

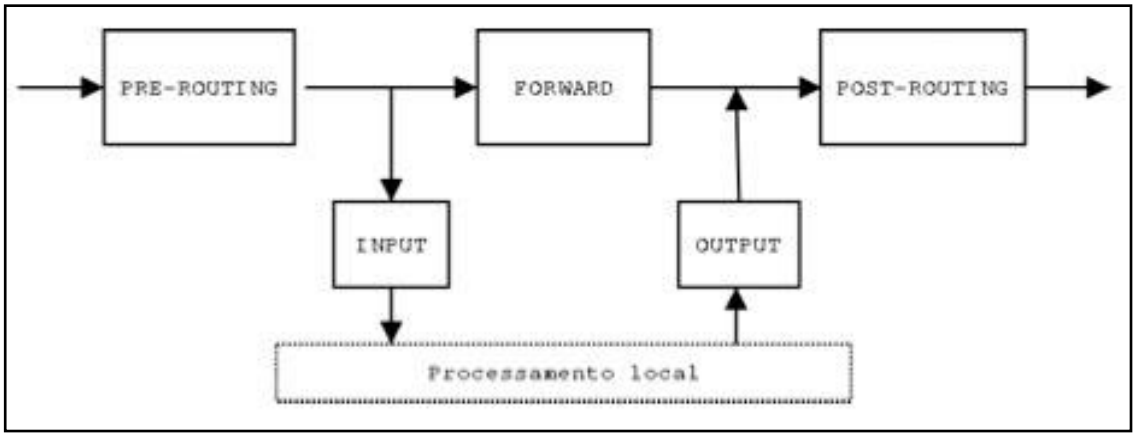

Figura 15 – *Netfilter*. Fonte: Artigo “Segurança Computacional” (1999).

Neto (2004), explicando de um modo mais simples, diz que o *Netfilter* pode ser visto como um grande banco de dados, que contém três listas padrões, chamadas de *Filter*, NAT e *Mangle*. Cada tabela dessa possui protocolos direcionados as suas funções. As três possuem características de fluxo as capacitam para a execução de suas funções.

A seguir será explicado, segundo Neto (2004), com mais detalhes a funções de cada uma dessas tabelas:

Tabela *Filter*: Assemelha-se ao já citado *firewall* filtro de pacotes, contendo os elementos: *Input* que seria tudo o que entra na no *host*, o elemento *Forward*, tudo que entra no dispositivo, porém tem que ser redirecionado a um dispositivo secundário da rede ou a uma terceira interface de rede, e por último o *Output* que seria basicamente o que sai dos dispositivos de rede.

Tabela Nat: Tem as tarefas de NAT, assim como o *Firewall* NAT, é constituída por três situações: *prerouting*, é usada no caso de se precisar fazer modificações nos pacotes de dados antes do roteamento, *output*, responsável pelos pacotes enviados pelo *host firewall* e por fim o *postrouting*, usada no caso quando se necessita fazer modificações nos pacotes de dados após o roteamento.

Tabela *Mangle*: Essa tabela implementa modificações especiais em dados de um tipo com maior complexidade. A *Mangle* tem a capacidade de modificar o nível de

importância de entrada e saída de um dado tomando por base a categoria do serviço ao qual o mesmo se destinava. Tem como situações: *prerouting*, que altera os dados lhes dando uma tratativa especial antes do roteamento, e o *output*, que faz a tratativa nos dados de modo especial criados no local antes do roteamento.

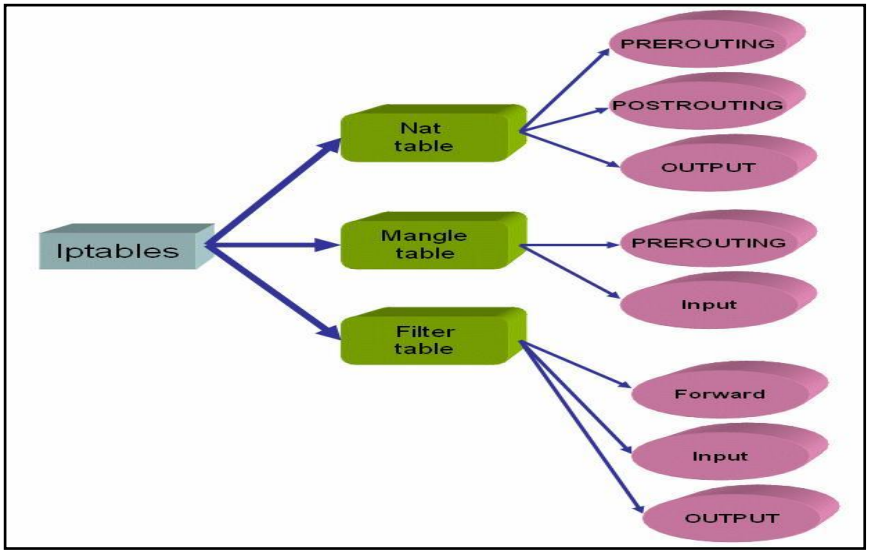

Figura 16 – Tabelas do *Netfilter*. Fonte: Site “OnlyTutorials”. <http://www.onlytutorials.com.br/2008/11/26/iptables-tabela-mangles/>

De acordo com Neto (2004), essas três tabelas do *Netfilter* nos dá a possibilidade de monitorar e controlar todas as ocorrências de um dispositivo de rede, no entanto para que tenhamos a capacidade de modelar o *Netfilter* de acordo com nossas preferências, levando em conta que ele tem que estar compilado com o *kernel* do Linux, é necessária uma ferramenta que faça o papel de um *Front-End* nesta função. Um *Front-End* nos dá a capacidade de administrar as ocorrências das tabelas acoplando os mesmos protocolos de tráfego. Protocolos esses que são as regras pré-determinadas aplicadas com o intuito de regularizar o trânsito de informações em uma rede ou *host*, o *Front-End* disponibilizados pelo sistema operacional Linux nos dias de hoje é chamado de *Iptables*, que será explicado com mais detalhes no capítulo a seguir.

Neste capítulo foram revisados vários detalhes importantes para o entendimento sobre a intrusão e proteção dos dados contidos nos computadores, maneiras de manter e transferir dados seguros através da criptografia. E o mais importante de todos a proteção através de *FIREWALL*, que existem para impedir a entrada de acesso a computadores que não tenham a devida permissão de acessar dados internos.

4 - *IPTABLES*

Neste capítulo será apresentado a ferramenta *Iptables*, um *Front – End* usado para manipular as tabelas do *Netfilter*, mostrando assim as características desse sistema de defesa, suas vantagens e pontos fortes, assim como possíveis falhas e vulnerabilidades. Também serão mostrados os métodos de manipulação dessa ferramenta.

Conforme já abordado no capítulo anterior, o *Iptables* é um *front – end*, responsável para administrar as situações das tabelas, inserindo regras de fluxo nestas tabelas. Segundo Dos Santos, para realização de tal tarefa o *Iptables* utiliza-se de três chains: *INPUT, OUTPUT* e *FORWARD*. As informações que chegam percorrem um módulo de pré encaminhamento dos dados, que decide se esses dados serão enviados ao conjunto de protocolos *INPUT*, ou ao agregado de regras *FORWARD* (Neto, 2004).

Quando um pacote de dados ao *firewall* for aceito pela *chain INPUT*, ele é entregue localmente, já se ele for aceito pela *chain FORWARD*, ele é enviado para a interface melhor apropriada. Os pacotes de dados de processos locais são enviados para a *chain OUTPUT*, caso o pacote de dados seja aceito, é enviado à interface apropriada, os pacotes são filtrados somente uma vez, sendo exceção os pacotes de *loopback*, pois estes são filtrados duas vezes (Neto, 2004).

