UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

PÓS-GRADUACÃO EM CIÊNCIA DA COMPUTACÃO

Ana Maria Gomes do Valle

"ANÁLISE CRÍTICA E COMPARATIVA DE TAXONOMIAS DE SISTEMAS DE DETECÇÃO DE INTRUSÃO"

ESTE TRABALHO FOI APRESENTADO A PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: FABIO QUEDA BUENO DA SILVA, PhD.

RECIFE, OUTUBRO/2002

Resumo

Segurança é um problema muitas vezes presente nas redes de computadores. Existem diversos mecanismos para abordá-la, dentre eles, *firewalls*, análise de vulnerabilidades, criptografia, certificados digitais, *VPNs (Virtual Private Network)*, detecção de intrusão e, até, uma política de segurança bem planejada e aplicada. Todos são utilizados de forma que haja uma chance elevada de se evitar maiores comprometimentos.

Os riscos crescentes de ataques a sistemas computacionais advindos do aumento das atividades não autorizadas, não somente de atacantes externos, mas também de internos (como no caso de empregados descontentes e usuários abusando de seus privilégios), mostra a importância dos Sistemas de Detecção de Intrusão (IDS) no aumento da segurança dos sistemas computacionais.

O grande número de IDS desenvolvidos e disponibilizados atualmente, tanto comercialmente, quanto para pesquisa, bem como as muitas questões que envolvem os mesmos, trazem consigo problemas e restrições que precisam ser identificados para possíveis melhoramentos.

A área de detecção de intrusão é muito dinâmica, em constante atualização e, com o surgimento de novos IDS, sempre em busca do aprimoramento. Diante disto, é proposta uma taxonomia dos IDS em relação às características mais destacadas encontradas nos 3 (três) módulos principais de um IDS: coleta de dados, análise e componente de resposta. Também é elaborado um *survey* de diversos IDS, escolhidos pela diversidade de seus mecanismos. Em face deste estudo foi possível propor um modelo de um IDS, derivado de um modelo conceitual já validado por aplicação na área de Administração de Sistemas.

Abstract

Security is often a problem present in computer networks. Diverse mechanisms exist to approach it. Amongst them, firewalls, analysis of vulnerabilities, cryptography, digital certificates, VPN (Virtual Private Network), intrusion detection and a well planned and applied security policy. All have a high possibility of preventing a greater compromise.

The increasing risks of attacks in computational systems, due to the increase in non authorized activities, not only by external intruders, but also by insiders (as in the case of users abusing its privileges), show the importance of Intrusion Detection Systems (IDS) in the improvement of computational system's security.

The great number of commercial IDS developed and available currently, as well as the many questions that involve them, brings with its problems and restrictions that they need to be identified for possible improvements.

The area of intrusion detention is very dynamic, in constant update and, with the development of new IDS, always in search of improvement. Ahead of this, a taxonomy of the IDS is proposal, in relation to the more detached features in the 3 (three) main modules of a IDS: data collect, analysis and response component. Also elaborate a survey of diverse IDS, chosen for the diversity of its mechanisms. In face of this study it was possible to consider a model of a IDS, derived from a conceptual model already validated by application in the area of System Administration.

Sumário

Índice de Figuras

Dedico este trabalho:
a você, César, meu eterno namorado;
a você Felipe, meu herói;
a você Patrícia, minha melhor amiga.
Amo vocês mais que tudo nesta vida.

Agradecimentos

Foi uma caminhada que, confesso, não foi fácil... Mas como meu orientador disse, em uma das muitas reuniões, “existem coisas na vida que não são tão fáceis”. Talvez por isso eu tenha dado tanta importância a esta vitória. E não consegui sozinha. Também envolveu pessoas sem as quais não teria obtido sucesso. A elas minha eterna gratidão:

A Deus, que sempre esteve ao meu lado;

À minha mãe, referência de fortaleza, perseverança, exemplo de honestidade, respeito, dedicação, fonte de carinho, amparo, afeto, devoção e amor incondicional, obrigada por ter nascido;

Ao meu marido, amante e amigo, que soube esperar e me amar tanto. Obrigada pelas inúmeras correções no meu trabalho, principalmente aquelas dicas em telefonemas interurbanos;

Ao meu lindo filho Felipe, que há quatorze anos me dá lições de vida;

À minha meiga e carinhosa filha Patrícia, amiga e confidente, obrigada meu Deus, por ela ter nascido e ser tão única para mim;

Ao meu orientador Fabio, pela compreensão das minhas dificuldades e confiança depositada em mim;

Ao Evandro, pelas muitas horas que deixou sua família e trabalho para esclarecer minhas dúvidas, bem como a toda a equipe da Tempest Security Technologies;

Ao Claudino, por me escutar nos bate-papos diários e sempre ter uma palavra de conforto;

À minha querida irmã Bia, por ter me recebido em sua casa e nunca perguntado quando meu mestrado ia terminar;

Aos meus irmãos, Paola e Totonho, e também à querida tia Niza, que, mesmo distantes, apoiaram-me tanto e não deixaram que eu desanimasse;

Aos meus sogros, Leonor e Cecílio, por sempre me fazerem acreditar que eu conseguiria;

Aos meus professores, aos meus colegas da turma “98”, ao pessoal da Administração de Sistemas e ao de Suporte (Tepedino, Nádia e Fernando) e às funcionárias, Neide e Lilia, sempre tão solícitas. Vocês “fazem” nosso Centro de Excelência;

Aos meus colegas do Núcleo de Informática do IBAMA, Maurício, Wilson, Barbosa e Eduardo, que souberam compreender minha ausência ao trabalho e àqueles que me deram a oportunidade do meu aperfeiçoamento.

Ao Gen Mello e à Gabriela e, também, ao Gen Seixas e à Vera por lutarem para que me fosse possível concretizar um sonho. Àqueles muitos amigos da 11ª Cia E Bld, por me receberem na minha nova família e acreditarem no meu sucesso. Vocês são pessoas muito especiais.

Capítulo 1

Introdução

Esta dissertação de mestrado trata de uma nova taxonomia de Sistemas de Detecção de Intrusão (IDS). A construção desta taxonomia se baseou, como será visto, nas taxonomias de [Debar, 1999] e [Axelsson, 2000] e envolveu, também, a construção de um *survey* de IDS, que pode ser visto como um método de pesquisa com a finalidade de descrição dos sistemas diante de suas características principais, definidas nesta nova taxonomia apresentada. Este estudo nos leva a uma comparação crítica das taxonomias de IDS, identificando problemas e restrições. Diante deste trabalho foi possível, ainda, propor um modelo de um IDS que contempla uma nova abordagem.

Este capítulo irá descrever os conceitos de segurança computacional, apresentar o estado da arte em taxonomias e *survey* de IDS, a motivação, os objetivos e a metodologia utilizada nesta dissertação.

1.1 Segurança Computacional

Segundo [Geus, 1999], há uma forte tendência de que a segurança de um sistema seja vista como sendo um modelo em camadas, que se inicia pela segurança física do mesmo, pois, em muitos casos, por mais seguro que seja um sistema, um acesso indevido ao console, ao hardware ou mesmo ao meio físico pode comprometer todo ele. Logo após o nível citado, vem a segurança do *kernel* do sistema operacional, pois é este que responde pelas principais funções do sistema. Em seguida, temos os protocolos de comunicação, onde são muitas as fragilidades conhecidas. No mesmo nível temos o sistema de arquivos, que responde pelo acesso e os direitos de acesso que os usuários têm sobre os arquivos, e, também, os sistemas de autenticação, que têm como finalidade, confirmar se o usuário é quem ele diz ser. Em um nível superior estão os serviços oferecidos por este sistema, como *DNS, NIS, NFS, HTTP, SSH, SMTP* e outros. Finalmente, encontram-se as aplicações do usuário, onde diversos problemas podem acontecer, como por exemplo, levando a exaurir os recursos do sistema (por exemplo, espaço em disco e memória do sistema) causando uma negação de serviço. Cada componente deste modelo pode apresentar potenciais pontos de vulnerabilidade e, como tais, devem ser tratados.

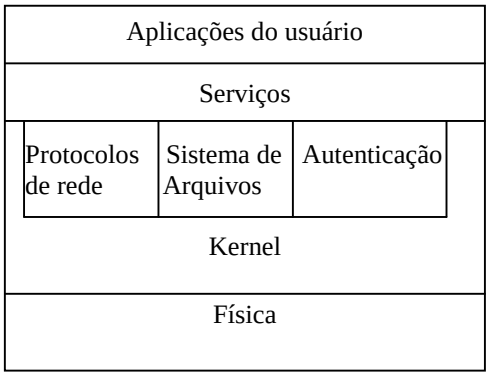

Figura 1: O modelo em camadas de segurança

Os ataques acontecem onde existem vulnerabilidades. As mesmas podem ser classificadas como de *host* ou de rede. São comuns as de *host* provenientes:

- do projeto de um sistema;
- da implementação do projeto em software ou hardware;

- do erro de configuração de um sistema.

Uma vulnerabilidade de rede pode ser entendida como aquela que está relacionada ao tráfego de informação. Ataques podem também ocorrer através de falhas atribuídas a administradores, ou a usuários mal informados ou mal intencionados.

A segurança, embora seja um problema constante nas redes, visa garantir que o sistema se comporte como esperado [Spafford,1996]. A defesa em profundidade, que ajuda nesta segurança, é uma estratégica básica, na qual não se deve depender apenas de um mecanismo de segurança, mas sim, dispor de múltiplos mecanismos de segurança, alocando um à retaguarda do outro.

Existem diversos mecanismos para abordar este problema, dentre eles: *firewalls*, análise de vulnerabilidades, criptografia, certificados digitais, *VPNs (Virtual Private Network)*, detecção de intrusão e, até, uma política de segurança bem planejada e aplicada. Todos são utilizados de forma que haja uma maior chance de se evitar comprometimentos.

A Detecção de Intrusão, como um destes mecanismos, pode ser definida como o problema da identificação de tentativas ou o uso do sistema de computação sem autorização e, quando com legítimo acesso ao sistema, o abuso de privilégios, [Spafford, 1998a]. É também definida como sendo qualquer cenário com ações que tentem comprometer a integridade, a confidencialidade e a disponibilidade dos recursos [Heady, 1990], neglicenciando o sucesso ou a falha destas ações [Zamboni, 2000].

Uma metodologia de detecção de intrusão tem, como objetivo, detectar atividades que violem a Política de Segurança, ou que comprometam a segurança do sistema. Em termos mais simples, a detecção de intrusão pode ser formada de três componentes funcionais [Bace, 2000]:

- A informação de origem, que fornece um fluxo de gravação de eventos.
- A análise, que detecta sinais de intrusão.
- A resposta, que gera reações baseadas na saída da análise.

Um sistema de detecção de intrusão (IDS), como um sistema de segunda linha de defesa não é, *per se*, uma defesa no sentido usual, mas visa monitorar o estado da rede e de

seus *hosts*, supondo-se que o atacante já invadiu a rede. Grosso modo, trata-se de um alarme [Hora, 1999]. O mesmo pode ser configurado para informar sobre um evento observado que pode ser um ataque, a suposta origem de um ataque, bem como sugerir algum tipo de ação reativa.

1.2 Estado da Arte

A área de detecção de intrusão é muito dinâmica, em constante atualização e, com o surgimento de novos IDS, sempre em busca do aprimoramento. Diante disto é grande a importância das taxonomias e *survey* nesta área.
Uma taxonomia serve para muitas finalidades [Flood e Carson, 93] como por exemplo:

- Descrição: ajuda a descrever os sistemas para melhor compreendê-los;
- Predição: onde a classificação aponta numa direção para futuros empreendimentos pesquisados;
- Explanação: fornece indícios de como explicar fenômenos observados.

Um *survey* pode se basear numa taxonomia para descrever os sistemas e oferece condição de:

- Incluir novos IDS na comparação;
- Melhor entender os IDS comparados;
- A partir deles, levantar definições, requisitos ou características.

[Axelsson, 2000], em seu estudo, apresenta uma taxonomia que consiste, em primeiro lugar, de uma classificação dos princípios de detecção e, em segundo, de uma classificação de certos aspectos operacionais dos IDS.

A literatura mostra muitas sugestões e implementações de detectores de intrusão, entretanto, através da documentação disponibilizada é difícil classificar o princípio de operação usado no detector, porque não está freqüentemente explícito nesta documentação. A motivação do trabalho de Axelsson, em se aprofundar na abordagem dos diferentes princípios de operação dos detectores empregados, torna-se lógica se admitido que diferentes princípios de detecção se comportam diferentemente sob a mesma circunstância.

Segundo [Mukherjee, Heberlein e Levitt, 1994], os métodos de detecção podem ser categorizados como:

- *Detecção de anomalia,* que tenta identificar, na rede, o desvio de comportamento e sintomas que divirjam de uma rede em uso normal;
- *Detecção de mau uso,* que procura identificar padrões de comportamento que coincidam com cenários conhecidos de ataques.

Na taxonomia de Axelsson, a *detecção de anomalia* é categorizada em:

- *Sistema auto-aprendizagem,* que aprende, através de exemplos, o que se constitui como "normal":
    - *Série não temporal,* que modela o comportamento normal do sistema pelo uso de um modelo estocástico, que não leva em conta o comportamento temporal;
    - *Série temporal,* que leva em conta o comportamento de séries de tempo.
- *Sistema programado,* baseado em técnicas[1] de representação do conhecimento na expectativa de detectar eventos anômalos:
    - *Estatística descritiva;*
    - *Default negar*.

