Centro Universitário de Brasília

Faculdade de Ciências Exatas e Tecnologia

PROJETO FINAL DE GRADUAÇÃO EM

ENGENHARIA DE COMPUTAÇÃO

Um Estudo sobre Segurança da Informação para Ambientes Conectados à Internet

Brasília, DF – Brasil

Julho de 2004

Centro Universitário de Brasília

Faculdade de Ciências Exatas e Tecnologia

PROJETO FINAL DE GRADUAÇÃO EM
ENGENHARIA DE COMPUTAÇÃO

Um Estudo sobre Segurança da Informação para Ambientes Conectados à Internet

Autor: Leonardo Barbosa de Andrade
R.A: 996608/0

Orientado por: Msc. Luiz Otávio Botelho Lento

Brasília, DF – Brasil
Julho de 2004

Resumo

Este trabalho é um estudo sobre segurança para ambientes conectados à Internet. Este estudo efetua uma análise sobre as principais vulnerabilidades hoje existentes em sistemas conectados à Internet, como essas vulnerabilidades sofrem prospecções e são exploradas causando assim os ataques, apresentando então, elementos fundamentais para a elaboração de um bom plano de segurança como forma de minimizar a exposição destes ambientes na Internet.

Abstract

This work is a study on security for environments hardwired to the Internet. This study realize an analysis on the main existing vulnerabilities today in systems hardwired to the Internet, as these vulnerabilities suffer prospections and are explored causing the attacks, presenting then, basics elements for the elaboration of a good plan of security as form to minimize the exposure of these environments in the Internet.

Sumário

Índice de Figuras

Índice de Tabelas

Índice de Símbolos/Definições

TCP/IP – Trasmission Control Protocol / Internet Protocol

IETF – Internet Engineering Task Force

Buffer overflow – Falha quando o buffer não consegue comportar todos os dados que estão entrando no sistema

Firewall – Sistema de segurança baseado na seleção de informações permitidas

CGI – Common Gateway Interface

Capítulo 1 – Introdução

Quando analisamos o crescimento espetacular da Internet nos últimos anos, percebemos que a necessidade de fornecimento de serviços atraentes se tornaram vitais para as empresas assim como para os clientes que demandam cada vez mais formas ágeis e fáceis de realizar transações e diminuir custos.

Há alguns anos, a informática possuía dois problemas: rapidez no fornecimento de informações e segurança destas informações. A questão da rapidez foi equacionada, mas a segurança e o tratamento confiável destas informações se tornaram pontos críticos, pois é muito difícil para as empresas se protegerem visto que com a Internet, a troca de informações e experiências entre os hackers se tornou muito fácil e as empresas não conseguem acompanhar essa quantidade de informações no mesmo ritmo. A cada nova atualização de antivírus ou novo patch de segurança para um sistema operacional, novas formas de ataque e novas falhas nos sistemas operacionais são encontrados.

Hoje, a Internet começa a agregar valores antes não existentes. Por exemplo, é vital que uma instituição financeira forneça vários serviços antes só encontradas em agências, via um portal Web assim como o comércio eletrônico onde vendas de livros, CD's, equipamentos eletrônicos e todo o tipo de produtos se encontram a disposição na Web. Esse valor agregado é um dos principais fatores de diferencial mercadológico e com enorme potencial de geração de receitas.

Chegamos a um ponto onde as informações geradas e trafegadas pela rede são muito mais valiosas que os equipamentos que as transportam. A informação se tornou um dos ativos mais importantes das empresas que buscam protege-la da melhor maneira possível.

Por isso a importância da segurança. Nos dias atuais, é imprescindível garantir confiabilidade, integridade e disponibilidade dos dados. É necessário melhorar a imagem dos serviços oferecidos pela Internet e entre sites. Para que essa imagem possa ser melhorada é fundamental uma estrutura de segurança consistente e confiável. Porém, essa segurança não pode ser criada instantaneamente. A organização tem que ser consultada. Parâmetros bem elaborados de gerência, análise de requisitos e riscos

precisam ser criados de acordo com o escopo da organização. Políticas sólidas, bem embasadas e pautadas em documentações, regras e padrões existentes são essenciais para que a segurança se incorpore nas rotinas diárias.

Este trabalho apresenta alguns dos principais pontos relevantes ao se tratar de segurança na Internet, assim como as ameaças e vulnerabilidades mais comuns, apresentando conceitos, técnicas, tecnologias e dispositivos que tem o intuito de fortalecer a segurança em seu ambiente.

Como forma de validação de uma parte deste estudo, tratando mais especificamente de dispositivos para aumentar sua segurança em relação as conexões na Internet, foi desenvolvido um simulador de filtragem de pacotes baseado na tecnologia de firewall conhecida como filtro de pacotes. Este simulador foi desenvolvido em linguagem C para ambientes Linux.

Capítulo 2 – A Internet e suas Vulnerabilidades

Criada em 1969 a partir de um acordo de pesquisa firmado entre o governo norte americano e universidades, a ARPANET pode ser considerada a precursora da Internet dos dias de hoje.

Essas pesquisas foram desenvolvidas inicialmente nos Estados Unidos pelo ARPA (Advanced Research Projects Agency), uma agência dentro do DoD (Departament of Defense) durante o período da guerra fria. O interesse do DoD era criar uma rede de comunicação entre diversos sistemas espalhados pelo mundo que funcionassem mesmo se determinados pontos da rede ficassem inativos. Porém a falta de compatibilidade entre os sistemas computacionais de diversos fabricantes que possuíam sistemas operacionais diferentes, topologias e protocolos tornava muito difícil a obtenção de uma solução de integração e compartilhamento de dados.

Algumas universidades norte americanas porém, estavam também desenvolvendo um projeto de protocolo de comunicação com uma nova tecnologia de comutação de pacotes. Unindo os interesses dos dois lados, foi criada a ARPANET em 1969. Inicialmente, ela possuía quatro nós (Universidade da Califórnia em Los Angeles, Stanford Research Institute, Universidade de Santa Bárbara e Universidade de Utah) operando a 50 Kbps.

Em 1971 já haviam 18 nós e no final dos anos 70, esta rede inicial evoluiu e teve seu protocolo principal desenvolvido e transformado na base para o TCP/IP (Transmission Control Protocol / Internet Protocol). A aceitação mundial do conjunto de protocolo TCP/IP deveu-se principalmente a versão (gratuita) do sistema operacional UNIX de Berkeley que além de incluir estes protocolos, colocava-os em domínio público onde qualquer organização poderia modificá-los e assim garantir seu desenvolvimento.

Dentre as várias organizações e comitês que participaram deste desenvolvimento e divulgação, podemos destacar a IETF (Internet Engineering Task Force), que atualmente tem como função principal a manutenção e o apoio aos padrões da Internet, principalmente através da série de documentos denominados RFC (Request for Comments). Estes documentos descrevem as diversas tecnologias

envolvidas e servem de base para as novas tecnologias que deverão manter a compatibilidade com as anteriores quando possível.

Em resumo, o maior trunfo do TCP/IP está no fato destes protocolos aumentarem a interoperabilidade de comunicação entre diversos tipos de hardware e sistemas operacionais, outros pontos importantes como a segurança da operação não foram considerados inicialmente e foram sendo inseridos como “remendos” sempre que possível. [1]

2.1 Evolução

O crescimento da Internet foi assustador nos últimos 10 anos. Em 1984 haviam 1.000 host – servidores de recursos. Já em 1989 haviam 100.000, em 1992 haviam 1.000.000, em 1994 3.200.000 com 20 milhões de usuários aproximadamente. Em 1997 cerca de 10.000.000 de hosts e mais de 30 milhões de usuários. Já no início de 2001, existiam cerca de 110.000.000 de hosts e mais de 400 milhões de usuários [2].

A Internet que no início era utilizada somente para troca de pequenas mensagens de correio eletrônico (o e-mail surgiu em 1971), com o passar do tempo começou a ser utilizada para outros fins graças ao desenvolvimento de novas aplicações, mais interativas e com maiores necessidades. O próprio e-mail tomou uma posição de grande importância, carregando atualmente toda sorte de informações. Hoje já é possível ouvir músicas, assistir vídeos, enviar fotos e até participar de uma videoconferência utilizando a Internet.

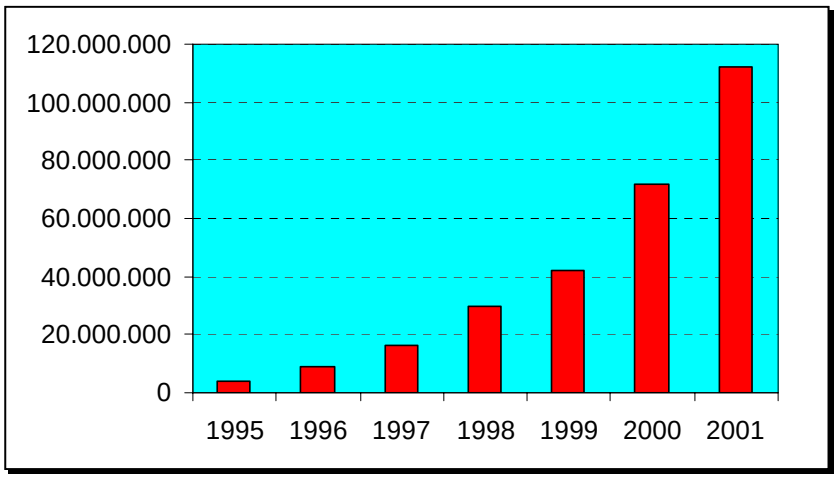


Figura 1 - Evolução do número de hosts no mundo
(Adaptado do Internet Software Consortium [3])

Esta explosão está muito ligada ao uso comercial da Internet, onde negócios da “velha economia” migraram para a rede. Estes negócios que vão desde a simples

compra de um livro, carro ou mesmo apartamento, até complexas aplicações no mercado de títulos/capitais podem ser feitas de qualquer computador conectado à Internet.

2.2 Vulnerabilidades

A Internet começou com um projeto de fins militares. Imaginavam que essa rede de comunicação iria trabalhar em um ambiente não hostil. O seu projeto não tinha como foco principal a segurança das informações trafegadas mas sim uma comunicação eficiente que funcionasse mesmo se determinados pontos da rede ficassem inativos. Por isso, até hoje, a Internet e seus protocolos possuem vulnerabilidades. Exemplos disso são a grande quantidade de pesquisas hoje existentes em cima dos protocolos de comunicação visando sempre melhorar a segurança dos mesmos. Exemplos disso são o Ipv6, Ipsec, SSL, SSH, VPN. Essa falta de preocupação com segurança tanto nos protocolos quanto em aplicações desenvolvidas para funcionamento em ambientes interconectados gera uma enorme quantidade de falhas e vulnerabilidades nos sistemas.

Apesar de inúmeras, algumas poucas vulnerabilidades de software contabilizam a maioria dos ataques bem sucedidos, simplesmente pelo fato dos atacantes serem oportunistas, isto é, escolherem o caminho mais fácil e conveniente. Eles exploram as falhas mais comuns usando as mais efetivas e difundidas ferramentas de ataque. Os atacantes partem do princípio que as organizações não corrigem seus sistemas e saem vasculhando sistemas vulneráveis na Internet [4]. Dentre as vulnerabilidades existentes mais comuns que afetam todos os sistemas podemos citar:

2.2.1 Instalações padrão de sistemas operacionais e aplicativos

A maioria dos softwares, incluindo sistemas operacionais e aplicativos, possuem programas e scripts utilizados para facilitar e tornar mais rápida a instalação destes sistemas, minimizando os esforços por partes dos administradores e garantindo máxima funcionalidade. Para que isso seja feito, estes programas normalmente instalam mais componentes do que a maioria dos usuários realmente necessitam. Isso é

feito pois os fabricantes acham que é melhor habilitar funções que não são necessárias, do que obrigar o usuário a instalar funções adicionais quando for preciso. Este pensamento, apesar de conveniente para o usuário, abre espaço para a ocorrência de muitas das mais críticas vulnerabilidades de segurança pois os usuários não realizam manutenções periódicas nem corrigem ou desabilitam componentes de software não utilizados. Isso acontece pois a maioria dos usuários desconhecem o que realmente está instalado, deixando programas perigosos no sistema, simplesmente porque eles não sabem que estão lá.

Estes serviços vulneráveis fornecem meios para os atacantes invadirem seus sistemas.

Em termos de sistemas operacionais, as instalações padrão incluem uma série de serviços adicionais. Como cada serviço precisa funcionar sobre uma porta, a instalação deixa portas abertas. É através destas portas que os ataques geralmente invadem o sistema. Ou seja, quanto menor a quantidade de portas abertas, menor a probabilidade do sistema ser invadido.

Em se tratando de aplicativos, as instalações padrão geralmente incluem programas de exemplo e/ou scripts que normalmente são desnecessários. Esses scripts e programas de exemplo normalmente não passam por um rígido controle de qualidade e quase sempre são escritos sem nenhuma preocupação com segurança. São através deles que os atacantes invadem os sistemas ou obtêm informações relevantes. Ataques comuns a essa vulnerabilidade são os ataques de buffer overflow devido ao fato de que a verificação de erros nestes programas é freqüentemente esquecida.

2.2.2 Contas sem senhas ou com senhas fracas

Grande parte dos sistemas são configurados de forma que as senhas são a única forma de defesa existente. Isso é um grande problema pois a identidade do usuário (User ID) é razoavelmente fácil de obter, e ainda existem agravantes como por exemplo usuários que utilizam conexões discadas para se esquivar de firewalls. Se um atacante descobrir uma conta de cliente válida (ID e senha) ele terá acesso normal a sua rede.

Além disso existem as senhas de fácil adivinhação como nome de parentes, datas importantes, telefones e as contas sem senhas que podem ser consideradas como o problema mais grave.

Outro grande problema com relação a senhas são as contas de usuários que fazem parte da instalação padrão como “administrador” e “root”. Usuários mal intencionados sempre procuram e tentam atacar primeiro essas contas. Logo, é preciso que elas sejam identificadas e removidas do sistema na medida do possível.

2.2.3 Backups incompletos ou inexistentes

Muitas organizações, geram backups diariamente, porém nunca verificam e testam os dados gravados nestes backups. Na ocorrência de um incidente, é de fundamental importância que os backups estejam completos e íntegros. Muitas organizações descobrem que a sua geração de backups estava com problema justamente quando seus dados são destruídos ou arruinados. Uma política de restauração e de testes de backup são tão importante quanto as políticas de criação.

Outro problema grave é a falta de proteção física das mídias de backup. Os backups possuem a mesma informação sensível que esta armazenadas nos servidores, logo devem ser protegidos da mesma maneira.

2.2.4 Ausência de filtro de pacotes

Vários tipos de ataques exploram funcionalidades dos protocolos ou dispositivos de redes. Por exemplo, um ataque muito comum chamado “smurf” se aproveita de uma funcionalidade dos roteadores para enviar pacotes a milhares de máquinas. Cada pacote possui o endereço de origem forjado de uma vítima e não do atacante. Os computadores recebem estes pacotes e em resposta, inundam à vítima com outros pacotes de resposta chegando a retirar a vitima da rede em alguns casos.

Utilizando um simples filtro de pacotes corretamente configurado pode-se aumentar consideravelmente seu nível de segurança impedindo que pacotes com endereços forjados entrem ou saiam de sua rede.

2.2.5 Sistema de logs incompletos ou inexistentes

Todos os sistemas que estão conectados e possuem um tráfego com a Internet estão correndo o risco de serem atacados. A quantidade de vulnerabilidades descobertas nos sistemas operacionais e protocolos aumenta a cada dia e poucas são as chances de se defender de uma nova vulnerabilidade. A partir do momento em que uma rede tenha sido atacada, a melhor forma de se descobrir o que os atacantes fizeram é através de um sistema de logs. Sem os registros de log, a probabilidade de se descobrir o que foi feito em seu sistema pode ser mínima além de sempre ficar a dúvida se o sistema ainda está sendo controlado pelo atacante.

Os logs servem para fornecer detalhes sobre o que está acontecendo em um sistema. Através dele pode-se perceber se o sistema está sendo atacado ou se já foi invadido. Não se pode detectar um ataque se não se sabe o que está acontecendo na rede.

O registro de log deve ser feito de maneira regular em todos os sistemas críticos e devem ser armazenados e arquivados pois nunca se sabe quando serão necessários. Eles devem ser gravados preferencialmente em um servidor central utilizando uma mídia somente de leitura que não possa ser apagada. Invasores mais cuidadosos costumam entrar em sistemas de log e alterar os registros de forma a apagar dados que possam permitir sua detecção.

2.2.6 Programas CGI vulneráveis

A grande maioria dos servidores, suportam programas CGI (Common Gateway Interface). Além disso, vários servidores web fornecem programas CGI de exemplo. Estes programas servem para proporcionar interatividade em páginas web, permitindo algumas funções como o levantamento e a verificação de dados. Porém, os programas CGI oferecem uma ligação direta com o sistema operacional da máquina onde está o servidor web. As vulnerabilidades nestes programas aparecem, pois eles são programas fáceis de serem localizados, e funcionam com os privilégios do próprio servidor web.

Estes programas são sempre muito atraentes para os atacantes devido a grande quantidade de programas CGI feitos sem preocupação com segurança.

Além destas vulnerabilidades mencionadas, existem outras que também são bastante comuns, porém, específicas de determinados ambientes como Windows e

Unix. Elas não serão apresentadas por não serem foco deste trabalho que pretende apresentar problemas de segurança em um âmbito geral e não problemas específicos de plataforma.

Capítulo 3 – Ataques

Os ataques são simplesmente a exploração das vulnerabilidades existentes em um sistema. Essas vulnerabilidades podem ser de cunho técnico, organizacional, operacional ou até mesmo causada por falta de atenção humana.

O importante de se entender é que os ataques existem e podem acontecer a qualquer momento. A Internet principalmente, abre um grande espaço para invasões e ataques pois ela possui a característica de interligar qualquer máquina ao resto do mundo. O importante é saber quais os principais tipos de ataque, como funcionam e como se proteger.

Um fator agravante para a questão de segurança é o grande número de técnicas que os hackers tem acesso, além da quantidade de ferramentas automatizadas que facilitam o trabalho de descoberta de dados e ataques. Essas técnicas se tornaram muito dinâmicas e funcionam como uma brincadeira de espionagem e contra-espionagem: na máquina é colocada uma proteção, mas logo alguém descobre uma maneira de quebrá-la e novamente o fornecedor aparece com uma nova versão mais robusta que, após algum tempo também é violada. Dois simples exemplos são: o caso dos programas antivírus com versões semanais, em virtude da grande propagação de vírus e os patches que são constantemente criados para os sistemas operacionais como forma de minimizar ou eliminar brechas de segurança encontradas [5].

A automatização e o crescente número de ferramentas que facilitam os ataque a ambientes conectados à Internet aumenta espantosamente a cada dia. Essas ferramentas acabam permitindo que pessoas com o mínimo de conhecimento e experiência em informática possam atacar sites e redes. Porém, a maioria dos ataques bem elaborados normalmente seguem uma linha de ação que pode ser traçada e dividida em duas partes principais que podem ser resumidas como: Prospecção e Ataque.

3.1 Prospecção

A prospecção é a fase de busca de informações sobre fraquezas e vulnerabilidades existentes em um sistema ou em um host específico. Nem sempre as prospecções são feitas de forma indevidas pois elas também podem ser utilizadas por analistas de rede com o intuito de descobrir vulnerabilidades em suas máquinas ou na rede. Porém, essa fase de busca é extremamente importante para o atacante pois fornece informações úteis que irão auxiliar e facilitar sua invasão ao sistema. Essas informações indicam pontos abertos e/ou vulneráveis em um sistema. São a partir destes pontos que os atacantes vão iniciar suas tentativas de penetração. Algumas das técnicas de prospecção [6] mais comuns são mostradas a seguir.

3.1.1 TCP/SYN

Esta é umas das técnicas mais básicas e utilizadas de prospecção devido a sua rapidez no tratamento e resposta das requisições.

Esta técnica se baseia no protocolo TCP e consiste em enviar um pacote com a flag SYN ativada e aguardar a resposta. Se a resposta for um SYN/ACK significa que a porta está aberta, caso contrário ela se encontra fechada.