4.1 TABELAS DE REGRAS

O *Iptables* é constituído por três tabelas de protocolos para pacotes de dados processados divergentes um do outro. As tabelas são utilizadas como módulos com separadas funções. De acordo com Neto (2004), os três principais módulos são a tabela de protocolos *filter*, a tabela *Network Adress Translation* (*NAT*), e a tabela usada para manipulação de dados intitulada tabela *Mangle*. Conforme a imagem 17 a seguir, todas essas três tabelas têm extensões que são dinamicamente iniciadas logo ao serem referenciadas.

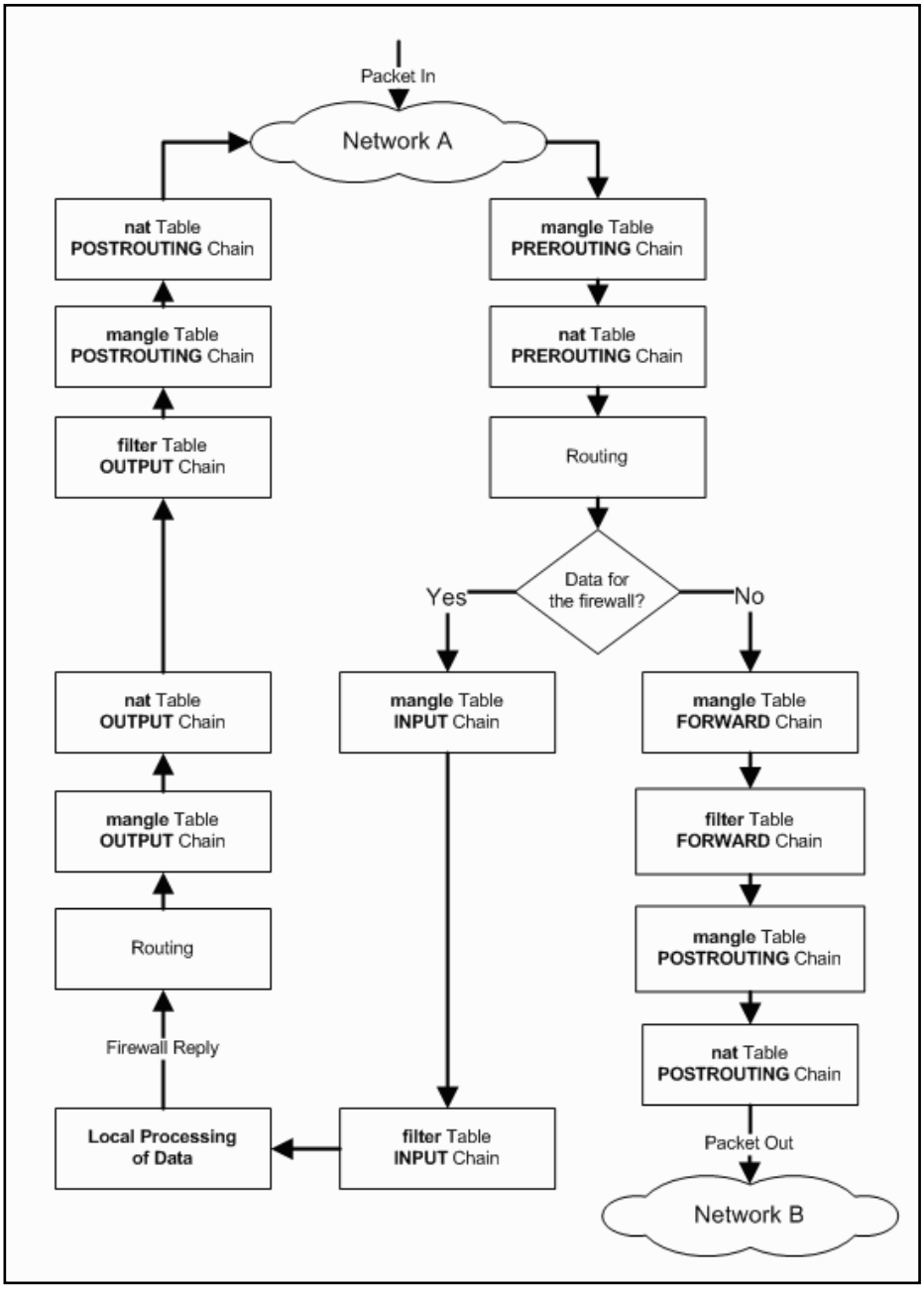

Figura 17 – Diagrama *Iptables*. Fonte: Site “LHN”. <http://www.linuxhomenetworking.com/wiki/index.php/Quick HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables#.VumexelrLIU>.

4.1.1 Tabela Filter

Segundo Santos, (2002), a tabela *filter* é responsável por operações relacionadas com as três *chains* de protocolos base, sendo elas: *Input*, *Output*, e *Forward*, e também as chains que são pré-determinadas pelo administrador da rede. É também função da filter aceitar ou negar determinados pacotes de dados, encontrar *matchs*, nos cabeçalhos dos IP’s, dos protocolos, dos endereços origem e destino, das interfaces de entrada e saída, e de manuseio de fragmentos. Ainda é tarefa desta

tabela executar *matchs* de dados contidos nos campos dos cabeçalhos UDP, ICMP e TCP.

Há na tabela filter dois módulos de extensão intitulados de *match* e *target*. A seguir será especificado o funcionamento de ambos de acordo com Santos (2002):

Módulo de extensão *match*: Este módulo identifica pacotes utilizando-se das funções:

Estado de ligação TCP: Utiliza-se de todas as *flags*, inspecionando-as e tomando decisões de filtragem por meio dos resultados das inspeções, por meio destas *flags* torna-se possível identificar se um determinado pacote faz parte de uma ligação pré-determinada e aceito pelo *firewall*, tendo a possibilidade de aceitar ou não o pacote de dados sem a necessidade que o mesmo passe pelos protocolos que o pacote inicial passou.

Campo *option* do TCP: Um pacote TCP tem a possibilidade de determinar qual o máximo tamanho de um segmento que o emissor aceita em retorno. Há possibilidade de filtragem por meio deste valor.

Listas de Portas: É uma lista de portas origem e destino (suportado pelo módulo *multiport*), que tem a capacidade de suportar no máximo quinze portas, estas listas de portas não tem a capacidade de serem combinadas.

Endereço *ethernet* MAC: Quando um pacote chega ao *firewall*, ele pode ser filtrado por meio do endereço MAC do usuário emissor, porém esta função só é válida como autenticação local, tendo em vista que os endereços MAC's só estarem contidos nas informações entre dispositivos de redes adjacentes.

Utilizador, grupo, processo, ou ID do grupo de processos: Os dados criados localmente, tem a possibilidade de serem filtrados por meio do utilizador, grupo, processo, ou ID do grupo de processos do *software* que criou o pacote. Sendo assim, o acesso remoto a certos dispositivos pode ser permitido a nível do *firewall* por meio do utilizador.

Campo TOS (*Type of Service*): Este campo, contido no cabeçalho do pacote IP, tem somente interesse histórico, ele é geralmente ignorado ou utilizado como diferenciação de serviços por routers intermediários, utilizando como exemplo para um tipo de fluxo ter um encaminhamento determinado.

Campo *Iptables mark*: Determinado pela tabela *Mangle.*

Delimitar o *Matching* de pacotes: A aceitação de certos tipos de dados tem a possibilidade de ser delimitada a uma taxa inicial de dados por segundo, caso essa taxa seja atingida, cria-se uma nova taxa com maior restritividade para ser usada nos dados seguintes.