Uma abordagem, na taxonomia de Axelsson, é a *detecção de mau uso,* que apresenta somente a categoria *programada.* É classificada nas seguintes técnicas[2]:

- *Modelagem de estado;*
- *Sistema especialista;*
- *Casamento de string;*
- *Baseado em regra simples.*

Assinatura inspirada, definida como *auto-aprendizagem,* são detectores compostos, que aparecem na taxonomia. Aprendem, automaticamente, o que constitui comportamento intrusivo e normal para um sistema. A técnica é descrita como *seleção de característica automática.*

[1] Técnicas descritas no capítulo 3, seção 3.2

[2] Técnicas descritas no capítulo 3, seção 3.2.

Axelsson também classifica os IDS em relação a 08 (oito) características. São identificadas como:

- Tempo de detecção;
- Granularidade do processamento dos dados;
- Origem dos dados auditados;
- Resposta à detecção de intrusão;
- Localização da coleta de dados;
- Localização do processamento dos dados;
- Segurança;
- Grau de interoperabilidade.

Outra taxonomia [Debar, 1999] também importante, classifica os IDS com relação a 04 (quatro) características principais, tanto funcionais quanto não funcionais. São elas:

- O método de detecção;
- O comportamento da detecção;
- A localização da origem da auditoria;
- A freqüência de uso.

O *survey* de [Lunt, 1988] descreve poucos IDS e suas respectivas técnicas de análise do *Audit Trail*. As características dos IDS apresentadas no seu trabalho já estão incluídas nas taxonomias de Debar e Axelsson, por isso, não serão citadas separadamente.

Algumas conclusões foram identificadas nas taxonomias e *surveys* estudados, são elas:

- nenhuma das abordagens de detecção de intrusão é sozinha suficiente, cada uma trata de ameaças diferentes [Lunt, 1988].
- o foco corrente de protótipos pesquisados, bem como de produtos é dirigido para a proteção da infraestrutura, mais do que da estação do usuário final. Este paradigma introduziu o uso de *sniffers* para a análise de pacotes. A análise de assinaturas é predominante nos sistemas comerciais, entretanto é insuficiente para detectar todos os tipos de ataque. Uma abordagem que trabalhe com os dois

métodos de detecção está sendo pesquisada. A detecção de ataque do tipo abuso de privilégio é aquela que necessita de maior atenção [Debar, 1999].

- IDS com características diversas podem utilizar a mesma tecnologia para diferentes implementações. Detectores podem trabalhar em camadas ou como detectores em separado [Axelsson, 2000].

1.3 Motivação e Objetivo

Os riscos crescentes de ataques a sistemas computacionais advindos do aumento das atividades não autorizadas, não somente de atacantes externos mas, também, de internos (como no caso de empregados descontentes e usuários abusando de seus privilégios) mostra a importância dos IDS no aumento da segurança dos sistemas computacionais. As organizações são alvos freqüentes de numerosos tipos de ataque de diversas origens. De acordo com pesquisa da CSI/FBI/2002[3], 42% das companhias levantadas (321 respondentes) disseram que tiveram de um a cinco incidentes de segurança. Adicionalmente, 5% delas relataram a ocorrência de 60 ou mais incidentes.

1.3.1 Motivação

O grande número de Sistemas de Detecção de Intrusão (IDS) desenvolvidos e disponibilizados atualmente, tanto comercialmente quanto para pesquisa, bem como as muitas questões que envolvem os mesmos, trazem consigo problemas e restrições que precisam ser identificados para possíveis aprimoramentos. Apesar desses esforços, muitos daqueles problemas ainda persistem nos IDS existentes:

- IDS baseados em conhecimento são inadequados para detectar todos os tipos de ataque. O aparecimento de novos ataques e vulnerabilidades no sistema leva à necessidade do emprego de IDS baseados em outros métodos de detecção.
- O emprego de IDS baseados em comportamento leva à alta taxa de alarmes falsos na detecção como resultado de eventuais mudanças no comportamento normal do usuário.

[3] 2002 Computer Crime and Security Survey – Computer Security Institute (Relatório do ano de 2002 de crimes de computadores e levantamentos de segurança) Disponível em: http://www.gocsi.com/forms/fbi/pdf.html.

- O uso de sistemas especialistas no mecanismo de detecção de intrusão é a abordagem mais freqüentemente aceita para detecção de ataques [Cannady and Harrell, 1996]. Tem sido largamente incorporada em muitos IDS, principalmente nos comerciais. No entanto, por usarem um modelo computacional de raciocínio de um especialista humano, devem se submeter a uma contínua manutenção.
- IDS que tem seu princípio de detecção baseado em regras, geralmente sofre com a falta de flexibilidade na representação das regras e também com a inabilidade em detectar ataques que ocorrem durante longo período de tempo e cenários de intrusão no qual múltiplos atacantes operam de maneira coordenada.

1.3.2 Objetivo

Esta dissertação tem como objetivo apresentar uma nova taxonomia baseada em [Debar, 1999] e [Axelsson, 2000] com as características destacadas a seguir:

Em relação à coleta de dados:
- Localização da informação de origem;
- Mecanismo de coleta;
- Arquitetura de coleta dos dados.

Em relação à análise de dados:
- Método de detecção;
- Aprendizagem da análise;
- Técnica utilizada no método;
- Arquitetura de análise dos dados.

Em relação ao componente de resposta:
- Tipo de resposta.

A elaboração de um *survey* dos IDS também é um dos objetivos desta pesquisa. Este estudo tem como finalidade, a descrição dos sistemas diante de suas características já definidas na nova taxonomia. A escolha dos IDS se baseou na diversidade das tecnologias empregadas em cada um deles.

Enfatizou-se, também, a importância de serem pesquisados trabalhos anteriores, para melhor entender suas limitações e lacunas, visando colher informações para o desenvolvimento de novas tecnologias.

Como objetivo final é apresentado um modelo, que contempla uma nova abordagem para os Sistemas de Detecção de Intrusão.

1.4 Metodologia

A metodologia desta dissertação envolve, como será visto, a apresentação de duas taxonomias que dão subsídios a uma nova taxonomia de IDS, que por sua vez apóia-se numa teoria que pode sofrer modificações mais tarde, o que é provável, tendo em vista ser uma área muito dinâmica. Mesmo assim, será possível retornar à taxonomia e adicionar-lhe características.

Diante das características apresentadas na nova taxonomia proposta, a comparação das taxonomias conhecidas e a apresentação de uma nova proposta envolvem, também, a construção de um *survey* de IDS. Este método de pesquisa possibilita, caso haja alteração nas características da taxonomia, retornar ao conjunto descrito e reanalisá-lo sob a nova perspectiva teórica. Esta nova análise não poderia ser realizada tão facilmente no caso de se usar outro método de pesquisa, como por exemplo, o estudo de caso.

A definição de critérios e a identificação das características de um IDS na nova taxonomia permitem que seja proposto um modelo de IDS. Prevendo-se possíveis alterações, este modelo é construído com características de flexibilidade, escalabilidade e extensibilidade para aceitar, facilmente, o acréscimo de funcionalidades à arquitetura.

Como parte prática desta dissertação, foi escolhido para teste o protótipo AAFID2[4]. As características apresentadas por este protótipo se assemelham com as utilizadas no modelo proposto nesta dissertação, como:

[4] Autonomous Agent for Intrusion Detection – versão 2, desenvolvido por Eugene H. Spafford, e Diego Zamboni, no projeto COAST da Purdue University em West Lafayette e disponibilizado para teste em 07/09/1999.

- uma arquitetura de coleta e análise de dados distribuída;
- coleta de dados de múltiplas localizações de origem;
- escalabilidade com o uso de agentes de software, independentes e hierárquicos.

Como algumas características do IDS são derivadas de sua arquitetura, a esta deve ser atribuído o mesmo grau de importância que aos componentes do IDS, citados por [Bace, 2000].

O modelo proposto precisa, numa fase posterior que contemple sua implementação, de atender a 10 (dez) requisitos básicos desejáveis para um IDS, segundo [Barrus, 1997]:

- Um IDS deve reconhecer qualquer atividade suspeita ou evento que possa ser um ataque conhecido;
- Deve ser detectado, no nível mais baixo possível, o comportamento de escalação de privilégios por parte do intruso;
- Componentes em vários *hosts* devem se comunicar a respeito dos níveis de alerta e de detecção de intrusão;
- O sistema deve responder apropriadamente a mudanças no nível de alerta;
- O IDS deve ter algum mecanismo de controle manual para permitir que administradores controlem as várias funções e níveis de alerta do sistema;
- O sistema deve ser capaz de se adaptar a mudanças nos métodos de ataque;
- O sistema deve ser capaz de tratar múltiplos ataques concorrentes;
- O sistema deve ser escalável e expansível como as mudanças da rede;
- O sistema deve ser resistente a comprometimento;
- O sistema deve ser eficiente e confiável.

1.5 Organização

Os capítulos que se seguem serão apresentados da seguinte forma:

O Capítulo 2 descreve conceitos, métodos e componentes funcionais de um sistema típico de detecção de intrusão, suas características e limitações atuais, com o objetivo de apresentar um contexto da detecção de intrusão no problema de segurança computacional.

O Capítulo 3 discute as taxonomias conhecidas com uma visão crítica e comparativa, objetivando identificar vantagens e deficiências. Apresenta uma proposta de uma nova taxonomia a partir das taxonomias estudadas e, também, 3 (três) tabelas classificatórias de 20 IDS, segundo as taxonomias de Debar, Axelsson e a taxonomia proposta.

O Capítulo 4 propõe um modelo de IDS que contempla uma nova abordagem, considerando-se as características apresentadas na nova taxonomia proposta e o *survey* de IDS.

O Capítulo 5 apresenta as conclusões, as principais contribuições desta dissertação e as propostas de trabalhos futuros.

O Apêndice A apresenta o survey de 20 (vinte) Sistemas de Detecção de Intrusão.

O Apêndice B apresenta o diagrama de classes da arquitetura de um IDS mostrada no Capítulo 4.

Capítulo 2

Princípios da Detecção de Intrusão

Este capítulo apresenta conceitos básicos sobre Detecção de Intrusão (ID), como seus princípios, *modus operandi*, finalidade, preocupações, vantagens e limitações.

2.1 Introdução

A Detecção de Intrusão, como um mecanismo de segurança, pode ser definida como o problema da identificação de tentativas ou o uso do sistema de computação sem autorização e, quando com legítimo acesso ao sistema, o abuso de privilégios existentes, [Spafford, 1998a]. É também definida como sendo qualquer cenário com ações que tentem comprometer a integridade, a confidencialidade e a disponibilidade dos recursos [Heady, 1990], negligenciando o sucesso ou a falha destas ações [Zamboni, 2000].

Respeitando-se os aspectos de segurança da informação, a obtenção, manutenção e disseminação da informação são necessárias. Para tanto, seu armazenamento e distribuição devem ser protegidos. Neste contexto, e dependendo dos serviços mantidos,

são classificadas as propriedades relacionadas com a segurança da informação em computadores e redes [Stallings, 1998]:

- **Confidencialidade** dos dados: está relacionada com a proteção dos dados contra acesso por usuários não autorizados.
- **Autenticação** de origem e destino: relaciona-se com o impedimento de personificação de usuários e de *hosts*.
- **Integridade**: está relacionada com a garantia de que a informação não sofra duplicação, alteração, inserção ou reordenação no seu conteúdo.
- **Não repudiação***:* está relacionada com o impedimento de usuários negarem ações executadas ou mensagens enviadas/recebidas.
- **Controle de acesso***:* relaciona-se à exigência de que o acesso aos recursos de computação possa ser controlado pelo sistema alvo.
- **Disponibilidade***:* relaciona-se à exigência de que sistemas de computação devem estar disponíveis para partes autorizadas quando necessário.

2.2 Diferença entre Ferramenta de Segurança e Sistema de Detecção de Intrusão

Pode-se identificar dois grupos que desempenham a detecção de intrusão, dando ênfase ao monitoramento das ações que ocorrem no sistema alvo:

- O primeiro engloba aqueles que tentam de modo contínuo desempenhar sua análise. Pode ser feita dinamicamente, onde a informação de origem é transportada para a análise à medida que os eventos acontecem, sendo processada imediatamente. Ou em lotes, revisando arquivos de auditoria ou pacotes de rede, acumulados num período de tempo particular. São identificados como Sistemas de Detecção de Intrusão – IDS.

- E o segundo grupo, aqueles que periodicamente fazem uma captura instantânea do ambiente e analisam as informações, procurando por vulnerabilidades de software, erros de configuração, etc. Estas ferramentas avaliam o estado do

sistema para um dado momento. Exemplos incluem, COPS, *Tiger*, AIDE e a ferramenta *Tripwire*, que checam a integridade do sistema de arquivos. Embora possam detectar intrusões, não serão consideradas IDS por somente analisarem periodicamente o ambiente, procurando por vulnerabilidades, erros de configuração, etc, em oposição ao monitoramento contínuo feito pelos IDS. São identificadas como ferramentas de segurança.

O uso do termo "tempo real" no contexto de controle do processo se refere a um processo que é suficientemente rápido para permitir que o resultado do IDS afete o progresso ou a conseqüência de um ataque em andamento.

2.3 *Modus operandi* da detecção de intrusão

A ID é desempenhada por meio de atividades de monitoração (coleta e análise de dados/eventos) que estão ocorrendo em um sistema alvo em busca de sinais de intrusão. Um sistema alvo pode ser um computador ou rede, onde atividades e comportamentos estão sendo examinados a partir de uma perspectiva de detecção de intrusão. O sistema alvo deve apresentar vantagens dignas que levem ao monitoramento antes que qualquer IDS possa ser considerado. Neste contexto, a ID pode utilizar mecanismos que tornem automático o processo de monitoração de eventos.

Uma metodologia pode, como objetivo, detectar atividades que violem a Política de Segurança ou que comprometam a segurança do sistema. Em termos mais simples, a detecção de intrusão é formada por 03 (três) componentes funcionais [Bace, 2000], mostrados na figura 3:

- A informação de origem, que fornece um fluxo de gravação de eventos. Este componente é identificado como coleta de dados;
- A análise de dados, que detecta sinais de intrusão;
- E um componente de resposta que gera reações baseadas na saída da análise de dados, identificado como resposta.

