Guilherme Gonsales Panes

Firewall Dinâmico: Uma implementação Cliente/Servidor

São José do Rio Preto
2011

Guilherme Gonsales Panes

Firewall Dinâmico: Uma implementação Cliente/Servidor

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração “Sistemas de Computação” na linha de pesquisa “Arquitetura de Computadores e Sistemas Distribuídos”, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Adj. Marcos Cavenaghi
Co-orientadora: Profa. Adj. Roberta Spolon

São José do Rio Preto
2011

Panes, Guilherme Gonsales.
Firewall dinâmico : uma implementação cliente/servidor / Guilherme Gonsales Panes. – São José do Rio Preto: [s.n.], 2012.
71 f. : il. ; 30 cm.

Orientador: Marcos Cavenaghi
Coorientador: Roberta Spolon
Dissertação (mestrado) - Universidade Estadual Paulista, Instituto de Biociências, Letras e Ciências Exatas

1. Computação. 2. Redes de computadores – Medidas de segurança. 3. Firewalls. I. Cavenaghi, Marcos. II. Spolon, Roberta. III. Universidade Estadual Paulista, Instituto de Biociências, Letras e Ciências Exatas. IV. Título.

CDU – 004.7

Campus de São José do Rio Preto - UNESP

Guilherme Gonsales Panes

Firewall Dinâmico: Uma implementação Cliente/Servidor

Dissertação apresentada para obtenção do título de Mestre em Ciência da Computação Área de Concentração “Sistemas de Computação” na linha de pesquisa “Arquitetura de Computadores e Sistemas Distribuídos”, junto ao Programa de Pós- Graduação em Ciência da Computação do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Campus de São José do Rio Preto.

Banca Examinadora

Prof. Dr. Marcos Antonio Cavenaghi
Professor Adjunto
UNESP – Bauru
Orientador

Prof. Dr. Aparecido Nilceu Marana
Professor Doutor
UNESP – Bauru

Prof. Dr. Hélio Crestana Guardia
Professor Doutor
UFSCAR

São José do Rio Preto
29/Julho/2011

AGRADECIMENTOS

Agradeço a Deus, a minha querida esposa e a essa nova bênção que está chegando em nossas vidas.

Agradeço aos mestres Marcos Cavenaghi e Roberta Spolon que sempre me apoiaram em todo o trabalho.

Agradeço ao grande amigo e parceiro Marcelo Oliveiros pelo grande apoio neste projeto.

RESUMO

A proteção dos perímetros de segurança realizada através de firewalls somente é eficiente quando o cliente se encontra dentro do perímetro protegido por este firewall. Como mobilidade é um item essencial para as empresas, há um grande desafio na proteção destes clientes computacionais (Laptops e PDAs), pois é necessário aplicar a Política de Segurança da empresa independentemente de onde estes equipamentos estejam se conectando já que a segurança dos dados é fator essencial para garantia e continuidade dos negócios. Este trabalho tem como objetivo apresentar uma solução para este problema, de forma a utilizar ferramentas de firewall existentes, independente da plataforma e do software utilizado. Para tanto, desenvolveu-se um software baseado na arquitetura cliente/servidor, que analisa o ambiente em que o equipamento está conectado, através de um Intrusion Detection System (IDS), e baseado nestas informações recebe do Servidor Firewall um conjunto de regras para ser aplicado no firewall nativo do equipamento independentemente do sistema operacional utilizado. Desta forma é possível garantir que, independentemente do ambiente em que o equipamento esteja conectado, não se deixe de aplicar as regras contidas na Política de Segurança da corporação. O software foi desenvolvido em Java utilizando contents web visando portabilidade de plataforma e usabilidade para os administradores. Os testes desenvolvidos demonstram que o software cumpre o papel proposto de gerenciar as regras de firewall de forma coerente com o ambiente de rede conectada à máquina cliente.

Palavras-chave: *Firewall, Segurança, EndPoint Security*

ABSTRACT

The protection of the secure areas performed through firewalls is only effective when the client machine is inside the perimeter protected by the firewall. As mobility is an essential item for companies, there is a big challenge in protecting these mobile devices (Laptops and PDAs). It is necessary to apply the Security Policy company regardless of where these devices are to connecting as data security is an essential factor for securing and business continuity. This paper aims to propose a solution to this problem in order to use firewall tools existing, regardless of platform and application software. We have developed a software architecture based on a client / server approach, which analyzes the environment in which the equipment is connected using an Intrusion Detection System (IDS), and based Server receives this information a set of firewall rules for be applied on the native firewall system independent of the equipment operational use. This way it is possible to ensure that, regardless of the environment in which the equipment is connected, not be sure to apply the rules contained in the Security Policy corporation are always in effect. The software was developed in Java using web contents aiming platform portability, and usability for administrators. The performed tests show that the developed software meets the proposed role of managing the firewall rules consistent with the network environment connected to the client machine.

Key Words: Firewall, Security, EndPoint Security

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AES: Advanced Encryption Standard
API: Application Programming Interface
CERT: Computer Emergency Response Team
DDoS: Distributed Denial of Service
DES: Data Encryption Standard
DHCP: Dynamic Host Configuration Protocol
DMZ: Demilitarized Zone
DNS: Domain Name System
DoS: Denial of Service
EJB: Enterprise JavaBeans
FTP: File Transfer Protocol
GPL: GNU General Public License
GPRS: General Packet Radio Service
HTTP: Hypertext Transfer Protocol
HTTPS: Hypertext Transfer Protocol Secure
ICMP: Internet Control Message Protocol
IDS: Intrusion Detection System
IP: Internet Protocol
ISO: International Organization for Standardization
J2EE: Java 2 Enterprise Edition
J2SE: Java 2 Standard Edition
JSP: Java Server Pages
LDAP: Lightweight Directory Access Protocol
NIDS: Network Intrusion Detection System
OSI: Open Systems Interconnection
P2P: Peer-to-peer
PDA: Personal Digital Assistant
RC4: Rivest Cipher 4
RPC: Remote Procedure Call

SQL: Structured Query Language

SSL: Secure Socket Layer

TCP/IP: Transmission Control Protocol/ Internet Protocol

TCP: Transmission Control Protocol

TFTP: Trivial File Transfer Protocol

TFTP: Trivial File Transfer Protocol

TFTP: Trivial File Transfer Protocol

TIS: Trusted Information Systems

TSL: Transport Layer Security

UDP: User Datagram Protocol

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Como a competitividade entre as empresas vem aumentando a cada dia, houve a necessidade de que os ambientes computacionais evoluíssem rapidamente para um cenário em que desktops e notebooks podem estar conectados em qualquer lugar (*lan houses*, aeroportos, *cyber* cafés e etc.), usando as diversas tecnologias disponíveis (Wi-fi, GPRS, Cabeamento Estruturado, etc.). Essa mudança de contexto obrigatoriamente forçou a mudança da estratégia de segurança da maioria das organizações. Ao invés de despender esforços na garantia da segurança da rede de forma centralizada, a nova estratégia propõe prover maior segurança no host, onde quer que ele esteja conectado.

As soluções de *Firewall* atuais objetivam a proteção de uma arquitetura centralizada onde os hosts da rede possuem um papel passivo, e toda a garantia de segurança é centralizada nos *firewalls* localizados normalmente nas bordas dos perímetros de segurança, implementando regras genéricas sobre o tráfego da rede para aplicar os controles preconizados na Política de Segurança da empresa.

A grande inconveniência desta arquitetura é que os hosts não têm a capacidade de se autodefender, então todo tráfego que estiver dentro do mesmo perímetro de segurança poderá afetar diretamente este host sem a necessidade de passar pelo filtro do *firewall* centralizado. Por exemplo, é possível que vírus ou worms explorem vulnerabilidades em serviços ativos neste host ou simplesmente que se faça um acesso remoto não permitido a um host de uma origem qualquer. Normalmente estes hosts possuem algum tipo de *firewall*, mas na maioria das vezes estes estão desabilitados ou mal configurados, pois esta tarefa consome muito tempo do administrador da infraestrutura e normalmente os próprios usuários e até mesmo administradores não entendem a importância deste recurso.

As estruturas de segurança aplicadas atualmente nas corporações não realizam a integração das regras do *firewall* principal, normalmente implementado no *backbone* entre as redes internas e a rede onde se encontram os servidores, conhecida como Rede Desmilitarizada (DMZ), e dos *firewalls* instalados nos clientes desta estrutura de rede. Normalmente as regras implementadas nos clientes nem refletem a política de segurança da

empresa ou em muitos casos, nem estão habilitados. Fator ainda mais crítico é que uma vez implementadas as regras ficarão estáticas independentemente do ambiente no qual estão sendo utilizadas, o que as torna ineficazes para a grande maioria das situações às quais o cliente é exposto rotineiramente como citado no primeiro parágrafo.

1.2 OBJETIVOS GERAIS E ESPECÍFICOS

O objetivo desta monografia é apresentar um sistema de gerenciamento de regras de firewall, baseado na arquitetura cliente/servidor, que permite manter a coerência das regras aplicadas nos firewalls da empresa. Através da porção servidora do software realiza-se o cadastramento destas regras que garantam a efetividade da aplicação das Políticas de Segurança da empresa, independentemente de onde o cliente esteja localizado.

A porção cliente do software, por sua vez será responsável pela verificação do ambiente de rede na qual ela está inserida e pelo posterior envio das informações à porção servidora, onde estas informações serão processadas retornando para o cliente um conjunto de regras que devem ser aplicadas no firewall nativo do sistema operacional para finalização do processo.

Diante da estrutura apresentada, os objetivos específicos serão:

- Modelagem de estrutura de dados para armazenamento das regras e parâmetros a serem implementados;
- Utilização de servidor de aplicação para centralização das regras de firewall, contando com criptografada no envio e no recebimento dos dados através do *Hypertext Transfer Protocol Secure* (HTTPS);
- Desenvolvimento do software, em linguagem que garanta portabilidade, dividido em: cliente e servidor;
- Desenvolvimento da porção servidora do software contendo um painel de gerenciamento de regras e parâmetros de forma simplificada e objetiva;
- Desenvolvimento da porção cliente do software totalmente automatizada e integrada a um *Intrusion Detection System* (IDS) para que seja possível

realizar as verificações necessárias na infraestrutura a qual o cliente está conectado.

- Realização de testes para verificar a eficiência do gerenciamento das regras em contextos onde há segurança e onde não há.

1.3 ESTRUTURA DA MONOGRAFIA

Os assuntos abordados nesta monografia estão divididos em oito capítulos relacionados da seguinte forma:

Capítulo 1: Contextualiza o escopo da pesquisa, ilustrando as principais motivações para o desenvolvimento do trabalho proposto e os objetivos pretendidos.

Capítulo 2: Apresenta os conceitos de Segurança de Informação e todas as definições envolvidas neste tema utilizadas para a implementação do projeto.

Capítulo 3: Fornece uma visão geral e discute as arquiteturas e as principais funcionalidades dos *firewalls*.

Capítulo 4: Descreve as principais características dos produtos Endpoint Security existentes no mercado.

Capítulo 5: Descreve os principais trabalhos relacionados ao tema.

Capítulo 6: Detalha as tecnologias utilizadas e a implementação do software com todas as suas características.

Capítulo 7: Apresenta os resultados obtidos com os testes.

Capítulo 8: Descreve as possibilidades de trabalhos futuros.

2 SEGURANÇA DA INFORMAÇÃO

2.1 INTRODUÇÃO

Nos dias atuais, segurança é algo imprescindível que pode significar a continuidade dos negócios, a divulgação de segredos industriais e até mesmo a morte de uma empresa. Cada vez mais, as empresas buscam fazer o possível para se adequarem à realidade existente no mundo da informática e procuram buscar bons administradores para projetarem e administrarem suas redes de forma a não permitirem um acesso indevido às mesmas. Contudo, nada do que se relaciona à segurança pode ser definido como algo que sempre valerá sobre quaisquer circunstâncias. Na verdade, esse conceito se traduz em minimizar a vulnerabilidade de bens (qualquer coisa de valor) e recursos. Entendendo-se por vulnerabilidade, qualquer falha ou fraqueza que pode ser explorada para se ter acesso a um sistema ou os dados, segundo definido por Soares (SOARES, 1995) ou ainda como sendo uma falha existente em um ou mais ativos que pode ou não ser explorada por uma ameaça. As vulnerabilidades por si só não causam incidentes, elas apenas são brechas para que uma possível ameaça os cause (SÊMOLA, 2003).

A preocupação com a segurança em redes de computadores é algo que não deve só ser levada em consideração como um fator de potencial comprometimento da integridade dos dados informatizados de uma empresa. Ela deve ser julgada como um problema cujas consequências resultam muito provavelmente em um prejuízo financeiro que, em alguns casos, podem ter proporções catastróficas.

2.2 NECESSIDADE DA SEGURANÇA DA INFORMAÇÃO

A partir das vulnerabilidades e fraquezas, invasores de sistemas podem atacar sistemas desprotegidos. Segundo o organismo internacional de resposta a incidentes *Computer Emergency Response Teams* - CERT (CERT, 2006), esses ataques geralmente visam:

- Acesso a informações sigilosas: geralmente o atacante realiza o ataque visando obter informações importantes de um determinado alvo. Ex: documentos e planilhas importantes, registro de banco de dados confidenciais, senhas de clientes de banco etc.;
- Uso do PC alvo como um atalho para invadir outra vítima: esse tipo de ataque considera o PC da vítima como uma porta de passagem para se invadir outra vítima ou, na maioria dos casos, um servidor de grande porte, preservando o anonimato do invasor (seria como se o PC alvo fosse efetivamente o autor do ataque ao servidor pois a conexão maliciosa será realizada a partir dele);
- Negação de serviço (*DoS*): é um tipo de ataque em que o atacante utiliza conexões múltiplas e simultâneas para gerar uma sobrecarga em um sistemas de forma a indisponibilizá-la ao(s) seu(s) usuário(s);
- Negação de serviço distribuída (*DDoS*): constitui em um tipo de ataque *DoS* ampliado, no qual um atacante utiliza vários computadores de vítimas "escravos" para atacar um determinado sistema. Atualmente o ambiente é ainda mais atrativo pois com o aumento dos usuários de banda larga (aumento na velocidade de conexão), permite-se que haja um ataque muito mais intenso não só devido à quantidade de computadores usados no processo; e
- Danificar dados: às vezes, os atacantes utilizam os ataques para destruir, apagar ou simplesmente alterar dados e informações de uma determinada vítima. O mais assustador é que muitas vezes, algumas empresas contratam esses atacantes para prejudicarem outras empresas concorrentes. Isso também ocorre em relação ao primeiro item na forma de roubo de informações.

Apesar do contexto de defesa realizada em camadas (várias barreiras entre o atacante e o alvo), normalmente conhecido como modelo "cebola", nenhum administrador de redes, apesar de deter o mais alto nível de conhecimento no assunto, conseguirá garantir de forma plena que esta barreira será intransponível (CERT, 2006). Algumas informações importantes a serem consideradas são as seguintes (SÊMOLA, 2003):

- Não existem sistemas de segurança e sistemas operacionais totalmente seguros e sem falhas;

- Não existem ferramentas de segurança nas quais um atacante nunca conseguirá detectar falhas para superá-las;
- Não existe sequer um sistema ou servidor inviolável;
- Não existe nada que não possa ser vencido por um atacante experiente e determinado; e
- Não existe segurança plena ou definitiva.