3.1.2 TCP/SYN (half open)

Esta técnica é chamada de half open pois a conexão TCP não é totalmente aberta. Assim como na prospecção TCP/SYN convencional, um pacote TCP com a flag SYN ativada é enviado como se os procedimentos para uma comunicação normal estivessem se iniciando. De acordo com a resposta da máquina de destino, tira-se conclusões sobre o estado da porta no host de destino:

- Se o host de destino retornar um pacote com a flag SYN/ACK, temos que sua porta encontra-se aberta. Neste caso, a ação tomada deve ser a de enviar um pacote TCP com a flag RST ativada para cancelar essa conexão.
- Se o host de destino retornar um pacote com a flag RST, temos que a porta não está aberta, logo não permite tentativas de conexão.

A vantagem desta técnica é que raros são os sistemas ou máquinas que armazenam esse tipo de condição anômala em seus registros de log, é difícil de se descobrir e comprovar que uma prospecção está sendo ou foi feita.

3.1.3 TCP/FIN (stealth)

Apesar de extremamente eficaz, as prospecções com flags SYN são descobertas muito facilmente por ferramentas de auditoria e/ou segurança como IDSs ou firewalls.

Por outro lado, prospecções que utilizam a flag FIN não são normalmente detectados por estes sistemas. Esta prospecção se baseia no fato de que portas abertas não respondem a esse tipo de pacote enquanto portas fechadas enviam um pacote com a flag RST ativada.

3.1.4 SYN/FIN utilizando fragmentos IP

Este tipo de prospecção é uma variação da técnica de TCP/SYN que também faz uso da opção de fragmentação do IP enviando assim uma série de pequenos pacotes onde as flags SYN e FIN do cabeçalho estão ativas. A questão de se fragmentar o pacote é utilizada como forma de dificultar o trabalho de detecção.

3.1.5 UDP recvfrom()

Apesar de ser um protocolo bem mais simples que o TCP, prospecções com o protocolo UDP são mais difíceis de se realizar devido ao fato do UDP ser um protocolo sem conexão. Logo, as portas independentemente de estarem abertas ou fechadas não enviam nada como retorno a um pedido de conexão.

Desta forma, se utiliza um artifício de se tratar o retorno de mensagens ICMP já que quando uma determinada porta UDP está fechada e uma tentativa de conexão é efetuada, o host de destino envia uma mensagem ICMP do tipo CONNECTION REFUSED para indicar que a conexão foi recusada. Um dos motivos da conexão ter sido recusada pode ser o fato da porta não estar aberta.

3.1.6 Impressão Digital (Fingerprint)

A impressão digital ou fingerprint de uma determinado sistema é composto basicamente de valores característicos de alguns campos do protocolo TCP/IP que cada sistema operacional utiliza ao enviar pacotes pela rede. Campos como Time to Live (TTL ou tempo de vida), flags do cabeçalho IP e tamanhos de janelas são alguns destes campos.

Cada sistema operacional envia ou responde pacotes de uma determinada forma. Através da análise destes campos do protocolo TCP/IP e do desenvolvimento de códigos que classifiquem estas diferenças, os atacantes conseguem obter um nível de refinamento considerável podendo distinguir versões dentro de um mesmo sistema operacional. Algumas das formas mais comuns de se traçar uma impressão digital [6] de sistemas são:

3.1.6.1 Investigação de FIN (FIN probe)

Esta investigação pode ser feita pois alguns sistemas operacionais como Windows, MVT, IRIX e HP/UX não implementam a RFC 793 (Transmission Controle Protocol) de forma correta. A RFC 793 diz que não se deve responder a um pacote com a flag FIN ativada (ou algum pacote sem as flags ACK ou SYN). O correto seria não responder, porém, vários sistemas como os mostrados anteriormente respondem a estes pacotes com um RST.

3.1.6.2 Investigação FALSA (BOGUS flag)

Nesta investigação, é enviado uma flag desconhecida no cabeçalho TCP de um pacote SYN. Alguns sistemas operacionais Linux mais antigos principalmente em versões anteriores a 2.0.35 mantém a flag enviada na resposta deste pacotes. Outros sistemas operacionais porém enviam como resposta um RST.

3.1.6.3 Padrão TCP de ISN (TCP ISN sampling)

Em uma comunicação que utiliza o protocolo TCP/IP, um número de seqüência é requerido quando um pedido de conexão é recebido. Esse número de seqüência inicial, também conhecido como ISN (Initial Sequence Number) é um número “aleatório”, criado e incrementado de diferentes formas em diferentes sistemas

operacionais. Alguns sistemas operacionais como o Windows utilizam uma numeração seqüencial que é dependente da hora, enquanto outros utilizam incrementos aleatórios. Porém, estes números de seqüência acabam seguindo determinados padrões que através de uma análise mais detalhada pode acabar por diferenciar um sistema operacional de outro.

3.1.6.4 Janela deslizante inicial do TCP/IP (TCP Initial Window)

Um outro parâmetro definido no momento do pedido de conexão entre sistemas é o tamanho da janela de dados aceita pelo receptor. O valor de janela inicial aceita varia de sistema para sistema. Assim, pode se eliminar alguns sistemas com mais esta informação.

3.1.6.5 Mensagem ICMP de erro (ICMP Message Quoting)

As RFCs que tratam dos diferentes tipos de mensagem ICMP determinam que na ocorrência de algum erro, o pacote ICMP contenha uma parte de mensagem causadora do erro. Porém, essa parte não possui um tamanho fixo e cada sistema operacional envia uma certa quantidade da mensagem. Analisando qual a quantidade da mensagem existente na mensagem ICMP pode se definir qual sistema operacional gerou o pacote ICMP.

3.1.6.6 Opções do TCP (TCP Options)

O protocolo TCP possui uma grande quantidade de opções não obrigatórias. Por não serem obrigatórias, alguns sistemas optam por não implementar determinadas funções. Desta forma, pode-se enviar pacotes com estas opções ativadas. Sistemas operacionais que implementam estas funções, costumam responder a estes pacotes.

Estas são apenas algumas formas de tentar se descobrir a plataforma que esta sendo utilizada em um determinado sistema através de sua impressão digital. Existem várias outras formas, a grande maioria delas trabalhando com os bits e opções fornecidas pelo cabeçalho TCP analisando o comportamento da pilha de protocolos TCP/IP frente a diferentes opções de pacotes. Uma tabela com algumas impressões digitais conhecidas de diferentes sistemas pode ser encontrada no ANEXO A [7].

3.1.7 Scanners de Portas

Os scanners de portas são basicamente ferramentas de automatização das técnicas de prospecção. Essas ferramentas executam rapidamente, a maioria dos tipos de prospecção existentes em uma grande quantidade de portas. A grande maioria de scanners de portas mais modernos permitem uma enorme quantidade de opções referentes ao tipo de prospecção e portas a serem varridas além de gerar relatórios de resultados extremamente completos, confiáveis e úteis.

3.1.8 Scanners de Vulnerabilidades

Os scanners de vulnerabilidades são a melhoria dos scanners de portas. Eles realizam as mesmas funções básicas de varredura de portas. Porém, os scanners de vulnerabilidades possuem a capacidade de aplicar explorações já conhecidas nas portas que estão abertas. Sendo assim, os relatórios gerados são muito mais completos dizendo além de quais portas estão abertas, quais são as explorações às quais estas portas estão vulneráveis.

3.1.9 Sniffers

Um outro método de se obter informações de um sistema ou rede é através da utilização de sniffers. Os sniffers (do inglês farejador) é um dispositivo inserido em alguma máquina, programa ou diretamente na rede. Este dispositivo é configurado para trabalhar em um modo promíscuo (capturando todos os pacotes da rede) desta forma conseguindo visualizar tráfegos que não seriam destinados a ele. Em uma rede onde não existe criptografia ou qualquer outro método de tratamento seguro de dados durante a transmissão, um sniffer consegue capturar e entender praticamente tudo que ele captura. Em redes muito grandes, geralmente os intrusos utilizam filtros para que eles possam capturar apenas pacotes que eles julgam interessantes.

As principais formas de se evitar este tipo de ataque é configurando transmissões seguras e nunca transmitir dados (principalmente críticos) em texto claro.

3.1.10 Engenharia Social

A engenharia social é a forma de obter informações sobre um determinado sistema ou organização através de pessoas. Esse tipo de prospecção se aproveita da característica do ser humano que normalmente tenta ajudar e ser prestativo além de sua curiosidade e ingenuidade. Um ataque de engenharia social pode acontecer das mais diversas formas possíveis sejam elas direta ou indiretamente. Alguns tipos de ataque de engenharia social comuns incluem [8]:

- Anexos de e-mail: Muitas vezes, as pessoas recebem anexos em e-mails com origem desconhecida ou até mesmo conhecida, porém, não executam antivírus sobre esses anexos. Usuários mal intencionados enviam arquivos com vírus, trojans ou backdoors com títulos sugestivos como “I love you” ou “Ganhe dinheiro rapidamente” se aproveitando assim da ingenuidade e curiosidade da vítima.
- Salas de bate-papo: Fraudadores muitas vezes ficam em salas de bato papo para se aproveitarem do lado emocional das pessoas. Se dizem pessoas bonitas, atraentes e com bom papo para obterem informações e convencê-las a baixar ou executar arquivos maliciosos sem que ela saiba.
- Solicitação de Informação: Atacantes se fazem passar por outra pessoa por e-mail, telefone ou até mesmo pessoalmente. Eles se dizem administradores do sistema ou até mesmo usuários de um determinado serviço e precisam de contas, senhas ou qualquer outra informação relevante. Se aproveitam que funcionários de suporte ou helpdesk sempre tendem a ser prestativos e tentam enganá-los obtendo informações úteis.

3.2 Formas de Ataque

Após obter as informações necessárias sobre os sistemas, organizações ou até mesmo pessoas, os atacantes irão utilizar suas técnicas e ferramentas para invadir, capturar informações úteis até mesmo indisponibilizar o fornecimento de serviços da organização. Ou seja, o ataque ocorre no sentido real da palavra. Alguns dos tipos de ataque mais comuns e suas características são mostrados neste tópico.

3.2.1 Vírus

Os vírus de computador são programa pequenos, que por definição não funcionam sozinho. Na maioria das vezes são ou estão presentes em arquivos executáveis esperando sua execução para que possa realizar suas ações maliciosas. Os vírus na maioria das vezes, após executados, conseguem se espalhar e infectar outros arquivos.

O ataque por vírus explora o descuido e desatenção dos usuários que não costumam checar os arquivos que recebem. Com a explosão da Internet, estes vírus agora se espalham muito facilmente pela Internet principalmente por e-mails.

Os vírus são agrupados de acordo com características especiais existentes em cada um. Porém um vírus pode possuir uma ou mais destas características. Os tipos de vírus mais comuns são:

3.2.1.1 Vírus de boot

Este tipo de vírus infecta o registro mestre do sistema também chamado de MBR – Master Boot Record dos discos rígidos ou área de Boot (Boot Sector) dos disquetes. Como estas áreas são executadas antes de qualquer outro programa, esses vírus são muito bem sucedidos.

A contaminação por este tipo de vírus ocorre quando um boot é feito a partir de um disquete contaminado. O setor de boot de um disquete possui a informação necessária para informar se o disquete é ou não “bootável”. É este código no setor de boot que ao ser contaminado assume o controle do micro. Assim que o vírus é executado ele entra na memória do micro e infecta a MBR.

Uma forma de se evitar este tipo de vírus é nunca realizar boot por disquetes desconhecidos. Sempre execute antivírus em disquetes de boot antes de utilizá-los.

3.2.1.2 Vírus de programa

Os vírus de programa infectam programas executáveis. Alguns deles se replicam e vão infectando outros arquivos de forma silenciosa e sem atrapalhar a execução dos programas que já estão contaminados. Essa infecção pode ser via Internet, rede local ou até mesmo disquetes. Alguns agem imediatamente após a

infecção enquanto outros ficam se replicando até determinada data ou alguma outra condição de parada e somente aí começa sua ação.

3.2.1.3 Vírus multipartite

Os vírus multipartite são uma mistura dos vírus de boot e vírus de programa. São muito eficazes na tarefa de se multiplicar e espalhar e são muito difíceis de serem detectados e removidos.

3.2.1.4 Outras características

Além de pertencerem a algum dos grupos acima, os vírus podem possuir algumas características especiais. Algumas das principais delas são:

a) Polimorfismo: O vírus possui uma constante mutação, ou seja, muda sua forma a cada arquivo infectado. Porém, suas cópias são tão funcionais quanto a original. Essa mutação visa dificultar a ação dos antivírus.
b) Encriptação: O código do vírus fica encriptado tornando muito difícil a ação dos antivírus.
c) Invisibilidade: O vírus possui a capacidade de retirar seu código da memória do computador temporariamente tentando assim escapar da ação dos antivírus.

3.2.1.5 Vírus de Macro

Um vírus de macro é um vírus que se aproveita da capacidade que certos programa tem de executar macros. Uma macro é uma série de funções personalizadas que são executadas automaticamente pelo aplicativo. Assim sendo, quando uma pessoa mal intencionada insere algum comando malicioso dentro de uma macro e envia esse arquivo para outra pessoa, quando esse arquivo é aberto, o aplicativo começa a executar os comando existentes na macro, logo, executa o comando malicioso que infecta seu computador.

Esse vírus têm se proliferado de maneira espetacular na Internet devido aos e-mails e falta de atenção dos usuários ao abrirem arquivos.

3.2.1.6 Retrovírus

Conhecido também como vírus-antivírus, o retrovírus ataca diretamente o antivírus afim de desativá-lo, indo nos arquivos de definição de vírus do software.

3.2.2 Worms

Os worms foram desenvolvidos nos anos 70 e eram utilizados como mecanismos legítimos para o gerenciamento e execução de tarefas em sistemas de recursos distribuídos. Porém, devido a sua capacidade de autoduplicação e pelo fato de serem entidades autônomas (não precisam estar anexados a um programa ou qualquer outro arquivo hospedeiro), os worms acabaram se tornando a grande praga da Internet.

As medidas preventivas com relação aos worms são exclusivamente cuidado antes da abertura de arquivos de qualquer gênero, principalmente de origem desconhecida, não devendo ser abertos sem antes ser executado um antivírus sobre o arquivo.

3.2.3 Cavalo de Tróia

Um cavalo de tróia (trojans ou trojan horse) é um programa malicioso que fica escondido dentro de algum programa interessante que incentive o usuário a executá-lo (um jogo por exemplo). Ao ser executado, o programa insere o cavalo de tróia no seu computador que começa a executar suas ações maliciosas. Muitas vezes, estes cavalos de tróia criam backdoor (literalmente porta dos fundos) abrindo alguma porta TCP ou UDP (por exemplo) instalando um serviço que ficará sempre em estado de escuta (listening) esperando receber algum comando com instruções do que deve ser feito. Apesar de freqüentemente ser confundido com vírus, os cavalos de tróia não possuem capacidade de replicação.

3.2.4 Quebra de Senha

Como o próprio nome já diz, os programas de quebra de senha tem por objetivo descobrir senha de acesso a sistemas e aplicativo. Esses programas são nada mais, nada menos que programas de força bruta que ficam tentando incansavelmente

testando senhas geralmente guardadas dentro de um dicionário possuído pelo programa.

Uma maneira fácil de impedir que o programa de quebra de senha funcione é limitando o número de tentativas erradas de uma determinada senha antes de bloqueá-la. Porém esse método algumas vezes acaba se tornando um método de negação de serviço pois um usuário mal intencionado pode tentar acessar várias contas com senhas erradas por diversas vezes até conseguir bloquear a senha e por conseguinte o acesso legítimo ao aplicativo ou sistema.

3.2.5 Negação de Serviço (Denial of Service – DoS)

Um ataque de negação de serviço (DoS – Denial of Service) é caracterizado por uma intenção explícita de um determinado atacante de impedir que usuários possam acessar serviços legítimos e autorizados. Exemplos:

a) Tentativas de impedir que um determinado usuário acesse um serviço;
b) Inundações de dados através da rede, impedindo assim que tráfego autorizado trafegue;
c) Tentativas de desconectar máquinas ou dispositivos impedindo assim o acesso a recursos e serviços.

Nem todas as quedas no fornecimento de serviços são necessariamente ataques de negação de serviços a não ser que a queda neste fornecimento tenha acontecido de forma intencional. Vários outros tipos de ataque incluem a negação de serviço como componente do ataque.

O impacto gerado por um ataque desta forma pode ser desastroso. Ataques de negação de serviço podem derrubar o seu computador de uma rede e, dependendo da natureza da organização podem derrubar toda a estrutura de comunicação de uma organização.

Existem várias formas de ataques de negação de serviço. O CERT Coordination Center [9] tenta dividir todas estas formas entre três principais vertentes:

a) Excessos que geram escassez de Recursos
b) Destruição ou Alteração de Informações de Configuração
c) Destruição Física ou Alteração de Componentes da Rede

3.2.5.1 Excessos que geram escassez de Recursos

Para um bom funcionamento, as redes de computadores precisam de alguns dispositivos para funcionar como: memória, largura de banda, tempo de CPU além de diversos outros fatores. Freqüentemente, ataques de negação de serviço tentam gerar algum excesso em algum destes componentes comprometendo assim seu funcionamento através da escassez destes recursos como por exemplo:

a) Conectividade da rede: muitas vezes, ataques de negação de serviço são executados visando o ataque a conectividade da rede. Um exemplo deste tipo de ataque é a inundação de sincronização (SYN flood). Neste ataque, o ataque inicia um processo de sincronização com a vítima mas faz isso de uma maneira onde essa comunicação nunca se complete. Quando isso acontece, a vítima reserva uma estrutura de dados para completar essa conexão. Porém, este número de estruturas é limitado e o atacante começa a abrir um grande número de pedidos de sincronização. Assim, a vítima vai aumentando seu número de conexões abertas porém não completadas até a vítima não ter mais capacidade de aceitar nenhuma conexão.

   Este tipo de ataque não consome nem depende da largura de banda disponível. Neste caso, o que é consumido são estruturas de dados do núcleo do sistema operacional (kernel) envolvidas na tentativa de estabilização de conexões.

b) Utilizando seus próprios recursos contra você: Um intruso pode utilizar seus próprios recursos contra você das mais diferentes e inesperadas formas: Um exemplo disso é a inundação de UDP onde um intruso simula pacotes UDP para se conectar com o serviço de eco (echo service) para que uma máquina fique trocando mensagens de eco com outra. O resultado disso é que tanto as máquinas nos dois extremos, quando a largura de banda entre elas ficam totalmente consumidas. Ou seja, além das máquinas, a largura de banda da rede também é afetada.

   Este ataque precisa utilizar IP spoofing (que será aplicado mais adiante) para simular um endereço de origem falso.

c) Consumo de largura de banda: Um intruso pode estar apto a consumir toda a largura de banda disponível em sua rede gerando uma enorme quantidade de pacotes diretamente na rede. Geralmente estes pacotes são ICMP ECHO

mas o princípio pode ser utilizado com outros tipos. Entretanto normalmente o usuário não utiliza sua máquina para realizar este ataque, ao invés disso ele coordena uma enorme quantidade de máquinas em outra rede para realizar este serviço.

d) Consumo de outros recursos: O princípio utilizado no consumo de largura de banda pode ser utilizado de diversas outras formas consumindo não só recursos da rede como recursos da própria máquina como estrutura de dados do sistema operacional ou mesmo quantidade de processos que podem ser abertos. Estas estruturas são limitadas e podem impactar o funcionamento e disponibilidade de serviços caso sejam inundados.

3.2.5.2 Destruição ou Alteração de Informações de Configuração

Uma configuração incorreta de seus dispositivos ou componentes de redes pode fazer com que sua rede funcione de maneira incorreta ou simplesmente não funcione. Um intruso pode alterar ou destruir um arquivo de configuração que irá impedir outras pessoas de acessarem a rede.

Por exemplo, se um intruso altera a configuração de rotas de um roteador, pacotes poderão não chegar a seu destino.

3.2.5.3 Destruição Física ou Alteração de Componentes da Rede

Este tipo de ataque se referência unicamente a segurança física de sua rede. Para evitar incidentes indesejados, é indispensável que somente pessoas autorizadas tenham acesso aos computadores, switches, roteadores, backbones e demais componentes de sua rede.