De acordo com Dos Santos, caso a delimitação de *match* de dados esteja ativa, o procedimento por omissão tem como limite introdutório cinco pacotes de dados por segundo, caso essa delimitação for atingido, um limite de três pacotes por hora é imposto.

Módulo de Extensão *Target*:

Target LOG: O módulo de extensão *target* tem a capacidade de efetuar *log's* de alguns dados filtrados. As mensagens de *log* têm a possibilidade de serem associadas a diferentes níveis de *Kernel Logging*, determinado no diretório "etc" do sistema operacional Linux, mais precisamente no arquivo "syslog.conf", permitindo assim que os *log's* dos dados sejam ligados ou desligados e também autorizando definir os ficheiros de *outputs* para os mesmos.

Target Reject: Este módulo pode, por opção, especificar qual o tipo do erro ICMP devolver ou deletar um pacote. O *Standart* IPv4 determina que o TCP aceita tanto um pacote RST quanto um pacote ICMP como determinação de erro. O procedimento por omissão é o de nada devolver, ou devolver apenas um erro ICMP.

Target QUEUE: Este target permite enviar os dados por meio *netlink* a tratativa do pacote a um determinado *software*, caso o mesmo não se encontrar a escuta, o pacote é deletado.

Target Return: o *Target Return* dá a capacidade de retornar de momento imediato de uma *chain* de utilizador, mesmo que o processamento dos dados nessa cadeia tenha terminado.

4.1.2 Tabela NAT

Santos (2002), diz que essa tabela possui três tarefas principais para sua utilização: NAT tradicional unidirecional, NAT bidirecional e o NAT duplo. A seguir uma especificação mais detalhada de cada tarefa de acordo com o mesmo.

NAT Tradicional Unidirecional: Esta função possui o NAT básico, onde se traduz o endereço dos pacotes de dados, geralmente é usado para mapear endereços particulares com um agregado de endereços públicos. Há também o NAPT, usado para mapear endereços particulares para somente um endereço público.

NAT Bidirecional: Esta tarefa traduz biredicionalmente de endereços de dados que chegam ou que saiam do *firewall*, tomamos por exemplo a conversão bidirecional entre endereços IPv4 e IPv6.

NAT Duplo: Tem a capacidade de traduzir biredicionalmente os endereços de origem e destino. Tem a possibilidade de ser usado caso os endereços de rede de origem e de destino se colidam ou quando os endereços de uma rede forem

modificados e o usuário não deseja modificar os endereços de cada máquina da rede, uma por uma.

Esta tabela também contém módulos de extensão para a tradução de endereços origem e destino e para traduzir portas (PAT). Eles suportam as formas de NAT: SNAT, DNAT, *MASQUERADE* e *REDIRECT*.

Além disso a tabela NAT contém três *chains* de protocolos base:

***Prerouting*:** Esta cadeia de regras tem a capacidade de modificar o endereço destino de os pacotes de dados antes dos mesmos serem roteados, a modificação do destino pode ser tanto ao *firewall*, quanto a outro dispositivo.

***Input*:** A cadeia de regras *Input* tem a capacidade de modificar o destino de um pacote de dados criado no próprio *firewall*, antes mesmo das decisões de encaminhamento, geralmente usado para redirecionar um dado que saí do *firewall* e vai para um *proxy* local, e ainda para fazer *port forwarding* para um *host* diferente.

***Postrouting*:** É usada para modificar o endereço origem de pacotes de dados de que saem do *firewall* depois que eles forem roteados.

4.1.3 – Tabela Mangle

Santos (2002), diz que tabela *Mangle* tem a capacidade de marcar ou associar um valor em um pacote de dados, e alterar o campo TOS do mesmo para situações antes do roteamento do pacote. Assim como as outras tabelas ela também possui tabelas, *Prerouting e Output*:

***Prerouting*:** Tem a capacidade de alterar as informações conforme cheguem ao *firewall*, antes do roteamento do pacote de dados.

***Output*:** Esta cadeia dá a possibilidade de alterar os dados criados no local.

4.2 SÍNTESE E LÓGICA

O *Iptables* é um módulo do *kernel* do Linux, sendo assim ele tem que estar sendo processado pelo sistema com intuito que esteja funcionando. A seguir serão mostrados sínteses e comandos para manipulação do *iptables*, de acordo com Neto (2004):

Para listar os módulos ativos no sistema deveremos usar o comando “ismod”, desta maneira:

 # ismod

Caso a ferramenta *Iptables* não esteja ativa deve-se usar o comando “insmod” para ativa-lo, deste modo:

#insmodip_tables Using /lib/modules/2.4.182cl/kernel/net/ipv4/netfilter/ip_table.o

Como o *iptables* está agregado diretamente ao kernel, suas configurações não são feitas através de arquivos de configuração, ela é efetuada por meio de comandos digitados no *shell*.

A partir do momento que aplicamos uma regra ao *shell*, ela valerá apenas para aquela sessão gravada temporariamente na memória, sendo assim, quando o computador *Firewall* é desligado ou reiniciado, as configurações efetuadas são perdidas sem possibilidade de efetuar um resgate das mesmas. Resumindo, quando se efetua uma configuração no *iptables*, ela é salva na memória RAM do sistema, que é volátil, por esta razão, deve-se salvar as regras implementadas ao sistema. Para tal salvamento é necessário utilizar o comando “iptables-save”, como mostrado a seguir:

 # iptables-save > /bin/rc.firewall

Deste modo o comando acima salvará as regras aplicadas no diretório “bin”, no arquivo “rc.firewall”.

É importante ressaltar que há possibilidade de substituir o comando “iptables-save”, por algum tipo de *shell script* que tenha nele inseridas as regras no sistema, ele funciona sendo iniciado em conjunto como o sistema, de modo automático adicionando uma chamada ao fim do arquivo /etc/rc.d/rc.local.

A síntese do *iptables*, é uma síntese simples, intuitiva e lógica:

 # iptables [tabela] [comando] [ação] [alvo]

Deste modo, os comandos de manipulação do *iptables* iniciam-se apontando em qual tabela a regra será aplicada:

Tabela *Filter*:

 # iptables –t filter [comando] [ação] [alvo]

Tabela *Mangle*:

 # iptables –t mangle [comando] [ação] [alvo]

Tabela NAT:

 # iptables –t nat [comando] [ação] [alvo]

É importante ressaltar que a tabela *filter*, é a tabela padrão do *Netfilter*, e por este motivo, caso não seja especificado qual a tabela o comando deverá ser aplicado, o *iptables* entende que o mesmo se aplica a tabela *filter*, exemplo de síntese:

 # iptables [comando] [ação] [alvo]

Sendo assim o comando acima será aplicado na tabela *filter*. O próximo passo são os comandos, o comando "-A" adiciona uma entrada ao final da lista de protocolos:

 # iptables –A [ação] [alvo]

Já o comando "-D" tem o efeito inverso, apagando uma especificada regra:

 # iptables –D [ação] [alvo]

O comando "-L" lista todas as regras contidas na lista:

 # iptables –L [ação] [alvo]

De início, todas as *chains* de uma tabela, estão configuradas no modo *ACCEPT*, isso significa que elas estão configuradas para aceitar todo e qualquer tipo de fluxo de pacotes de dados, para modificar tal configuração, é usado o comando "-P":

 # iptables –P [ação] [alvo]

Existe ainda um comando que remove todas as entradas que foram adicionadas ao *iptables*, sem alterar a política padrão da ferramenta, o "-F":

 # iptables –F [ação] [alvo]

Semelhante ao "-A", o comando "-I" também adiciona uma nova entrada na lista de protocolos, o que os diferenciam é que no caso do "-I" a entrada é acrescentada no início da lista, ao contrário do "-A", onde a regra é adicionada ao final da lista:

 # iptables –I [ação] [alvo]

O comando "-R" é utilizado para substituir uma regra já adicionada por outra:

 # iptables –R [ação a ser substituída] [nova ação] [alvo]

Para se acrescentar uma nova regra a uma tabela específica usa-se o comando "-N":

 # iptables [tabela] –N [ação][alvo]

Ainda temos o comando "-E" usado para renomear uma ação criada pelo usuário:

 # iptables –E [nome anterior] [novo nome]

E por fim o comando “-X”, utilizada para deletar uma *chain* criada pelo administrador do *firewall*:

 # iptables –X [ação] [alvo]

Segundo Neto (2004), as regras já contidas nas tabelas do *Netfilter* já atendem a demanda de segurança em nosso sistema, porém é importante ter a possibilidade de se criar e manipular novas regras para tornar a organização do *firewall* mais simples, assim facilitando a administração do mesmo. No momento em que seu tráfego é maximizado, torna-se necessário seccionar seu fluxo por meio de regras criadas pelo próprio usuário.