Embora todos esses fatores sejam preocupantes, o foco deve estar na garantia da aplicação dos controles implementados. As ferramentas de segurança nos sistemas de informação, sobretudo as que serão tratadas neste trabalho, cumprem muito bem o seu papel de mitigar os riscos gerados pelos ambientes de redes que, atualmente, encontram-se muito mais interconectados do que há alguns anos.

2.3 PRINCIPAIS FATORES DE PREOCUPAÇÃO

Conforme descrito por Nakamura (2007), na proteção de um sistema inicialmente é preciso relacionar todos os recursos que podem estar suscetíveis a algum risco. Deve-se considerar o que é necessário proteger, os recursos mais importantes e de que forma as informações estão armazenadas (disco rígido e memória). Em um sistema computacional, os tipos de recursos mais comuns são:

- Hardware: processadores, unidades de disco, placas, teclados, terminais, computadores pessoais, estações de trabalho, impressoras, servidores, roteadores, linhas de comunicação;
- Software: aplicativos utilitários, programas de diagnóstico, sistemas operacionais, programas de comunicação, aplicativos;
- Dados: em processamento, armazenados on-line, em trânsito nos dispositivos de linhas de comunicação, backups, logs de auditoria, bases de dados;
- Pessoas: usuários e funcionários que estarão habilitados a operarem os sistemas;
- Documentação: sobre sistemas, hardware, programas, procedimentos administrativos, fluxo de dados, modelagem de dados; e
- Suprimentos: papéis, formulários, fitas, CD-ROM, DVDs e etc.

2.4 ERROS DE CONFIGURAÇÕES NOS EQUIPAMENTOS DE SEGURANÇA

De acordo com Domingues (2006), muitas falhas de segurança são originadas pela má configuração nos equipamentos de segurança. Isso pode ocorrer a partir de um simples serviço de rede, iniciado por um software que nem sempre o utiliza ou até mesmo por meio de um *Firewall* mal configurado. Além disso, podem ocorrer falhas no sistema operacional ou em aplicativos que resultem na disponibilização de acesso indevido a um host. Os riscos resultantes vão permanecer até que se desative o serviço de rede iniciado pelo software, complementem-se as regras do *Firewall* e corrijam-se as falhas do sistema e dos softwares vulneráveis suscetíveis a ataques.

Um bom exemplo de falha de segurança são os serviços de impressão compartilhados e de compartilhamento de arquivos. Esses podem ser ativados em uma instalação de atualização ou como padrão em um sistema operacional, deixando-o inseguro. É necessário saber identificar quando estão ativos e desativá-los.

2.5 TRÊS PILARES DA SEGURANÇA DA INFORMAÇÃO

Segundo Nakamura (2007) a expectativa de todo usuário, em relação à segurança de dados, é que as informações das quais se necessita, e que estão armazenadas em algum dispositivo computacional, estejam sempre nele disponível e que nunca sejam acessadas por qualquer pessoa não autorizada. Mesmo que o conceito de segurança num sistema computacional seja comumente relacionado à sua garantia de funcionamento de acordo com o que lhe é solicitado pelo usuário, o usuário espera que suas informações sempre estejam no local que ele estabelecer, que sejam confiáveis corretas e mantidas longe do acesso de pessoas não autorizadas. Essas expectativas dos usuários relacionadas à segurança de suas informações podem ser traduzidas em objetivos de segurança.

Como existem várias formas de implementação de segurança em informática, os objetivos de segurança variam de acordo com o tipo de ambiente computacional e a natureza do sistema em relação à sua área de atuação (administrativo, financeiro, militar, governamental, etc.). Dentre esse tipo de panorama é necessário analisar a natureza da

aplicação, dos riscos e impactos prováveis para se identificar os objetivos mais prioritários para a organização. Todos os envolvidos na implementação da segurança de uma empresa, desde um usuário comum até os profissionais de informática, devem se preocupar com os seguintes objetivos de segurança.

De acordo com Kurose (2010), podem-se identificar como desejáveis as seguintes características:

- Confidencialidade: Somente o remetente e o destinatário pretendido devem poder entender o conteúdo da mensagem transmitida;
- Autenticação: O remetente e o destinatário precisam confirmar a identidade da outra parte envolvida na comunicação, confirmando assim que a outra parte é quem alega ser.
- Integridade: Mesmo que o remetente e o destinatário consigam se autenticar reciprocamente, eles também querem assegurar que o conteúdo de sua comunicação não seja alterado, por acidente ou má intenção, durante a transmissão ou armazenamento.
- Disponibilidade: A necessidade imperiosa de segurança na rede ficou dolorosamente óbvia nos últimos anos devido a numerosos ataques de recusa de serviço (Denial of Service – DoS) que inutilizaram uma rede, um hospedeiro, ou qualquer outro componente da infraestrutura da rede, para seu usuários legítimos.

2.6 CRIPTOGRAFIA

Segundo Tanenbaum (2003) o propósito da criptografia é levar uma mensagem ou um arquivo, chamado texto plano, e criptografá-lo em um texto cifrado, de tal modo que somente a pessoa autorizada saiba convertê-lo novamente para um texto plano. E ainda, segundo CERT (2006), criptografia é a ciência e arte de escrever mensagens em forma cifrada ou em código. É parte de um campo de estudos que trata das comunicações secretas, usadas, dentre outras finalidades, para:

- Autenticar a identidade de usuários;
- Autenticar e proteger o sigilo de comunicações pessoais e de transações comerciais e bancárias;

- Proteger a integridade de transferências eletrônicas de fundos.

Uma mensagem codificada por um método de criptografia deve ser privada, ou seja, somente aquele que enviou e o destinatário pretendido devem ter acesso ao conteúdo da mensagem. Além disso, uma mensagem deve poder ser assinada, ou seja, a pessoa que a recebeu deve poder verificar se o remetente é mesmo a pessoa que diz ser e ter a capacidade de identificar se uma mensagem pode ter sido modificada. (Kurose, 2010)

A mensagem original é denominada texto limpo. Após a transformação por meio de um algoritmo, a mensagem passa a ser conhecida como texto cifrado ou texto criptografado. Um algoritmo de cifragem transforma o texto limpo em texto cifrado, enquanto um algoritmo de decifragem reverte o processo, transformando o texto cifrado em texto limpo. (Kurose, 2010)

Atualmente, os algoritmos criptográficos são divulgados à comunidade e o sigilo das informações é garantido apenas pela chave. O que significa que, se alguém descobrir a chave para decifrar uma determinada informação, todas as outras informações cifradas com esse algoritmo ainda estarão protegidas, por terem chaves diferentes.

Já um algoritmo criptográfico que não utiliza o recurso de chaves para cifrar as informações pode resultar em perigosas consequências. Por exemplo, se este algoritmo for quebrado, todas as informações cifradas por ele estarão desprotegidas, pois o que garante o sigilo das informações é o próprio algoritmo.

Por esses motivos, toda criptografia moderna opera com chaves. Portanto, para criptografar uma mensagem, precisa-se de um algoritmo de cifragem, uma chave e um texto limpo. Deste conjunto, origina-se o texto cifrado. Para decifrar uma mensagem, precisamos de um algoritmo de decifragem, uma chave e um texto cifrado. Deste conjunto revela-se o texto limpo original. (Kurose, 2010)

2.6.1 Criptografia com Chave Simétrica

Na criptografia com chave simétrica, “A” usa uma determinada chave e um algoritmo de cifragem para criptografar a mensagem, enquanto “B” usa a mesma chave e um algoritmo de decifragem recíproco para decifrar a mensagem. Por recíproco, deve ficar

subtendido que o algoritmo de decifragem realiza operações inversas com relação ao algoritmo de cifragem. Na Figura 1 pode-se visualizar o processo de criptografia com chave simétrica.

Os algoritmos de chave simétrica são eficientes, porém a cada par de usuários deve-se associar uma única chave (FOROUZAN, 2006). Isto significa que se “n” pessoas quiserem se comunicar usando este método, serão necessários n(n−1)/2 chaves simétricas. O que gera o problema de armazenamento e distribuição das chaves.

Os algoritmos criptográficos com chaves simétricas mais utilizadas são: DES (*Data Encryption Standard*), AES (*Advanced Encryption Standard*), 3-DES e RC4 (*Rivest Cipher 4*).

Figura 1 – Comunicação utilizando criptografia com chave simétrica.

2.6.2 Criptografia com Chave Pública

Na criptografia com chave pública, há duas chaves: uma chave privada e uma chave pública. A chave privada é mantida em segredo pelo receptor, enquanto a chave pública é distribuída publicamente. Uma restrição, com relação a estas chaves, é que a chave privada não pode ser obtida a partir da chave pública. Por exemplo, se Alice desejar enviar uma mensagem secreta para Bob, ela deverá usar a chave pública de Bob para cifrar a mensagem. Quando a mensagem for recebida por Bob, a chave privada dele será usada para decifrar a ensagem, conforme mostrado na Figura 2 (FOROUZAN, 2006).

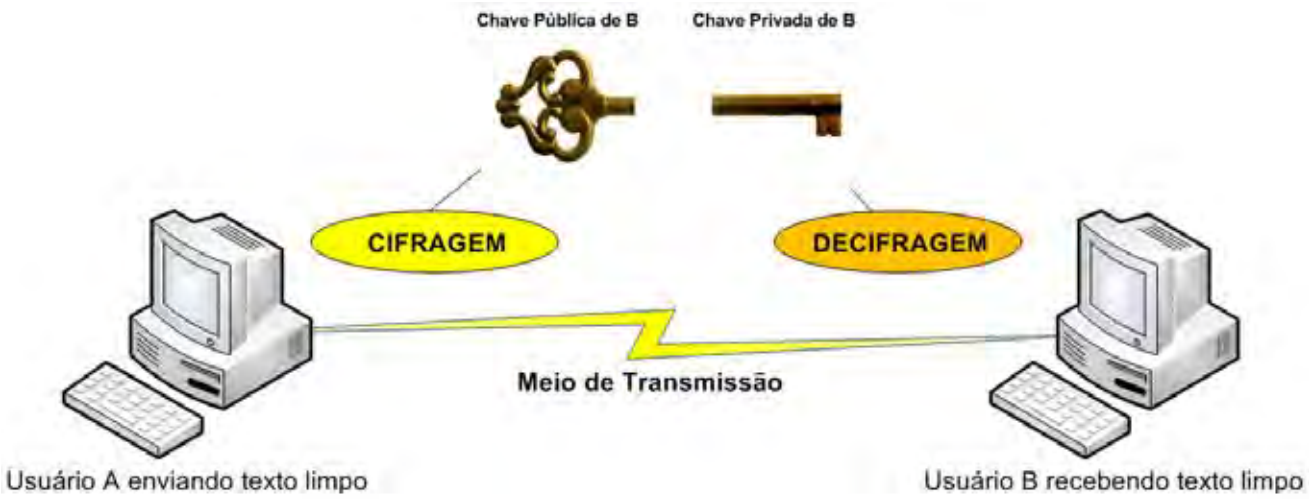

Figura 2 – Comunicação utilizando criptografia com chave pública.

A cifragem com chave pública remove a necessidade de uma chave compartilhada entre as duas entidades. Desta forma, para uma comunicação segura com n entidades, serão necessários apenas 2n chaves. Por exemplo, se utilizarmos a criptografia com chave pública em uma rede com 100 entidades precisaríamos de 200 chaves, enquanto com criptografia com chave simétrica se fazem necessárias 4.950 chaves.

A maior desvantagem da criptografia com chaves públicas é a complexidade dos algoritmos. Se desejarmos que o método seja relativamente efetivo, precisamos de chaves muito extensas. Porém, o cálculo do texto cifrado a partir do texto limpo usando chaves muito extensas leva a uma quantidade de tempo relativamente grande (FOROUZAN, 2006).

A distribuição de chaves também é um problema nestes tipos de algoritmos. Para evitar problemas de segurança, a associação entre uma entidade e sua respectiva chave pública deverá ser verificada. Por exemplo, se o usuário “A” enviar a sua chave pública por e-mail para o usuário “B”, como “B” terá certeza que foi realmente “A” que enviou. Uma solução faz uso de uma nova entidade na rede chamada de autoridade certificadora, que visa a garantir a validade jurídica aos atos documentados ou realizados de forma eletrônica (ICP, 2011). Esta nova entidade se responsabilizará por entregar as chaves públicas de forma confiável.

2.7 AUTENTICAÇÃO

Segundo Silberschatz (2008) a autenticação do usuário é baseada em um ou mais de três itens: posse de algo pelo usuário (uma chave ou cartão), conhecimento pelo usuário de algo (um identificador de usuário e senha) e/ou um atributo do usuário (impressão digital, padrão de retina ou assinatura). Por exemplo, se o usuário “A” deseja retirar certa quantia em dinheiro da sua conta corrente. Para isso, ela envia uma mensagem de solicitação de saque para o usuário “B”. Através do serviço de autenticação, “B” poderá ter certeza que quem enviou a mensagem foi realmente “A” e não outro.

Existem duas questões centrais relativas ao problema de troca de chave de autenticação: confidencialidade e adequação no tempo. Para evitar que um atacante se passe por uma entidade autorizada, as informações de identificação precisam ser comunicadas de forma criptografada. A segunda questão, adequação no tempo, é importante devido às ameaças de repetições de mensagem.

Uma técnica para lidar com ataques por repetição é chamada desafio/resposta. Nesta técnica o usuário “A” primeiro envia a “B” um desafio e exige que a mensagem subsequente (resposta) recebida de “B” contenha o valor correto.

A definição mais geral de autenticação dentro de sistemas de computação engloba a verificação da identidade, autenticação da origem e conteúdo da mensagem. O conceito de verificação de identidade, especificamente, aplica-se a usuários humanos, sistemas de computação e processos executando nesses sistemas. A autenticação pelo conhecimento de uma informação secreta ou a pela posse de um dispositivo físico de autenticação único são igualmente válidos para todos os tipos de entidades descritos acima. Por outro lado, autenticação biométrica apenas tem sentido no contexto de seres humanos.

Mecanismos de autenticação confiáveis são críticos para a segurança de qualquer sistema de informação automatizado. Quando um usuário legítimo é verificado, são aplicadas técnicas de controle de acesso para permitir seu acesso aos recursos do sistema. Se a identidade dos usuários legítimos puder ser verificada com um grau aceitável de certeza, as tentativas de acesso ao sistema sem a devida autorização podem ser negadas.

Existe uma variedade de métodos para executar autenticação de usuários, os quais formam a base dos sistemas de controle de acesso. As três categorias de métodos para

verificação da identidade de um usuário são baseadas em: algo que ele sabe, tal qual uma senha; algo que o ele possui, tal qual um token de autenticação; e alguma característica física do usuário, tal qual a impressão digital ou padrão de voz. De modo a usar estas características para verificar a identidade de um indivíduo, os sistemas de computadores usam software, hardware ou a combinação dos dois. (FOROUZAN, 2006).

2.8 CONSIDERAÇÕES FINAIS

Este capítulo apresentou conceitos envolvidos em Segurança da Informação e os conceitos necessários para realizar a proteção de informações em sua transmissão ou armazenamento.

Um grande fator descrito neste capítulo é a ocorrência de falhas nas configurações das tecnologias existentes. Não é incomum equipamentos serem comprometidos por falhas nas configurações (IDGNOW, 2007), apesar de serem construídos com tecnologias que permitiriam o controle destes acessos indevidos, não são utilizados por falta de conhecimento.

