UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

CLEYTON STANG

SISTEMA INTELIGENTE DE DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES

CRICIÚMA, JULHO DE 2009

CLEYTON STANG

SISTEMA INTELIGENTE DE DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES

Trabalho de Conclusão do Curso apresentado para obtenção do Grau Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador: Prof. MSc. Paulo João Martins

Co-orientadora: Profa. MSc. Priscyla Waleska Targino de Azevedo Simões

CRICIÚMA, JULHO DE 2009

CLEYTON STANG

Protótipo de um Sistema Inteligente de Detecção de Intrusão em Redes de Computadores

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Prof. MSc. Rogério Antônio Casagrande
Coordenador Adjunto do Curso de Ciência da Computação

Banca Examinadora:

Prof. MSc. Paulo João Martins (UNESC)
Orientador

Profa. MSc. Priscyla Waleska Targino de Azevedo Simões (UNESC)
Co-Orientador

Prof. MSc. Rogério Antônio Casagrande (UNESC)

Profa. MSc. Merisandra Côrtes de Mattos (UNESC)

Aos meus pais, Tarcísio Michels Stang e Mariléia Wessler Kuerten Stang, meus colegas e amigos pela força.

AGRADECIMENTOS

Agradeço a Deus, perante todas as coisas, pelo auxílio nas minhas decisões e por me confortar nas horas difíceis.

Agradeço também:

A minha família, por todo o incentivo e apóio que me deram no decorrer do curso, por sempre me oferecerem ajuda nos momentos difíceis e por me proporcionar uma boa educação.

A minha namorada Cristiny, por estar sempre ao meu lado, pelo seu amor, carinho, incentivo e compreensão, além de sempre me oferecer ajuda independente de dia e hora.

Aos meus amigos pelo apoio, amizade e pelos momentos alegres que passamos ao longo da nossas vidas.

Aos meus colegas de classe pela ajuda durante a implementação efetuada neste trabalho de pesquisa.

Agradeço aos grandes incentivadores desta pesquisa que não mediram esforços para auxiliar-me na conclusão da mesma, meus orientadores Paulo e Priscyla, contribuindo também, além do conhecimento, com sua amizade.

Enfim, a todos que contribuíram de alguma forma para a execução desse trabalho.

“Muitos dos fracassos da vida são pessoas que não perceberam o quão perto elas estavam do êxito quando elas desistiram.”

(Thomas A. Edison)

RESUMO

O projeto de segurança de redes costuma abranger uma boa política da mesma onde entram *firewalls,* antivírus, bloqueios de portas entre outros, porém somente estes podem não ser suficientes, pois caso venham a falhar, o sistema protegido poderá ficar exposto ao agente invasor. Para solucionar esta questão, foram criados os Sistemas de Detecção de Intrusão (IDS), que possuem a função de encontrar invasores mesmo que tenham passados pelo perímetro de segurança. Estes sistemas são baseados em assinaturas que identificam um ataque, caso sejam muito restritas podem ocasionar falsos negativos ou falsos positivos. Com o objetivo de minimizar estes erros, foi desenvolvido nesta pesquisa um sistema IDS baseado em agentes inteligentes. Na modelagem do agente utilizou-se a teoria dos Fatores de Certeza (FC) para redução da incerteza presente nas situações geradas. A metodologia desta pesquisa contou principalmente com as seguintes etapas: modelagem do sistema de detecção de intrusão, do agente inteligente, da matemática dos fatores de certeza, e implementação do sistema. Assim, esta pesquisa resultou em uma aplicação desenvolvida na linguagem Java a partir do ambiente NetBeans IDE 6.1, para representar um IDS baseado em agentes inteligentes e na teoria dos Fatores de Certeza. Com o intuito de realizar os testes, desenvolveu-se um sistema de IDS tradicional para comparação dos resultados, a partir de uma situação de ataques hipotéticos que o sistema poderia receber. Nesta análise verificou-se que os resultados foram satisfatórios considerando que o sistema inteligente conseguiu diminuir significativamente o número de falsos positivos além de identificar ataques automaticamente.

Palavras-chave: Inteligência Artificial, Segurança de Redes, Agentes Inteligentes, Fatores de Certeza, Sistema de Detecção de Intrusão Inteligente.

ABSTRACT

The design of network security policy usually cover a good go of it where firewalls, antivirus, locks of doors among others, but they may just not be enough, because if they fail, the protected system could be exposed to the agent attacker. To resolve this issue, we created the Intrusion Detection Systems (IDS), which have the same function to find invaders that have passed the security perimeter. These systems are based on signatures to identify an attack, if narrow can cause false negatives or false positives. Aiming to minimize these errors, was developed in this research an IDS system based on intelligent agents. In modeling the agent used the theory of certainty factors (CF) to reduce the uncertainty in this situation generated. The methodology of this research has mainly the following steps: modeling the intrusion detection system, the intelligent agent, the mathematics of the factors of certainty, and implementation of the system. Thus, this research resulted in an application developed in Java environment from the NetBeans IDE 6.1, to represent an IDS based on intelligent agents and the theory of certainty factors. In order to perform tests, has developed a system of traditional IDS to compare results from a hypothetical situation of attacks that the system could receive. This analysis found that the results were satisfactory considering that the smart was able to reduce significantly the number of false positives and identify attacks automatically.

Keywords: Artificial Intelligence, Network Security, Intelligent Agents, Certainty Factors, intrusion detection systems Intelligent.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS

BIND	*Berkeley Internet Name Domain*
CIDIF	*Common Intrusion Detection Framework*
CPU	*Central Processing Unit*
DIDS	*Distributed Intrusion Detection System*
DNS	*Domain Name System*
FTP	*File Transfer Protocol*
HD	*Hard Disk*
HIDS	*Hybrid Intrusion Detection Systems*
HTTP	*Hypertext Transfer Protocol*
IA	*inteligência artificial*
IBGE	*Instituto Brasileiro de Geografia e Pesquisa*
IDS	*Intrusion Detection System*
IP	*Internet Protocol*
ISP	*Internet service provider*
LAN	*Local Area Network*
NICS	*Network Interface Card*
NIDIS	*Network Intrusion Detection Systems*
PIPS	*Proactive Intrusion Prevention Systems*
POP3	*Post Office Protocol version 3*
SMTP	*Simple Mail Transfer Protocol*
TCP	*Transfer Control Protocol*
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

Segurança é um fator importante em uma rede, porém, mesmo que esta possua uma boa política de segurança, com ferramentas de *firewall* e antivírus, não está imune a um ataque bem sucedido de pacotes invasores, devido aos inúmeros tipos que podem ocorrer e das mais diversificadas formas. Se a rede de computadores interna não possuir um software que monitore e alerte sobre o tráfego de pacotes indesejados, a mesma pode ficar vulnerável aos mesmos, que podem por conseqüência chegar a um *host* específico, a um conjunto deles ou até mesmo ao servidor, acessando desta forma suas informações e comprometendo a integridade do ambiente.

Atualmente existem alguns softwares para o controle de tráfego, chamados de *Intrusion Detection System* (IDS), contudo estes softwares não satisfazem integralmente este determinado tipo de controle, pois como dependem de regras e caso estas não sejam criadas de forma correta o sistema pode gerar alertas falsos.

Segundo Northcutt (2002) algumas ferramentas utilizam a filtragem de pacotes com regras genéricas e fazem vários alertas de pacotes inofensivos que são identificados como falsos positivos, outras possuem esta regra de forma muito específica diminuindo o número de falsos positivos, mas aumentando o número de falsos negativos, quando um pacote infectado não é localizado.

Neste sentido, este trabalho propõe uma ferramenta que utilize a abordagem de agentes inteligentes integrada a um sistema de detecção de intrusão. Esta ferramenta inteligente tem por objetivo armazenar em sua base de conhecimento[1] os eventos gerados pelos IDSs, que poderão ser utilizados para a análise de eventos futuros melhorando a performance na detecção de invasões e conseqüentemente eliminando alguns problemas de

[1] A base de conhecimento é o local responsável por armazenar as informações necessárias para a resolução dos problemas a que o sistema se propõe.(REZENDE et al, 2005).

tráfego de pacotes, como os supracitados.

1.1 OBJETIVO GERAL

Desenvolver um protótipo de uma ferramenta Intrusion Detection System (IDS) que utilize uma técnica dos agentes inteligentes para monitorar redes de computadores.

1.2 OBJETIVOS ESPECÍFICOS

Esta pesquisa possui alguns objetivos específicos que são:

a) enunciar conceitos sobre políticas de segurança;
b) relatar conceitos sobre inteligência artificial;
c) compreender o funcionamento e os métodos de softwares Intrusion Detection System (IDS);
d) analisar as características inerentes do desenvolvimento de agentes inteligentes;
e) utilizar uma técnica de agentes inteligentes, FC, e IDS’s para monitorar o controle do tráfego de redes, no intuito de localizar e aplicar uma ação sobre os pacotes invasores;
f) aplicar os métodos mencionada a uma rede para verificar seu funcionamento.

1.3 JUSTIFICATIVA

Nos últimos tempos houve um crescimento considerável com relação à utilização da Internet, sendo que o número de informações consideradas importantes que são inseridas em servidores ou em computadores convencionais também acompanhou esse crescimento, estas informações podem estar de certa forma vulneráveis a pessoas de má índole ou não autorizadas, necessitando assim de alguma proteção.

Segundo Wadlow (2000) a segunda metade dos anos 1990 foi uma época de surpreendente desenvolvimento da Internet, como também da utilização de redes em empresas. Capacidades que eram um passatempo em 1995 passaram a ser sistemas críticos para atividades profissionais em 1999 e essa tendência não mostra sinais de reduzir sua velocidade de crescimento. Com esse crescimento, ocorreu um aumento de crimes, fraudes e outros abusos na rede.

Assim, para que as informações possam estar seguras e consistentes, é necessária a incorporação de uma política de segurança. Um destes mecanismos que se torna indispensável para garantir tal política são os sistemas Intruction Detection System (IDS). Estes sistemas possuem a função de monitorar a rede em busca de pacotes que ofereçam algum risco à mesma e notificar o administrador desta sobre a existência do mesmo, porém estes ainda apresentam certa defasagem na questão de detecção e análise de pacotes.

Em virtude desta defasagem, pesquisadores vêm fazendo estudos e pesquisas no intuito de inserir nestas ferramentas técnicas de Inteligência Artificial, mais especificamente de agentes inteligentes. Esta modificação faz com que o software de monitoração trabalhe da mesma forma que o administrador da rede trabalharia, ou seja, fazendo o trabalho árduo que o mesmo teria que fazer ao verificar todas as tentativas de invasão que fossem identificadas, utilizando para isto os conhecimentos deste administrador.

Segundo Rezende (2005) agentes são personagens computacionais que atuam de acordo com um *script* definido, direta ou indiretamente, por um usuário. Eles podem atuar

isoladamente ou em comunidade formando sistemas multiagentes. O uso de agentes vem aumentando, como uma maneira de lidar com um mundo onde informações e conhecimentos crescem a uma velocidade humanamente impossível de lidar.

Tendo em vista que o número de pacotes que circulam em uma rede é grande e os ataques ou tentativas de invasões podem acontecer de diversas formas, a necessidade desta ferramenta dita como inteligente torna-se indispensável, pois seria inviável um administrador analisar cada pacote intitulado como duvidoso.

Contudo, por meio das técnicas de IDS's e agentes inteligentes integrados em uma mesma ferramenta, pode-se reduzir os problemas provenientes de uma invasão, aumentar a segurança da rede e reduzir o número de pacotes que o administrador tenha de ajuizar, facilitando assim o seu trabalho.

1.4 ESTRUTURA DO TRABALHO

Neste tópico será apresentada uma síntese dos capítulos que compõem este trabalho.

O primeiro capítulo apresenta uma visão ampla do trabalho, onde é relatada sua importância, objetivo geral, objetivos específicos, a justificativa e a estrutura do mesmo.

Os fundamentos sobre a importância da segurança das informações, os métodos utilizados atualmente para se garantir tal segurança, a incorporação de um sistema de detecção de intrusão para aumentar a confiabilidade deste ambiente, assim como suas características, conceitos e abordagens estão descritos no segundo capítulo.

O terceiro apresenta a fundamentação teórica necessária para integrar o agentes inteligentes ao sistema IDS supracitado. Dentro deste encontra-se a definição de agentes, ambientes que o mesmo pode ser inserido, taxonomias e propriedades.

É apresentado no quarto capítulo os fundamentos sobre as técnicas dos fatores de certaza que são utilizados para diminuir as incertezas geradas pelo sistema.

No quinto capítulo são apresentados os trabalhos correlatos desenvolvidos nos últimos tempos por pesquisadores e acadêmicos.

É apresentado por fim, no sexto capítulo, o trabalho desenvolvido, relatando detalhada mente todos os processos utilizados para implementação do sistema proposto.

Tendo em vista as informações acima, a partir do próximo capítulo tem-se a fundamentação teórica necessária para o desenvolvimento do projeto de pesquisa proposto, onde é iniciado com o estudo sobre segurança das informações e técnicas de obtenção da mesma.

2 IMPORTÂNCIA DA SEGURANÇA DAS INFORMAÇÕES PESSOAIS E EMPRESARIAIS

Neste capítulo será relatado à importância da segurança das informações, principalmente as que são armazenadas em servidores ou computadores convencionais, devido às suas possíveis vulnerabilidades. Serão levantadas algumas formas de proteção destas informações assim como a necessidade de um sistema de detecção de intrusão para usuários finais e corporações, mostrando suas funcionalidades, características e boas práticas para a implantação das técnicas que serão abordadas.

2.1 SEGURANÇA DAS INFORMAÇÕES INSERIDAS EM REDES DE COMPUTADORES

Nos últimos tempos o número de pessoas que possuem acesso à Internet cresceu de forma espantosa. Em uma pesquisa realizada pelo Instituto Brasileiro de Geografia e Estatística (IBGE) mostra que em 2006 mais de 16% da população brasileira já possui acesso a Internet. Esta realidade vem se tornando cada vez mais comum devido ao fácil acesso ao computador que atualmente possui um custo de aquisição menor em relação ao mesmo há alguns anos atrás, esta característica também pode-se ser vista em relação ao acesso à web.

Este acesso possibilita vários benefícios para seus usuários, como um maior acesso às informações e notícias de todo o mundo, espalhadas em vários sites disponíveis, você pode conhecer diversos lugares bastando apenas dar um *click* no mouse. Também serve de auxílio nos estudos devido à enorme diversidade de assuntos para pesquisa que a mesma pode nos oferecer, tornando-se uma enorme biblioteca virtual.

As vantagens não são exclusivas de usuários finais, as empresas ou pessoas jurídicas, também podem possuir as mesmas com a utilização da web. Muitas destas empresas estão utilizando deste recurso para aumentar suas vendas e melhorar o marketing, disponibilizando seus produtos ao acesso dos clientes para consulta e compra sem sequer sair de casa. Hoje a quantidade de sites de vendas é enorme e a quantidade de pessoas que compram pela Internet também.

Em vista destas facilidades virtuais, algumas pessoas de má índole estão usando a mesma para aplicar golpes virtuais, e ter acesso as informações valiosas de pessoas ou empresas, como por exemplo, senhas bancárias. O número de pessoas que aplicam estes crimes virtuais vem aumentando, sem contar com os *trojans* [2] e vírus [3] que podem se instalar em computadores ou em um conjunto deles, ou seja, em uma podendo ocasionar uma série de danos.

Antes de iniciar qualquer projeto de segurança em uma rede empresarial, pessoal ou qualquer outro ambiente, é necessário uma boa análise, para definir o que se deseja proteger. Cada ambiente possui características particulares exigindo então uma política de segurança específica na qual possa atender uma grande parte de suas necessidades. O mesmo projeto de segurança aplicado em ambientes diferentes pode trazer resultados bastante distintos. Em um determinado ambiente uma falha pode resultar a parada de um servidor de e-mails por alguns minutos ou horas, por exemplo, não causando prejuízos, já em outro ambiente uma parada de um servidor Web que alimenta um site de venda de mercadorias pode ocasionar uma perda financeira considerável (WADLOW, 2000).

Como podemos os cuidados tomados podem ser considerados bons para alguns

[2] Os *Trojans* são programas que infectam num computador sem que o utilizador, e fornecem algum tipo de cesso para o computador ser utilizado por um estranho. Embora tecnicamente não sejam vírus, eles encontram-se na mesma categoria de programas (NORTHUCUTT, 2002).

[3] Um vírus de computador é um programa malicioso desenvolvido por programadores que faz cópias de si mesmo e tenta se espalhar para outros computadores na tentativa de corromper, apagar ou modificar arquivos do sistema (NORTHUCUTT, 2002).

casos e ruim para outros. Para determinar a qualidade da segurança de uma rede não basta apenas analisar como esta se comporta ou que componentes ela utiliza, é necessário primeiramente analisar o que está sendo protegido e que tipo de ameaça a mesma pode sofrer.

Em uma organização, por exemplo, é necessária a garantia da consistência e o sigilo total de seus dados em relação a invasores, não permitindo a também entrada *trojans* e vírus, pois podem infectar o ambiente computacional e ocasionar problemas ao mesmo.

Os dados de uma empresa são seu maior patrimônio, e estas informações estão cada vez mais sendo armazenadas em servidores. Em uma matéria feita pela revista PC WOLD diz que no Brasil o volume vendido de equipamentos de armazenamento corporativo somou-se em 2004 sete petabytes, ou seja, sete mil terabytes que equivalem a quase 70 vezes os dados armazenados da Biblioteca do Congresso Americano. De 2002 para 2004 o volume de dados dobrou e a expectativa é que quadriplique até 2009.

Com estas informações podemos perceber a importância inerente à segurança das informações, tendo em vista que estas estão sendo cada vez mais utilizadas pelas empresas em formato digital.

Segundo pesquisa realizada pelo instituto de pesquisas Computer Economics no ano de 2005 , mostra que os prejuízos causados por pragas virtuais foram somados em 14,2 bilhões de dólares (COMPUTER ECONOMICS, 2006). Isto leva a ser considerada a necessidade do investimento em equipamentos tanto para a integridade dos dados como também para a garantia de uma disponibilidade do sistema da empresa.

Mesmo com toda a segurança que é atribuída à rede, quem garante que esta não venha a falhar? Uma empresa que tenha uma grande necessidade da integridade de seus dados precisa de algo que indique caso esta falha venha a acontecer, podendo assim eliminar o invasor antes que o mesmo cause algum dano.

Um dos sistemas utilizados para este tipo de verificação é o *Intrusion Detection*

System (IDS), o qual tem o papel de identificar caso um invasor esteja trafegando na rede. Este sistema será um dos principais focos desta pesquisa.

2.2 NECESSIDADE DA DETECÇÃO DE INTRUSÃO PARA A SEGURANÇA DA INFORMAÇÃO

Quando nos conectamos à Internet estamos nos conectando a diversas redes de computadores e tendo acessos a determinadas partes da mesma como, por exemplo, e-mails, sites e outros, podendo utilizar todos os recursos que estes componentes podem nos trazer. Porém, não podemos deixar de analisar que como podemos acessar tais ambientes externos, outras pessoas também podem acessar o nosso ambiente interno.

Para conseguir certa privacidade em relação a acessos indesejados, é incorporado mecanismos de segurança nos servidores e roteadores onde são usados softwares que agem em conjunto para isolar uma determinada parte da rede ou restringir algumas informações.

Muitos podem pensar que somente estas precauções são suficientes para manter a rede segura contra ataques, mas infelizmente não são. Há uma grande necessidade de um sistema que venha a alertar caso uma tentativa de invasão esteja sendo iniciada ou uma que já tenha sido concluída, para que o administrador do ambiente computacional possa estar contornando o problema antes que o mesmo venha a se tornar maior.

O IDS é um sistema de alarme que monitora a rede onde ele está inserido. Sem este sistema de detecção de Intrusão, nunca se saberá se o sistema está prestes a ser invadido ou se já esta sendo alvo de uma invasão. O objetivo destes sistemas é buscar e alertar sobre suspeitas de invasões, atividades duvidosas e ataques que já estejam em andamento (THOMAS, 2007).

Muitos ataques são difíceis de serem flagrados devido à forma como os quais

agem. Estes podem envolver várias etapas ou fases, nas quais podem ser executados em tempos diferentes. Por exemplo, o mesmo poderia começar disparando um *scam* [4]que envia uma consulta DNS contendo uma solicitação de *version.bind* a cada endereço IP em determinado intervalo. Alguns servidores respondem a esta solicitação com seu número de versão do BIND. Com isto é identifica quais *hosts* são servidores e quais versões BIND estes estão usando. Com base nos resultados obtidos com a primeira fase, o invasor prepara um segundo voltado especificamente para estes servidores com o intuito de explorar suas vulnerabilidades de estouro de *buffer*. Esta segunda etapa poderia ocorrer em minutos, horas ou dias após a primeira. Isto depende das ferramentas utilizadas e da metodologia que o ataque abrange (NORTHUCUTT, 2002).

Como podemos observar existem invasões nas quais são realizadas gradativamente de forma sutil, podendo demorar dias, horas, ou minutos para se completarem. Estas podem tanto ser percebidas rapidamente quando, por exemplo, ocorre uma deformação de um *site* Web, mas também podem passar sem deixar suspeita e ficarem instaladas em seus equipamentos aguardando, por exemplo, que você acesse sua conta bancária para que possa roubar sua senha ou pode estar simplesmente varrendo seu HD em busca de informações.

Caso você não possua um sistema de detecção de intrusão, fraudes como o mencionado acima dificilmente poderão ser detectadas, e podem assim infestar toda a sua rede. Os IDS além de avisarem sobre ataques bem sucedidos também servem para identificar tentativas do mesmo que ainda não tenham sido concluídas, agilizando assim a ação sobre esta e aumentando a privacidade do ambiente.

[4] O *scam* ou golpe é qualquer ou ação enganosa e/ou fraudulenta adquiridas principalmente pela internet por meio de sites e e-mails

2.3 SISTEMAS DE DETECÇÃO DE INTRUSÃO

Os sistemas de detecção de intrusão são sistemas que nos auxiliam na monitoração do ambiente que estamos inseridos, estes procuram os ataques ou pacotes invasores que estão tentando invadir ou que já conseguiram passar pela segurança do *Firewall* ou qualquer outro software de segurança inserido. Os IDS não têm a função de somente monitorar as ameaças externas, eles podem também se utilizadas para o monitoramento de ameaças internas como, por exemplo, de *host* tentando violar algum privilégio atribuído a ele. Tais sistemas buscam informações de diversas fontes e de diversos pontos da rede onde são inseridos, analisam estas no intuito de encontrar alguma anomalia ou sinal de tentativa de intrusão (NORTHCUTT, 2002).

São sistemas projetados para a monitoração de ambientes computacionais, examinam o tráfego com o intuito de identificar ameaças como *scans*, sondas e ataques. Seu objetivo principal é auxiliar-nos identificando os ataques que podem estar sendo direcionados ao seu ambiente ou às ameaças que o seu próprio *host* possa estar passando para o ambiente que esteja inserido. Para que por meio destas informações, ou avisos, possamos iniciar as ações necessárias para eliminar as ameaças antes que elas atinjam seus objetivos (NORTHCUTT, 2002).

Um IDS é uma união de capacidades para descobrir e responder às ameaças, e suas principais funções são (PROCTOR, 2001, tradução nossa):

a) analisar logs anteriores para verificação de ataques conhecidos;
b) analisar o perímetro da rede para verificar se existe alguma ameaça;
c) administrar a configuração da rede;
d) verificar a integridade dos arquivos.

Um sistema pode estar provido de diversos IDS nos quais são posicionados em

diversos pontos estratégicos. A localização da fonte de análise é o que os divide em dois grupos: os baseados em rede (NIDS) e os baseados em *host* (HIDS). Os sensores NIDS geralmente são instalados em sub-redes nas quais são diretamente ligas ao Firewall ou em pontos críticos. Já os sensores HIDS residem em um host individual e faz o monitoramento do mesmo (NORTHCUTT, 2002).

Como podemos verificar os IDS são sistemas de monitoramento que podem ser implantado tanto em uma rede quanto em um *host* específico. Os implantados em rede visão proteger todo o perímetro da mesma, caso um pacote invasor venha a burlar a segurança implantada na sua "entrada", já os IDS que residem em *host* visão à monitoração dos mesmos, caso a verificação da rede venha a falhar. Em outras palavras, os IDS têm como foco garantir a segurança da rede caso a segurança inicial seja quebrada e para isto ele utiliza artifícios de alertas para que o invasor seja eliminado antes de causar qualquer dano.

Com o intuito de melhorar o entendimento do funcionamento do sistema de detecção de intrusão, faremos uma analogia entre o mesmo e o sistema de vigilância de um condomínio. Inicialmente podemos dizer que o perímetro da rede equivale a todo o espaço interno do condomínio, os *hosts* se referem às residências, as ruas equivalem à rede, a entrada do condomínio ou a portaria mostra-se como o servidor, ou a ligação com a Internet. O condomínio deve oferecer aos seus moradores segurança e para que esta seja alcançada, é necessário tomar algumas ações como, por exemplo, a constrição de um muro que se pode ser atribuído ao *Firewall*. A construção de um muro já aumentou significativamente a segurança do condomínio impedindo assim o acesso de pessoas não autorizadas. Porém, a construção deste não vem a atender completamente a necessidade, pois caso algum intruso venha a conseguir atravessá-lo, toda a segurança será quebrada. Para atender esta defasagem é necessária a instalação de câmeras de segurança que equivalem aos IDS. Estas são instaladas em lugares estratégicos, algumas focam o lado externo do condomínio, para que se possa

identificar a aproximação de pessoa e prevenir tentativas de invasão. Outras são posicionadas no interior do condomínio conseguindo assim detectar a presença de algum intruso que tenha conseguido passar pelo muro, e por fim cada residência pode possuir suas próprias câmeras para a prevenção da falha de toda a segurança antes mencionada (PROCTOR, 2001, tradução nossa).

Como podemos observar, toda a segurança atribuída à rede tem uma enorme similaridade com a situação citada acima, a analogia nos leva a entender melhor a importância dos sistemas de detecção de intrusão, além de demonstrar de uma forma simples a diferença entre os NDIS que são representados pelas câmeras do condomínio que monitoram o exterior e o interior do mesmo, e os HIDS que representam as câmeras que cada residência possui para seu controle interno.

2.4 O QUE É UMA INTRUSÃO?

Para podermos ter um melhor entendimento sobre a detecção de intrusão, faz-se necessário sabermos primeiramente o que é uma intrusão. Segundo Ximenes (1999, p. 359) “Ação de Intruso (Indivíduo) que se induz em lugar, função, posto, entre outros, sem ter este direito”.

Na nossa área é simplesmente uma atividade não autorizada em um ambiente de rede, esta não autorização pode vir de um usuário interno, tentando burlar o privilégio que lhe foi concedido em busca de alguma informação restrita do seu ambiente, ou de usuários externos que podem ser chamados de hackers, nos quais tentam acessar recursos da rede remotamente (CRONKHITE; MCCULLOUGH, 2001).

Seguindo a afirmação acima podemos dizer que uma intrusão também pode ser um vírus proliferado pela rede por meio de um e-mail infectado ou de qualquer outro recurso

infectado.

Em nossos propósitos, é simplesmente uma atividade não autorizada no sistema ou na rede por um de seus computadores. Isto pode incluir um usuário legítimo do sistema, tentando escalonar seus privilégios e ganhar maior acesso do que foi permitido a um usuário remoto e não autenticado, comprometendo um serviço em execução, a fim de criar uma conta no sistema. Uma intrusão também pode ser um vírus proliferando pelo seu sistema de e-mail e outros cenários similares. As intrusões podem vir de qualquer invasor deliberadamente malicioso, ou de um usuário final qualquer. Por exemplo, ao se abrir anexos de e-mails enviados, apesar de repetidas advertências para não fazerem. As intrusões também podem vir de alguém totalmente desconhecido, de um ex-empregado, ou do próprio pessoal de confiança (BAKER; CASWELL; POOR, 2004).

2.5 FUNCIONAMENTOS DO IDS

A detecção de intrusão pode ser feita por dois métodos, manual ou automática. Onde o primeiro é ideal para este tipo de sistema, podendo detectar explorações sutis em tempo real que podem ocorrer tão rapidamente que o segundo método não seria capaz de identificar. Com o intuito de realizar destas funcionalidades o sistema utiliza diversos métodos, como a detecção de anomalia e análise de padrão (CRONKHITE; MCCULLOUGH, 2001).

2.5.1 Modelo CIDF

O *Common Intrusion Detection Framework* (CIDIF) é um modelo no qual define

de uma forma mais simples ou genérica os componentes básicos de um sistema de detecção de intrusão. Neste modelo o sistema é dividido em quatro partes básicas, nas quais são ilustradas na Figura 1 pelas caixas A, C, D e E, onde estas representam respectivamente, análise, contra-medidas, armazenamento de dados e geração de eventos. O início do processo encontra-se na caixa E que é responsável pela coleta de informações da fonte de eventos que são gerados pelo sistema. Em seguida, na caixa A ocorrem as análises destes onde com seus respectivas resultados são passados para o modulo de armazenamento correspondente à caixa D. Por fim, com os devidos eventos analisados e salvos, são tomadas as medidas ou ações geradas por meio do módulo de contra-medidas que encontra-se na caixa C que é o ponto final do modelo. Também é possível observar o relacionamento destes módulos ou caixas, representadas pelas setas pontilhadas (PORRAS, 2008).

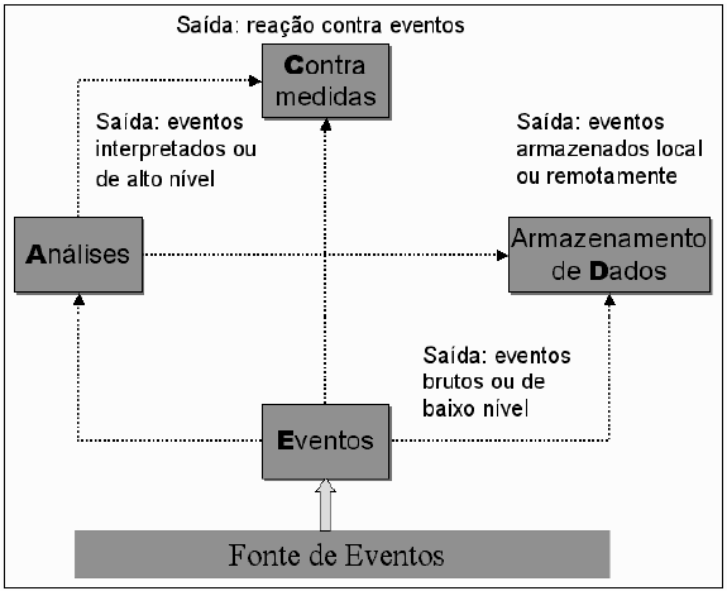

Figura 1. Modelo CIDIF
Fonte: PORRAS, P. (2008)

Este modelo permite um fácil entendimento e fixação dos processos que envolvem as ações dos sistemas de detecção de intrusão, mostrando de forma superficial o funcionamento geral do sistema a partir da coleta de dados até as respostas geradas.

2.5.2 IDS – Rede

Um IDS baseado em rede captura os dados da mesma, recolhendo pacotes e associando-os com outros já coletados para identificar se este novo pacote seja de tráfego normal ou um pacote com conteúdo perigoso. Para conseguir fazer esta análise de forma eficiente o mesmo deve coletar o maior número de pacotes possíveis trabalhando assim com um grande montante de dados e diminuindo a chance de erro (LISTON; SKOUDIS, 2002, tradução nossa).

O NIDS são sensores implantados para o monitoramento de uma rede interna recolhendo pacotes e comparando com outros já analisados podendo assim verificar se este já tenha ocasionado alguma irregularidade em alguma outra ocasião.

Estes IDS que são espalhados também podem estar vulneráveis aos ataques, caso estes consigam encontrá-los, podem danificar os mesmos deixando o ambiente desprotegido e vulnerável a ataques posteriores (NORTHCUTT, 2002).

Para a resolução deste problema utilizam-se algumas técnicas nas quais dificultam a identificação do IDS.

Um modelo ou técnica utilizada na situação acima é a criação de uma rede de gerenciamento separada para que se possa usar com mais eficiência a comunicação entre os sensores IDS, uma caixa de coletada de dados centralizada e consoles de analises. Para o funcionamento deste modelo cada sensor possui pelo menos duas Placas de interface de Rede (NIC). Uma ou mais NICs são usadas exclusivamente para a monitoração, sendo esta sua única função, ou seja, não transmite dados. Uma última NIC teria a função exclusiva para a transmissão de dados, comunicando-se com o sistema de gerenciamento que se encontra separado, onde é usada somente para a transferência de dados do IDS e atualizações de

configuração. Além disto, alguns NICs de monitoração são *sniffers* [5] puros e não utilizam um endereço IP. Se um sensor de IDS utilizar um endereço IP e algum ataque souber este IP, ele poderia lançar um ataque sobre este sensor fazendo com que este não consiga mais identificar um determinado tipo de ataque (NORTHCUTT, 2002).

Este modelo possui também outras vantagens, como o isolamento do tráfego de gerenciamento fazendo com que qualquer outra pessoa que estiver monitorando a rede não veja as comunicações entre os sensores da mesma. Isto também impede que os sensores monitorem seus próprios tráfegos e por fim também poderia ser uma boa alternativa para lidar com problemas relacionados com a transferência de dados do sensor pelo *firewall* ou por redes públicas não criptografadas (NORTHCUTT, 2002).

Na Figura 2 é apresentada uma rede usando três NIDS. Os mesmos foram implantados em lugares estratégicos e podem monitorar o tráfego de todos os equipamentos pertencentes à rede.

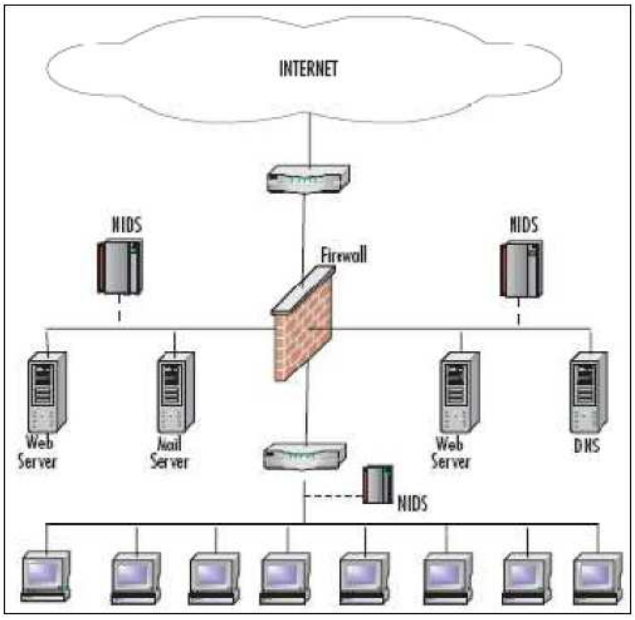

Figura 2. Rede com NIDS
Fonte: CASWELL, B. (2003)

[5] Os *sniffers* são programas que capturam os pacotes recebidos em seu estado bruto e os transforma em texto para serem lidos (HORA, 2008).