O procedimento de sínteses sequente são as sínteses das ações, começando pela ação “-p”, que especifica o protocolo aplicado a regra, tendo a possibilidade de ser qualquer valor numérico determinado no arquivo “*protocol*” do diretório “etc”, ou o próprio nome do protocolo:

 # iptables –p [protocolo]

A ação “-i” determina a interface de entrada que será usada, pois os *firewall’s* contém mais de uma interface, sendo assim esta regra é aplicada para distinguir qual interface de rede o filtro deverá ser aplicado:

 # iptables –i [interface de entrada]

Semelhante a ação “-i”, a ação “-o” determina qual a interface de saída será utilizada:

 # iptables –o [interface de saída]

Para se especificar a origem do pacote de dados a regra tem que ser aplicada, usa-se a ação “-s”:

 # iptables –s [ip e máscara] ou [endereço da rede]

Já para se especificar o destino do pacote utiliza-se a ação “-d”:

 # iptables –d [ip e máscara] ou [endereço de rede]

Quando se deseja acrescentar uma exceção a uma regra, usa-se “!”, juntamente com outras ações:

 # iptables –p ! [protocolo]

Para se definir o alvo do pacote de dados no caso do mesmo se encaixar a uma regra usa-se o “-j”:

 # iptables –j [alvo]

Para se implementar filtros às portas de origem de um pacote de dados usa-se a regra “---sport”:

 # iptables –p [protocolo] ---sport [porta]

Por último, para se determinar qual a porta destino do pacote utiliza-se o “--dport”:

 # iptables –p [protocolo] --dport [porta]

O passo final da síntese dos comandos é o alvo, usado na situação em que um pacote de dados se adequa a uma regra previamente originada, assim devendo ser direcionado para um alvo:

ACCEPT: É usado para permitir determinado fluxo do pacote.

DROP: Utilizado para deletar um pacote de dados destinado a um determinado alvo. Este comando não avisa ao *host* que enviou o pacote o que aconteceu com o mesmo.

REJECT: Possui a mesma função do comando “*drop*”, o que os diferenciam é que este comando informa ao emissor que o pacote foi rejeitado.

LOG: Cria uma entrada de log no arquivo “*messages*”, do diretório “/var/log”, com informações dos demais alvos, por este motivo deve ser utilizado anteriormente aos demais alvos.

RETURN: Este comando retorna à execução da *chain* antecedente sem executar o restante da *chain* atual.

QUEUE: Determina qual *software* em nível de usuário deve administrar a execução do tráfego atribuído ao mesmo.

SNAT: Modifica o endereço de origem dos dispositivos antes do roteamento das informações.

DNAT: Modifica o endereço de destino dos dispositivos.

REDIRECT: Efetua um redirecionamento das portas juntamente com o comando “--to ---port.

TOS: Este comando tem a função de dar prioridade a entrada e saída de dados tomando por base suas funções, funções essas especificadas no cabeçalho do IPv4.

A partir destas regras e comandos é possível tornar um sistema seguro, a seguir serão mostradas tarefas onde, de acordo com Neto (2004), são o básico para proteger uma determinada rede:

- Alterar o alvo padrão das cadeias de regras referente a tabela *filter* para *drop*, o que significa que estamos fechando o filtro de pacotes do sistema operacional, para liberarmos somente o necessário.
- Liberar todo o trafego de entrada da interface de *loopback* do sistema operacional, para que haja comunicação entre os processos.
- Bloquear que *hosts* indesejados consigam enviar pacotes para a rede.
- Bloquear que a rede envie pacotes para *hosts* indesejados.
- Permitir o acesso de pacotes à rede provenientes de *hosts* necessários e confiáveis.
- Configurar as interfaces de rede.
- Configurar as portas de entrada do sistema operacional.

Essas são as configurações básicas para proteger um sistema computacional.

Neste capítulo foram apresentadas todas as maneiras, formas para a construção de regras de *IPTABLES*, para a construção da proteção ao acesso a dados internos, por computadores através da rede. O intuito é mostrar a capacidade e também as dificuldades de lembrar de todos os comandos, funções, opções e argumentos para escrever os scripts.

5 - Desenvolvimento

Neste capítulo é possível visualizar o desenvolvimento da aplicação implementada com o objetivo de criar *scripts* com algumas das regras mais utilizadas na configuração de um *firewall*, a aplicação tem a função de auxiliar seus usuários, a gerar os arquivos de *scripts*, usando uma interface gráfica, facilitando esta tarefa que via de regra é executada utilizando linhas de comandos no sistema operacional Linux.

A aplicação foi desenvolvida na linguagem Java utilizando suas classes responsáveis por edições em arquivos-texto, e a programação será efetuada usando o compilador Netbeans.

5.1 LINGUAGEM JAVA

Segundo Deitel (2009), em 1991 a empresa Sun Microsystems, financiou um projeto de pesquisa interna que teve como resultado uma linguagem de programação criada a partir de outra, o C++, nomeada de Oak pelo seu desenvolvedor, James Gosling, quando mais tarde constatado que este nome já era utilizado em outra linguagem de programação, lhe foi sugerido por uma equipe da Sun, o nome Java, que é uma cidade importadora de café, e o nome foi aceito. Após algumas dificuldades de desenvolvimento no mercado, a linguagem ganhou seu espaço juntamente quando a *web* começou a se desenvolver de forma rápida ganhando grande popularidade, pois o Java oferecia para este ramo, um conteúdo dinâmico, interativo e animado. Nos dias atuais a linguagem é utilizada para o desenvolvimento de aplicativos corporativos de grande porte, aprimoramento em servidores para *web*, aplicativos para plataformas populares como *desktops*, *notebooks* e celulares, entre outros propósitos.

Classes de entrada e saída de dados: Para a implementação da aplicação serão usadas classes da linguagem Java com funcionalidades para processar dados, lendo e gravando estes na memória e em arquivos. Utiliza-se o método *read* para leitura dos dados, e o método *write* para a gravação, estas ações se encontram em classes como a *InputStream* e *OutputStream*, disponibilizadas pela biblioteca *java.io* do Java.

***InputStream*:** Esta classe abstrata disponibiliza a função de simples leitura de dados, utilizando o método *read*. Esta ação retorna à aplicação um valor do tipo inteiro com o número de *bytes* lidos por ela. Declara-se a classe *InputStream* inicializando-se uma classe dependente a ela, a *FileInputStream*, que por sua vez

comporta um argumento do tipo *String*, que identifica à aplicação a localização do arquivo que será lido.