Cada vez mais, surgem oportunidades para isso, seja por meio de falhas ou vulnerabilidades que, segundo Soares (1995), é qualquer fraqueza que pode ser explorada para se violar um sistema ou as informações que ele contém. Estas vulnerabilidades existem abundantemente nos sistemas operacionais e em aplicativos utilizados atualmente. Uma boa política aplicada a segurança é a defesa em camadas. Por isso, todos os aspectos deverão ser considerados e a colocação de muitas barreiras, na prática, torna-se mais eficiente que uma única barreira por mais forte que seja.

Nos próximos capítulos são descritos os tipos de firewall e quais os conceitos necessários para configurá-los.

3 *FIREWALL*

3.1 INTRODUÇÃO

Firewall é uma ferramenta de segurança muito importante no cenário de mundo atual. À medida que o uso de informações e sistemas é cada vez maior, a proteção destes dos dados requer a aplicação de ferramentas e conceitos de segurança eficientes.

Este capítulo mostra o que é firewall, seus tipos, modos de funcionamento e a aplicabilidade em cada cenário.

3.2 DEFINIÇÕES E CONCEITOS

Firewall é o nome dado ao dispositivo de uma rede de computadores que tem por objetivo aplicar uma política de segurança definida pela corporação a um determinado ponto de controle da rede (KUROSE, 2010). Sua função consiste em regular o tráfego de dados entre redes distintas e impedir a transmissão e/ou recepção de acessos nocivos ou não autorizados de uma rede para outra. Este conceito inclui os equipamentos de filtros de pacotes e de proxy de aplicações, comumente associados a redes TCP/IP (*Transmition Control Protocol / Internet Protocol*) (SOARES, 1995). Ainda pode ser definido, de acordo com (NAKAMURA, 2007), como “componente ou conjunto de componentes que restringem o acesso entre uma rede protegida e a Internet, ou entre outros conjuntos de redes” .

Um dispositivo para ser denominado *Firewall* deverá:

- Analisar todo o tráfego que sai ou entra em uma rede;
- Somente permitir o tráfego de dados autorizados pela Política de Segurança da corporação;
- Ser imune a violações.

O termo inglês *firewall* faz alusão comparativa da função que este desempenha para evitar o alastramento de acessos nocivos dentro de uma rede de computadores a uma parede corta-fogo (*firewall*), que evita o alastramento de incêndios pelos cômodos de uma edificação (FIREWALLa, 2007).

Firewalls podem ser implementados em software e hardware, ou como uma combinação de ambos, neste caso, normalmente chamado de *appliance*. Sua complexidade depende muito do tamanho da rede, da política de segurança, da quantidade de regras que autorizam o fluxo de entrada e saída de informações e do grau de segurança desejado (Kurose, 2010)

3.3 HISTÓRICO E EVOLUÇÃO

Os sistemas *firewall* nasceram no final dos anos 80, fruto da necessidade de criar restrição de acesso entre as redes existentes. Nesta época a expansão das redes acadêmicas e militares, que culminou com a formação da ARPANET e, posteriormente, a Internet e a popularização dos primeiros computadores tornou-se um prato cheio para a incipiente comunidade hacker. Casos de invasões de redes, de acessos indevidos a sistemas e de fraudes em sistemas de telefonia começaram a surgir, e foram retratados no filme Jogos de Guerra ("*War Games*"), de 1983. Em 1988, administradores de rede identificaram o que se tornou a primeira grande infestação de vírus de computador, que ficou conhecido como Internet Worm. Em menos de 24 horas, o worm escrito por Robert T. Morris Jr disseminou-se por todos os sistemas da então existente Internet (formado exclusivamente por redes de ensino e governamentais), provocando um verdadeiro "apagão" na rede (FIREWALLb, 2010).

3.3.1 GERAÇÕES DE FIREWALL

Primeira Geração (Filtros de Pacotes)

A primeira geração de firewalls foi disseminada em 1988 através de pesquisa sustentada pela empresa DEC (*Digital Equipment Corporation*), mas o primeiro modelo para prova de conceito foi desenvolvido por Bill Cheswick e Steve Bellovin da *AT&T* (*American Telephone and Telegraph*) (FIREWALLb, 2010).

Esta geração inicial de *firewall* tratava-se de um filtro de pacotes responsável pela avaliação de pacotes do conjunto de protocolos TCP/IP, controlando somente origem e o destino dos pacotes. Apesar do principal protocolo de transporte do modelo TCP/IP, o TCP orientar-se a um estado de conexões, o filtro de pacotes não tinha o objetivo de usar estas informações inicialmente (uma possível vulnerabilidade) (FIREWALLb, 2010).

O *ipchains* é exemplo recente de um *firewall* que utiliza a tecnologia desta geração. Hoje o *ipchains* foi substituído pelo *iptables* que é nativo do Linux e com maiores recursos. Atualmente, a filtragem de pacotes é implementada na maioria dos roteadores e é transparente aos usuários, porém pode ser facilmente contornada com IP *Sniffers/Spoofers* (técnicas usadas por hackers para iludir os roteadores, com a troca de endereços de origem). Por isto, o uso de roteadores como única defesa para uma rede corporativa não é aconselhável.

Até hoje, este tipo de tecnologia é adotada em equipamentos de rede para permitir configurações de acesso simples (as chamadas "listas de acesso") (FIREWALLb, 2010).

Regras Típicas na Primeira Geração

- Restringir tráfego baseado no endereço IP de origem ou destino;
- Restringir tráfego através da porta (TCP (*Transmition Control Protocol*) ou UDP *(User Datagram Protocol*) do serviço.

Segunda Geração (Firewall de Aplicação ou Firewall Proxy)

A segunda geração foi baseada nos trabalhos de Gene Spafford (coautor do livro *Practical Unix and Internet Security*), Marcos Ranum (fundador da empresa TIS), e Bill Cheswick (AT&T).

O grande avanço nesta geração é que esta geração de firewalls é capaz de "entender" alguns protocolos e aplicações (como FTP (*File Transfer Protocol*), DNS (*Domain Name System*) e HTTP (*Hypertext Transfer Protocol*)), e pode detectar quando um protocolo não desejado está tentando furtivamente conectar por uma porta não padrão para aquele protocolo ou quando o protocolo está tentando causar algum dano ao sistema (FIREWALLb, 2010).

Um *firewall* de aplicação é muito mais seguro e confiável se comparado ao um *firewall* da primeira geração porque ele trabalha em todas as camadas, de enlace a aplicação, do modelo ISO/OSI (*International Organization for Standardization / Open Systems Interconnection*). O modelo OSI é um sistema criado pela ISO, pois se reconheceu a necessidade da criação de um modelo de rede para ajudar os desenvolvedores a implementar redes que poderiam comunicar-se e trabalhar juntas. Assim ISO lançou em 1984 o modelo de referência OSI, representado por 7 camadas, para que os pacotes de dados trafeguem de uma origem até um destino usando a mesma linguagem, ou protocolo (TANENBAUM, 2003).

Esta geração pode filtrar protocolos de camadas altas como FTP, TELNET, DNS, DHCP (*Dynamic Host Configuration Protocol*), HTTP, TCP, UDP e TFTP (*Trivial File Transfer Protocol*). Por exemplo, se uma organização deseja bloquear todas as informações relacionadas com a palavra "envio", esta filtragem de conteúdo pode ser habilitada no *firewall* para bloquear essa palavra em particular. Apesar de conseguir executar esta função, esta geração de *firewall* gera uma perda de desempenho na rede.

Foi nesta geração que se lançou no mercado o primeiro *firewall* comercial, datado de 13 de Junho de 1991 o SEAL (*Screening External Access Link*), da DEC, detalhado na **Figura 3** (RANUM, 2010). Após isto, diversos produtos comerciais surgiram e se popularizaram na década de 90, como os *firewalls Raptor*, *Gauntlet*, *Sidewinder*, entre outros.

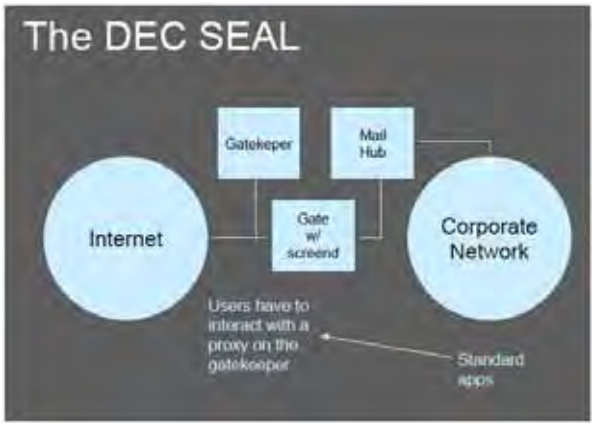

Figura 3 - Primeiro *Firewall* comercial (RANUM, 2010).

Regras Típicas na Segunda Geração

- Todas as regras da 1.ª Geração;
- Restringir o tráfego para início de conexões (*NEW*);
- Restringir o tráfego de pacotes que não tenham sido iniciados a partir da rede protegida (*ESTABLISHED*);
- Restringir o tráfego de pacotes que não tenham número de sequência corretos.

Terceira Geração (Firewall Stateful)

A criação da terceira geração de firewall normalmente é creditada a Nir Zuk e seu time da empresa Checkpoint, na década de 90. A grande característica de esta geração conseguir prover a função “stateful packet inspection (SPI)” ou “stateful inspection”, que é a capacidade de manter rastreável o estado das conexões de rede que trafegam através do firewall. O firewall é programado para distinguir pacotes legítimos para diferentes tipos de conexões. Somente pacotes que combinam com um estado de conexão conhecido serão permitidos pelo firewall, os demais pacotes serão rejeitados (FIREWALLb, 2010).

Um *firewall stateful* é capaz de guardar, em memória, atributos significativos de cada conexão que é estabelecida no *firewall*, do início até o final. Este pacote de atributos é normalmente conhecido como “estado da conexão” e pode incluir detalhes como os endereços IP, portas envolvidas na conexão e os números de sequência dos pacotes que estão nesta conexão. O maior gasto de

CPU é feito no momento da configuração da conexão. Todos os demais pacotes (para cada sessão) são processados rapidamente, porque é simples e rápido determinar se ele já pertence a uma sessão já existente ou se relaciona a alguma (por exemplo, pacotes ICMP (*Internet Control Message Protocol*) relacionadas a alguma sessão). Uma vez a sessão terminada, esta entrada na tabela é descartada.

Quando o protocolo utilizando na sessão é TCP o *firewall stateful* utiliza o *three-way handshak* (método utilizado pelo protocolo TCP para estabelecer ou encerrar uma conexão de forma confiável. Através da troca de três mensagens apenas é suficiente para garantir um acordo não ambíguo*)* (TANENBAUM, 2003) para formar a tabela de estados de conexão. Quando o protocolo é UDP ele não necessita deste recurso. Os passos em um estabelecimento de conexão TCP são:

- Quando um cliente inicia uma nova conexão, ele envia um pacote com o bit SYN (*flag* no cabeçalho do pacote que indica pedido de conexão) (TANENBAUM, 2003) marcada no cabeçalho do pacote.
- Todos os pacotes com o bit SYN setado, são considerados pelo *firewall* como novas conexões. Se o serviço que o cliente solicitou está disponível no destino, o serviço irá responder ao pacote SYN com um pacote em que tanto o bit SYN e ACK (flag no cabeçalho do pacote que indica confirmação de recebimento) (TANENBAUM, 2003) estão setados.
- O cliente irá então responder com um pacote em que apenas o bit ACK está definido. Após isto, a conexão vai entrar no estado *ESTABLISHED* (Estabelecido).
- O *firewall* irá transmitir todos os pacotes passando por suas interfaces, mas só irá permitir pacotes de entrada se forem parte de uma conexão estabelecida, assegurando que atacantes não possam iniciar conexões não solicitadas com a máquina protegida (LIMA, 2000).

A fim de evitar que a tabela de estado torne-se cheia, as sessões terão tempo limite de tráfego. Se ultrapassar um determinado período, estas sessões obsoletas são removidas da tabela de estado. Muitos aplicativos, portanto, enviam mensagens de *keepalive* (mensagem que indica que uma conexão está ativa)

periodicamente, a fim de não permitir que o *firewall* remova a conexão durante um período de inatividade da conexão do usuário. É interessante notar que o ataque mais comum do tipo *Denial of Service* (Negação de serviço) é o SYN *flood* (inundação de pacotes SYN), onde um usuário mal-intencionado envia grandes quantidades de pacotes SYN ao servidor, excedendo sua tabela de estado, não permitindo que o servidor aceite outras conexões (FIREWALLb, 2010).

Muitos *firewalls stateful* são capazes de controlar o estado do tráfego em protocolos sem garantia de conexão fim a fim, como UDP. Essas sessões começam geralmente no estado ESTABLISHED, imediatamente após o primeiro pacote ser enxergado pelo *firewall*. Sessões em protocolos sem conexão só pode terminar por *timeout* (ultrapassar o tempo máximo permitido de inatividade) (LIMA, 2000).

Ao se manter a rastreabilidade do estado da conexão, *firewalls stateful* proveem eficiência adicional em termos de inspeção de pacotes. Isso ocorre porque, para conexões existentes no *firewall* somente é necessário consultar a tabela de estado, em vez de verificar todo conjunto de regras do *firewall*. Há também um custo adicional quando o conjunto de regras do *firewall* é atualizado, pois a tabela de estado deve ser limpa completamente (FIREWALLb, 2010).

Regras Típicas na Terceira Geração

- Todas as regras das gerações anteriores;
- Restringir acesso FTP a usuários anônimos;
- Restringir acesso HTTP para portais de entretenimento;
- Restringir acesso a protocolos desconhecidos na porta 443 (HTTPS);
- Estados das conexões e fluxos UDP.

Quarta Geração e subsequentes

Com os avanços das gerações anteriores, nesta geração, o firewall consolida-se como uma solução comercial para implantação das políticas de segurança em redes de computadores.

Com o aumento da demanda por proteção e a evolução das arquiteturas de firewalls, muitas empresas como Fortinet, SonicWALL, Juniper, Checkpoint e Cisco começaram a investir maciçamente no aprimoramento de características de gerações anteriores, a citar:

- Melhoramentos no mecanismo *Stateful Inspection* para inspecionar pacotes e tráfego de dados baseado nas características de cada aplicação;
- Prevenção de Intrusão mesmo em conexões aparentemente legítimas;
- Implementação do *Deep Packet Inspection* (DPI) que examina um pacote ou fluxo de tráfego para determinar seu impacto e tomar decisões sobre sua significância.

Outra característica marcante desta geração é o início, a partir do ano de 2000, da utilização de "*Firewall* Pessoal", tecnologia de *firewall* aplicada também em estações de trabalho e computadores domésticos, o que trouxe um novo conceito chamado *Endpoint security*, que visa garantir a segurança na informação a partir do ponto onde ela é gerada ou disseminada (FIREWALLc, 2010).

O surgimento de soluções de *firewall* dedicado a servidores e aplicações específicas como servidores Web, banco de dados e email trouxe novos conceitos de proteção específicos para a aplicação e por isso podem especializar os testes de proteção trazendo uma eficácia diferenciada a esta geração (FIREWALLc, 2010).

3.3.2 Classificação de Firewalls

Os *firewalls* recebem uma classificação pelo comportamento de sua arquitetura. Esta classificação foi sendo construída ao longo do tempo, refletindo as novas características implementadas (FIREWALLb, 2010).

Filtros de Pacotes

São sistemas que analisam individualmente os pacotes na medida em que estes são transmitidos, verificando as informações das camadas de enlace e de rede (Modelo de referência ISO/OSI) (LIMA, 2000).