2.5.3 IDS – Host

IDS baseado em *host*, ou HIDSs, diferentemente dos NIDS, são voltados para a segurança do mesmo, eles são implantados principalmente para detectar e examinar atividades maliciosas nos seus diversos estágios. Eles podem também ser configurados para interagir suas descobertas a fim de oferecer uma visão unificada de vários sistemas por toda a rede (SCHULTER, 2006).

Uma função do HIDS é gerar um alerta quando o mesmo verifica que houve um aumento de privilégio do computador monitorado. Este tipo de comportamento, por sua vez, é difícil de ser verificado por um sensor de rede, pois estes não podem ver ou interpretar ações que ocorrem em um *host* (CASWELL, 2003).

Os HIDS possuem algumas vantagens em relação aos sensores de rede, como podemos verificar abaixo (NORTHCUTT, 2002):

a) estes sensores podem controlar as atividades de tráfegos específicas de um usuário no sistema, pois este possui acesso a informações especificas do mesmo como a listagem de processos, arquivos de *logs* e outros. Como os NIDS não possuem estas informações eles não conseguem determinar se o fluxo de tráfego de um determinado *host* está infringindo seu próprio privilégio;
b) HIDS podem monitorar as trocas de dados dos fluxos de rede criptografados analisando a extremidade da conexão, ou seja, o pacote é examinado em sua forma de texto limpo, antes de ser criptografado na saída ou antes de criptografar os que entram. Como os NIDS não conseguem examinar estes pacotes e como é crescente a utilização da criptografia nas empresas, o aumento na utilização dos IDS baseados em

Host torna-se necessário;

c) sensores baseados em *Host* podem detectar ataques que utilizam técnicas de evasão do IDS de rede, ou seja, caso um pacote que utilize alguma técnica de evasão consiga iludir ou enganar o NIDS, provavelmente o processo gerado por este pacote será detectado pelo HIDS.

Como podemos ver o IDS baseado em *Host* também desempenha uma função importante para a segurança, caso um pacote invasor consiga passar pelos IDS de rede, ele pode ser detectado pelo HIDS.

Em vista de um bom funcionamento e segurança é viável a utilização dos dois tipos de IDS e com estes trabalhando em conjunto, diminui significativamente as chances de um ataque ser bem sucedido.

Um ponto de vista que devemos ter é que se um IDS de rede detecta um ataque a um determinado host, como saber se este ataque foi bem sucedido ou não? É nesta hora que entra os HIDS, pois eles que podem auxiliar o NIDS informando os efeitos do ataque sobre o sistema infectado (NORTHCUTT, 2002).

Uma boa solução para as empresas seria a implantação de HIDS em todos os computadores da empresa, porém esta alternativa não torna-se muito viável levando em consideração a quantidade de recursos necessários para a implantação e manutenção deste sensores. O ideal seria analisar quais computadores são mais críticos e quais despertam mais interesse ao invasor, como por exemplo, um servidor que possui informações importantes e sigilosas sobre uma determinada empresa e que seja acessível pela Web, este sim seria uma boa opção para a implantação de um sistema IDS baseado em host, tornando-se assim mais seguro e confiável (CASWELL, 2003).

Um exemplo da situação pode ser encontrado na Figura 3 onde podemos observar a implantação de HIDS somente em alguns computadores do ambiente computacional.

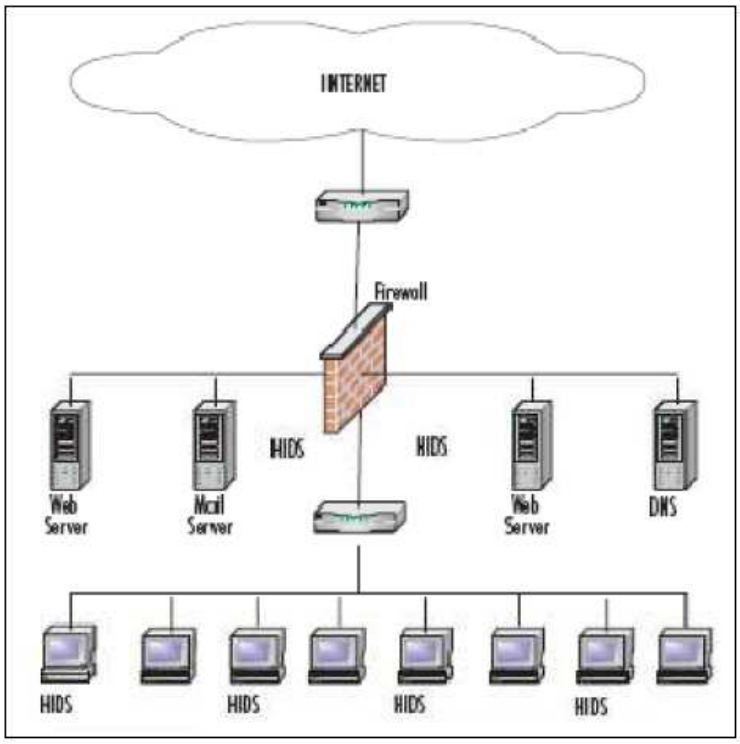

Figura 3. Rede com HIDS
Fonte: CASWELL, B. (2003)

A implantação dos IDS baseados em *host* encontra alguns desafios como garantir a eficiência e a finalidade do componente após o sistema estar comprometido e configurar e monitorar os componentes de defesa do *host*. Em vista disto, para minimizar ainda mais a possibilidade de um ataque ser bem sucedido é necessária a instalação de componentes que eliminem o escopo da influência do ataque no host comprometido. É viável também a implantação de softwares que restrinjam os recursos do sistema, indicando quais destes podem ser acessados por aplicações instaladas no local, um exemplo deste software é o Tiny Trojan Trap encontrado em http://www.tinysoftware.com/ (NORTHCUTT, 2002).

Outro desafio na implantação é a de controlar de forma centralizada todos os HIDS implantados. Quanto mais componentes são vinculados à política de segurança de nossa rede, mais difícil será de gerenciar a mesma. Para suprir estas deficiências alguns fabricantes de componente de defesa como, por exemplo, os de IDS baseados em *host*,

oferecem produtos para fazer o gerenciamento em questão. Esses produtos são estruturados para agir nas seguintes camadas principais (CASWELL, 2003):

a) agentes de imposição, como sensor de IDS baseado em *host*;
b) servidor de gerenciamento, que coletam eventos dos agentes de imposição;
c) banco de dados Back-end, usado para arquivar os eventos gerados;
d) uma aplicação cliente, usada para os administradores se comunicarem com o servidor de gerenciamento.

2.5.4 IDS - Distribuído

DIDS representam a união de HIDS, NIDS e outros componentes de defesa que agem de forma uniforme, porém, distribuídos na rede gerando *logs*, para a identificação de possíveis ataques. Estes serviços por meio dos *logs* gerados pelos *sites* fazem comparações para a identificação de novos possíveis ataques. Dois dos sistemas de IDS distribuídos são Attack Registry & Intelligence Service (ARIS), localizado em http://aris.securityfocus.com, e DShield, localizado em http://www.dshield.org (NORTHCUTT, 2002).

Na Figura 4 podemos encontrar uma demonstração deste tipo de estrutura onde os IDS estão distribuídos pelo ambiente computacional, porém trabalhando de forma integra para que a detecção dos ataques seja realizada de uma maneira mais eficaz.

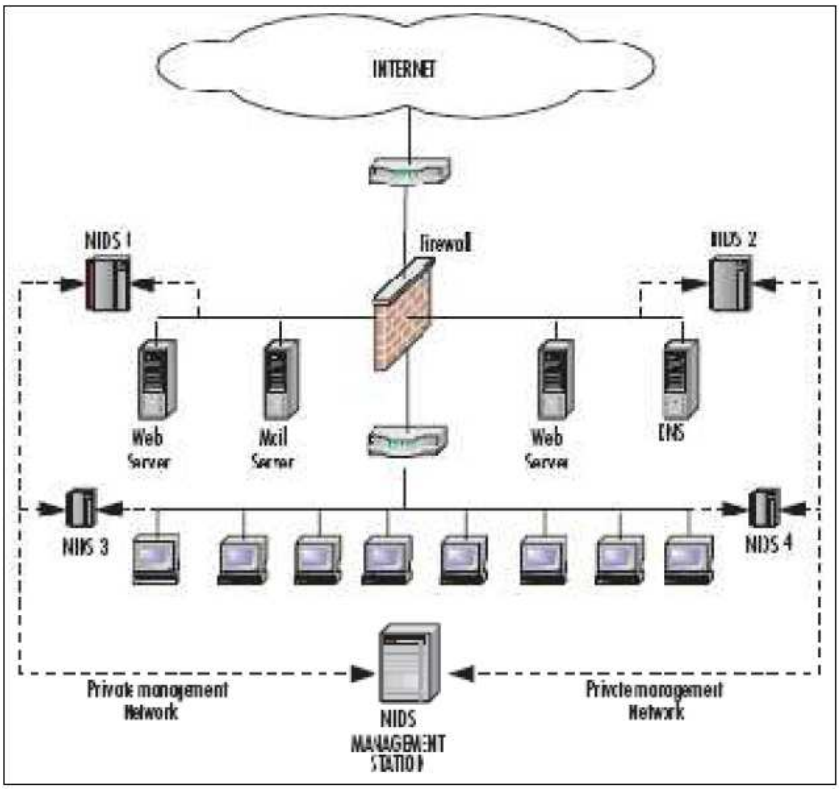

Figura 4. Rede com DIDS
Fonte: CASWELL, B. (2003)

Os DIDS possuem alguns benefícios como os recebimentos de *logs* de diversos ambientes, possuindo desta forma uma enorme riqueza de dados e podendo fazer por meio destes uma melhor análise da intrusão. Os IDS distribuídos conseguem localizar padrões entre os ataques devido à riquezas de dados mencionado anteriormente, coisa que sites individuais não conseguem fazer (CASWELL, 2003).

Como podemos verificar eles são componentes que utilizam tanto IDS de rede quanto baseados em host e por meio dos logs gerados por cada site deste componente consegue-se ter uma maior precisão na análise das invasões ou na identificação de novos ataques.

2.5.5 Informações de uma Aplicação Específica

Os três tipos de IDSs são capazes de monitorar pelo menos alguma informação específica de uma aplicação, isto pode ser feito de servidores da Web à aplicações Host.

Para realizar a análise de detecção de intrusão os IDS utilizam dos recursos das assinaturas. Estas podem ser de combinação de texto simples onde somente combinam os pedidos provenientes dos pacotes para a identificação de um ataque. Como estes pedidos podem aparecer de diversas formas e com valores ligeiramente diferentes, seria necessária uma assinatura separada para cada um dos pedidos mencionados acima. Um exemplo disto seria uma assinatura para o verme Nimda. A assinatura proveniente para este verme seria a GET/..%c0%af../winnt/system32/cmd.exe?/c+dir onde %cO%af é o equivalente a uma barra, de modo que o comando, na realidade, está tentando atravessar o diretório raiz para explorar suas vulnerabilidades. Uma assinatura de combinação de texto simples procuraria /scripts/..%c0%af../ para identificar a tentativa de ataque (NORTHCUTT, 2002).

Estas assinaturas específicas não são muitos indicadas, pois as mesmas tornam-se vulneráveis com o passar dos tempos devido a possibilidade dos vírus alterarem suas assinaturas no intuito de passarem despercebidos pelos sensores IDS.

Uma assinatura de vírus é uma seqüência de código exclusiva a ele. Os autores de vírus começam a criptografar ou embaralhar as mesmas toda vez que é criado um novo, de modo que a assinatura deste mude e com isto passe despercebida pelos sensores de detecção de intrusão (CRONKHITE; MCCULLOUGH, 2001).

Na Figura 5 encontra-se um exemplo de uma assinatura na procura de acesso a um PHP vulnerável.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-PHP
Advanced Poll admin_tpl_misc_new.php access"; flow:to_server,established;
uricontent:"/admin_tpl_misc_new.php"; nocase; reference:bugtraq,8890;
classtype:web-application-activity; sid:2299; rev:2;)
```

Figura 5. Assinatura de acesso PHP

Para minimizar o problema da mutação da assinatura de vírus, como foi visto acima, usa-se outro tipo de IDS, no qual tenha um maior poder de identificação de tipo de pedidos que possam ocasionar um ataque. Ao invés de realizar a simples combinação de texto, eles realmente decodificam o pedido e depois analisam a validade da URL decodificada. Com isto o sensor IDS emite um alerta caso a tentativa de invasão tenha sido confirmada. Seguindo o mesmo exemplo do verme Nimda que foi atribuído à assinatura simples, o comando %c0%af seria substituído pela barra fazendo desta forma a decodificação da assinatura do vírus e possibilitando desta forma a análise da URL (NORTHCUTT, 2002).

Uma assinatura específica poderia ser extremamente precisa na identificação de um ataque específico, porém, se o atacante modificar o ataque, essa assinatura não poderá identificá-lo. Por outro lado, uma assinatura geral pode ser muito mais rápida e poderia encontrar com mais facilidade novos ataques e variantes dos existentes, porém sua desvantagem é que pode classificar muitas atividades benignas como um ataque (NORTHCUTT, 2002).

Como podemos ver cada tipo de assinatura possui suas vantagens e desvantagens, cabendo ao administrador da rede escolher que tipo de assinatura é mais conveniente para o ambiente que administra.

2.5.6 Informações Específicas do *Host*

HIDS são responsáveis pela monitoração dos eventos que acontecem nos *Hosts*,

estes, por sua vez, têm acesso a informações específicas dos *hosts* que podem ser usadas para a detecção de uma possível intrusão nos mesmos, estas informações são provenientes de diversas fontes como sistema de arquivos, conexões de rede e arquivos de *log*. O principal motivo para o acesso a estas informações é porque a atividade maliciosa em um host pode se apresentar de varias maneiras, havendo assim a necessidade de um grande quantidade de informações para a detectá-las (NORTHCUTT, 2002).

2.5.7 Informação Específica da Sub Rede

Uma assinatura de rede é o padrão que você está procurando no tráfego, quando uma desta é encontrada, o ataque está presente no tráfego observado, um alerta é gerado para o administrador da rede ou então um evento é lançado no qual será armazenado de alguma maneira. Existem assinaturas bastante simples e também outras um pouco mais complexas e muitas destas são específicas do protocolo de aplicação (NORTHCUTT, 2002).

Alguns sistemas IDS focalizam longas seqüências de código para a análise do pacote, enquanto outros sistemas de detecção de intrusão efetuam uma análise de protocolo completa, examinando e validando valores de cabeçalhos e *payload*. Mesmo que a análise completa exija mais recursos, ela costuma a desenvolver uma solução de assinatura muito mais fortalecida (CASWELL, 2003).

2.6 COMO O IDS OBSERVA SUA REDE

A seguir serão informados os métodos mais utilizados pelos IDS para monitoração de pacotes na rede, assim como suas características, pontos positivos e negativos.

2.6.1 *Sniffing* de Pacotes

Os *Sniffings* são dispositivos de hardware implantados na rede que capturam informações sobre os pacotes que nela trafegam. Porém deve-se levar em consideração que os mesmos também podem apresentar uma ameaça a rede, pois se não forem criptografados e se senhas e logins não forem implantados, informações de controle e gerencia, terão seu conteúdo revelados completamente para a utilização do invasor (HORA, 2008).

2.6.2 *Log* Parsing

Outra fonte de dados para a monitoração de ataques ao *host* é a de arquivos de *logs*. Os monitores destes arquivos monitoram os conteúdos e alertam os administradores quando eventos suspeitos são detectados. Uma das vantagens deste sistema é que podem monitorar tais arquivos gerados por vários componentes de segurança do *host*. Além disto, caso tais componentes estejam agrupando seus registros em um local especifico, o monitor pode atuar neste verificando os mesmos que também podem ser provenientes de dispositivos do perímetro de defesa (NORTHCUTT, 2002).

2.6.3 Monitoramento de Chamadas do Sistema

Como os HIDS possuem acesso a informações especifícas de um host como listagem de processos, arquivos de log, privilégios dos usuários no host e entre outros, o mesmo é capaz de analisar as transações feitas pelo usuário e verificar se estas não estão infligindo seus privilégios. Com estas informações o HIDS é capas de monitorar a chamada

do sistema, pois se caso um comando no fluxo de tráfego venha a fazer uma solicitação na qual esteja fora do alcance do privilégio do usuário, provavelmente este comando esta tentando iniciar algum tipo de invasão (COMPAGNO, 2005).

2.6.4 Observação do Sistema de Arquivo (Filesystem)

As ferramentas de verificação de integridade do sistema de arquivos de um host operam apanhando os arquivos do sistema em um estado confiável, quando todos são considerados válidos. Por meio destes arquivos iniciais realizam-se analises comparando-os com os arquivos do sistema em busca de desvios que merecem ser informados. Para realizar esta verificação deve-se indicar quais destes podem ser ignorados pela análise e quais devem conter sua integridade. Para aumentar a eficiência deste sistema torna-se necessário a proteção do banco de dados de assinaturas que podem acontecer de diversas formas (NORTHCUTT, 2002):

a) camuflagem do conteúdo do banco de dados usando um formato binário em vez de texto limpo;
b) colocar o banco de dados em uma mídia apenas de leitura como um *pen driver* ou um CD-ROM;
c) assinar digitalmente o banco onde o administrador informe as chaves criptográficas e as senhas necessárias para acessá-lo.

2.6.5 Periodicidade

A periodicidade na coleta dos dados e a geração de eventos do sistema podem ser

divididas em sistema de tempo real ou em processamento em lote. No sistema de tempo real os dados são coletados e os eventos são gerados em paralelo à atividade normal dos sistemas. Já no processamento em lote os registros de auditoria das aplicações e dos sistemas operacionais são periodicamente analisados a procura de suspeitas (MARTINS, 2008).

Devido ao volume de dados trafegados na maioria das redes torna inviável a utilização do processamento em lote. Este, por sua vez, é indicado para situações específicas onde o tempo utilizado para a detecção de intrusão não é aceitável e pode prejudicar substancialmente o funcionamento de outras aplicações, porém nestes casos os sistemas somente poderão detectar o ataque depois de ocorrido impossibilitando a reação a ataques enquanto os mesmos ocorrem (MARTINS, 2008).

2.7 COMO O IDS CAPTURA DADOS

Neste tópico serão mostrados os recursos necessários para que se possa realizar uma boa análise do tráfego da rede. Serão ilustradas também as técnicas usadas pelos IDS para a captura dos dados assim como os entraves que algumas destas técnicas podem trazer.

2.7.1 Tentativas de Intrusão

Antes que se possa responder ou agir sobre uma intrusão na rede, é preciso encontrá-la, não adianta ter uma boa ação para eliminá-la se a sua ferramenta de encontro de anomalias estiver defasada.

Um ataque pode ocorrer de forma bastante transparente podendo ser detectado em tempo real. Porém, outros ataques podem ocorrer de forma mais sutis, onde para sua detecção

há necessidade informações precisar sobre o sistema assim como uma auditoria cuidadosa dos logs da rede. Para a verificação destas anomalias o sistema utiliza de alguns recursos como verificação do tráfego, o uso da CPU, o uso de recursos ou comportamento do usuário e o compara com um comportamento normal (CRONKHITE; MCCULLOUGH, 2001).

Para que se possa detectar uma tentativa de intrusão de forma eficaz é necessária a maior quantidade possível de informações sobre o sistema protegido, pois são por meio destas que se definem padrões de comportamento de fluxo ou ações sobre o sistema em questão. Por meio destes padrões pode-se determinar se o sistema esta sofrendo algum tipo de anomalia (NORTHCUTT, 2002).

2.7.2 Tecnologias para implementação da estratégia do IDS

Para realizar a coleta de dados e analisá-los os sistemas de detecção de intrusão utilizam meios distintos, cada tipo possui suas particularidades para a realização desta análise.

Para a monitoração dos *host* em específicos, os HIDS utilizam três técnicas que consistem na análise do sistema de arquivo do host em busca de arquivos alterados, analise das conexões de rede em busca de informações especificas como a abertura ou fechamento de transações e análise dos logs para a verificação de eventos suspeitos (COMPAGNO, 2005).

Cada sistema de monitoração visto acima tem suas vantagens, cabendo ao administrador decidir que tipo de análise será incorporada aos seus hosts e quais deles necessitam de tal monitoração.

Na monitoração de rede alguns IDS utilizam assinaturas, estas por sua vez podem ser escritas pelos próprios usuários, podem estar pré-definidas pelos fabricantes não havendo possibilidade de inserção ou alteração, ou ainda podem estar pré-definidas e permitirem que os usuários as alterem. Outra forma de monitoração de rede que alguns IDS utilizam é a de

análise de tráfego, onde são coletadas estatísticas sobre as conexões e por meio destas pode-se constatar violações de política de segurança, como o uso de serviço não autorizado e tráfego estranho em geral (NORTHCUTT, 2002).

Outros produtos IDS oferecem ainda soluções em camadas, nestas soluções são implantados sensores em diversos pontos estratégicos. Cada sensor gera seus dados e os transmite para um servidor centralizado, conseguindo unir, desta forma, uma grande quantidade de informações, podendo utilizá-las para a realização de comparações e montagem de padrões de tráfego para que se possa realizar a análise dos pacotes de uma forma mais precisa (CASWELL, 2003).

2.7.3 Falsos positivos ou falsos negativos

Falsos positivos e falsos negativos são dois problemas resultantes da má análise de assinaturas feitas pelos IDS, porém, não existe sistema de detecção de intrusão que não gere estes problemas.

Os falsos positivos ocorrem quando um sensor intitula uma instrução benigna de um determinado pacote como sendo uma tentativa de ataque. Já os falos negativos são ao contrário, ou seja, quando um pacote invasor passa sem ser visto não gerando desta maneira o alerta ao administrador. Todo IDS gera falsos positivos, estes por sua vez são fáceis de perceber, pois gera o alerta ao administrador, já os falsos negativos não são percebidos, pois não geram o tal alerta, e infelizmente só são constatados depois de terem obtido sucesso em sua invasão (NORTHCUTT, 2002).

Um ponto relevante e que influência diretamente na geração de falsos positivos e falsos negativos é a forma como as assinaturas estão implementadas, umas possuem características mais abrangentes outras possuem mais restritas, que são voltadas mais

especificamente para um tipo de invasão e agindo sobre este de forma bastante eficiente. Porém pode deixar de atender os outros tipos de ameaças que o sistema possa sofrer, podendo desta forma, gerar uma grande quantidade de falsos negativos. Por outro lado existem assinaturas que possuem características mais globais onde conseguem atender um maior número de intrusões, porém, devido a esta abrangência, começam a intitular tráfegos benignos como ameaças, gerando desta forma os chamados falsos positivos (CASWELL, 2003).

Para entender melhor a situação descrita acima vejamos um exemplo simples. Vamos supôr que você escreva uma assinatura que procure cmd.exe em qualquer lugar da URL, esta assinatura é geral e combina com diversos ataques que tentam explorar o diretório raiz dos servidores, porém esta assinatura combina com nascmd.exe que não esta relacionado com estas explorações, ocasionando desta maneira falsos positivos. Para minimizar estes erros você decide reescrever sua assinatura, essa nova assinatura é uma URL que contem /winnt/system32/cmd.exe. Como esta é mais específica ela conseqüentemente diminui os falsos positivos, porém aumenta a quantidade de falsos negativos, pois deixa de localizar alguns pacotes invasores como, por exemplo, os que possuam URL que inclui /winnt/system32/../system32/cmd.exe (NORTHCUTT, 2002).

Este impasse nas assinaturas leva os administradores a terem que optar que tipos de falhas são mais aceitáveis no seu ambiente, ou seja, se desejarem que suas redes tenham uma análise diversificada optando para isto de assinaturas mais gerais, conseqüentemente terão vários falsos positivos, e se optarem por capturar somente um ataque em especifico terão vários falsos negativos .

Para tentar suprir esta defasagem dos IDS são desenvolvidas assinaturas mais complexas que eliminam consideravelmente os erros supracitados.

Assinaturas mais complexas normalmente são melhores que as de simples combinação de texto, devido ao fato de que alguns ataques são configurados para evitar a

detecção por assinaturas simples, ou seja, evasão de IDS. Nas mais complexas a URL inteira é decodificada e posteriormente avaliada, minimizando desta forma tanto os falsos positivos como também os negativo (NORTHCUTT, 2002).

2.8 O QUE O IDS FAZ QUANDO ENCONTRA UMA TENTATIVA DE ATAQUE

Serão mostrados a seguir os tipos de respostas que os sistemas de detecção de intrusão podem retornar quando encontram uma tentativa de intrusão ou um pacote suspeito e por fim será informado as possíveis formas de implantação deste sistemas em uma rede de computadores.

2.8.1 Resposta Passiva (*Passive Response*)

Neste tipo de resposta apenas notifica-se a autoridade, ou seja, o administrador, por meio de email, mensagens de aviso ou qualquer outro tipo de alerta, informando-o sobre a existência de uma tentativa de ataque ou de um invasor já existente, e passando para o mesmo a responsabilidade de tomar as medidas adequadas (VALLE, 2008).

Como constatamos é um tipo de resposta onde o sistema de IDS interage somente com o administrador da rede, sem agir diretamente sobre o pacote invasor, ou seja, nesta ocasião o sistema não move nenhum tipo de ação que venha impedir o pacote invasor de trafegar sobre a rede.

2.8.2 Resposta Ativa (*Active Response*)

Outro tipo de resposta que um pode ser gerado após a identificação de um agente invasor é a resposta ativa. Este tipo, diferente da passiva, pode gerar após a confirmação do ataque uma ação sobre si mesmo, o invasor ou o sistema que esta sendo atacado (COMPAGNO, 2005).

Quando um sensor identifica um ataque, ele muda sua própria configuração, ou a de outros dispositivos da rede para responder diretamente ao tráfego do host que esta sendo atacado. Estas respostas podem ser envidas da seguinte forma (NORTHCUTT, 2002):

a) enviando pacotes *rests* ao invasor para terminar a conexão TCP suspeita;
b) enviar um conjunto de regras ou configurações para os dispositivos de firewall ou dispositivos de filtragem de pacotes com intuito de bloquear temporariamente ou permanentemente todas as tentativas de conexão do atacante;
c) monitorar todo o tráfego do atacante mais perto.

Embora este tipo de resposta venha a trazer benefícios, deve-se ter certo cuida em sua implementação, pois estas respostas podem vir a cancelar recursos no sistema devido a erros na análise de tráfego.

Os endereços IP normalmente são falsos de modo que o atacante pode não estar localizado no mesmo, o sensor poderia também estar gerando um alerta falso, quando um pacote benigno é intitulado com um invasor. Desta forma o sistema de detecção de intrusão poderia estar negando um serviço a um pacote inofensivo (NORTHCUTT, 2002).

2.8.3 Posicionamento do IDS na rede

O posicionamento dos sensores na rede pode se feito de diferentes formas, não existindo um padrão ideal para o mesmo. Cada rede possui suas particularidades havendo a

necessidade de uma boa análise para verificar qual a melhor forma de posicionar os sensores na mesma.

Na maioria dos ambientes você terá que implantar vários sensores, cada um monitorando um segmento. Em uma empresa de pequeno porte com rede simples e tráfego restrito, um IDS poderia ser adequado, mas não muito indicado, pois este pode falhar. Em empresas de grande porte, com muitos segmentos e com conexão com a Internet, certamente vários sensores seriam necessários para uma monitoração adequada (NORTHCUTT, 2002).

Um bom ponto para a implantação do sensor seria perto de firewalls e dispositivos de filtragem de pacotes, caso possua-se um acesso a Internet o ideal seria posicionar um sensor na parte externa da rede, para que se possa verificar os possíveis ataques que a mesma pode receber, e outro na interna, logo após o firewall, este por sua vez irá indicar os pacotes que conseguiram passar por ele, além de indicar o mau funcionamento do mesmo no caso da passagem de muitos invasores (GOMES, 2008).

Como nem sempre podemos ter o número de recursos que desejamos e se de alguma forma for restringido o número de sensores a serem implantados, provavelmente em algum momento teremos que fazer escolha e sacrifícios, como por exemplo, colocar um sensor na rede externa ou interna. Para que esta escolha seja feita de forma adequada para sua rede faz-se necessário o conhecimento de algumas informações.

Geralmente recomenda-se a implantação do sensor na rede externa, pois o mesmo poderia detectar todos os ataques incluindo os que não passam pela filtragem. Esses sensores por estarem localizados na rede externa, principalmente os que estão ligados a Internet, têm mais chances de serem atacados, e eles também processaram muito mais tráfego do que um de rede interna. Outro fator relevante é o número de alertas que este sensor pode vir a gerar, caso possua-se uma equipe com pouco tempo para a análise de alertas, o ideal seria a implantação dos sensores internamente, pois os mesmos irão processar e gerar alertas somente do tráfego

que entrasse na rede (NORTHCUTT, 2002).

Os ataques que seus hosts enviam para hosts externos também levam a ser considerado no posicionamento do sensor assim como a quantidade de dados que trafegam em sua rede diariamente. Se o seu firewall possui uma política de negação default para tráfego que sai, um sensor interno é essencial para a identificação de ataques dos seus hosts para os externos, porém se você possui uma política de saída o posicionamento do mesmo é menos importante. Além disto, o volume de informações que passam pela rede também deve ser considerado, pois se você tiver um volume de dados extremamente grande, é indicado a implementação de vários sensores com diferentes configurações para dividir o tráfego, e se sua rede externa possuir tráfegos extremamente altos, à a possibilidade de incluir um sensor nesta com configurações especificas para identificar ataques mais perigosos (NORTHCUTT, 2002).

A quantidade de possibilidade para a implementação de sensores de detecção de intrusão no seu perímetro de rede é grande, necessitando então uma análise para verificar que tipo de ataques você quer prevenir, a quantidade de recursos que você pode utilizar e o volume de informações que trafegam em sua rede diariamente.

2.9 UM SISTEMA DE DETECÇÃO DE INVASÃO: SNORT

O Snort teve início em 1998, onde era chamado de APE. Este software foi desenvolvido por Marty Roesh onde apenas analisava pacotes para sistemas Linux, a princípio possuía somente interesses para o uso do próprio desenvolvedor, mas como o projeto saiu melhor do que o mesmo imaginava, ele começou a se empenhar cada vez mais e aplicar novos recursos ao sistema, alterando então o nome do aplicativo para Snort. No ano de 1999 foi implantado no mesmo o recurso de análise baseada em assinaturas, promovendo com isto o

programa para a área de detecção de intrusão (BORGES; COUTINHO, 2009).

Ele é um sistema múltiplataforma, múltipla-arquitetura e de código fonte aberto, o mesmo podendo executar de forma passiva ou ativa realizando análise nos pacotes que trafegam na rede onde o mesmo está inserido, observando-os da mesma forma que um *sniffer*, buscando padrões de conteúdos entre os mesmos (SILVA; VITALI, 2009).

A ferramenta Snort é de poder público devido sua particularidade *open-source.* Ela trabalha como se fosse um *sniffer*, porém tem como diferencial a capacidade de analisar os dados que trafegam no *payload* dos pacotes, ou seja, os dados do mesmo, e a criação de assinaturas contra ataques, podendo com isto detectar uma invasão (BORGES; COUTINHO, 2009).

A base deste sistema é a biblioteca *Liccap* que possui os componentes necessários para dar acesso aos recursos de baixo nível da rede, onde estes são utilizados pelos componentes do Snort. A arquitetura do mesmo é dividida em componentes onde estes trabalham em conjunto para detectarem invasões e gerarem respostas de acordo com as assinaturas ativadas. Esta arquitetura é dividida em e quatro partes: mecanismo de captura, pré-processadores, mecanismo de detecção e mecanismo de resposta (BORGES; COUTINHO, 2009).

Na Figura 6 podemos verificar os componentes de uma forma mais geral, apresentando a relação entre eles.

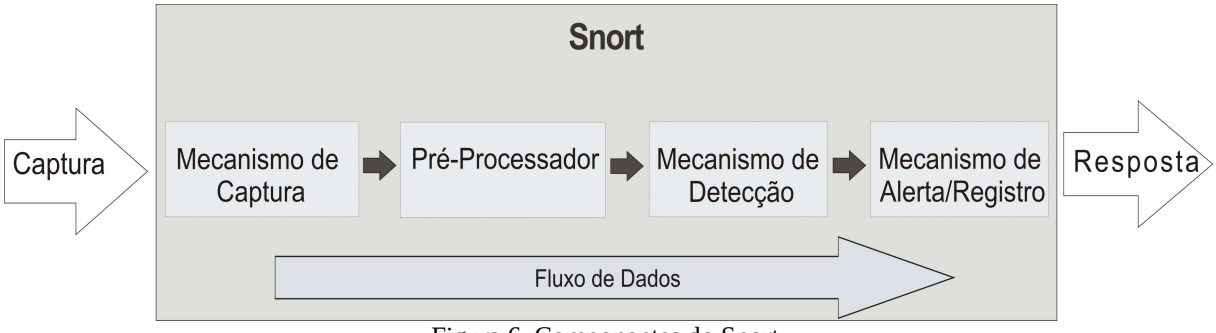

Figura 6. Componentes do Snort.
Fonte: Adaptado de SILVA, M.; SAMPAIO, M. (2009)

2.9.1 Mecanismo de Captura

Este mecanismo é responsável pela captura dos pacotes que trafegam na rede. Para que esta possa ser realizada é necessária a instalação de alguns recursos e a realização de algumas configurações.

A primeira alteração necessária é a configuração da placa de rede para o modo promiscuo, conseguindo com isto o privilégio de poder capturar todas as informações que passam por elas. Já a segunda é a instalação da biblioteca *libcap*/*winpcap* para capturar os pacotes que foram liberados por meio da primeira configuração (CASWELL, 2003).

A biblioteca *libcap* foi desenvolvida como parte de um programa *TCPDump*, para ser capaz de capturar pacotes das camadas dos modelos TCP/IP, ou seja, diretamente da placa de rede, com isto esta fica independente dos *drivers* dos sistemas operacionais. Ela foi desenvolvida especialmente para sistemas operacionais Unix, porém um grupo de pesquisadores da Itália desenvolveu uma versão para o sistema operacional Windows que recebeu o nome de *wincap* (CASWELL, 2003).

Após a configuração da rede e a captura dos pacotes, é necessário um mecanismo que venha a processar as informações coletadas. Esse mecanismo de processamento é chamado de pré-processamento.

2.9.2 Pré-processadores

Os pré-processadores são responsáveis por classificar os dados coletados pelo mecanismo anterior. Após a classificação, esta estrutura permite a geração de alerta, a eliminação do pacote ou a modificação do mesmo, antes de ser encaminhado para o próximo

mecanismo (SANTOS, 2005).

Este mecanismo utiliza *plug-ins* para determinar o comportamento dos pacotes e através destes realiza as ações necessárias. Possibilita também a desativação ou ativação de alguns destes *plug-ins,* de acordo com a necessidade e o perfil da rede (CASWELL, 2003).

Após a captura e o pré-processamento o mecanismo de detecção recebe os dados já arrumados, podendo fazer uma análise mais eficaz, ou seja, diminuindo a chance da geração de uma análise incorreta (CASWELL, 2003).

2.9.3 Mecanismo de Detecção

Este mecanismo é considerado a parte mais importante, pois é nele que as informações coletadas são analisadas, ou seja, é neste processo que o IDS compara as mesmas com o conjunto de assinaturas dos ataques já conhecidos. Uma vez que estas informações coincidam com alguma das assinaturas, o pacote que a possui será intitulado como tráfego suspeito (CASWELL, 2003).

Esta parte do processo de detecção de intrusão é que se deve ter uma atenção especial, pois é nela que as assinaturas serão comparadas com as informações coletadas e onde será gerada a resposta que identificará se um pacote é invasor ou não, fazendo com que este processo seja um dos maiores responsáveis pela eficiência das detecções da ferramenta (BORGES; COUTINHO, 2009).

Estas assinaturas são encontradas em um conjunto de regras armazenadas na forma de texto em arquivos no formato “.rules”, geralmente encontradas em um subdiretório do diretório de instalação do Snort (SANTOS, 2005).

Após a comparação dos dados do pacote com as assinaturas, é necessário um mecanismo que venha a alertar caso uma invasão tenha sido confirmada, para realizar esta

tarefa utiliza-se o mecanismo chamado de Alerta ou Registro.

2.9.4 Mecanismo de Alerta/Registro

Após os processos de coleta, modificação e análise, caso haja um invasão confirmada, há necessidade da geração de um alerta ao administrador e o armazenamento dos registros gerados.

O processo de Alerta/Registro tem este papel, onde recebem do mecanismo de detecção as informações resultantes da análise das informações do pacote e por meio destas verificam qual ação deverá ser gerada sobre o pacote analisado.

O mecanismo de alerta ou registro podem interagir com *firewall*, enviar alertar na tela ou via e-mail, gravar *logs* em arquivos texto, ou em banco de dados, entre outros. O Snort também suporta várias ferramentas adicionais, como *PHP*, Servidores *Web, Swatch* e outros (MONTORO, 2009).

Depois dos alertas gerados e os registros gravados em seus devidos lugares, o processo de detecção de intrusão chega ao fim, porém os *logs* gerados poderão ser utilizados pelo administrador da rede para análises de ameaças futuras.

2.9.5 Regras

Para que o Snort consiga identificar se um pacote que trafega na rede analisada é realmente uma tentativa de invasão, é necessária a criação de regras. Essas regras devem possuir parâmetros que possibilitem ao administrador filtrar os pacotes de forma dinâmica, ou sejam, podendo implementar regras que sejam utilizadas para todos os tipos de tráfegos ou para tráfegos específicos (SANTOS, 2005).

Fazendo um estudo mais detalhado, foi possível identificar que as mesmas possuem duas partes lógicas, como nos ilustra a Figura 7.

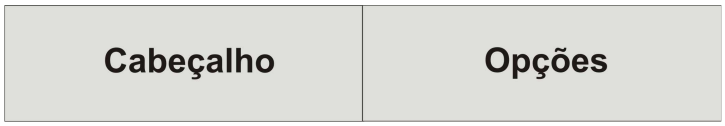

Figura 7. Estrutura de uma regra.
Fonte: SILVA, M.; SAMPAIO, M. (2009)

Na Figura 8 podemos observar mais detalhadamente o conteúdo da parte do cabeçalho da regra, é esta parte que possui os atributos necessários para identificação dos pacotes. Os atributos existentes nela são: O IP origem e destino, porta de origem e destino, pelo protocolo no qual está sendo estabelecida a conexão, a direção do tráfego de informações e a ação na qual se deve ser tomada no caso de encontrar um pacote que venha a atender todos os requisitos anteriores (SANTOS, 2009).

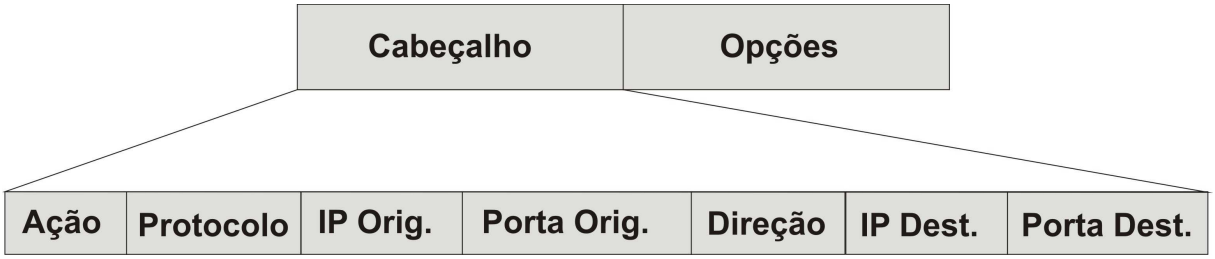

Figura 8. Estrutura do cabeçalho de uma regra.
Fonte: SILVA, M.; SAMPAIO, M. (2009)

Identificar o IP que originou o pacote ou a quem está sendo destinado o mesmo, não basta para encontrar uma real ameaça à rede protegida, há também a necessidade de uma análise minuciosa no conteúdo do mesmo, buscando por assinaturas, comandos ou expressões, que já sejam conhecidas e que possam vir danificar a rede. Também se necessitam de atributos que possam classificar e identificar o tipo de ataque detectado. Para realizar estas análises, as regras do Snort concentram na sua parte lógica, chamada opções, os atributos que possibilitam tais tarefas. A Figura 9 ilustra a separação desta segunda parte da regra dividida

em 4 categorias (SILVA, SAMPAIO, 2009).

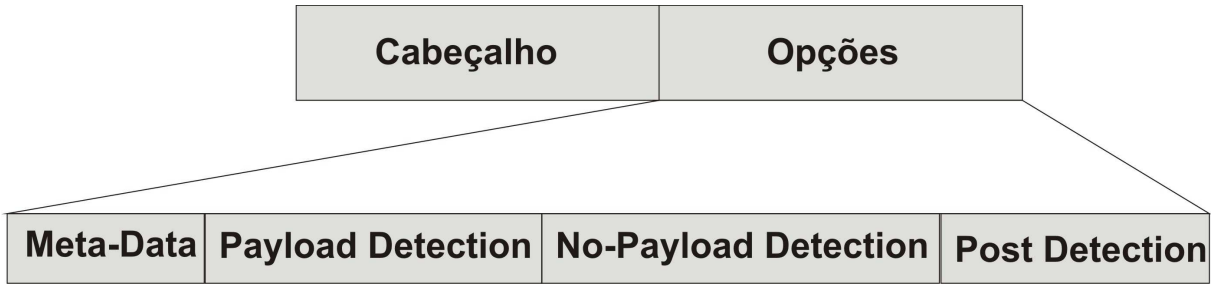

Figura 9. Estrutura das opções de uma regra.
Fonte: SILVA, M.; SAMPAIO, M. (2009)

Tendo o conhecimento da sintaxe das regras e dos atributos que a mesma possui, é possível desenvolver regras personalizadas nas quais atendam a necessidade da rede a ser protegida. Estas regras podem ser criadas com o passar do tempo, pois são carregadas toda vez que o sistema iniciar uma nova análise.

As variáveis do cabeçalho da regra, como já descrito anteriormente, são utilizadas para fazer uma seleção mais superficial, ou seja, seleciona os pacotes originados de um determinado endereço e/ou que estão destinados a outros endereços pré-determinados, selecionam os pacotes por protocole ou por direção de fluxo, abaixo se encontram as possíveis variáveis (SANTOS, 2009):

a) ação: é o primeiro atributo da regra, por meio deste é definida a ação a ser tomada. Existem 5 ações permitidas:
    - *pass*: indica ao Snort para ignorar o pacote, não executando nenhuma verificação sobre o mesmo,
    - *log*: usado para somente registrar o evento geradao,
    - *alert*: utilizado para enviar uma mensagem de alerta quando um pacote satisfizer todos os requisitos da regra,

- *active*: gera um alerta e em seguida chama outra regra, é utilizado quando se necessita fazer mais teste sobre o pacote capturado,
- *dynamic*: são regras implementadas para serem ativadas somente por outras regras do tipo Active;

b) protocolo: apenas verifica os pacotes que possuam o protocolo determinado na regra;

c) endereço IP origem e destino: por meio deste campo podem-se filtrar pacotes tanto por endereços IPs ou por endereços de redes, também é possível atribuir o sinal de exclamação ‘!’ antes do parâmetro para pegar todos os pacotes que não possuem o tal, e para capturar todos os pacotes utiliza-se a expressão ‘any’, também é possível;

d) porta de origem e destino: realiza a verificação dos pacotes que possuam as portas de origem e destino informadas na regra, este campo permite a inserção do número da porta, do sinal de exclamação funcionando da mesma maneira que funciona na regra dos endereços IP e também suporta a expressão “any” para capturar sem observar a porta de origem ou destino;

e) direção: determina a aplicação da regra à pacotes que possuem uma determinada direção. Os possíveis valores para esta regra são:

- o símbolo ‘->’: significa que o IP e a porta da direita são de origem e o IP e porta da esquerda é de destino,
- o símbolo ‘<-‘: é o contrário do símbolo acima, ou seja, o IP e porta da esquerda são a origem enquanto os da direita é o destino,
- o símbolo ‘<>’: não faz critério algum em relação ao sentido do tráfego, ou seja, aplica a regra a todos os pacotes que estão na rede.

Para um melhor entendimento da sintaxe do cabeçalho vejamos a seguir um exemplo do mesmo onde é informado o tipo de ação, protocolo, o IP de origem, a porta de origem, a direção do tráfego, o IP de destino e a porta de destino, respectivamente.

Ex. ALERT UDP ANY !8080 -> 207.172.16.155 80

No exemplo de regra acima ela irá capturar e gerar alertar para topos os pacotes que possuam o protocolo UDP que são originados de qualquer IP que a porta de origem seja diferente de 8080 e que serão destinados para o IP 207.172.16.155 através da porta 80.

As opções das regras também possuem um conjunto determinado de variáveis que podem ser utilizadas para formar as regras. Estas variáveis encontram-se dentro dos parênteses da regras e separadas por ponto e vírgula, geralmente a variável é dividida em duas partes, a palavra chave que a identifica e o valor da mesma, onde estas duas partes são separadas por dois pontos ‘: ’, além destas duas partes algumas variáveis podem possuir algum argumento adicional (SILVA, SAMPAIO, 2009).

Para que uma ação seja tomada é necessário que todas as variáveis definidas na regra sejam verdadeiras, ou seja, o pacote tem que satisfazer todos os requisitos pré-definidos na formação da regra na parte lógica da opção para que a parte do cabeçalho possa aplicar a ação programada.

No grupo Meta-Data da regra encontram-se as variáveis responsáveis pela definição de informações personalizadas para o ataque, podendo auxiliar na sua classificação, nesta opção as variáveis não tem nenhum papel ou efeito sobre a detecção propriamente dita. Abaixo temos algumas das variáreis suportadas por este grupo (SANTOS, 2009):

a) MSG: utilizada para se definir a mensagem que irá aparecer no log, ou no alerta gerado pelo IDS no momento da detecção. Normalmente esta opção é

utilizada da seguinte forma: msg: “Texto Escolhido”;

b) REFERENCE: é utilizado para fazer uma referencia, este atributo referencia outros sistemas existentes na web como o CVE (Common Vulnerabilities and Exposures) e Nessus, onde os mesmos possuem informações adicionais aos ataques já conhecidos;

c) SID: este campo foi desenvolvido no intuito de criar um identificador único para as regras do Snort. OS valores deste campo são números inteiros maiores ou iguais a zero, onde estes estão com compreendidos em três grupos:

   - 0 à 99 – Reservados para o uso futuro,
   - 100 à 1000000 – Reservados para regras criadas pelo Snort,
   - acima de 1000000 – Podem ser utilizadas nas regras criadas pelos usuários;

d) REV: esta variável tem a função de identificar a versão da regra, ou seja, cada vês que uma regra é alterada, ganha um novo numero REV, e permanecendo com o mesmo numero do SID, podendo assim ser diferenciada as diferentes versões;

e) PRIORITY: esta palavra chave é utilizada para definir o grau de prioridade que uma determinada regra possui, este campo aceita somente valores inteiros maiores que 0 onde o valor com maior prioridade de é o de numero 1, tendo esta prioridade diminuída com o aumento do valor numérico do campo;

f) CLASSTYPE: é utilizado para incluir classificações e prioridades as regras. Este campo é geralmente combinado com o campo priority, onde são agrupas para redefinir a prioridade da regra.

Já na parte do Payload Detection encontram-se as variáveis mais importantes para a análise e identificação de uma invasão, pois é através destas que observa-se os dados que estão inseridos nos pacotes suspeitos. A palavra-chave CONTENT é responsável por esta função, ou seja, um pacote será detectado caso possua nas suas informações o valor definido à variável Content. O valor atribuído a este campo pode ser através de *string* ASCII ou através de caracteres hexadecimais. Existem algumas variáveis que podem trazer alguns critérios a mais e que são utilizadas em conjunto co a variável “content”, são elas (SANTOS, 2009):

a) offset e depth – restringe a verificação á uma parte das informações contidas no pacote de dados;
b) nocase – indica se a verificação irá ou não observar letras maiúsculas e minúsculas.

O Non-Payload Detection possui os atributos específicos para cada protocolos IP, ICMP, TCP e UDP suportados pelas regras IDS. Com estes atributos incluídos pode-se fazer uma melhor verificação dos pacotes, podendo analisar alguns campos específicos contidos no cabeçalho dos pacotes como, por exemplo, *flags*, seqüência de pacotes, direção dos mesmos entre outros e conseqüentemente aumentado a eficiência na análise dos mesmos (SILVA, SAMPAIO, 2009).

Para uma melhor explicação e visualização destas variáveis, criou-se um quadro onde é possível identificar os atributos que cada protocolo pode possuir com suas respectivas funções ou contribuição para o conjunto de variáveis de análise de pacotes. Estas variáveis podem ser vistas no Quadro 1 que nos mostra cada protocolo com seus respectivos campos e a função na identificação do ataque como mencionado anteriormente.

Protocolo	Campo	Função
IP	ipopts	Utilizado para identificar o tipo de ataque
	fragbits	Verifica os bits MF (More Fragments), DF (Don't Fragment) e RB (Reserverd Bits) existentes no cabeçalho IP do pacote
	fragoffset	É comparado com o campo fragment offset do cabeçalho IP
	ttl	Verifica o campo ttl (time-to-live)
ICMP	icmp_ip	Permite comparar um "id" presente no cabeçalho do pacote com um valor decimal
	Icmp_seq	Permite comparar o numero de seqüência presente no cabeçalho do pacote com um valor decimal
	itype	Examina o valor do campo ICMP *type* presente no cabeçalho
	icode	Examina o valor do campo ICMP *code* presente no cabeçalho
TCP	seq	Verifica o número de seqüência de um pacote TCP
	ack	Verifica o numero de ack do protocolo que consiste no próximo número de seqüência que o transmissor do pacote TCP está à espera
	flow	Verifica pacotes fluem em uma determinada direção, pode-se utilizar as seguintes palavras chaves para determinar a direção: to_client, to_server, from_client e from_server
	flags	Verifica quais *flag* bits estão ativos dentro do cabeçalho TCP de cada pacote.
UDP	rpc	Detecta pedidos baseados em RPC e aceita três números como argumento: numero da aplicação, número do procedimento e número da versão

Quadro 1. Atributos dos protocolos das regras IDS

O Post Detection também possui seu conjunto de variáveis específicas, como podemos ver a seguir (SILVA, SAMPAIO, 2009):

a) *logto* – indica ao Snort para gravar os *logs* das regras que possuem este atributo em um arquivo especial;

b) *session* – extrai dados do utilizador de uma sessão TCP, este atributo permite apenas dois valores, "printable" e "all". O primeiro imprime os dados que o utilizador veria normalmente e o segundo substitui os caracteres não imprimíveis nos seus equivalentes hexadecimais;

c) *resp* – é utilizada para tentar fechar uma sessão quando um alerta é gerado;

d) *react* – realiza uma reação ao tráfego detectado, onde basicamente

bloqueia sites que os utilizadores queiram utilizar;

e) *tag* – permite marcar o tráfego de origem e/ou destino quando a regra é ativada, para permitir uma análise dos códigos de resposta e tráfego pós-ataque.

Para reforçar a sintaxe das regras do Snort encontram-se na Figura 10 alguns exemplos das mesmas.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"COMMUNITY
ICMP undefined code"; icode:>18; classtype:misc-activity; sid:100000197; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"COMMUNITY BOT
Agobot/PhatBot bot.about command"; flow: established;
flowbits:isset,community_is_proto_irc; content:"bot.about"; classtype: trojan-
activity; sid:100000242; rev:2;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"COMMUNITY BOT
Agobot/PhatBot bot.die command"; flow: established;
flowbits:isset,community_is_proto_irc; content:"bot.die"; classtype: trojan-activity;
sid:100000243; rev:2;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"COMMUNITY BOT
Agobot/PhatBot bot.dns command"; flow: established;
flowbits:isset,community_is_proto_irc; content:"bot.dns"; classtype: trojan-activity;
sid:100000244; rev:2;)
```