Os ataques de Negação de Serviço têm se tornado mais elaborados a cada dia. Hoje, a maioria destes ataques são feitos de forma distribuída (DDoS – Distributed Denial of Service), aumentando muito sua eficácia. Um ataque de negação de serviços distribuídas funciona da mesma forma que um ataque de negação de serviços convencional, porém, utilizando as facilidades e recursos da computação distribuída.

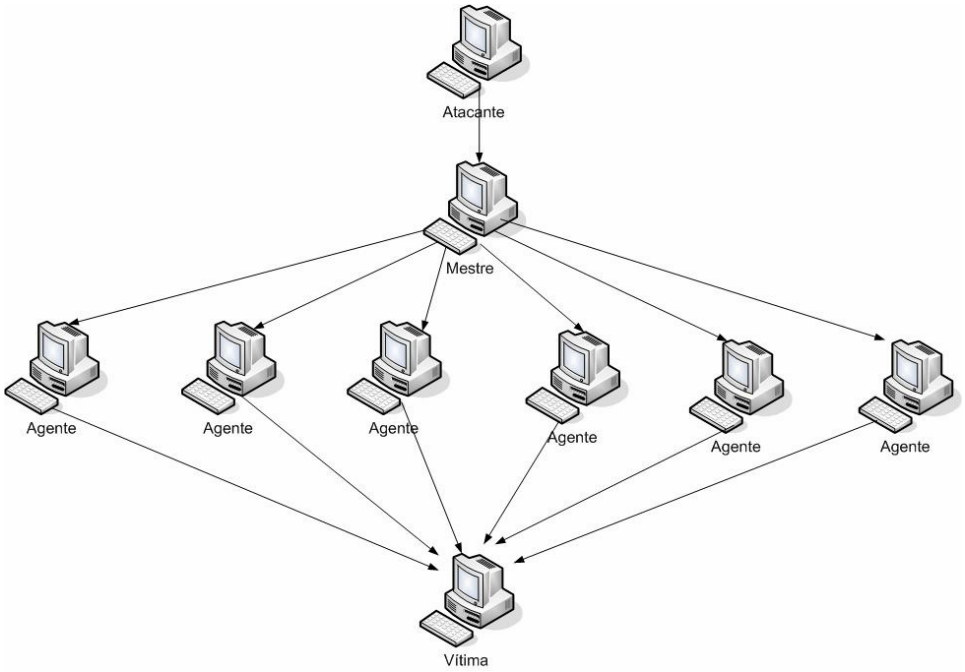


Figura 2 - Exemplo de ataque de Negação de Serviço

3.2.6 Bomba de e-mail (mail bomb)

A bomba de e-mail (mail bomb) é uma forma de ataque de negação de serviço específica onde uma enorme quantidade de mensagens são enviadas para o e-mail de um usuário específico travando assim sua caixa postal e impossibilitando o acesso ao serviço e impedindo que outras pessoas possam enviar qualquer mensagem para aquela caixa postal.

Antigamente, a bomba de e-mail era utilizada para punir pessoas que haviam violado a Netiquette (regras de etiquetas da Internet – por exemplo enviado spams).

As bombas de e-mail são um grande inconveniente tanto para a pessoa atingida pelo ataque como para as outras pessoas que utilizam aquele servidor atingido, pois muitas vezes, quando o serviço não possui um limite máximo por usuário, um ataque de bomba de e-mail pode inundar um servidor inteiro interrompendo o serviço para vários usuários.

3.2.7 Spoofing

O ataque de spoofing se baseia basicamente em técnicas de falsificação. Essas técnicas podem ser para falsificação de tabelas ARP, endereços IP, DNS e diversas

outras. Na falsificação de endereço IP por exemplo, o atacante falsifica o endereço do remetente de um determinado pacote fazendo com que o receptor trate o atacante como se fosse outra pessoa. Este ataque explora o fato de que muitas comunicações na Internet se baseiam em uma relação de confiança. Por exemplo, podemos ter um computador X que pode se comunicar com um computador Y sem que haja uma constante verificação de autenticidade na comunicação. O intruso então se camufla dizendo para o computador X que ele é o computador Y. Assim sendo, o computador X irá permitir que o computador intruso faça o que quiser como se fosse o computador Y. Porém, é necessário que o intruso derrube o computador Y pois o computador X continuará respondendo para o computador Y. Logo Y poderia cancelar a conexão pois algo está errado; ele está recebendo respostas sem que tenha feito perguntas.

Além disso, conexões possuem números de seqüência que são gerados aleatoriamente, logo esse ataque exige muita inteligência e criatividade. Os tipos de ataques de falsificação mais comuns são o spoofing de IP, ARP e DNS.

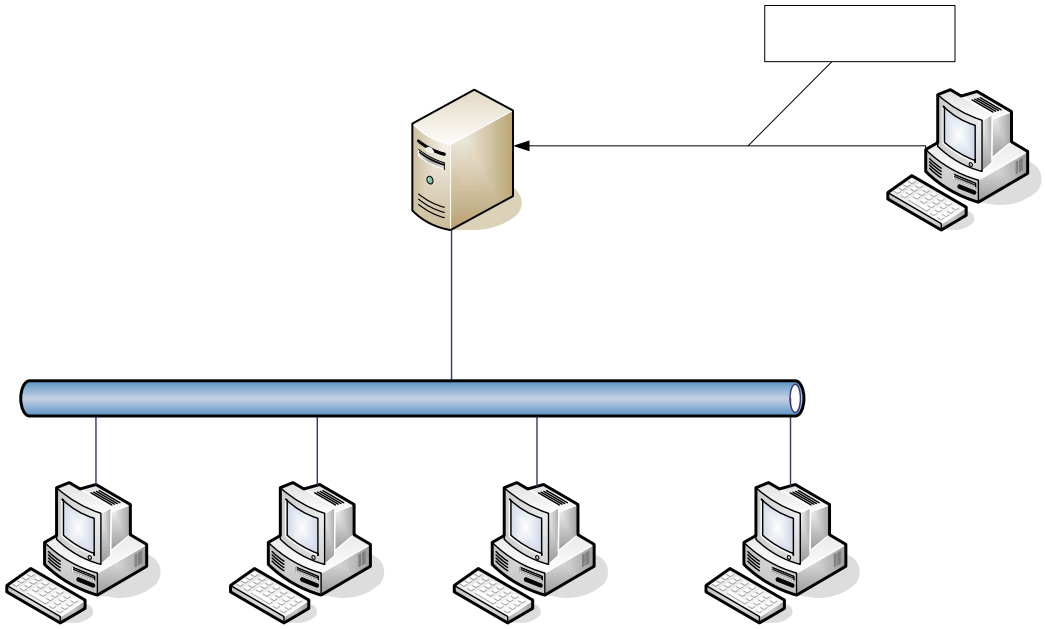


Figura 3 - Ataque de Spoofing

3.2.8 Man in The Middle

Este ataque também é conhecido como seqüestro de sessão. Neste tipo de ataque, o atacante se insere entre uma comunicação, lê os pacotes, realiza as alterações desejadas e reenvia o pacote novamente para a rede. Apesar da idéia ser simples, este ataque é extremamente complexo pois envolve conceitos de spoofing, descoberta de números seqüências e confiança entre hosts. Normalmente, o atacante perde algum tempo realizando prospecções para descobrir tendências de relacionamento entre vítima e alvo.

Além de complexo, este ataque é extremamente perigoso principalmente quando o atacante se insere entre conexões sem criptografia, pois a partir do momento que o atacante é bem sucedido na captura da sessão, é difícil para o alvo ou para a vítima descobrirem que a sessão foi capturada pois o tráfego parece ser completamente normal e íntegro.

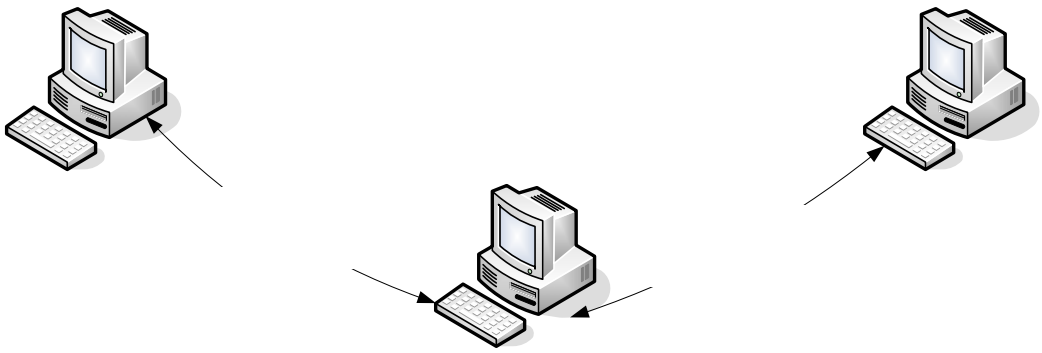


Figura 4 - Ataque "MAN IN THE MIDDLE"

Capítulo 4 – Fundamentos de Defesa do Site

A melhor forma de proteger um ambiente conectado à Internet é com um bom projeto de defesa. Este projeto deve se adequar o máximo possível ao modelo da organização e tentar atender todas as necessidades de segurança da mesma.

Não existe um projeto de defesa que seja bom para todas as situações. Diferentes empresas possuem diferentes necessidades. Não é a organização que precisa se adequar ao plano de segurança, é o plano de segurança que deve ser adequado à empresa.

Um bom plano de defesa deve ser completo e tentar abranger ao máximo todos os pontos da empresa. Deve-se fazer uma análise minuciosa de qual o objetivo a ser atingido e tentar envolver todas as pessoas da organização independente de hierarquia.

A informação hoje é um ativo tão importante como qualquer outro no negócio, possuindo um valor para as organizações e necessitando ser protegido. A segurança da informação [10] é caracterizada basicamente pela preservação da:

- Confidencialidade – Garantir que somente as pessoas autorizadas irão ter acesso à informação
- Integridade – Proteger a exatidão e a inteireza da informação e métodos de processamento
- Disponibilidade – Garantir que os usuários autorizados terão acesso à informação e aos métodos de processamento

Essa segurança da informação e dos processos relacionados a ela são fundamentais, principalmente, em ambientes conectados à Internet pois os mesmos são uma fonte interminável de ameaça à segurança das organizações. Todas essas ameaças à segurança da informação podem acabar comprometendo seriamente a competitividade, a imagem comercial além de causar prejuízos financeiros as organizações.

Este capítulo tenta apresentar alguns pontos fundamentais que devem ser considerados na elaboração de um plano de defesa, sempre lembrando que estes pontos são apenas um guia de opções relevantes. O plano de defesa de uma organização deve ser feito com base nas necessidades particulares da organização.

4.1 Análise de Riscos

A análise de riscos consiste basicamente em tentar determinar o que é necessário proteger e de quem se proteger. Nesta etapa da elaboração de um plano de segurança é necessário fazer uma lista de todos os pontos que oferecem risco para o negócio, separando estes pontos de acordo com o nível de risco associado a cada um deles.

Durante essa análise devem ser identificados quais são os recursos que devem ser protegidos e quais as ameaças existentes a esses recursos.

Na identificação dos recursos a serem protegidos, deve-se considerar recursos como: softwares, hardwares, informações, documentações e pessoas.

Após identificados os recursos, é necessário apontar ameaças as quais esses recursos estão expostos. Essas ameaças também devem ser identificadas de acordo com a sua potencialidade de perda e destruição. Algumas das ameaças mais comuns foram apresentadas no capítulo 3.

4.2 Política de Segurança

A política de segurança é a forma de se garantir um bom gerenciamento e suporte da segurança da informação. Ele é o documento base para todas as questões de segurança e deverá conter todos os pontos considerados relevantes para a segurança da informação de uma organização.

A elaboração deste documento deve envolver todos os setores da empresa e ter sempre a aprovação e concordância da diretoria da empresa.

É neste documento que estarão documentadas todos os pontos críticos da empresa, como devem funcionar, implicações legais na eventualidade de desastres e no uso diário dos sistemas de informações, políticas de testes, delegação de responsabilidades e todos os demais tópicos existentes neste capítulo.

4.2.1 Política de Permissão

As duas principais e mais conhecidas políticas de permissão existentes são:

- Permitir tudo (Allow All)

- Negar tudo (Deny All)

A política de Permitir tudo parte do princípio de se permitir acesso a tudo e depois analisar o que não deve ser permitido e aplicar as negações no que for necessário.

A política de Negar tudo supõe que o usuário deve possuir o mínimo de acessos necessários. Inicialmente nega-se todos os acessos para depois analisar o que é realmente necessário para o trabalho diário para então aplicar os acessos devidos.

A política de permitir tudo é extremamente fácil de implementar porém, possui sérias implicações e furos em termos de segurança. A política de negar tudo é mais sólida e segura sempre imaginando que quanto menos acessos forem permitidos, menores as chances de falhas na segurança. Os usuários geralmente reclamam quando algo que deveria ser fornecido não é, porém, nunca abrem a boca quando possuem acessos que não deveriam.

É importante lembrar que todas as políticas devem tentar ao máximo se adequar ao escopo da organização. Deve ser segura, porém, a sua flexibilidade é fundamental.

4.2.2 Política de Acesso

A política de acesso tem a função de gerenciar as questões de acesso on-site, off-site além de acessos de serviços terceirizados e acordos entre organizações que irão permitir acesso de outras pessoas ao sistema de informação.

4.2.2.1 Físico

A política de acesso físico deve prover uma estrutura segura garantindo que o acesso físico aos locais de trabalho sejam coordenados de forma que pessoas não autorizadas sejam mantidas fora de seu ambiente.

A política deve estar apta a coordenar acesso de pessoas não só de funcionários efetivos, como de colaboradores, terceirizados e pessoas que irão permanecer na organização apenas por curtos períodos de tempo como empresas de suporte a hardware.

4.2.2.2 Lógico

Assim como o acesso físico, o acesso lógico em uma organização deve estar preparada para garantir que as pessoas tenham acesso lógico ao seu sistema somente quando necessário e pelo tempo mínimo necessário.

Os acessos lógicos devem ser criados de acordo com o tipo de responsabilidades e requisitos de acesso, tendo um tempo de validade pré-determinado que poderá ser prorrogado em caso de necessidade.

Todos as políticas de acesso devem funcionar de maneira integrada.

4.2.3 Política de Antivírus

A política de antivírus deve ser criada como forma de minimizar a exposição dos sistemas conectados à Internet com relação a vírus, worms e alguns outros softwares maliciosos.

Essa política deve vir incorporada de uma melhoria na cultura de segurança dos usuários. Juntamente com a política de antivírus, deve constar um manual de boas práticas de segurança em relação ao tratamento de informações contra vírus, worms e outros códigos maliciosos. O objetivo deste manual é educar os usuários a sempre executarem o antivírus em arquivos recebidos de origem desconhecida, quando desconfiarem do conteúdo de arquivos, quando receberem arquivos por vias inseguras (Internet por exemplo) e periodicamente como medida de prevenção.

4.2.4 Política de Senhas

A política de senhas tem o objetivo de validar as identidades dos usuários no acesso aos sistemas ou serviços de informação.

A política de senhas deve ser gerenciada de forma rígida obrigando os mesmos a renovar a suas senhas periodicamente. Essa política deve possuir controles que proíbam os usuários de:

- Utilizar senhas com menos de 6 dígitos
- Construir senhas somente com números ou somente com letras

- Construir senhas de fácil dedução utilizando datas importantes, aniversários ou nomes de familiares ou com vários números ou letras iguais ou consecutivos.
- Repetir as últimas 6 senhas

Estes são apenas alguns dos controles que devem ser aplicados. Os usuário precisam estar cientes das implicações legais no caso de compartilhamento de senhas, ou quando deixarem senhas em lugares de fácil acesso, ou quando colocarem suas senhas em scripts de execução automática.

Processos biométricos e outros tipos de dispositivos de identificação e autenticação de usuários são recomendados e podem ser utilizados como forma de aumentar a segurança na validação de usuários

4.3 Classificação da Informação

A informação deve ser classificada como forma de otimizar seu tratamento. Sabendo qual a real importância e necessidade de segurança das informações, os serviços de log, back-up e outros tratamentos podem focar necessidades específicas dando prioridade a informações mais críticas e não realizando manipulações desnecessárias em arquivos.

Em grandes organizações, a classificação da informação pode gerar uma melhoria extraordinária em questão de tempo de processamento, segurança da informação e quantidade de informações de back-up e logs armazenadas.

Os tipos de classificação de informação mais comuns são:

- Uso confidencial: São as informações mais importantes. São o ativo da empresa. Sua disseminação deve ser estrita e controlada pois o uso indevido desta informação pode causar prejuízo para a organização.
- Uso Interno: São informações restrita apenas a funcionários internos dentro da organização.
- Uso Público: São as informações que podem ser disponibilizadas a terceiros, clientes e outros.

4.4 Política de Firewall

Dentro de uma política de defesa, devem existir regras para implantação, gerenciamento e suporte de ferramentas que possam aumentar a segurança em seu ambiente. Essas regras devem incluir a forma e o local onde as ferramentas devem ser instaladas, boas práticas de instalação, configuração e gerenciamento e suporte.

Uma destas ferramentas é o firewall.

É importante reforçar que a palavra firewall não se refere a um dispositivo, mas sim a um conceito. Um conceito de se garantir uma segurança através de uma análise e seleção sobre o que deve ou não ser permitido em um sistema onde essa análise e seleção podem ser realizadas através de vários conceitos e tecnologias existentes. Porém, algumas vezes, podemos introduzir um conceito de firewall que será representado apenas por um dispositivo.

Neste tópico, serão apresentados os principais conceitos e definições de firewall, além de tecnologias e principais tipos de firewalls hoje utilizados assim como suas vantagens e desvantagens. Será apresentado também, como as arquiteturas de firewall se comportam frente a algumas tecnologias, conceitos e implementações de transmissão de pacotes em uma rede de computadores.

O conceito de um firewall é tão útil e as vezes tão simples que é comum vermos um firewall configurado com menos de uma dezena de regras conseguir solucionar 50% dos problemas relacionados com a entrada de pacotes indevidos em uma rede.

Esta seção tem por objetivo apresentar algumas das principais tecnologias de firewalls utilizadas hoje em dia, apresentando seus conceitos e componentes que podem ser usados para a criação de arquiteturas de firewalls diferenciadas e quais são as principais diferenças conceituais e de projeto de cada um deles. As tecnologias são separadas de acordo com o tipo e a forma como será feito o tratamento dos pacotes.

4.4.1 Tecnologias

Durante a elaboração de um plano de defesa, a decisão da tecnologia de firewall utilizada é muito importante. O firewall deve ser a primeira linha de defesa da

organização criando um ponto de estrangulamento que obrigue todo o tráfego a passar por um ponto que se possa gerenciar.

Muitas vezes, deve-se optar por uma solução composta de mais de uma tecnologia, em pontos diferentes do sistema criando uma defesa em profundidade.

4.4.1.1 Filtro de Pacotes

A tecnologia de filtragem de pacotes efetua uma seleção do que deve ser ou não encaminhado entre o interior e o exterior do seu sistema. Essa decisão é efetuada basicamente com base em dados contidos nos cabeçalhos dos pacotes.

Entre as informações contidas nos cabeçalhos dos pacotes e que devem ser utilizadas para a decisão de encaminhamento podemos citar algumas:

- Endereços IP de origem e destino
- Máscaras de endereços IP de origem e destino
- Protocolo utilizado (ICMP, TCP, UDP)
- Portas de origem e destino (para protocolos TCP e UDP)
- Tipo da mensagem ICMP

Estes são alguns dos campos mais comuns que podem ser utilizados pelos filtros de pacotes. Porém, algumas opções mais detalhadas como o nome da página da Web que está sendo solicitada também é possível.