***OutputStream*:** Esta classe abstrata grava dados em sequência em uma localização pré-determinada utilizando a ação *write*, escrevendo no formato de *bytes* no local em que está gravando os dados.

De acordo com Deitel (2009), utiliza-se elementos que intermediaram múltiplas entidades tanto na leitura quanto na gravação de dados. Estes são chamados de *buffers*, que contém uma área de memória usada para guardar temporariamente os dados que foram lidos, mas ainda não foram imprimidos, ou que já foram produzidos, mas ainda não foram gravados. Estes métodos são encontrados na classe *File*, do Java.

5.2 NETBEANS IDE

Segundo a Organização Netbeans (2016), em meados do ano de 1996, na República Tcheca, o Java IDE Netbeans começou como um projeto estudantil nomeado de Xelfi. Por ser o primeiro Java IDE escrito em Java, o projeto atraiu bastante interesse pelos estudantes, que após o termino da graduação, decidiram comercializar como um produto. Após serem contratados por Roman Stanek, os ex-alunos, agora desenvolvedores, continuaram com a ideia inicial de implementar elementos Java ativados para redes. Surge assim o nome Netbeans, auto indicativo das funções da aplicação. Já em 1999, o Netbeans DeveloperX2 foi lançado, suportando o Swing e com melhoramentos no desempenho acompanhados com o JDK 1.3, o Netbeans se tornou uma ferramenta viável para o desenvolvimento de aplicações na linguagem Java. Ainda em 1999 a empresa Sun Microsystems, desejando melhores ferramentas para desenvolvimento em Java, se interessou pelo Netbeans e o adquiriu, e durante esta negociação, seus desenvolvedores demonstraram interesse de fazer do Netbeans um programa de código-fonte aberto. A ideia agradou a empresa Sun, que apesar de já ter contribuído com projetos de código-fonte aberto anteriormente, fez do Netbeans seu primeiro projeto neste modelo, o patrocinando integralmente. Nos dias atuais o Netbeans IDE é uma das ferramentas mais usadas para desenvolvimento de aplicações na linguagem Java.

5.3 APLICAÇÃO

A aplicação consiste em criar *scripts*, contendo as principais regras para a manipulação do *firewall* do Linux, o *Netfilter*. Os comandos do *Iptables* serão gravados em arquivos texto, sem a necessidade de o usuário criar estes conjuntos de regras utilizando as linhas de comando do sistema operacional, mas sim uma

aplicação gráfica, facilitando todo o processo de criação destes *scripts*, diminuindo assim tanto o tempo utilizado neste processo, quanto a ocorrência de potenciais erros que possam ser causados por erro humano. O *software* ainda possibilitará ao usuário a visualização dos arquivos de regras, possibilitando a comparação de até dois *scripts*, e também a adição de regras a um arquivo já existente, se beneficiando da reutilização de um conjunto de regras já utilizado anteriormente.

Será possível utilizar as seguintes regras do *Iptables* na aplicação:

- Alterar a política padrão das cadeias de regras *Input*, *Forward* e *Output* da tabela *Filter*.
- Liberar o fluxo de dados da interface de *loopback*.
- Rejeitar os pacotes oriundos de um determinado *host* para outro.
- Aceitar os pacotes oriundos de um determinado *host* para outro.
- Rejeitar os pacotes que entrem através de determinada interface.
- Aceitar os pacotes de entrem através de determinada interface.
- Abrir determinada porta do sistema operacional.
- Fechar determinada porta do sistema operacional.

5.3.1 – Casos de Uso

- Novo *script*: O usuário clica no botão *New* e a aplicação libera todas as opções para que seja montado seu arquivo de regras.
- Abrir *script*: O usuário clica no botão *Open* e a aplicação abre uma janela onde pode ser encontrado o arquivo desejado.
- Editar *script*: O usuário clica no botão *Edit* e a aplicação libera as opções específicas para que sejam adicionadas regras a um arquivo de regras já existentes.
- Salvar *script*: O usuário clica no botão *Save* e a aplicação salva o arquivo que estava sendo criado ou editado.
- Limpar tela: O usuário clica no botão *Clear* e a aplicação volta a tela ao estado inicial.

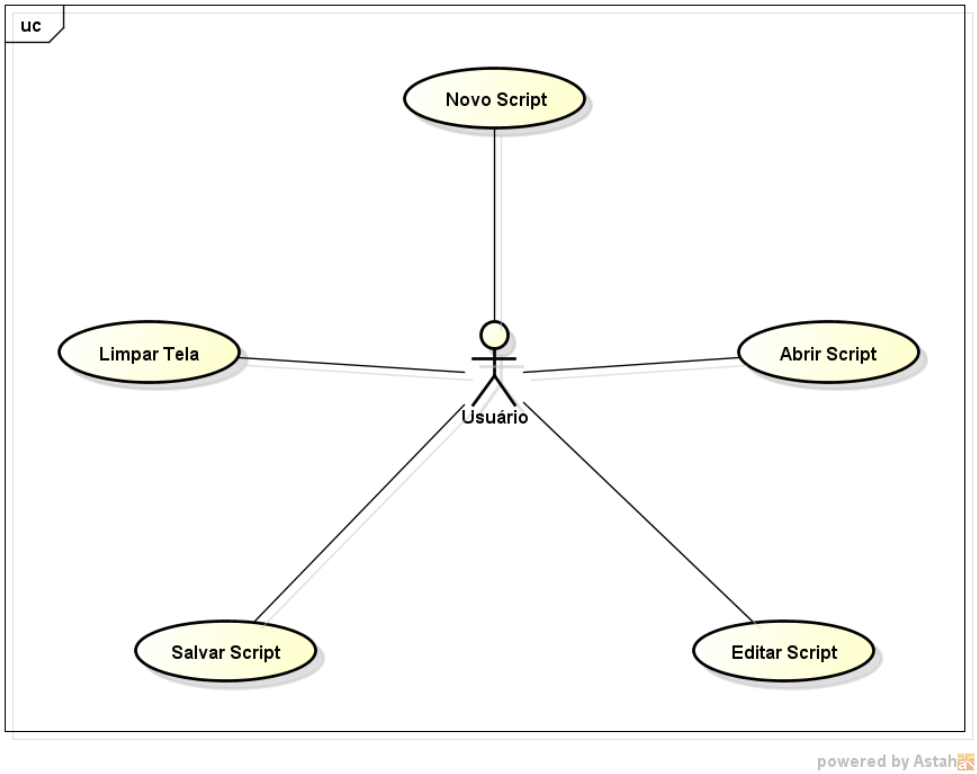

Figura 18 – Diagrama de casos de uso.

5.3.2 – Classes da Aplicação

- ***ScriptsIptables*:** Esta classe conterá um método *main* responsável por configurar e abrir a interface, *jframe*, da aplicação.
- ***EdititingScripts*:** Esta classe conterá o método do tipo *void* chamado *createTextFile*, que receberá dois argumentos do tipo *String*, um com o nome do arquivo, e outro com o corpo do arquivo, e será responsável pela gravação do arquivo de *script*. Conterá mais um método, do tipo *String*, chamado *openTextFile*, que recebe um argumento do tipo *String* com o nome do arquivo, e retorna uma variável também do tipo *String* com o conteúdo do arquivo, este método é responsável por abrir os arquivos de *script*.