Figura 10. Exemplos de regras do Snort.

Com todo o conhecimento adquirido até o momento em relação à criação de regras, já é possível desenvolver regras específicas que venham a atender nossas necessidades. Vale lembrar que as regras são um das partes mais importantes e essenciais para o bom funcionamento de um sistema de detecção de intrusão, pois é por meio delas que os ataques são identificados e também são através das mesmas que se escolhe a ação a ser tomada. Como vimos, estas regras podem ser desenvolvidas pelo próprio administrador da rede podem ser encontradas no *site* do Snort http://www.snort.org.br/. Neste ultimo caso encontram-se regras desenvolvidas pelo próprio pessoal de desenvolvimento e também pelos usuários cadastrados.

Por fim, podemos dizer que quanto mais atributos e especificações forem

incorporados à regra menos vulnerável a avisos falsos ela ficará, porém poderá gerar um numero maior de falsos negativos. Para evitar estas situações, há necessidade da incorporação de algum método que venha a diminuir tais erros, um dos mesmos bastante relatados pela literatura é o de agentes inteligentes.

3 AGENTES INTELIGENTES

Os agentes inteligentes representam uma área da computação com grande relevância científica, e por conseqüência bastante pesquisada, pois a partir desta teoria surgem softwares com capacidade de raciocinar, tomar decisões e até mesmo de aprender. Devido a estas características os mesmos são destinados a desempenharem papéis, ações ou funções que são de difícil realização como tarefas repetitivas ou que possuam um grande número de parâmetros a serem verificados (RUSSELL; NORVIG, 2004).

Em vistas destes benefícios, cientistas de várias nacionalidades vêm realizando pesquisas, buscando desenvolver agentes com maior capacidade de raciocínio, melhor aprendizagem e com a habilidade de tomada de decisão mais desenvolvidas, aumentando o número de tarefas e ações que o mesmo possa realizar.

Dentro deste contexto os agentes tornam a tecnologia em potencial para a integração com os sistemas de detecção de intrusão relatados nesta pesquisa, pois por meio destas capacidades de raciocínio e tomada de decisão podem melhorar o desempenho do IDS.

3.1 DEFINIÇÃO DE AGENTES

A definição de agentes inteligentes é difícil de ser determinada, pois os próprios autores possuem definições diferentes, algumas com similaridades, outras bastantes distintas, formando um variado quadro de definições. Assim, as mais citadas pela literatura são listadas a seguir.

Agente é um elemento de uma sociedade que pode perceber mudanças no aspecto da mesma e aplicar ações afetando-a, e estas percepções e ações podem ser feitas individualmente ou em cooperativa com outros agentes (LUGER, 2004).

Tudo o que pode ser capaz de perceber o ambiente por meio de sensores e agir sobre este mesmo ambiente por meio de atuadores é considerado um agente. Ele é algo que interage com o ambiente que está inserido, podendo mudar sua forma de atuação com o passar do tempo, devido a sua capacidade de aprendizagem, diferentemente dos programas computacionais tradicionais que somente executam uma seqüência pré-determinada de comandos (RUSSELL; NORVIG, 2004).

Uma agente é uma entidade virtual, capaz de agir sobre o ambiente que está inserido, podendo comunicar-se com outros agentes inseridos no mesmo. É capaz de perceber de modo limitado e age, por meio de recursos próprios, sobre o mesmo para que possa atingir seus objetivos levando em conta os resultados de suas funções de percepção e comunicação (REZENDE, 2005).

Observando os conceitos supracitados verifica-se que os mesmos possuem a definição de algumas características bastante similares, assim como a capacidade de percepção dos eventos que acontecem no ambiente em que o agente esteja inserido, bem como o poder de ação que o mesmo possa exercer sobre o ambiente. Na Figura 11 está ilustrada a arquitetura de um agente assim, como a forma que ele interage com o ambiente.

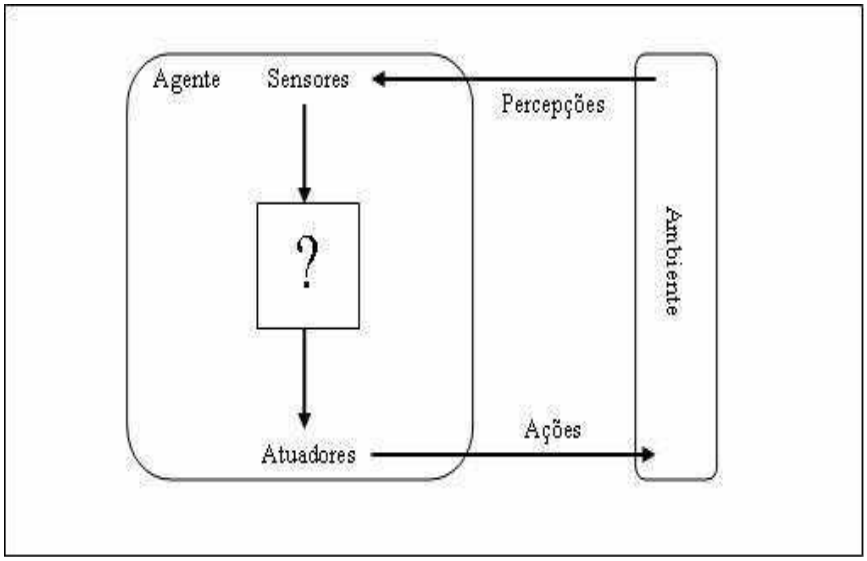

Figura 11. Interação dos agentes com o ambiente por meio de sensores e atuadores
Fonte: RUSSEL, S.; NORVIG, P. (2004)

3.2 AMBIENTES DOS AGENTES

No campo da IA pode surgir uma grande quantidade de ambientes diferentes, que podem ser divididos em categorias. Em grande parte são estas dimensões que auxiliam na determinação do projeto de agente apropriado e na aplicabilidade das técnicas de implementação dos mesmos. Cada dimensão possui uma característica distinta como podemos observar a seguir (RUSSELL; NORVIG, 2004):

a) completamente observável *versus* parcialmente observável: quando o agente tem acesso ao estado completo do ambiente e detecta todos os fatores relevantes para a escolha da ação, dizemos que o ambiente é totalmente observável. Já um ambiente pode ser considerado parcialmente observável quando ocorre uma falha como um ruído, ou caso o sensor que o monitore esteja ausente de alguma funcionalidade;
b) determinístico *versus* estocástico: se por meio do estado atual pode-se determinar completamente o próximo estado, dizemos que ele é determinístico. Caso contrário ele é considerado estático;
c) episódico *versus* seqüencial: em um ambiente de tarefas episódico a experiência do agente é dividida em episódios atômicos, ou seja, onde cada episódio é independente das ações executadas por episódios anteriores. Os ambientes de tarefas seqüenciais possuem características opostas as do episódico, ou seja, as decisões ou ações tomadas podem afetar decisões futuras;
d) estático *versus* dinâmico: caso um ambiente possa se alterar enquanto um agente realiza análises, dizemos que o ambiente de tarefa é dinâmico, caso contrário o mesmo é considerado estático;

e) discreto *versus* contínuo: para diferenciar ambientes discretos e contínuos deve-se levar em consideração basicamente o estado do ambiente, o modo como o tempo é tratado e as percepções e ações do agente. Por exemplo, em um jogo de xadrez encontra-se um número finito de estados distintos, assim como um conjunto discreto de percepções e ações, já o ambiente de direção de um veículo apresenta um problema de estado e tempo contínuos;

f) agente único *versus* multiagente: agente único indica o ambiente que possui um único agente implantado, já ambientes multiagentes são os que possuem dois ou mais agentes inseridos.

Após verificarmos os possíveis ambientes que os agentes possam ser inseridos, podemos verificar também os tipos dos mesmos, suas características e particularidades.

3.3 TAXONOMIA DE AGENTES

Pode-se considerar agente qualquer sistema que esteja situado em um ambiente e que tenha a capacidade de atuar sobre este sem a intervenção direta ou indireta de outros agentes, sendo estes humanos ou não (COSTA; SIMÕES, 2004).

Levando em consideração a afirmação anterior, podemos dizer que existem vários outros tipos de agentes além dos computacionais. Em vista desta diversidade de agentes, ocorre a necessidade de classificação geral para distinção dos agentes pela sua área de atuação.

Os agentes podem ser divididos em três grandes grupos: biológicos, robóticos e computacionais. O primeiro grupo se refere aos agentes naturais, o segundo grupo aos artificiais e o terceiro aos agentes existentes em aplicativos computacionais. Dentre estes três

ainda pode-se dividir os agentes computacionais em dois outros grupos: os agentes de *software* e os agentes de vida artificial (COSTA; SIMÕES 2004).

A Figura 12 ilustra a hierarquia supracitada. Mesmo existindo três grupos de agentes, na realização da pesquisa, iremos focar somente em um deles, os agentes computacionais.

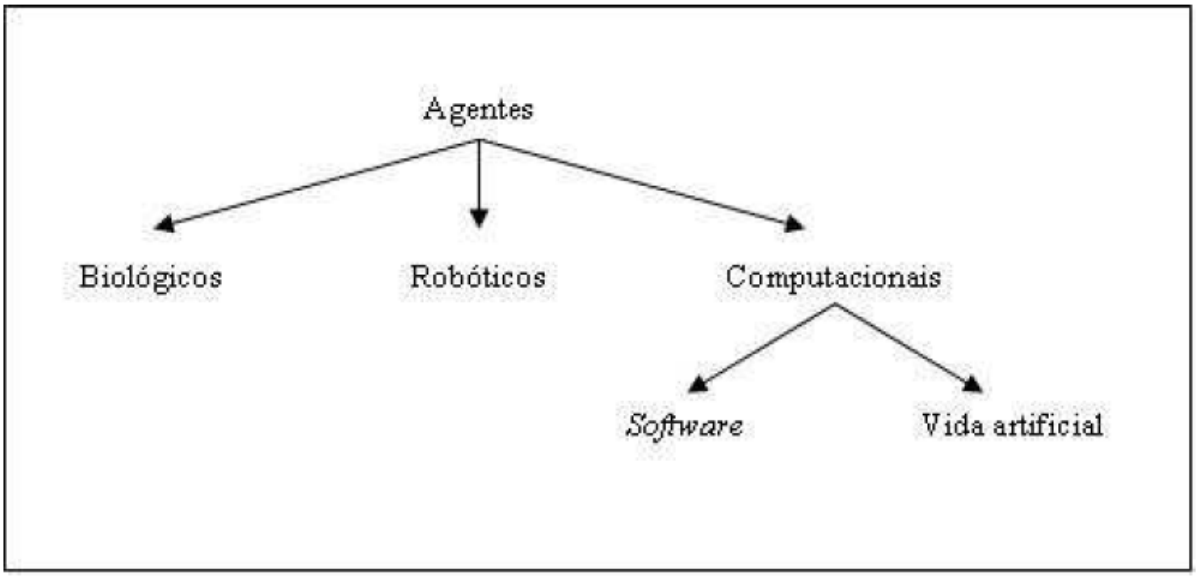

Figura 12. Tipos de agentes
Fonte: COSTA, E.; SIMÕES, A. (2004)

A classificação dos agentes também pode ser feita de forma mais específica, ou seja, pode-se fazer uma nova classificação dentro dos grandes grupos utilizando fatores mais específicos para o agrupamento dos mesmos.

Para classificar os agentes computacionais de maneira mais específica usa-se alguns integrantes chaves (HUHNS; SINGH, 1997):

a) autonomia de decisão: capacidade de analisar uma situação, gerar alternativas de ação e escolher a que melhor atenda seus objetivos;
b) autonomia de execução: capacidade de ser independente, de atuar e agir sem a intervenção de outro agente;

c) competência de decidir: capacidade de atuação sem a intervenção externa;
d) existência de uma agenda própria: capacidade de criar listas que contenham os objetivos, onde podem contribuir no alcance das metas do agente;
e) reatividade: capacidade de reagir as mudanças do ambiente;
f) adaptabilidade: capacidade de adaptar suas decisões na decorrência de eventos desconhecidos;
g) mobilidade: capacidade de mover-se e de execução em outros ambientes;
h) personalidade: capacidade de comportamento semelhantemente ao comportamento humano, como emoção;
i) interatividade com o usuário: capacidade de interagir com usuário levando em consideração possíveis mal-entendimentos e falhas;
j) ambiente de atuação: refere-se ao local de atuação do agente, por exemplo, *desktop* ou Internet;
k) comunicabilidade: capacidade do agente comunicar-se com outros agentes, para que com isto possa chegar em suas metas com maior facilidade.

A partir das características apresentadas como, autonomia de decisão e de execução, competência para decidir, adaptabilidade, modalidade, ambiente de atuação e outros, foi criada a taxonomia apresentada na Figura 13 que classifica os agentes de acordo com os seguintes eixos (REZENDE, 2005):

a) eixo cognitivo: modelo de representação interna de ambiente e dos outros agentes baseados em estados mentais; um modelo racional (cognitivo) ou apenas baseado em reação aos estímulos provocados pelo ambiente (reativo);
b) eixo de foco: um agente pode enfatizar similaridade com características

humanas (estrutural) ou com comportamentos (comportamental) ;

c) eixo de atuação: um agente pode atuar isoladamente (isolada) ou junto com outros agentes (social);

d) eixo ambiental: um agente pode atuar no *desktop* (agente de *desktop*) ou em rede Internet ou Intranet (agente de Internet).

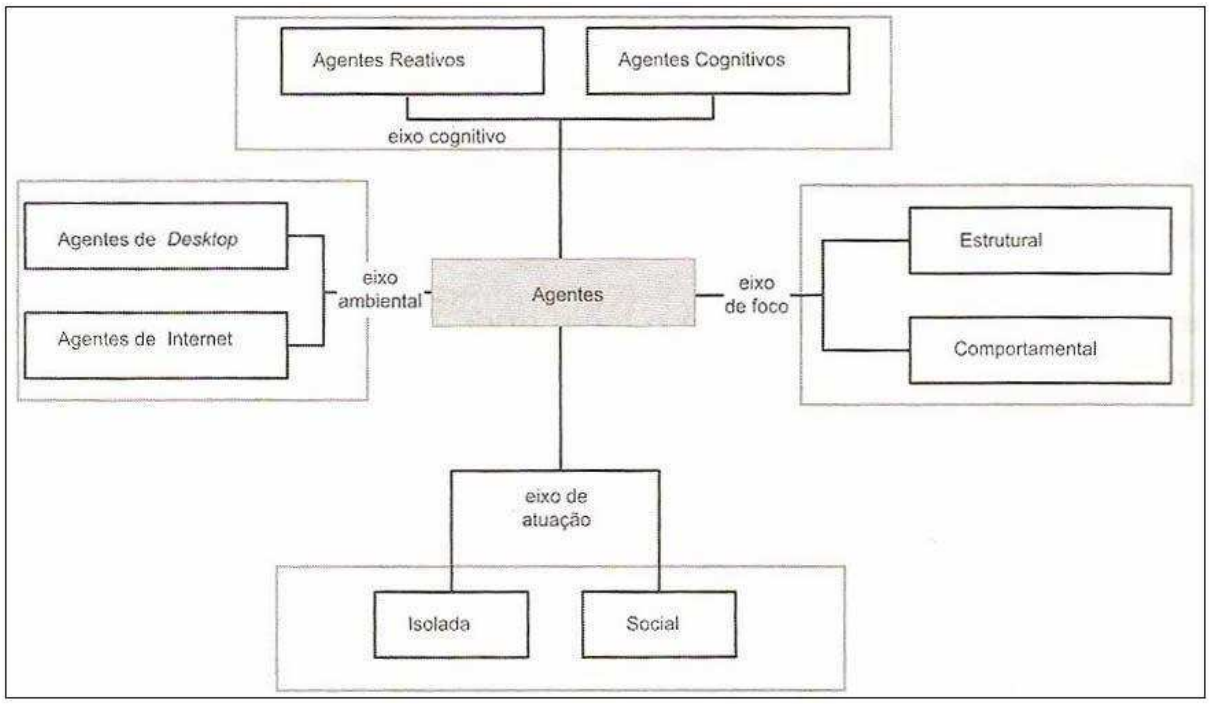

Figura 13. Taxonomia de agentes
Fonte: REZENDE, S. (2005)

Por fim após verificarmos as taxonomias dos agentes, podemos afirmar que o que mais dispertou interesse nesta pesquisa e o que melhor se enquadrou com o objetivo da mesma é o agente computacional que possui o eixo de atuação ambiental.

3.4 PROPRIEDADES DOS AGENTES

Os agentes são aplicados em diferentes ambientes e voltados para a resolução de diferentes tipos de problemas. Para que o agente venha a atingir um desempenho

considerável, é necessária uma boa estruturação na qual garanta o suporte para a tomada de decisão que o mesmo desempenhará.

Os agentes devem possuir mecanismos de decisão que o permitam escolher de que maneira atuar, considerando para isto uma agenda de objetivos, um perfil de atuação e um ambiente de atuação (REZENDE, 2005).

Levando em consideração as configurações internas dos agentes, podemos dividi-los em quatro grupos: agentes reativos simples, baseados em modelos, baseados em objetivos, baseados em utilidades e agentes com aprendizagem (RUSSEL; NORVIG, 2004).

A seguir encontram-se mais detalhadamente as características inerentes aos tipos de agentes inteligentes, como as do reativo simples que será o modelo utilizado para no desenvolvimento desta pesquisa.

3.4.1 Agente Reativo Simples

É a estrutura mais simples que um agente pode possuir, nela o mesmo baseia-se, para a tomada de decisão, somente na percepção da ação atual e na posição em que se encontra, não se preocupando com o histórico de ações anteriores (RUSSEL; NORVIG, 2004).

Na Figura 14 encontra-se a estrutura interna do agente reativo simples, que ilustra como as regras de condição-ação permitem ao agente fazer conexão entre a percepção e a ação.