As regras podem ser formadas indicando os endereços de rede (de origem e/ou destino) e as portas TCP/IP envolvidas na conexão. A principal desvantagem desse tipo de tecnologia para a segurança reside na falta de controle de estado do pacote, o que permite que agentes maliciosos possam produzir pacotes simulados (com endereço IP falsificado, técnica conhecida como IP *Spoofing*), fora de contexto ou ainda para serem injetados em uma sessão válida. Esta tecnologia foi amplamente utilizada nos equipamentos de Primeira Geração (incluindo roteadores), não realizando nenhum tipo de decodificação do protocolo ou análise na camada de aplicação.

Proxy Firewall ou Gateways de Aplicação

Os conceitos de gateways de aplicação (*application-level gateways*) e "*bastion hosts*" foram introduzidos por Marcus Ranum em 1995. Trabalhando como uma espécie de eclusa, o *firewall* de proxy trabalha recebendo o fluxo de conexão, tratando as requisições como se fosse uma aplicação e originando um novo pedido sob a responsabilidade do mesmo *firewall* (*non-transparent proxy*) para o servidor de destino. A resposta para o pedido é recebida pelo *firewall* e analisada antes de ser entregue para o solicitante original.

Os gateways de aplicações conectam as redes corporativas à Internet através de estações seguras (chamadas de *bastion hosts*) rodando aplicativos especializados para tratar e filtrar os dados (os proxy *firewalls*). Estes gateways, ao receberem as requisições de acesso dos usuários e realizarem uma segunda conexão externa para receber estes dados, acabam por esconder a identidade dos usuários nestas requisições externas, oferecendo uma proteção adicional contra a ação dos atacantes.

Esta arquitetura possui algumas desvantagens, a citar:

- Para cada novo serviço que aparece na Internet, o fabricante deve desenvolver o seu correspondente agente de *Proxy*.
- Os *proxys* introduzem perda de desempenho na rede, já que as mensagens devem ser processadas pelo agente do *Proxy*.
- A tecnologia atual permite que o custo de implementação seja bastante reduzido ao utilizar CPUs de alto desempenho e baixo custo, bem como sistemas operacionais abertos (Linux), porém, exige-se manutenção específica para assegurar que seja mantido nível de segurança adequado.

Stateful Firewall (Firewall de Estado de Sessão)

Esta tecnologia permite que o *firewall* decodifique o pacote, interpretando o tráfego sob a perspectiva do cliente/servidor, ou seja, do protocolo propriamente dito e inclui técnicas específicas de identificação de ataques (LIMA, 2000).

Com a tecnologia *SMLI/Deep Packet Inspection,* implementada na quarta geração de *firewalls*, passou-se a utilizar mecanismos otimizados de verificação de tráfego para analisá-los sob a perspectiva da tabela de estado de conexões legítimas. Simultaneamente, os pacotes também vão sendo comparados a padrões legítimos de tráfego para identificar possíveis ataques ou anomalias. A combinação permite que novos padrões de tráfegos sejam entendidos como serviços e adicionados às regras válidas em poucos minutos. Supostamente a manutenção e instalação são mais eficientes (em termos de custo e tempo de execução), pois a solução se concentra no modelo conceitual do TCP/IP. Porém, com o avançar da tecnologia e dos padrões de tráfego da Internet, projetos complexos de *firewall* para grandes redes de serviço podem ser tão custosos e demorados quanto uma implementação tradicional.

Firewall de Aplicação

Com a explosão do comércio eletrônico, percebeu-se que mesmo a mais recente tecnologia em filtragem de pacotes para TCP/IP poderia não ser tão

efetiva quanto se esperava. Com todos os investimentos despendidos em tecnologia de *stateful firewalls*, os ataques continuavam a prosperar de forma avassaladora. Somente a filtragem dos pacotes de rede não era mais suficiente. Os ataques passaram a se concentrar nas características (e vulnerabilidades) específicas de cada aplicação. Percebeu-se que havia a necessidade de desenvolver um novo método que pudesse analisar as particularidades de cada protocolo e tomar decisões que pudessem evitar ataques maliciosos contra uma rede (FIREWALLc, 2010).

A primeira proposta de implementação desta arquitetura foi o projeto do *TIS (Trusted Information Systems)*, concebido por Marcos Ranum, que orientava a verificação dos métodos de protocolos de comunicação, mas o projeto TIS não traduz o conceito atual de *Firewall* de Aplicação, que nasceu principalmente pelo fato de se exigir a concentração de esforços de análise em protocolos específicos, tais como servidores Web e suas conexões de hipertexto HTTP. A primeira implementação comercial nasceu em 2000 com a empresa israelense *Sanctum*, porém, o conceito ainda não havia sido amplamente difundido para justificar uma adoção prática (FIREWALLc, 2010).

Se comparado com o modelo tradicional de *Firewall* (orientado a redes de dados), o *Firewall* de Aplicação é frequentemente instalado junto à plataforma da aplicação, atuando como uma espécie de procurador para o acesso ao servidor.

Alguns projetos de código-aberto, como por exemplo, o Mod*Security* para servidores Apache, têm por objetivo facilitar a disseminação do conceito para as aplicações Web (FIREWALLc, 2010).

A utilização desta arquitetura possui algumas vantagens, a citar:

- Pode suprir a deficiência dos modelos tradicionais e mapear todas as transações específicas que acontecem na camada da aplicação Web proprietária;
- Por ser um terminador do tráfego SSL, pode avaliar hipertextos criptografadas (HTTPS) que originalmente passariam despercebidos ou não analisados por *firewalls* tradicionais de rede;

E algumas desvantagens:

- Pelo fato de embutir uma grande capacidade de avaliação técnica dos métodos disponibilizados por uma aplicação (Web), este tipo de *firewall* exige um grande poder computacional;
- Ao interceptar aplicações Web e suas interações com o cliente, pode acabar por provocar alguma incompatibilidade no padrão de transações (LIMA, 2000).

3.4 CONSIDERAÇÕES FINAIS

Com o conteúdo deste capítulo verifica-se que houve um grande desenvolvimento nas arquiteturas de *firewalls* nos últimos anos. A necessidade de proteção de dados muito relevantes para os negócios forçou esta rápida evolução.

Estes equipamentos não podem garantir sozinhos a implementação completa da política de segurança e tão pouco garantir risco em nível zero, já que esta situação é somente utópica, mas é possível mitigar o risco a níveis aceitáveis para haja garantia de continuidade dos negócios e proteção coerente das informações.

4 ESTADO DA ARTE

4.1 INTRODUÇÃO

Os softwares de segurança para garantia de confidencialidade, integridade e disponibilidade, pilares da Segurança da Informação, estão crescendo 11% ao ano (OGLOBO, 2010) e são essenciais para garantia da continuidade dos negócios nos dias atuais. Como a quantidade de equipamentos móveis está crescendo significativamente, as soluções estão se voltando para a garantia da segurança nas redes internas e nas demais redes em que o equipamento móvel corporativo obtiver conexão. Este novo conceito de proteção visa a garantir que a Política de Segurança da empresa seja aplicada em qualquer ambiente, independente da hostilidade imposta ao aparelho. Um conceito que vem se tornando muito utilizado na implementação destes softwares é o *Endpoint Security*.

Este capítulo detalha o conceito de *Endpoint Security*, conceito este que foi utilizado como base para concepção do software base deste projeto, bem como testes realizados em alguns softwares de mercado que já implementam este conceito de *Endpoint*. São apresentados também neste capítulo, trabalhos relacionados a este e que trazem melhorias incorporadas por este trabalho.

4.2 CONCEITO DE ENDPOINT SECURITY

É uma estratégia na qual a segurança do software é distribuída para os equipamentos dos usuários, mas gerenciados de forma centralizada. Este sistema funciona em um modelo cliente/servidor, onde um programa é instalado em todo equipamento de usuário, que no contexto utilizado neste trabalho é todo dispositivo que se conecta à rede corporativa de uma empresa. *Endpoints* podem ser *PCs*, laptops, *haldhelds* entre outros. Um servidor que

contém um software que centraliza a segurança verifica o login destes softwares clientes e envia atualizações e informações quando necessário (ENDPOINTb, 2005).

Algumas implementações simples de *endpoint security* podem incluir *firewalls* pessoais ou softwares antivírus que são distribuídos, monitorados e atualizados a partir do servidor. Para prover ainda mais segurança estes podem conter sistemas de detecção e prevenção de intrusos, anti-spywares e software por bloqueio de comportamento (softwares com intenções maliciosas são bloqueados automaticamente) (ENDPOINTa, 2008).

Os modelos mais complexos de *endpoint security* utilizam controle de acesso à rede para garantir autenticação e especificam forma de acesso para os dispositivos. Quando um dispositivo tentar se conectar à rede o programa valida as credenciais do usuário e também vasculha o dispositivo para garantir que ele está de acordo com as Políticas de Segurança impostas pela empresa. Os elementos para permitir este acesso podem passar pela aprovação de um sistema operacional, um *firewall*, uma VPN ou algum anti-malware (ENDPOINTa, 2008).

4.2.1 ESTUDO DE CASO 1: *Sophos Endpoint Security and Data Protection*

Muitos softwares no conceito de *endpoint security* foram lançados no mercado mundial nos últimos anos (*Symantec Endpoint Protection, AVG Internet Security Network Edition, McAfee Total Protection Service–Advanced, Sophos Endpoint Security and Data Protection*) (ENDPOINTa, 2008) . Um dos softwares que mais se aproxima do modelo ideal é o SOPHOS ENDPOINT SECURITY AND DATA PROTECTION. Na sua versão NAC Advanced, ele é capaz de validar políticas de segurança pré-definidas no software, gerenciar, e também realizar checagem de atualizações em sistemas operacionais e antivírus existentes nos dispositivos de usuário. É possível realizar autenticação do acesso à rede através do protocolo 802.1x em servidores RADIUS ou LDAP (*Active Directory e OpenLDAP*) (ENDPOINTa, 2008).

Na Tabela 1 estão descritas as funcionalidades disponíveis para este software. Nota-se que há uma série de funcionalidades para verificação de malwares e características de configurações desejáveis para acesso a rede além de autenticações via protocolos abertos.

Tabela 1 : Funcionalidade do SOPHOS NAC. (SOPHOS, 2011)

Policy definition
Policies for managed, contractor and guest computers
Check if antivirus and other security applications are active and up to date
Check if OS service packs are current
Check if Microsoft/Windows Update active
Check 600 predefined OS patch definitions
Define policies by user groups
Create custom applications and policies
Assessment and remediation
Assess and autoremediate managed computers with a permanent NAC agent
Assess and control unmanaged computers with a dissolvable agent
Reporting
Standard reporting on endpoint compliance with policy
Advanced reporting and exporting capabilities for trending analysis
Configurable alerting to proactively track down potential problems
Enforcement
Agent Based - control managed computer network access
Microsoft DHCP - prevent unprotected / unauthorized computers accessing the network
802.1x, Cisco NAC, VPN, other DHCP - prevent unprotected / unauthorized computers accessing the network
Deployment
NAC agent that can be installed on managed computers running third party anti-virus
NAC agent that can be installed separately using software distribution tools

Na análise proposta por este capítulo e visando checar a execução das funcionalidades descritas pelo fabricante do software SOPHOS, foi instalado em um computador com sistema operacional Windows XP a versão NAC deste software. Foram habilitadas duas funções: *Policy Definition* e *Enforcement*.

Na funcionalidade *Policy Definition* foram habilitadas as seguintes verificações: checar se o software de antivírus e outros aplicativos de segurança estão ativos e atualizados e checar se o sistema operacional está atualizado. Após isto, foi instalado o cliente do software em computador que não continha antivírus, mas possuía sistema operacional atualizado. O software detectou com clareza tanto a falta do software antivírus como a situação de atualização do sistema operacional.

Por fim, na funcionalidade *Enforcement* foi testada a opção *Microsoft DHCP* em um Windows 2008 Server. Para isto, foi instalado um servidor DHCP, na opção do software foi negada a de distribuição de IP dinâmico para um computador Windows XP utilizado nos testes anteriores. Ficou comprovado que o computador não conseguia adquirir IP devido ao bloqueio do software.

De maneira geral, as opções do software cumprem o propósito a que foram criadas.

4.2.2 ESTUDO DE CASO 2: *Symantec Network Access Control 11*

Outro produto muito semelhante ao SOPHOS NAC é o *Symantec Network Access Control*. Ele é responsável por realizar uma série de validações no equipamento que contém um módulo cliente. Sendo assim, o acesso à rede dependente que estas características estejam de acordo com uma série de regras pré-estabelecidas.

Um grande diferencial com relação ao SOPHOS NAC é que este software realiza uma análise mais completa do ambiente de rede na qual o equipamento está conectado e informa, através de um relatório, o estado em que o equipamento será colocado: quarenta, ativo, entre outros. Ainda nesta mesma verificação, como é comum aos demais *Endpoints*, são checadas atualizações de sistemas operacional e *malwares* em geral.

Os principais recursos disponibilizados por este softwares, de acordo com a Symantec (2011) são:

- Bloqueiam ou colocam em quarentena os dispositivos que não estão em conformidade, e impede os seus acessos à rede e a recursos da empresa.
- A Integridade do host realiza testes de acordo com modelos predefinidos, como nível de *patch, service packs*, antivírus e status do *firewall* pessoal, bem como verificações personalizadas para o ambiente da empresa.
- Oferece cobertura completa de endpoints para *laptops*, *desktops* e servidores, gerenciados ou não, que estejam localizados dentro ou fora da rede da empresa.

E os principais benefícios, também de acordo com a Symantec (2011) são:

- Reduz a propagação de códigos maliciosos como vírus, worms, Cavalos de Tróia, spyware e outras formas de *malware*.

- Aumenta a disponibilidade da rede e reduz a interrupção dos serviços para os usuários finais.
- Produz informações para a verificação da conformidade organizacional através da verificação de conformidade do endpoint quase em tempo real.
- Verifica se os investimentos na segurança de endpoints, como antivírus e firewall do cliente, estão corretamente ativados.

4.3 TRABALHOS RELACIONADOS

Existem inúmeros trabalhos (CHAURE, 2010), (ZALIVA, 2009), (BARBOSA, 2006) e (MONTEIRO, 2006), e também softwares como SOPHOS ENDPOINT SECURITY AND DATA PROTECTION, da empresa SOPHOS (SOPHOS, 2011), no mercado implementando o conceito de Endpoint Security, mas nenhum visa à verificação do ambiente no qual a máquina está conectada. Todos eles fazem checagem de situações pontuais como: se o *firewall* está habilitado, se o antivírus está atualizado, se o sistema operacional possui as últimas correções, se o usuário possui requisitos para determinadas operações.

No trabalho *"An Implementation of Anomaly Detection Mechanism For Centralized and Distributed Firewalls"* (CHAURE, 2010), há uma análise sobre a garantia da efetividade do conjunto de regras de firewalls em ambientes centralizados e distribuídos e como estas alterações podem afetar a garantia de segurança. Porém, o trabalho mais completo no contexto proposto por este trabalho é o intitulado "Gestão Centralizada de *Firewalls* Distribuídas em Ambientes Heterogêneos", onde é proposto um gerenciamento centralizado das regras implementadas nos clientes, mas sem levar em conta o contexto de rede na qual a máquina cliente está envolvida (MONTEIRO, 2006). O objetivo geral do trabalho é a criação de uma plataforma de gestão de *firewalls* distribuída, recorrendo à utilização de políticas de segurança, sendo estas políticas regras que determinam o funcionamento dos equipamentos existentes numa rede e que, consequentemente, determinam o funcionamento global da rede.