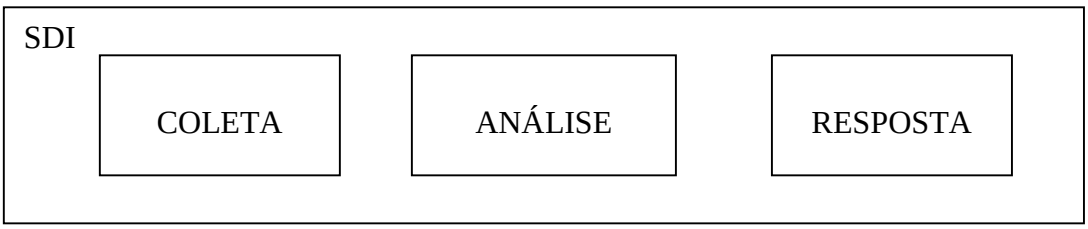

Figura 2: Componentes funcionais clássicos de um IDS.

2.3.1 Coleta de Dados

A coleta de dados representa a interface de entrada primária entre o sistema-alvo e o IDS, pois sem um meio de extrair a informação do sistema alvo, a ID não pode ser desempenhada. As duas abordagens comumente aceitas para produzir a informação coletada do sistema alvo são:

1. *Packet feed*. Esta abordagem utiliza pacotes do tráfego de informação para um *host* ou para um segmento de rede ou toda a rede;
2. *Audit trail*. Esta abordagem utiliza dados de auditoria localizados no *host*, como *logs* de atividade de *login*, do *kernel*, de aplicações, etc.

O local da coleta da informação de origem apresenta vantagens quanto à abordagem adotada.

1. Vantagens da abordagem *packet feed* que tornam clara a necessidade de IDS deste tipo em qualquer política e sistema de segurança:

    - Pode detectar ataques que a abordagem *audit trail* não consegue, pois examina o cabeçalho de pacotes a procura de sinais maliciosos e atividades suspeitas; investiga o conteúdo do *payload* em busca de comandos usados em ataques específicos. Exemplos de ataque: IP *spoofing*, a falsificação de conteúdo de pacotes, a fragmentação de pacotes (*TearDrop*).
    - Pode detectar e monitorar atividades suspeitas em portas conhecidas, como a porta TCP 80, que é utilizada pelo HTTP.
    - Pode dificultar a remoção de evidências porque utiliza a detecção em tempo de execução;

- Fornece rápida notificação e resposta, podendo detectar ataques em tempo de execução;
- Pode ser empregada sem afetar substancialmente o desempenho da rede;
- Não precisa ser instalada em todos os *hosts*;
- Opera de forma independente do sistema operacional dos *hosts* monitorados;
- Pode realizar a detecção de intenção maliciosa para recursos atrás do *Firewall*, mesmo que este possa rejeitar essas tentativas de ataque, sendo, portanto, conhecida a freqüência e os tipos de ataque feitos à rede.
- Pode complementar e verificar componentes implementados pela política de segurança. No caso do *Firewall*, pode ajudar a verificar se ele realmente impede certos tipos de tráfego e endereços que devam ser rejeitados.
- Como permite o emprego estratégico a pontos de acesso crítico, para obter uma visão do tráfego destinado a numerosos sistemas que necessitam ser protegidos (considerando a abordagem para um segmento de rede ou toda a rede), não exigem o carregamento e o gerenciamento de um programa em vários *hosts*. Portanto, menos pontos de detecção são requeridos, melhorando a relação custo-benefício do gerenciamento em um ambiente empresarial.

2. Algumas vantagens da abordagem *audit trail* que não são encontradas na abordagem anterior:

   - Forte análise. Desde que use registros contendo eventos que tenham realmente ocorrido, pode informar se um ataque obteve sucesso ou não, com poucos falsos-positivos;
   - Monitora atividades específicas. Por exemplo, monitora atividades do usuário, acessos a arquivos de sistema e executáveis, tentativas de acesso a serviços privilegiados e de instalação de *trojan horses* ou *backdoors*. Monitora, também, atividades que devam e possam ser executadas somente por administradores.
   - Monitoram componentes-chaves. Alerta quando arquivos são executados ou modificados (*.rhosts, /etc/passwd*), como também, alerta sobre o nível de uso do espaço em disco.

- Pode detectar ataques que ocorrem fisicamente no servidor (*keystroke attack*).
- Adapta-se bem a ambientes com *switches* e sob criptografia, pois reside em vários *hosts*. Mesmo que os *switches* permitam que o ambiente seja gerenciado como muitos segmentos de rede e que exista criptografia em algum lugar da pilha de protocolos, estes não são fatores limitantes, pois o fluxo de dados, no tráfego de entrada, já chega decifrado.
- Não requer uma plataforma de hardware dedicada. Pode residir em recursos de rede existentes, servidores de arquivos, servidores de *Web* e em outros recursos compartilhados e críticos. Nestes casos, tem um custo mais efetivo, não sendo apenas outra "caixa" instalada na rede, a exigir endereçamento, manutenção e gerenciamento.

Cada tipo apresenta vantagens e desvantagens. Uma boa estratégia é utilizar, ao mesmo tempo, as duas abordagens num IDS, *audit trail e packet feed*, pois ocorrem ataques que podem ser detectados por uma abordagem e não por outra.

2.3.2 Análise de dados

É um componente complexo e pode incluir mais de um componente de análise de dados. Eles podem ser autônomos ou dependentes e são referidos como filtros em alguns sistemas. A disposição dos componentes é uma característica relevante e pode ser:

- um modelo em multicamadas de detecção, onde os níveis mais baixos alimentam os níveis mais altos.
- componentes distintos que processam os dados recebidos. O resultado é enviado para outros componentes, que coordenam os relatórios e iniciam atividades de resposta específica.

Este componente deve possuir a função de correlação para combinar resultados dos processamentos dos componentes individuais.

IDS que utilizam mais de um analisador paralelamente, não representam aumento no nível de segurança. Apenas trazem maior flexibilidade ao permitir a utilização de mais

de um tipo de analisador. No entanto, IDS que utilizam diferentes analisadores em série trazem benefícios para o nível de segurança da rede da organização.

Este componente pode decidir quando os eventos descritos indicam se intrusões estão em curso ou, se já ocorreram. As abordagens de análise de dados mais comuns são duas:

- a que dirige seu foco às atividades que fogem significativamente dos perfis estabelecidos de comportamento normal.
- a que procura por eventos ou conjunto de eventos, que se encaixam em um padrão pré-definido, que descrevam um ataque conhecido;

As duas abordagens de análise apresentam vantagens e desvantagens.

1. Quanto à primeira abordagem:

    a. Vantagens:
        - Detecta tentativas de serem exploradas vulnerabilidades novas e imprevistas, podendo mesmo contribuir para serem descobertos, automaticamente, pelo menos parcialmente, novos ataques;
        - É menos dependente de mecanismos específicos de sistemas operacionais;
        - Ajuda a detectar abusos de privilégio, que são tipos de ataque que não envolvem a exploração de qualquer vulnerabilidade de segurança.

    b. Desvantagens:
        - Alta taxa de alarmes falsos, uma vez que todo o espaço do sistema de informação não pode ser coberto durante a fase de aprendizagem. Também o comportamento pode mudar com o tempo, introduzindo a necessidade de instruções on-line periódicas de perfil normal de comportamento, resultando tanto em indisponibilidade do IDS, quanto em adicionais alarmes falsos;
        - O sistema pode estar sendo submetido a ataques na mesma hora em que o IDS estiver aprendendo. Como resultado, o perfil normal de

comportamento contém comportamento intrusivo, que não será detectado como anômalo;

- Dependendo do tamanho do *audit trail* e da habilidade de processamento do sistema, a revisão dos dados de auditoria resulta na perda da capacidade de análise em tempo real.

2. Quanto à segunda abordagem:

   a. Vantagens:
      - Baixa taxa de alarmes falsos em relação à abordagem baseada em comportamento;
      - A análise contextual proposta é detalhada, facilitando o uso deste sistema pelo analista de segurança para ação de prevenção e correção.

   b. Desvantagens:
      - Necessidade de atualização freqüente da base de conhecimento com as novas vulnerabilidades descobertas. É uma tarefa de manutenção que exige análise cuidadosa de cada vulnerabilidade, além do consumo de tempo. Esta situação se agrava em razão da exigência de serem representadas todas as possíveis facetas de um ataque, como assinaturas. Existe a possibilidade de um ataque ser representado por um número de assinaturas, no mínimo uma para cada tipo de sistema operacional, no qual o IDS deve ser portado;
      - Degradação da performance resultante da dependência na entrada do *audit trails*. Esta desvantagem pode ser minimizada aumentando-se a performance do sistema e reduzindo-se as gravações de auditoria.

A combinação das duas abordagens permite a monitoração por indicações de ataques que podem ser detectados por uma abordagem e não por outra. A desvantagem seria o uso de duas bases de conhecimento no IDS, o que leva ao crescimento da quantidade de recursos do sistema que deve ser dedicado ao IDS. Também espaço adicional em disco será necessário para armazenamento de perfil normal do usuário/sistema, bem como, a

exigência de aumento da memória como mecanismo para comparação das atividades do usuário com as informações nas duas bases de conhecimento.

O objetivo de um IDS é detectar cenários ou sintomas de intrusão. Não parece ser possível, para um sistema, ter conhecimento de todos os ataques possíveis, nem do comportamento normal dos usuários e do sistema todo o tempo. Isto representa situações ideais. No entanto, é desejável que os IDS tenham condição de adquirir conhecimento.

A aquisição do conhecimento aparece na análise de dados modificando as abordagens da seguinte forma:

- IDS baseado na primeira abordagem pode adquirir o conhecimento da seguinte maneira:
    - Automática, que aprende, através de exemplos, o que se constitui como “normal”. Estes IDS podem ter sua base de dados atualizada com o novo comportamento do usuário/sistema de maneira periódica e automática por natureza.
    - Programada, que necessita da adição na sua base de conhecimento, a partir de uma origem externa, de perfil normal de comportamento do usuário/sistema.

- IDS baseado na segunda abordagem pode adquirir o conhecimento da seguinte maneira:
    - Automática, que é um método de detecção de grande utilidade para descobrir leves variações nas assinaturas de ataques, permitindo a descoberta de ataques não detectados por mecanismo de aprendizagem programada.
    - Programada, onde, para que novas regras e assinaturas sejam acrescentadas na base de dados do IDS, primeiramente as mesmas são construídas off-line, manualmente e quando se tornam conhecidos novos tipos de vulnerabilidades e intrusões.

As primeiras pesquisas em detecção de intrusão reconhecem a ineficiência de qualquer abordagem que necessite de revisão manual dos sistemas de auditoria. A determinação de como a informação coletada será revisada é um elemento importante na estrutura básica de um IDS. Das muitas técnicas apresentadas para revisão da informação, algumas conclusões são tiradas da aplicação das mesmas:

- Quanto à análise baseada em regras:

É uma abordagem de defesa estruturada e demonstra ser relativamente eficiente se as características exatas de um ataque são conhecidas. A desvantagem reside na falta de flexibilidade na representação das regras. Leves variações na seqüência de um ataque podem ocasionar alterações que não serão detectadas por este mecanismo. Ainda que cresça o nível de abstração na regra, somente fornece uma solução parcial para o problema. Por outro lado, reduz a granularidade do detector.

Quando esta técnica é associada à estatística tem a habilidade de detectar padrões de intrusão não antecipados, tendo em vista que os padrões específicos não têm que ser pré-definidos. A técnica procura elementos individuais, que podem ser parte de uma intrusão sem que se confie na conclusão de uma seqüência inteira de atividade específica. A desvantagem é sua incapacidade de, rapidamente, adaptar-se a mudanças legítimas no comportamento do usuário.

Análises estatísticas e baseadas em regras sofrem com a inabilidade de detectar ataques que podem ocorrer durante um longo período de tempo. Enquanto instâncias individuais de atividades suspeitas podem ser detectadas pelo sistema, as mesmas não podem ser relatadas se ocorrerem de forma isolada. Cenários de intrusão, nos quais múltiplos atacantes operam de maneira coordenada, são também difíceis para a detecção, porque não visam a transição de estado de um ataque. Em vez disto, concentram-se na ocorrência de elementos individuais. Qualquer distribuição de um ataque, tanto temporal quanto entre atacantes, aparentemente sem um relacionamento, apresenta dificuldade para detecção por estas técnicas. No entanto, técnicas de detecção mais correntes no processo de detecção de intrusão usam alguma forma de análise estatística ou baseada em regras.

Quando as regras são modeladas como transição de estado trazem grande flexibilidade e velocidade. A técnica também conserva estatísticas sobre o desempenho da rede, utilizando um cálculo estatístico para determinar quanto tempo a informação de estado da conexão deve ser retida antes de descartá-la.

Sistema especialista (SE) é um princípio de detecção de considerável poder e flexibilidade em relação aos sistemas baseados em regras simples, permitindo mecanismos como a unificação. O uso de técnica de sistema especialista no mecanismo de detecção de intrusão foi um marco significativo no desenvolvimento de sistemas de segurança da informação para detecção. Este princípio de detecção, explorado como análise de assinaturas de ataque, tem sido largamente incorporado em muitos IDS, principalmente nos comerciais. No entanto, por usarem um modelo computacional de raciocínio de um especialista humano, devem se submeter a uma contínua manutenção para um bom desempenho. Esta técnica de detecção usa um conjunto de regras conhecidas (programadas) e pode incorporar um componente estatístico ou probabilístico. Contudo, pode ser menos eficiente em velocidade de execução que a abordagem estatística para processamento de grande quantidade de informações de auditoria.