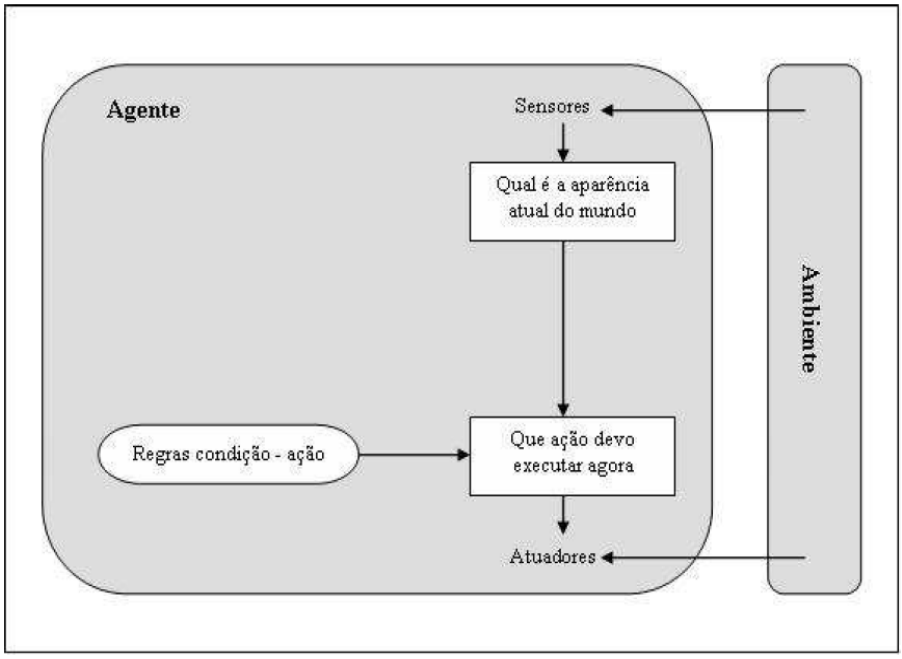

Figura 14. Diagrama esquemático de um agente reativo simples
Fonte: RUSSEL, S; NORVIG, P. (2004)

Esta estrutura tem o benefício de ser simples, porém este acarreta alguns pontos fracos como a inteligência limitada. Além disso, este modelo só irá funcionar caso a decisão correta a ser tomada possa ser realizada levando em consideração somente o estado atual, ou seja, se o ambiente for completamente observável (RUSSEL; NORVIG, 2004).

3.4.2 Agentes Reativos Baseados em Modelos

Para suprir a possibilidade de observação parcial do ambiente onde o agente está inserido, foi criado um estado interno que depende do histórico de percepções para que o mesmo possa controlar a parte do ambiente que ele não pode ver. Para o controle das atualizações das informações do estado é necessário a inclusão de dois tipos de conhecimentos[6] no agente: informação de como o ambiente evoluiu e como as ações dos

[6] O conhecimento é a base do sistema, este contém fatos sobre o problema a ser resolvido e regras que mostram como o especialista raciocina para chegar a uma conclusão (BARRETO, 2001).

agentes afetam o ambiente (RUSSEL; NORVIG, 2004).

Assim, esta estrutura apresenta um nível de inteligência consideravelmente maior do que a dos agentes simples, pois possui variáveis que guardam informações sobre o ambiente que o mesmo está inserido, permitindo desta forma que consiga se adaptar com as alterações do meio.

A Figura 15 ilustra a estrutura interna do agente reativo baseado em modelos, apresentando as suas relações internas entre as percepções, os estados, as variáveis de conhecimento e as ações do agente.

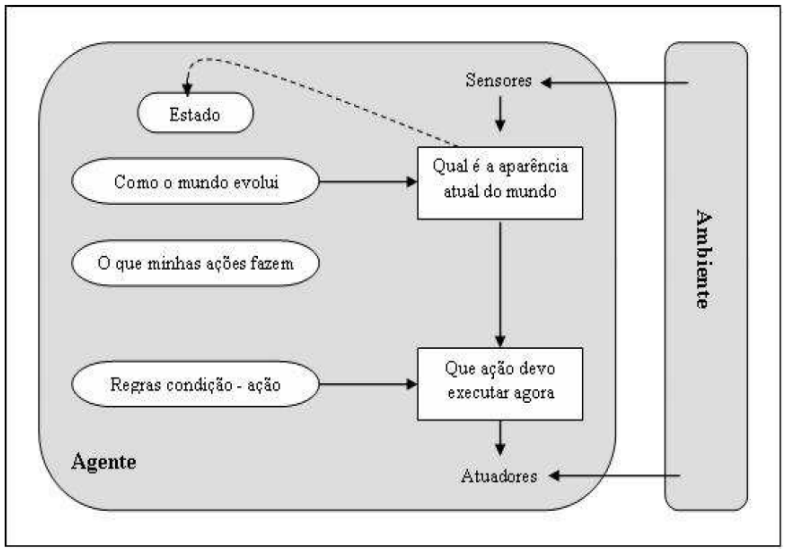

Figura 15. Um agente reativo baseado em modelo
Fonte: RUSSEL, S; NORVIG, P. (2004)

3.4.3 Agentes Baseados em Objetivos

Para tomar a decisão certa nem sempre as informações sobre o estado atual ou histórico de percepções são suficientes, necessita-se também de informações sobre objetivos. Existem situações que a seleção de ações é direta, porém em outras esta seleção pode resultar em longas seqüências de ações para encontrar um meio de atingir o objetivo (RUSSEL;

NORVIG, 2004).

Com base neste fundamento pode-se observar que é necessária a utilização de um sistema de informação de objetivos em situações nas quais o agente tenha que analisar grandes quantidades de ações. Na Figura 16 está ilustrada a estrutura interna do agente baseado em objetivos e também a capacidade deste modelo influenciar no resultado das ações e de escolher qual o deixará mais próximo de seu objetivo.

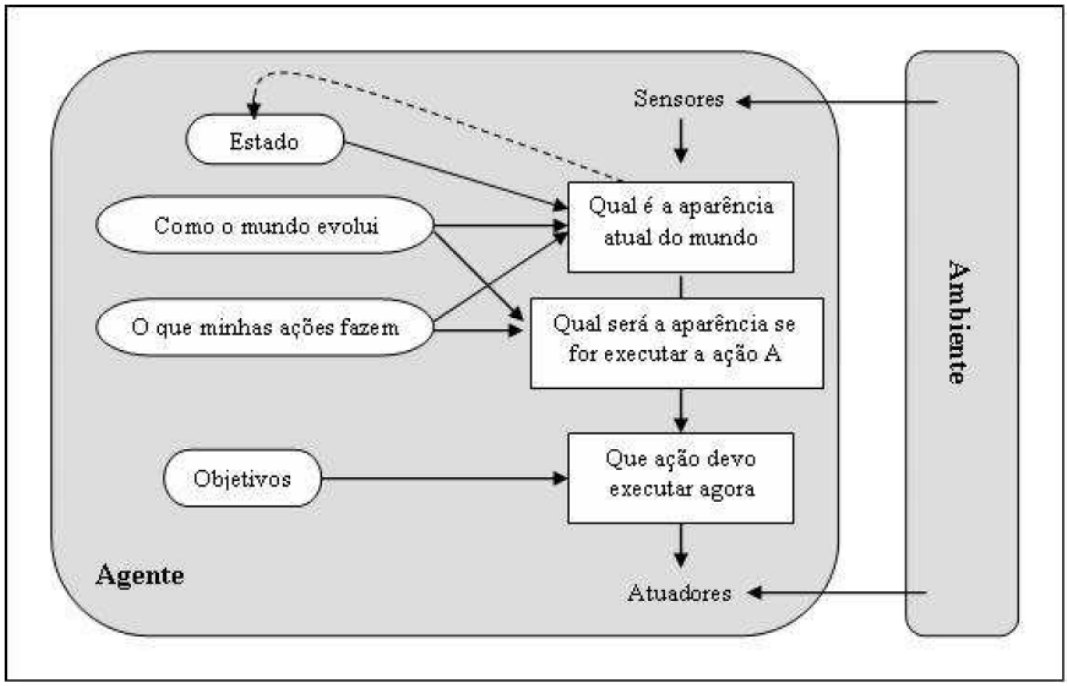

Figura 16. Um agente baseado em modelos e orientado para objetos
Fonte: RUSSEL, S; NORVIG, P. (2004)

3.4.4 Agentes Baseados em Utilidade

Os objetivos sozinhos não são capazes de gerar um comportamento de alta qualidade aos agentes. Assim, os baseados em utilidade fazem uma comparação entre diferentes estados do ambiente e verifica-se o grau de satisfação do estado para o agente, ou seja, escolhe-se o estado de maior utilidade (RUSSEL; NORVIG, 2004).

Uma especificação completa permite ao agente tomar decisões nos casos onde os objetivos são inadequados, como em situações que possuem objetivos contraditórios e quando existem vários objetos a serem alcançados e nenhum possa ser atingido com certeza (RUSSEL; NORVIG, 2004).

A Figura 17 ilustra a estrutura interna dos agentes baseados em utilidade, onde possui uma função de utilidade que mede suas preferências entre os estados e em seguida escolhe a ação que melhor alcance e a utilidade esperada (RUSSEL; NORVIG, 2004).

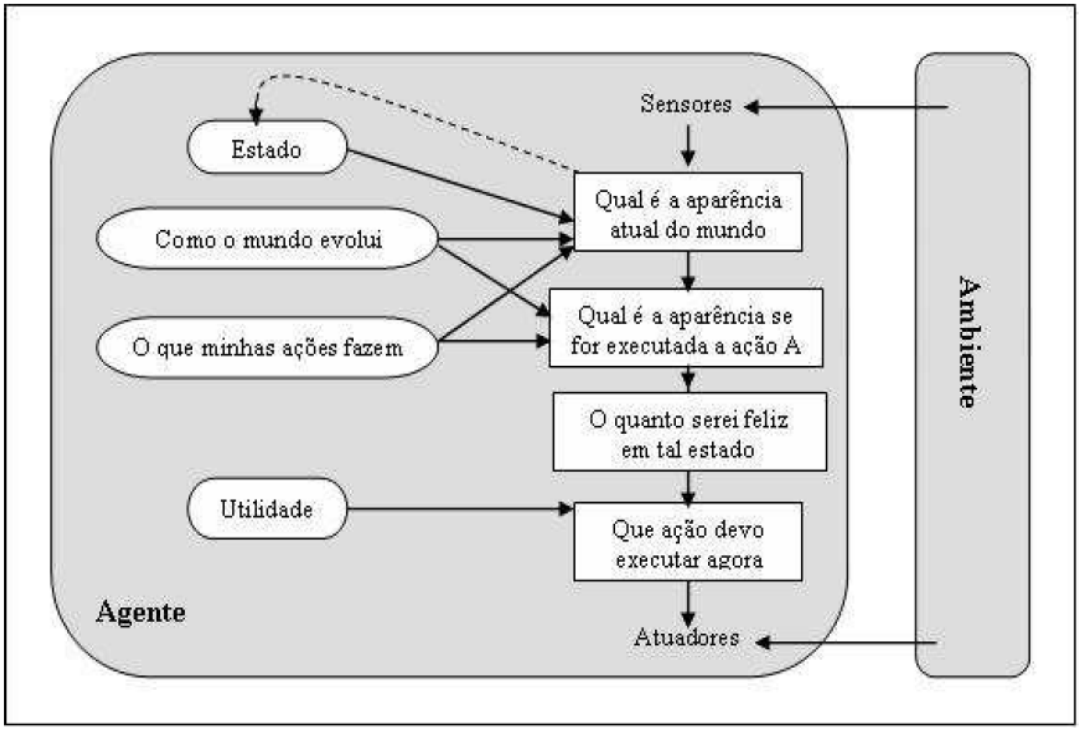

Figura 17. Agente baseado em modelo e orientado a utilidade
Fonte: RUSSEL, S; NORVIG, A. (2004)

3.4.5 Agentes com Aprendizagem

Aprender denota mudanças em um sistema que é adaptativo, e que permita realizar as tarefas da mesma classe, de modo mais eficiente e com maior perspectiva para o

futuro (SIMON, 1983).

A necessidade de máquinas que possuíssem capacidade de aprendizado já era relatada desde 1950 quando Alan Turing considerou a idéia de realmente programar máquinas inteligentes à mão, o que levou o mesmo a estimar que seria um trabalho árduo, concluindo então a necessidade de algum mecanismo ou método mais eficiente para realizar esta função (RUSSEL; NORVIG, 2004).

A aprendizagem pela máquina preocupa-se com o desenvolvimento de programas que são capazes de construir um novo conhecimento ou modificar de modo útil conhecimentos já existentes, utilizando para isto informações de seu ambiente (MICHALSKI; KODRATOFF, 1990).

Levando em consideração os fundamentos acima, podemos concluir que é essencial a introdução de aprendizagem às maquinas em específico aos agentes, que possam estar sendo introduzidos em ambientes ou situações até antes desconhecidas, sem a necessidade de inclusão manual do conhecimento referente a este ambiente ou à estrutura do agente.

Um agente de aprendizagem pode ser dividido em quatro elementos: componente de aprendizagem, de desempenho, crítico e o gerador de problemas. A distinção mais importante se dá entre o elemento de aprendizado, responsável pelo aperfeiçoamento, e o elemento de desempenho, pela seleção de ações externas. Este último corresponde ao agente completo visto anteriormente que percebe o ambiente e age sobre o mesmo. A estrutura de aprendizado utiliza realimentação do elemento crítico para verificar como o agente está se comportando e determina de que maneira o elemento de desempenho deve ser modificado para funcionar melhor no futuro. Já o gerador de problemas serve para sugerir ações que levarão a novas experiências, ou seja, é responsável por ações exploratórias para a busca de melhorias (RUSSEL; NORVIG, 2004).

A Figura 18 ilustra a estrutura interna de um agente baseado em conhecimento, nela podendo-se observar os componentes que a estrutura possui além da relação entre os mesmos.

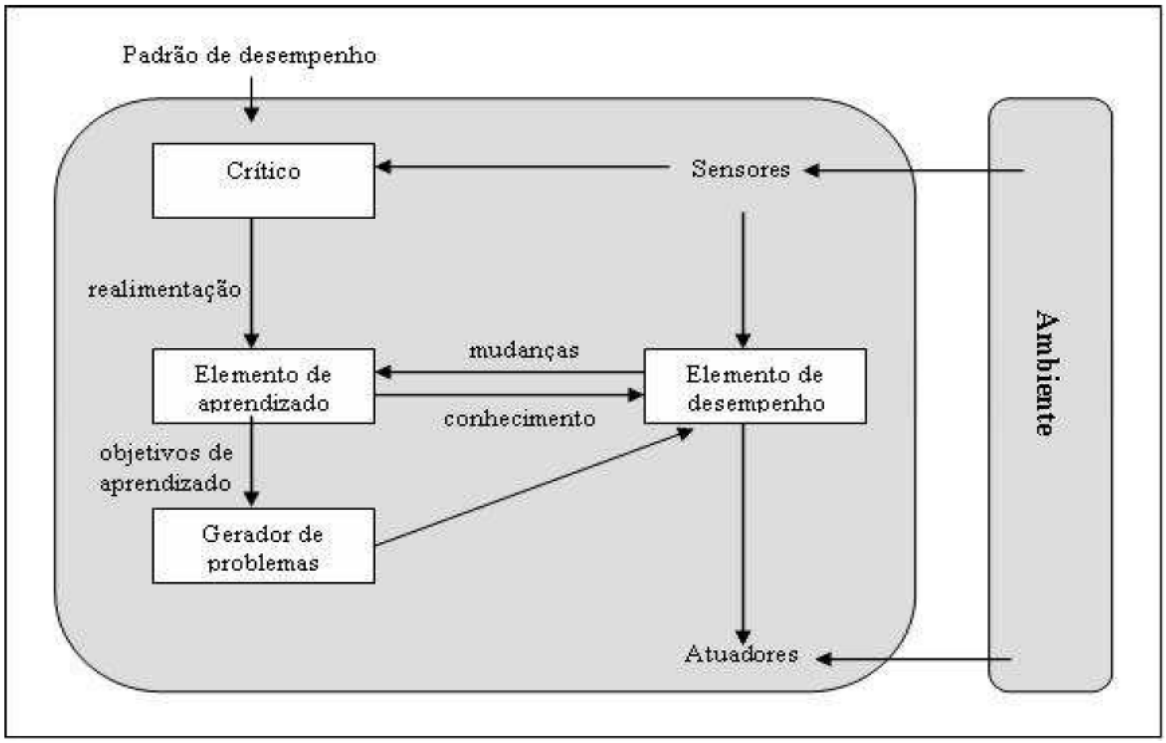

Figura 18. Um modelo geral de agentes com aprendizado
Fonte: RUSSEL, S; NORVIG, P. (2004)

3.5 SISTEMAS MULTIAGENTES

Sistemas multiagentes (SMA) representam um conjunto de agentes autônomos, que têm uma existência própria e objetivos próprios a atingir (REZENDE, 2005).

Como não existem problemas previstos nos ambientes onde os agentes serão implantados, os mesmos são inseridos de modo genérico, para que possa ser instanciado em um caso particular quando um problema é colocado para eles resolverem. Deve-se permitir meios que garantam que os agentes consigam se comunicar para que possam resolver os

problemas aplicados aos mesmos (RUSSEL; NORVIG, 2004).

Sendo assim, podemos dizer que os sistemas multiagentes têm a capacidade de auto-adaptação, pois é configurado inicialmente de uma forma genérica onde possam ser implantados em aplicações ou ambientes distintos e ao longo de sua monitoração o mesmo irá se adaptando a estas aplicações ou ambientes.

Todas as características apresentadas como, por exemplo, a auto-adaptação e a aprendizagem reforçam ainda mais a idéia da utilização de agentes para auxiliarem nas análises dos sistemas de detecção de intrusão inserindo conseqüentemente inteligência ao sistema citado.

Como os agentes inteligentes são baseados nos conhecimentos do administrador que podem estar representados por meio de regras, estas podem gerar resultados com certo grau de incerteza de pendendo as regras que forem ativas, em vista disto são utilizados outros métodos para minimizar tal grau. Um método bastante utilizado para resolver tal situação são os Fatores de Certeza.

4 FATORES DE CERTEZA

O desenvolvimento deste método teve início em 1970, onde foi projetado para a utilização em um sistema de diagnóstico médico baseado em regras chamado de MYCIN[7]. Por meio deste novo método atribuía-se a cada uma das regras do sistema um Fator de Certeza (FC), no qual ao final da análise quantificavam-se os graus de certeza que cada uma das regras, concluindo desta forma o grau de certeza da resposta (COSTA; SIMÕES, 2004).

Problemas que possuem um número pequeno de informações são difíceis de serem resolvidos, gerando desta forma um grande grau de incerteza sobre a solução do mesmo. Para tentar minimizar esta incerteza, neste processo atribuem-se fatores de certeza, onde cada hipótese recebe um grau de certeza, diminuindo desta forma a incerteza geral do problema.

FC é uma maneira simples de combinar crença e descrença em um único número que fica geralmente entre o intervalo -1 e 1. Um FC positivo significa que a evidência suporta a hipótese caso a média de crença seja maior que a média da descrença, já um FC negativo significa que a evidência favorece a negação da hipótese caso a média de crença seja menor que a de descrença (BARRETO, 2001).

O resultado final da análise deste método é um valor numérico que corresponde à diferença entre o valor do grau de certeza gerado pelas regras e o grau de incerteza gerado por estas mesmas regras. Caso resultado gerado seja negativo, a descrença da hipótese é favorecida, caso seja positivo quem se favorece é a crença da hipótese.

Os métodos dos fatores de certeza são baseados em sistemas periciais baseados em regras, ou seja, sistemas baseados em regras SE ENTÃO.

Na Figura 19 podemos verificar um exemplo do formato que a regra mencionada

[7] O MYCIN é um sistema baseado em regras utilizado para identificar as bactérias que causam infecções graves e para recomendar antibióticos, com a dose ajustada para o peso corporal do doente (COSTA; SIMÕES, 2004).

possui. A expressão, SE, no início da regra representa a condição para a execução da regra enquanto a expressão, ENTÃO, corresponde à resposta caso a regra tenha sido válida, já a letra “k” indica o grau de certeza que a regra possui.

Figura 19. Regra se então
Fonte: COSTA, E; SIMÕES, A. (2004)

O grau de confiança foi originalmente definido como Fator de Certeza (FC) que é que a diferença entre as médias da crença e descrenças, onde o mesmo é obtido da seguinte formula (BARRETO, 2001):

$$FC[H,E] = MC[H,E] - MD[H,E] \quad (1)$$

Onde:

a) $FC = [H,E]$ é o fator de certeza na hipótese *H* dada à evidência *E*;

b) $MC = [H,E]$ média de crença em *H* dado *E*;

c) $MD = [H,E]$ média de descrença em *H* dado *E*.

Como podemos observar o fator de certeza é gerado por meio do desdobramento de dois componentes, o $MC[H,E]$ e o $MD[H,E]$, ou seja, a medida da crença e da descrença considerando a evidência *E* (COSTA; SIMÕES, 2004).

Para se chegar ao valor do FC é necessário primeiramente realizar tanto o cálculo da média de crença e a da descrença das hipóteses geradas.

A média de crença, $MC[H,E]$, representa em que medida nossa crença em *H* vem aumentado dado a evidência *E*, conforme definido na função 2 (COSTA; SIMÕES, 2004).

Podemos verificar que o MC recebe a hipótese P(H), caso a crença na hipótese tenha sido confirmado sem que nenhuma evidência tenha ocorrido, ou seja, caso a hipótese P(H) for 1. Caso contrário, terá um aumento na crença desta hipótese levando em consideração a evidência $P(H|E)$.

$$MC[H,E] = \begin{cases} 1 \, se P(H) = 1 \\ \dfrac{\max[P(H|E), P(H)] - P(H)}{\max[1,0] - P(H)} \end{cases} \quad (2)$$

Quando há o aumento da crença em uma mesma hipótese por duas evidências distintas, os graus de crença destas duas hipóteses são combinados gerando então um novo MC, como mostra a função 3 (BUCHANAN; SHORTLIFFE, 1984, tradução nossa).

$$MC[H, E_1 \Lambda E_2] = \begin{cases} 0 \, se MD[H, E_1 \Lambda E_2] = 1 \\ MC[H, E_1] + MC[H, E_2] * (1 - P[H, E_1]) \end{cases} \quad (3)$$

A média de descrença, $MD[H,E]$, representa em que medida nossa descrença em *H* vem aumentando dado a evidência *E*, conforme definido na função 4 (COSTA; SIMÕES, 2004).

Podemos verificar que o MD recebe a hipótese P(H), caso a descrença na hipótese tenha sido confirmada sem que nenhuma evidência tenha ocorrido, ou seja, caso a hipótese P(H) for 0. Caso contrário, terá uma redução na crença desta hipótese levando em consideração a evidência $P(H|E)$.

$$MD[H,E] = \begin{cases} 1 se. P(H) = 0 \\ \dfrac{\min[P(H|E), P(H)] - P(H)}{\min[1,0] - P(H)} \end{cases} \quad (4)$$

Quando há a redução da crença uma mesma hipótese por duas evidências distintas, os graus de crença destas duas hipóteses são combinados gerando então uma nova MD, como mostra a função 5 (BUCHANAN; SHORTLIFFE, 1984, tradução nossa).

$$MD[H, E_1 \Lambda E_2] = \begin{cases} 0 .....se.....MC[H, E_1 \Lambda E_2] = 1 \\ MD[H, E_1] + MD[H, E_2] * (0 - P[H, E_1]) \end{cases} \quad (5)$$

Por meio da análise das formas podemos perceber que os resultados dos fatores de certeza variam em -1 e 1, percebemos também a simetria entre as medias de crença (MC) e de descrença (MD), levando em consideração esta simetria, podemos observar no Quadro 02 algumas característica.

Características	Valores
Variações	$0 \le MC \le 1$ $0 \le MD \le 1$ $-1 \le FC \le 1$
Certeza das hipóteses verdadeiras $P(H\|E) = 1$	$MC = 1$ $MD = 0$ $FC = 1$
Certeza das hipóteses falsas $P(\neg H\|E) = 1$	$MC = 0$ $MD = 1$ $FC = -1$
Perda de Evidência $P(H\|E) = P(H)$	$MC = 0$ $MD = 0$ $FC = 0$

Quadro 2. Características de medidas de crença, descrença e fatores de certeza
Fonte: BARRETO, I. (2001)

Por meio da análise feita sobre as informações adquiridas até o momento podemos observar que o FC final de uma hipótese estará entre um intervalo entre -1 e 1, assim quanto mais próximo dos extremos este fator estiver maior será a confiança depositada na resposta, como nos mostra a Figura 20.

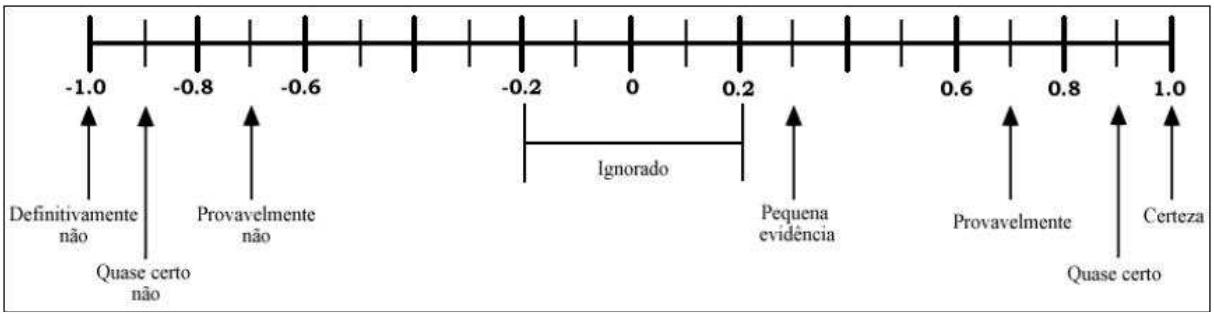

Figura 20. Intervalo dos fatores de certeza
Fonte: NICOLLET, M.C.; SANTOS, F.O. (1996)

Como podemos observar na Figura 20 os FC resultantes entre -0.2 e 0.2 estão sendo ignorados, pois constatou-se que os valores gerados entre este intervalo estavam prejudicando a eficiência do método.

Quando o sistema NYCIN foi desenvolvido, notou-se que o mesmo estava ativando uma grande quantidade de regras pouco relevantes, fazendo com que o desempenho do sistema caísse consideravelmente devido a demora na obtenção dos resultados. Definiu-se então, que para a ativação de uma regra é necessário que o antecedente dela possuísse FC maior que $|0{,}2|$, para que a regra analisada seja ativada, melhorando desta forma a eficiência do sistema (BARRETO, 2001).

Tendo em vista que houve esta mudança na validação das regras, necessitou-se a alterar a fórmula do calculo do FC, para que diminuísse o desequilíbrio causado por uma evidência fortemente negativa ou positiva.

Como a fórmula inicial de cálculo do (FC), representado pela Fórmula 1, permitia que evidências que possuíssem grau fortemente negativos ou positivos desequilibrassem o

sistema, E. H. Shortliffe e B. G. Buchanan modificaram a fórmula, inicialmente proposta no intuito de suprir a defasagem que a mesma possuía, como podemos verificar na formula 6 (BUCHANAN; SHORTLIFFE, 1984, tradução nossa).

$$FC = \frac{MC - MD}{1 - \min(MC, MD)} \tag{6}$$

Como podemos verificar o FC que anteriormente era apenas a diferença entre a medida de crença e a medida da descrença, passou a receber também a divisão do intervalo entre o maior valor de crença, que é 1, e o menor valor de média entre a crença e a descrença .

Tendo em vistas todos os métodos e características, podemos analisar de forma detalhadamente os trabalhos desenvolvidos que estão relacionados na mesma linha de pesquisa que este projeto possui.

5 TRABALHOS CORRELATOS

Devido ao crescente número de ataques às redes de computadores e fraudes que vem ocorrendo por meio da Internet, encontram-se diversos trabalhos relacionados à segurança da rede onde relatam métodos ou alternativas para minimizar a possibilidade de uma possível invasão. Um dos métodos que vem sendo adotado é a integração de Inteligência Artificial e Sistemas de Detecção de Intrusão (IDS), onde detecta de forma inteligente uma tentativa de intrusão.

Sendo assim, este capítulo tem o intuito de relatar alguns trabalhos que utilizam técnicas de IA integradas com os Sistemas de Detecção de Intrusão.

5.1 PIS: UM SISTEMA ROTATIVO DE PREVENÇÃO CONTRA INTRUSÕES

Esta pesquisa refere-se a um trabalho de conclusão de curso de Pós-Graduação em Ciências da Computação do Centro de Informática da Universidade Federal de Pernambuco (UFPE), realizado no ano de 2005 (MARTINS, 2005).

Devido à defasagem dos sistemas de detecção de intrusão tradicionais, o trabalho propõe integrar dados de diversas fontes, aumentar a qualidade das análises e integrar agentes distribuídos e agentes inteligentes, isto tudo será feito utilizando Proactive Intrusion Prevention Systems (PIPS) (MARTINS, 2005)..

Este trabalho apresenta uma nova abordagem para o problema da contínua monitoração que é denominada de Proactive Intrusion Prevention Systems (PIPS). O PIPS é responsável pela monitoração constante da rede através de varreduras periódicas que montam um perfil ativo da mesma. Este perfil é correlacionado com os eventos dos IDS produzindo desta forma uma análise mais refinada. Esta abordagem consiste na utilização de agentes que

realizam a coleta de dados de forma distribuída e os disponibiliza para serem processados por um analisador central que tem uma visão global da rede (MARTINS, 2005).;

Na Figura 21 encontra-se uma tela na qual foi implementada para que o usuário possa estar configurando e analisando o sistema proposto.

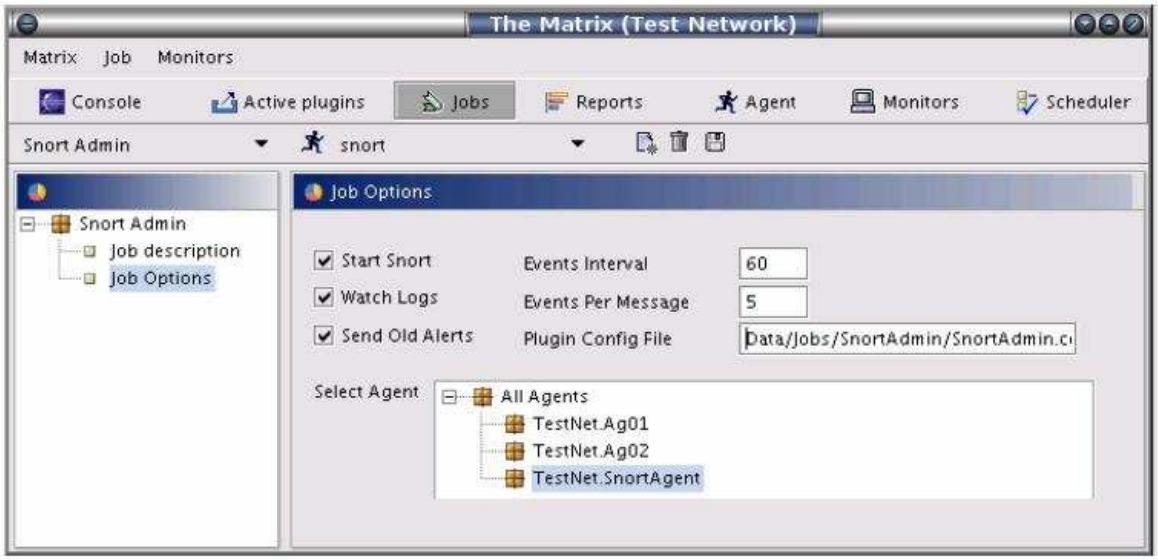

Figura 21. GUI: Console de Operações
Fonte: MARTINS,C. S. (2005)

O sistema atingiu os resultados esperados com a utilização do *framework* distribuído e com a arquitetura flexível, agregando e correlacionando dados de diferentes ferramentas. As análises e correlações têm permitido uma redução significativa no volume de falsos positivos, produzindo resultados satisfatórios no dia-a-dia da operação da ferramenta. Pode-se criar também um inventário de rede onde possibilita a geração de análises históricas e detectar tendências na evolução da segurança no ambiente (MARTINS, 2005).

5.2 UMA ABORDAGEM DE DETECÇÃO DE INTRUSÃO BASEADA EM SISTEMAS IMUNOLÓGICOS ARTIFICIAIS E AGENTES MÓVEIS

A presente pesquisa refere-se a dissertação de Mestrado em Ciências da Computação submetido à Universidade Federal de Santa Catarina (UFSC), realizado no ano de 2005 (MACHADO, 2005).

Este trabalho apresenta uma abordagem para a detecção de intrusão inspirada nos conceitos, princípios e características do sistema imunológico humano. Devido a esta forte inspiração, proporcionou-se a modelagem e implementação de um sistema de detecção de intrusão com importantes características, as quais permitem a construção de uma solução computacional para os processos de reconhecimento, análise, memorização e geração de respostas pró-ativas (MACHADO, 2005).

É utilizado neste trabalho técnica de detecção baseada em anomalias e tem por base a monitoração dos registros de auditoria dos sistemas operacionais *Unix-like.* A arquitetura é baseada e m *host* e distribuída, o processo de geração de eventos é realizado pelo *Syslog-ng*, a análise é feita pela ferramenta *Logcheck* e a persistência e distribuição dos registros, assim como a incorporação das reações reativas e pró-ativas são realizadas por uma arquitetura baseada em agentes móveis. Os resultados gerados pelo processo de análise são classificados como ataques, violações de segurança e eventos de segurança. Para contribuir com o sistema foram incorporadas algumas técnicas dos sistemas imunológicos artificiais assim como a detecção de anomalias, memorização e segurança de agentes móveis (MACHADO, 2005).

O sistema desenvolvido permitiu uma redução significativa do número de registros analisados, como também facilitou a observação e análise eficaz das atividades dos

hosts e possibilitou a implementação de respostas pró-ativas (MACHADO, 2005).

5.3 A FRAMEWORK FOR AN ADAPTIVE INTRUSION DETECTION SYSTEM WITH DATA MINING

Esta pesquisa foi desenvolvida em 2002, submetida à The Pennsylvania State University CiteSeer Archives, ela tem como objetivo de melhorar o desempenho dos sistemas de detecção de intrusão utilizando regras *fuzzy* associadas à mineração de dados (HOSSAIN, BRIDGES, 2002, tradução nossa).

O objetivo do trabalho é desenvolver um perfil para as atividades normais do ambiente, aplicando técnicas de mineração de dados, para que atividades intrusas anormais possam ser detectadas comparando-as com o perfil das atividades normais geradas e por fim fornecer um quadro geral de uma detecção por meio da associação de regras *fuzzy* e mineração de dados (HOSSAIN, BRIDGES, 2002, tradução nossa)..

A auditoria dos dados é formatada por uma tabela de dados onde cada linha é um registro de auditoria e de cada coluna é um campo (característica do sistema) dos registros de auditoria. As regras dos usuários são mineiradas a partir de dados da auditoria e adicionados em um formulário agregando regras para o usuário normal do perfil. Para analisar um login de usuário, padrões freqüentes são extraídos a partir da seqüência de comandos durante a sessão, e este novo padrão é fixado e comparado com o perfil padrão estabelecido anteriormente. A similaridade das funções é utilizada para avaliar desvios de geras e alertar em caso de comportamentos suspeitos. Para minimizar a possibilidade de erro na verificação destes comportamentos, aplicou-se lógica *fuzzy* ao sistema (HOSSAIN, BRIDGES, 2002, tradução nossa)..

A Figura 22 apresenta uma arquitetura do quadro da verificação de perfil proposto pelo trabalho. O processo começa com um conjunto inicial de auditoria de dados. Algoritmos genéticos seriam usados para sintonizar a chamada dos parâmetros da função *fuzzy* . A função *fuzzy* será associada às regras mineiradas e será aplicada a regulamentação do perfil. Durante cada processo, a auditoria de dados na parte incremental realizará a comparação com a regra do perfil definida como normal. Dependendo do grau de similaridade entre os mesmos, pode haver três possibilidades. O sistema pode continuar com o perfil normal atual, um alerta de intrusão pode ser gerado e o normal conseqüentemente não será atualizado ou o mesmo poderá ser atualizado com os dados da auditoria atual.