Há também neste trabalho, a implementação de um console de administração, onde é possível realizar, independentemente, quatro tipos de tarefas: definição de políticas de segurança, conversão de políticas de segurança, difusão e aplicação de políticas de segurança, assim como a sua verificação. Dependendo da tarefa que o administrador deseja efetuar, tem à sua disposição diversos utilitários.

Os trabalhos citados trouxeram imensas contribuições para o evidenciamento da necessidade de controlar de forma centralizada as regras de *firewall* existentes independente da plataforma. O que este trabalho traz de novo é um ambiente integrado ao um IDS fazendo com que o *firewall* ganhe capacidade de scanear um banco de dados de assinaturas de ataques de rede e verificar online os riscos ao qual a máquina está exposta.

4.4 CONSIDERAÇÕES FINAIS

Após a análise de muitos trabalhos, nota-se que há inúmeras contribuições para verificação da efetividade do conjunto de regras, porém nota-se que não há verificação de um requisito importante para garantia da integridade de equipamentos móveis independentemente da rede à qual estão conectadas (externas ou internas): nenhum deles checa o ambiente no qual o equipamento está conectado e quais são os riscos aos quais os equipamentos estão expostos, fornecendo assim parâmetros de níveis de criticidade, ou seja, quanto o host está exposto, mais refinados para que o servidor possa fornecer um conjunto de regras ou definições de comportamento que permita que o equipamento seja utilizado, porém sem oferecer riscos à Política de segurança implementada pela corporação.

Outro detalhe fundamental desta análise é a incompatibilidade de utilização das soluções entre si e com os firewalls nativos dos sistemas operacionais. Conforme apontado pelos testes da TechWorld (TECHWORLD, 2008), todos os produtos necessitam da utilização de sua suíte completa. Nenhum deles permite integração de soluções parciais ou simplesmente a utilização do *Firewall* do Windows ou o *Iptables* nativo no Linux.

No próximo capítulo será discutida uma proposta de solução que visa preencher esta lacuna dos softwares atuais visando à utilização dos recursos já existentes na máquina e de forma transparente conseguir integrar aplicações.

5 FIREWALL DINÂMICO: UMA IMPLEMENTAÇÃO CLIENTE/SERVIDOR

5.1 INTRODUÇÃO

Este capítulo apresenta definições de tecnologias utilizadas na implementação do sistema desenvolvido neste trabalho: criptografia, servidor de aplicações, banco de dados e etc. As tecnologias aplicadas visam a fornecer ao software: segurança, escalabilidade e eficiência na centralização das regras de firewall.

Na construção desta solução foram utilizadas tecnologias com conceito de portabilidade como, por exemplo, a linguagem Java permitindo ao sistema a flexibilidade necessária para atender os cenários corporativos atuais: complexos e heterogêneos.

Na segunda parte do capítulo são apresentados os módulos criados para desenvolvimento do software. Por se tratar de uma arquitetura cliente/servidor são demonstrados os papéis desenvolvidos por cada módulo e a interação entre eles.

5.2 DEFINIÇÕES E CONCEITOS DAS TECNOLOGIAS UTILIZADAS

5.2.1 HTTPS (*HyperText Transfer Protocol Secure*)

É uma implementação do protocolo HTTP sobre uma camada TLS. Essa camada adicional permite que os dados sejam transmitidos através de uma conexão criptografada e que se verifique a autenticidade do servidor e do cliente através de certificados digitais.

O protocolo HTTPS é utilizado, em regra, quando se deseja evitar que a informação transmitida entre o cliente e o servidor seja visualizada por terceiros, como, por exemplo, no caso de compras online. (Tanenbaum, 2003).

5.2.2 Glassfish

É um servidor de aplicações de código aberto de nível corporativo que oferece desempenho, confiabilidade, produtividade e facilidade de uso superior, a uma fração do custo de servidores de aplicações proprietários. Como a implementação de referência Java EE é construída em código aberto, o GlassFish elimina a dependência de fornecedores, e permite que clientes aproveitem os mais recentes padrões e inovações do setor (NETO, 2009).

Glassfish é um servidor de aplicação de código livre (open source) desenvolvido pela Sun e distribuído juntamente com o IDE NetBeans 6.0. De forma geral, é um programa capaz de dar suporte ao desenvolvimento e à execução de aplicações web criadas segundo os padrões da plataforma Java. Na **Figura 4** pode-se visualizar a tela inicial padrão do servidor (NETO, 2009).

Figura 4 – Página inicial padrão do servidor Glassfish.

5.2.3 Java 2 Enterprise Edition (J2EE)

J2EE é a especificação de uma plataforma que é executa em um servidor. Esta plataforma fornece uma infraestrutura robusta que permite a criação de sistemas corporativos, e esses sistemas podem ser distribuídos e acessados por várias pessoas simultaneamente num ambiente web. O padrão J2EE proporciona que a aplicação desenvolvida seja independente de plataforma (TORTELLO, 2003). As principais

características da plataforma J2EE são: segurança, escalabilidade, independência de sistema operacional e arquitetura distribuída (TORTELLO, 2003).

A J2EE fornece uma estrutura de múltiplas camadas físicas e lógicas e seu modelo proporciona que sejam construídas soluções baseadas em componentes que permitem a reutilização dos mesmos. Os componentes lógicos podem ser divididos em componentes webs e componentes *Enterprise JavaBeans* (EJB). Os componentes da camada do cliente são executados na máquina do cliente, enquanto os componentes da camada web e de negócios são executados no servidor J2EE. O componente web pode ser classes servlets ou *páginas JavaServer Pages* (JSP). Já os componentes EJBs são componentes de negócios que são executados no servidor (MORAES, 2002).

Os componentes web são processados no contêiner web e os componentes EJB são processados no contêiner EJB no lado do servidor, enquanto que a interface é apresentada no lado do cliente. O contêiner web manipula as informações apresentadas ao cliente, processando servlets ou páginas JSPs, e gerenciando o seu ciclo de vida, enquanto o contêiner EJB processa e gerencia a execução de *enterprise javaBeans* (TORTELLO, 2003).

5.2.4 Mysql

Um banco de dados é uma coleção de dados estruturados. Ele pode ser qualquer coisa desde uma simples lista de compras a uma galeria de imagens ou a grande quantidade de informação da sua rede coorporativa. Para adicionar, acessar, e processar dados armazenados em um banco de dados de um computador, você necessita de um sistema de gerenciamento de bancos de dados, como o Servidor MySQL. Como os computadores são muito bons em lidar com grandes quantidades de dados, o gerenciamento de bancos de dados funciona como a engrenagem central na computação, seja como utilitários independentes ou como partes de outras aplicações (MYSQL, 2011).

O MySQL é um sistema de gerenciamento de bancos de dados relacional e, por isso armazena dados em tabelas separadas em vez de colocar todos os dados um só local. Isso proporciona velocidade e flexibilidade. A parte SQL do "MySQL" atende pela "*Structured Query Language* - Linguagem Estrutural de Consultas". SQL é a linguagem

padrão mais comum usada para acessar banco de dados e é definida pelo Padrão ANSI/ISO SQL.

5.2.5 SNORT

O SNORT é uma ferramenta NIDS *(Network Intrusion Detection System)* desenvolvida por Martin Roesch de código aberto e bastante popular por sua flexibilidade nas configurações de regras e pela constante atualização frente às novas formas de invasão. Outro ponto forte desta ferramenta é o fato de ter o maior cadastro de assinaturas, ser leve, pequeno, fazer escaneamento do micro e verificar anomalias dentro de toda a rede a qual um computador pertence (SNORT, 2011).

O código fonte é otimizado, desenvolvido em módulos utilizando linguagem de programação C e, junto, com a documentação, são de domínio público.

O Snort conta, ainda, com permanente desenvolvimento e atualização, que são feitos diariamente, tanto em relação ao código propriamente dito, como das regras de detecção. Os padrões utilizados na construção das regras de detecção das subversões são introduzidos no sistema de configuração, tão rápidamente quando são enviados os alertas originados pelos órgãos responsáveis, como por exemplo, o CERT, Bugtraq (lista de discussão), entre outros (SNORT, 2011).

5.3 ARQUITETURA DA SOLUÇÃO

O software projetado neste trabalho foi desenhado para contemplar uma arquitetura que provesse segurança em dois cenários: quando o dispositivo estivesse dentro da rede interna protegida por um firewall principal e em outras redes desconhecidas, protegidas por firewalls ou não.

O fluxo da informação é descrito na **Figura 5**, onde se pode ver que há uma centralização das tomadas de decisões baseadas nas regras criadas no Firewall Servidor. Independentemente da topologia lógica da rede na qual o software será utilizado, ele se comporta da mesma maneira já que é baseado em regras pré-definidas.

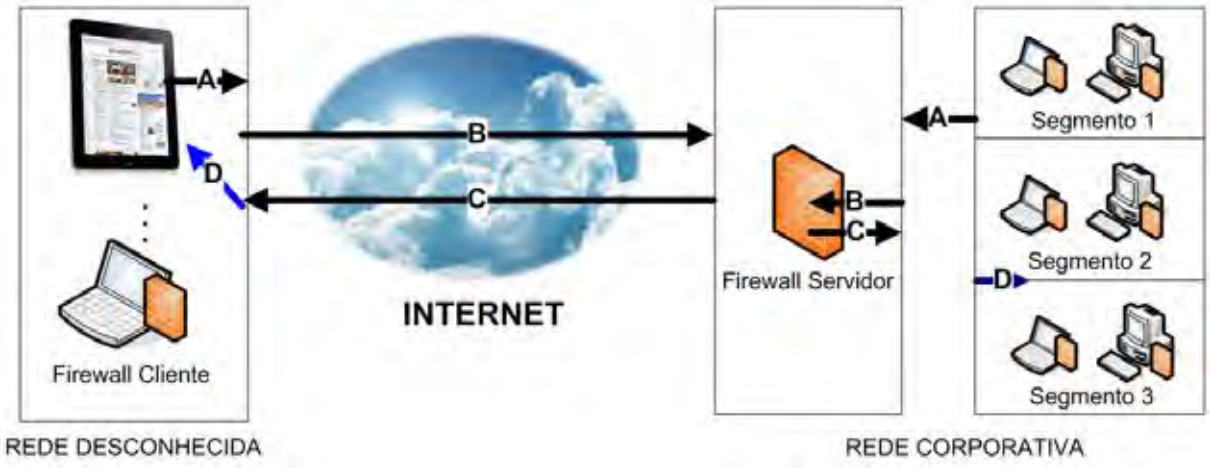

Figura 5 – Fluxo de ações da arquitetura.

Pode-se verificar que no fluxo exposto na **Figura 5**, o comportamento do software em um equipamento da rede interna ou da rede externa é idêntico, pois o seu comportamento independe de onde ele esteja e sim de qual o nível de segurança vigente naquela infraestrutura. A seguir são descritos os quatro passos do fluxo de ações:

a) O software cliente inicia o firewall nativo bloqueando todas as conexões de entrada e saída, liberando somente uma porta para realizar uma verificação do ambiente de rede no qual está conectado, coletando assim parâmetros do comportamento da rede (portas liberadas para o mundo externo, tentativa de comprometimento do host, etc.) para informar ao *firewall* servidor;

b) Após esta primeira fase é realizada uma conexão HTTPS com o *firewall* servidor para enviar os parâmetros colhidos no passo A, utilizando-se de qualquer porta liberada no firewall que controla a estrutura, em nossa implementação preferencialmente a 8181.

c) O *firewall* servidor recebe estes parâmetros enviados pelo cliente no passo B e os compara com sua base de dados de regras, que implementa a Política de Segurança criada pela empresa, verificando assim quais conjuntos de regras são mais pertinentes para implementar em determinado cliente neste momento.

d) Após receber do servidor firewall o conjunto de regras que se adéquam melhor ao ambiente e o momento vivenciado pelo equipamento, o software cliente aplica as regras no firewall nativo do sistema operacional;

5.4 INFRAESTURUTRA E DESENVOLVIMENTO DO SOFTWARE

Para viabilização da solução foi necessário instalar e configurar uma infraestrutura que permitisse o funcionamento do software desenvolvido. Na **Figura 6** podem-se verificar os componentes desta infraestrutura e a suas interações.

Todas as soluções adotadas na infraestrutura foram selecionadas por proverem três características: portabilidade, escalabilidade e segurança.

5.4.1 Instalação e configuração da infraestrutura: servidor e cliente

No servidor firewall instalou-se um servidor de aplicações (Glassfish), um servidor de aplicações Java que segue a referência de implementação do J2EE. Ele é responsável por executar o software e fornecer suporte a inúmeras *Application Programming Interfaces (API)* que possibilitam entre outras funções o acesso ao banco de dados, a criação das telas e interação entre os módulos utilizando criptografia.

Instalou-se também um Sistema Gerenciador de Banco de Dados (SGBD), o MySQL Community Server 5.5.13, onde são armazenadas todas as informações de controle e gerenciamento do conjunto de regras.

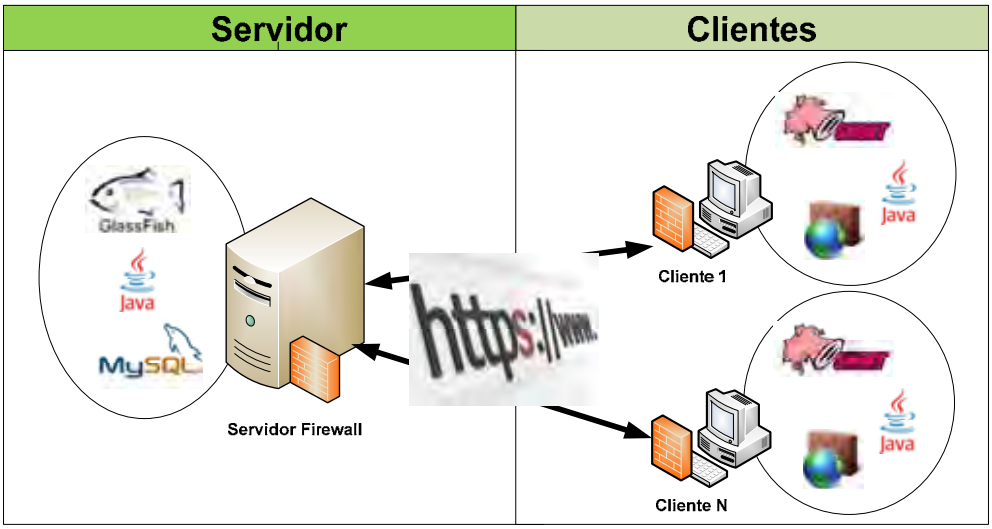

Figura 6 – Infraestrutura necessária para execução do software desenvolvido.

Como no cliente o intuito é utilizar o firewall nativo do sistema operacional e não criar outro firewall, pois esta estratégia geraria problemas de portabilidade, o firewall nativo foi habilitado e o software desenvolvido garante que as regras existentes sejam apagadas na inicialização para evitar problemas de má configuração.

Um *Instrusion Detection System* (IDS), o SNORT, foi instalado no cliente para realizar a varredura da rede à qual o dispositivo cliente está se conectando, visando detectar possíveis ameaças.

Por fim, foi instalado o Java 2 Standard Edition (J2SE), que é a edição básica e contém a máquina virtual que permite a execução de aplicações criadas na linguagem Java.