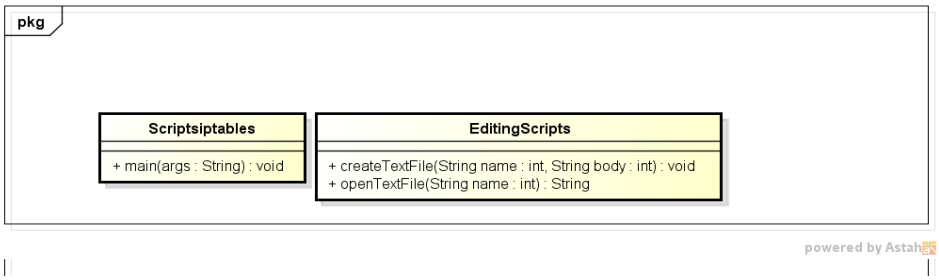

Figura 19 – Diagrama de classes.

5.3.3 – *Layout* da Aplicação

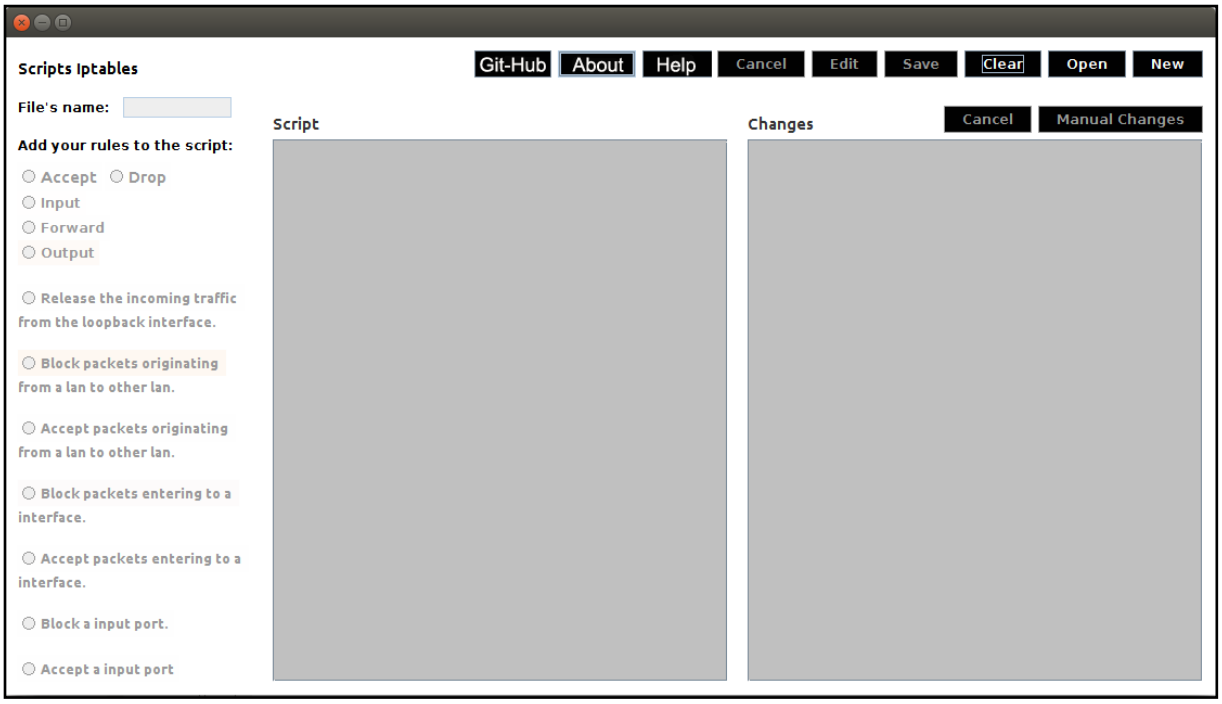

Figura 20 – Tela inicial da aplicação.

A figura acima mostra a tela inicial da aplicação, onde tem-se os seguintes itens:

- Caixa de texto *File's name*: Local que conterá o nome do *script*.
- Botão de Rádio *Accept*: Indicará ao programa que o usuário deseja alterar a política padrão das *chains* da tabela *filter* para *drop*.
- Botão de Rádio *Drop*: Indicará ao programa que o usuário deseja alterar a política padrão das *chains* da tabela *filter* para *accept*.

- Botões de Rádio *Input*, *Forward* e *Output*: Indicará ao programa quais *chains* da tabela *filter* serão alteradas pela configuração acima.
- Botão de Rádio *Release the incoming traffic from the loopback interface*: Indicará ao programa que o usuário deseja liberar totalmente o tráfego na interface de *loopback*.
- Botão de Rádio *Block packets originating from a host to other host*: Abrirá duas caixas de textos, uma por vez, onde o usuário inserirá respectivamente o *ip* da rede que será impossibilitada de enviar dados para o segundo *ip* inserido.
- Botão de Rádio *Accept packets originating from a host to other host*: Abrirá duas caixas de textos, uma por vez, onde o usuário inserirá respectivamente o *ip* da rede que será aceitada a enviar dados para o segundo *ip* inserido.
- Botão de Rádio *Block packets entering to a interface*: Abrirá uma caixa de texto para o usuário inserir uma interface de rede que será impossibilitada de receber pacotes.
- Botão de Rádio *Accept packets entering to a interface*: Abrirá uma caixa de texto para o usuário inserir uma interface de rede que será autorizada a receber pacotes.
- Botão de Rádio *Block a input port*: Abrirá uma caixa de texto para o usuário inserir uma porta do sistema operacional a ser fechada.
- Botão de Rádio *Block a input port*: Abrirá uma caixa de texto para o usuário inserir uma porta do sistema operacional a ser aberta.
- Botão *New*: Permite ao usuário criar um novo *script*.
- Botão *Open*: Permite ao usuário abrir um *script*.
- Botão *Clear*: Volta a tela no seu estado inicial.
- Botão *Save*: Salva o novo *script* ou as edições feitas.
- Botão *Edit*: Abre as edições possíveis para o *script* já aberto.
- Botão *Cancel*: Cancela as edições efetuadas no *script*.
- Botão *Manual Changes*: Libera a área de texto *Changes* para o usuário escrever manualmente regras para o *script* que está sendo criado / alterado.
- Botão *Git-Hub*: Mostra ao usuário o endereço onde é possível acessar o código da aplicação no servidor *Git-Hub.*
- Botão *About*: Mostra ao usuário informações técnicas da aplicação.

- Botão *Help*: Mostra ao usuário tutoriais para orientá-lo na criação de *scripts*. Ao final do trabalho é também apresentado um anexo com um tutorial de ajuda para manipulação da aplicação

- Área de texto *Script*: Mostra ao usuário o *script* que está sendo criado ou editado.

- Área de texto *Changes*: Mostra ao usuário as *edições* que ele está acrescentando ao *script* contido na área de texto *Script*.

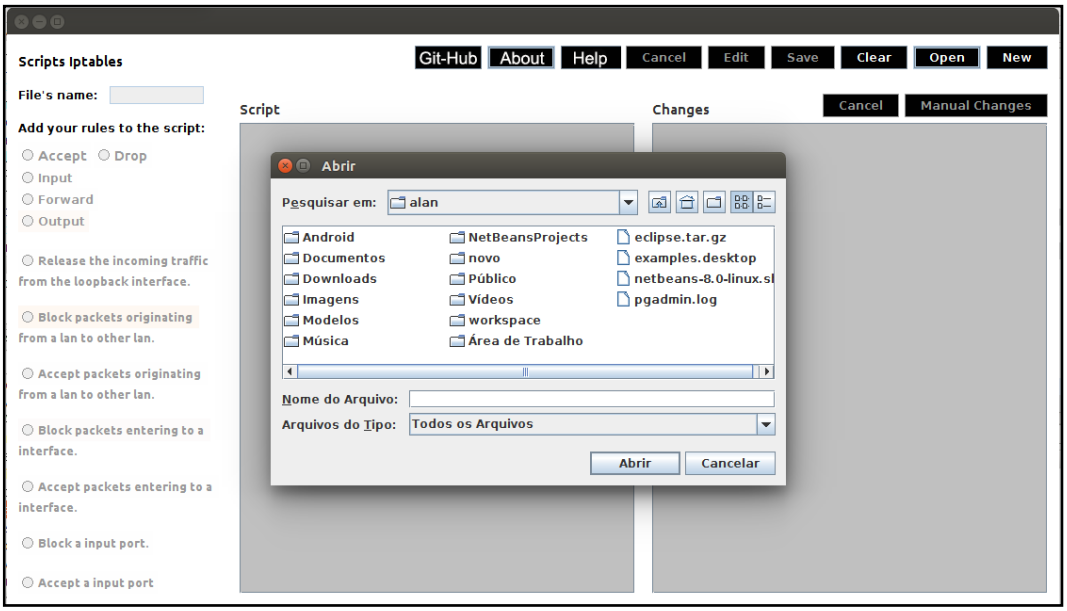

Figura 21 – Tela de abertura de arquivos da aplicação.