Já a indução baseada em regras, em contraste com o SE, automaticamente desenvolve regras para explicar, historicamente, os dados que coletam. Abordagens que usam linguagens baseadas em regras têm algumas limitações:

- Engenharia do conhecimento. É difícil extrair conhecimento sobre ataques e ainda mais difícil traduzir este conhecimento para regras.
- Velocidade do processamento. A técnica exige que toda a informação seja importada como fatos para, somente então, poder raciocinar. Mesmo que a ferramenta permita compilação de regras, a performance geral é baixa, sendo mais utilizada em protótipos.
- A falta de manutenção na freqüência de atualizações exigidas leva a um sistema com capacidades reduzidas e, pior, com um falso senso de segurança.

- Quanto à análise baseada em exemplos - Rede Neural (RN):

Outras técnicas de aprendizagem são empregadas na detecção de intrusão para melhorar a performance de busca e análise de dados nos sistemas computacionais, como a redução de dados e os sistemas classificadores. Exemplos de sistemas classificadores são a árvore de decisão e a rede neural [Frank, 1994]. Dentre as vantagens de se empregar uma rede neural na detecção de intrusão está a habilidade de aprender as características de um ataque de mau uso do sistema e, identificar instâncias que sejam diferentes de qualquer uma que tenha sido observada antes pela rede. É um sistema de defesa com capacidade flexível de reconhecimento de padrão. Similarmente, a RN possui a habilidade de conduzir a análise dos dados de maneira não linear, possibilitando o processamento dos dados de múltiplas origens.

Uma RN pode ser treinada para reconhecer eventos suspeitos conhecidos com um alto grau de precisão. Também deve ganhar a habilidade de aplicar este conhecimento para identificar instâncias de ataque que não casam com as características exatas de intrusões prévias. A probabilidade de um ataque contra um sistema pode ser estimada e uma ameaça em potencial ser sinalizada sempre que a probabilidade exceder a uma entrada especificada. Desvantagens também aparecem quanto à dependência da RN no treinamento dos dados e nos métodos de treinamento que são usados pelo sistema e, também, na obtenção dos dados para treinamento. Contudo, a mais significante desvantagem da aplicação da RN para ID é a natureza “caixa preta”. Enquanto um SE tem regras altamente codificadas para análise de eventos, as RNs adaptam sua análise de dados, em resposta ao treinamento que é conduzido na rede.

Diferentemente de SE, que podem fornecer ao usuário uma resposta definitiva, se as características que são revisadas casam exatamente com aquelas que foram codificadas na base de regra, a RN conduz a uma análise da informação e fornece uma estimativa da probabilidade do dado casar com as características que a RN foi treinada para reconhecer. Enquanto a probabilidade de um casamento determinado pela RN pode atingir 100 %, a precisão das decisões confia totalmente na experiência que o sistema ganha analisando exemplos do problema indicado. Enquanto que SE são capazes de reconhecer uma mudança

rápida em um sistema, a RN é útil na identificação de mudanças graduais no sistema ou, no comportamento do usuário.

A RN ganha experiência, inicialmente, pelo treinamento do sistema para corretamente identificar exemplos pré-selecionados do problema. A resposta da RN é revisada e a configuração do sistema é refinada, até a análise da RN do treinamento dos dados alcançar um nível satisfatório. Em adição ao período de treinamento inicial, a RN também ganha experiência com o tempo e da forma como conduz a análise de dados relacionada ao problema.

A solução de problemas através de RN é bastante atrativa, já que a forma como estes são representados internamente pela rede e o paralelismo natural, inerente à arquitetura das RNs, criam a possibilidade de um desempenho superior aos dos modelos convencionais.

RN tem sido proposta como alternativa do componente de análise estatística. Análise estatística envolve comparação estatística de eventos correntes para um predeterminado conjunto de critérios. A técnica é mais freqüentemente empregada em detecção de desvios de comportamento típico e determinação de similaridade de eventos com aqueles que são indicativos de um ataque. RN são especificamente propostas para identificar características típicas de usuários do sistema e identificar, estatisticamente, variações significativas do comportamento estabelecido do usuário.

Outras técnicas podem ser associadas à análise de dados:

- Algoritmo Genético (AG):

As altas taxas de alarmes falsos na detecção, como resultado de mudança no comportamento normal do usuário, levam à necessidade de uma abordagem utilizando-se técnicas de aprendizagem, para aprender o comportamento do usuário por observação contínua das interações com o sistema computacional [Balajinath, 2000]. O comportamento corrente do usuário pode ser prognosticado pelo uso de algoritmos genéticos, baseados na observação passada

do seu comportamento. Embora simplista do ponto de vista biológico, estes algoritmos são suficientemente complexos para fornecer mecanismos de adaptação robustos e poderosos. A vantagem dos algoritmos genéticos sobre as redes neurais é que as regras de diagnóstico que geram são de mais fácil entendimento para as pessoas.

- Imunologia Computacional:

Esta técnica constrói um modelo de comportamento normal de serviços de rede UNIX, ao invés de modelo de usuário. Este modelo consiste de curtas seqüências de chamadas ao sistema feitas pelos processos. Em primeiro lugar, a ferramenta coleta um conjunto de auditoria de referência, que representa o comportamento normal do serviço e extrai uma tabela de referência contendo todas as boas seqüências de chamadas ao sistema conhecidas. Esses padrões são, então, usados para checar se as seqüências geradas são listadas na tabela. Se não, o IDS gera um alarme. Esta técnica tem uma baixa taxa de alarmes falsos se a tabela de referência é suficientemente exaustiva. Uma limitação é que esta técnica não protege contra erros de configuração no serviço, como, por exemplo, quando ataques usam ações legítimas do serviço para ganhar acesso não autorizado.

É uma abordagem de aprendizagem através de exemplos de classificação. Seu diagnóstico é baseado em dados empíricos. Deste modo, é mais sensível o significado dos dados em si do que a interpretação humana sobre eles. Outras questões:

- Este método exige estar altamente afinado para operar com eficiência máxima.
- O processo de aprendizado pode ser computacionalmente caro, particularmente se não pode ser desempenhado de maneira incremental. Pode, também, ser necessário desaprender velhas experiências. Isto introduz complexidade no retreinamento.

- Uma grande quantidade de dados é exigida para treinar estes sistemas. O problema torna-se mais complicado uma vez que o estado de "normal" depende de tempo e espaço.
- Exigem pouca comunicação entre os *hosts*, embora o IDS seja altamente distribuído [Hofmeyr, 1999].

Sistemas que utilizam algoritmos de aprendizagem, como por exemplo, imunologia computacional, RN, AG são:

- De pesquisa interessante, pois se acredita que possam suportar reconhecimento de intrusões que não tenham sido vistas previamente e, que não tenham prévios padrões descritos;
- A clareza das regras para serem entendidas por humanos não é uma característica forte;
- São computacionalmente eficientes;
- Exigem pouco armazenamento pois, se comparados a técnicas que não utilizam esta tecnologia de compressão, fornecem meios para realizá-la;
- Podem adaptar-se a novos dados;
- Não são largamente empregados em IDS.

- Mineração de Dados:

O maior desafio para a utilização da técnica de mineração de dados na detecção de intrusão é a imensa quantidade de dados de auditoria necessários para computar os perfis de uso do sistema. Como o processo de aprendizado (mineração) é caro em termos de tempo e de armazenamento e, como o processo de detecção em tempo real precisa ser leve e rápido para ser utilizável, seria impossível haver um sistema de detecção de intrusão monolítico. A técnica foi descrita como detectores compostos, mas existe proposta de ser feita baseada em comportamento e em conhecimento. Usando módulos sensores de sistemas como o SNORT[5], instalado estrategicamente em uma rede e técnicas específicas,

[5]Lightweight Intrusion Detection for Network, http://www.snort.org.

como modelos de busca que possam automatizar o processo, de modo a fornecer primitivas ou funções que, recebendo como entrada, dados específicos do tipo de análise desejada, realizem busca sobre a base de dados, selecionem seqüência de dados relevantes e, aplicando técnicas específicas, retornem um conjunto de dados que caracterizem o tipo de evento desejado. Deve-se levar em consideração, também, se a filtragem de dados pelos sensores não causa uma falta de dados para o componente análise.

- *Clustering*:

É uma técnica de aprendizagem que, através da redução de dados, ajuda na performance e análise de dados na detecção de intrusão, com mínimo treinamento individual. A filtragem de dados pode ser feita, também, utilizando-se heurísticas ou métodos *ad hoc*, que podem ser vistas como regras de especialistas. Ou ainda, filtrando-se dados de uma maneira mais adaptativa e dinâmica, como é o caso de uma RN. *Clustering* é o agrupamento de dados, de acordo com algum critério, para revelar padrões escondidos nos dados e características significativas, classificando os dados e utilizando técnicas não paramétricas[6] para desempenhar análise estatística. Diferentemente das primeiras análises estatísticas, que usavam abordagens parametrizadas para caracterizar o comportamento padrão de usuários e de outras entidades do sistema, estas se referem a abordagens analíticas, nas quais suposições são feitas sobre a distribuição dos dados analisados. Além do *clustering* estatístico, que se baseia somente em medidas numéricas de similaridade do objeto, tem-se o *clustering* conceitual, definido como o processo de construção de rede de conceitos caracterizando a coleção de objetos como nós marcados por conceitos, descrevendo classes de objetos e *links* marcados pelo relacionamento entre as classes. Atributos são usados para caracterizar estes objetos, produzindo uma classificação hierárquica de classes de objetos, onde cada nó deve formar um conceito coerente, compacto e de fácil representação em termos de definição ou regras que tenham uma interpretação natural pelo homem.

[6] Proposta por Linda Lankewicz e Mark Benard da Universidade de Tulane.

2.3.3 Mecanismo de resposta

Uma vez que as informações sobre os eventos tenham sido obtidas e analisadas para perceber ataques, podem ser geradas respostas adequadas. As respostas podem ser constituídas de ações automáticas, tomadas quando certos tipos de intrusão são detectados ou, que dependam de uma entidade externa, para a realização de ações cabíveis subseqüentes, como por exemplo, o Administrador de Sistemas (AS).

O mecanismo de resposta apresenta restrições em relação às abordagens. A existência de muitos tipos de intrusão pode levar a uma baixa probabilidade de um diagnóstico correto. Isto dificulta o IDS com ações automáticas, podendo mesmo comprometer a disponibilidade do sistema computacional. Para contornar este problema, possivelmente o IDS, que dependa de uma entidade externa, possa ser melhor empregado se for implementado adequadamente com mecanismos de visualização das respostas, para que as mesmas sejam disponibilizadas ao AS, visando a tomada de decisões por parte deste.

IDS ativos somente ganharão espaço à medida que se tornem mais confiáveis.

2.4 Qual a finalidade da ID

A ID é empregada na tentativa de aumentar a confiança e a credibilidade dos sistemas e redes, juntamente com outras estratégias de segurança computacional.

A detecção de intrusão é um foco significativo de pesquisa em segurança de sistemas computacionais e de redes. [Denning, 1987] e [Neumann, 1985] identificaram 04 (quatro) razões para se utilizar um mecanismo efetivo de detecção de intrusão como parte de um *framework* de segurança de sistemas computacionais:

- Muitos sistemas existentes têm falhas de segurança que os tornam vulneráveis, falhas estas difíceis de serem identificadas e eliminadas devido a razões econômicas e técnicas.
- Existem sistemas com falhas de segurança que não são facilmente substituídos por outros, mais seguros, devido a aplicações e considerações econômicas.

- O desenvolvimento de sistemas completamente seguros é, provavelmente, impossível.
- Mesmo os sistemas altamente seguros são vulneráveis ao mau uso por usuários legítimos.

2.5 Principais tipos de ataques

A preocupação primordial no estudo da detecção de intrusão é a percepção do ataque, ou sintoma do mesmo, em um sistema computacional. Um ataque pode obter sucesso pela presença de alguma vulnerabilidade no sistema computacional da vítima, a qual é explorada pelo atacante com um objetivo definido. Tal cenário, atualmente, possui uma enorme variedade de formas, mas, em geral, inclui-se nas seguintes categorias:

- Programa para varreduras, que mapeia uma rede com o objetivo de encontrar serviços nela disponibilizados e eventuais vulnerabilidades nestes serviços, os quais são passíveis de serem explorados.
- Programa para elevação de privilégios/invasão (*exploits*), que tenta ganhar controle local ou remoto, explorando as vulnerabilidades de um sistema.
- Negação de serviço local/remota, que é um programa que tenta parar ou prejudicar o desempenho de um sistema computacional, consumindo, excessivamente, recursos nele disponíveis.
- Programa para quebra de senhas, que tenta descobrir senhas criptografadas por meio de comparações com um banco de dados, ou de teste por força bruta.
- Programa para captura de pacotes de rede, que efetua a captura de todo o tráfego da rede à qual está conectado, utilizando interfaces de rede em modo promíscuo.

Do mesmo modo que os ataques podem ser descritos sob diferentes pontos de vista, o processo de detecção dos ataques também o pode ser. Atualmente, a tecnologia para tal processo está em constante evolução devido, principalmente, à grande velocidade com que esta área de pesquisa sofre mudanças.

2.6 Vantagens no emprego de mecanismos de Detecção de Intrusão

As vantagens de se empregar mecanismos de detecção de intrusão - ID são atestadas pela crescente popularidade no emprego dos mesmos. Segundo pesquisa[7] da CSI/FBI *Computer Crime and Security Survey*, em 2002, num universo de 500 corporações, entre governamentais, instituições financeiras, universidades, etc, o uso de mecanismos de ID chega a 60%. Outra pesquisa[8], realizada pela *Information Week*, num universo de 2.700 profissionais de segurança de 49 países, 37% deles relatou o uso de produtos de ID, contra 29% no ano anterior. Acredita-se na tendência de aumento deste percentual, uma vez que, segundo dados extraídos do *State of the Practice in Intrusion Detection Technologies* [9], CMU/SEI, 1999, todos os usuários de mecanismos de ID são capazes de encontrar algum intruso. Sendo assim, está comprovado que os sistemas realmente funcionam.