5.4.2 Modelagem da base de dados

Após a organização da divisão das funções do software foi realizada a modelagem de dados, como demonstrado na **Figura 7**, contemplando um cadastro de regras que permitisse a separação e posterior tratamento das saídas geradas pelo SNORT.

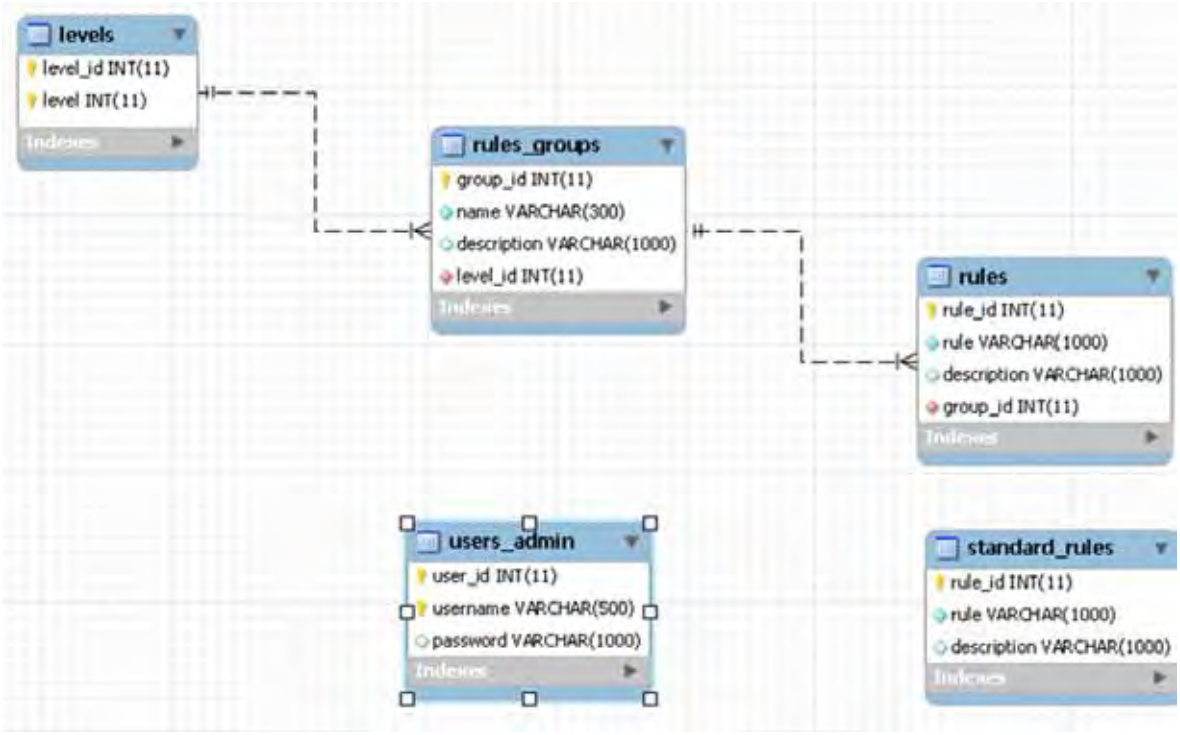

Figura 7 – Modelagem de dados do sistema.

Na modelagem foram contemplados os aspectos de segurança e organização das regras. Na **Tabela 2** estão descritas as funções de cada tabela criada e seus aspectos relevantes.

Tabela 2 – Tabelas criadas na modelagem de dados do sistema.

Nome da Tabela	Funções	Campos Relevantes
users_admin	Tabela que armazena os usuários que terão acesso ao Painel de Controle do sistema. Este cadastro é essencial para criação de sessões e garantia de confidencialidade.	• ***username*** (nome do usuário); • ***password*** (senha do usuário).
levels	É necessário criar níveis de criticidade (*level*) para diferenciar as situações às quais o cliente está exposto. Cada nível (*level*) vai indicar quais grupos de regras serão aplicados no cliente. Quanto maior o *level*, maior a restrição aplicada pelo grupo de regras criado.	• ***level*** (indica o número do *level*. Por padrão foram criados 1, 2 e 3), onde 3 é o mais crítico e 1 é o menos crítico.
rule_groups	Tabela para criação de grupos de regras. Esta estruturação facilita a aplicação de várias regras em um único grupo, evitando a desorganização das regras, o que normalmente acarreta erros de configuração.	• ***name*** (nome que identifica o grupo de regras); • ***level_id*** (indica em que nível (*level*) este grupo de regras será aplicado).
rules	Tabela que contém as regras para cada grupo criado na tabela *rule_groups*	• ***rule*** (regras padrão que serão aplicadas no cliente); • ***group_id*** (indica a qual grupo a regra pertence).
standard_rules	Tabela criada para aplicar conjunto de regras quando não há nenhum nível de criticidade informado pelo identificador do cliente.	• ***rule*** (regras padrão que serão aplicadas no cliente).

5.4.3 Desenvolvimento do software: Módulo Servidor

Como a arquitetura utilizada nesta solução é cliente/servidor, o software foi dividido em dois módulos com funções específicas: módulo servidor e módulo cliente. Cada um deles é responsável por executar uma parte da solução e manter a integridade e a coerência das informações enviadas e recebidas pelos módulos.

Na **Figura 8** visualiza-se a divisão dos módulos e todas as funções específicas de cada módulo.

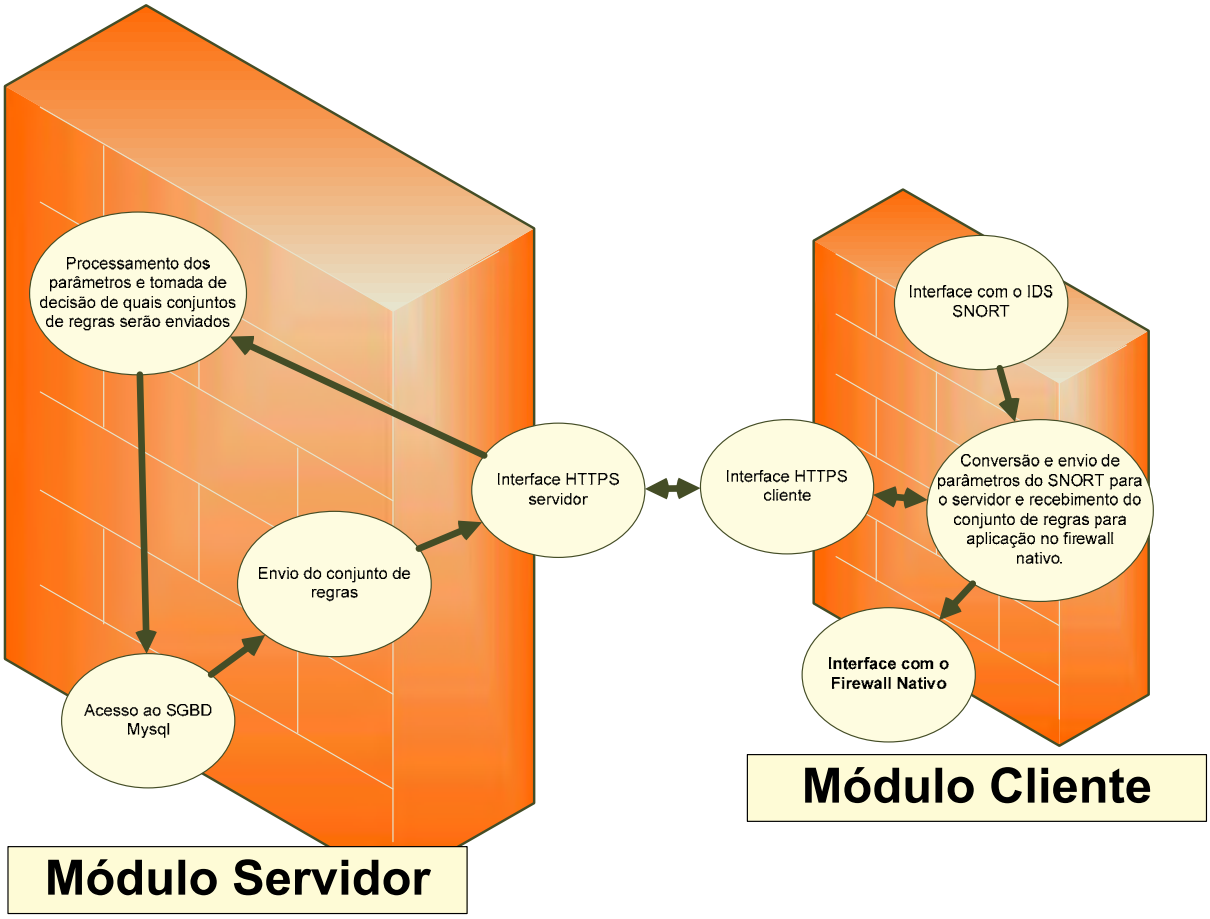

Figura 8 – Divisão modular do software.

Na implementação do módulo servidor, a primeira função criada foi a interface web, utilizando-se um *content* web. A interface foi criada para ser responsável por permitir a administração das regras, bem como transmitir e receber dados criptografados entre os módulos. Esta criptografia é realizada através do protocolo HTTPS utilizando-se um par de chaves pública e privada. Realizando este tipo de criptografia na comunicação há uma redução significativa do risco dos dados serem interceptados nos equipamentos pelos quais os pacotes deverão passar para realizar a comunicação entre os clientes e o servidor *firewall*. Esta solução garante confidencialidade na comunicação independente do ambiente em que o equipamento cliente se encontre.

Visando a garantir o controle de acesso à interface web do software foi criado um ambiente para criação e garantia de sessões, autenticando somente usuários cadastrados na tabela “users_admin” existente no banco de dados. Na **Figura 9**, é possível visualizar a interface de autenticação que garante que as páginas do aplicativo só possam ser acessadas por usuários legítimos.

Figura 9 – Login do sistema.

Após a criação da interface foi desenvolvida a função para recepção dos parâmetros do SNORT que depende de uma função implementada no cliente que faz tratamento do nível de criticidade informado pelo SNORT. Para isso, é preciso entender como o SNORT analisa a situação da conexão do cliente e que parâmetros ele irá enviar. Após a realização da varredura no cliente, o software SNORT informa para cada evento encontrado uma criticidade variando de 1 a 4, onde 1 é mais crítico e 4 menos crítico. (SNORT, 2011).

Como os grupos de regras dependerão do nível de criticidade para informar o conjunto de regras a ser aplicado no cliente, foi convencionado que seriam adotados três níveis para a implementação do software, já que os dois últimos níveis (baixo e muito baixo) são frutos de detecções praticamente idênticas e não seria necessário separá-los, conforme é possível visualizar na **Tabela 4**. Na **Tabela 3** é demonstrada a conversão entre os níveis do SNORT e os adotados no desenvolvimento deste software.

Tabela 3 – Conversão entre o nível informado pelo SNORT e o adotado pelo software.

Criticidade Informada pelo SNORT	Significado	Nível adotado pelo software
1	Criticidade Alta	3
2	Criticidade Média	3
3	Criticidade Baixa	2
4	Criticidade Muito Baixa	1

Tabela 4 – Classificação das criticidades dos eventos informadas pelo SNORT.

Tipo da Classe	Descrição	Prioridade	Classificação no Software
attempted-admin	Tentativa de obter privilégios de administrador	alta	Nível 3
attempted-user	Tentativa de obter privilégios de usuário	alta	Nível 3
inappropriate-content	Conteúdo inadequado foi detectado	alta	Nível 3
policy-violation	Potencial Violação de Privacidade Empresarial	alta	Nível 3
shellcode-detect	Código executável detectado	alta	Nível 3
successful-admin	Obtenção de privilégios de administrador bem sucedido	alta	Nível 3
successful-user	Obtenção de privilégios de usuário bem sucedido	alta	Nível 3
trojan-activity	Trojan detectado	alta	Nível 3
unsuccessful-user	Obtenção de privilégios de usuário mal sucedido	alta	Nível 3
web-application-attack	Ataque a Aplicações Web	alta	Nível 3
attempted-dos	Tentativa de Denial of Service	média	Nível 2
attempted-recon	Tentativa de vazamento de informação	média	Nível 2
bad-unknown	Tráfego potencialmente ruim	média	Nível 2
default-login-attempt	Tentativa de login com usuário e senha padrão	média	Nível 2
denial-of-service	Detecção de ataque Denial of Service	média	Nível 2
misc-attack	Ataque Misc	média	Nível 2
non-standard-protocol	Detecção de protocolo ou evento não padrão	média	Nível 2
rpc-portmap-decode	Decodificação de uma consulta RPC	média	Nível 2
successful-dos	Denial of Service	média	Nível 2
successful-recon-largescale	Vazamento de informação em larga escala	média	Nível 2
successful-recon-limited	Vazamento de informação	média	Nível 2
suspicious-filename-detect	Arquivo suspeito detectado	média	Nível 2
suspicious-login	Tentativa de login com usuário suspeito detectado	média	Nível 2
system-call-detect	Chamada de sistema detectada	média	Nível 2
unusual-client-port-connection	Cliente utilizando uma porta não padrão	média	Nível 2
web-application-activity	Acesso a uma aplicação web potencialmente vulnerável	média	Nível 2
icmp-event	Evento ICMP genérico	baixa	Nível 1
misc-activity	Atividade Misc	baixa	Nível 1
network-scan	Detecção de scaneamento de rede	baixa	Nível 1
not-suspicious	Tráfego não suspeito	baixa	Nível 1
protocol-command-decode	Comando de protocolo genérico detectado	baixa	Nível 1
string-detect	Sequência suspeita detectada	baixa	Nível 1
unknown	Tráfego desconhecido	baixa	Nível 1
tcp-connection	Conexão TCP detectado	Muito baixa	Nível 1

Na **Tabela 4** estão descrito todos os eventos passíveis de detecção pelo SNORT e qual será a criticidade pelo módulo de conversão do cliente para o servidor, para que seja possível realizar a tomada de decisão de quais conjuntos de regras serão aplicados no cliente.

Baseada nas informações recebidas, a função responsável pela tomada de decisão no módulo servidor faz acesso ao banco de dados Mysql através da função de acesso ao SGBD. As regras presentes no banco de dados devem ser previamente cadastradas no sistema e visam refletir e implementar a Política de Segurança adotada pela corporação.

A grande vantagem da manutenção do conjunto de regras ser feita pelo próprio usuário do sistema é permitir que o sistema seja utilizado com qualquer firewall nativo. Porém, em um ambiente de clientes homogêneos, o servidor pode ser executado em qualquer sistema operacional que possua suporte a Java.

Após o envio pelo servidor do conjunto de regras a serem aplicadas, o cliente inicializa uma instância *bash* (interpretador de comandos) onde serão aplicadas as regras puramente como foram cadastradas no software. Esta ação é similar ao próprio usuário adicionando uma regra no firewall de forma manual, porém a grande vantagem que o software proporciona é o gerenciamento das regras de forma centralizada e baseada no contexto que o cliente se encontra. Este controle de forma manual seria inviável, pois o tempo perdido neste sincronismo seria imenso. Na **Tabela 5** são demonstrados exemplos que podem ser cadastrados no software.

Tabela 5 – Exemplos de regras para cada tipo de sistema operacional cliente.