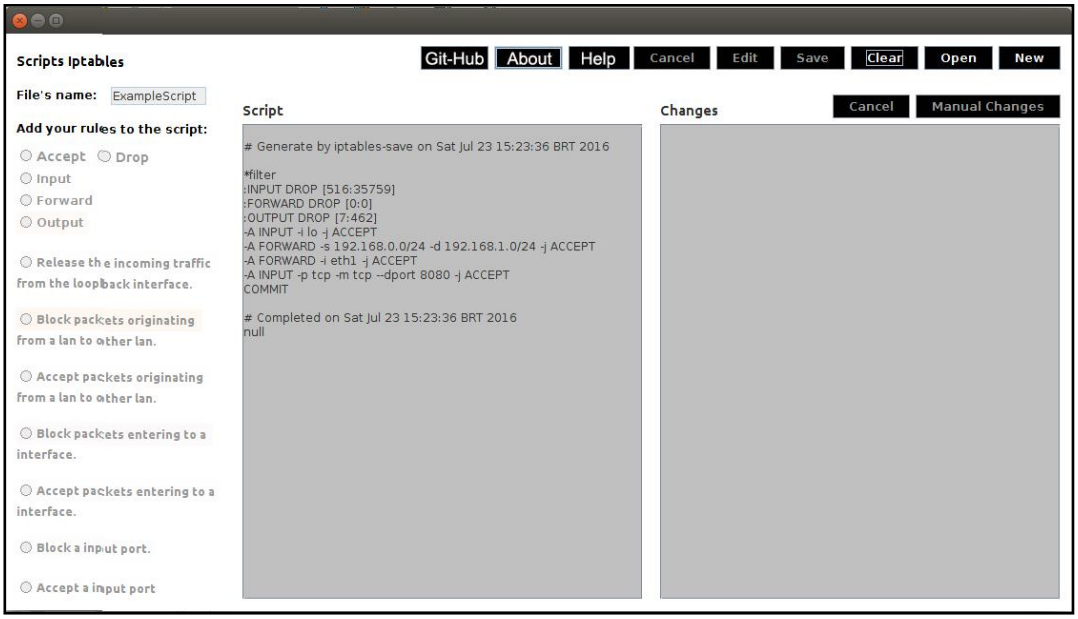

Figura 22 – Tela da aplicação com *script* aberto.

5.4 DESEMPENHO E CONCLUSÕES SOBRE A APLICAÇÃO

A aplicação foi testada criando *scripts* que poderiam ser lidos pelo programa *Iptables* como um *script* tradicional criado por linhas de comando e foi eficiente nesta tarefa, facilitando o processo de criação de *scripts* para o usuário, e também consumindo menos tempo. A seguir dois *scripts*, um criado de modo tradicional e um segundo criado pela aplicação:

Script1 ×

```
# Generated by iptables-save v1.4.21 on Thu Jul 14 13:51:42 2016
*filter
:INPUT DROP [8:1199]
:FORWARD DROP [0:0]
:OUTPUT DROP [289:30004]
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j DROP
-A FORWARD -s 10.0.0.0/8 -d 192.168.0.0/24 -j DROP
-A FORWARD -s 10.0.0.0/8 -d 192.168.1.0/24 -j ACCEPT
-A FORWARD -i eth0 -j REJECT --reject-with icmp-port-unreachable
COMMIT
# Completed on Thu Jul 14 13:51:42 2016
```

Figura 23 – *Script* gerado por linha de comando.

Script2 ×

```
# Name:Script2
# Generate by iptables-save on Thu Jul 14 13:59:20 BRT 2016

*filter
:INPUT DROP [516:35759]
:FORWARD DROP [0:0]
:OUTPUT DROP [7:462]
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j DROP
-A FORWARD -s 10.0.0.0/8 -d 192.168.0.0/24 -j DROP
-A FORWARD -s 10.0.0.0/8 -d 192.168.1.0/24 -j ACCEPT
-A FORWARD -i eth0 -j REJECT --reject-with icmp-port-unreachable
COMMIT

# Completed on Thu Jul 14 13:59:20 BRT 2016
```

Figura 24 – *Script* gerado pela a aplicação.

O objetivo das regras inseridas nestes dois *scripts* é o mesmo:

- Alterar a política padrão das *chains* da tabela *Filter* para o modo *Drop*.
- Liberar o trafego de pacotes pela interface de *loopback*.

- Rejeitar pacotes que entrariam pela porta número 80 do sistema operacional.
- Descartar pacotes oriundos da rede 10.0.0.0 para a rede 192.168.0.0.
- Aceitar pacotes oriundos da rede 10.0.0.0 para a rede 192.168.1.0.
- Rejeitar pacotes que entrariam pela interface de rede eth0.

A seguir o resultado no sistema operacional Linux após a execução de ambos os *scripts*:

Figura 25 – Situação do S.O. após a execução do *script* criado por linha de comando.

Figura 26 – Situação do S.O. após a execução do *script* criado pela aplicação.

Pode-se concluir que além da aplicação ter a capacidade de gerar um *script* praticamente idêntico ao *script* criado por linha de comando, seus resultados no sistema operacional também são iguais. A aplicação, dentro da limitação de regras implementadas a ela, consegue ter os mesmos resultados tanto na criação de *scripts*, quanto na leitura e resultados dos mesmos.

Com o intuito de adicionar a aplicação à comunidade de *software* livre, a aplicação está disponível no servidor *Git – Hub*, onde demais programadores interessados na área de *firewall* do sistema operacional Linux, terão acesso ao código – fonte da

aplicação, tendo a possibilidade de melhorá-lo, e acrescentar novos recursos e funções a ela.

Neste capítulo foi possível verificar algumas partes da construção do software que auxiliará qualquer usuário na construção de *scripts* de *Iptables* para as regras gerais e específicas do seu *firewall*.

CONCLUSÃO

O sistema operacional *Linux*, corretamente configurado, tem plena capacidade de tornar-se um sistema operacional protegido contra invasões. Seu diferencial é com certeza o fato de todos os pacotes de dados que nele adentrarem, passarem obrigatoriamente pelo seu *kernel*, resultando em um total controle sobre eles, este *kernel* é acoplado ao *firewall*, o *Netfilter*, que monitora todos os processos passantes pelo *Linux*, basicamente pode-se dizer que todos os processos do sistema operacional serão monitorados pelo seu *firewall*.

A configuração do *firewall* é dinâmica e adaptável para cada situação em que estas configurações serão usadas, desde um sistema mais rígido onde o administrador da rede necessite bloquear toda sua rede e liberar somente os tráfegos necessários até um sistema com configurações mais brandas, onde o administrador pode deixar sua rede aberta e bloquear somente os tráfegos de maior risco.