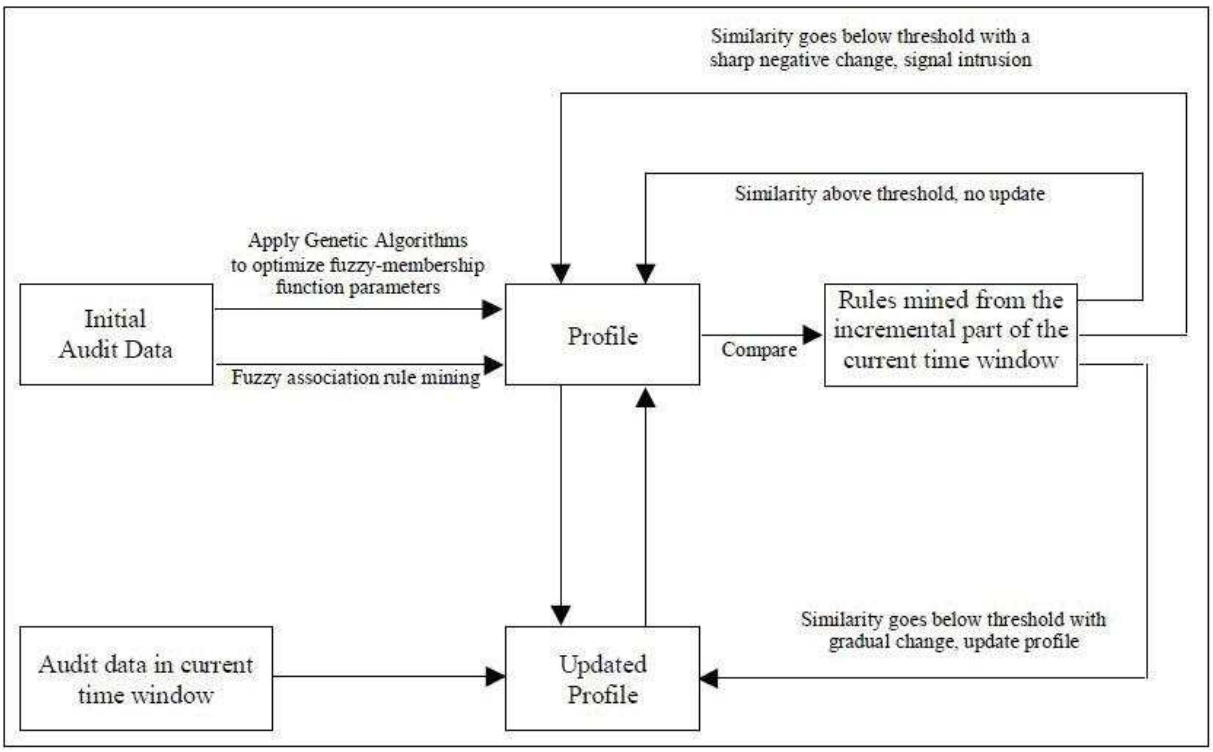

Figura 22. Quadro adaptativo do Sistema de detecção de Intrusão
Fonte: (HOSSAIN, M; BRIDGES, S (2002).

Contudo, o autor concluiu que com a mineração de dados integrada com lógica *fuzzy* foi possível desenvolver métodos automáticos de detecção de intrusão eliminando desta forma umas das limitações dos IDS tradicionais que é a falta de adaptabilidade à mudanças de padrão e comportamento, tornando a ferramenta mais robusta e independente.

6. TRABALHO DESENVOLVIDO

Após ter adquirido o conhecimento necessário, relatados nos capítulos anteriores, foi possível desenvolver o software proposto nesta pesquisa, onde foi denominado de Dimitry[8], a implementação do mesmo será descrito neste capítulo. Serão enunciadas as ferramentas e conceitos incorporados, a modelagem do sistema, as características que possui restrições, entre outros. Por fim, serão mostrados os testes realizados e os resultados obtidos.

6.1 FERRAMENTAS UTILIZADAS

No desenvolvimento do projeto proposto foi preciso à utilização de alguns recursos ou ferramentas já prontos, nesta etapa da pesquisa será relatado a importância e a função que cada ferramenta utilizada possui dentro do software proposto.

Inicialmente foi preciso realizar a escolha da linguagem de programação a ser utilizada para o desenvolvimento do programa, após algumas pesquisas verificou-se que a tecnologia Java melhor se enquadrou com as necessidades da pesquisa, pois esta possui recursos já prontos que auxiliam na identificação de interfaces de rede e na captura de pacotes. Outro ponto forte na escolha desta linguagem é que a mesma é gratuita além ser compatível com diversos sistemas operacionais.

A ferramenta de desenvolvimento escolhida foi o NetBeans 6.1, pois também é gratuito, possui recursos que facilitam a programação, a conexão com o banco de dados e o desenvolvimento de interfaces gráficas.

No intuito de registrar todos os *logs* de tráfego gerado foi incorporado ao sistema

[8] O nome Dimitry foi escolhido em virtude de Dimitry Ivanovsky ser o responsável pela descoberta do vírus biológico em 1890 (ANDRADE, 2009).

um banco de dados Firebird que também é gratuito e de fácil manipulação.

O *Jpcap* é uma API implementada na biblioteca *WinPcap* que possui funcionalidades para identificação de interfaces de rede e captura de pacotes, onde permite manipular de forma fácil os pacotes que trafegam em uma interface para que possam ser analisados pelas regras de detecção de invasão da ferramenta.

6.2 ANÁLISE E SELEÇÃO DA SINTAXE DAS REGRAS DO SNORT

Após realizar, no Capítulo 1, o levantamento das variáveis que podem ser incorporadas às regras do Snort, foi possível definir quais delas são essenciais para o funcionamento de um IDS.

Como o sistema desenvolvido utiliza as regras do Snort com banco de assinaturas, a avaliação destas variáveis foi essencial no desenvolvimento do protótipo sugerido. Porém, nem todas as variáveis suportadas pelo Snort foram incorporadas ao Dimitry, descartando-se as que não possuíam relevância para o objetivo deste trabalho, no qual é a comprovação de que um IDS com métodos inteligentes venha a ter uma performance melhor do que o mesmo sem estes métodos.

Mesmo o sistema não relevando algumas das variáveis das regras do Snort é possível que as mesmas possam ser utilizadas sem quaisquer alteração, pois o Dimitry as interpreta coletando somente as que lhe são pertinentes. Estes atributos são mostrados no Quadro 3 onde podemos observar também uma breve explicação sobre sua funcionalidade e a categoria a qual pertence.

Categoria	Atributo	Descrição	Funcionalidade
Cabeçalho	Ação	Ação da Regra	Destro das ações existentes o sistema suporta somente Alerta, ou seja, Ação do tipo Alert
	Protocolo	Protocolo a ser Analisado	Dentro dos protocolos existentes suporta somente protocolos TCP e UDP
	Rede Origem	IP que Originou o Pacote	Suporta qualquer numero IP
	Porta Origem	Porta que Originou o Pacote	Suporta qualquer numero de Porta
	Direção	Direção da Transferência	Suporta somente transferências do tipo "->", ou seja, da direita para esquerda
	Rede Destino	IP que se Destina o Pacote	Suporta qualquer numero IP
	Porta Destino	Porta que se Destina o Pacote	Suporta qualquer numero de Porta
Opções	msg	Mensagem do Alerta	Mostrar a mensagem da regra para o operador do sistema
	reference	Referencia da Regra	Mostrar a referencia atribuída à regra
	classtype	Classificação do Ataque	Mostra a classificação do ataque (Caso seja Confirmado)
	content	Assinatura Suspeita	Comparar esta assinatura suspeita com as assinaturas dos pacotes que trafegam
	nocase	Restrições Maiúsculas e Minúsculas	Desativar a restrição entre letras maiúsculas e minúsculas
	flags	Flags da Regra	Verifica quais flags do pacote estão ativas
	sid	Identificador da Regra	Identifica a regra de maneira única
	ver	Revisão da Regra	Indica qual versão da regra
	priority	Prioridade da Regra	Indica qual a prioridade da regra

Quadro 3. Atributos das regra suportados pelo Dimitry

Como podemos verificar no Quadro 3, foram selecionados os elementos principais para o funcionamento do Dimitry, existindo também alguns deles que dão suportes a valores limitados, como por exemplo, o atributo *ação* que suporta somente alertas e o campo “protocolo” onde suporta somente TCP e UDP.

As regras vistas neste tópico são carregadas todas as vezes que o sistema inicializada uma nova consulta, onde estas são implantadas em uma classe específica do mesmo, ou seja, pode-se criar novas regras mesmo com o programa em execução, onde para carregá-las basta apenas parar a análise e iniciá-la novamente.

6.3 IMPLEMENTAÇÃO DO APLICATIVO IDS

Neste tópico iremos abordar todas as etapas pertencentes ao desenvolvimento do sistema de detecção de intrusão tradicional. Basicamente encontramos três etapas: modelagem do banco de dados de *logs*, modelagem da aplicação e desenvolvimento da aplicação.

6.3.1 Modelagem do Banco de Dados de *Logs*

Após o sistema conseguir encontrar um ataque e gerar um alerta, é necessário uma estrutura onde possa ser salvo os eventos e *logs* gerados pelo mesmo, para que futuramente possa ser analisado o desempenho do protótipo. Com o intuito de suprir esta necessidade optou-se pelo desenvolvimento de um banco de dados Firebird, devido a sua simplicidade e por ser gratuito.

A princípio, nesta etapa do desenvolvimento o banco criado não influencia na análise dos pacotes que trafegam, ou seja, ele apenas armazena os eventos e *logs* gerados pelo IDS, como já descrito.

Atendendo os requisitos desta etapa, na qual é o desenvolvimento de um IDS tradicional, foi necessário apenas criar duas tabelas, onde a primeira salva todo o tráfego que foi capturado pelo sistema, e a segunda guarda as informações dos pacotes que o sistema suspeitou ser um ataque, como também armazena o resultado da análise do mesmo. Na Figura 23 podemos verificar mais detalhadamente cada uma das tabelas.

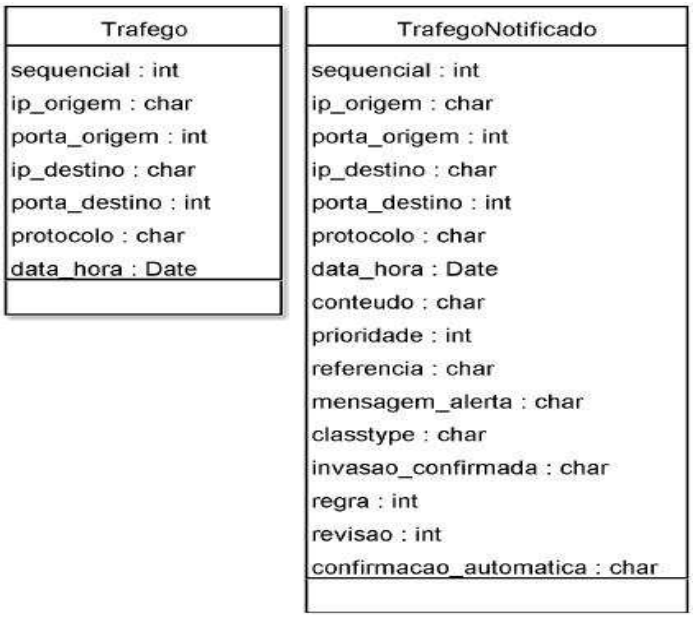

Figura 23. Estrutura do banco de dados do Dimitry tradicional

Como podemos observar as tabelas não possuem relação, ou seja, uma não depende da outra, pois as mesmas existem apenas para guardarem os *logs gera*dos. A de tráfego é relativamente menor que a do tráfego notificado, esta diferença existe devido a de tráfego guarda as informações antes do IDS verificar as assinaturas não havendo então os dados relativos às regra e por conseqüência não necessitando salvá-las, o contrário ocorre com a tabela de tráfego notificado.

A tabela chamada “tráfego” possui sete itens onde cada um deles possui uma função específica, como podemos verificar a seguir:

a) seqüencial: tem por objetivo atribuir um código único ao *log* salvo para que este possa ser facilmente encontrato;
b) ip_origem: identifica o IP que originou o pacote que foi encontrado e gerado o *log*;
c) porta_origem: salva o número da porta pela qual o IP de origem encaminhou o pacote;
d) ip_destino: guarda o número IP da máquina que irá ou que recebeu as

informações enviadas pelo IP de origem;

e) porta_origem: serve para identificar por qual porta os dados estavam sendo enviados;
f) protocolo: informa qual era o protocolo utilizada para a transmissão das informações;
g) data_hora: nos mostra a data e hora que o *log* foi gerado.

Já a outra, chamada de “trafegonotificado” possui dezesseis campos, onde sete deles são idênticos aos da Figura 23, possuindo também as mesmas funções. As funções das outras nove variáveis são:

a) conteúdo: mostrar a assinatura duvidosa que o pacote possuía;
b) prioridade: deixar salvo a prioridade da regra que detectou a anomalia;
c) referencia: informar a referência da regra que gerou o *log*;
d) mensagem_alerta: identifica a mensagem que foi gerada ao administrador da rede no momento da detecção;
e) classtype: mostra a classificação da invasão detectada;
f) invasão_confirmada: mostra se o administrador confirmou ou negou a suspeita de ataque;
g) regra: identifica o número da regra que gerou o evento;
h) revisão: mostra a versão da regra que gerou o evento;
i) confirmação_automatica: mostra se a resposta dada á suspeita foi informada pelo administrador ou pelo próprio sistema.

6.3.2 Modelagem do Aplicativo

Após ser finalizada a montagem do banco de dados, iniciou-se a modelagem da

aplicação, nomeada Dimitry. Esta modelagem foi feita com a preocupação de manter o sistema com uma estrutura de fácil entendimento e manipulação.

O sistema foi dividido em seis *Packages* onde cada uma delas possui um conjunto de classes. no Quadro 4 tem-se a divisão, ilustrando as *package* e suas respectivas classes.

Packages	Classes
Analise	Captura.java
	Pacotes.java
BancoDados	Conexao.java
	Trafego.java
	TrafegoNotificado.java
Rules	Rules.java
	RulesList.java
Arquivos	AbrirArquivo.java
Openrules	OpenRules.java
IDS_Desktop	IDS_DesktopView.java
	IDS.java
	IDS_Trafego.java
	IDS_Trafego_Notificado.java

Quadro 4. Packages e classes do Dimitry tradicional

A *package* "Analise" é uma das mais importante do Dimitry pois é nela que são realizadas as ações sobre os pacotes propriamente ditos, ou seja, são nelas que ocorrem as capturas dos mesmos e suas respectivas análises, cruzando as assinaturas dos pacotes com as do sistema. A captura das informações é realizada pela classe "Captura.java" e a análise é feita pela classe "Pacotes.java".

As classes existentes no "BancoDados" são responsáveis por todas as ações que dizem respeito a comunicação com banco de dados, onde nesta etapa do projeto diz respeito apenas a inserção de *logs*. A classe "Conexao.java" é responsável por conectar-se com o banco e fazer a ponte entre o mesmo o objeto solicitante. Já "Trafego.java" é utilizada para realizar qualquer ação sobre a tabela "trafego" existente no banco de dados, utilizando um objeto da "Conexão.java" para realizar suas ações. A classe "TrafegoNotificado.java" trabalha da mesma maneira que a "Trafego.java", porém ao invés de realizar ações sobre a tabela

“trafego”, realiza as ações sobre a “trafegonotificado”, ou seja, salva os eventos gerados pela detecção de uma invasão.

Outra *package* bastante importante no sistema é a “Rules”, pois é nela que encontram-se as classes onde as regras carregadas pelo sistema são armazenadas. A classe “Rules.java” é a que recebe os atributos das regras onde cada um deles é implantado em uma variável específica da classe, e cada objeto da mesma corresponde a uma regra. Já a “RulesList.java”, como o próprio nome diz, é a classe que possui a lista com todas as regras, ou seja, os objetos.

Como as regras são criadas no formato de texto e salvas em um diretório específico do sistema operacional, o Dimitry necessitou de classes que conseguissem capturar estes arquivos, interpretá-los e importá-los, para realizar estas funções foram criadas duas classe divididas em duas *packages*, “Openrules” e “Arquivos”. A classe da primeira *package* é responsável por abrir todos os arquivos do diretório “C:\rules” que possuam a extensão “.rules” e caminhar linha por linha de cada arquivo, visto que cada linha corresponda a uma regra, passando estas para o interpretador do sistema. Já a classe da segunda *package* é responsável por interpretar as regras, ou seja, é nela que encontra-se o interpretador.

A *package* “IDS_Desktop” é onde encontra-se as classes que fazem a interação com o usuário, ou seja, nela está localizado as interfaces do Dimitry. Na classe *IDS_DesktopView* verifica-se a tela principal do sistema, possuindo apenas um menu de opções do mesmo. Já a classe *IDS* tem-se desenvolvida a janela onde a análise é feita. As classe *IDS_Trafego* e *IDS_Trafego_Notificado*, foram criadas com a intenção de realizar consultas dos *logs* gravados no banco, porém pela indisponibilidade do tempo, não foi possível concluí-las.

Após o conhecimento da estrutura do sistema, podemos verificar mais detalhadamente o interior de algumas das classes mais importantes.

Uma destas classes mais é a *captura,* é nela que a busca pelo pacote é inicializada, onde recebe os atributos e pelo procedimento *capturar* inicia tal operação, a estrutura desta classe pode ser melhor analisada na Figura 24.

Figura 24. Estrutura da classe *Captura*

Na Figura 25 podemos verificar o desenvolvimento do procedimento *Capturar(),* onde são capturados os pacotes decorrentes dos protocolos TPC e UDP, e posteriormente passada para uma outra classe, onde são realizas as análises necessárias. Vale lembrar que esta classe foi desenvolvida á partir de uma classe pronta disponível na Internet encontrada no site (http://wmagician.wordpress.com/2009/02/11/packet-capture-em-java-com-jpcap/).

```
public void capturar() throws IOException, ParseException, InterruptedException{

    //Simples contador.
    int i = 0;

     jpcap.NetworkInterface[] interfaces = JpcapCaptor.getDeviceList();

    //Filtra os pacotes que serão capturados
    Packet p = null;

    captor = JpcapCaptor.openDevice(interfaces[ this.interfase], 65535, false, 20);

    captor.setFilter("tcp or udp", true);

    //Cliclo para capturar Infinito, somente para com o comando stop() do botão;
    while (i < i+1) {
        //Captura um pacote.
        p = captor.getPacket();

        Pacotes pacote = new Pacotes(ids, p, regras, home_net, home_net_port, http_servers, http_port,
                                     dns_servers, smtp_servers);
        if (pacote.getAtaque() == true) {
            ids.insere_log("Aconteceu uma Invasão");
        }
        i++;
    }

    //Fecha a captura de pacotes.
    captor.close();
}
```

Figura 25. Comando do método *capturar* da classe *Captura*

O método descrito na Figura 25, captura o pacote e o passa por parâmetro para outra classe juntamente com as configurações do ambiente de rede que o IDS pertence, na Figura 26 podemos verificar as variáveis e os métodos do objeto *Pacote* utilizado na Figura referida anteriormente.

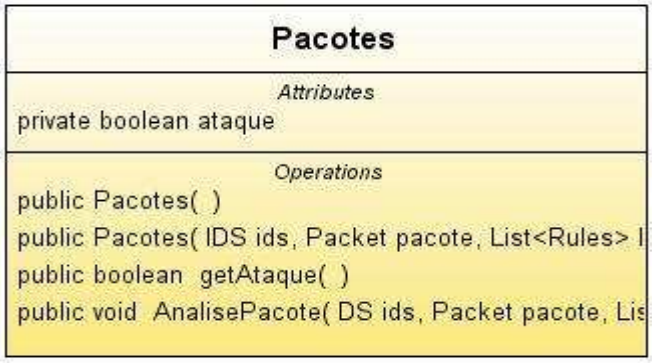

Figura 26. Estrutura da classe *Pacotes*

A classe apresentada possui apenas um atributo, onde este é utilizado para identificar se ocorreu ou não um ataque, no método *AnalisePacotes* é feita a análise do pacote recebido por parâmetro, comparando sua assinatura com as assinaturas das regras carregadas na classe *Rules.*

Por fim, outra classe que podemos citar como importante na verificação de uma intrusão é a classe *Rules* é nela que estão guardadas todas as informações das regra, na Figura 27 podemos visualizar os componentes desta classe.

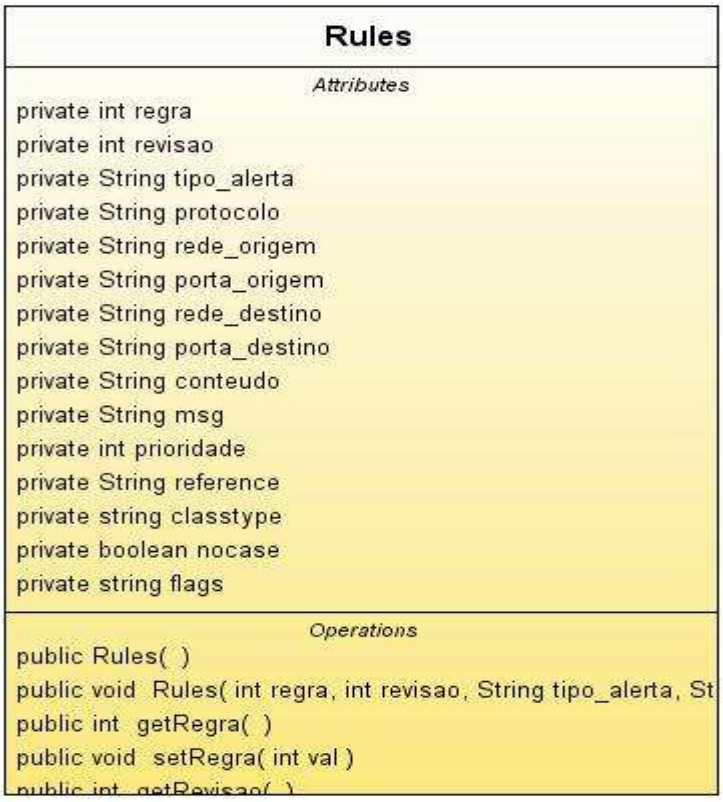

Figura 27. Atributos da classe *Rules*

Após termos conhecimento de toda a estrutura da aplicação, e suas principais classes podemos verificar de que forma foi desenvolvido o aplicativo final e como o mesmo interage com o usuário.

6.3.3 Desenvolvimento do Aplicativo

O desenvolvimento da interface do aplicativo foi realizada por meio do componente Java Desktop Application, oferecido pela ferramenta NetBeans 6.1, este componente por sua vez a construção de aplicações gráficas, ou seja, permite a criação de menus, botões, tabelas entre outros, possibilitam, por meio destes recursos, uma fácil interação entre sistema e usuário.

O Dimitry foi dividido em duas partes principais, ou seja, em duas telas, a primeira é a tela principal e a segunda é a tela de análise dos pacotes.

A tela principal possui somente um menu. Esta tela foi criada com a intenção de possibilitar mais funcionalidades ao sistema, permitindo adicionar um novo item no menu para cada nova funcionalidade. A princípio este menu possui somente uma, que é a análise de pacotes por meio do IDS, como podemos verificar na Figura 28.

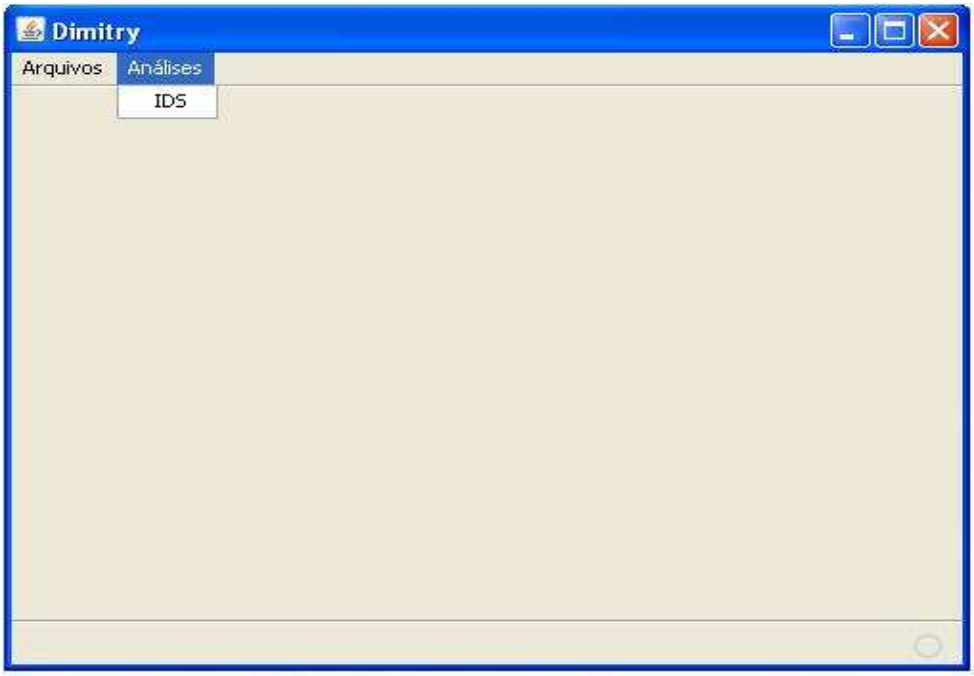

Figura 28. Tela Principal do Dimitry

Entrando na opção IDS do menu irá ser chamada a principal tela do sistema

demonstrada na Figura 29. Nesta tela é iniciada a análise dos pacotes, para que isto ocorra faz-se necessário primeiramente informar os dados da rede, nos campos existentes dentro do quadro “Parâmetros da Análise”, no campo “Interfaces Disponíveis” são listadas todas as interfaces existentes no ambiente em que o Dimitry está inserido, cabendo então ao usuário selecionar a interface desejada e então iniciar a análise clicando no botão “Iniciar Análise”

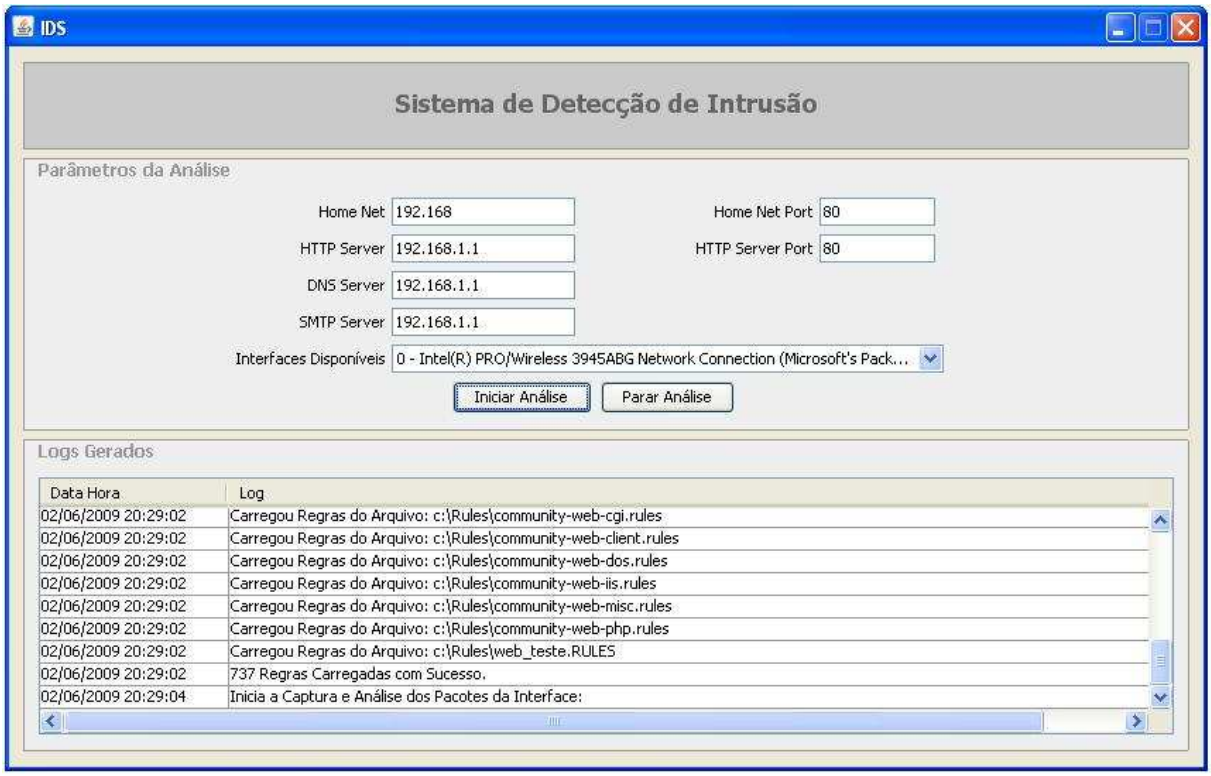

Figura 29. Tela da análise dos pacotes do Dimitry

Após o início da análise o sistema irá ficar observando a rede e salvando no banco de dados do mesmo as informações dos pacotes que circulam na mesma. Também fica à procura de pacotes que venham a conter assinaturas existentes em uma das regras carregadas, quando um destes for encontrado o administrador receberá um aviso na tela, como na Figura 30, mostrando as informações da regra que o capturou e os dados provenientes do mesmo.

Figura 30. Aviso de suspeita de invasão do Dimitry

Após a resposta do administrador um *log* é gerado na tela do IDS, além de ser salvo no banco de dados a ocorrência suspeita com sua respectiva resposta, para que esta possa ser consultada futuramente de forma manual pelo usuário, podendo ajudá-lo a decidir sobre ataques futuros.

Definindo as funções desempenhadas pelo usuário do sistema, que neste caso é o administrador da rede, foi elaborado um diagrama de casos de uso, que encontra-se demonstrado na Figura 31, que mostra tais funções.

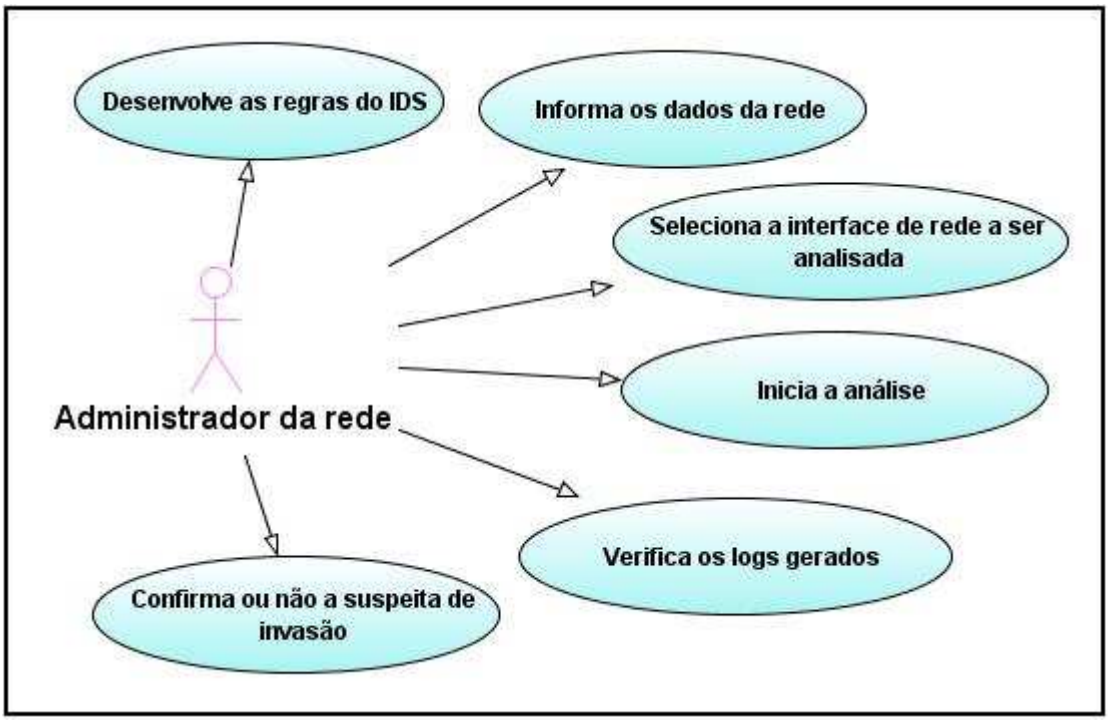

Figura 31. Diagrama de casos de uso do Dimitry

Outra maneira de entender o funcionamento do sistema é montar um fluxograma geral do mesmo, mostrando as possíveis ações que ele possa tomar. Na Figura 32 este fluxograma pode ser observado, onde o mesmo tem seu inicio com a análise, assim posteriormente são carregadas as regras, logo após começa-se então a se captur os pacotes. Para cada pacote capturado é salvo um *log* de tráfego e posteriormente analisado sua assinatura. Caso esta venha a acusar uma suspeita de invasão, um alerta é gerado solicitando a análise do administrador da rede, salvando por fim o *log* da suspeita com a resposta do mesmo. Caso não possua assinatura suspeita, o pacote trafega normalmente sem gerar nenhum alerta.

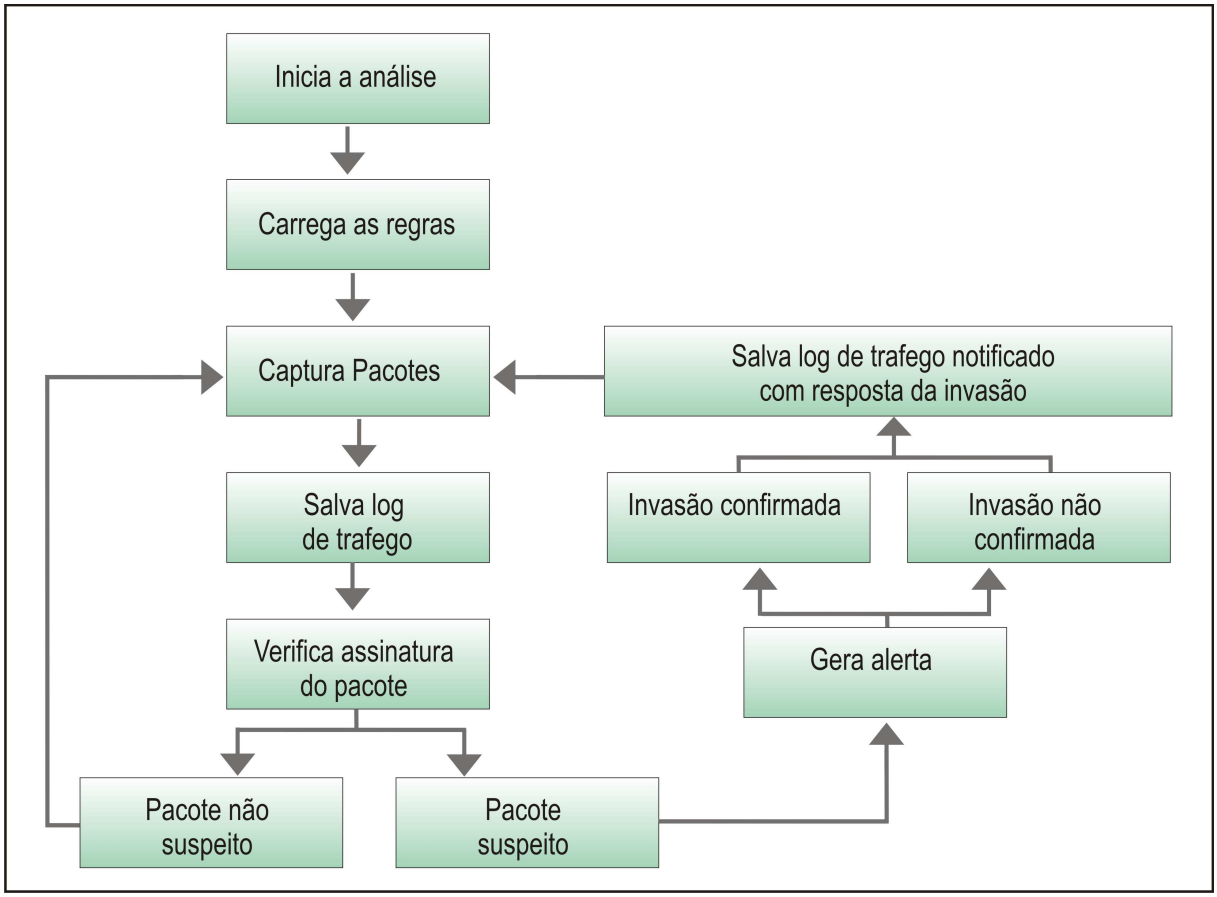

Figura 32. Fluxo de ações do Dimitry

O sistema desenvolvido nesta etapa do projeto, não possui nenhum outro artifício para analisar as informações dos pacotes a não ser as regras utilizadas pelo IDS, podemos observar também, que o sistema não utiliza os *logs* de tráfego e de suspeita de invasão para realizar as análises.