Após examinar estas informações, o filtro de pacotes [11] pode executar opções como:

- Permitir que o pacote seja repassado ao seu destino
- Descartar o pacote sem notificar o remetente
- Rejeitar o pacote notificando o remetente
- Armazenar informações sobre o pacote
- Disparar um alarme para notificar alguém (por exemplo o administrador) sobre o pacote imediatamente

Filtros de pacotes mais sofisticados também podem executar ações como:

- Modificar o pacote (na conversão de endereços de rede por exemplo)
- Redirecionar o pacote a outro destino
- Modificar regras de filtragem

A filtragem de pacotes é muito eficiente e pode ser realizada com um baixo overhead já que o filtro examina apenas informações de alguns cabeçalhos.

4.4.1.2 Proxys

A tecnologia de proxy é baseada no fornecimento de serviços através de programas aplicativos ou servidores especializados em receber as solicitações dos usuários ou outros sistemas da rede e efetuar o encaminhamento aos serviços reais.

Existem serviços de proxys para realizar diversas funções porém, neste ponto, iremos tratar apenas serviços de proxy para fins de segurança.

Assim como na tecnologia de filtro de pacotes, a utilização de um serviço de proxy é útil por criar um ponto de estrangulamento obrigando todas as conexões a passarem pelo proxy tendo assim um ponto de controle. Porém, a utilização de proxy só fornece uma qualidade e melhoria de segurança considerável quando se utiliza um mecanismo em conjunto para impedir que conexões entre o interior de sua rede e o mundo exterior (redes externas ou Internet) possam ser abertas diretamente contornando assim seu serviço de proxy.

As vantagens da tecnologia de proxies é o fato de poderem trabalhar com camadas superiores fornecendo assim mais informações para que possam ser efetuados testes de validade e consistência dos pacotes, além permitir:

- Um melhor registros dos logs
- Filtragem mais inteligente
- Autenticação a nível de usuário

As principais desvantagens dos serviços de proxy se baseiam no fato de:

- Existir serviços que não trabalham em conjunto com proxies criando assim um furo na sua política de firewall
- Perda na performance do sistema devido ao overhead
- Algumas vezes cada serviço requer um servidor proxy diferente

4.4.2 Arquiteturas

Existem várias formas de se reunir, configurar e instalar componentes de firewalls. Estas arquiteturas deverão ser escolhidas com atenção para que a segurança possa ser obtida através da opção que melhor se enquadre nas necessidades em pontos críticos de uma organização.

4.4.2.1 Caixa Única

Pode ser considerada como a mais simples arquitetura de firewall. Seu principal conceito consiste em englobar toda a sua solução de firewall em um único ponto criando assim uma solução de configuração de fácil entendimento e visualização.

Essa é uma solução muito fácil de entender em comparação com arquiteturas complexas e integradas de várias camadas. Daí temos a sua principal vantagem que é a simplicidade e praticidade da solução.

Porém, essa solução gera um ponto crítico na sua rede. No caso de falha no firewall, toda sua rede ficará exposta ou toda sua rede ficará inacessível (dependendo da configuração utilizada) além de não criar uma defesa em profundidade.

Muitas vezes, essa solução é feita através de roteadores de triagem (screening routers) por questões financeiras devido ao aproveitamento de componentes já existentes na sua rede, pois quase sempre uma organização precisará de um roteador para se conectar a Internet.

É uma solução que pode ser feita através de firewalls já existentes, hosts dual-homed ou roteadores de triagem. Essa solução é mais indicada:

- Para pequenos sites onde o acesso a Internet não é crítico em termos de negócio
- O número de protocolos e serviços usados é pequeno
- O tráfego Internet é pequeno
- Não existe tráfego muito sigiloso na rede
- Quando já existe um alto nível de segurança de host

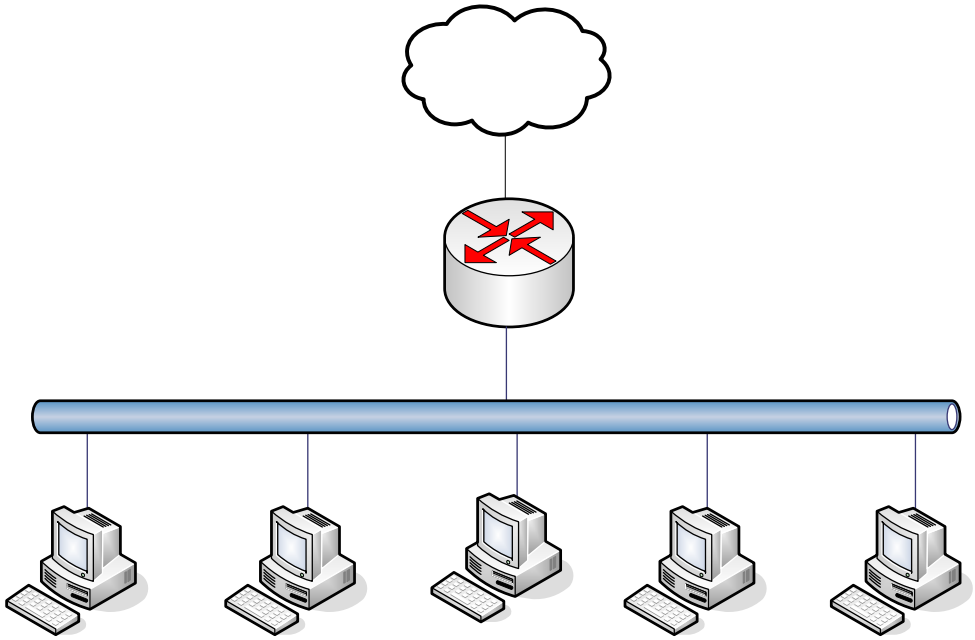


Figura 5 - Arquitetura de Caixa Única

4.4.2.2 Host com Triagem

Nesta arquitetura continua-se utilizando um roteador de triagem, porém acrescentando um host de bastião na rede interna como forma de incrementar a segurança. Desta forma, a solução pode ser configurada de tal forma que somente o host de bastião consiga efetuar conexões com a rede externa e somente conexões permitidas (de acordo com a sua política de regras através do roteador de triagem). Os host internos da rede deverão criar suas conexões através do host de bastião que funcionará como um proxy. Porém, a estrutura pode ser configurada de outras formas tais que computadores internos também abram conexões com a rede externa porém isso tem um efeito negativo na segurança. Nesta arquitetura, a tecnologia de filtragem de pacotes tem um papel fundamental pois ela irá impedir que computadores internos tentem burlar as regras de segurança e contornem o host de bastião que está atuando como um proxy para acessar a rede externa diretamente.

Rede I

Neste ambiente, o host de bastião fica sendo um ponto muito exposto na sua estrutura. Devido a isso não é interessante fornecer serviços importantes neste host de bastião como por exemplo um servidor de e-mail e é fundamental que sua rede tenha um bom nível de segurança de host.

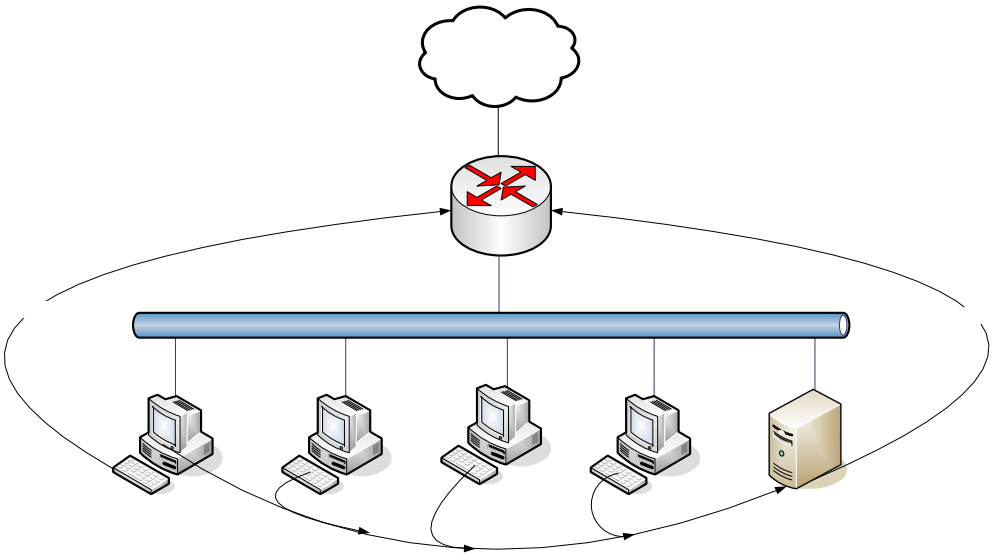


Figura 6 - Arquitetura de Host com triagem

4.4.2.3 Sub-Rede com Triagem

Assim como a arquitetura de host com triagem provê uma maior segurança que a arquitetura de caixa única, a arquitetura de sub-rede com triagem visa acrescentar mais uma camada extra de segurança, adicionando uma rede perímetro para isolar ainda mais a rede interna da Internet. Essa rede de perímetro é conhecida também como DMZ (De-militarized zone).

Por serem seu principal ponto de conexão com o mundo externo, seus hosts de bastião são sempre o ponto mais vulnerável de seu sistema. Na arquitetura de host com triagem, seu host de bastião está localizado dentro de sua rede interna. Se ele for atacado, nada impede do atacante ter acesso a seu hosts internos.

Tráfego não permitido

Na arquitetura de sub-rede com triagem, os seus hosts de bastião ficam localizados na rede de perímetro. Com essa configuração, existirá dois firewalls em sua rede, um firewall externo que liga o mundo externo a sua rede de perímetro e um firewall interno para interligar a rede de perímetro com seu ambiente interno.

Ao se trabalhar com essa configuração é extremamente interessante configurar o sistema de modo que não haja comunicação direta entre o ambiente interno e externo de sua rede. Assim, você poderá obrigar determinados tráfegos de entrada ou saída a passar por proxys localizados em sua rede de perímetro. Desta forma, usuários internos mal intencionados não conseguirão identificar diretamente sua estrutura interna. Essa

decisão de quais serviços serão fornecidos por proxys deve ser realizado na sua análise de necessidades onde foi definido quais serviços são críticos e podem oferecer brechas de segurança.

A escolha de seus host de bastião deve ser feita cuidadosamente. Este host deverá ser configurado de uma forma extremamente segura. Ele é a sua presença pública na Internet. Por isso, alguns cuidados devem ser tomados como:

a) Esteja sempre preparado para indisponibilidades no seu host de bastião: Apesar de todos os esforços com segurança, mantenha em sua mente que você será atacado. Esteja preparado para isso.
b) Escolha bem a localização e segurança física de seus host de bastião.
c) Escolha bem hardwares e softwares
d) Tente não sobrecarregar um host de bastião com vários serviços. Realize uma análise. Tente agrupar serviços de acordo com seu nível de importância, segurança e quantidade de acessos.
e) Desative todos os serviços e portas desnecessários. Deixe funcionando somente o mínimo necessário para a execução das tarefas desejadas.
f) Tente manter atualizado seu host de bastião contra os ataques e ameaças. Várias organizações lançam periodicamente (às vezes diariamente) relatórios sobre falhas em aplicativos, serviços e sistemas operacionais sobre falhas de proteção encontradas, e patches de correção.
g) Tenha um log para analise de ataques, catástrofes porém decida o que realmente é necessário guardar em log para não ter uma enorme quantidade de informações inúteis.
h) Sempre realize back-up (de preferência off-line) de seu host de bastião.

Além de seu host de bastião, outro ponto que deve ser tratado com cuidado são seus roteadores de triagem externos e internos.

O seu roteador interno é o ponto de estrangulamento da sua rede. Ele é o ponto de conexão e filtragem entre sua rede interna e sua rede de perímetro. Sua configuração deve ser feita com muita atenção sempre seguindo as regras elaboradas na política de firewalls e proibindo na medida do possível, conexões diretas entre a rede interna e o mundo exterior sem passar pelo host de bastião.

O seu roteador externo é o ponto de contato direto com o mundo exterior. Ele tem a função de proteger tanto sua rede de perímetro quanto a rede interna. Um tipo de filtragem obrigatório neste roteador é a filtragem de endereços IPs forjados na entrada (endereços reservados por exemplo).

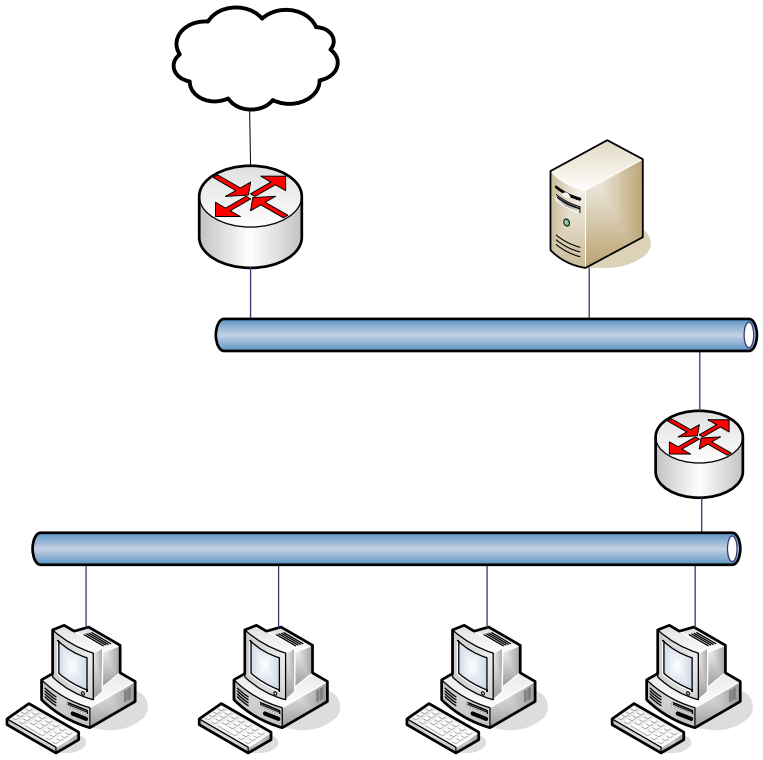


Figura 7 - Arquitetura de sub-rede com triagem

4.4.2.4 Firewalls Internos

Apesar da maior parte das considerações se relacionarem com a utilização de um firewall protegendo sempre um ambiente interno de um ambiente externo, freqüentemente existem situações onde precisamos dentro de uma organização, garantir uma segurança entre as redes internas. Isso se deve ao fato de que em um ambiente um pouco maior, podemos ter redes internas mais inseguras e expostas a problemas e ataques, redes que trafegam tráfegos sigilosos dentro da empresa e precisam ser superprotegidas, redes se conectando a redes internas de outras organizações através de parcerias ou simplesmente redes de testes e laboratórios que não devem ser acessadas por outras redes e que podem gerar situações indesejadas para o restante da organização.

Roteado
Triagem E

Desta forma, é necessário proteger uma rede interna de outra rede interna dentro do seu ambiente. Para isso é preciso inserir firewalls entre estas redes internas e definir regras mais especificas para a configuração deste tipo de firewall, tendo em conta as necessidades dos segmentos relacionados ao firewall. Nestas situações, podemos ter uma completa mistura de tecnologias e arquiteturas, tendo dentro de empresas arquiteturas como hosts de bastião, redes de perímetro todas trabalhando em conjunto gerando uma boa defesa em profundidade.

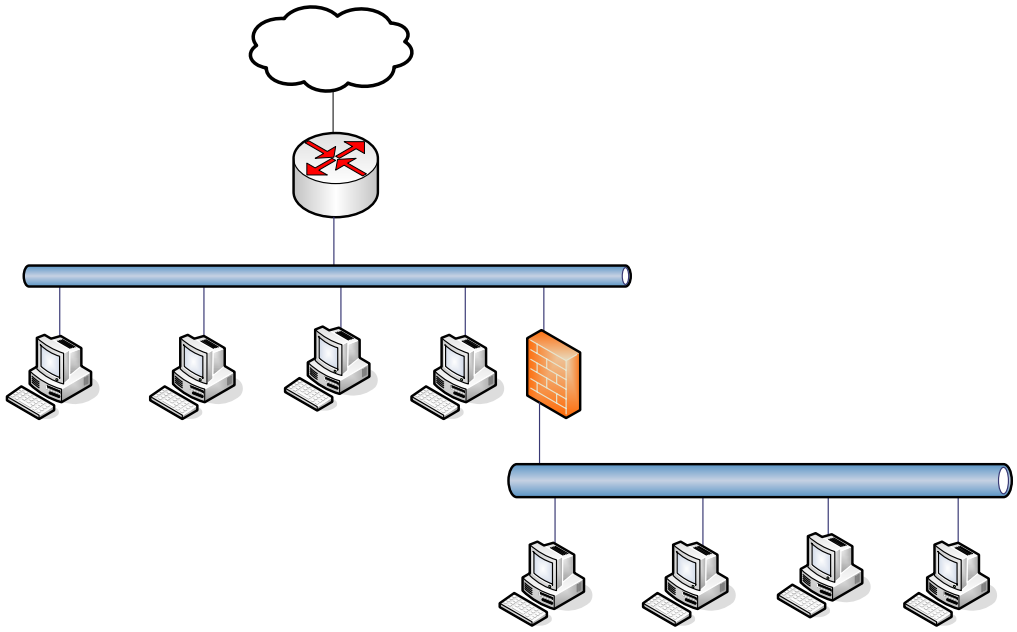


Figura 8 - Exemplo de firewall interno

4.4.3 Projeto

Ao projetar uma solução de firewall, muitos pontos devem ser levados em consideração. Uma solução muito provavelmente não envolverá uma única tecnologia. Uma boa solução de um projeto de firewalls normalmente envolve várias tecnologias e arquiteturas diferentes com o intuito de tentar resolver a maior parte dos problemas possíveis.

Um bom ponto de partida para o desenvolvimento do seu projeto seria sua política de segurança. Um firewall irá interferir diretamente com o que será ou não permitido na sua rede, logo um projeto feito sem a existência ou consideração de sua política de segurança pode gerar problemas futuros.

Se você está tentando se proteger, é interessante saber o que você precisa proteger e de quem você precisa se proteger. Os projetos devem ser feitos analisando a real estrutura da organização e suas reais necessidades. Não existe uma solução única para todos os casos. Questões negociais e financeiras interferem e são muito importantes nestes projetos. Analise o ambiente utilizado, quais são os dados e serviços críticos, como, quando e por quem eles são utilizados. Defina os pontos críticos e quais os impactos e custos gerados quando o sistema parar. Avalie a necessidade de redundância. Estudar o ambiente onde o firewall irá trabalhar é fundamental. Certos ambientes possuem facilidades e pontos que podem ser aproveitados na sua solução, enquanto outros irão gerar dificuldades e mais questões a serem analisadas.

É importante estudar e analisar o produto que deseja utilizar com relação a suporte, facilidade de configuração e gerenciamento, além de opções extras como preço, processos de log e auditoria.

A melhor solução muitas vezes não é a solução mais completa e sim a solução que melhor se enquadra nas necessidades de segurança e orçamento da organização. A relação custo benefício também é muito importante. Soluções caras muitas vezes podem se tornar impeditivas devido a natureza da organização. Em outros casos, a segurança é o ponto fundamental levando as questões financeira a segundo plano.

4.5 Política de Sistema de Detecção de Intrusão (SDI)

Outra ferramenta que desempenha um papel fundamental na fortificação de sistemas conectados à Internet é o Sistema de Detecção de Intrusão (SDI).

Um sistema de detecção de intrusão é basicamente uma ferramenta de monitoração da rede. Esta ferramenta monitora a rede em busca de tráfegos suspeitos. Em alguns casos, o SDI pode tomar ações em relação ao tráfego malicioso bloqueando usuários ou alterando regras de firewall.

Neste item, serão apresentados os principais tipos, tecnologias e arquiteturas de SDIs. Cabe ao administrador analisar estas informações e decidir se deve ou não utilizar um SDI em seu sistema e qual a tecnologia e arquitetura devem ser utilizadas. Soluções mistas (envolvendo mais de uma tecnologia e arquitetura) podem garantir uma segurança em profundidade extremamente eficiente.

4.5.1 Tipos de SDI

Os SDIs são divididos basicamente de acordo com o tipo de tráfego monitorado. Sistemas baseados em host e em rede são os mais comuns.

4.5.1.1 Baseado em Host

Os SDIs baseados em host monitoram o tráfego em um host específico. Eles são instalados diretamente em um host e são configurados para monitorar todo o tráfego de entrada e saída e gerar alertas para o administrador de rede no caso de tráfego suspeito naquele host.