Foram identificadas, a seguir, algumas das principais vantagens de se empregar tal mecanismo [Schultz, E., 1999]:

- Redução da quantidade de recursos empregados, especialmente quanto à necessidade de recurso em pessoal para monitorar eventos que ocorrem em sistemas e rede. Em vez de aumentar as tarefas relacionadas com segurança, a tecnologia de detecção de intrusão reduz a necessidade de intervenção humana para o monitoramento de sistemas e rede.
- Capacidade de descobrir um ataque que o ser humano não consegue detectar.
- Capacidade de descobrir ataque que não pode ser detectado através de sistema de *logging* normal, principalmente por meio daqueles sistemas que correlacionam as informações recebidas de múltiplas máquinas na rede.

[7] Center for the Study of Intelligence / Federal Bureau of Investigation Disponível em: http://www.gocsi.com/forms/fbi/pdf.html.

[8] Disponível em: www.informationweek.com.br/, em julho de 1999.

[9] Disponível em: http://www.sei.cmu.edu/publications/documents/99.reports/99tr028/99tr028abstract.html.

- Resolução de problemas relacionados com vulnerabilidades de segurança nos softwares em proliferação, minimizando o estrago e a perda de informações.
- Importância dos dados produzidos por mecanismos de detecção de intrusão em evidências legais perante a lei, assunto importante na forense computacional.

2.7 Limitações no emprego de mecanismos de Detecção de Intrusão

Limitações existem e impedem que mecanismos de ID sejam largamente empregados. As principais estão identificadas abaixo:

- A maioria dos mecanismos de ID, hoje existente, ainda é relativamente rudimentar em suas capacidades. Os disponibilizados atualmente, descobrem apenas um subconjunto do total do número de intrusões que ocorrem e ainda, perdem muitas outras, por utilizarem métodos que não são capazes de detectar ataques novos. É importante entender que, às vezes, a aquisição e emprego de um ou mais mecanismos de ID não apresenta toda a capacidade anunciada.
- Para algumas empresas, continuidade operacional é a necessidade mais importante no emprego de mecanismos de ID, deixando para um segundo plano a confidencialidade dos sistemas e dados. Neste caso, um ataque de negação de serviço pode ser mais danoso e destrutivo do que outros tipos de ataques. E pode ser de difícil detecção pelo mecanismo.
- Alguns mecanismos de ID são superiores a outros. Enquanto uns utilizam, em seu projeto, requisitos de engenharia de software que trazem resultados na qualidade do método de detecção, outros já negligenciam a qualidade do projeto e da implementação.
- Mecanismos de ID não são invencíveis, pois estão sujeitos a ataques, principalmente aqueles de negação de serviço direcionado ao mesmo, ou à plataforma computacional que o abriga. Portanto, ter um grupo independente ou organização para conduzir testes nestes sistemas, constitui uma maneira de se determinar quão resistentes a ataques são estes mecanismos de ID. Conduzindo, significativamente, em direção à seleção e uso da tecnologia de ID, bem como a plataforma que o suporta.

- Nem todo ataque está relacionado ao conteúdo, entretanto, ataques à rede na forma de pacotes criptografados não serão notados por mecanismos de ID, a menos que estes estejam programados para detectar o fluxo de pacotes anormais, como numa inundação repentina de um tipo particular de pacote, mesmo que seu conteúdo esteja criptografado.

Além do custo financeiro de aquisição e emprego de mecanismos de ID, estes também envolvem o custo de manutenção. Manter um sistema em execução e atualizado é extremamente importante para sua funcionalidade e segurança. Certas tarefas precisam ser executadas, como por exemplo, *bugs* nos mecanismos de ID que precisam ter seus *patches* instalados, tratamento do volume de dados produzidos e manutenção da plataforma de funcionamento do sistema. São tarefas que, se não executadas, podem debilitar o desempenho da rede.

2.8 Influência da Arquitetura em IDS

A arquitetura de um IDS apresenta vantagens e desvantagens em relação às suas abordagens. Sistemas de detecção de intrusão centralizados podem ser caracterizados pela coleta de dados centralizada ou distribuída. Na maioria dos casos, os dados são coletados em sistemas individuais e, então, destinados a uma localização centralizada, onde a análise da detecção de intrusão pode ser desempenhada. Esta abordagem trabalha bem em situação de pequenas redes, mas é inadequada para redes maiores, devido ao grande volume de dados que deve ser analisado pelo componente de análise central.

O IDS deve ser capaz de fazer a distinção entre as muitas origens de dados, uma vez que diferentes cenários de intrusão podem ser encontrados em diferentes sistemas operacionais. Este é o problema com relação ao desempenho da análise centralizada da informação coletada a partir de sistemas heterogêneos. Abordagens centralizadas apresentam algumas necessidades em relação a abordagens distribuídas [Bace, 2000]:

- Tornar seguro o tráfego de mensagens entre os componentes do sistema, pois todos os dados devem ser transportados para o componente de análise de forma segura.

- Determinar se um elemento do sistema foi alterado ou desabilitado num dado tempo. Estes sistemas têm menos componentes, no entanto, são maiores, mais complexos e mais difíceis de serem monitorados.
- Disparar e interromper componentes do sistema "graciosamente". Se um componente de análise para de trabalhar, todo o sistema para. Cada componente é um ponto único de falha.
- Consolidar a informação e a apresentar ao usuário final de uma forma significativa.
- Manter um *host* dedicado para a tarefa de análise devido à sobrecarga imposta.
- Configurar o sistema para características específicas dos sistemas alvos monitorados.
- Escalar o sistema para um número maior de *hosts* monitorados. O componente de análise necessita de mais computação e recursos de armazenamento.
- Reconfigurar o componente de análise. Todo o sistema deve ser reiniciado, afetando o funcionamento do mesmo.

Sistema de detecção de intrusão distribuído é caracterizado pela coleta distribuída de dados, seguido pela análise distribuída da detecção de intrusão. Estes sistemas podem ser modelados como hierarquias. Diferentemente dos sistemas de detecção centralizados, apresentam maior capacidade de escalar para grandes redes, porque os componentes de análise são distribuídos, fazendo com que menos informações devam ser compartilhadas entre os diversos componentes. IDS distribuídos também apresentam requisitos:

- Manter o sistema rodando continuamente. Um grande número de componentes precisa manter-se rodando;
- Ser tolerante a falhas. É difícil armazenar de maneira recuperável e consistente;
- Controlar a sobrecarga nos *hosts* monitorados;
- Detectar mudanças no comportamento global do sistema;
- Disponibilizar a informação para o mecanismo de resposta.

Para uma abordagem distribuída, deve existir alguma maneira de repartir todo o sistema em diferentes e pequenos domínios para o propósito de comunicação. O domínio é um subconjunto da hierarquia, consistindo em um nó, responsável pela coleta e análise de todos os dados de todos os outros nós no domínio. Este nó que analisa, representa o

domínio para os outros acima dele na hierarquia. Domínios podem ser construídos pela divisão do sistema baseado em:

- geografia/topologia;
- controle administrativo;
- coleção de plataformas de software similares, ou;
- partições baseadas em tipos antecipados de intrusão.

Por exemplo, dados coletados de nós rodando o mesmo sistema operacional podem ser enviados para um ponto de coleta central. Assim, os sistemas homogêneos podem ser analisados em acordo.

2.9 Conclusão

Não existe nenhuma fórmula ou "regra de ouro" para se conseguir um retorno desejado do emprego de tecnologia de detecção de intrusão. A necessidade do seu uso é crescente, mas implica em vantagens e limitações como as que foram vistas. Também envolvem questões como finalidade e *modus operandi*, que precisam ser compreendidos e aceitos. Um maior conhecimento da área de ID será visto com a apresentação de taxonomias de IDS. No terceiro capítulo será feita uma análise crítica e comparativa de duas taxonomias de IDS, [Debar, 1999] e [Axelsson, 2000], mostrando suas limitações e propondo uma nova taxonomia de IDS baseada nos três componentes funcionais apresentados por [Bace, 2000]: a coleta de dados, a análise e o componente de resposta.

Capítulo 3

Taxonomias

Este capítulo trata da apresentação de duas taxonomias, [Debar, 1999] e [Axelsson, 2000], analisando-as e identificando suas limitações. Elas foram escolhidas por representarem o estado da arte em taxonomias de Sistemas de Detecção de Intrusão.

Também propõe uma nova taxonomia, com um novo critério de classificação do componente coleta de dados e novas características da análise de dados. A contribuição da nova taxonomia é mostrada com uma comparação, entre as três taxonomias, da abrangência de critérios/características obtida para a classificação dos componentes de um IDS.

3.1 Taxonomia de Debar

A taxonomia de [Debar, 1999] apresenta os IDS tomando por base 04 (quatro) características principais, divididas em:

1) Características funcionais. São elas:

    a) Método de detecção:
        i) Baseado em comportamento, quando a informação usada versa sobre o comportamento normal do sistema monitorado:
            (1) Estatística

(2) Sistemas Especialistas
(3) Rede Neural
(4) Identificação de Intenção do Usuário[10]
(5) Imunologia Computacional

ii) Baseado em conhecimento, se a informação usada versa sobre ataques:
(1) Sistemas Especialistas
(2) Análise de Assinaturas de Ataques
(3) Rede de Petri
(4) Análise de Transição de Estado

b) Comportamento da detecção, quando descreve a ação do componente de resposta ao ataque, podendo ser:
i) Ativa
ii) Passiva

c) Localização da origem da auditoria, que pode ser baseada em:
i) rede
ii) *host*

2) Característica não funcional:

a) Freqüência de uso:
i) Com monitoramento contínuo
ii) De análise periódica.

Pode-se observar que os métodos de detecção se referem a somente duas categorias: comportamento e conhecimento. As técnicas de análise apresentadas para cada método, correspondem somente a um subconjunto do estado da arte. A classificação da análise não menciona o conceito de aprendizagem, que pode ser embutido na mesma, nem a hipótese dela ter sido construída com mais de um detector, empregando mais de um

[10] É uma técnica desenvolvida durante o projeto SECURENET. Ela modela o comportamento normal do usuário pelo conjunto de tarefas que desempenha no sistema. *Computers & Security, Volume 15, Issue 5, pp. 395, 1996.*

método de análise. A característica não funcional, freqüência de uso, se relaciona a duas categorias de mecanismos de ID: IDS e ferramentas de segurança respectivamente.
Com isto nota-se, na figura 3, que se trata de uma taxonomia superficial, não entrando em detalhes quanto às principais características dos três componentes de um IDS (coleta, análise e resposta). Assim pode ser interessante um maior aprofundamento nas características dos componentes de um IDS, levando a pensar em novos critérios para que os mesmos sejam classificados.

COLETA	ANÁLISE	RESPOSTA
ORIGEM	MÉTODO	TIPO
	conhecimento	
host		**ativa**
	sistema especialista	
rede	assinatura de ataque	**passiva**
	rede de Petri	
	transição de estado	
	comportamento	
	estatística	
	sistema especialista	
	rede neural	
	identificação de intenção do usuário	
	imunologia computacional	

Figura 3: Representação dos critérios e das características da taxononia de Debar.

A Figura 4 mostra a classificação de 20 IDS, segundo a taxonomia de Debar, realizada como parte desta dissertação. Note-se que algumas características não identificam, por completo, um critério na classificação. Por exemplo, para os IDS RIPPER[11] e ESP[12], que trabalham tanto com modelo intrusivo, quanto com modelo normal, não existe uma característica na taxonomia que represente seu método de detecção da análise. Outro exemplo seria o IDS AAFID2, que trabalha com coleta pacotes de rede dirigido ao próprio *host*, não existe uma característica que represente sua origem de coleta.

[11] www.cs.columbia.edu/~sal/hpapers/USENIX/usenix.html.

[12] [Zamboni, 2000] *Doing Intrusion Detection Using Embedded Sensors* – Thesis proposal. CERIAS technical Report 2000-21, Purdue University, W. Lafayette, IN, october 18, 2000.

IDS	CRITÉRIO			
	Origem da coleta	**Método da detecção**	**Resposta**	**Freqüência de uso**
IDES	Host	Comportamento	Passiva	Contínuo
MIDAS	Host	Comportamento Conhecimento	Passiva	Contínuo
W&S	Host	Comportamento	Passiva	Contínuo
NSM	Rede	Comportamento Conhecimento	Passiva	Contínuo
Hiperview	Host	Comportamento Conhecimento	Passiva	Contínuo
DIDS	Host/rede	Conhecimento	Passiva	Contínuo
NIDES	Host	Comportamento Conhecimento	Passiva	Contínuo
AID	Host	Conhecimento	Passiva	Contínuo
IDIOT	Host	Conhecimento	Passiva	Contínuo
GASSATA	Host	Conhecimento	Passiva	Contínuo
EMERALD	Host/rede	Comportamento Conhecimento	Ativa	Contínuo
NFR	Rede e N/A	Comportamento Conhecimento	Passiva	Contínuo
SNORT	Rede	Conhecimento	Passiva	Contínuo
ACME	Rede	Comportamento Conhecimento	Ativa	Contínuo
RIPPER	Host	N/A	Ativa	Contínuo
AAFID2	Host e N/A	Comportamento	Passiva	Contínuo
GBID	Host	Comportamento	Passiva	Contínuo
ESP	Host	N/A	Ativa	Contínuo
LISYS	N/A	NA	Passiva	Contínuo
IDA	Host e N/A	Comportamento	Ativa	Contínuo

Figura 4: Classificação dos 20 IDS, segundo Debar. (N/A = Não aplicável)

3.2 Taxonomia de Axelsson

Nessa taxonomia, a análise de dados é classificada em três métodos (e não em dois, como na anterior). A necessidade de identificar IDS que trabalham tanto com modelo intrusivo, quanto com modelo normal, aparece na classificação como um novo método. Também aparecem melhor discriminados os conceitos de aprendizagem que um IDS pode ter, como por exemplo o *self-learning* e o *programmed*.