Para melhorar o desempenho do Dimitry, optou-se por utilizar técnicas de agentes inteligentes, utilizando para tanto os dados de seus próprios *logs*, o desenvolvimento desta melhoria é demonstrado na segunda etapa do projeto, a qual é descrita no próximo subtítulo.

6.4 IMPLEMENTAÇÃO DO APLICATIVO IDS INTELIGENTE

Um sistema de detecção de intrusão convencional, como o desenvolvido na etapa

anterior não é capaz de realizar análises em seus registros anteriores, ou seja, não possui nenhum artifício que possa fazer com que sua análise possa ser melhorada com o passar do tempo.

Suprindo-se esta defasagem sugeriu-se desenvolver um agente inteligente que será descrito nos próximos itens.

6.4.1 Modelagem do Banco de Dados de Logs do IDS Inteligente

A princípio, para suprir as necessidades do sistema inteligente, foi desenvolvida uma nova estrutura de banco de dados, similar a do Dimitry tradicional, porém foi atribuída a ela uma nova tabela chamada de *ListaNegra*, onde serão salvos os *IP* nos quais foram identificados como uma ameaça que está sendo protegida.

Esta nova tabela possui um campo em especial chamado de *ip_origem*, como podemos verificar na Figura 33, que se refere ao número do *IP* invasor. Este campo será analisado pelo sistema inteligente no momento da análise do pacote. Os outros atributos pertencentes á tabela são os dados da análise que o levaram o *IP* a ser intitulado como uma ameaça.

As tabelas que já existiam no banco do sistema tradicional funcionam da mesma maneira, uma utilizada para armazenar todos os registros de tráfegos, a outra para guardar todos os eventos de tráfego que geraram uma suspeita, como já explicado anteriormente.

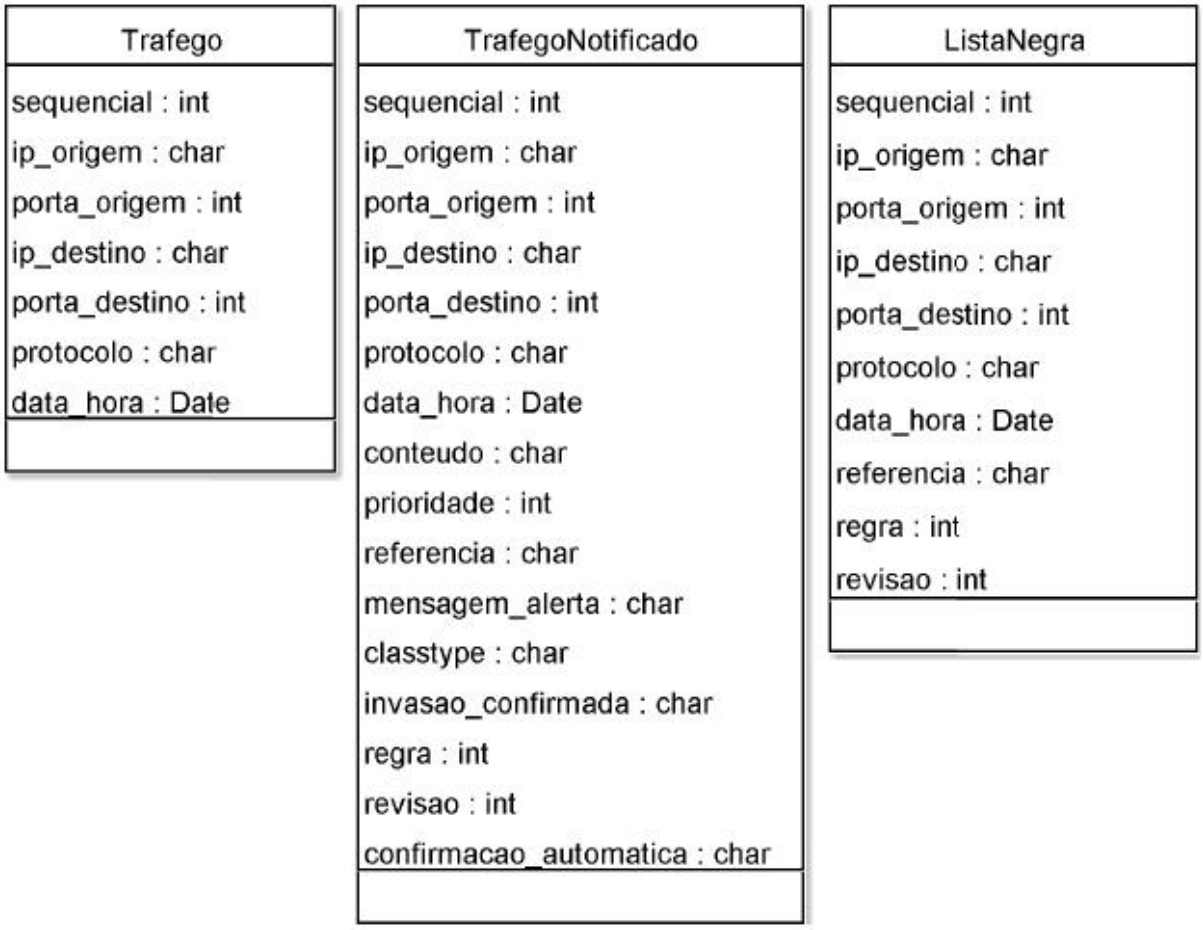

Figura 33. Estrutura do banco de dados Dimitry inteligente

Assim, para que o sistema possa utilizar os dados armazenados nesta nova estrutura do banco de dados é necessária a modelagem de um método inteligente que faça as relações entres os registros para que os mesmos possam auxiliar na análise dos pacotes.

6.4.2 Modelagem do Agente

Na modelagem do agente inteligente escolheu-se os fatores de certeza para agregar inteligência em sua concepção, que tem por objetivo auxiliar no processo de tomada de decisão em sistemas com incertezas associadas. Tal modelo foi escolhido em virtude de ser bastante utilizado em sistemas que possuem graus de incertezas consideráveis que é o caso do

sistema desenvolvido que possui um pequeno conjunto de regras necessitando desta forma de tal técnica para o tratamento das incertezas encontradas.

Com o objetivo de um melhor entendimento da modelagem do agente, a Figura 34 apresenta um esquema com a modelagem. Podemos observar por meio desta que o agente possui um módulo de percepção que é ativado quando uma regra do IDS é ativada, após isto o agente realiza as buscas das informações do pacote suspeito encontrado. Estas informações são então enviadas para a base de regras do agente onde serão tratadas e geram cinco hipóteses onde serão analisadas pelo FC. O resultado final desta análise será um valor entre o intervalo de -1 a 1, onde dependendo do valor o sistema identificará o tipo de ação que tomará. Esta ações podem ocorrer de três formas distintas, a suspeita de invasão é confirmada automaticamente, a suspeita de invasão é anulada automaticamente ou o sistema gera um alerta quando o grau de certeza para resposta automática não é atingido.

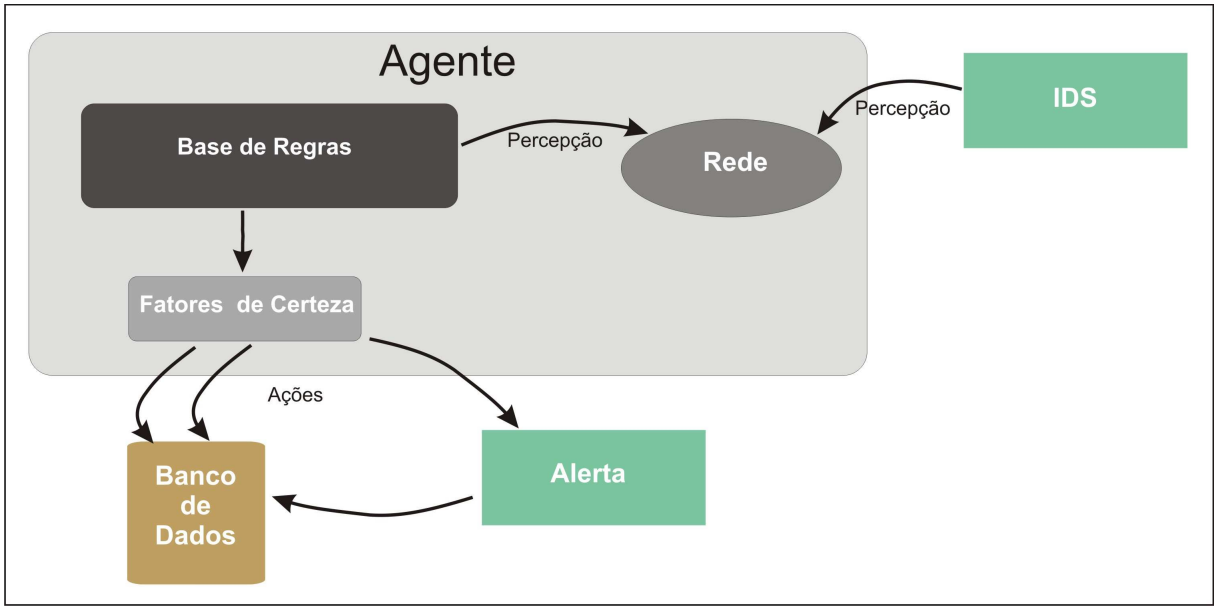

Figura 34. Esquema de representação do agente

Como podemos observar o agente utilizado é o reativo simples e a sua modelagem se resume em basicamente dois módulos, na base de regras onde encontram-se as evidências e

hipóteses do sistema e o FC que realiza o cálculo para minimizar as incertezas das hipóteses geradas.

6.4.3 Utilização da Teoria dos FC no Agente Inteligente

Os fatores de certeza vem sendo muito utilizados para auxiliar a solução de problemas que envolvam a questão de confiança da informação. Este foi utilizado pois é um método clássico para a modelagem da incerteza possuindo uma boa eficiência em sistemas que possuam elevado grau de incerteza em suas hipóteses, problema encontrado no sistema desenvolvido nesta pesquisa.

6.4.3.1 DEFINIÇÃO DAS REGRAS

As regras foram elaboradas a partir de cinco cenários baseados em algumas das possíveis maneiras de detectar ataques sem utilização das assinaturas do IDS, onde cada um destes possui um conjunto de regras e uma variável final. Estes podem ativar somente uma regra que possui o fator de confiança que será atribuído à variável final e este valor por sua vez, representa a confiança de que a regra que ativou encontrou uma invasão. Os valores utilizados nas regras são gerados por meio de cruzamento de informações existentes no banco de dados, como o registro de tráfego, registros de análises anteriores, os dados da lista negra e os dados da regra do IDS.

O primeiro cenário é o da *lista negra*, é o mais simples pois somente verifica se existe ou não um determinado *IP* na lista, a variável final deste cenário é a *lista_negra_fc* que receberá o fator de confiança da regra a ser ativada. O modelo deste cenário pode ser

observado no Apêndice A e na Figura 35, onde é consultado se um determinado *IP* está ou não inserido na lista negra. Caso a consulta seja verdadeira, a variável final do cenário recebe um fator de confiança de 0.75 que representa o quanto é confiável que esta regra tenha encontrado uma invasão. Caso a consulta à *lista negra* retorne falso, a variável final recebe um FC no valor de 0.10, ou sejam, esta regra não é tão confiável.

```
if (ln.consultaip(ip)){
    atributo.setInvasao_lista_negra_fc(0.75f);
}
else{
    atributo.setInvasao_lista_negra_fc(0.10f);
}
```

Figura 35. Regras do cenário da lista negra

Outro cenário utilizado é o padrão de acesso, este por sua vez realiza uma consulta nos registros do banco de dados em busca do padrão de acesso do pacote. Primeiramente, é realizada a média ponderada dos acessos feitos pelo suspeito, retornando o horário médio de acesso, posteriormente é verificada qual a média de acesso do pacote analisado no horário do tráfego e por fim é feita uma média entre estas duas médias para chegar no percentual de igualdade, onde este é analisado nas regras do cenário gerando então um FC para a variável final que é chamada de *padrao_trafego_fc*.

Registros anteriores referentes às análises e conclusões realizadas nos pacotes também são usados no terceiro cenário, as regras deste verificam o percentual de invasões confirmadas, ou seja, é verificada a quantidade de alertas gerados por um determinado pacote e posteriormente a quantidade de invasões confirmadas dentro destes alertas, formando desta maneira um percentual de invasão. Este percentual então é tratado pelas regras que irão "alimentar" a variável *analises_anteriores_fc* que é a resposta deste cenário.

O quarto cenário representa uma verificação bastante simples, pois nele é tratado

o grau de prioridade da regra ativada pelo IDS. Neste é utilizado o número da prioridade da regra, e verificado o intervalo que ele pertence, passando então em uma das regras do agente inteligente. No Apêndice A e na Figura 36 verifica-se as regras deste cenário que têm como resultado final a variável chamada de *prioridade_regra_fc*.

```
if (prioridade_regra == 0){
    atributo.setInvasao_prioridade_regra_fc(0.1f);
}
else if (prioridade_regra >= 1 && prioridade_regra <= 6){
     atributo.setInvasao_prioridade_regra_fc(0.6f);
 }
 if (prioridade_regra > 6 && prioridade_regra <= 13){
     atributo.setInvasao_prioridade_regra_fc(0.3f);
 }
 else{
     atributo.setInvasao_prioridade_regra_fc(0.2f);
 }
```

Figura 36. Regras do cenário da prioridade da regra do IDS

Por fim, o último cenário utilizado é o que busca o tempo médio de envio dos pacotes de um determinado *IP* nos últimos cinco segundos. Este cenário foi desenvolvido com o intuito de encontrar possíveis tentativas de estouros de *buffer*. A média gerada por este cenário é analisada pelas regras, e gerado o FC da variável final *media_tempo_trafego_fc.*

6.4.3.2 MODELAGEM MATEMÁTICA DOS FATORES DE CERTEZA

Buscando compreender melhor o funcionamento do cálculo do fator de certeza, a forma mais clara é por meio da realização de exemplos. A seguir iremos verificar passa a passo este cálculo, desde a geração dos fatores de confiança das regras até os resultados finais.

Com o intuito de verificar os prováveis ataques, o sistema define previamente os seus objetivos (invasão, não invasão) que em um primeiro momento são desacreditados,

recebendo então valor zero, ou seja, como não foi definida nenhuma evidência que validada-se estes objetivos, sua crença (P(H)) é zero conforme Quadro 5.

Objetivo	P(H)
invasao	0,00

Quadro 5. Probabilidades a priori dos ataques

Com o intuito de demonstrar tal exemplo definiu-se a seguinte situação: o sistema de detecção de intrusão inteligente capturou um pacote que ativou uma regra do IDS que possui grau de prioridade igual a 15, ativando então o módulo de inteligência onde por sua vez realizou consultas na base de dados do sistema buscando algumas informações sobre o pacote suspeito.

Estas consultas feitas sobre o IP que originou o pacote conseguiram as seguintes informações: o mesmo encontra-se na *lista negra*, a média de igualdade entre o tráfego atual e o padrão de tráfego é de 65%, o número registros em relação ao tráfego notificado é menor que cinco e a média de segundos entre os pacotes nos últimos cinco segundos é de 0.01 segundos. Por meio destas informações o agente acusou cinco regras como podemos verificar na Figura 37.

1- *SE* possui_na_lista_negra *ENTAO* invasao (FC = 0.75)
4- *SE* 25% < media_padrao_trafego *ENTAO* invasao (FC = 0.2)
5- *SE* numeros_trafego_notificado <= 5 *ENTAO* invasao (FC = 0.2)
12- *SE* prioridade_regra > 13 *ENTAO* invasao (FC = 0.20)
13- *SE* media_segundos_entre_pacotes <= 0.01 *ENTAO* invasao (FC = 0.60)

Figura 37. Regras ativadas

O conjunto de regras apresentam o domínio do sistema especialista em relação ao cada evento gerado, onde cada conhecimento possui um grau de confiabilidade representado pelo FC. A partir de cada evidência que é confirmada o conseqüente (ENTÃO) é ativado

resultando em um objetivo do sistema, ou seja, em uma resposta. Assim, a partir das entradas e saídas geradas pelo sistema, o cálculo do fator de certeza é realizado chegando desta forma em um resultado.

Tendo em mente os resultados da consulta ao banco realizada pelo sistema, verifica-se que o IP de origem consta na lista negra, por meio desta informação o agente ativou a regra 1 onde sua conseqüência é um dos objetivos do sistema (invasão). Assim, a probabilidade de invasão representada por *p(H)* é comparada com a probabilidade da hipótese de ter invasão, dada a evidência de a possuir na lista negra representada por *p(H|E).* Nota-se que houve um aumento na crença de zero, valor inicial, para 0,75, FC da regra que atingiu o objetivo.

Com o aumento da crença na hipótese invasão representada pela regra 1 da Figura 36 calcula-se de acordo com a fórmula 2 a nova MC. Tendo que a probabilidade de invasão *p(H)* é definida inicialmente por 0 e a probabilidade de invasão dado que possui na lista negra *p(H|E)* é de 0,75 conforme FC da regra 1.

$$MC = \frac{P(H \mid E) - P(H)}{1 - P(H)} = \frac{p(Invasao \mid ListaNegra) - p(Invasao)}{1 - p(Invasao)} = \frac{0{,}75 - 0}{1 - 0} = \frac{0{,}75}{1} = 0{,}75$$

A partir do aumento da crença de invasão para 0,75, calcula-se o novo FC de acordo com a fórmula 6.

$$FC = \frac{MC_{Invasao} - MD_{Invasao}}{1 - \min[MC_{Invasao}, MD_{Invasao}]} = \frac{0{,}75 - 0}{1 - 0} = 0{,}75$$

Desta forma, chega-se ao novo FC da invasão considerando a MC na hipótese de invasão confirmada pela evidência que possui na lista negra, e a MD da mesma hipótese inicial.

Também foi possível verificar por meio da consulta no banco uma segunda

evidência que é o padrão de acesso do IP que originou o pacote. Esta consulta demonstrou que o pacote está trafegando com um grau de igualdade de 65% em relação ao seu padrão de acesso geral, ativando desta forma a regra de número 5 (Figura 37), onde a mesma tem por hipótese o mesmo objetivo do sistema (invasão).

Neste caso a probabilidade de não invasão representada por *p(H)* é comparada com a probabilidade da hipótese de ter não invasão dada a evidencia média padrão de tráfego representada por *p(H|E).* Nota-se que também houve uma diminuição na crença de 0,75 para 0,20.

$$MD = \frac{P(H) - P(H \mid E)}{P(H)} = \frac{p(Invasao) - p(Invasao \mid MediaPadraoTrafego)}{p(Invasao)} = \frac{0,75 - 0,2}{0,75} = \frac{0,55}{0,75} = 0,73333$$

A partir da diminuição da crença da invasão de 0,75 para 0,20 calcula-se o novo FC também de acordo com a fórmula 6.

$$FC = \frac{MC_{NaoInvasao} - MD_{NaoInvasao}}{1 - \min[MC_{NaoInvasao}, MD_{NaoInvasao}]} = \frac{0,75 - 0,7333}{1 - 0,7333} = 0,0626$$

Desta forma chega-se também ao novo FC de não considerar a MC na hipótese de não invasão confirmada pela evidencia média do padrão de tráfego, e a MD da mesma hipótese inicial.

A terceira evidencia gerada pela consulta é a da média de ataques confirmados, porém o número de tráfego notificado é menor que 5 ativando assim a regra de número 5 do agente, onde tem por hipótese o objetivo invasão e possui o FC de 0,20. Como podemos verificar anteriormente a hipótese de ser uma invasão *p(H)* passou de 0,75 para 0,0626 devido a evidencia do padrão de acesso. Esta por sua vez é comparada com a hipótese de invasão dado a média de ataques confirmados *p(H|E)* que é de 0,2, demonstrando desta forma um novo aumento na crença da invasão.

Nesta situação calcula-se primeiramente o novo MC levando em conta a nova evidencia gerada por meio da fórmula 2.

$$MC = \frac{P(H \mid E) - P(H)}{1 - P(H)} = \frac{p(Invasao \mid MediaAtaquesConfirmados) - p(Invasao)}{1 - p(Invasao)} = \frac{0{,}20 - 0{,}0626}{1 - 0{,}0626} = \frac{0{,}1374}{0{,}9374} = 0{,}1466$$

A partir do novo MC gerado é calculado o novo FC da hipótese de invasão utilizando a formula 6.

$$FC = \frac{MC_{Invasao} - MD_{Invasao}}{1 - \min[MC_{Invasao}, MD_{Invasao}]} = \frac{0{,}1466 - 0{,}7333}{1 - 0{,}1466} = -0{,}6875$$

A regra do sistema de detecção de intrusão que encontrou a suspeita possui um atributo que indica a prioridade da mesma onde também é utilizada pelo módulo do agente para identificar um ataque, este atributo compõe a quarta evidencia gerada neste exemplo e possui um grau de prioridade igual a 15. Por meio deste grau o agente inteligente ativou a regra 12. Esta regra por sua vez ativa a hipótese de não invasão com o FC de 0,2, constatando desta forma um novo aumento na crença da invasão, pois o *p(H)* da mesma foi anteriormente definido -0,6875 pela regra 6. Desta forma, calcula-se a nova MC da hipótese de invasão a partir da fórmula 4.

$$MC = \frac{P(H \mid E) - P(H)}{1 - P(H)} = \frac{p(Invasao \mid \Pr ioridade\,\mathrm{Re}\,gra) - p(Invasao)}{1 - p(Invasao)} = \frac{0{,}20 - (-0{,}6875)}{1 - (-0{,}6875)} = \frac{0{,}8875}{1{,}6875} = 0{,}5259$$

Como houve duas evidências que aumentaram a crença de uma mesma hipótese o grau de crença delas é combinado gerando uma nova MC conforme fórmula 3.

$$MC = MC[H, E_1] + MC[H, E_2] * (1 - P[H, E_1]) = MC[Invasao, MediaAtaquesConfirmados] + MC[Invasao, \Pr ioridade\,\mathrm{Re}\,gra] * (1 - p[Invasao, MediaAtaquesConfirmados]) = 0{,}1466 + 0{,}5259 * (1 - 0{,}1466) = 0{,}5954$$

Após o cálculo da nova MC, calcula-se o novo FC a partir da hipótese de ser invasão a partir da fórmula 6.

$$FC = \frac{MC_{NaoInvasao} - MD_{NaoInvasao}}{1 - \min[MC_{NaoInvasao}, MD_{NaoInvasao}]} = \frac{0{,}5954 - 0{,}7333}{1 - 0{,}5954} = \frac{-0{,}1379}{0{,}4046} = -0{,}3408$$

Por fim, o sistema realiza o calculo da quinta evidencia gerada que é a média do tempo entre os pacotes nos últimos cinco segundos. Esta média conforme informada no exemplo é de 0,01 segundos ativando desta forma a regra 13 do sistema que tem o FC de 0,60 para a hipótese de invasão. Como a hipótese de invasão *p(H)* passou a ser de -0,3408 (gerada pela regra 12), e a nova hipótese de invasão dado a média de segundos entre pacotes *p(H|E)* é de 0,60, confirmou-se um novo aumento da crença de invasão calculando-se desta forma a nova MC a partir da fórmula 2.

$$MC = \frac{P(H \mid E) - P(H)}{1 - P(H)} = \frac{p(Invasao \mid MediaSegundosPa\cot es) - p(Invasao)}{1 - p(Invasao)} = \frac{0{,}6 - (-0{,}3408)}{1 - (-0{,}3408)} = 0{,}7016$$

Como houve novamente duas evidências que aumentaram a crença de uma mesma hipótese o grau de crença delas é combinado gerando uma nova MC conforme fórmula 3.

$$MC = MC[H, E_1] + MC[H, E_2] * (1 - P[H, E_1]) = MC[Invasao, \Pr ioridade \operatorname{Re} gra] + MC[Invasao, MediaSegundosPa\cot es] * (1 - p[Invasao, \Pr ioridade \operatorname{Re} gra]) = 0{,}5954 + 0{,}7016 * (1 - 0{,}5954) = 0{,}8792$$

Com a nova MC de invasão calculada, calcula-se também o novo FC desta hipótese a partir da fórmula 6.

$$FC = \frac{MC_{NaoInvasao} - MD_{NaoInvasao}}{1 - \min[MC_{NaoInvasao}, MD_{NaoInvasao}]} = \frac{0{,}8792 - 0{,}7333}{1 - 0{,}7333} = \frac{0{,}1459}{0{,}2667} = 0{,}5470$$

O resultado final do sistema é o último FC gerado representando o fator de certeza da invasão. O resultado final desta análise é a geração de um alerta ao administrador, pois o

valor do FC da invasão é menor que o índice do grau de confiança definido para que o sistema possa gerar respostas automáticas que é de 0,75, caso fosse maior a resposta á invasão seria gerada de forma automática.

6.4.3.3 DEFINIÇÃO DAS HIPÓTESES INICIAIS

Neste sistema todas as regras geram apenas uma hipótese que é a de invasão, a única diferença entre as hipóteses é o grau FC que as mesmas possuem e são estes que irão determinar a resposta do agente.

Como podemos observar as análises feitas pelo agente inteligente gera sempre cinco hipóteses iguais, porém com fatores de confiança distintos onde cada uma delas é gerada por um cenário diferente, que são relacionadas para gerarem uma hipótese final que é chamada de *fc_invasao*.

A variável final é que possui o fator de certeza se um determinado pacote é visto como uma invasão, ou seja, ela recebe o calculo FC das cinco variáveis adquiridas por meio dos cenários. Caso o resultado deste cálculo seja maior ou igual a 0.75 a confirmação de invasão é gerada automaticamente, se este resultado seja menor ou igual a -0.75 a não confirmação da invasão também é feita de forma automática. Por fim, os resultados fora destes intervalos gerarão um alerta mostrando as informações do pacote e da regra ativada pelo IDS, além do resultado do fator de certeza do mesmo, para que o administrador decida sobre a resposta a ser dada ao pacote.

6.4.3 Modelagem da Aplicação

O sistema inteligente, para suportar análises de pacotes por meio de técnicas inteligentes, necessitou de uma nova modelagem, por este motivo desenvolveu-se uma nova aplicação baseada no Dimitry tradicional.

De um ponto de vista geral, a modelagem do Dimitry inteligente é bastante semelhante á modelagem do mesmo na sua versão tradicional, as diferenças encontram-se apenas na inclusão de um novo conjunto de classes desenvolvidas dentro da *packge* chamada de *AgenteInteligente*, vista no Quadro 5, e na alteração do pacote *Pacotes.java*.

Packages	Classes
Analise	Captura.java
	Pacotes.java
BancoDados	Conexao.java
	Trafego.java
	TrafegoNotificado.java
Rules	Rules.java
	RulesList.java
Arquivos	AbrirArquivo.java
Openrules	OpenRules.java
IDS_Desktop	IDS_DesktopView.java
	IDS.java
	IDS_Trafego.java
	IDS_Trafego_Notificado.java
AgenteInteligente	Atributos.java
	CalculoFC.java
	Regras.java

Quadro 6. Packages e classes do Dimitry Inteligente

A classe *Pacotes.java*, é a única classe modificada nesta nova etapa do processo, como já comentado anteriormente, esta sofreu tal mudança pois é nela que são realizadas as análises dos pacotes coletados. Esta alteração consistiu apenas na inserção da chamada do agente inteligente no momento da ativação de uma regra do IDS, ou seja, o mecanismo inteligente é chamado antes que o sistema alerte o administrador, como era feito no sistema tradicional.

Nesta nova *package* denominada de *AgenteInteligente* encontram-se todos os

pacotes que fazem parte do módulo inteligente da ferramenta. Como podemos observar são três classes que compõem o conjunto, onde cada uma possui funções específicas no processo de análise do sistema.

Na primeira classe deste conjunto citua-se a *Atributos.java* nela encontram-se as cinco variáveis responsáveis por guardarem os fatores de confiança de cada cenário como foi explicado anteriormente, estas variáveis podem ser vistas na Figura 38.

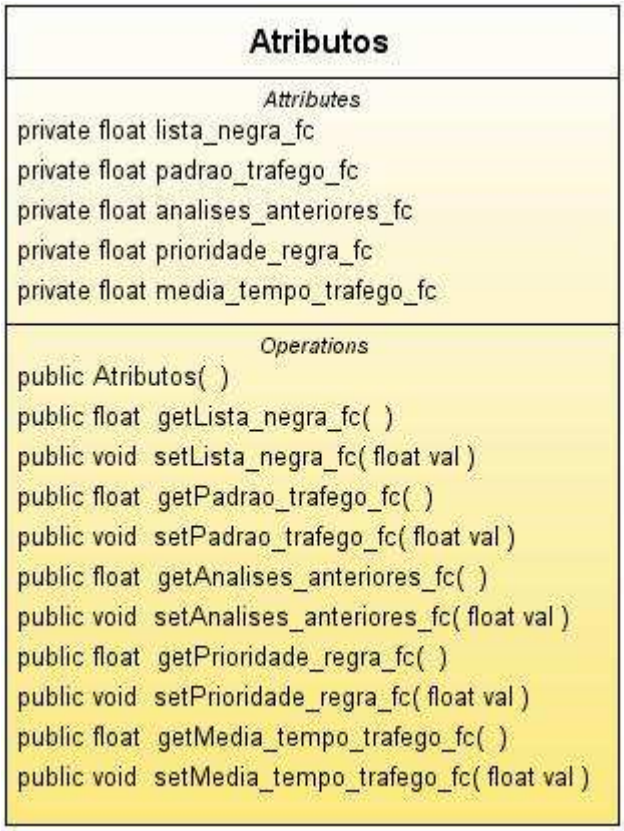

Figura 38. Estrutura da classe *Atributos*

É por meio destas cinco variáveis que o sistema chega a uma resposta, onde pode por meio desta interpretar uma invasão, um tráfego normal, ou definir incerteza sobre a resposta solicitando então a análise do usuário do sistema. Dentro da classe *Regras.java*. é que são encontradas as regras do módulo inteligente onde estas são divididas em cinco grupos ou cenários. Cada um destes cenários possuem um conjunto de regras e cada regra possui um FC. Toda análise gera cinco variáveis, uma para cada cenário, gerando também cinco fatores de confiança, constituindo assim a estrutura adequada para o calculo do fator de confiança e

gerando o resultado final.

Com as variáveis já carregadas e guardadas, falta somente a realização do cálculo para geração do resultado final da análise do módulo inteligente. Esta função é realizada pela classe *CalculoFC.java*, onde através das técnicas dos Fatores de Certeza FC, são realizados os cálculos e gerando e gerando por fim um resultado dentro do intervalo -1 à 1, tendo que -1 é a certeza de que o pacote analisado não é uma invasão e que 1 é a certeza da invasão. O atributo desta classe que representa o resultado final e os métodos que a mesma possui, podem ser vistos na Figura 39.

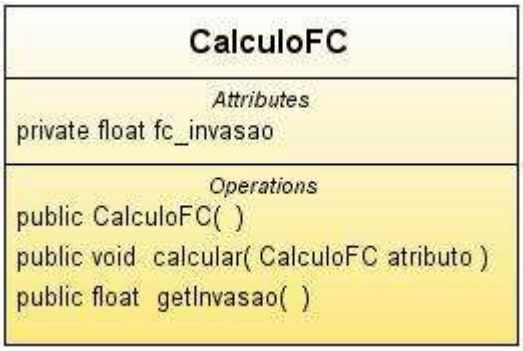

Figura 39. Estrutura da classe *CalculoFC*

Após todo o cálculo e resultado final gerado, o sistema verifica então se é possível ou não confirmar a suspeita de invasão. Os procedimentos desta análise e as ações que o usuário deve tomar são melhores vistas a seguir.

6.4.4 Desenvolvimento da Aplicação

O desenvolvimento do Dimitry inteligente foi realizado no mesmo ambiente do sistema tradicional por se tratar do mesmo sistema, porém com algumas melhorias como a incorporação de um sistema inteligente.

A interface deste novo sistema também é bastante semelhante, onde possui duas telas principais, uma é a tela inicial do sistema, onde encontra-se apenas um menu que serve

para chamar as funcionalidades atuais do sistema, que neste é a detecção de intrusão inteligente. Na Figura 40 encontramos a imagem da tela principal do Dimitry mostrando o caminha para se chegar à tela de análise.