Sistema Operacional	Exemplo de Regra
Windows	netsh advfirewall firewall add rule name="permitir8181" protocol=TCP dir=out localport=8181 action=allow
Linux	iptables -A INPUT –p tcp --dport 8181 –i eth1 –j ACCEPT
Unix	ipfw add 2 allow tcp from any to any 8181

Como o conjunto de regras cadastrados no Servidor *Firewall* de acordo com os níveis (*levels*) encontrados é de suma importância para a aplicação da Política de Segurança da corporação. É neste conjunto de regras que será possível evidenciar o acesso que a máquina poderá ou não realizar em determinado cenário. Com este objetivo,

foi criado um Painel de Controle, para que de forma simples e objetiva, o conjunto de regras possa ser cadastrado para cada nível de criticidade. Como já esclarecido este software possui 3 níveis de criticidade e, de acordo com a modelagem também já detalhada, os conjuntos de regras devem ser atrelados ao nível de criticidade que sejam pertinentes. Na **Figura 10**, pode-se visualizar o leiaute do Painel de Controle contendo os *botões Level 1, Level 2 e Level 3* utilizados para cadastramento dos grupos de regras.

Figura 10 – Painel de Controle do sistema.

Realizando acesso a qualquer nível (*level*), será apresentada uma tela, conforme se visualiza na **Figura 11**, onde é possível cadastrar o grupo de regras desejado. O nome do grupo é campo obrigatório para organização dos grupos, já a descrição é opcional.

Após o cadastramento, é possível visualizar na **Figura 12** a lista de grupos de regras cadastrados.

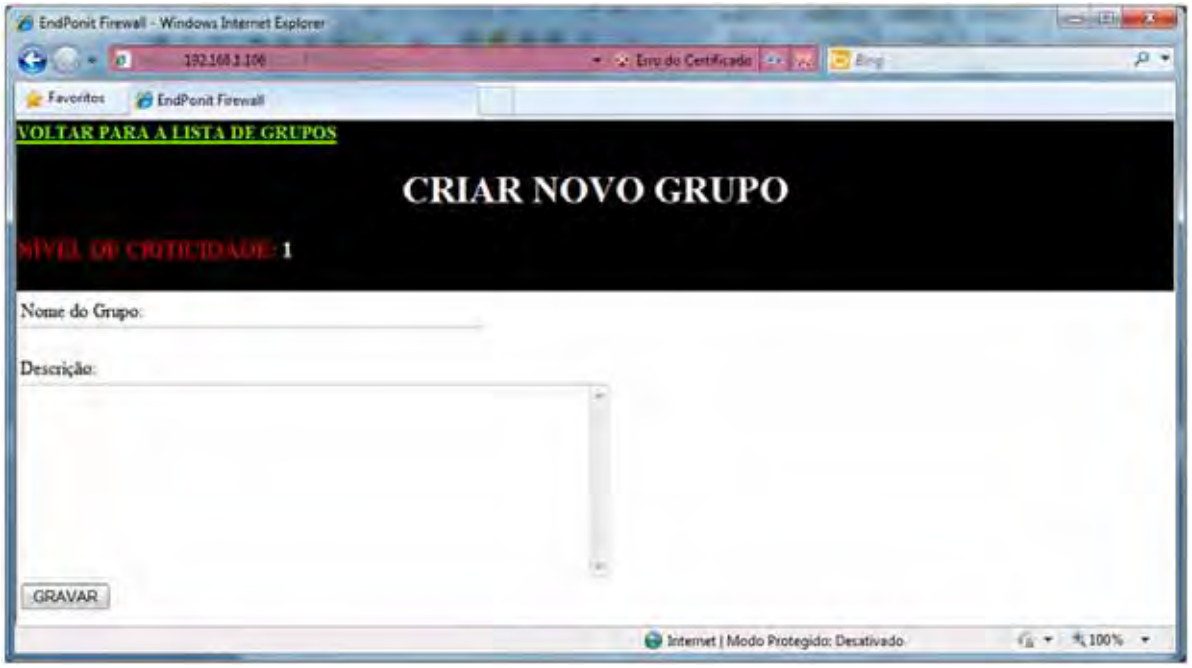

Figura 11 – Cadastro do Grupo de regras aplicadas no nível de criticidade 1.

Figura 12 – Lista de Grupos de Regras cadastrados.

5.4.4 Desenvolvimento do software: Módulo Cliente

O módulo cliente da aplicação foi desenvolvido na linguagem Java, com o objetivo de ser totalmente transparente ao usuário final. Este módulo como descrito na **Figura 8**, é responsável por realizar quatro ações: coletar informações do ambiente de rede utilizando o software SNORT, realizar a conversão dos níveis de criticidade informados pelo SNORT para os adotados pelo software, enviar e receber dados do servidor e aplicar o conjunto de regras recebido do servidor no firewall nativo da máquina.

Como o cliente será utilizado em redes onde o ambiente e as regras de segurança não são controláveis, o primeiro teste realizado pelo módulo cliente é tentar estabelecer uma conexão com o servidor firewall. Se esta tentativa de conexão for mal sucedida pode significar que há bloqueios de firewall ou algum roteamento inadvertido, o que pode ocasionar danos aos sistemas do equipamento cliente ou vazamento dos dados que este possui. Esta política visa garantir que em ambiente hostil, a máquina não seja comprometida por estar em um ambiente menos seguro. Então, o módulo cliente aplicará um conjunto de regras padrão que foram recebidas na última conexão válida e armazenadas no arquivo "c:\client\client.properties" do equipamento cliente, que bloqueará as demais portas que não sejam necessárias para demais tentativas de conexão com o Servidor *Firewall*. Na **Figura 13** é demonstrado o formulário web para cadastramento de Regras Padrão no Painel de Controle no servidor firewall.

Figura 13 – Grupo de regras padrão.

Depois de feita a conexão com o servidor, o módulo cliente ativa pela primeira vez o software SNORT e, como há a necessidade executar uma varredura na estrutura de rede para evidenciar a situação atual, o SNORT será executado por um tempo pré-determinado no Painel de Controle do servidor, que é repassado para o módulo cliente. Na **Figura 14** têm-se os Parâmetros do Cliente presentes no Painel de controle do servidor em detalhes, onde são controlados três parâmetros:

- Intervalo de tempo entre uma negociação e outra com o cliente;
- Intervalo de tempo entre as execuções do SNORT;
- Tempo total de execução da varredura da rede feita pelo robô SNORT.

Os parâmetros de tempo de execução do *robot* (programa automatizado que gerencia uma aplicação) do SNORT e o tempo de execução do SNORT são impactantes no desempenho global do cliente, pois, se os intervalos entre as execuções forem muito curtos isto sobrecarregará muito o equipamento cliente e se o tempo for longo demais, pode ser que existam ataques não detectados. Foram realizados testes quanto aos valores dos parâmetros e os apresentados na **Figura 14** se demonstram satisfatórios, não afetando o desempenho das demais aplicações utilizadas pelo usuário.

Dependendo da infraestrutura o administrador do sistema poderá ajustar os parâmetros para refletir a realidade da rede sem interferir no desempenho global do cliente.

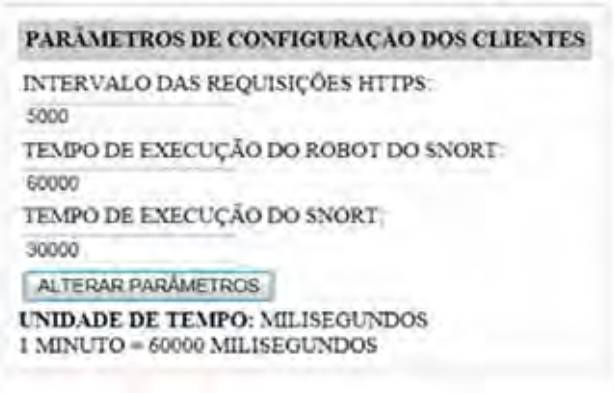

Figura 14 – Parâmetros configuráveis através do Painel de Controle do módulo servidor.

No módulo cliente não há nenhuma interface com o usuário, somente são exibidas em um *console bash* mensagens para identificação dos eventos. Na **Figura 15** se pode visualizar a ocorrência destes eventos e sua identificação. No item I indicado na **Figura 15** tem-se o comando de inicialização do programa, já no item II após a verificação de conexão com o servidor o *robot* do SNORT é iniciado. No item III o robot do SNORT encerra sua varredura, sendo importante salientar que as ações II e III foram temporizadas conforme os parâmetros indicados na **Figura 14**. No item IV o módulo cliente informa via função HTTPS os níveis de criticidade (*levels*) encontrados no cliente, no exemplo, foi encontrado o nível máximo de criticidade 3. Em seguida, no item V o servidor atualiza os parâmetros do cliente e, finalmente, no item VI pode-se visualizar a aplicação no firewall nativo do conjunto de regras recebidas do servidor.

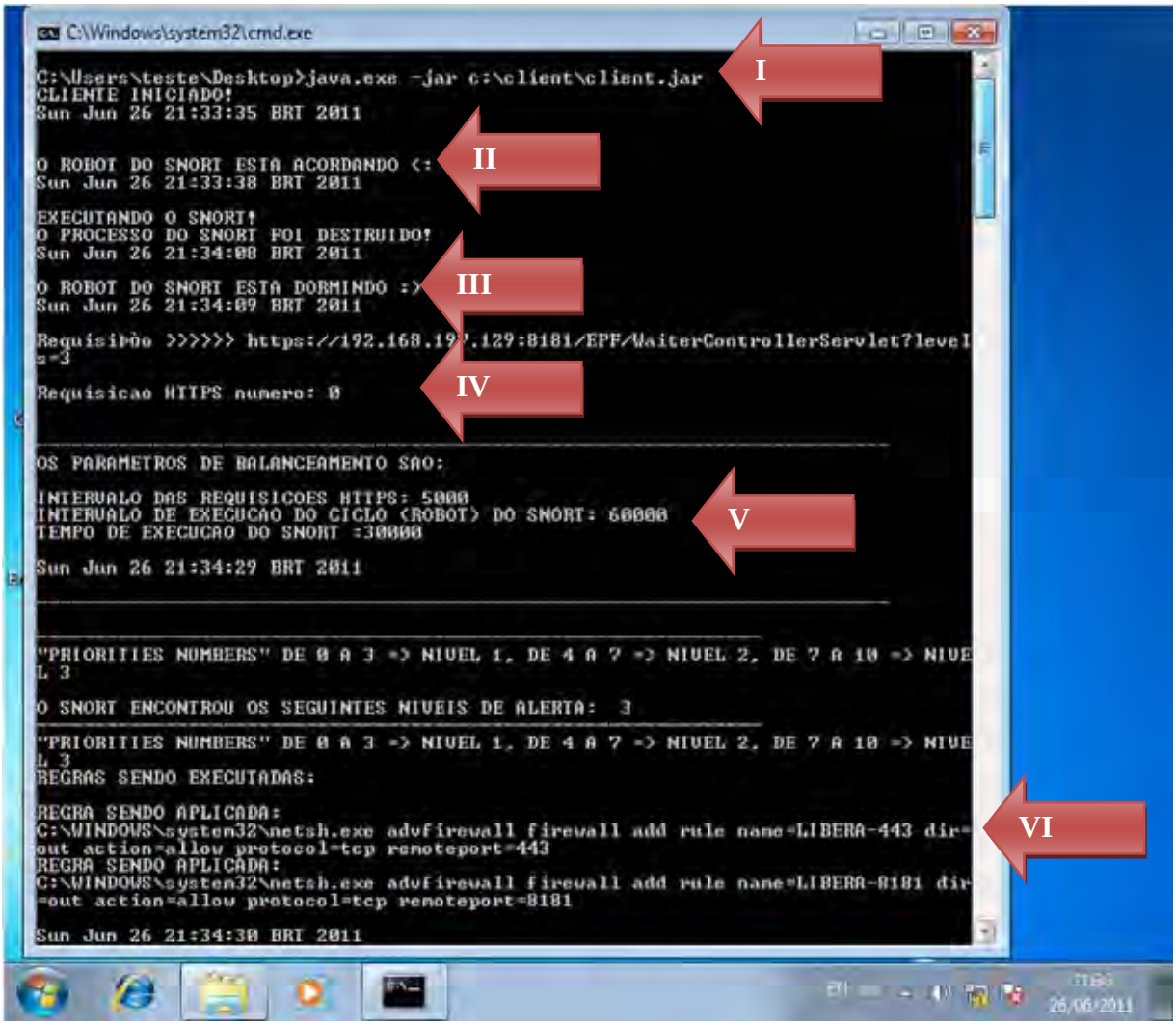

Figura 15 – Fases de execução do módulo cliente.

Depois de verificado o primeiro ciclo de execução do software, este aguardará o tempo determinado pelos parâmetros do software e executará a mesma rotina verificando se houve alteração no nível de criticidade informado. Se houver, o módulo cliente substituirá as regras aplicadas anteriormente pelo conjunto de regras necessário para o novo cenário, conforme se visualiza na **Figura 16**.

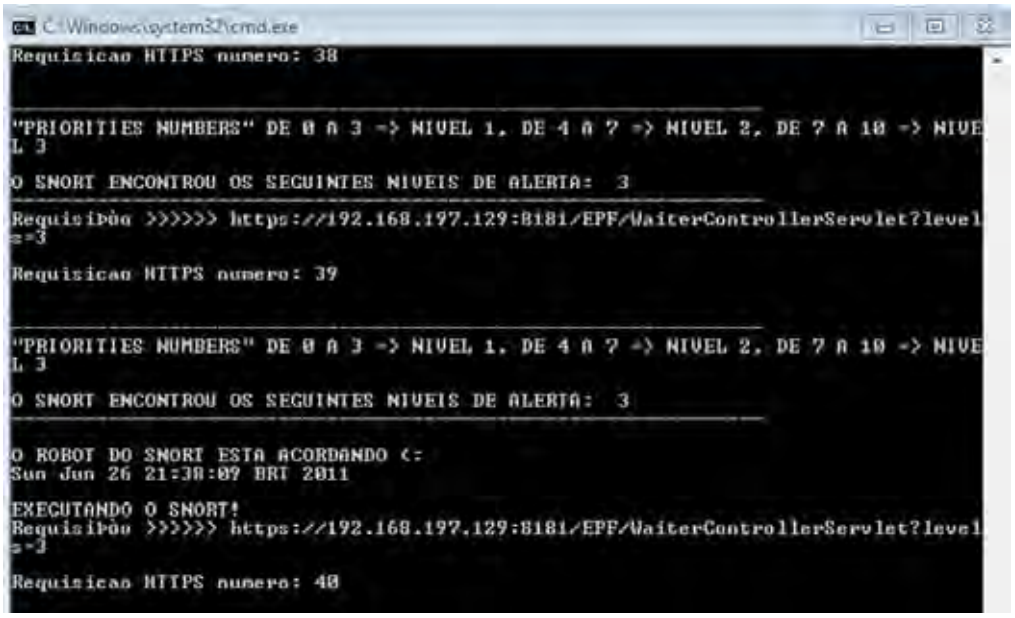

Figura 16 – Verificações subsequentes do módulo cliente.

5.5 CONSIDERAÇÕES FINAIS

Neste capítulo foram descritas as tecnologias utilizadas na implementação, bem como todas as fases e interfaces criadas no software. Como é possível notar, o software não contém em sua essência nenhuma amarra com nenhuma Política de Segurança pré-definida. Cada corporação pode implementar de acordo com suas regras de segurança as proteções que julgar necessárias de acordo com o nível de criticidade apresentado.