A taxonomia é estruturada da seguinte forma:

1) A *detecção anômala* é separada nas seguintes categorias:

   a) *Sistema de auto-aprendizagem*, que aprende, por meio de exemplos, o que se constitui como "normal":

      i) *Série não temporal*, que modela o comportamento normal do sistema pelo uso de um modelo estocástico, que não leva em conta o comportamento temporal:

         (1) *Modelagem de regra*. O sistema estuda o tráfego e formula um número de regras que descrevem a operação normal do sistema. No estágio de detecção o sistema aplica as regras e aciona um alarme se o tráfego observado não se relacionar adequadamente com a base de regras.

         (2) *Estatística descritiva*. O sistema coleta estatísticas simples de certos parâmetros do sistema em um perfil normal e constrói um vetor de distâncias entre o tráfego observado e o perfil normal. As discrepâncias podem justificar um alarme.

      ii) *Série temporal*, que leva em conta o comportamento de séries de tempo:

         (1) *Rede Neural Artificial*. O tráfego normal do sistema é alimentado para uma RNA, que subseqüentemente aprende o padrão de tráfego normal. A saída da RNA é então aplicada a um novo tráfego e usada para tomar a decisão da detecção de intrusão.

b) *Programado,* que supõe uma técnica de representação do conhecimento especialista, programando-o para detectar certos eventos anômalos:

i) *Estatística descritiva,* que constrói um perfil de comportamento estatístico normal:
(1) *Estatística simples*. As estatísticas coletadas são usadas para tomar a decisão da detecção de intrusão mais abstrata.
(2) *Baseada em regra simples*. As regras são usadas para serem aplicadas nas estatísticas coletadas.
(3) *Baseada em entradas*. Entradas definidas podem ser programadas para serem usadas com as estatísticas coletadas.

ii) *Default negar*, a idéia desta técnica é fixar (declarar) explicitamente as condições de "normalidade" sob as quais o sistema monitorado (observado) opera e classificar todos os desvios de operação como intrusivo:
(1) *Modelagem de série de estados*: que codifica o comportamento normal do usuário/sistema como um número de diferentes estados que compõem um modelo. Cada um destes estados tem que estar presente no espaço de observação para ser considerado como ocorrido. Se as ações monitoradas são permitidas o sistema continua, se leva a um estado que não é mencionado explicitamente o sistema emite um alarme.

2) A *detecção de mau uso,* definida como *assinatura,* somente apresenta a categoria *programada* e é dividida em:

i) *Modelagem de estado:* que codifica um comportamento intrusivo como um número de diferentes estados. Cada um destes estados tem que estar presente no espaço de observação da intrusão para ser considerado como ocorrido:
(1) *Transição de estado.* Os estados que formam uma intrusão têm que atravessar uma cadeia simples do começo ao fim.
(2) *Rede de Petri*. Os estados formam uma estrutura de árvore mais geral, no qual muitos estados preparatórios podem ser preenchidos em qualquer ordem, independente de onde ocorrem no modelo.

ii) Sistema especialista: máquina de classificação de detecção, que consiste de uma base de conhecimento, a ferramenta de aquisição, contendo descritores de comportamento suspeito, baseado em conhecimento sobre intrusões passadas; máquina de inferência, que organiza o raciocínio e tira conclusões usando regras e fatos e, finalmente, a memória de trabalho.

iii) *Casamento de string:* que busca por cadeia de caracteres transmitidas ou armazenadas entre sistemas.

iv) *Baseado em regra simples:* regras usualmente estruturadas na forma de procedimentos "*if-then-else"* ou construções "*case"*.

3) A *assinatura inspirada* aparece na taxonomia definida como *seleção de característica automática*, que aprende automaticamente aquilo que constitui comportamentos intrusivo e normal.

A taxonomia de [Axelsson, 2000] mostra-se mais completa que a anterior sob o ponto de vista da análise de dados. O foco principal do seu trabalho foi este componente. O método *assinatura inspirada* aparece como uma nova característica do critério "método de detecção".

A Figura 5 mostra os mesmos 20 IDS anteriores classificados segundo a taxonomia da análise de dados de Axelsson. Nota-se, na tabela desta figura, que algumas características não identificam, por completo, um critério na classificação, assim como na taxonomia de Debar. Por exemplo, para os IDS GASSATA[13] e Genetic Based Intrusion Detection[14] (GBID), que trabalham com algoritmo genético, não existe característica na taxonomia que represente a aprendizagem e a técnica da sua análise. O mesmo ocorre com os IDS ACME[15] e Embedded Sensors Project [16](ESP), que trabalham respectivamente com o processo de aprendizagem supervisionada (rede neural) e não supervisionada (*clustering*), não se enquadrando nas características de aprendizagem e técnica da taxonomia de Axelsson.

---

[13] Genetic Algorithm toll for Simplified Security Audit Trail Analysis, www.supelec-rennes.fr/rennes/si/equipe/lme/PUBLI/raid98.pdf.

[14] www.netlab.cs.iitm.ernet.in/publications/ refereed-journal.html.

[15] Advanced Counter Measures Environment, www.acme-ids.org http://www.acme-ids.org.

[16] http://www.cerias.purdue.edu/projects/esp/.

IDS	CRITÉRIO		
	Método MA	**Aprendizagem**	**Princípio da Detecção**
IDES	Anômala	Auto-aprendizagem	Série não temporal/ estat. descritiva
MIDAS	Anômala Assinatura	Programada Programada	Estatística descritiva/ estat. simples Sistema especialista
W&S	Anômala	Auto-aprendizagem	Série não temporal/ modelag. regras
NSM	Anômala Assinatura	Programada Programada	Estatística descritiva/ regras simples Casamento de string
Hiperview	Anômala Assinatura	Auto-aprendizagem Programada	Série temporal/ rede neural Sistema especialista
DIDS	Assinatura	Programada	Sistema especialista
NIDES	Anômala Assinatura	Auto-aprendizagem Programada	Série não temporal/ estat. descritiva Sistema especialista
AID	Assinatura	Programada	Modelagem de estado/ transição de estado
IDIOT	Assinatura	Programada	Modelagem de estado/rede de Petri
GASSATA	Assinatura	N/A	*N/A*
EMERALD	Anômala Assinatura	Auto-aprendizagem Programada	Série não temporal/ estat. descritiva Sistema especialista
NFR	Anômala Assinatura	Programada Programada	Default negar /modelag. série estado Modelagem de estado /trans. estado
SNORT	Assinatura	Programada	Base de regras simples
ACME	Anômala Assinatura	Programada N/A	N/A N/A
RIPPER	Assinat. Inspirada	Auto-aprendizagem	Seleção de característica automática
AAFID2	Anômala	Programada	Estatística descritiva/ regras
GBID	Anômala	Auto-aprendizagem	N/A
ESP	Assinat.Inspirada	Auto-aprendizagem	N/A
LISYS	Assinat. Inspirada	Auto-aprendizagem	N/A
IDA	Anômala	Programada	Estatística descritiva/ regras

Figura 5: Classificação da análise de dados, dos 20 IDS, segundo Axelsson. (N/A = Não aplicável).

A outra parte da taxonomia de Axelsson classifica os IDS quanto às suas características, que não se relacionam diretamente com a análise de dados. Ela identifica 08 (oito) características:

- Tempo de detecção (tempo real ou não);
- Granularidade do processamento dos dados (contínuo ou em lotes);
- Origem dos dados auditados (baseados em *host* ou em rede);
- Resposta à detecção de intrusão (ativos ou passivos);
- Localização da coleta de dados pode ser:
    - Coleta de dados centralizada, onde os dados usados pelo IDS são coletados em localizações fixas e independe dos componentes monitorados.
    - Coleta de dados distribuída, onde os dados usados pelo IDS são coletados em localizações proporcionais aos componentes monitorados.

- Localização do processamento dos dados:
    - Processamento dos dados centralizado, onde a análise dos dados é feita em localizações fixas e independe dos componentes monitorados.
    - Processamento dos dados distribuído, no qual a análise de dados é feita em localizações proporcionais aos componentes monitorados.

  O que classifica um IDS como centralizado ou distribuído é seu modo de análise dos dados.

- Segurança (alta, moderada ou baixa);
- Grau de interoperabilidade (alto ou baixo).

A figura 6 mostra a tabela representativa dos critérios e das características da taxonomia de Axelsson.

IDS		
COLETA	**ANÁLISE**	**RESPOSTA**
ORIGEM	MÉTODO	TIPO
	assinatura	
host	*programada*	**ativa**
rede	modelagem estado sistema especialista casamento string regras simples	**passiva**
	anomalia	
	automática	
	série temporal série não temporal	
	programada	
	estatística descritiva “default” negar	
	“assinatura inspirada”	
	seleção característica automática	
LOCALIZAÇÃO	LOCALIZAÇÃO	
centralizada	**centralizada**	
distribuída	**distribuída**	

Figura 6: Representação dos critérios e das características da taxonomia de Axelsson.

A tabela da Figura 7 aparece incompleta tendo em vista o enfoque desta dissertação não tratar de identificar determinados critérios como tempo de detecção, granularidade, segurança e grau de interoperabilidade.

IDS	CRITÉRIOS							
	Tempo de detecção	Granula-ridade	Origem da Coleta	Respos-ta	Processa-mento dados	Coleta dados	Segu-rança	Interope-rabilidade
IDES	Real	Contínua	Host	Passiva	Centralizada	Distribuída	Baixa	Baixa
MIDAS	Real	Contínua	Host	Passiva	Centralizada	Centralizada	Baixa	Baixa
W&S	Real	Contínua	Host	Passiva	Centralizada	Centralizada	Baixa	Baixa
NSM	Real	Contínua	Rede	Passiva	Centralizada	Centralizada	Baixa	Baixa
Hiperview	Real	Contínua	Host	Passiva	Centralizada	Centralizada	Baixa	Baixa
DIDS	Real	Contínua	Host/rede	Passiva	Distribuída	Distribuída	Baixa	Baixa
NIDES	Real	Contínua	Host	Passiva	Centralizada	Distribuída	Baixa	Alta
AID	N/T	N/T	Host	Passiva	Centralizada	Distribuída	N/T	N/T
IDIOT	Real	Contínua	Host	Passiva	Centralizada	Centralizada	Baixa	Baixa
GASSATA	N/T	N/T	Host	Passiva	Centralizada	Distribuída	N/T	N/T
EMERALD	Real	Contínua	Host/rede	Ativa	Distribuída	Distribuída	Moderada	Alta
NFR	N/T	N/T	Rede/ N/A	Passiva	Distribuída	Distribuída	N/T	N/T
SNORT	N/T	N/T	Rede	Passiva	Centralizada	Centralizada	N/T	N/T
ACME	N/T	N/T	Rede	Ativa	Centralizada	Centralizada	N/T	N/T
RIPPER	N/T	N/T	Host	Ativa	Distribuída	Distribuída	N/T	N/T
AAFID2	N/T	N/T	Host/ N/A	Passiva	N/A	N/A	N/T	N/T
GBID	N/T	N/T	Host	Passiva	Distribuída	Distribuída	N/T	N/T
ESP	N/T	N/T	Host	Ativa	N/A	N/A	N/T	N/T
LISYS	N/T	N/T	Host/ N/A	Passiva	Distribuída	Distribuída	N/T	N/T
IDA	N/T	N/T	Host/ N/A	Ativa	Centralizada	Distribuída	N/T	N/T

Figura 7: Classificação dos 20 IDS, segundo Axelsson. (N/A=não aplicável e N/T=não tratados nesta dissertação)

3.3 Proposta de uma nova taxonomia

Tendo em vista que as taxonomias conhecidas não abordam certos critérios/características, alguns IDS não se enquadram nas mesmas e poderiam ser melhor classificados, como por exemplo:

- O IDS GASSATA utiliza, na sua análise de dados, o método baseado em conhecimento, implementando regras e algoritmo genético que podem classificar eventos do sistema e procurar por assinaturas de ataques.

- O IDS ACME utiliza um modelo com duas camadas de análise de dados (em série), onde a primeira camada de análise, baseada em comportamento, implementa um Sistema Especialista, no qual valores limites são passados para a base de conhecimento, que define o comportamento do sistema (níveis de segurança de serviços, por exemplo: *telnet*, *ftp*, *finger*, etc). Quando este limite é ultrapassado, certos procedimentos são disparados. A segunda camada de análise utiliza o método conhecimento, mas implementa uma RN supervisionada, podendo detectar ataques dissimulados, que têm leve variação na assinatura de ataque, em que o intruso tenta esconder suas reais intenções, seguindo um conjunto de ações não correlatas com o ataque.

- O IDS AAFID2 utiliza, na sua coleta de dados, pacotes de rede direcionados ao próprio *host* para serem analisados pelos agentes estáticos que implementa.

- O IDS GBID utiliza o método baseado em comportamento e implementa regras e algoritmo genético para observar anomalia no uso de comandos por usuários individuais. O comportamento do usuário é aprendido continuamente para capturar desvio que resulte em um número menor de alarmes falsos na detecção de intrusão.

- O IDS ESP utiliza o método composto e implementa uma técnica de aprendizagem não supervisionada, o *clustering*. Também apresenta uma

arquitetura altamente distribuída, onde a análise é desempenhada em número proporcional ao dos locais de coleta.