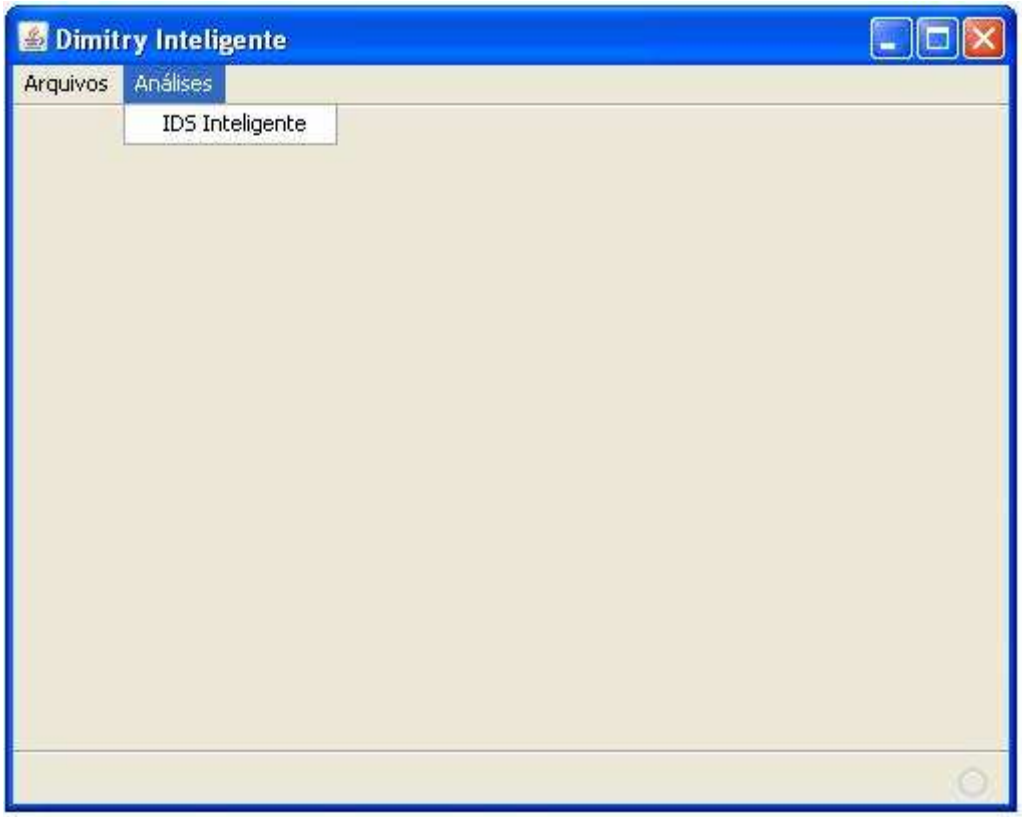

Figura 40. Tela principal do Dimitry Inteligente

Ao clicar no item do menu chamado de IDS Inteligente, o sistema irá abrir a tela principal, onde é iniciada a análise. Esta tela por sua vez possui as mesmas características do sistema da etapa anterior, ou seja, possui um botão iniciar análise outro para parar a mesma, uma tabela onde são mostrados os *logs* gerados alguns campos onde são informados os parâmetros da análise. Esta semelhança pode ser melhor vista na Figura 41 que mostra a interface deste módulo do Dimitry Inteligente.

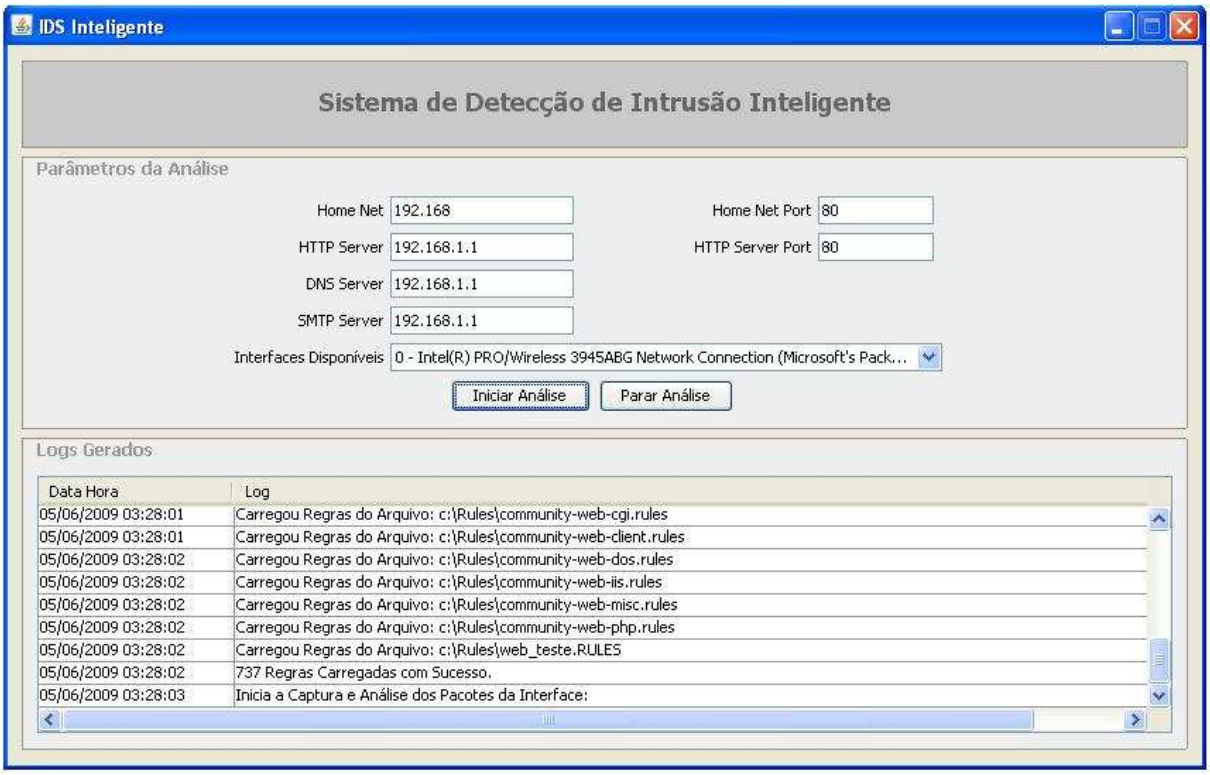

Figura 41. Tela de análise de pacotes do Dimitry Inteligente

As reais diferenças entre os dois sistemas encontram-se após ser iniciada a análise dos pacotes, inicialmente o sistema inteligente trabalha da mesma forma do tradicional, ou seja, vai coletando as informações que trafegam na rede e vai salvando logs de tráfego no banco de dados. É no momento da análise que o sistema inteligente realiza algumas verificações a mais sobre os pacotes.

Os dados são capturados normalmente e analisados pelas regras do IDS, caso uma regra desta venha a ser ativada, junto com ela é ativado o sistema inteligente do Dimitry, onde antes de gerar a mensagem na tela, como era feito anteriormente, ele inicializa uma análise inteligente sobre as informações coletadas.

Na realização da análise, o sistema inteligente utiliza os dados já salvos no banco que têm relação com o pacote suspeito. Nesta análise são abrangidos: consultado à *lista negra* do sistema, verificação do padrão de acesso, respostas à suspeitas anteriores, prioridade indexada à regra que encontrou a suspeita e o tempo médio de envido dos pacotes nos últimos

cinco segundos. Cada consulta mencionada gera um valor de certeza onde estes são trabalhados através dos calculo de fator de certeza gerando um valor final que poderá identificar o ataque. Caso este valor não atinja o grau de confiança suficiente para determinar a resposta automaticamente, um alerta é gerado para o administrador da rede mostrando todos os dados da análise inclusive o fator de certeza calculado, como pode ser visto na Figura 42.

Figura 42. Aviso de suspeita de invasão do Dimitry Inteligente

A relação entre o usuário e o sistema de detecção de intrusão inteligente também continua praticamente igual, em relação ao sistema tradicional. A diferença é que no sistema inteligente o usuário poderá ter que realizar algumas ações mais em alguns casos, como por exemplo, identificar se um determinado *IP* deve ou não ser inserido na lista negra, caso o sistema não consiga determinar esta resposta automaticamente.

Verificando-se as novas ações que o administrador da rede deve realizar, foi montado um novo diagrama de caso de uso, observado na Figura 43.

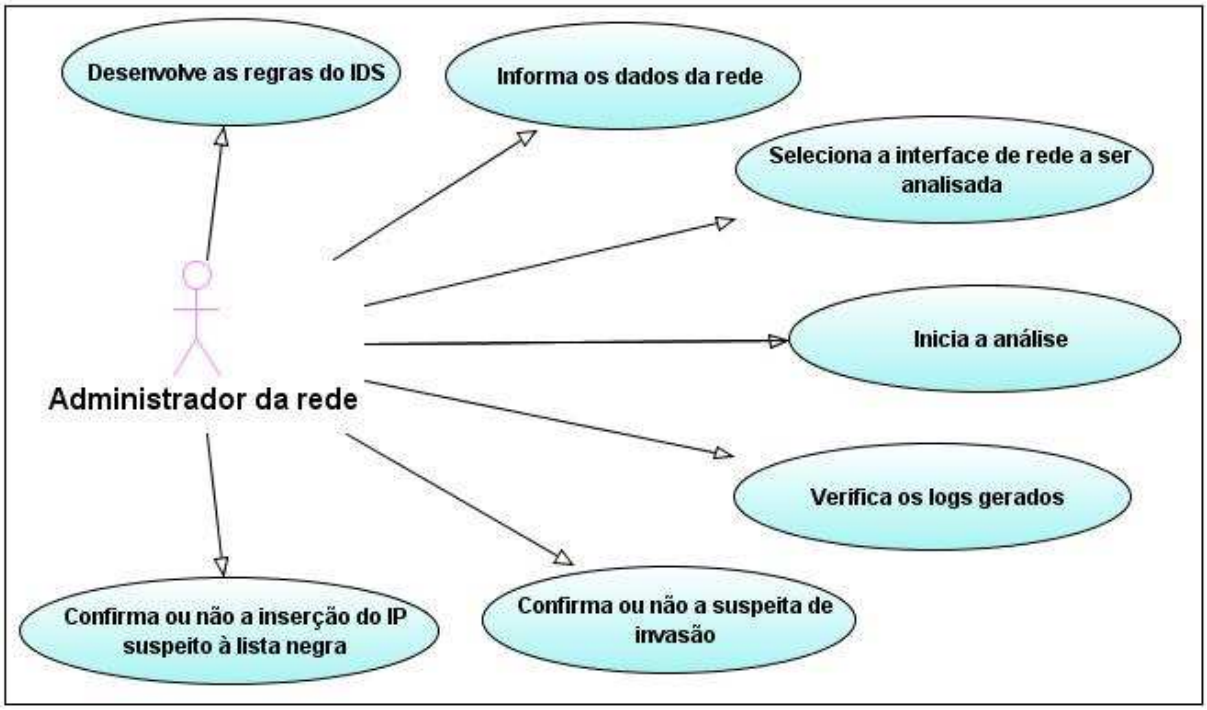

Figura 43. Diagrama de caso de uso do Dimitry Inteligente

Na Figura 44 podemos observar o fluxograma do Dimitry inteligente e por meio deste o funcionamento pode ser entendido de uma forma mais fácil. Como podemos ver a diferença entre os dois sistemas encontra-se após a indicação de pacote suspeito, onde neste sistema é utilizada as técnicas do fator de certeza para identificar o ataque. Caso o resultado deste cálculo venha a gerar um índice de certeza satisfatório as ações do sistema são realizadas de forma automática, caso contrário o sistema irá gerar um alerta pedindo a análise do usuário do sistema.

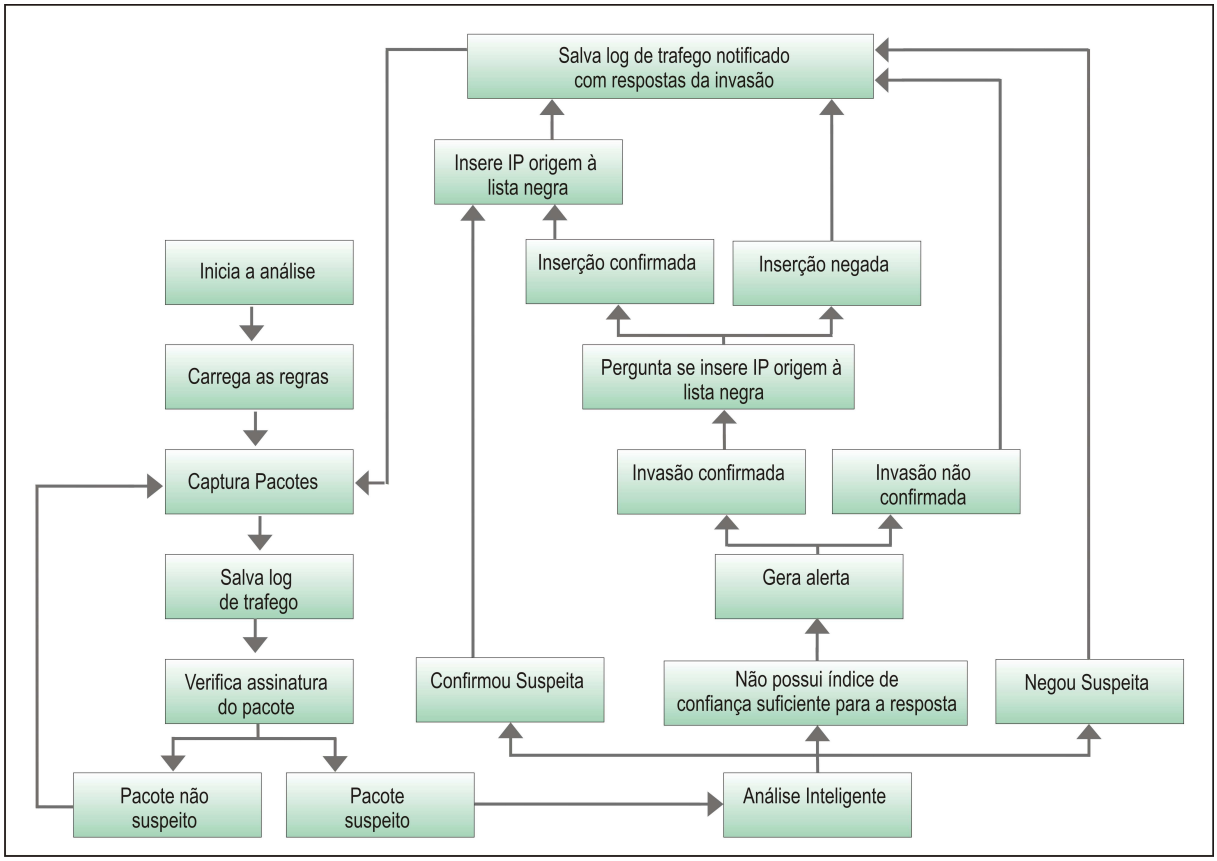

Figura 44. Fluxograma de ações do Dimiry Inteligente

Por fim como podemos perceber as diferenças entre os dois sistemas são consideráveis, onde no sistema inteligente possui um maior número de possibilidades em relação às ações a serem tomadas após a detecção do pacote suspeito.

6.5 REALIZAÇÃO DE TESTE

Com o objetivo de verificar a performance e funcionamento do IDS inteligente e tradicional desenvolvidos foram realizados alguns testes. A seguir iremos verificar mais detalhadamente o desenvolvimento destes testes, desde o ambiente de teste até os resultados obtidos.

6.5.1 Cenário dos Testes

Os testes foram desenvolvidos em um cenário no qual existem três componentes como ilustra a Figura 45, onde há dois computadores ambos com o mesmo sistema operacional, ligados por meio de um *hub*. O computador de *IP* 192.168.1.99 é o que irá realizar os ataques, já o computador 192.168.1.101 será o atacado, lembrando que é neste último que encontra-se o sistema de detecção de intrusão.

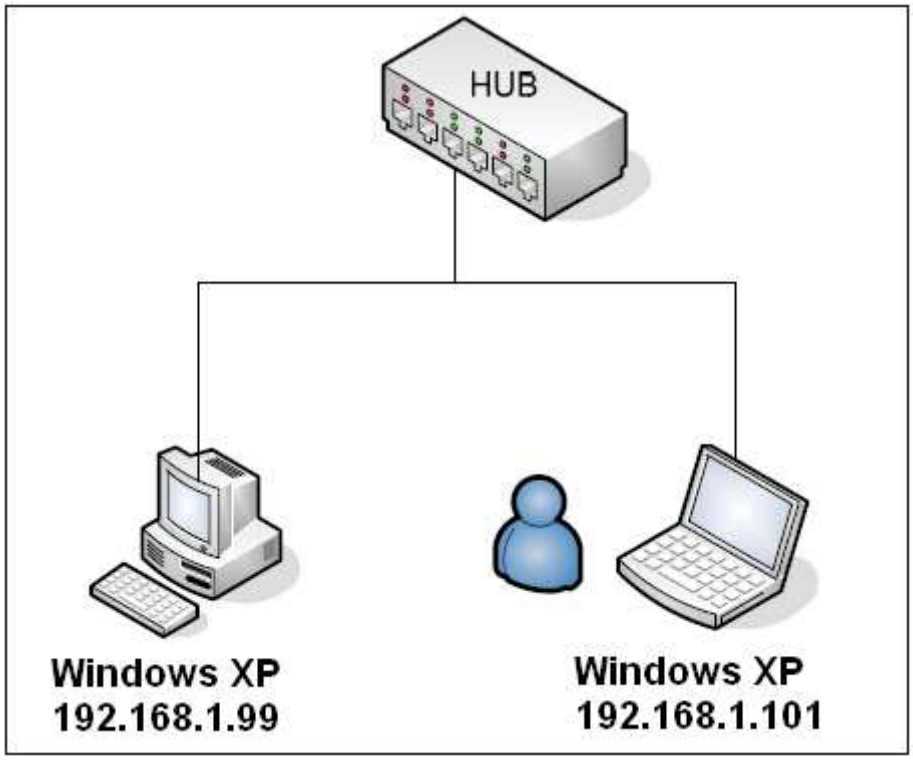

Figura 45. Cenário dos testes

Este cenário foi utilizado para a realização de todos os testes, ou seja, tanto para a verificação de desempenho do Dimitry tradicional quanto para a verificação do Dimitry inteligente. No Quadro 7 pode-se observar as configurações de *hardware* e *software* existentes nas máquinas utilizadas.

Máquina	Hardware	Software
 **Windows XP 192.168.1.99**	CPU: AMD Atholn™ MP 950 MHz RAM: 512 MB Disco: 80 GB Placa Rede: SIS 900 PCI Fast Ethernet Adapter	Windows XP SP1
 **Windows XP 192.168.1.101**	CPU: Intel® Core™ 2 Duo CPU T5250 1.66GHz RAM: 2 GB Disco: 160 GB Placa Rede: Broadcom NetLink™ Gigabit Ethernet	Windows XP SP2 Dimitry Tradicional Dimitry Inteligente WinPcap 4.1 beta2 Jpcap Firebird 2.1 Java™ 6 Update 7

Quadro 7. *Hardwares* e *softwares* das máquinas utilizadas nos testes

Após o detalhamento do ambiente dos testes também é necessário mostrar de que forma os mesmos foram realizados. A seguir iremos verificar quais foram os métodos escolhidos e de que forma os mesmos foram utilizados.

6.5.2 Métodos Utilizados

O desenvolvimento dos testes foi feito por meio de um método bastante simples que consiste na passagem de um arquivo, que possui assinaturas de ataques, de um computador para outro. Atualmente existem alguns softwares desenvolvidos para realização de testes em sistemas de IDS como, por exemplo: Nmap e Nessus. A realização de testes por

meio destes sistemas foi descartada, pois os mesmos utilizam pacotes criptografados e o protótipo desenvolvido ainda não possui nenhum mecanismo capaz de descriptografar tais pacotes.

Como os testes por meio dos sistemas especificados acima não foram possíveis, optou-se por um teste de simples transferência de arquivos, como já mencionado. Na a realização dos mesmos, primeiramente foram criadas algumas regras para o módulo de detecção de intrusão do sistemas. Com o intuito de verificar o funcionamento do sistema perante à ataques verdadeiros, foram criadas 50 regras com assinaturas distintas como, por exemplo: *bot.die, bot.dns, bot.execute, bot.id, bot.nick, bot.open, bot.remove, bot.removeallbut, bot.rndnick, bot.status, bot.sysinfo, bot.longuptime, bot.highspeed, bot.quit, bot.flushdns, bot.secure, bot.unsecure, bot.command, /killthread\s+\d+\b/, cdkey, getcdkey, rndnick, c_rndnick, c_nick, stopspy, redirectspy, loadclones, killclones e rawclones.*

Após serem criadas tais regras, estas assinaturas foram incluídas em um arquivo texto que foi salvo no computador que irá realizar os ataques passando tal arquivo para o computador alvo através da rede.

O método para testar o comportamento do sistema diante de ataques falsos tem o mesmo raciocínio do método de testes dos ataques verdadeiros. Na verificação dos ataques falsos (falsos positivos), foi criada uma única regra que é uma clássica geradora de falsos positivos. A assinatura desta regra é *cmd.exe* este comando pode ser tanto utilizado por vírus para prejudicar o sistema, mas também é utilizado pelo sistema operacional em ações sem intuito de invasão.

Esta mesma assinatura também esta contida em um arquivo de texto salvo no computador de ataque, onde será passado para o computador alvo da mesma forma que os ataques verdadeiros.

Com todas as regras que serão utilizadas pelo sistema de detecção de intrusão do

sistema devidamente e os arquivos com as assinaturas dos ataques montados, basta apenas o início da realização dos testes que será explicado a seguir.

6.5.3 Realização dos Testes

Como podemos verificar anteriormente, foram criados dois arquivos um que possui cinqüenta assinaturas de ataques verdadeiros e outro que possui uma assinatura de um ataque falso, onde este ultimo possui cinqüenta copias, para que também possa realizar cinqüenta ataques falsos.

Em um primeiro momento foi realizado os testes do Dimitry tradicional, onde inicialmente foi zerado todo o banco de dados do mesmo, ou seja, todas as informações foram apagadas e em seguia iniciou-se a transferência do arquivo contendo os ataques verdadeiros, conforme ilustra a Figura 46.

Após o término da primeira transferência a mesma foi repetida mais quatro vezes, totalizando cinco transferências do mesmo arquivo realizadas de forma seqüencial. Esta repetição tem por objetivo verificar o comportamento do sistema diante de um mesmo ataque inúmeras vezes.

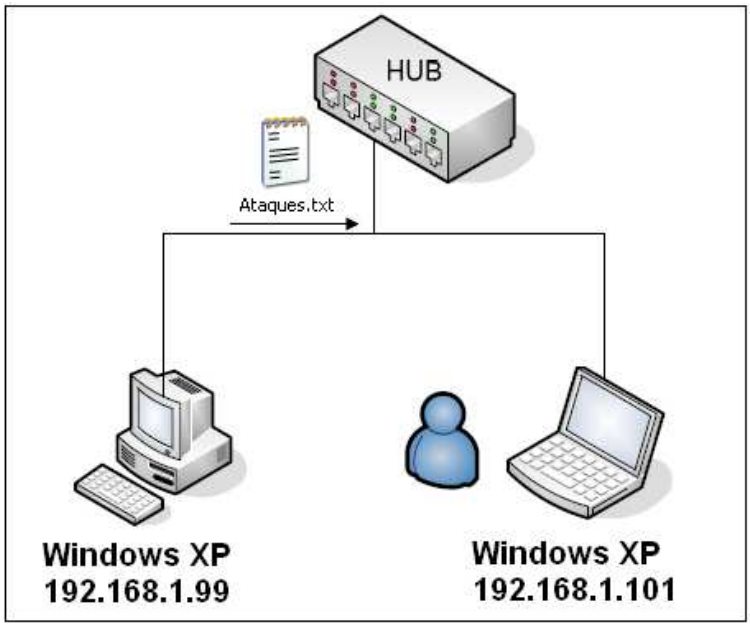

Figura 46. Ambiente de ataques reais

Após o término das análises das assinaturas do primeiro arquivo iniciou-se então o segundo teste sobre este sistema. A realização deste foi por meio da transferência de uma pasta contendo os cinqüenta arquivos que possuem a mesma assinatura falsa, como mostra a Figura 47.

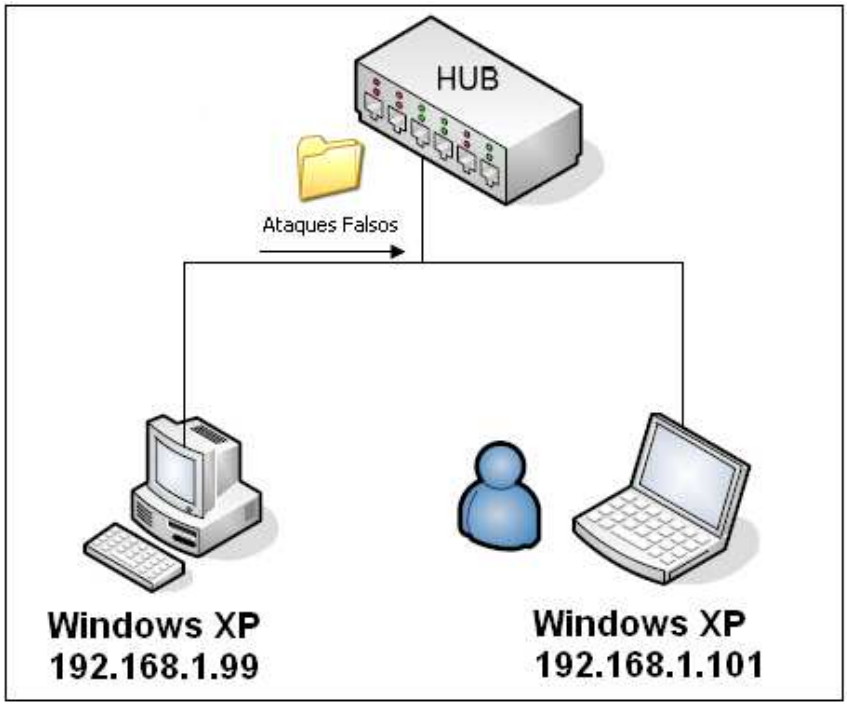

Figura 47. Ambiente de ataques falsos

Os testes do Dimitry inteligente foram desenvolvidos da mesma maneira que os testes anteriores. Inicialmente também foram apagados todos os registros do banco de dados deste sistema e posteriormente para verificar o comportamento do sistema diante de um ataque real foi enviado o mesmo arquivo dos testes anteriores ilustrado na Figura 46.

A análise do comportamento do mesmo em relação a ataques falsos também não fugiu do método que vinha sendo utilizado, onde é realizado por meio da passagem de uma pasta contendo arquivos com assinaturas falsas.

Os resultados dos testes podem ser vistos por meio dos *logs* salvos no banco de dados de cada um dos sistemas verificados acima. Com estes dados é possível verificar a quantidade de ataques confirmados e não confirmados além de saber se o evento salvo foi analisado pelo administrador ou pelo módulo inteligente da ferramenta.

6.5.4 Resultados Obtidos

Com a execução dos testes e com seus dados salvos nos banco de dados dos sistemas, foi fácil verificar os resultados dos testes e verificar um bom desempenho do sistema inteligente perante o tradicional.

Inicialmente iremos verificar os resultados obtidos pelo sistema de detecção de intrusão tradicional chamado Dimitry. No Quadro 8 mostra que foram realizados cinqüenta ataques reais e cinqüenta falsos, também nos detalham o número de ataques confirmados e o de ataques não confirmados em ambas a situações.

Primeiramente observamos que dos ataques reais todos foram detectados, porém estes ataques encontrados geraram um alerta para o administrador analisar, ou seja, foram confirmados pelo mesmo de forma manual totalizando 250 alertas. Os ataques falsos também foram todos identificados como suspeita de invasão, porém foram negados pelo

administrador, gerando assim mais 250 alertas ao mesmo. Por fim, o total de alertas analisados foi de 500.

Ataques	Análises	1ª Invasão	2ª Invasão	3ª Invasão	4ª Invasão	5ª Invasão	Totais
Reais	Nº Ataques	50	50	50	50	50	250
	Ataques (Detecção Automática)	0	0	0	0	0	0
	Ataques (Detecção Manual)	50	50	50	50	50	250
	Não Ataques (Detecção Automática)	0	0	0	0	0	0
	Não Ataques (Detecção Manual)	0	0	0	0	0	0
Falsos	Nº Ataques	50	50	50	50	50	250
	Ataques (Detecção Automática)	0	0	0	0	0	0
	Ataques (Detecção Manual)	0	0	0	0	0	0
	Não Ataques (Detecção Automática)	0	0	0	0	0	0
	Não Ataques (Detecção Manual)	50	50	50	50	50	250

Quadro 8. Resultados dos testes do Dimitry tradicional

Os resultados do sistema de detecção de intrusão inteligente denominado Dimitry inteligente foi mais satisfatório, onde conseguiu diminuir consideravelmente o número de alertas gerados para o administrador, além de conseguir identificar automaticamente os falsos positivos. No Quadro 9 pode-se verificar os resultados deste sistema em cada momento do testes diferenciando também o número de respostas automáticas e manuais.

Primeiramente podemos visualizar que todas as invasões foram identificadas onde obtiveram na primeira invasão um grau de detecção automática bastante razoável conseguindo identificar pouco mais do que a metade, porém após a contínua repetição do ataque este grau de acerto foi melhorando conseguindo a partir da quarta invasão identificar automaticamente todos os ataques. As análises resultaram em um total de 250 identificações onde 214 foram feitas de forma automática pelo sistema e 36 foram realizadas pelo administrador da rede.

A identificação de ataques falsos também obteve um bom resultado onde o sistema conseguiu identificar automaticamente a maioria dos falsos positivos. Na primeira seqüência de invasões os sistema identificou 6 suspeitas de invasões onde as mesmas não foram confirmadas pelo administrador, e as outras 44 o sistema conseguiu identificar automaticamente, não gerando alertas para elas. A partir da segunda seqüência de invasões

foram identificados todos os falsos positivos automaticamente não gerando então mais nenhum alerta de invasão. Das 250 invasões falsas 244 foram identificadas e ignoradas automaticamente e somente 6 delas geraram alerta suspeito, porém também ignorados pelo administrador da rede.

Ataques	Análises	1ª Invasão	2ª Invasão	3ª Invasão	4ª Invasão	5ª Invasão	Totais
Reais	Nº Ataques	50	50	50	50	50	250
	Ataques (Detecção Automática)	29	39	46	50	50	214
	Ataques (Detecção Manual)	21	11	4	0	0	36
	Não Ataques (Detecção Automática)	0	0	0	0	0	0
	Não Ataques (Detecção Manual)	0	0	0	0	0	0
Falsos	Nº Ataques	50	50	50	50	50	250
	Ataques (Detecção Automática)	0	0	0	0	0	0
	Ataques (Detecção Manual)	0	0	0	0	0	0
	Não Ataques (Detecção Automática)	44	50	50	50	50	244
	Não Ataques (Detecção Manual)	6	0	0	0	0	6

Quadro 9. Resultados dos testes do Dimitry inteligente

Por meio das análises pode-se verificar que o sistema inteligente possui um desempenho considerável em relação ao sistema de detecção de intrusão tradicional. Também verifica-se que a princípio o sistema inteligente gera uma quantidade de alertas razoáveis, porém com o passar do tempo, o mesmo começa a identificar as respostas dadas pelo administrador e começa a agir de forma automática podendo identificar tanto ataques reais quanto ataques falsos. Com estes resultados também pode-se dizer os objetivos do sistema foram alcançados, onde se conseguiu diminuir o número de análises realizadas pelo administrador e também minimizar o número de falsos positivos que o sistema tradicional gera.

CONCLUSÃO

Esta pesquisa apresentou o desenvolvimento de um sistema de detecção de intrusão inteligente para a minimização de falsos positivos e alertas excessivos a partir de agentes inteligentes baseados em regras e dos modelos dos fatores de certeza, bastantes utilizados em problemas que envolvem confiança em informações.

Durante a realização desta pesquisa compreendeu-se os aspectos relacionados às políticas de segurança, inteligência artificial e o IDS, abrangendo a análise de pacotes realizados pelos sistemas de detecção de intrusão, os métodos de desenvolvimento das geras dos mesmos assim como os atributos existentes nestas regras e o processo de construção de agentes inteligentes.

Compreendeu-se os possíveis problemas que algumas assinaturas de regras dos sistemas de detecção de intrusão podem gerar, assim como a necessidade de um sistema inteligente que possa suprir boa parte destes problemas, podendo também eliminar alertas excessivos que podem ser gerados.

Neste contexto entendeu-se a complexidade inerente ao desenvolvimento do sistema inteligente baseado em regras que possui uma modelagem de conhecimento incerto e direcionar o sistema por meio da ativação das regras até uma solução.

Visando o tratamento da incerteza do agente inteligente, foi estudada e compreendida a teoria dos agentes inteligentes e FC, que auxilia de maneira simplificada tal agente na solução de problemas a partir do grau de crença em determinadas hipóteses geradas pelo agente inteligente no momento de sua análise abrangendo-se, desta forma sua utilização na construção do sistema inteligente.

Aliado as técnicas do FC foi necessário a utilização de um exemplo fictício para a modelagem matemática que buscou auxiliar na compreensão das etapas envolvidas na

utilização de tal técnica.

Esta pesquisa resultou em dois módulos, um sistema de detecção de intrusão tradicional e outro em um sistema de detecção de intrusão inteligente permitindo uma comparação entre os dois sistemas. Os testes foram realizados a partir de uma situação de ataque hipotética que ambos os sistemas podem sofrer, e foram considerados satisfatórios demonstrando uma performance consideravelmente melhor do sistema inteligente em relação ao sistema tradicional.

Pode-se concluir então que a interdisciplinaridade inerente as ciências exatas é positiva, podendo trazer um diferencial as aplicações desenvolvidas com destaque para esta pesquisa que contemplou as áreas de Redes de Computadores e Inteligência Artificial, resultando em um aplicativo que apresentou maior eficiência que o método tradicional de IDS.

Assim, sugere-se como trabalhos futuros:

a) desenvolver uma modelagem a partir da lógica *fuzzy* ao agente inteligente;
b) realizar a análise estatística que comprova a eficiência do modelo inteligente desenvolvido;
c) desenvolver um módulo ao sistema IDS capaz de analisar os dados criptografados que possam estar presentes nos pacotes;
d) modificar o sistema desenvolvido para que possa analisar os dados que são transferidos aos computadores da rede que não possuam o sistema instalado do ambiente;
e) utilizar a técnica de aprendizagem no desenvolvimento do agente.
f) atualizar o módulo de regra do IDS desenvolvido para a utilização de todos os atributos presentes nas regras do Snort.

REFERÊNCIAS

ANDRADE, Erica. **Um desafio à Ciência.** Disponível em: <http://www.ibvivavida.org.br/noticias.asp?id=3940>. Acessado em: 06 jun. 2009.

BAKER, Andrew R; et al. **Snort 2.1 Intrusion Detection Second Edition.** Syngress Publishing. 2004. 608 p.

BARRETO, Jorge Muniz. **Inteligência artificial no limiar do século XXXI.** 3.ed Florianópolis: Duplic, 2001. 392 p.

BORGES, Pedro Célio; COUTINHO, Rodrigo Trinck. **Análise de Sistemas de Detecção de Intrusos em Redes de Computadores.** Disponível em: <http://www.snort.org.br>. Acessado em: 26 mai. 2009.

BUCHANAN, B. G.; SHORTLIFFE, E. H. **Rule-Based Expert Systems**: the MYCIN Experiments of the Stanford Heuristic Programming Project. California: Addison-Wesley, 1984. p. 3-19.

CASWELL, Brian et al. **Snort 2: Sistema de detecção de intrusão.** Rio de Janeiro: Alta Books, 2003.

COMPAGNO, Ronaldo. **Geração automática para geração de políticas de detecção de intrusões baseadas em evidência de ataque**. Monografia de Mestrado. UNICAMP – Departamento de Ciência da Computação, Campinas - SP, 2005.

COMPUTER ECONOMICS. **Annual Worldwide Economic Damages from Malware**, 2006. Disponível em: <http://www.computereconomics.com/>. Acesso em: 17 set 2008.

COSTA, Ernesto; SIMÕES, Anabela. **Inteligência artificial: fundamentos e aplicações.** Lisboa: FCA, 2004. 582 p.

CRONKHITE, Cathy; MCCULLOUGH, Jack. **Hackers, acesso negado:** o guia completo para a proteção dos seus negócios on-line. Rio de Janeiro: Campus, 2001. 253 p.