Como as regras são implementadas por grupos, a organização e manutenção das mesmas tornam-se mais eficiente e ao mesmo tempo mais acessível já que os grupos podem conter nomes que identifiquem e classifiquem as regras com clareza.

Como fator inerente ao conceito do software, toda a da troca de informação entre o módulo cliente e o módulo servidor é mantida confidencial e autentica pela utilização de criptografia e autenticação por chaves públicas e privadas, tornando a ferramenta útil em qualquer infraestrutura, independente do nível de segurança implementado nesta rede.

No próximo capítulo será demonstrado um cenário de teste criado para validar o comportamento da aplicação em situações reais de ameaças.

6 RESULTADOS

6.1 INTRODUÇÃO

Os experimentos realizados neste trabalho visam a criar um cenário que represente os ambientes encontrados nas redes privadas e públicas nas quais os equipamentos se conectam, e validar o comportamento do software. Como o software foi desenvolvido em dois módulos, será necessário validá-los de forma integrada.

Para que o ambiente de teste reproduza de forma fiel ambientes com maiores e menores níveis de criticidade, foram introduzidas máquinas virtuais, que são uma duplicata eficiente e isolada de uma máquina real (LAUREANO, 2006), criando assim o cenário para a execução dos testes.

6.2 CENÁRIO DE TESTES

Como não é possível validar este software utilizando um único cliente ou uma única rede, foi necessário criar um cenário que contivesse redes contendo alguns equipamentos, obrigando os tráfegos de dados dos equipamentos atravessarem o firewall servidor, podendo realizar testes de funcionalidade, bem como verificar qual o nível de criticidade informado pelo cliente, através do software SNORT, em cada cenário apresentado. Com este objetivo, foram criadas três redes, como se pode visualizar na **Figura 17**.

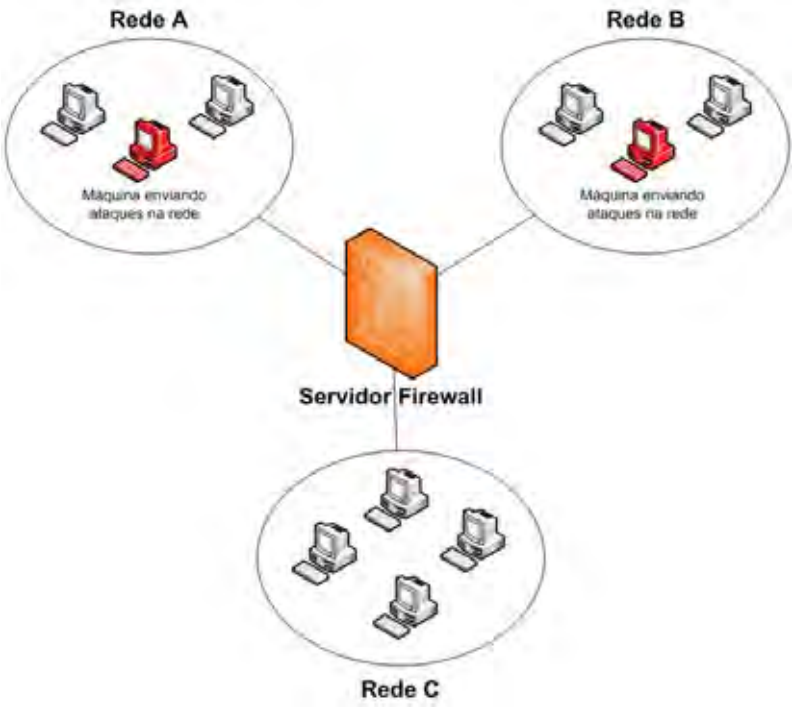

Figura 17 – Cenário de testes.

Para validação do módulo servidor, este foi instalado em um máquina Linux Debian juntamente o servidor de aplicações Glassfish e o servidor de banco de dados Mysql. Para permitir uma fácil operacionalização dos testes com os clientes foram criadas 10 máquinas virtuais, todas com sistema operacional Windows 7 Professional e o software de virtualização de máquinas VMWare Workstation. Nestas máquinas virtuais as únicas condições configuradas foram a instalação do módulo cliente e a ativação do firewall nativo do Windows.

Após a criação das máquinas virtuais, elas foram instaladas em três equipamentos reais e iniciadas através do software VMWare, visando a simular a criação das três redes proposta na **Figura 17**. Para garantir a conectividade da infraestrutura, cada equipamento real possui uma ligação física com o servidor firewall.

Para efeitos de testes existem duas redes contendo atividades suspeitas (A e B) e uma contendo atividades normais (C), conforme se visualiza na **Tabela 3**. Assim será possível mensurar quais são as respostas do software em cenários distintos.

Visando a produzir níveis de criticidade altos, foram instalados em duas máquinas virtuais das redes A e B software para simular comportamentos suspeitos para que o IDS possa acusar estas ocorrências. Na máquina virtual da rede A foram instalados os softwares NMAP (utilizado para realizar scanner de redes) e WinArpSpoofer (utilizado

para modificar a tabela ARP das máquinas clientes e modificar o encaminhamento de pacotes). Já na máquina virtual da rede B foi desenvolvido um *robot* para executar o e*xploit* dcom.c, que é um software que explora uma vulnerabilidade conhecida na função *Remote Procedure Call* (RPC) do Windows.

Na rede C, como já citado, não houve incremento de nenhuma funcionalidade. Todas as máquinas se comportam de forma legítima na rede, não havendo nenhuma situação de possível ameaça.

Tabela 6 – Redes criadas no cenário de testes.

Rede	Quantidade de Clientes	Existência de atividades suspeitas?
A	3	Sim
B	3	Sim
C	4	Não

Um cuidado adicional foi tomado para que o servidor Firewall não realizasse roteamento entre as redes, pois este comportamento poderia distorcer os resultados apresentados pelos softwares clientes.

Como os ciclos de análise do cliente são executados de 1 em 1 minuto, o cenário foi testado por 10 ciclos em situações de ataque eminente e em situações de suspensão destes ataques.

6.3 RESULTADOS OBTIDOS

Durante os ciclos onde o software foi analisado, percebeu-se que nas Redes A e B foram evidenciados níveis de criticidade 2 e 3, respectivamente em todos os ciclos em que havia atividades maliciosas nas máquinas virtuais modificadas, sinalizando que o software SNORT havia detectado tentativas de ataque naquela rede. O nível da criticidade somente variava pela quantidade de simulações de ataque, pois nos ciclos em que estas aplicações foram finalizadas, o software deixou de informar ao servidor níveis de criticidade, aplicando assim a regra padrão vinda do servidor.

Já na Rede C, o que se pode evidenciar foi que, na maioria das varreduras, o software SNORT não evidenciou nenhum nível de criticidade. Em raros casos, nos ciclos avaliados, o software SNORT encontrava alguns falso-positivos (situações detectadas pelos software como atividade maliciosa, mas na verdade eram acesso legítimos) e por isso era enviado nível de criticidade 1 para o servidor, aplicando-se assim no cliente as regras cadastras neste grupo de regras no servidor.

7 CONCLUSÃO

7.1 CONCLUSÕES GERAIS

Este trabalho visou propor o estudo de novas formas de garantir a implementação das Políticas de Segurança definidas pelas corporações de uma forma eficaz e que não consumam muito tempo do administrador da infraestrutura lógica, pois quando isto acontece, a reação natural é a descontinuidade do uso da ferramenta. No contexto atual, ferramentas que não possuam um gerenciamento simples e objetivo, perdem mercado, pois, ao invés de tornarem uma solução, tornam-se mais uma obrigação para o administrador, como aponta a pesquisa feita por Santos e Kon (2008).

A ferramenta desenvolvida visa a trazer um novo conceito de centralizar as regras, divididas em níveis de criticidade e a partir deles tomar decisões para a defesa do host. De uma forma geral cumpre o propósito para qual foi desenhada e desenvolvida, abrindo novas possibilidades de implementações.

7.2 TRABALHOS FUTUROS

Para trabalhos futuros, há a necessidade de um levantamento menos empírico do impacto que o software cliente gera no desempenho global da máquina cliente, já que o software SNORT necessita verificar o ambiente de tempos em tempos, e como balancear os parâmetros de forma que esta verificação não inviabilize o uso de outras aplicações.

Outro fator que necessita de um maior levantamento de dados é a definição de quais regras deveriam ser aplicadas a cada nível de criticidade para garantir que a ameaça apontada pelo SNORT possa ser realmente neutralizada de forma eficaz. Através do cenário de testes proposto é possível coletar dados, utilizando o banco de dados Mysql presente no servidor, de quais os tipos de ataques foram detectados, e qual a eficiência

das regras aplicadas para esta situação, pois a implementação das regras dependerá da criticidade que está envolvida.

Um último fator que poderá ser analisado em trabalhos futuros é o acoplamento de outras ferramentas de checagem anti-malware, fazendo com que o software tenha funcionalidade razoavelmente parecida com os Endpoints existentes no mercado, com a inovação da verificação de contexto realizada pelo software SNORT.

REFERÊNCIAS

BARBOSA, A. **Um sistema para análise ativa de comportamento de Firewall**.2006, 121 f Dissertação (mestrado em Engenharia) – Universidade de São Paulo São Paulo, 2001.

CHAURE, Rupali. **An Implementation of Anomaly Detection Mechanism for Centralized and Distributed Firewalls**. NRI Institute of Information Science and Technology. 2010. Bhopal, India.

Computer Emergency Response Teams - CERT.. **Cartilha de Segurança para Internet**.2006 Disponível em: http://cartilha.cert.br/conceitos/sec8.html. Acesso em: 14 de Abril de 2010.

DOMINGUES, A. **Segurança de redes com uso de um aplicativo Firewall nativo do Sistema Linux**. 2006 Dissertação (mestrado em Engenharia) – Universidade de São Paulo São Paulo, 2006

ENDPOINTa. **Endpoint-Protection Products**. 2008. Disponível em: http://www.windowsitpro.com/article/product-review/endpoint-protection-products/1.aspx, Acesso em: 22 de Março de 2011.

ENDPOINTb. **Endpoint Security**. 2005, Disponível em: http://searchmidmarketsecurity.techtarget.com/definition/endpoint-security. Acesso em: 22 de Março de 2011.

FIREWALLa. **Firewall - Segurança nas Redes**. 2007 Disponível em: http://www.gta.ufrj.br/grad/07_1/firewall/index_files/Page350.htm. Acesso em: 28 de Janeiro de 2010.

FIREWALLb. **Firewall 2010**. Disponível em: http://pt.wikipedia.org/wiki/Firewall. Acesso em: 28 de Janeiro de 2010.

FIREWALLc. (2010), **Firewall – Gerações.** Disponível em: http://segurancadredes.hdfree.com.br/firewall/quarta_geracao.html. Acesso em: 27 de Março de 2010.

FOROUZAN, Behrouz A., **Comunicação de dados e Redes de Computadores**. 2006, 3ª edição, Bookman.

IDGNOW. **Estudo sugere que configuração falha em roteadores facilita ataques online**. 2007. Disponível em: http://idgnow.uol.com.br/seguranca/2007/02/15/idgnoticia.2007-02-15.1659526620/IDGNoticia_view/. Acesso em: 11 de Maio de 2011.

Infra-Estrutura de Chaves Públicas Brasileira - ICP-Brasil. 2011 Disponível em: http://www.iti.gov.br/twiki/bin/view/Certificacao/CertificadoConceitos. Acesso em: 10 de Junho de 2011.

KUROSE James F., ROSS, Keith W. (2010), **Redes de Computadores e a Internet 5a. Edição**. 2010. Editora Pearson.

Laureano, Marcos. **Máquinas Virtuais e Emuladores**. Ed. Novatec, pg. 15, 2006.

LIMA, Marcelo Barbosa. **Provisão de Serviços Inseguros Usando Filtros de Pacotes com Estados**. 2000, 128 f Dissertação (mestrado em Ciências da Computação) - Universidade Estadual de Campinas, 2000.

MONTEIRO, P. **Gestão Centralizada de Firewalls Distribuídas em Ambientes Heterogéneos**.2006,135 f. Dissertação (mestrado em Engenharia Elétrica) - Universidade do Porto, 2006

MORAES, Stephanie. **Tutorial do J2EE**. Tradução Altair Dias Caldas de Moraes, Carlos Augusto Caldas de Moraes. Rio de Janeiro: Campus, 2002.

MYSQL. **Visão Geral do Sistema de Gerenciamento de Banco de Dados MySQL**. 2011 Disponível em: http://xoops.net.br/docs/mysql/manual/ch01s02.php. Acesso em: 11 de Maio de 2011.

NAKAMURA, Emilio. **Segurança de Redes: em ambientes cooperativos. 1º Ed**. São Paulo: Novatec, 2007.

Neto, Alvaro A. **Uma Abordagem Para gestão da produção de software em larga escala baseada em metaprocesso**. 2009. 249 f. Dissertação (Doutorado em Computação Aplicada) – INPE São José dos Campos, 2009.

OGLOBO. Mercado **de software de segurança deve crescer 11% em 2010, diz Gartner**. 2010 Disponível em: http://oglobo.globo.com/economia/mat/2010/08/16/mercado-de-software-de-seguranca-deve-crescer-11-em-2010-diz-gartner-917410589.asp. Acesso em: 31 de Março de 2011.

RANUM, Marcus J. **Tales From The Early Days of the Firewall**. 2010. Disponível em: http://www.ranum.com/security/computer_security/archives/firewall-early-days.pdf. Acesso em: 14 de Abril de 2010.

SANTOS, Paula Oliveira; KON, Fabio. **Metodologias e Ferramentas para Avaliação da Qualidade de Sistemas Web de Código Aberto com Respeito à Usabilidade**. 2008, 70 f. Dissertação (mestrado em Computação) - Instituto de Matemática e Estatística da Universidade de São Paulo, 2008.

SÊMOLA, Marcos. **Gestão da Segurança da Informação: Uma visão executiva. 1ª Ed**. Rio de Janeiro: Campus, 2003.

SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. **Sistemas operacionais com Java 7ª Ed**. Rio de Janeiro: Elsevier, 2008.

SNORT. **O Snort. 2011**. Disponível em: http://www.snort.com.br/snort.asp. Acesso em: 11 de Maio de 2011.

SOARES, Luiz Fernando Gomes. **Redes de computadores: das LANs, MANs e WANs às redes ATM**. Rio de Janeiro: Campus, 1995.

SOPHOS. **Endpoint Assessment and Control.** 2011. Disponível em: http://www.sophos.com/security/topic/your-nac-compliance/how-much-control.html. Acesso em: 11 de Maio de 2011.

SYMANTEC. **Symantec Endpoint Protection 11.0.** 2011. Disponível em: http://www.symantec.com/pt/br/business/products/family.jsp?familyid=endpointsecurity, Acesso em: 11 de Maio de 2011.

Tanenbaum, Andrew S. **Redes de Computadores - Tradução da 4ª Edição.** 2003. Editora Campus / Elsevier.

TECHWORLD. **Ultimate guide to network access control products**. 2008 Disponível em: http://review.techworld.com/security/3227897/ultimate-guide-to-network-access-control-products/?pn=1. Acesso em: 11 de Maio de 2011.

TORTELLO, Martin. **Aprenda J2EE em 21 dias: com EJB, JSP, servlets, JNDI, JDBC e XML**. Tradução João Eduardo Nóbrega Tortello. São Paulo: Pearson Education, 2003.

Zaliva, Vadim. **Applying static code analysis to firewall policies for the purpose of anomaly detection.** Cornell University Library, Nova York, EUA, 2009.