- O IDS LISYS[17] utiliza o método composto com aprendizagem não supervisionada e implementa tecnologia de imunologia computacional, observa conexões TCP direcionadas ao próprio host e também o comportamento normal de cada programa de interesse, por meio de seqüências de chamada ao sistema.

Devido a estes exemplos é proposta uma taxonomia que se baseia nos 3 (três) componentes típicos de uma arquitetura clássica (coleta, análise e resposta) dando, igualmente, importância aos componentes. Embora simplista sob o ponto de vista do grande número de características que poderiam ser referidas a um IDS, classifica os IDS em relação aos principais critérios de funcionalidade.

No componente coleta de dados é interessante a inclusão do critério mecanismo de coleta. Os dados de auditoria são muitos e devem ser reduzidos na coleta. Para tanto, é importante ser determinado o mecanismo que será usado na coleta dos dados.

Um componente de análise do IDS também pode ser aprofundado em detalhes quanto à técnica utilizada. Algumas técnicas permitem agregar conhecimento e utilizam tecnologias de inteligência artificial.

Uma arquitetura clássica de um IDS envolve a arquitetura de coleta e também de análise. À medida que as tecnologias de coleta e análise vão se desenvolvendo, torna-se viável desempenhar a coleta e a detecção tão próximas quanto possível (no tempo e no espaço), levando a uma nova abordagem para a arquitetura de um IDS.

Não foram levadas em consideração, na proposta da nova taxonomia, características que não estivessem diretamente ligadas aos três principais componentes de um IDS, uma vez que o foco principal do trabalho é voltado para as características que se referem ao trabalho interno dos componentes, identificando sua informação de entrada, seus mecanismos de análise e sua interação com o mecanismo de resposta. Note que se

[17] Lightweigth Intrusion detection System

outras características fossem incluídas na taxonomia, algumas delas perderiam o sentido:

- Tempo de detecção, real ou não. Um sistema de tempo real deve responder dentro de restrições de tempo estritas (tipicamente varia de um milisegundo a 1 minuto) (Pressman, 1995). Pode ser difícil classificar se forem vistos completando as fases de coleta, análise e resposta.

- Granularidade do processamento dos dados discriminados em contínuo ou em lotes. Como é uma característica ligada a característica tempo de detecção também fica difícil de ser analisada.

- Segurança discriminada em alta e baixa. A segurança do próprio IDS é uma característica difícil de ser determinada.

- Grau de interoperabilidade alto e baixo. Interoperar com outros IDS envolve questões ortogonais ao próprio IDS.

- Freqüência de uso, contínuo e periódico. Pode não depender da arquitetura.

Esta nova taxonomia engloba diferentes classificações de IDS, que serão utilizadas para compor uma tabela (figura 11) que possibilitará uma análise comparativa de 20 (vinte) sistemas tratados anteriormente. Nota-se, na figura 8, que embora os critérios arquitetura de coleta e análise de dados apareçam respectivamente nos componentes coleta e análise, eles serão tratados numa seção separada (3.3.4); para facilitar a compreensão da arquitetura de um IDS, além de estarem intrinsecamente ligados.

As classificações se relacionam:

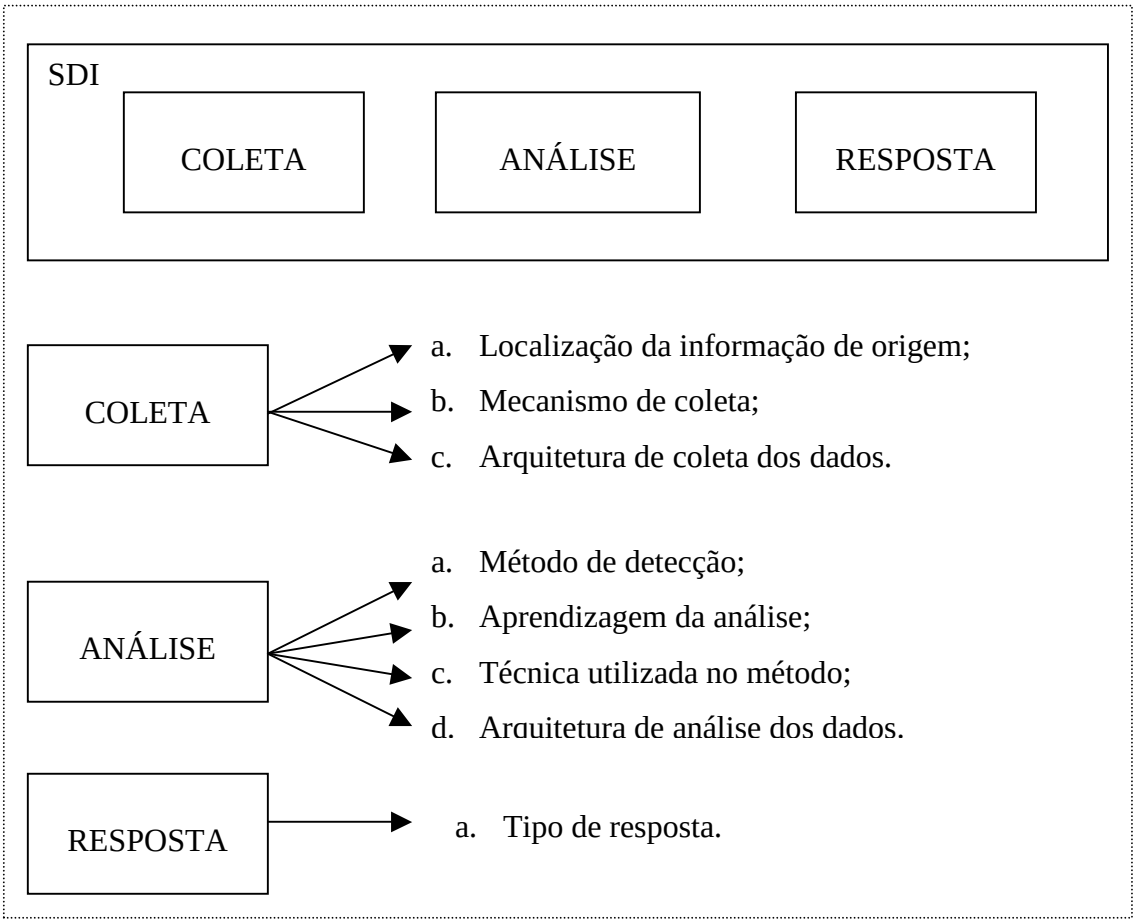

Figura 8: Critérios da nova taxonomia proposta

3.3.1 Coleta de Dados

Os IDS podem ser classificados segundo o critério "localização da informação de origem". São três [Geus, 2002]:

- IDS baseados em *host,* que fazem o monitoramento do sistema com base em informações de arquivos de *logs* ou de agentes de auditoria. Podem ser capazes de monitorar acessos e alterações em arquivos do sistema, modificações nos privilégios dos usuários, nos processos do sistema que estão sendo executados, no uso da CPU, entre outros aspectos.

- IDS baseados em rede, que monitoram o tráfego da rede, ou de um segmento da rede, geralmente com a interface de rede em modo promíscuo. A detecção é

realizada com a captura e análise dos cabeçalhos e conteúdos dos pacotes, que são comparados com padrões e assinaturas de ataques conhecidos.

- IDS híbridos, que assim como os IDS baseados em rede coletam o tráfego de rede, processando os pacotes, detectando e respondendo a ataques. Processam, ainda, os pacotes endereçados ao próprio sistema, como no caso do IDS baseado em *host*.

Os IDS podem ser classificados segundo o critério “mecanismo de coleta de dados”, Utilizando componentes responsáveis pela captura dos dados, sendo especificados conforme, a seguir:

- Coletores. A função do coletor é coletar informações de origem e enviá-las para um gerente central tão logo as entradas ocorram. O gerente central faz toda a análise e decisão de acordo com a política. Coletores podem ser:

  - Coletores de *host*. Que são aplicações leves que residem no *host*. A função do coletor é coletar informações de arquivos e *logs* do *host* e, enviá-las para um gerente central.

  - Coletores de rede. Com a função de coletar pacotes da rede e enviá-los para um gerente central.

- Sensores. Que são dispositivos de software que monitoram a origem de dados “por ação específica” ou, “por um evento suspeito”, relatando suas decisões para o componente de análise. Podem ser:

  - Sensores de rede. Permitem monitorar o tráfego da rede. São colocados em locais específicos, no perímetro da rede ou dentro desta. São dispositivos de processamento intensivo e, geralmente, exigem uma máquina dedicada. O sensor analisa todo o tráfego da rede, procurando por uma evidência de intrusão e, então, relata a informação para um componente de análise, seguindo os parâmetros da política do IDS em rede.

- Sensores de *host*. Monitoram uma variável específica, atividade ou condição de um *host*. Suas observações são relatadas como valores numéricos, podendo ser discretos ou contínuos. Têm a função de monitorar arquivos específicos, relatando suas permissões por exemplo. Também fazem uso de comandos do S.O., como por exemplo o uso do comando *ps* para observar o carregamento da CPU no *host* UNIX, extraindo, diretamente, dados carregados da estrutura de dados correspondentes no k*ernel*. Funcionam de acordo com uma política de segurança baseada em *host*.

3.3.2 Análise de Dados

Para compreendê-la deve-se atentar para três conceitos: método de detecção, aprendizagem da análise de dados e técnica utilizada pelo método de detecção, também referenciada por mecanismo, tecnologia e princípio de detecção.

a) Método de detecção. Este conceito descreve as características do analisador, dando ênfase ao tratamento dos dados. Podem ser:
   - Baseado em comportamento, também conhecido como baseado em anomalia, que ocorre quando o IDS usa uma informação sobre o comportamento normal do sistema que monitora, ou de seus usuários, observando desvios que sinalizem sintomas ou cenários de ataque. O modelo normal ou de comportamento válido é extraído de informação de referência coletada por vários meios. Mais tarde, o IDS compara este modelo com a atividade corrente. Se um desvio é observado, um alarme é gerado. Em outras palavras, qualquer comportamento que não corresponda a um comportamento previamente aprendido é considerado intrusivo.
   - Baseado em conhecimento, também referido como mau uso, baseado em aparência ou assinatura, assim definido quando o IDS usa informações de conhecimento acumulado sobre ataques específicos, ou vulnerabilidades conhecidas no sistema. Estas informações podem ser descritas por um padrão específico, ou seqüência de eventos ou dados. A máquina de classificação de dados é construída em cima de descrição em termos de regras e outros padrões descritores. Estas regras ou descritores são assinaturas. Quando uma

assinatura de ataque é detectada um alarme é disparado. Em outras palavras, qualquer ação que não seja explicitamente reconhecida como um ataque, é considerada aceitável.

- Composto. A combinação dos dois métodos leva a detectores compostos. Este método, automaticamente, aprende o que se constitui como comportamentos intrusivo e normal para um sistema.

b) Aprendizagem. Independentemente do método utilizado, é preciso que o conhecimento, tanto das assinaturas de ataque quanto do comportamento normal do usuário, ou sistema, seja passado para o IDS. O novo conceito apresentado é definido como o estudo de métodos computacionais para adquirir novos conhecimentos, novas habilidades e novas maneiras de organizar o conhecimento existente.

Pode-se aprender:

- Um conjunto de novos fatos;
- Como fazer alguma coisa;
- Como melhorar a habilidade de alguma coisa já aprendida;
- Como fazer a mesma tarefa mais eficientemente.

Os processos de aprendizagem podem ser definidos segundo a árvore da figura 9:

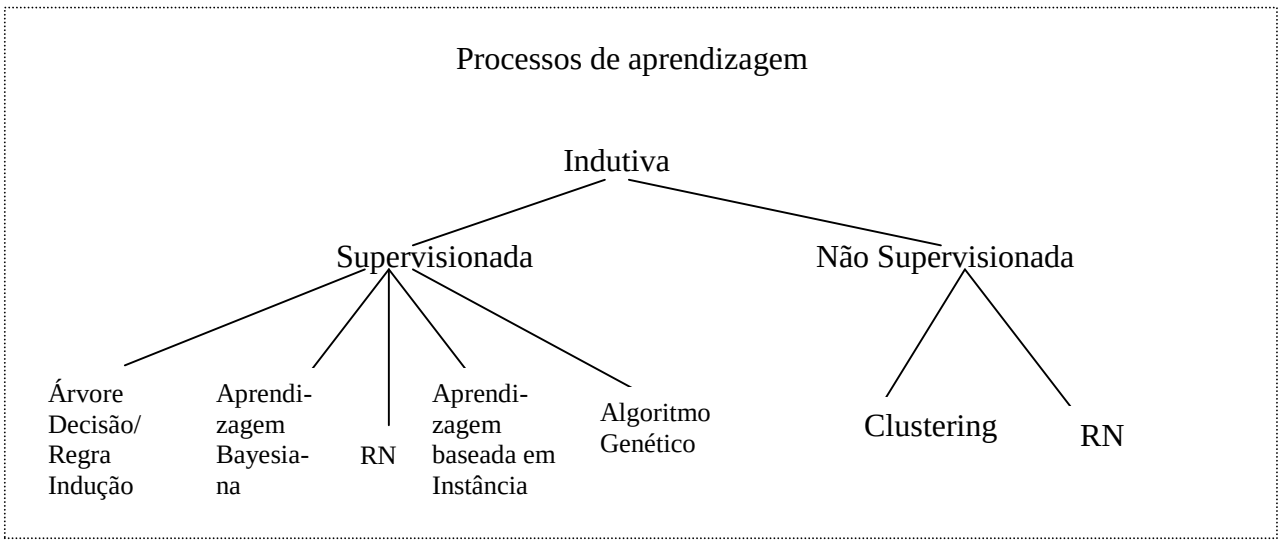

Figura 9: Árvore representativa dos processos de aprendizagem utilizados na taxonomia

Os processos de aprendizagem podem ser agrupados em:

- Automática. Através de casos e exemplos, leva à indução automática e gera as regras. Aprender denota mudanças no sistema, que são adaptativas, no sentido que elas permitem que o sistema faça a mesma tarefa de maneira mais eficiente. Os processos de aprendizagem podem ser ainda:

    - Supervisionados, quando usa dados fornecidos por um supervisor, como um conjunto de exemplos pré-classificados para aprender a descrição geral, que encapsula a informação nestes exemplos, podendo ser usada para previsão ou prognóstico, como árvore de decisão, regra de indução, rede *bayesiana*, rede neural, aprendizagem baseada em instância e algoritmos genéticos (otimização da técnica de busca).
    - Não supervisionados, quando é fornecida uma coleção de dados não classificados para buscar por regularidades, como *clustering* e rede neural.