4.5.1.2 Baseado em Rede

Os SDIs baseados em rede são instalados em lugares específicos da rede para que possam monitorar o tráfego de entrada e saída de todos os dispositivos da rede. É muito importante que um projeto bem elaborado de um SDI baseado em rede possa monitorar todo o tráfego desejado sem que se torne um gargalo em sua rede.

4.5.2 Tecnologias

As tecnologias de SDI são pautadas essencialmente na forma como o SDI irá analisar o tráfego para dizer se é mesmo é ou não suspeito de ser um ataque. As duas principais tecnologias são mostradas a seguir:

4.5.2.1 Baseado em Assinatura

Os SDIs baseados em assinaturas possuem uma base de dados similar as bases de dados dos antivírus, com assinaturas e atributos de ameaças conhecidas. Essa tecnologia compara todo o tráfego com a base de dados para identificar ameaças ao sistema.

Essa tecnologia possui uma falha pois quando uma nova ameaça é descoberta, o SDI precisa ser atualizado para tratar aquela ameaça. Durante este espaço de tempo entre a descoberta da falha e a atualização do SDI, o sistema fica vulnerável ao novo tipo de ataque.

4.5.2.2 Baseado em Anomalia

Os SDIs baseados em anomalias comparam o tráfego com uma linha base estipulada. Esta linha base serve para identificar um tráfego “normal” do sistema. Para compor esta linha base, alguns parâmetros são utilizados. Os mais comuns são:

- Largura de banda
- Protocolos utilizados
- Portas abertas

Estes parâmetros são os mais comuns, porém, não são os únicos. Comparando o tráfego lido com a linha base, o SDI efetua suposições sobre o tráfego indicando, se o mesmo está ou não fora da normalidade e se esta anormalidade pode significar um ataque.

4.5.3 Arquiteturas

Os SDIs pode ser configurados para trabalhar em diferentes arquiteturas de acordo com a necessidade. As duas principais arquiteturas utilizadas são apresentadas a seguir.

4.5.3.1 Passivo

Quando configurado para trabalhar no modo passivo, o SDI simplesmente realiza detecções e alertas. Quando tráfego malicioso ou suspeito é detectado, um alerta é gerado e enviado para um determinado usuário ou administrador para que o mesmo possa tomar alguma ação para bloquear a atividade ou responder de alguma forma [12].

4.5.3.2 Reativo

O SDI reativo não somente detecta tráfego malicioso ou suspeito como também é capaz de realizar ações pró-ativas pré-determinadas para responder a ameaça. Essa resposta pode vir através de um bloqueio da conexão ou até mesmo reconfiguração das regras de firewall para proibir que determinados IPs de origem efetuem novas conexões.

4.5.4 Falso Positivo / Falso Negativo

Um grande problema existente nos SDIs são os falsos positivos e falsos negativos.

Os falsos positivos ocorrem quando um tráfego que deveria ser considerado normal e permitido, por algum motivo, é considerado pelo SDI como um tráfego suspeito. Isso muitas vezes pode ocorrer quando o pacote ou a conexão possui alguma característica que se assemelha a uma ameaça.

Os falsos negativos ocorrem quando um tráfego que deveria ser considerado suspeito, não é detectado pelo SDI e penetra em seu sistema.

Na política de Sistemas de Detecção de Intrusão deve constar quais são os níveis aceitáveis de falsos positivos e negativos aceitos. Na aquisição de um SDI, deve-se exigir do fornecedor uma tabela de estatísticas a respeito destas informações.

4.6 Educação e Treinamento

A educação e o treinamento em segurança da informação são fundamentais para garantir que todos os usuários estejam cientes da política implantada na organização com relação a procedimentos operacionais e implicações legais.

Os usuários devem receber um treinamento sobre boas práticas de segurança em questões operacionais e procedimentos na ocorrência de incidentes de segurança e/ou mal funcionamento do sistema. Neste treinamento deve ser reforçado que estes incidentes devem ser informados a equipe responsável pela segurança o mais rápido possível. Esta integração entre todos os usuários da organização visa ajudar na questão do comprometimento geral dos funcionários com relação a segurança e contribuir para que os mesmos aprendam com os erros anteriores minimizando assim que esse problema venha a acontecer com a mesma intensidade no futuro.

4.7 Segurança dos Equipamentos e dos Discos Removíveis

A política de segurança deve impor regras para garantir a segurança dos equipamentos, assim como dos discos removíveis. Isso é importante para prevenir o comprometimento ou perdas de bens e até mesmo a interrupção do negócio.

Através das regras impostas, a política de segurança dos equipamentos deve considerar os pontos fundamentais para implantar e proteger os equipamentos. Questões como a forma e o local da instalação devem ser considerados para minimizar estes riscos. Algumas das questões a serem consideradas são:

- Fogo, roubo, inundações e explosões
- Problemas no fornecimento da energia elétrica
- Segurança e manutenção de equipamentos dentro e fora da organização

Meios de armazenamento ópticos e magnéticos também constituem pontos a serem analisados e considerados na política de segurança, devido, principalmente, aos dados que podem estar contidos nestes meios. A política de segurança deve prever questões como:

- Manipulação de meios removíveis
- Tratamento dos meios magnéticos em trânsito
- Destruição dos meios magnéticos

4.8 Desenvolvimento, Manutenção e Gerenciamento de Sistemas

O desenvolvimento de software deve ser feito pautado em regras de segurança estabelecidas pela política de segurança elaborada. De nada adianta uma boa estrutura de segurança dos equipamentos e ativos de rede se os aplicativos e serviços executados sobre essa estrutura possuem brechas consideráveis. Tanto no momento do desenvolvimento e implantação como também na fase de manutenção, gerenciamento, homologação e testes, as questões de segurança devem ser constantemente consideradas.

4.9 Processos para garantia da segurança da informação

No início deste capítulo, foi citado que o fornecimento de segurança da informação se dá basicamente pela garantia da integridade, confidencialidade e disponibilidade da informação. Este tópico vem explicar alguns conceitos e

tecnologias que podem ser utilizadas como formas de aumentar a segurança dos sistemas conectados à Internet.

4.9.1 Criptografia

A criptografia entra neste contexto como forma de garantir a confidencialidade dos dados e informações. O processo criptográfico consiste basicamente em embaralhar e codificar os dados para que, caso aconteça uma captura destes dados ou um seqüestro de sessão, o atacante não consiga ler as informações.

As técnicas e algoritmos utilizados podem variar bastante de acordo com a necessidade de uma criptografia forte ou o tipo de criptografia.

4.9.2 Assinatura Digital

A assinatura digital visa garantir a integridade e autenticação de mensagens. Através da assinatura digital, é possível saber quem está enviando as mensagens e se a mesma foi modificada no meio do caminho.

Esse método é obtido através da aplicação de técnicas criptográficas das mais variadas (assim como os serviços de criptografia). Os tipos de assinaturas digitais mais comuns são as assinaturas de:

- Chave simétrica: É utilizado apenas uma chave de criptografia para codificar e decodificar as informações
- Chave assimétrica: É utilizado um par de chaves (chave pública e chave privada). O que é codificado por uma só pode ser decodificado pela outra chave e vice-versa. A chave privada é utilizada para assinar documentos digitalmente enquanto a chave pública é utilizada para verificar a assinatura.

4.9.3 Serviços de não-repúdio

Através da utilização combinada de técnicas de criptografia e assinatura digital, pode ser criado um ambiente extremamente útil na resolução de questões referentes a veracidade das mensagens trocadas entre sistemas (principalmente no comércio eletrônico). Com isso, se pode afirmar que a mensagem recebida foi realmente enviada pelo remetente da mensagem e o mesmo não tem como negar esse envio.

4.10 Auditoria

Um bom plano de auditoria deve ser incluído em uma política de segurança. A auditoria serve para garantir que as regras existentes na política de segurança e defesa de uma organização estão sendo executadas e se essa execução está sendo feita de forma correta.

A auditoria deve examinar ponto a ponto a política e fazer uma análise minuciosa para que falhas no plano de defesa ou pontos que não foram considerados possam ser revisados, atualizados ou incluídos o mais rapidamente na política de segurança.

Outro ponto importante é que a auditoria deve ser feita preferencialmente por pessoas que não estejam envolvidas com funções críticas para a segurança de forma que o resultado da auditoria possa ser imparcial e o mais real possível. Contratação de empresas terceirizadas especializadas em auditorias pode ser considerada uma boa solução.

4.11 Plano de Continuidade de Negócios

O plano de continuidade de negócios (PCN) tem o objetivo de prover a continuidade de serviços, informações e processos de uma organização e tentar diminuir ao máximo os impactos de um eventual desastre no menor tempo possível. Um plano de continuidade de negócios deve ser elaborado com cuidado. A seguir, são apresentadas algumas características que devem estar contidas em um plano de continuidade de negócios.

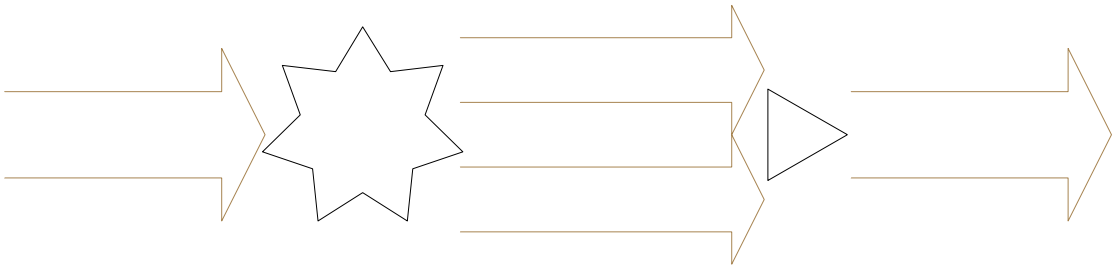


Figura 9 - Plano de Continuidade de Negócios

4.11.1 Análise de Impactos

A etapa de análise de impactos é a parte do plano de continuidade de negócios onde é necessário estudar o negócio da empresa. Nesta fase deve-se utilizar os dados coletados na análise de riscos e estudar cada uma das ameaças e seus impactos no caso de um desastre. É necessário estudar e analisar as chances de que um problema físico ou natural (fenômenos naturais devem ser levados em conta) aconteça, desastres que podem ser gerados por equipamentos obsoletos ou por falta de treinamento adequado na operação ou manutenção dos sistemas, quais serão os custos do "down time" além dos custos de recuperação. Essa análise é fundamental para uma melhor alocação de recursos e como tentativa de minimizar os efeitos e impactos gerados por um eventual desastre.

Operação Normal

4.11.2 Plano de Contingência

Após definidos todos os pontos de prováveis falhas, e suas respectivas probabilidades, deve ser criado o plano de contingência. O plano de contingência deve conter as medidas a serem tomadas no caso de uma falha. Deve utilizar todos os pontos descritos na analise de impactos e definir soluções viáveis no caso de falhas. Isso pode ser feito criando cenários de falhas no qual se possa observar claramente o que pode acontecer no caso de cada falha. Esse plano deve conter todas as soluções para cada tipo de falha e medidas a serem tomadas em um eventual desastre.

Os cenários de testes também devem ser descritos, juntamente com o plano de testes destes cenário para que não ocorram surpresas e imprevistos na hora de uma falha. O teste é fundamental. Muitos profissionais de segurança descobrem que seu

plano e soluções de contingência não funcionam justamente na hora em que precisam delas.

É importante definir nesta etapa, as condições de ativação de cada plano construído assim como os procedimentos de emergência que devem ser tomados no caso de um desastre.

4.11.3 Plano de Recuperação de Desastres

O plano de recuperação de desastres visa promover uma recuperação da atividade normal da organização. Neste plano deve estar definido o que era normal antes do incidente e o que será considerado normal após o mesmo. Neste estágio, é necessário elaborar um planejamento dinâmico de recuperação. Neste ponto deve ser decidido como será feita a substituição dos softwares e hardwares afetados pelo incidente assim como as políticas de back-up, replicação e recuperação de dados, planejamentos relativos a alocação de recursos e pessoas no caso de um desastre, além de uma equipe de administração de crises. Essa fase termina quando o serviço volta a sua normalidade.

4.11.4 Notificação

É extremamente importante que a organização possua uma equipe responsável unicamente pela manutenção da segurança das informações e do sistema. Uma política de segurança bem elaborada deve possuir informações e contatos de pessoas que devem ser notificadas no caso de falhas e desastres.

Essas pessoas deverão ter o conhecimento técnico necessário e possui contatos com outras equipes de segurança para que se possa resolver os problemas de forma adequada.

4.12 Conformidade (Compliance)

Durante a elaboração de todas as etapas de uma política de segurança, questões para garantir a conformidade do plano de segurança devem ser consideradas. A

conformidade serve para evitar que seu plano de segurança viole quaisquer leis civis ou criminais, obrigações contratuais ou qualquer requisito de segurança.

A legislação sobre segurança em informática irá variar de país para país, porém, seu plano de segurança deve ser feito de acordo com a legislação. Existem algumas questões que devem ser cuidadosamente verificadas para que futuramente não venham a atrapalhar a política de segurança. De nada adiantará uma política de segurança se, quando precisar dela, problemas de cunho legal possam interferir na execução das regras escritas na política. Algumas destas questões são:

- Direitos de propriedade intelectual
- Direitos autorais
- Direitos de projeto
- Marcas registradas
- Direitos autorais de software
- Privacidade de informações pessoais
- Regulamentação de controles criptográficos
- Regras para adequação, admissibilidade, qualidade e exatidão das evidências

Capítulo 5 – Simulador de Filtragem de Pacotes

Este capítulo apresenta o simulador de filtragem de pacotes criado como mecanismo de validação deste projeto de pesquisa. O intuito do desenvolvimento deste simulador foi o de apresentar as vantagens em termos de segurança que um dispositivo de filtragem de pacotes pode trazer para um ambiente conectado à Internet.

Para que haja uma melhor compreensão do projeto e dos detalhes de implementação, o início deste capítulo irá apresentar algumas considerações importantes ao se tratar de detalhes internos da estrutura do software de protocolo TCP/IP.

5.1 Detalhes Internos do TCP/IP

O TCP/IP é na verdade um conjunto de protocolos que tem por objetivo possibilitar uma comunicação segura e com qualidade entre hosts ou entre redes e tem esse nome devido a alguns (porém não os únicos) de seus principais protocolos que são o protocolo TCP e o protocolo IP.

O conjunto de protocolos TCP/IP é sem dúvida o protocolo mais utilizado nas tecnologias de rede nos dias de hoje. Apesar de utilizado por milhões de pessoas em todo o mundo, os detalhes internos do TCP/IP são ainda desconhecidos para a maioria de profissionais de informática. Isso é um erro pois esse entendimento é de fundamental importância. Desenvolvedores que conhecem a fundo o protocolo TCP/IP sabem que podem desenvolver produtos mais robustos e mais seguros.

Uma razão que pode explicar esse desconhecimento do protocolo se dá devido a documentação do TCP/IP onde os protocolos são tratados individualmente, deixando assim uma dúvida com relação à como se dá o funcionamento destes protocolos em conjunto. Isso se torna complicado pois quando tentamos ver esse funcionamento em conjunto percebemos que vários efeitos gerados por essa interação são complicados. É preciso criar, entender e utilizar as estruturas de dados de um protocolo pensando não só naquele protocolo mas em todo o conjunto dos demais protocolos que serão utilizados.

O software TCP/IP (é assim que chamaremos o conjunto de protocolos TCP/IP utilizados em conjunto) se localiza na maioria das vezes no sistema operacional podendo ser utilizado por aplicativos e processos existentes na máquina [13].

Nos tópicos a seguir serão mostrado detalhes internos dos protocolos utilizados no desenvolvimento do simulador de filtragem de pacotes.

5.1.1 Ethernet

O protocolo ethernet é um protocolo utilizado pela camada de enlace. Sua principal função é realizar a comunicação entre outros softwares de protocolo e os dispositivos da rede. Uma outra função de muita utilidade é que esse protocolo faz uma abstração das interfaces de rede, separando assim os protocolos de alto nível dos detalhes do hardware.

Este protocolo é subdivido nos padrões IEEE 802.3, que trata sobre o modo de comunicação da camada física do protocolo utilizado pela placa de rede, o IEEE 802.2 que realiza os tratamentos e interligações lógicas entre a camada física e os protocolos superiores.

As definições utilizadas pelo simulador de filtragem de pacotes com relação a estrutura do pacote ethernet, são definidas na biblioteca if_ether.h. Essa estrutura pode ser visualizada a seguir:

```
struct  ether_header {
        u_char    ether_dhost[6];
        u_char ether_shost[6];
        u_short ether_type;
}
```

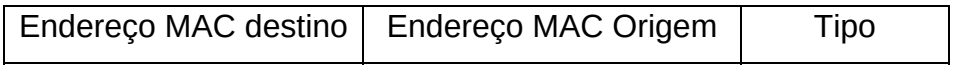


Figura 10 – Formato do cabeçalho ethernet

onde: ether_dhost e ether_shost representam respectivamente os endereços da interfaces de rede de destino e origem do pacote e ether_type representa o tipo do protocolo (por exemplo: ARP, REVARP ou IP).

5.1.2 IP

O protocolo IP pode ser considerado basicamente como um ponto central de comutação em todo o software TCP/IP. Ele é executado como um processo autônomo que realiza todas as decisões de roteamento, decisões de entrega dos datagramas de entrada para os protocolos superiores ou montagem e decisão de destino para os datagramas de saída. Ele utiliza filas de entrada e saída e roteamento uniformes, além de tratar a interface de pseudo-rede como forma de diminuir a existência de casos especiais no código.

É o código IP que realiza toda a escolha de datagramas a serem enviados ou roteados e aplica as políticas necessárias com relação a prioridades e tratamentos especiais.

Trabalhando como processo autonômo e sendo um comutador central, o protocolo IP poderá tratar várias interfaces de rede ao mesmo tempo (quando existirem). O protocolo efetua uma política de rodízio entre todas as interfaces existentes de modo que, a princípio, nenhuma tenha prioridade sobre a outra não onerando nem sub-utilizando nenhuma interface.

Questões como roteamento e fragmentação também são executados pelo protocolo IP porém não serão explicados aqui, pois o tópico roteamento não se inclui no escopo deste trabalho e o simulador de filtragem de pacotes não foi implementado para tratar pacotes fragmentados. O protocolo IP é definido pela RFC 791.