De forma geral, a respeito da segurança de computadores, uma vez que num sistema, necessita estar conectado às redes de informática, é praticamente impossível torná-lo blindado, sem nenhuma chance de ser invadido ou atacado. O que os administradores de redes devem fazer em seu sistema, é torná-lo o mais seguro possível, criando o máximo possível de dificuldades e complicações para um potencial *hacker* ou *cracker* que desejem invadi-lo. Quanto maior a dificuldade de se invadir um sistema, mais tempo o atacante mal-intencionado levará para conseguir seu objetivo. Quanto maior o tempo gasto, mais inviável se torna a operação de ataque. Uma vez que a invasão se torna inviável pelo tempo gasto, pode se considerar mínimas as chances de ela acontecer, já que, o objetivo de invasores é obter algum tipo de lucro ou vantagem sobre suas invasões nas obtenções de informações.

A respeito da aplicação desenvolvida, pode concluir que, limitando-se as regras nela implementada, ela tem plena capacidade de substituir a escrita de *scripts* através do modo tradicional, obtendo o mesmo desempenho na realização de seus objetivos, além de facilitar o processo para o usuário, vindo a se tornar uma boa alternativa para administradores de rede.

A aplicação contém somente as principais regras da ferramenta *Iptables*, sendo ainda possível adicionar a ela, em trabalhos futuros, todas as regras possíveis para a manipulação de tabelas.

Ainda é possível adicionar a aplicação funções da ferramenta *Squid*, um programa também usado para adicionar regras em sistemas computacionais como o bloqueio de sites, bloqueio de determinados textos e palavras-chave na internet, entre outras funções, tornando a aplicação mais completa e com mais recursos, visto o comum

uso por administradores de redes da ferramenta *Iptables* juntamente com esta ferramenta.

REFERÊNCIAS

A INFORMÁTICA; **Topologia das Redes.** Disponível em < https://carlos1990.wordpress.com/category/pmmri/> Acesso em: 19 Fevereiro 2016.
BERNARDES, Mauro César; **Segurança Computacional.** Alfenas – MG – Brasil: Universidade de Alfenas (UNIFENAS). 18 p. 1999.

DEITEL, Harvey M.; **Java Como Programar 8ª Edição.** São Paulo, Brasil: Pearson Prentice Hall. 1152p. 2009.

DOCPLAYER; **Criptografia e Segurança em Redes.** Disponível em < http://docplayer.com.br/10451297-Criptografia-e-seguranca-em-rede-capitulo-1-de-william-stallings.html> Acesso em 11 março 2016.

INFO: **Aparelhos com Bluetooth são mais Vulneráveis**. Disponível em: < http://info.abril.com.br/noticias/seguranca/aparelhos-com-bluetooth-sao-mais-vulneraveis-23092011-9.shl >. Acesso em: 26 julho 2015.

JUNIOR, Nilton Lopes de Souza; **Segurança de Redes Utilizando Recursos do Sistema Operacional Linux.** Juiz de Fora – MG – Brasil: Universidade Federal de Juiz de Fora (UFJF). 65 p. 2014.

KURTZ; João; **Techtudo: Brasil é o País com mais Ataques Hackers a Sites do Governo diz Pesquisa**. Disponível em: < http://www.techtudo.com.br/noticias/noticia/2014/11/brasil-e-o-pais-com-mais-ataques-hacker-sites-do-governo-diz-pesquisa.html >. Acesso em: 20 julho 2015.

LHN; **Linux Firewall's Using Iptables.** Disponível em < http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables#.VumexeIrLIU> Acesso em: 10 março 2016.

Machado, André; **O Globo: Prejuízo com Mercado Negro de Hackers pode Atingir US$ 1 Trilhão.** Disponível em:< http://oglobo.globo.com/sociedade/tecnologia/prejuizo-com-mercado-negro-de-hackers-pode-atingir-us-1-trilhao-12428155> Acesso em: 02 Setembro 2015.

MELO, Robson Gomes de; GEUS, Paulo Lício de; **Proposta de um Modelo de Ferramenta de Administração e Segurança Computacional.** Campinas – SP - Brasil: Universidade Estadual de Campinas (UNICAMP). 2 p. 2010.

MORIMOTO, Carlos; **Guia do Hardware: Uma Rápida Explicação do Modelo OSI.** Disponível em:< http://www.hardware.com.br/livros/redes/uma-rapida-explicacao-modelo-osi.html> Acesso em: 09 Fevereiro 2016.

MOTTA, Gustavo; **Introdução a Segurança Computacional.** 36 p. 2006.

NETO, Urubatan; **Dominando Linux Firewall Iptables.** Rio de Janeiro – RJ - Brasil: Editora Ciência Moderna Ltda. 98 p. 2004.

ONLY TUTORIALS; **Iptables: Tabela Mangle.** Disponível em < http://www.onlytutorials.com.br/2008/11/26/iptables-tabela-mangles/> Acesso em: 25 Fevereiro 2016.

ORGANIZAÇÃO NETBEANS; **Um Breve Histórico do NetBeans.** Disponível em <https://netbeans.org/about/history_pt_BR.html> Acesso em: 13 Julho 2016.

SANTOS, Antônio Pedro Freitas Fortuna dos; **Firewall & Linux: Tutorial de Iptables.** Porto, Portugal: Instituto Superior de Engenharia do Porto. 104 p. 2002.

TABORDA, Cauã; **Info: Sistemas Wireless de Carros são Vulneráveis diz Estudo**. Disponível em: <http://info.abril.com.br/noticias/blogs/bitnocarro/seguranca/sistemas-wireless-de-carros-sao-vulneraveis-diz-estudo/ >. Acesso em: 17 Agosto 2015.

TANEMBAUM, Andrew S.; **Computer Networks - Vol: 4.** Amsterdam, Holanda: Vrije Universiteit. 632 p. 2003.

TECINFORMÁTICA; **Curso de Redes.** Disponível em < http://tecinformaticadecomputadores.blogspot.com.br/2015/04/introducao-1_11.html> Acesso em: 10 março 2016.

TELECO; **Tutorial Banda Larga.** Disponível em < http://www.teleco.com.br/tutoriais/tutorialmpls/pagina_1.asp> Acesso em: 21 Fevereiro 2016.

ANEXO I – TUTORIAL DE AJUDA DA APLICAÇÃO

- **Criar *Script*:**

  O usuário inicia a criação do *script* clicando no botão *new*, a aplicação abrirá uma caixa de diálogo onde ele deve inserir o nome desejado para o arquivo. Após a inserção do nome são liberadas as opções para que o usuário selecione as regras que deseja em seu *script*. Montado o arquivo basta clicar no botão *save* para que o *software* o salve na pasta destino.

- **Abrir *Script*:**

  Para abrir um arquivo *script* na aplicação, o usuário deve clicar no botão *open*, a aplicação abrirá um escolhedor de arquivos para que possa ser selecionado um arquivo já existente. Após a escolha a aplicação mostrará o *script* em sua área de texto *Script*.

- **Editar *Script*:**

  Automaticamente: Com um arquivo de *script* já aberto na aplicação, o usuário clica em *edit*, o programa liberará a ele as opções de regras que podem ser adicionadas ao arquivo, toda alteração é mostrada na área de texto *Changes*, ao final das adições o usuário deve clicar em *save*, e seu arquivo estará modificado.

  Manualmente: Com um arquivo de *script* já aberto na aplicação, o usuário clica em *manual changes*, o programa liberará a área de texto *Changes* para que seja digitado manualmente as alterações desejadas, ao final da digitação o usuário deve clicar em *save*, e seu arquivo estará modificado.