GOMES, Daniel A. O. **Detecção de Intrusão em Redes de Computadores Utilizando Classificadores One-Class**. Disponível em: <http://dsc.upe.br/~tcc/20061/DanielGomes.pdf> Acesso em: 10 set. 2008.

HORA, Evandro Cuervo. **Sobre a detecção remota de sniffers para detectores de intrusão em rede.** Disponível em: <http://www.bdtd.ufpe.br>. Acesso em: 24 set. 2008.

HOSSAIN, **Mahmood** , BRIDGES, Susan. A Framework For An Adaptive Intrusion Detection System With Data Mining. 2002. Disponível em: <**http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.4177**>. Acesso em: 08 nov 2008.

HUHNS, Michael N.; SINGH Munindar P. **Readings in agents.** Morgan Kaufmann Publishers Inc., San Francisco, CA, 1997. 523 p.

LISTON, Tom; SKOUDIS, Ed**. Counter Hack Reloaded**: a step-by-step guide to computer attacks and effective defenses. New Jersey: Prentice Hall, 2002. 564 p.

LUGER, George F. **Inteligência Artificial:** Estruturas e Estratégias para a Resolução de Problemas Complexos. 4ª ed. Porto Alegre: Bookman, 2004.

MACHADO, Renato Bobsin. **Uma Abodagem de Detecção de Intrusão Baseada em Sistemas Imunológicos Artificiais e Agentes Móveis**. Monografia de Mestrado. UFSC – Departamento de Ciência da Computação, Florianópolis - SC, 2005.

MARTINS, Cleyton Soares. **Um Sistema Proativo de Prevenção Contra Intrusão.** Disponível em: <http://www.bdtd.ufpe.br>. Acesso em: 03 set. 2008.

MICHALSKI, R e Y. Kodratoff (1990). Research in machine learning; recent progress, classification of methods, and future directions. **Macinhe Learning:** an artificial Intelligence approach (Volume III). Y. Kodratoff e M. R., Morgan Kaufmann: p. 3-30.

MONTORO, R., R., **Funcionamento do Snort**. Spooker Labs, Abril 2004. Disponível em: <http://www.spooker.com.br/papers/como-funcionasnort.pdf>. Acesso em: 06 set 2008.

NORTHCUTT, Stephen et al. **Desvendando:** segurança em redes. Rio de Janeiro: Campus, 2002. 650 p.

PORRAS P. et al. **The Common Intrusion Detection Framework Architecture.** Disponível em: <http://www.isi.edu>. Acesso em: 06 set. 2008

PROCTOR, Paul E. **Practical intrusion detection handbook.** Rio de Janeiro: Prentice Hall, 2001. 359 p.

REZENDE, Solange Oliveira. **Sistemas inteligentes:** fundamentos e aplicações. Barueri, SP: Manole, 2005. 525 p.

RUSSELL, Stuart J.; NORVIG, Peter. **Inteligência artificial.** Rio de Janeiro: Elsevier, 2004

SANTOS, Bruno Ribeiro. **Detecção de Intrusos Utilizando o Snort.** Disponível em <http://www.ginux.ufla.br/node/106>. Acessado em: 26 mai. 2009.

SCHULTER, Alexandre. **Integração de Sistemas de Detecção de Intrusão para Segurança de Grades Computacionais**. Monografia de Mestrado. UFSC – Departamento de Ciência da Computação, Florianópolis - SC, 2006.

SILVA, Halysson Freitas Alves; VITALI, Maycon Maia. **Sistema Inteligente de Detecção de Intrusão.** Disponível em <http://www.snort.org.br>. Acessado em: 26 mai. 2009.

SILVA, Marcos Paulo Crespo; SAMPAIO, Miguel Nuno Saraiva. **Estudo de Sistemas de Detecção e Prevenção de Intrusões:** Uma Abordagem Open Source. Disponível em: <http://mosel.estg.ipleiria.pt/node/91> Acessado em abril 2009.

SIMON, Herbert A; PEREIRA, Luiz Moniz. **As Ciências do artificial:** com um prefácio a edição portuguesa, atualizada de acordo a segunda edição americana de 1981. 2ª ed. Coimbra: Arménio Amado, 1981.

THOMAS, Thomas M. **Segurança de redes:** primeiros passos. Rio de Janeiro: Ciência Moderna, 2007. 395 p.

VALLE, Ana Maria Gomes do. **Crítica e Comparativa de Taxonomias de Sistemas de Detecção de Intrusão.** Disponível em: <http://www.bdtd.ufpe.br>. Acesso em: 03 set. 2008.

WADLOW, Thomas A. **Segurança de redes:** projeto e gerenciamento de redes seguras. Rio de Janeiro: Ed. Campus, 2000. 269 p

XIMENES, Sérgio. **Minidicionário Ediouro**. 7. ed. Rio de Janeiro: Ediouro, 1999.

APÊNDICE A – BASE DE REGRAS

Código	*Regras*	*FC*
1	SE possui na lista negra ENTÃO invasão	75%
2	SE não possui na lista negra ENTÃO invasão	10%
3	SE média padrão tráfego <= 25% ENTÃO invasão	40%
4	SE média padrão tráfego > 25% ENTÃO invasão	20%
5	SE número de tráfego notificado <= 5 ENTÃO invasão	20%
6	SE média ataques confirmados <= 33,33% E número de tráfego notificado <= 5 ENTÃO invasão	5,90%
7	SE média ataques confirmados > 33,33% E média ataques confirmados <= 66,66% E número de tráfego notificado <= 5 ENTÃO invasão	60%
8	SE média ataques confirmados > 66,66% E número de tráfego notificado <= 5 ENTÃO invasão	80%
9	SE prioridade da regra do IDS = 0 ENTÃO invasão	10%
10	SE prioridade da regra do IDS >= 1 E prioridade da regra do IDS <= 6 ENTÃO invasão	60%
11	SE prioridade da regra do IDS > 6 E prioridade da regra do IDS <= 13 ENTÃO invasão	30%
12	SE prioridade da regra do IDS > 13 ENTÃO invasão	20%
13	SE média de segundos entre pacotes <= 0,01 ENTÃO invasão	60%
14	SE média de segundos entre pacotes > 0,01 ENTÃO invasão	20%

APÊNDICE B – ARTIGO

Sistema Inteligente de Detecção de Intrusão em Redes de Computadores

Cleyton Stang[1], Paulo João Martins[2], Priscyla Waleska Targino de Azevedo Simões[2]

cleytonsl@hotmail.com, pjm@unesc.net, pri@unesc.net

[1]Acadêmico do Curso de Ciência da Computação – Unidade Acadêmica de Ciência, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma - SC

[2] Professor do Curso de Ciência da Computação – Unidade Acadêmica de Ciência, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma – SC

Abstract. *The design of network security often cover a good policy security which included firewalls, antivirus, blocks of other doors, but only they can not be sufficient, because if they fail, the protected system may be exposed to the agent attacker. To resolve this question, were created intrusion detection systems (IDS), they have the same function to find invaders that have past the perimeter security. These systems are based in signatures that identify an attack, if very restricted can cause false negative or false positive With the objective to minimize these errors, was developed in search based on an IDS system of intelligent agents. In modeling of the agent used the theory of certainty factors (FC) to reduce the uncertainty in this situation generated.*

Resumo. *O projeto de segurança de redes costuma abranger uma boa política de segurança onde entram firewalls, antivírus, bloqueios de portas entre outros, porém somente estes podem não ser suficientes, pois caso venham a falhar, o sistema protegido poderá ficar exposto ao agente invasor. Para solucionar esta questão, foram criados os sistemas de detecção de intrusão (IDS), que possuem a função de encontrar invasores mesmo que tenham passados pelo perímetro de segurança. Estes sistemas são baseados em assinaturas que identificam um ataque, caso sejam muito restritas podendo ocasionar falsos negativos ou falsos positivos Com o objetivo de minimizar estes erros, foi desenvolvido nesta pesquisa um sistema IDS baseado em de agentes inteligentes. Na modelagem do agente utilizou-se a teoria dos fatores de certeza (FC) para redução da incerteza presente nas situações geradas.*

Palavras-chave*: Inteligência Artificial, Segurança de Redes, Agentes Inteligentes, Fatores de Certeza, Sistema de Detecção de Intrusão Inteligente.*

1. Introdução

Segurança é um fator importante em uma rede, porém, mesmo que esta possua uma boa política, com ferramentas de *firewall* e antivírus, não está imune a um ataque bem sucedido de pacotes invasores, devido aos inúmeros tipos que podem ocorrer e das mais diversificadas formas. Se a rede de computadores interna não possuir um software que monitore e alerte

sobre o tráfego de pacotes indesejados, a mesma pode ficar vulnerável aos mesmos, que podem por conseqüência chegar a um *host* específico, a um conjunto deles ou até mesmo ao servidor, acessando desta forma suas informações e comprometendo a integridade do ambiente.

Atualmente existem alguns softwares para o controle de tráfego, chamados de Intrusion Detection System (IDS), contudo estes softwares não satisfazem integralmente este determinado tipo de controle, pois como dependem de regras e caso estas não sejam criadas de forma correta o sistema pode gerar alertas falsos.

Segundo Northcutt (2002) algumas ferramentas utilizam a filtragem de pacotes com regras genéricas e fazem vários alertas de pacotes inofensivos que são identificados como falsos positivos, outras possuem esta regra de forma muito específica diminuindo o número de falsos positivos, mas aumentando o número de falsos negativos, quando um pacote infectado não é localizado.

Neste sentido, este trabalho propõe uma ferramenta que utilize técnica de agentes inteligentes e Fatores de Certeza integradas a um sistema de detecção de intrusão. Esta ferramenta inteligente tem por objetivo armazenar em sua base de conhecimento os eventos gerados pelos IDS, que poderão ser utilizados para a análise de eventos futuros melhorando a performance na detecção de invasões e conseqüentemente eliminando alguns problemas de tráfego de pacotes, como os supracitados.

2. Sistemas de Deteção de Intrusão para Redes de Computadores

Os sistemas de detecção de intrusão são sistemas que nos auxiliam na monitoração do ambiente que estamos inseridos, estes procuram os ataques ou pacotes invasores que estão tentando invadir ou que já conseguiram passar pela segurança do *Firewall* ou qualquer outro software de segurança inserido. Os IDS não têm a função de somente monitorar as ameaças externas, eles podem também se utilizadas para o monitoramento de ameaças internas como, por exemplo, de *host* tentando violar algum privilégio atribuído a ele. Tais sistemas buscam informações de diversas fontes e de diversos pontos da rede, analisam estas informações no intuito de encontrar alguma anomalia ou sinal de tentativa de intrusão (NORTHCUTT, 2002).

São sistemas projetados para a monitoração de ambientes computacionais, examinam o tráfego com o intuito de identificar ameaças como *scans*, sondas e ataques. Seu objetivo principal é auxiliar-nos identificando os ataques que podem estar sendo direcionados ao seu ambiente ou às ameaças que o seu próprio *host* possa estar passando para o ambiente que esteja inserido. Para que, por meio destas informações, ou avisos, possamos iniciar as ações necessárias para eliminar as ameaças antes que elas atinjam seus objetivos (NORTHCUTT, 2002).

Um sistema de detecção de intrusão é uma união de capacidades para descobrir e responder às ameaças, e suas principais funções são (PROCTOR, 2001, tradução nossa):

a) analisar *logs* anteriores para verificação de ataques conhecidos;

b) analisar o perímetro da rede para verificar se existe alguma ameaça;

c) administrar a configuração da rede;

d) verificar a integridade dos arquivos.

Um sistema pode estar provido de diversos IDS nos quais são posicionados em diversos pontos estratégicos. A localização da fonte de análise é o que os divide em dois grupos: os baseados em rede (NIDS) e os baseados em *host* (HIDS). Os sensores NIDS geralmente são instalados em sub-redes nas quais são diretamente ligas ao *Firewall* ou em

pontos críticos. Já os sensores HIDS residem em um *host* individual e faz o monitoramento do mesmo (NORTHCUTT, 2002).

3. Falsos positivos e negativos

Falsos positivos e falsos negativos são dois problemas resultantes da má análise de assinaturas feitas pelos IDS, porém, não existe sistema de detecção de intrusão que não gere estes problemas.

Os falsos positivos ocorrem quando um sensor intitula uma instrução benigna de um determinado pacote como sendo uma tentativa de ataque. Já os falos negativos são ao contrário, ou seja, quando um pacote invasor passa sem ser visto não gerando desta maneira o alerta ao administrador. Todo IDS gera falsos positivos, estes por sua vez são fáceis de perceber, pois gera o alerta ao administrador, já os falsos negativos não são percebidos, pois não geram o tal alerta, e infelizmente só são constatados depois de terem obtido sucesso em sua invasão (NORTHCUTT, 2002).

Um ponto relevante e que influência diretamente na geração de falsos positivos e falsos negativos é a forma como as assinaturas estão implementadas, umas possuem características mais abrangentes outras possuem mais restritas, que são voltadas mais especificamente para um tipo de invasão e agindo sobre este de forma bastante eficiente. Porém pode deixar de atender os outros tipos de ameaças que o sistema possa sofrer, podendo então gerar uma grande quantidade de falsos negativos. Por outro lado existem assinaturas que possuem características mais globais onde conseguem atender um maior número de intrusões, porém, devido a esta abrangência, começam a intitular tráfegos benignos como ameaças, gerando os chamados falsos positivos (CASWELL, 2003).

Para entender melhor a situação descrita acima vejamos um exemplo simples. Vamos supor que se escreva uma assinatura que procure cmd.exe em qualquer lugar da URL, esta assinatura é geral e combina com diversos ataques que tentam explorar o diretório raiz dos servidores, porém esta assinatura combina com nascmd.exe que não esta relacionado com estas explorações, ocasionando desta maneira falsos positivos. Para minimizar estes erros você decide reescrever sua assinatura, essa nova assinatura é uma URL que contêm /winnt/system32/cmd.exe. Como esta é mais específica ela conseqüentemente diminui os falsos positivos, porém aumenta a quantidade de falsos negativos, pois deixa de localizar alguns pacotes invasores como, por exemplo, os que possuam URL que inclui /winnt/system32/../system32/cmd.exe (NORTHCUTT, 2002).

Este impasse nas assinaturas leva os administradores a terem que optar que tipos de falhas sejam mais aceitáveis no seu ambiente, ou seja, se desejarem que suas redes tenham uma análise diversificada optando para isto de assinaturas mais gerais, conseqüentemente terão vários falsos positivos, e se optarem por capturar somente um ataque em especifico terão vários falsos negativos.

4. Agente inteligente

Tudo o que pode ser capaz de perceber o ambiente por meio de sensores a agir sobre este mesmo ambiente por meio de atuadores é considerado um agente. Ele é algo que interage com o ambiente que está inserido, podendo mudar sua forma de atuação com o passar do tempo, devido a sua capacidade de aprendizagem, diferentemente dos programas computacionais tradicionais que somente executam uma seqüência pré-determinada de comandos (RUSSELL; NORVIG, 2004).

Uma agente é uma entidade virtual, capaz de agir sobre o ambiente que está

inserido, podendo comunicar-se com outros agentes inseridos no mesmo ambiente. É capaz de perceber de modo limitado seu ambiente e age, através de recursos próprios, sobre o mesmo para que possa atingir seus objetivos levando em conta os resultados de suas funções de percepção e comunicação (REZENDE, 2005).

Com base nas definições, desenvolveu-se a modelagem do agente. Nesta escolheu-se os fatores de certeza para agregar inteligência em sua concepção, que tem por objetivo auxiliar ao processo de tomada de decisão em sistemas com incertezas associadas. Tal modelo foi escolhido em virtude de ser bastante utilizado em sistemas que possuem graus de incertezas consideráveis que é o caso do sistema desenvolvido que possui um pequeno conjunto de regras necessitando desta forma de tal técnica para o tratamento das incertezas encontradas além da eficiência do mesmo relatada na literatura.

Com o objetivo de melhor fixar o entendimento da modelagem, a Figura 1 apresenta um esquema da mesma. Podemos observar por meio desta que o agente possui um módulo de percepção que é ativado quando uma regra do IDS é ativada, após isto o agente realiza as buscas das informações do pacote suspeito encontrado. Estas informações são então enviadas para a base de regras do agente onde serão tratadas e geraram cinco hipóteses onde serão analisadas pelo FC. O resultado final desta análise será um valor entre o intervalo de -1 a 1, onde dependendo deste o sistema identificará o tipo de ação a ser tomada. Estas ações podem ocorrer de três formas distintas, a suspeita de invasão é confirmada automaticamente, a suspeita é anulada automaticamente ou o sistema gera um alerta quando o grau de certeza para resposta automática não é atingido.

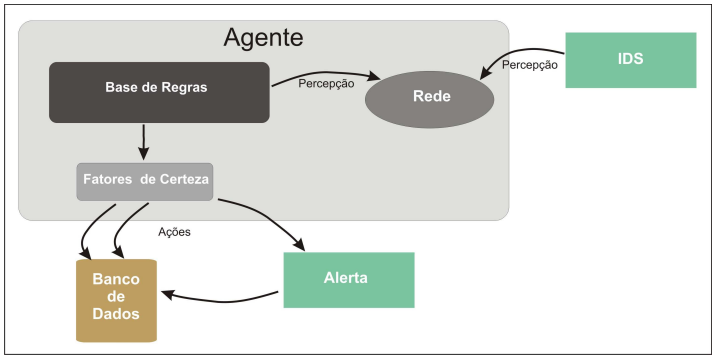

Figura 1. Esquema de representação do agente

5. Fatores de certeza

O desenvolvimento deste método teve início em 1970, onde foi projetado para a utilização em um sistema para diagnóstico medico baseado em regras chamado de MYCIN. Por meio deste novo método atribuía-se a cada uma das regras do sistema um Fator de Certeza (FC), no qual ao final da análise quantificavam-se os graus de certeza que cada uma das regras, concluindo desta forma o grau de certeza da resposta (COSTA; SIMÕES, 2004).

FC é uma maneira simples de combinar crença e descrença em um único número que fica geralmente entre o intervalo -1 e 1. Um FC positivo significa que a evidência suporta a hipótese caso a média de crença seja maior que a média da descrença, já um FC negativo significa que a evidência favorece a negação da hipótese caso a média de crença seja menor que a de descrença (BARRETO, 2001).

Uma forma de melhor compreender o funcionamento do calculo do fator de certeza é através da realização de exemplos. A seguir iremos verificar passa a passo este cálculo, desde a geração dos fatores de confiança das regras até os resultados finais.

O sistema define previamente seu objetivo (invasão) que em um primeiro momento é desacreditado, recebendo então valor zero, ou seja, como não foi definida nenhuma evidência que validada-se este objetivo, sua crença (P(H)) é zero conforme Quadro 1.

Objetivo	P(H)
invasao	0,00

Quadro 1. Probabilidades a priori dos ataques

Supondo uma situação hipotética: O sistema de detecção de intrusão inteligente capturou um pacote e ativou o módulo do agente que disparou cinco regras como podemos verificar na Figura 2.

1- *SE* possui_na_lista_negra *ENTAO* invasao (FC = 0.75)
4- *SE* 25% < media_padrao_trafego *ENTAO* invasao (FC = 0.2)
5- *SE* numeros_trafego_notificado <= 5 *ENTAO* invasao (FC = 0.2)
12- *SE* prioridade_regra > 13 *ENTAO* invasao (FC = 0.20)
13- *SE* media_segundos_entre_pacotes <= 0.01 *ENTAO* invasao (FC = 0.60)

Figura 2. Regras ativadas

O conjunto de regras apresentada o domínio do sistema especialista em relação ao cada evento gerado, onde cada conhecimento possui um grau de confiabilidade representado pelo FC. A partir de cada evidencia que é confirmada é ativado o conseqüente (ENTÃO) que sempre resultará em um objetivo do sistema, ou seja, em uma resposta. Contudo, a partir das entradas e saídas geradas pelo sistema, o calculo do fator de certeza é realizado chegando desta forma em um resultado.

Tendo em mente os resultados da consulta realizada pelo sistema, verifica-se que IP de origem consta na lista negra, através desta informação o agente ativou a regra. Assim a probabilidade de invasão representada por p(H) é comparada com a probabilidade da hipótese de ter invasão dada à evidência possui na lista negra representada por p(H|E). Nota-se que houve um aumento na crença de zero, valor inicial, para 0,75, FC da regra que atingiu o objetivo.

Com o aumento da crença na hipótese invasão representada pela regra 1 da Figura 2 calcula-se de a nova MC. Tendo que a probabilidade de invasão p(H) é definida inicialmente por 0 e a probabilidade de invasão dado que possui na lista negra p(H|E) é de 0,75 conforme FC da regra 1.

$$MC = \frac{P(H \mid E) - P(H)}{1 - P(H)} = \frac{p(Invasao \mid ListaNegra) - p(Invasao)}{1 - p(Invasao)} = \frac{0,75 - 0}{1 - 0} = \frac{0,75}{1} = 0,75$$

A partir do aumento da crença de invasão para 0,75, calcula-se o novo.

$$FC = \frac{MC_{Invasao} - MD_{Invasao}}{1 - \min[MC_{Invasao}, MD_{Invasao}]} = \frac{0,75 - 0}{1 - 0} = 0,75$$

Também foi possível verificar uma segunda evidencia que é a de padrão de acesso do IP que originou o pacote. Esta consulta demonstrou que o pacote está trafegando com um grau de igualdade de 65% em relação ao seu padrão de acesso geral, ativando desta forma a regra de número 5 onde a mesma tem por hipótese o mesmo objetivo do sistema (invasão).

Neste caso a probabilidade de não invasão representada por p(H) é comparada com a probabilidade da hipótese de ter invasão dada a evidencia média padrão de tráfego representada por p(H|E). Nota-se que também houve uma diminuição na crença de 0,75 para 0,20.

$$MD = \frac{P(H) - P(H \mid E)}{P(H)} = \frac{p(Invasao) - p(Invasao \mid MediaPadraoTrafego)}{p(Invasao)} = \frac{0,75 - 0,2}{0,75} = \frac{0,55}{0,75} = 0,73333$$

Com a diminuição da crença da invasão de 0,75 para 0,20 calcula-se o novo FC.

$$FC = \frac{MC_{NaoInvasao} - MD_{NaoInvasao}}{1 - \min[MC_{NaoInvasao}, MD_{NaoInvasao}]} = \frac{0,75 - 0,7333}{1 - 0,7333} = 0,0626$$

A terceira evidencia gerada pela consulta é a da média de ataques confirmados, porém o numero de tráfego notificado é menor que 5 ativando assim a regra de numero 5 do agente, onde tem por hipótese o objetivo invasão e possui o FC de 0,20. Como podemos verificar anteriormente a hipótese de ser uma invasão p(H) passou de 0,75 para 0,0626 devido a evidencia do padrão de acesso. Esta por sua vez é comparada com a hipótese de invasão dado a média de ataques confirmados p(H|E) que é de 0,2, demonstrando desta forma um novo aumento na crença da invasão.

Nesta situação calcula-se primeiramente o novo MC levando em conta a nova evidencia gerada.

$$MC = \frac{P(H \mid E) - P(H)}{1 - P(H)} = \frac{p(Invasao \mid MediaAtaquesConfirmados) - p(Invasao)}{1 - p(Invasao)} = \frac{0,20 - 0,0626}{1 - 0,0626} = \frac{0,1374}{0,9374} = 0,1466$$

A partir do novo MC gerado é calculado o novo FC da hipótese de invasão.

$$FC = \frac{MC_{Invasao} - MD_{Invasao}}{1 - \min[MC_{Invasao}, MD_{Invasao}]} = \frac{0,1466 - 0,7333}{1 - 0,1466} = -0,6875$$

A regra do sistema de detecção de intrusão que encontrou a suspeita possui um atributo que indica a prioridade da mesma onde também é utilizada pelo módulo do agente para identificar um ataque, este atributo compõe a quarta evidencia gerada neste exemplo e possui um grau de prioridade igual a 15. Através deste grau o agente inteligente ativou a regra 12. Esta regra por sua vez ativa a hipótese de invasão com o FC de 0,2, constatando desta forma um novo aumento na crença da invasão, pois o p(H) da mesma foi anteriormente definido -0,6875 pela regra 6. Desta forma calcula-se a nova MC da hipótese de invasão.

$$MC = \frac{P(H \mid E) - P(H)}{1 - P(H)} = \frac{p(Invasao \mid \mathrm{Pr}ioridade\,\mathrm{Re}\,gra) - p(Invasao)}{1 - p(Invasao)} = \frac{0,20 - (-0,6875)}{1 - (-0,6875)} = \frac{0,8875}{1,6875} = 0,5259$$

Como houve duas evidências que aumentaram a crença de uma mesma hipótese o grau de crença delas é combinado gerando uma nova MC.

$$MC = MC[H, E_1] + MC[H, E_2] * (1 - P[H, E_1]) = MC[Invasao, MediaAtaquesConfirmados] + MC[Invasao, \mathrm{Pr}ioridade\,\mathrm{Re}\,gra] * (1 - p[Invasao, MediaAtaquesConfirmados]) = 0,1466 + 0,5259 * (1 - 0,1466) = 0,5954$$

Após o calculo da nova MC, calcula-se o novo FC a partir da hipótese de ser

invasão.

$$FC = \frac{MC_{NaoInvasao} - MD_{NaoInvasao}}{1 - \min[MC_{NaoInvasao}, MD_{NaoInvasao}]} = \frac{0,5954 - 0,7333}{1 - 0,5954} = \frac{-0,1379}{0,4046} = -0,3408$$

Por fim o sistema realiza o calculo da quinta evidencia gerada que é a média do tempo entre os pacotes nos últimos cinco segundos. Esta média conforme informada no exemplo é de 0,01 segundos ativando desta forma a regra 13 do sistema que tem o FC de 0,60 para a hipótese de invasão. Como a hipótese de invasão p(H) passou a ser de -0,3408 (gerada pela regra 12), e a nova hipótese de invasão dado a média de segundos entre pacotes p(H|E) é de 0,60, confirmou-se um novo aumento da crença de invasão calculando-se desta forma a nova MC.

$$MC = \frac{P(H \mid E) - P(H)}{1 - P(H)} = \frac{p(Invasao \mid MediaSegundosPacotes) - p(Invasao)}{1 - p(Invasao)} = \frac{0,6 - (-0,3408)}{1 - (-0,3408)} = 0,7016$$

Como houve novamente duas evidências que aumentaram a crença de uma mesma hipótese o grau de crença delas é combinado gerando uma nova MC.

$$MC = MC[H, E_1] + MC[H, E_2] * (1 - P[H, E_1]) = MC[Invasao, Prioridade Regra] + MC[Invasao, MediaSegundosPacotes] * (1 - p[Invasao, Prioridade Regra]) = 0,5954 + 0,7016 * (1 - 0,5954) = 0,8792$$

Com a nova MC de invasão calculada, calcula-se também o novo FC.

$$FC = \frac{MC_{NaoInvasao} - MD_{NaoInvasao}}{1 - \min[MC_{NaoInvasao}, MD_{NaoInvasao}]} = \frac{0,8792 - 0,7333}{1 - 0,7333} = \frac{0,1459}{0,2667} = 0,5470$$

O resultado final do sistema é o último FC gerado representando o fator de certeza da invasão. O evento gerado nesta análise é a geração de um alerta ao administrador, pois o valor do FC da invasão é menor que o índice o grau de confiança definido para que o sistema possa gerar respostas automáticas que é de 0,75, caso fosse maior a resposta á invasão seria gerada de forma automática.

6. Definição das regras

As regras foram elaboradas a partir de cinco cenários baseados em algumas das possíveis maneiradas de detectar ataques sem utilização das assinaturas do IDS, onde cada um destes possui um conjunto de regras e uma variável final. Cada um destes cenários pode ativar somente uma regra que possui o fator de confiança que será atribuído à variável final e este valor por sua vez, representa a confiança de que a regra que ativou encontrou uma invasão. Os valores utilizados nas regras são gerados por meio de cruzamento de informações existentes no banco de dados, como o registro de tráfego, registros de análises anteriores, os dados da lista negra e os dados da regra do IDS.

O primeiro cenário é o da lista negra, é o mais simples, pois somente verifica se existe ou não um determinado IP na lista, a variável final deste cenário é a lista_negra_fc que receberá o fator de confiança da regra a ser ativada

Outro cenário utilizado é o padrão de acesso, este por sua vez realiza uma consulta nos registros do banco de dados em busca do padrão de acesso do pacote. Primeiramente é realizada a média ponderada dos acessos realizados pelo suspeito, retornando o horário médio de acesso, posteriormente é verificada qual a média de acesso do pacote analisado no horário do tráfego e por fim é feita uma média entre estas duas médias para chegar no percentual de igualdade, onde este é analisado nas regras do cenário gerando então um FC para a variável

final que é chamada de padrao_trafego_fc.

Registros anteriores referentes às análises e conclusões realizadas nos pacotes são usados no terceiro cenário, as regras deste verificam o percentual de invasões confirmadas, ou seja, é verificada a quantidade de alertas gerados por um determinado pacote e posteriormente a quantidade de invasões confirmadas dentro destes alertas, gerando desta maneira um percentual de invasão. Este percentual então é tratado pelas regras que irão “alimentar” a variável analises_anteriores_fc que é a variável resposta deste cenário.

Já o quarto cenário representa uma verificação bastante simples, pois nele é tratado o grau de prioridade da regra ativada pelo IDS. Neste é pego o numero da prioridade da regra, e verificado o intervalo que ele pertence, caindo então em uma das regras do agente inteligente. As regras deste cenário que tem como resultado final a variável chamada de prioridade_regra_fc.

Por fim o ultimo cenário utilizado é o que busca o tempo médio de envido dos pacotes de um determinado IP nos últimos cinco segundos. Este cenário foi desenvolvido com o intuito de encontrar possíveis tentativas de estouros de *buffer*. A média gerada por este cenário é analisada pelas regras deste e gerado o FC da variável final media_tempo_trafego_fc.

7. Resultados

Após a realização de testes, verificamos que os resultados do sistema de detecção de intrusão inteligente denominado foram satisfatórios, onde conseguiu diminuir consideravelmente o número de alertas gerados para o administrador, além de conseguir identificar automaticamente os falsos positivos. No Quadro 2 podemos verificar mais detalhadamente a realização dos testes e os resultados deste sistema em cada momento do testes diferenciando também o número de respostas automáticas e manuais.

Primeiramente podemos visualizar que todas as invasões realizadas foram identificadas onde obtiveram na primeira invasão um grau de detecção automática bastante razoável conseguindo identificar pouco mais do que a metade, porém após a contínua repetição do ataque este grau de acerto foi melhorando conseguindo a partir da quarta invasão, identificar automaticamente todos os ataques. As análises resultaram em um total de 250 identificações onde 214 foram feitas de forma automática pelo sistema e 36 foram realizadas pelo administrador da rede.

A identificação de ataques falsos também obteve um bom resultado onde o sistema conseguiu identificar automaticamente a maioria dos falsos positivos. Na primeira seqüência de invasões os sistema identificou 6 suspeitas de invasões onde as mesmas não foram confirmadas pelo administrador, e as outras 44 o sistema conseguiu identificar automaticamente, não gerando alertas. A partir da segunda seqüência de invasões foram identificados todos os falsos positivos automaticamente não gerando então mais nenhum alerta de invasão. Das 250 invasões falsas 244 foram identificadas e ignoradas automaticamente e somente 6 delas geraram alerta suspeito, porém também ignorados pelo administrador da rede.

Ataques	Análises	1ª Invasão	2ª Invasão	3ª Invasão	4ª Invasão	5ª Invasão	Totais
Reais	Nº Ataques	50	50	50	50	50	250
	Ataques (Detecção Automática)	29	39	46	50	50	214
	Ataques (Detecção Manual)	21	11	4	0	0	36
	Não Ataques (Detecção Automática)	0	0	0	0	0	0
	Não Ataques(Detecção Manual)	0	0	0	0	0	0
Falsos	Nº Ataques	50	50	50	50	50	250
	Ataques (Detecção Automática)	0	0	0	0	0	0
	Ataques (Detecção Manual)	0	0	0	0	0	0
	Não Ataques (Detecção Automática)	44	50	50	50	50	244
	Não Ataques (Detecção Manual)	6	0	0	0	0	6

Quadro 2. Resultados dos testes do IDS inteligente

8. Considerações finais

Esta pesquisa apresentou o desenvolvimento de um sistema de detecção de intrusão inteligente para a minimização de falsos positivos e alertas excessivos a partir de agentes inteligentes baseados em regras e dos modelos dos fatores de certeza, bastantes utilizados em problemas que envolvem confiança em informações.

Durante a realização desta pesquisa pôde-se compreender os aspectos relacionados às políticas de segurança, inteligência artificial e o IDS, abrangendo a análise de pacotes realizados pelos sistemas de detecção de intrusão, os métodos de desenvolvimento das geras dos mesmos assim como os atributos existentes nestas regras e o processo de construção de agentes inteligentes.

Compreendeu-se os possíveis problemas que algumas assinaturas de regras dos sistemas de detecção de intrusão podem gerar, assim como a necessidade de um sistema inteligente que possa suprir boa parte destes problemas, podendo também eliminar alertas excessivos que podem ser gerados.

Neste contexto entendeu-se a complexidade inerente ao desenvolvimento do sistema inteligente baseado em regras que possui uma modelagem de conhecimento incerto e direcionar o sistema através da ativação das regras até uma solução concisa.

Visando o tratamento da incerteza do agente inteligente, foi estudada e compreendida a teoria dos agentes inteligentes e FC que auxilia de maneira simplificada tal agente na solução de problemas a partir do grau de crença em determinadas hipóteses geradas pelo agente inteligente no momento de sua análise abrangendo-se desta forma sua utilização na construção do sistema inteligente.

Aliado as técnicas do FC fez-se necessário a utilização de um exemplo fictício para a modelagem matemática que buscou auxiliar na compreensão das etapas envolvidas na utilização de tal técnica.

Por meio das análises podemos verificar que o sistema inteligente demonstrou um bom desempenho, onde a principio gerou uma quantidade de alertas razoáveis, porém com o passar do tempo, começou a identificar as respostas dadas pelo administrador e agir de forma automática podendo identificar tanto ataques reais quanto ataques falsos.

Com estes resultados também podemos dizer que os objetivos do sistema foram alcançados, onde houve uma diminuição no número de análises realizadas pelo administrador e também minimizar o numero de falsos positivos que o sistema tradicional gerar.

Referência

BARRETO, Jorge Muniz. Inteligência artificial no limiar do século XXXI. 3.ed Florianópolis: Duplic, 2001. 392 p.

CASWELL, Brian et al. **Snort 2: Sistema de detecção de intrusão.** Rio de Janeiro: Alta Books, 2003.

COSTA, Ernesto; SIMÕES, Anabela. **Inteligência artificial: fundamentos e aplicações.** Lisboa: FCA, 2004. 582 p.

NORTHCUTT, Stephen et al. **Desvendando:** segurança em redes. Rio de Janeiro: Campus, 2002. 650 p.

PROCTOR, Paul E. **Practical intrusion detection handbook.** Rio de Janeiro: Prentice Hall, 2001. 359 p.

REZENDE, Solange Oliveira. **Sistemas inteligentes:** fundamentos e aplicações. Barueri, SP: Manole, 2005. 525 p.

RUSSELL, Stuart J.; NORVIG, Peter. **Inteligência artificial.** Rio de Janeiro: Elsevier, 2004