O formato completo do pacote IP é mostrado na figura abaixo:

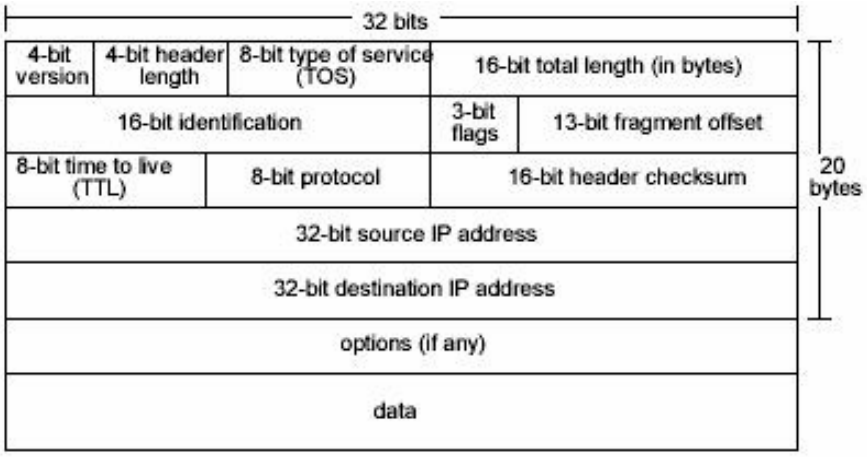


Figura 11 - Estrutura do pacote IP

Em termos de programação, a estrutura do cabeçalho ip é definida na biblioteca ip.h. A estrutura é apresentada a seguir:

```
struct ip {
#ifdef _IP_VHL
        u_char   ip_vhl;                /* versão */
#else
        u_char ip_hl:4;                 /* tamanho do header */
        u_char ip_v:4;                  /* versão */
#endif
        u_char ip_tos;                  /* tipo de serviço */
        u_char ip_len;                  /* tamanho total */
        u_char ip_id;                   /* identificação */
        u_char ip_off;                  /* campo de fragment offset */
#define IP_DF 0x4000                    /* flag de não fragmentar */
#define IP_MF 0x2000                    /* flag de mais fragmentos */
#define IP_OFFMASK 0x1FFF               /* máscara para bits fragmentados */
        u_char ip_ttl;                  /* Tempo de vida */
        u_char ip_p;                    /* protocolo */
        u_char ip_sum;                  /* checksum */
        struct in_addr ip_src, ip_dst   /* Endereços IP de origem e destino */
}
```

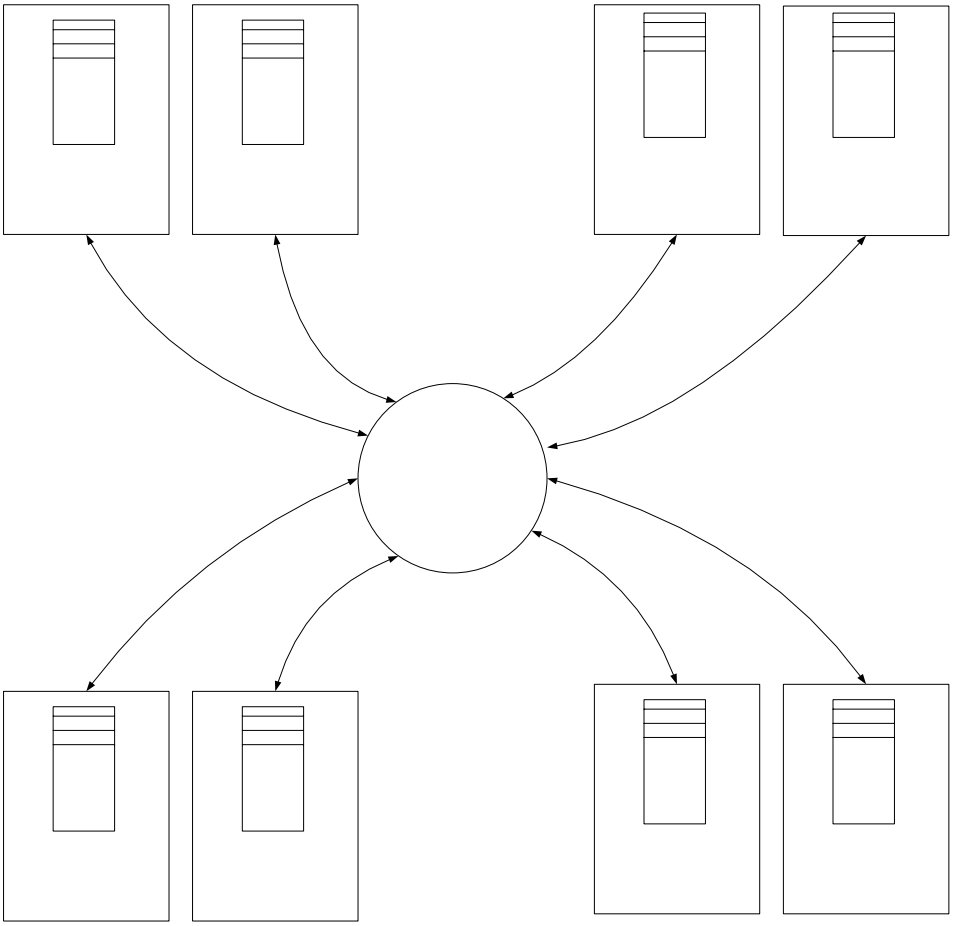


Figura 12 - IP: O comutador central

5.1.3 ICMP

O ICMP é um protocolo de camada de rede encapsulado pelo protocolo IP. Sua função é realizar o tratamento de vários tipos de erros e condições adversas ocorridas no protocolo IP, pois o mesmo não possui tratamento de erros no seu processo original.

Diferentemente da maioria de protocolos, o ICMP não possui um tipo de cabeçalho fixo. Seu formato de cabeçalho dependerá do tipo do pacote ICMP. Isso o torna um dos protocolos mais complexos do software TCP/IP.

Ele pode ser basicamente divido em dois processos principais. Um que realiza o tratamentos das mensagens ICMP que chegam e outro que trata mensagens que saem. Este protocolo é um pouco mais complexo em gateways pois a maioria dos pacotes que saem são destinados a eles.

O cabeçalho icmp é o mais complexo de todos, porém é um dos mais interessantes devido a uma série de recursos e possibilidades oferecidas por ele. O formato completo do pacote ICMP é apresentado a seguir:

```
struct icmphdr {
        u_int8_t type;                  /* message type */
        u_int8_t code;                  /* type sub-code */
        u_int16_t checksum;
        union {
                struct {
                        u_int16_t id;
                        u_int16_t sequence;
                } echo;                 /* echo datagram */
                u_int32_t gateway;      /* gateway address */
                struct {
                        u_int16_t __unused;
                        u_int16_t mtu;
                } frag;                 /* path mtu discovery */
        } un;
};
```

Porém por uma questão de simplicidade e utilização neste trabalho, iremos considerar apenas a seguinte estrutura que corresponde ao cabeçalho do protocolo ICMP:

```
struct {
        u_char icmp_type;       /* tipo da mensagem
        u_char icmp_code;       /* sub-codigo do tipo
        u_short icmp_cksum;     /* soma de checagem
}
```

5.1.4 UDP

O protocolo UDP é um protocolo bem simples utilizado pela camada de transporte. Sua função é proporcionar uma comunicação fim-a-fim entre um ou vários clientes e um servidor. Ele utiliza números de portas para criar servidores e separar os dados recebidos pelo protocolo IP em suas pilhas específicas.

O UDP é um protocolo que confia no tratamento dos demais protocolos nas outras camadas. Ele realiza poucos e simples tratamentos do seu datagrama. Seu esforço se concentra em uma rapida entrega e tratamento de datagramas UDP.

A estrutura do cabeçalho UDP é definida na biblioteca udp.h é está detalhada a seguir:

```
struct udp {                            /* formato de mensagem udp do DARPA */
        u_short u_src;                  /* número da porta UDP de origem */
        u_short u_dst;                  /* número da porta UDP de destino */
        u_short u_len;                  /* comprimento de dados UDP */
        u_short u_cksum;                /* soma de verificação UDP (0=> nada) */
        char u_data[U_MAXLEN];          /* dados contidos na mensagem UDP */
}
```

O formato completo do pacote UDP é definido como:

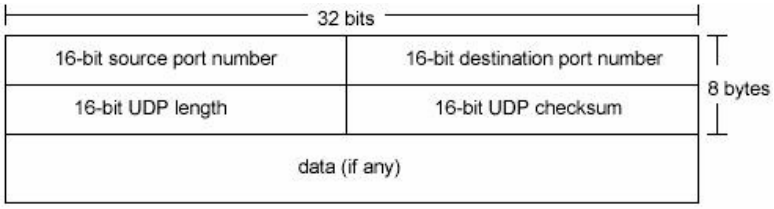


Figura 13 - Estrutura do pacote UDP

5.1.5 TCP

Assim como o protocolo UPD, o TCP tem por função proporcionar uma comunicação fim-a-fim entre um ou vários clientes a um servidor. Ele também se

utiliza de números de portas para criar servidores e separar os dados recebidos pelo protocolo IP em suas respectivas pilhas.

A principal e enorme diferença entre o TCP e o UPD é que o primeiro realiza um inúmera quantidade de testes e conferências no que diz respeito ao conteúdo e integridade do datagrama.

Dentro de sua estrutura, o TCP utiliza blocos de controle de transmissão chamados TCBs (Transmission Control Block) para controle de todas as conexões ativas.

Para realizar todo o tipo de processamento e gerenciamento, o TCP utiliza diversos tipos de controle como a máquina de estado finito, gerenciadores de timers, controle de fluxo e retransmissão adaptativa, prevenção e controle de congestionamento, processamento de dados urgentes entre outros. Alguns destes tipos de controle são extremamente complexos.

A estrutura do cabeçalho do protocolo TCP é definido na biblioteca tcp.h. Essa estrutura está definida a seguir:

```
struct tcp {
        u_short tcp_sport;          /* porta de origem */
        u_short tcp_dport;          /* porta de destino */
        tcpseq tcp_seq;             /* seqüência */
        tcpseq tcp_ack;             /* seqüência confirmada */
        u_char tcp_offset;          /* campo de fragment offset */
        u_char tcp_code;            /* flags de controle */
        u_char tcp_window;          /* anúncio de janela */
        u_char tcp_cksum;           /* soma de verificação */
        u_char tcp_urgptr;          /* ponteiro de urgência */
        u_char tcp_data[1];
}
```

Um exemplo da estrutura completa do protocolo TCP é mostrada a seguir:

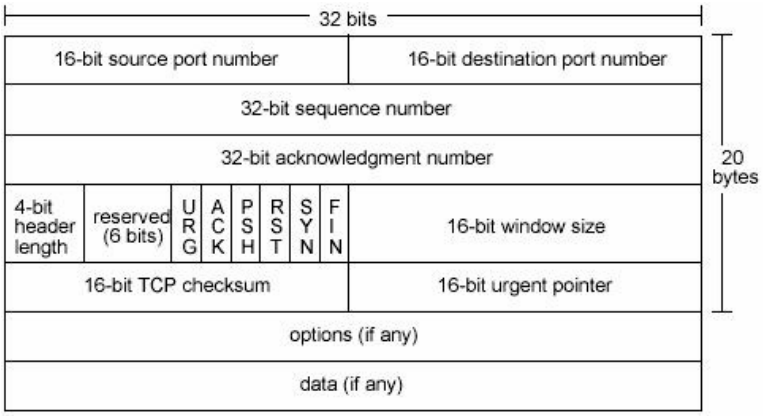


Figura 14 - Estrutura do pacote TCP

5.2 Projeto do Simulador de Filtragem de Pacotes

Para validar todo o estudo feito neste trabalho, foi implementado um simulador de filtragem de pacotes mostrando como trabalhar com a pilha TCP/IP e como alguns problemas de segurança comuns, existentes e altamente explorados, podem ser facilmente resolvidos com um simulador desta natureza.

O intuito do desenvolvimento deste simulador é poder validar alguns dos conceitos apresentados sobre segurança através de um modelo funcional, prático e relativamente simples, abrindo caminho para futuras pesquisas e melhoramentos neste projeto que ainda possui muitos tópicos a serem explorados e aperfeiçoados.

A idéia deste simulador é capturar um pacote que está entrando na placa de rede ethernet e validá-lo de acordo com as regras configuradas em um arquivo texto formatado especialmente para este fim.

As regras de configuração para o simulador são escritas em um arquivo texto devidamente formatado para facilitar sua compreensão e preenchimento. Quando o simulador é executado, ele lê todas essas regras validando o preenchimento do arquivo com relação ao conteúdo de alguns campos e guarda esses dados em memória através de uma estrutura de dados alocada dinamicamente e conhecida como lista encadeada simples onde cada regra aponta para a próxima regra.

O pacote capturado é lido e suas camadas de enlace, rede e transporte são validadas através de seus respectivos cabeçalhos. Caso o pacote seja válido no que diz respeito a consistência e formato de seus dados, ele será verificado com cada regra lida e quando o pacote se enquadrar em alguma regra, será aplicado sobre ele a respectiva regra contida na estrutura de dados. Caso o pacote não se enquadre em nenhuma regra,

será descartado devido a política de acesso utilizada que é a de Negação Padrão (DENY ALL).

Para um melhor entendimento, compreensão, estudo e futuras manutenções e melhorias deste simulador, ele foi criado em diversos módulos diferentes tentando separar funções semelhantes. Cada módulo, assim como suas respectivas funções, estruturas de dados utilizadas e funcionamento será explicado neste capítulo.

Lembramos que o objetivo deste trabalho não é exaurir todas as considerações sobre segurança e sobre a biblioteca libpcap entendendo que vários tópicos abordados neste trabalho são pontos de estudo aprofundado para aqueles que se interessarem.

Este trabalho foi desenvolvido tentando ao máximo seguir as definições padrões definidos nas RFCs além dos padrões do conjunto de protocolos TCP/IP utilizando variáveis, constantes e estruturas internas deste protocolo.

5.2.1 A biblioteca libpcap

Para realizar a captura dos pacotes, foi utilizado a biblioteca Libpcap. Esta biblioteca foi desenvolvida na Universidade da Califórnia e tem por função capturar pacotes e permitir sua recuperação e utilização de filtros BPF (Berkeley Packet Filter). Muitos projetos famosos utilizam as facilidades proporcionadas pela libpcap como snort, tcpdump e vários outros. É uma biblioteca extremamente poderosa, com várias funções úteis que permitem construção de códigos com técnicas extremamente avançadas e que podem ser usadas para segurança ou como ferramentas de prospecção e ataque [14].

5.2.2 A biblioteca simulador.h

Nesta biblioteca estão localizadas a maioria das variáveis necessárias para a utilização do simulador de filtragem de pacotes assim como as estruturas necessárias para a execução do simulador, como por exemplo a estrutura que guarda os dados lidos no arquivo de configuração. A biblioteca simulador.h é apresentada no ANEXO B.

5.2.3 O Arquivo de configuração de regras

O simulador decidirá que tipo de regra deve ser executada sobre o pacote de acordo com o arquivo de configuração de regras. Esse arquivo possui alguns campos a serem preenchidos pelo usuário de acordo com o que for desejado.

O simulador de filtragem de pacotes, utiliza uma política de menor privilégio e negação padrão. Ou seja, o simulador possui um NEGAR TUDO implícito no fim de seu arquivo de configuração de regras. Isso significa que caso um determinado pacote não se enquadre em nenhum regra existente no arquivo ele será sempre negado.

Os dados existentes no arquivo de configuração de regras do simulador de filtragem de pacotes são endereço IP de origem e destino, protocolo utilizado, portas de serviço de origem e destino e ação a ser executada sobre o pacote.
Um exemplo de arquivo de configuração de regras é apresentado no ANEXO C.

5.2.4 Instruções de preenchimento do arquivo de configuração de regras

Para que o simulador possa efetivamente capturar todos os dados para o uso e verificação da regras, esse arquivo além de estar formatado corretamente deve possuir algumas regras para que tenhamos informações consistentes. As instruções para o preenchimento do arquivo de configuração para cada campo são:

5.2.4.1 Endereço IP de origem/destino

O endereço IP de origem/destino deve ser um endereço IP válido (reservado ou não) devidamente formatado na notação decimal pontuada. Caso não queira que uma determinada regra verifique endereço de origem, deve-ser colocar um traço ‘-‘ para denotar que o campo não deve ser verificado.

5.2.4.2 Protocolo

Este campo deve possuir o nome do protocolo a ser testado. Os protocolos válidos para este projeto são:
-TCP
-UDP
-ICMP

O nome do protocolo deve ser escrito em caixa alta. Caso não deseje verificar o protocolo, deve-se colocar um traço ‘-‘ no campo protocolo.

Quando o protocolo filtrado for ICMP, o usuário terá a opção de filtrar ou permitir o protocolo ICMP total ou de acordo com o tipo de ICMP desejado. Para isso, deve-se separar o nome do protocolo ICMP por dois pontos (:) e colocar logo em seguida (sem espaçamento), o nome do tipo do ICMP desejado. Ex.: ICMP:ECHOREPLY.

Os tipos ICMP válidos são:

- ECHOREPLY - Resposta ao echo
- DEST_UNREACH - Destino inacessível
- SOURCE_QUENCH - Source quench (dissipação da origem)
- REDIRECT - Mensagem de redirecionamento
- ECHO - Solicitação de Eco
- TIME_EXCEEDED - Tempo excedido
- PARAMETERPROB - Problema de parâmetro
- TIMESTAMP - Solicitação de indicação de hora
- TIMESTAMPREPLY - Resposta de indicação de hora
- INFO_REQUEST - Solicitação de informação
- INFO_REPLY - Resposta de informação
- ADRESS - Solicitação de endereço
- ADRESSREPLY - Resposta de endereço

5.2.4.3 Porta de origem/destino

Deve ser um número de dois bytes que serve para especificar o processo cliente ou servidor de onde o pacote está vindo da máquina de origem (porta de origem) ou indo para a máquina de destino (porta de destino). Caso não deseje verificar a porta de origem/destino, deve-se colocar um traço ‘-‘ no respectivo campo. Caso o nome do protocolo tenha sido preenchido com ‘-‘ ou o protocolo utilizado for ICMP a porta de origem não será testada devido ao fato das portas estarem relacionadas com o protocolo da camada de transporte correspondente (TCP ou UDP) sendo que o ICMP não utiliza portas mas sim tipos de protocolo UDP (mostrados acima). É interessante para efeitos de organização, consistência e entendimento do arquivo de configuração que a porta de origem seja preenchida com ‘-‘ no caso do protocolo ser ICMP devido

ao fato do protocolo ICMP não possuir este conceito de portas de origem/destino ou o protocolo ter sido preenchido com ‘-‘.
Observação: O número porta de origem/destino poderá vir acompanhado de um sinal de > (maior) ou < (menor). Neste caso, o simulador irá verificar não só aquela porta específica, mas todas as portas maiores ou menores que o número informado. Ex: >1024.

5.2.4.8 Ação

Este campo informa a ação que deve ser executada sobre o pacote. Sempre deve estar preenchido (em caixa alta) com um dos seguintes valores:
PERMITIR – assim sendo, se o pacote se enquadrar nesta regra, ele será permitido.
NEGAR – assim sendo, se o pacote se enquadrar nesta regra, ele será bloqueado e a máquina não conseguirá enviá-lo ou recebê-lo.

5.2.5 Módulo de tratamento do arquivo de configuração de regras

Este módulo possui o nome formata-arq.c e possui as funções que realizam a leitura, tratamento, formatação e montagem dos dados do arquivo de configuração de regras do simulador de filtragem de pacotes em estruturas bem definidas e funções que efetuam o tratamento e formatação destes dados em formatos necessários para utilização pelo simulador.

O código fonte do módulo formata-arq.c pode ser visualizado no ANEXO D.

As funções definidas dentro deste modulo são definidas a seguir, juntamente com seus parâmetros e especificações.

5.2.5.1 Função monta_inf()

Recebe uma linha do arquivo de configuração de regras do simulador de filtragem de pacotes e retorna os dados devidamente formatados dentro de uma estrutura do tipo dados_arq definida na biblioteca simulador.h.

Parâmetros de entrada:

char *linha – ponteiro para uma linha do arquivo.

Parâmetros de saída:

struct dados_arq *d – ponteiro para uma estrutura do tipo dados_arq devidamente carregada.

Retorno: void

Como o simulador utiliza a rotina:

Ao iniciar o simulador, ele irá verificar o arquivo informado como portador das regras de configuração. O simulador irá ler cada linha e chamar a rotina monta_inf() passando a linha lida como parâmetro. Após algumas validações e utilizando quebras na linha recebida, o simulador irá montar todas as variáveis existentes na linha em uma estrutura do tipo dados_arq (localizada na biblioteca simulador.h).

5.2.5.2 Função quebra_protocolo()

A função quebra_protocolo tem a função de tratar e separar o nome do protocolo lido pelo arquivo de configuração separando o nome e o tipo de protocolo. Por exemplo, ao ler o arquivo, o simulador capturou o nome de protocolo ICMP:ECHO. O simulador irá receber um ponteiro para este nome e irá retornar duas variáveis, uma com o nome ICMP e outra com o tipo ECHO. No caso do protocolo não possuir um tipo, o simulador irá retornar null na variável *c_tp_proto e o próprio nome recebido na variável *c_proto na variável *c_nm_proto.

Parâmetros de entrada:

char *c_proto – nome do protocolo lido no arquivo de configuração

Parâmetros de saída:

char *c_nm_proto – nome do protocolo

*c_tp_proto – tipo do protocolo (no caso de protocolo ICMP)

Retorno:

void

Como o simulador utiliza a rotina:

Após realizar o tratamento das camadas de enlace, rede, o simulador irá chamar a função quebra_protocolo para realizar o tratamento do nome lido no arquivo de configuração e poder tratar o protocolo da camada de transporte corretamente. Após chamar a rotina, o simulador terá o nome do protocolo puro (sem o tipo) e poderá decidir qual tipo de protocolo irá tratar na camada de transporte.

5.2.5.3 Função converte_porta()

Recebe uma porta no formato do arquivo de configuração de regras e a transforma em um formato int (inteiro) retornando também um indicador dizendo qual o tipo de tratamento a ser feito.

Parâmetros de entrada:

char *porta – porta lida no arquivo de configuração

Parâmetros de saída:

struct u_short *in_porta – indicador de tratamento da porta lida. Valores válidos:

0 – tratamento normal (Verificar se porta do pacote é igual a porta do arquivo de configuração)

1 – tratamento menor (A porta lida possui um sinal de menor [<])

2 – tratamento maior (A porta lida possui um sinal de maior [>])

struct u_short *u_porta – número da porta após extraído o sinal

Retorno:

void

Como o simulador utiliza a rotina:

No momento da validação da(s) porta(s) de origem/destino, o simulador irá chamar a rotina converte_porta() para extrair o sinal do número (caso houver) e mover o número da porta para uma variável inteira que será comparada com o número da porta lido existente no pacote TCP/UDP lido.

5.2.6 Módulo de validação do pacote

Este módulo possui o nome pgm-valida.c e possui as funções que realizam toda a validação necessária do pacote lido com as regras existentes no arquivo de configuração de regras do simulador de filtragem de pacotes.
O código fonte do módulo pgm-valida.c pode ser visualizado no ANEXO E.

As funções definidas dentro deste módulo são definidas a seguir juntamente com seus parâmetros e especificações.

5.2.6.1 Função valida_endereço()

O objetivo desta função é verificar se o endereço IP do pacote lido é igual ao endereço IP de uma linha do arquivo de configuração de regras.

Parâmetros de entrada:

char *c_end_valida – endereço IP lido no arquivo de configuração

struct in_addr *end_pacote – endereço IP do pacote lido

Parâmetros de saída:

void

Retorno:

0 – endereço IP não deve ser verificado (está preenchido com ‘-‘) ou o endereço IP do pacote lido é igual ao endereço IP do arquivo de configuração. O endereço IP se enquadra na regra.

1 – o endereço IP do pacote lido é diferente do endereço do arquivo de configuração. O endereço IP não se enquadra na regra.

Como o simulador utiliza a rotina:

O simulador chama a rotina valida_endereço() passando o endereço IP de origem ou destino do pacote lido juntamente com o endereço IP de origem ou destino do arquivo de configuração de regras. A rotina irá verificar os parâmetros

retornando se o pacote se enquadra (retorno 0) ou não (retorno 1) em uma determinada regra.

5.2.6.2 Função valida_protocolo()

O objetivo desta função é verificar se o protocolo de camada de rede ou transporte do pacote lido é igual ao protocolo de camada de rede ou transporte do arquivo de configuração de regras.

Parâmetros de entrada:

char *proto – nome do protocolo no arquivo de configuração de regras

u_int8_t *ip_proto – número do protocolo de camada de rede (de acordo com a RFC 791).

u_int8_t *tp_icmp – no caso do protocolo informado for ICMP, este campo deve conter o tipo da mensagem ICMP. De acordo com a RFC 792 e com a biblioteca netinet/in.h).

Parâmetros de saída:

void

Retorno:

0 – o protocolo/tipo não deve ser verificado (está preenchido com ‘-‘) ou o protocolo/tipo do pacote lido é igual ao protocolo do arquivo de configuração. O protocolo/tipo se enquadra na regra.

1 – o protocolo/tipo do pacote lido é diferente do protocolo/tipo do arquivo de configuração. O protocolo/tipo não se enquadra na regra.

2 – erro na função

Como o simulador utiliza a rotina:

O simulador chama a rotina valida_protocolo() passando o protocolo e o tipo do protocolo (no caso de protocolo ICMP) do pacote lido juntamente com o protocolo e tipo do protocolo do arquivo de configuração de regras. A rotina irá verificar os parâmetros retornando se o pacote se enquadra (retorno 0) ou não (retorno 1) em uma determinada regra ou se por acaso acontecer algum erro na função (retorno 3).

5.2.6.3 Função valida_porta()

O objetivo desta função é verificar se o porta TCP/UDP do pacote lido é igual a porta TCP/UDP de uma linha do arquivo de configuração de regras.

Parâmetros de entrada:

char *c_porta_valida – porta TCP/UDP lida no arquivo de configuração

struct u_int16_t – porta TCP/UDP do pacote lido

Parâmetros de saída:

void

Retorno:

0 – a porta TCP/UDP não deve ser verificada (está preenchida com '-') ou a porta TCP/UDP do pacote lido é igual a porta TCP/UDP do arquivo de configuração. A porta TCP/UDP se enquadra na regra.

1 – a porta TCP/UDP do pacote é diferente da porta TCP/UDP do arquivo de configuração. A porta TCP/UDP não se enquadra na regra.

Como o simulador utiliza a rotina:

O simulador chama a rotina valida_porta() passando a porta TCP/UDP de origem ou destino do pacote lido juntamente com a porta TCP/UDP de origem ou destino do arquivo de configuração de regras. A rotina irá verificar os parâmetros retornando se o pacote se enquadra (retorno 0) ou não (retorno 1) em uma determinada regra ou se por acaso acontecer algum erro na função (retorno 3).

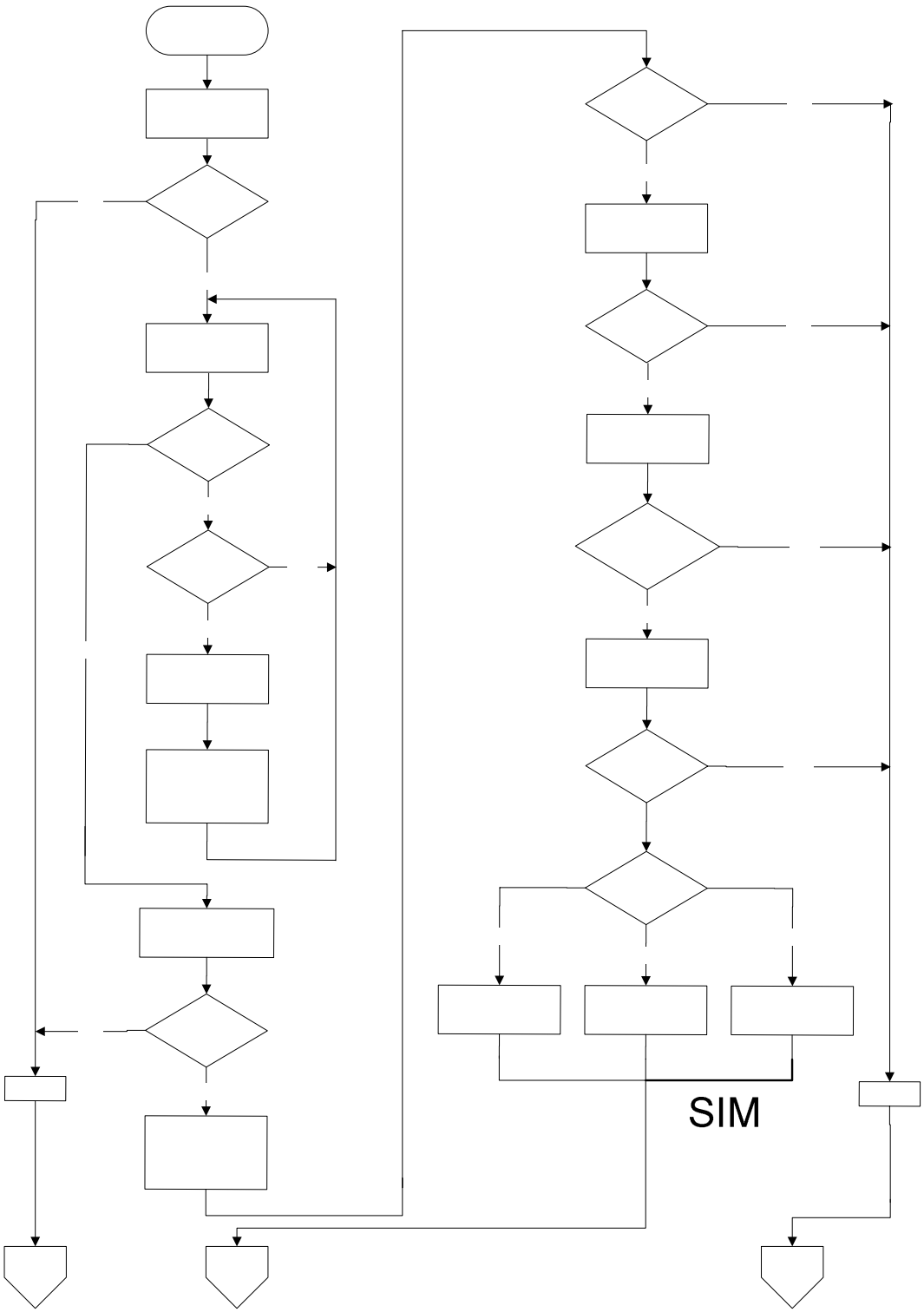
5.2.7 Módulo principal

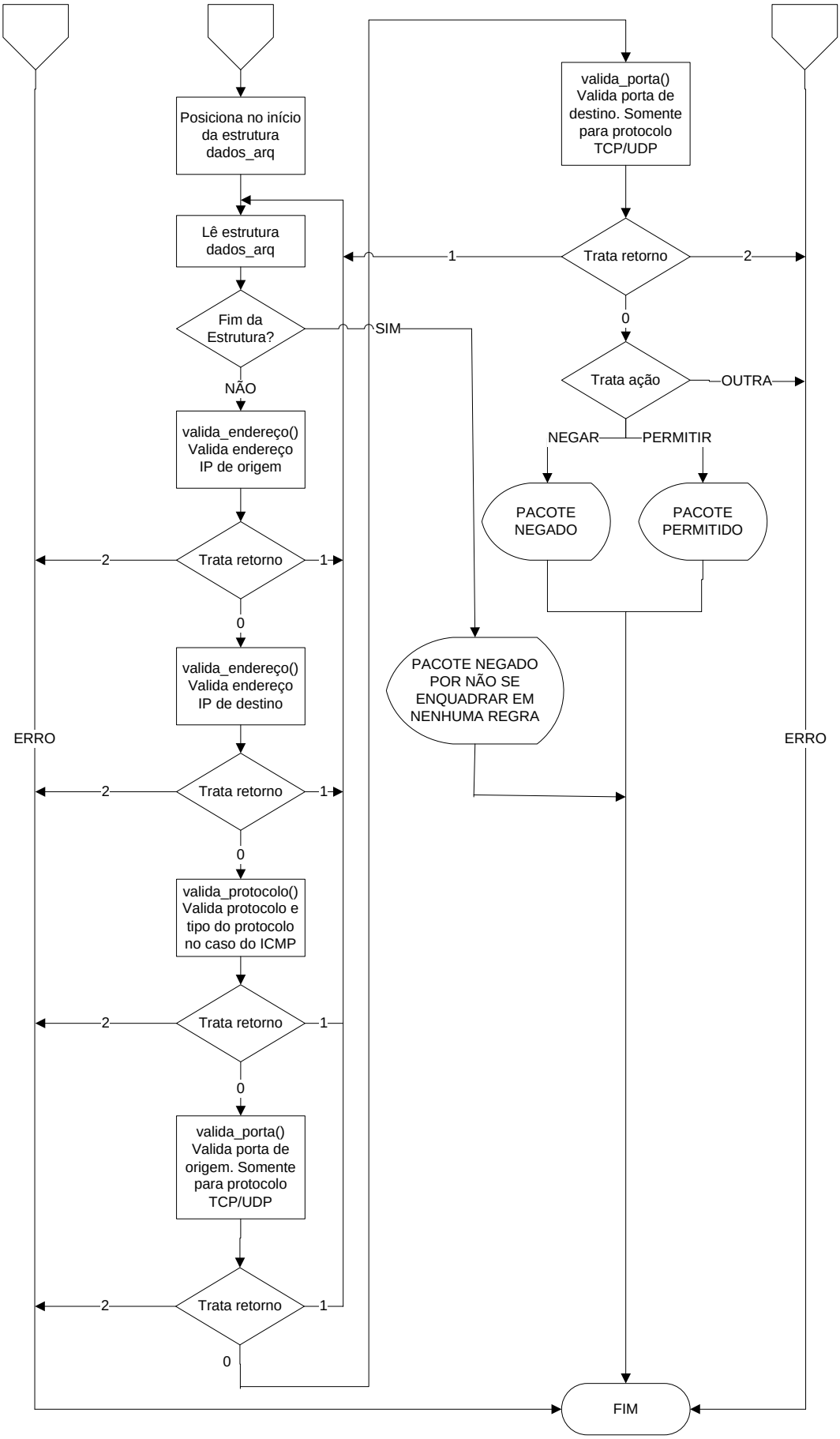
O módulo principal é definido pelo programa pgm-principal.c e é apresentado no ANEXO F. A principal função do programa pgm-principal.c é realizar toda a coordenação do simulador de filtragem de pacotes. É ele quem efetua todas as chamadas às principais funções, trata os retornos e finaliza o programa quando necessário.

No início do módulo, ele carrega todas as variáveis, estruturas e funções necessárias, chama as rotinas de validação e tratamento do arquivo de configuração de regras, captura o pacote a ser validado, trata os cabeçalhos das camadas de enlace, rede e transporte e chama as rotinas de validação do pacote.

Após realizar todas essas funções e tendo as variáveis de retorno necessárias, o programa principal realiza a decisão se o pacote deve ou não ser proibido.

5.2.8 Fluxograma do simulador




Posiciona no início da estrutura dados_arq
Lê estrutura dados_arq
Fim da Estrutura?
SIM
NÃO
valida_endereço() Valida endereço IP de origem
Trata retorno
2
1
0
ERRO
valida_endereço() Valida endereço IP de destino
Trata retorno
2
1
0
ERRO
valida_protocolo() Valida protocolo e tipo do protocolo no caso do ICMP
Trata retorno
2
1
0
valida_porta() Valida porta de origem. Somente para protocolo TCP/UDP
Trata retorno
2
1
0
valida_porta() Valida porta de destino. Somente para protocolo TCP/UDP
Trata retorno
1
2
0
Trata ação
OUTRA
NEGAR
PERMITIR
PACOTE NEGADO
PACOTE PERMITIDO
PACOTE NEGADO POR NÃO SE ENQUADRAR EM NENHUMA REGRA
FIM

Conclusão

O principal objetivo deste trabalho foi procurar reunir de forma estruturada e organizada, grande parte dos pontos importantes que devem ser observados ao se projetar um ambiente de segurança para ambientes conectados à Internet.

A grande contribuição deste trabalho foi o de realizar um estudo integrado sobre as questões de vulnerabilidades, ataques e formas de se proteger, já que grande parte dos estudos sobre estes pontos são realizados de forma separadas não dando ao leitor na maioria das vezes um entendimento sobre algumas das relações importantes entre determinados tipos de vulnerabilidades e ataques com as formas de se defender.

Este estudo também foi realizado em um momento oportuno devido a enorme repercussão que os problemas de segurança tem gerado nos dias de hoje devido a enorme quantidade de serviços que hoje são disponibilizados via web, em grande parte graças ao comércio eletrônico.

Ferramentas de segurança como firewall e IDS podem ser utilizados de forma a constituir uma barreira sólida para aumentar a segurança das organizações e devem ser utilizados quando possível. Porém, para se obter as reais vantagens sobre estas ferramentas, é necessário treinamento adequado para que a estas ferramentas não se tornem apenas mais pontos de brechas na segurança.

Um outro fator de grande importância, é a questão da conformidade que deve ser tomada como um papel fundamental durante a elaboração de políticas segurança principalmente depois que a lei de diretrizes de segurança da informação que está tramitando no congresso seja votada, o que dará um grande passo para que se possam criar políticas de segurança mais elaboradas e fundamentadas em leis civis e criminais.

Em suma, pode-se dizer que todo o estudo sobre segurança da informação para ambientes conectados à Internet visa minimizar a níveis aceitáveis a exposição destes ambientes frente a enorme variedade e ocorrência de ataques nos dias de hoje seja devido as questões de falhas nos aplicativos, softwares e sistemas operacionais desenvolvidos, seja pela grande quantidade de ferramentas de prospecção e ataques hoje existentes.

Trabalhos Futuros

O simulador de filtragem de pacotes desenvolvido como validação de parte do estudo sobre segurança em ambientes conectados à Internet realizado neste trabalho possui um enorme potencial de crescimento.

É um projeto que pode agregar funções e valores de grande importância para a comunidade acadêmica e comercial visto que várias empresas ainda necessitam melhorar a segurança de seus ambientes frente as ameaças da Internet.

Pontos que podem gerar futuras pesquisas e desenvolvimento em cima do simulador de filtragem de pacotes são:

- O desenvolvimento de uma interface gráfica mais amigável para usuários, de forma que o uso de sistemas de filtragem (firewalls) possa ser melhor disseminado e popularizado.
- A adequação do simulador para que o mesmo possa aceitar uma maior quantidade de protocolos e funções (como por exemplo fragmentação de pacotes)
- A inclusão de funções como validação de máscaras de rede e até mesmo estados das conexões

ANEXO A – Tabela de Impressões Digitais (fingerprint)

Tamanhos Iniciais de Janelas – Window Size

Sistema Operacional	Tamanho da Janela Inicial
Linux 2.4.1-14	5840
Solaris 2.6 or 2.7	8760
SunOS 5.8	24820
Linux 2.2.x	32120
Mac OS X	32768

Tabela 1 - Tamanho Inicial de Janelas

Tempo de Vida – Time do Live (TTL)

Sistema Operacional	Tempo de Vida
Solaris 2.6 or 2.7, Cisco IOS	255
Windows 2000	128
Linux 2.0, 2.2, 2.4	64
Windows CE 3.0 (Ipaq 3670)	32

Tabela 2 - Tempo de Vida

Tamanho Máximo do Segmento – Maximum Segment Size (MSS)

Sistema Operacional	Tamanho Máximo do Segmento
Windows 9x	536
Alcatel (Xylan) OmniStack 5024	1024
Windows NT, Mac OS9, BSD, Linux	1460

Tabela 3 - Tamanho Máximo do Segmento

Flag Não Fragmentar – Don't Fragment Flag (DF)

Sistema Operacional	Flag Não Fragmentar
Cisco IOS 2611/11.3(2)XA4, C2600/12.0(5)T1, 4500/12.0(9), 3640/12.1(2), 3620/12.0(8)	0
Linux 2.2.19	0
CacheOS 3.1 on a CacheFlow 6000	0
Windows XP Pro, Windows 2000 Pro	1
Windows 9x or NT4	1
Mac OS 7.x-9.x	1

Tabela 4 - Flag Não Fragmentar

ANEXO B – A biblioteca simulador.h

```
// simulador.h

// Definicoes de variaveis a serem utilizadas no simulador
// de filtragem de pacotes

// Leonardo Barbosa de Andrade - R.A: 996608/0
// Centro Universitario de Brasilia - UniCEUB
// Projeto Final do Curso de Engenharia de Computacao

#include <stdio.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <pcap.h>

#define TAM_END_ALFA        16
#define TAM                 128

struct dados_arq {
                    char c_end_ogm[TAM_END_ALFA];
                    char c_end_dst[TAM_END_ALFA];
                    char c_proto[TAM_END_ALFA];
                    char c_porta_ogm[TAM_END_ALFA];
                    char c_porta_dst[TAM_END_ALFA];
                    char c_acao[TAM_END_ALFA];
                    u_int nr_regra;
                    struct dados_arq *proximo;
};

u_int8_t tp_icmp_pacote;

u_int16_t porta_ogm_pacote, porta_dst_pacote;

struct in_addr iend_ogm, iend_dst;

char *dev,errbuf[PCAP_ERRBUF_SIZE];
pcap_t *descr;
const u_char *pacote;
struct pcap_pkthdr hdr;
struct ether_header *eptr;
struct ip *ip;
struct icmp *icmp;
struct tcphdr *tcp;
struct udphdr *udp;

int tam_hdr_ether,tam_hdr_ip;

u_char *ptr;
```

ANEXO C – Arquivo de configuração de regras (exemplo)

```
# ARQUIVO DE CONFIGURAÇÃO DE REGRAS
#
#
#          ENDEREÇO IP                                    PORTA
# ORIGEM          DESTINO          PROTOCOLO        ORIGEM          DESTINO          AÇÃO
-------------------------------------------------------------------------------------------------------------------------------
 10.10.1.1        -                TCP              -               -                NEGAR
-                 -                ICMP:REDIRECT    -               -                PERMITIR
-                 -                TCP              -               80               PERMITIR
-                 -                TCP              -               >1024            NEGAR
```

ANEXO D – formata-arq.c

```
// formata-arq.c

// Leonardo Barbosa de Andrade - R.A: 996608/0
// Centro Universitario de Brasilia - UniCEUB
// Projeto Final do curso de Engenharia de Computacao

// funcoes de tratamento e formatacao do arquivo de regras
// do simulador

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "simulador.h"
#include <netinet/in.h>
#include <ctype.h>

#ifndef TAM
#define TAM                     128
#endif
#ifndef TAM_END_ALFA
#define TAM_END_ALFA            16
#endif


// Recebe com entrada uma linha do arquivo de configuracao e
// realiza a formatacao desta linha retornando os valores
// lidos nas respectivas variaveis.

void monta_inf (char *linha, struct dados_arq *d)  {

        u_short i,j,qt_inf;
        char str_aux[TAM_END_ALFA];

        j=0;    qt_inf=0;
        for (i=0;i<TAM;i++) {
                if (linha[i] != ' ' && (!iscntrl(linha[i]))) {
                        str_aux[j] = linha[i];
                        j++;
                }
                str_aux[j]=NULL;
                if ((j==15) || (i==TAM-1) || (linha[i+1] == ' ')
                        || (linha[i+1] == NULL)) {
                        str_aux[j] = NULL;
                        j=0;
                        if (str_aux[0] != NULL) {
                                switch (qt_inf) {
                                        case 0: {
                                                strcpy(d->c_end_ogm,str_aux);
                                                break;
                                        }
                                        case 1: {
                                                strcpy(d->c_end_dst,str_aux);
                                                break;
```

```
                                        }
                                        case 2: {
                                                strcpy(d->c_proto,str_aux);
                                                break;
                                        }
                                        case 3: {
                                                strcpy(d->c_porta_ogm,str_aux);
                                                break;
                                        }
                                        case 4: {
                                                strcpy(d->c_porta_dst,str_aux);
                                                break;
                                        }
                                        case 5: {
                                                strcpy(d->c_acao,str_aux);
                                                break;
                                        }
                                }
                                if (qt_inf == 5)
                                        break;
                                qt_inf++;
                        }
                }
        }
}

// Recebe a descricao completa do protocolo e separa em nome
// do protocolo e tipo do protocolo
void quebra_protocolo(char *proto, char *nm_proto, char *tp_proto) {

        u_short ic_proto=0,ic_proto_aux=0,flag=0;
        char nm_proto_aux[TAM_END_ALFA];

        nm_proto[0] = NULL;
        tp_proto[0] = NULL;

        for (ic_proto=0;ic_proto<TAM_END_ALFA;ic_proto++) {
                if (proto[ic_proto] != NULL && proto[ic_proto] != ' ') {
                        if (proto[ic_proto] != ':') {
                                nm_proto_aux[ic_proto_aux] = proto[ic_proto];
                                ic_proto_aux++;
                                nm_proto_aux[ic_proto_aux] = NULL;
                        } else {
                                strcpy(nm_proto,nm_proto_aux);
                                flag++;
                                ic_proto_aux=0;
                                nm_proto_aux[ic_proto_aux] = NULL;
                        }
                }
                if (!proto[ic_proto]) {
                        nm_proto_aux[ic_proto_aux] = NULL;
                        if (flag == 0)
                                strcpy(nm_proto,nm_proto_aux);
                        else
                                strcpy(tp_proto,nm_proto_aux);
                        break;
                }
        }
}
```

```
// Recebe uma porta no formato do arquivo de configuracao de regras e a
// transforma as formato int retornando um indicador dizendo qual o tipo
// de tratamento a ser feito (nenhum, maior que porta ou menor que porta)
void converte_porta
(char *porta, u_short *in_porta, u_short *u_porta) {

        u_short i=0,j=0;
        char str_aux[TAM_END_ALFA];

        str_aux[0]   = NULL;
        *in_porta = 0;

        if (porta[0] == '-') {
                *in_porta = 0;
                *u_porta  = 0;
        }
        else {
                for (i=0;i<TAM_END_ALFA;i++) {
                        if (porta[i] == NULL)
                                break;
                        if ((porta[i] != ' ') && (porta[i] != '<')
                                && (porta[i] != '>')) {
                                str_aux[j] = porta[i];
                                j++;
                        }
                        else {
                                switch(porta[i]) {
                                        case '<': {
                                                *in_porta = 1;
                                                break;
                                        }
                                        case '>': {
                                                *in_porta = 2;
                                                break;
                                        }
                                        default: {
                                                *in_porta = 0;
                                                break;
                                        }
                                }
                        }
                }
                str_aux[j] = NULL;
        }
        if (str_aux[0] != NULL)
                *u_porta = atoi(str_aux);

}
```

ANEXO E – pgm-valida.c

```
// pgm-valida.c

// Leonardo Barbosa de Andrade - R.A: 996608/0
// Centro Universitario de Brasilia - UniCEUB
// Projeto Final do Curso de Engenharia de Computacao

// Funcoes de validacao do pacote lido com as regras
// existentes no arquivo de configuracao

// Funcao de validacao do protocolo de camada de
// transporte utilizado pelo datagrama IP

#include <netinet/in.h>
#include <netinet/ip_icmp.h>
#include <string.h>

#ifndef TAM_END_ALFA
#define TAM_END_ALFA 16
#endif

// Funcao de validacao de endereco de rede
u_short valida_endereco
(char *c_end_valida, struct in_addr *end_pacote) {

        struct in_addr iend_aux;

        if (c_end_valida[0] == '-')
                return(0);
        else if (inet_aton(c_end_valida,&iend_aux)) {
                if (iend_aux.s_addr == end_pacote->s_addr)
                        return(0);
                else
                        return(1);
        }

}

// Funcao de validacao do protocolo
u_short valida_protocolo(char *proto, u_int8_t *ip_proto, u_int8_t *tp_icmp) {

        u_int8_t nr_proto=0,tp_proto_icmp=0;
        char nm_proto[TAM_END_ALFA], tp_proto[TAM_END_ALFA];

        if (proto[0] == '-')
                return(0);

        quebra_protocolo(proto,nm_proto,tp_proto);

        if (!strcmp(nm_proto,"ICMP"))
                nr_proto = IPPROTO_ICMP;
        else if (!strcmp(nm_proto,"TCP"))
                nr_proto = IPPROTO_TCP;
        else if (!strcmp(nm_proto,"UDP"))
```

```
		nr_proto = IPPROTO_UDP;
	else {
		printf ("Modulo pgm-valida.c\tFuncao valida_protocolo()\n");
		printf ("Protocolo %s desconhecido\n",nm_proto);
		return(2);
	}

	if (nr_proto == ip_proto) {
		if (nr_proto != IPPROTO_ICMP)
			return(0);
		else {
			if (!tp_proto[0])
				return(0);
			if (!strcmp(tp_proto,"ECHOREPLY"))
				tp_proto_icmp = ICMP_ECHOREPLY;
			else if (!strcmp(tp_proto,"DEST_UNREACH"))
				tp_proto_icmp = ICMP_DEST_UNREACH;
			else if (!strcmp(tp_proto,"SOURCE_QUENCH"))
				tp_proto_icmp = ICMP_SOURCE_QUENCH;
			else if (!strcmp(tp_proto,"REDIRECT"))
				tp_proto_icmp = ICMP_REDIRECT;
			else if (!strcmp(tp_proto,"ECHO"))
				tp_proto_icmp = ICMP_ECHO;
			else if (!strcmp(tp_proto,"TIME_EXCEEDED"))
				tp_proto_icmp = ICMP_TIME_EXCEEDED;
			else if (!strcmp(tp_proto,"PARAMETERPROB"))
				tp_proto_icmp = ICMP_PARAMETERPROB;
			else if (!strcmp(tp_proto,"TIMESTAMP"))
				tp_proto_icmp = ICMP_TIMESTAMP;
			else if (!strcmp(tp_proto,"TIMESTAMPREPLY"))
				tp_proto_icmp = ICMP_TIMESTAMPREPLY;
			else if (!strcmp(tp_proto,"INFO_REQUEST"))
				tp_proto_icmp = ICMP_INFO_REQUEST;
			else if (!strcmp(tp_proto,"INFO_REPLY"))
				tp_proto_icmp = ICMP_INFO_REPLY;
			else if (!strcmp(tp_proto,"ADDRESS"))
				tp_proto_icmp = ICMP_ADDRESS;
			else if (!strcmp(tp_proto,"ADDRESSREPLY"))
				tp_proto_icmp = ICMP_ADDRESSREPLY;
			else {
				printf ("Modulo pgm-valida.c\tFuncao valida_protocolo()\n");
				printf ("Tipo icmp: %s desconhecido\n",tp_proto);
				return(2);
			}
			if (tp_proto_icmp == tp_icmp)
				return(0);
			else
				return(1);
		}
	} else
		return(1);

}

// Funcao de validacao de portas
u_short valida_porta(char *c_porta_valida, u_int16_t *porta_pacote) {

	u_short in_porta=0,u_porta=0;

	if (c_porta_valida[0] == '-')
```

```
                return(0);

        converte_porta(c_porta_valida,&in_porta,&u_porta);

        if ((in_porta == 0 && u_porta == porta_pacote) ||
           (in_porta == 1 && u_porta >  porta_pacote) ||
           (in_porta == 2 && u_porta <  porta_pacote))
                return(0);
        else
                return(1);

}
```

ANEXO F – pgm-principal.c

```
// pgm-principal.c

// Leonardo Barbosa de Andrade - R.A: 996608/0
// Centro Universitario de Brasilia - UniCEUB
// Projeto Final do Curso de Engenharia de Computacao

// programa principal do simulador. realiza todo o controle
// necessario.

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <pcap.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/if_ether.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/tcp.h>
#include <netinet/udp.h>
#include <net/ethernet.h>
#include "simulador.h"

int main(int argc, char **argv) {

        FILE *arq;

        char linha[TAM];
        struct dados_arq inicio, *node;

        u_short ret_valida,nr_regra_aux=0,pkt_ok=0;
        char acao_aux[TAM_END_ALFA];

// ------------------------------------------------------------ Tratamento do arquivo de regras

        if ((arq = fopen(argv[1],"r")) == NULL) {
                printf ("Erro ao tentar ler %s.\n",argv[1]);
                printf ("Programa pgm-principal.c\tFuncao int main()\n");
                exit(1);
        }
        inicio.proximo=NULL;
        node=&inicio;
        nr_regra_aux=1;
        while (fgets(linha, sizeof(linha), arq)) {
                if (linha[0] == ' ') {
                        node->proximo=(struct dados_arq *)malloc(sizeof(struct dados_arq));
                        node->nr_regra=nr_regra_aux;
                        nr_regra_aux++;
                        monta_inf(linha,node);
                        node=node->proximo;
                        node->proximo=NULL;
```

```
		}
	}

// ------------------------------------------------- Tratamento do dispositivo de captura de pacotes

		// Descobrindo o dispositivo que ira capturar os pacotes
	dev = pcap_lookupdev(errbuf);

	if (dev==NULL) {
		printf ("Erro ao tentar capturar dispositivo.\n");
		printf ("Programa pgm-principal.c\tFuncao int main()");
		printf ("%s\n",errbuf);
		exit(1);
	}

		// Abre o dispositivo para captura
	descr = pcap_open_live(dev,BUFSIZ,0,-1,errbuf);

	if (descr == NULL) {
		printf ("Erro ao tentar abrir o dispositivo %s\n",dev);
		printf ("pcap_open_live(): %s\n",errbuf);
		printf ("Programa pgm-principal.c\tFuncao int main()\n");
		exit(1);
	}

	pacote = pcap_next(descr,&hdr);

	if (pacote == NULL) {
		printf ("Nao foi possivel capturar pacote no dispositivo %s\n",dev);
		printf ("Programa pgm-principal.c\tFuncao int main()\n");
		exit(1);
	}

// ------------------------------------------------------- Tratamento do header da camada de enlace


	eptr = (struct ether_header *) pacote;


	if (ntohs(eptr->ether_type) != ETHERTYPE_IP) {
		printf ("Pacote recebido nao e um pacote IP\n");
		printf ("Simulador nao trabalha com pacote ethernet tipo: %x\n",ntohs(eptr-
>ether_type));
		exit(1);
	}

// ------------------------------------------------------- Tratamento do header da camada de rede

	ip = (struct ip *)(pacote + sizeof(struct ether_header));
	tam_hdr_ip = sizeof(struct ip);


		// Checagem da versao do IP
	if (ip->ip_v != 4) {
		printf ("Versao do protocolo IP: %u desconhecida.\n",ip->ip_v);
		printf ("Programa pgm-principal.c\tFuncao int main()\n");
		exit(1);
	}

		// Checagem do tamanho do header
```

```
	if (ip->ip_hl < 5) {
		printf ("Tamanho do header IP %u invalido.\n",ip->ip_hl);
		printf ("Programa pgm-principal.c\tFuncao int main()\n");
		exit(1);
	}


//  ---------------------- Tratamento do header da camada de transporte --------------------------------

		tp_icmp_pacote=0;
		switch(ip->ip_p) {
			case 1: {
				icmp = (struct icmp *)
				(pacote + sizeof(struct ether_header) + sizeof(struct ip));
				tp_icmp_pacote = icmp->icmp_type;
				break;
			}
			case 6: {
				tcp = (struct tcphdr *)
				(pacote + sizeof(struct ether_header) + sizeof(struct ip));
				porta_ogm_pacote = tcp->source;
				porta_dst_pacote = tcp->dest;
				break;
			}
			case 17: {
				udp = (struct udphdr *)
				(pacote + sizeof(struct ether_header) + sizeof(struct ip));
				porta_ogm_pacote = udp->source;
				porta_dst_pacote = tcp->dest;
				break;
			}
			default:
				break;

		}


// ------------------------------------------- Validacao do pacote lido com as regras do arquivo

	nr_regra_aux=0;
	node = &inicio;
	do {

		ret_valida=0;

		// Valida endereco IP de origem
		if (!ret_valida) {
			ret_valida = valida_endereco(node->c_end_ogm, ip->ip_src);
			if (ret_valida > 1)
				exit(1);
		}

		// Valida endereco IP de destino
		if (!ret_valida) {
			ret_valida = valida_endereco(node->c_end_dst, ip->ip_dst);
			if (ret_valida > 1)
				exit(1);
		}

		// Valida protocolo
```

```
		if (!ret_valida) {
			ret_valida = valida_protocolo(node->c_proto,ip->ip_p,tp_icmp_pacote);
			if (ret_valida > 1)
				exit(1);
		}

		// Valida porta de origem
		if (!ret_valida && ip->ip_p != 1) {
			ret_valida = valida_porta(node->c_porta_ogm,porta_ogm_pacote);
			if (ret_valida > 1)
				exit(1);
		}

		// Valida porta de destino
		if (!ret_valida) {
			ret_valida = valida_porta(node->c_porta_dst,porta_dst_pacote);
			if (ret_valida > 1)
				exit(1);
		}

		if (!ret_valida) {
			nr_regra_aux=node->nr_regra;
			strcpy(acao_aux,node->c_acao);
			if (!strcmp(acao_aux,"NEGAR"))
				pkt_ok=1;
			else if (!strcmp(acao_aux,"PERMITIR"))
				pkt_ok=2;
			else {
				printf ("A acao informada e desconhecida: %s\n",node->c_acao);
				printf ("Modulo pgm-principal.c\tFuncao int main()\n");
				pkt_ok=3;
			}
			break;
		}

		node = node->proximo;

	} while (node->proximo);

	if (pkt_ok==0) {
		printf ("A T E N C A O:  O PACOTE LIDO FOI DESCARTADO\n");
		printf ("		POR NAO SE ENQUADRAR EM NENHUMA REGRA\n");
	} else if (pkt_ok==1) {
		printf ("A T E N C A O:  O PACOTE LIDO FOI NEGADO\n");
		printf ("	PELA REGRA NUMERO %d do arquivo %s\n",nr_regra_aux,argv[1]);
	} else if (pkt_ok==2) {
		printf ("A T E N C A O:  O PACOTE LIDO FOI ACEITO\n");
		printf ("	PELA REGRA NUMERO %d do arquivo %s\n",nr_regra_aux,argv[1]);
	} else if (pkt_ok==3)
		return(1);

	fclose(arq);
	return(0);

}
```

Referências Bibliográficas

[1] Martins, Alessandro - A relação da Internet com o Comércio Eletrônico no Brasil e no mundo e os próximos acontecimentos

[2] CBPF-NT-005/01 – Internet, Evolução e Gestão
Centro Brasileiro de Pesquisas Físicas – http://www.cbpf.br

[3] Internet Software Consortium – http://www.isc.com, setembro/2002

[4] Sans home page - System Administration, Networking and Security
http://www.sans.org/top20/portuguese_v2.php

[5] Queiroz, Rodolfo Michel de Lima – Protegendo-se da Internet

[6] Júnior, Eurípedes Laurindo Lopes – Estudo e Análise sobre Prospecções Indevidas em Redes de Computadores.

[7] Fiddler, Matthew – Intrusion Detection in Depth
GCIA Practical Assignment, Version 3.0

[8] Symantec
http://www.symantec.com/region/br/enterprisesecurity/content/article2160.html

[9] Cert Coordination Center - http://www.cert.org

[10] Norma BS 7799 – Normatização de Práticas para Gerenciamento da Segurança da Informação

[11] Building Internet Firewalls
Zwicky, Elizabeth D. / Cooper, Simon / Chapman, Brent D. O'Reilly & Associates

[12] Bradley, Tony – Introduction to Intrusion Detection Systems (IDS)

[13] Internetworking with TCP/IP. Vol II: Design, Implementation and Internals
Comer, Douglas E. / Steve, David L. Prentice Hall PTR

[14] Bruno d4t – Libpcap: A biblioteca para Análise de Rede

Bibliografia

Security in Computing. 2nd edition
Pfleeger, Charles P. — Prentice Hall PTR

Internetworking with TCP/IP. Vol I: Principles, Protocols and Architecture
Comer, Douglas E. / Steve, David L. — Prentice Hall PTR

Computer Security Basics
Russel, Debby / Gangemi, G.T. Sr — O'Reilly &Associates

Pratical Unix & Internet Security. 3rd edition
Garfinkel, Simson / Spafford, Gene / Schwartz, Alan — O'Reilly & Associates

Complete Encyclopedia of Networking — Novell Press

Segurança e Prevenção em Redes
Northcutt, Stephen / Novak, Judy / McLachlan, Donald — Editora Berkeley

Hackers – Acesso Negado
Cronkhite, Cathy / McCullough, Jack — Editora Campus

Segurança Máxima – 3ª edição - SAMS
Autor Anônimo — Editora Campus

Hackers Expostos – 4ª edição
McClure, Stuart / Scambray, Joel / Kurte

C Completo e Total
Schildt, Herbert — Makron Books

1001 Visual C++ Programming Tips
Jamsa, Kris A

http://www.ieee802.org/3/ - Página da IEEE que possui informações sobre o padrão IEEE 802

RFC 2196 – Site Security Handbook
<http://www.faqs.org/rfcs/rfc2196.htm> Acesso em: 21/08/2003

RFC 791 – Internet Protocol
<http://www.faqs.org/rfcs/rfc791.html> Acesso em: 13/04/2004

RFC 793 – Transmission Control Protocol
<http://www.faqs.org/rfcs/rfc793.html> Acesso em: 18/06/2004

RFC 768 – User Datagram Protocol
<http://www.faqs.org/rfcs/rfc768.html> Acesso em: 18/06/2004

RFC 792 – Internet Control Message Protocol
<http://www.faqs.org/rfcs/rfc792.html> Acesso em: 22/06/2004