- Programada. Este mecanismo de aquisição de conhecimento seria manual a partir de origem externa. Como por exemplo, adição de filtros para assinaturas de ataques ou perfil normal de comportamento do usuário/sistema, numa base de conhecimento, ou mesmo adição de regras em sistemas especialistas.

c) Técnicas utilizadas pelo método de detecção. São técnicas de reconhecimento de padrão implementadas nos IDS. Às vezes, a mesma técnica aparece implementada em diferentes métodos de detecção e, também, com diferentes mecanismos de aprendizagem. Basicamente elas podem ser vistas como:

- Técnicas que trabalham com regras. Modela o comportamento do sistema/usuário pelo uso de um modelo estocástico, que não leva em conta o comportamento temporal. A análise baseada em regra conta com conjunto de regras pré-definidas, que são fornecidas por um administrador, ou criadas,

automaticamente, pelo sistema, ou ambas. Estas regras são usualmente estruturadas na forma de procedimento *if-then-else* ou construções *case*.

- Técnicas que trabalham com exemplos. Treinam inicialmente o sistema para corretamente identificar exemplos pré-selecionados do problema, ganhando experiência. Em adição ao período de treinamento inicial, também ganham experiência com o tempo e, na maneira que conduzem a análise de dados relacionada ao problema.

Elas podem ser associadas a:

- Estatística. A análise estatística envolve comparação de eventos específicos baseados num conjunto de critérios pré-definidos, que não leva em conta o comportamento temporal. Esta técnica mantém uma base de conhecimentos estatísticos constituída de *profiles*, que são definidos como um conjunto de métricas. Estas são medidas de aspectos particulares do comportamento do usuário no sistema. Cada métrica é associada a uma entrada ou intervalo de valores. Estes *profiles* são periodicamente atualizados para refletir mudanças no comportamento do usuário todo o tempo.

Ou levar em consideração o comportamento de séries de tempo:

- Modelagem de série de estados, que codifica o comportamento do usuário/sistema como um número de diferentes estados. Cada um destes estados tem que estar presente no espaço de observação da intrusão para ser considerado como ocorrido.

3.3.3 Mecanismo de Resposta

Uma classificação em relação ao componente de resposta pode ser:

- Passiva, notificando a autoridade por meio do envio de *email*, entradas para um *log* de segurança, etc, que conta com o AS para tomada de decisão.
- Ativa (ou reativa) que se divide em:

- As que exercem controle sobre os sistemas atacados, modificando seu estado ou atenuando o efeito do ataque. Exemplo: terminando conexões de rede, aumentando a segurança de *logging* ou matando processos.
- E as que exercem controle sobre os sistemas atacantes. Estes, por sua vez, atacam os intrusos, tentando remover sua plataforma de operação.

3.3.4 Arquitetura de um IDS

A detecção de intrusão pode ser vista como sendo realizada por três componentes distintos: coleta de dados, análise e resposta. Nesta seção, são descritas as diferentes estruturas que um IDS pode ter na coleta e análise de dados.

Arquitetura de Coleta de Dados

A arquitetura de coleta de dados para ID refere-se à classificação em relação ao local da coleta de dados, podendo ser:

- Coleta de dados centralizada, onde os dados usados pelo IDS são coletados numa única localização.
- Coleta de dados distribuída, onde os dados usados pelo IDS são coletados em localizações fixas e independe dos componentes monitorados.
- Coleta de dados altamente distribuída, onde os dados usados pelo IDS são coletados em localizações proporcionais aos componentes monitorados.

Arquitetura de Análise de Dados

Os IDS são classificados, segundo seu mecanismo de análise, em centralizados, distribuídos e altamente distribuídos. Suas definições são:

- IDS de análise centralizada, onde a análise dos dados é desempenhada numa única localização.
- IDS de análise distribuída, no qual a análise de dados é desempenhada em localizações fixas e independe dos componentes monitorados.

- IDS de análise altamente distribuída, no qual a análise de dados é desempenhada em localizações proporcionais aos componentes monitorados.

Localização é definida como uma instância de código em execução. Assim por exemplo, uma análise de dados é dita análise altamente distribuída se implementada como uma biblioteca compartilhada ligada a muitos programas, onde cada programa executa o mecanismo separadamente. Se a biblioteca é ligada a mais de um programa e não a todos, a análise é dita distribuída. Quando é ligada a um único programa e analisa toda a informação necessária pelo IDS, é dita análise centralizada.

A figura 10 mostra a tabela representativa dos critérios e das características da taxonomia proposta.

IDS		
COLETA	**ANÁLISE**	**RESPOSTA**
LOCALIZAÇÃO	MÉTODO	TIPO
host	**conhecimento**	**ativa**
rede	**comportamento**	**passiva**
híbrido	**composto**	
MECANISMO	APRENDIZAGEM	
coletor	**programada**	
sensor	**automática supervisionada**	
	automática não supervisionada	
	TÉCNICA	
	regras	
	exemplos	
ARQUITETURA	ARQUITETURA	
centralizada	**centralizada**	
distribuída	**distribuída**	
altamente distribuída	**altamente distribuída**	

Figura 10: Tabela representativa dos critérios e características da taxonomia proposta.

A seguir é mostrada uma tabela comparativa de 20 (vinte) IDS, tratados anteriormente.

IDS	CRITÉRIOS							
	Coleta			Análise				Resposta
	Local In-formação origem	Mecanismo Coleta	Arquitetura Coleta de Dados	Método de Detecção	Aprendizagem	Técnica de Detecção	Arquitetura Análise de Dados	Tipo de Resposta
IDES	Host	Coletor host	Distribuída	Comportamento	Automática Supervisionada	Regras/Estatística descritiva	Centralizada	Passiva
MIDAS	Host	Coletor host	Centralizada	Comportamento Conhecimento	Programada Programada	Regras/Estatística descritiva Regras/Sistema especialista	Centralizada	Passiva
W&S	Host	Coletor host	Centralizada	Comportamento	Automática	Regras indução	Centralizada	Passiva
NSM	Rede	Coletor rede	Centralizada	Comportamento Conhecimento	Programada Programada	Regras/Estat. descritiva Regras	Centralizada	Passiva
Hiperview	Host	Coletor host	Centralizada	Comportamento Conhecimento	Automática/Sup. Programada	Exemplos / RN Regras/Sist. especialista	Centralizada	Passiva
DIDS	Host/ Rede	Sensor host Sensor rede	Distribuída	Conhecimento	Programada	Regras/Sistema especialista	Distribuída	Passiva
NIDES	Host	Coletor host	Distribuída	Comportamento Conhecimento	Automática/Superv. Programada	Regras/Estat. descritiva Regras/Sist. especialista	Centralizada	Passiva
AID	Host	Coletor host	Distribuída	Conhecimento	Programada	Regras/Modelag. estado	Centralizada	Passiva
IDIOT	Host	Coletor host	Centralizada	Conhecimento	Programada	Regras/Modelag. estado	Centralizada	Ativa
GASSATA	Host	Coletor host	Distribuída	Conhecimento	Automática/Sup.	Regras/estatística/ Algoritmo Genético	Centralizada	Passiva

IDS	CRITÉRIOS							
	Coleta			Análise				Resposta
	Local In-formação origem	Mecanismo Coleta	Arquitetura Coleta de Dados	Método de Detecção	Aprendizagem	Técnica de Detecção	Arquitetura Análise de Dados	Tipo de Resposta
EMERALD	Host Rede	Sensor host Sensor rede	Distribuída	Comportamento Conhecimento	Automática/Sup. Programada	Regras/Estat. descritiva Regras/Sist. especialista	Distribuída	Ativa
NFR	Rede/Híbrido	Sensor rede/ Sensor host	Distribuída	Comportamento Conhecimento	Programada Programada	Regras/Modelag. estado Regras/Modelag. estado	Distribuída	Passiva
SNORT	Rede	Sensor rede	Centralizada	Conhecimento	Programada	Regras	Centralizada	Passiva
ACME	Rede	Sensor rede	Centralizada	Comportamento Conhecimento	Programada Automática/Sup.	Regras/Sistema Espec. Exemplos/ RN	Centralizada	Ativa
RIPPER	Host	Sensor host	Distribuída	Composto	Automática/Sup.	Reg. indução/mineraç.	Distribuída	Ativa
AAFID2	Host Híbrido	Sensor host	Distribuída	Comportamento	Automática/Sup.	Regras/Estat. descritiva	Distribuída	Passiva
GBID	Host	Sensor host	Distribuída	Comportamento	Automática/Sup.	Regras/estatíst./A.Genét.	Distribuída	Passiva
ESP	Host	Sensor host	Alta. distribuída	Composto	Automática/Não Sup	Regras/clustering	Alta. distribuída	Ativa
LISYS	Host/Híbrido	Sensor host	Distribuída	Composto	Automática/Não Sup.	Regra/Imunologia comp.	Distribuída	Passiva
IDA	Host Híbrido	Coletor host Sensor host	Distribuída	Comportamento	Automática/Sup.	Regras induçao	Centralizada	Ativa

Figura 11: Classificação dos 20 (vinte) IDS, segundo a nova taxonomia proposta.

3.4 Comparação das taxonomias

Tabela comparativa dos critérios usados que compõem três taxonomias de IDS:

Critérios	**Taxonomias**		
	Debar	Axelsson	**Taxonomia Proposta**
Local.inform. origem	Sim	Sim	**Sim**
Mecanismo coleta	Não	Não	**Sim**
Arquitetura coleta	Não	Sim	**Sim**
Método	Sim	Sim	**Sim**
Aprendizagem	Não	Sim	**Sim**
Técnica	Não	Sim	**Sim**
Arquitetura análise	Não	Sim	**Sim**
Mecanismo resposta	Sim	Sim	**Sim**

Figura 12: Comparação dos critérios da taxonomia proposta com as usuais.

Tabela comparativa das características usadas que compõem as três taxonomias de IDS;

Critério	**Característica**	**Taxonomia**		
		Debar	Axelsson	**Taxonomia Proposta**
Local.inf.origem	Rede	Sim	Sim	**Sim**
	Host	Sim	Sim	**Sim**
	Híbrido	Não	Não	**Sim**
Mecanismo coleta	Coletores	Não	Não	**Sim**
	Sensores	Não	Não	**Sim**
Arquitetura coleta	Centralizada	Não	Sim	**Sim**
	Distribuída	Não	Sim	**Sim**
	Altamente distribuída	Não	Não	**Sim**
Método	Comportamento	Sim	Sim	**Sim**
	Conhecimento	Sim	Sim	**Sim**
	Composto	Não	Sim	**Sim**
Aprendizagem	Automática supervisionada	Não	Sim	**Sim**
	Automática não supervisionada	Não	Não	**Sim**
	Programada	Não	Sim	**Sim**
Arquitetura análise	Centralizada	Não	Sim	**Sim**
	Distribuída	Não	Sim	**Sim**
	Altamente distribuída	Não	Não	**Sim**
Mecanismo resposta	Ativa	Sim	Sim	**Sim**
	Passiva	Sim	Sim	**Sim**

Figura 13: Comparação das características da taxonomia proposta com as usuais.

Se taxonomias de IDS forem vistas com relação aos três principais componentes de um IDS, que são a coleta de dados (CO), a análise (MA) e o mecanismo de resposta (RE), pode-se visualizar os critérios identificados na taxonomia, tanto funcionais quanto não funcionais, como um grande conjunto e três outros subconjuntos representando alguns dos critérios estudados até o presente, destes três componentes respectivamente.

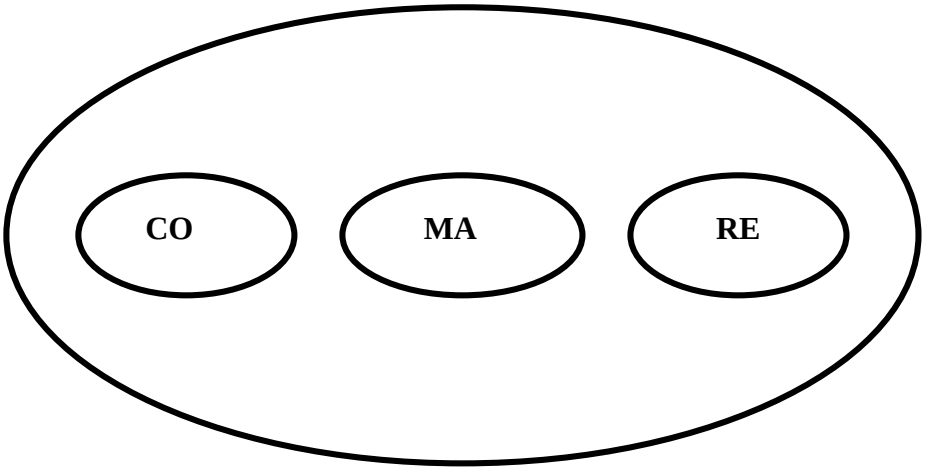

Figura 14: Visão global das características dos três componentes de um IDS

A taxonomia de Debar pode ser identificada atuando em:

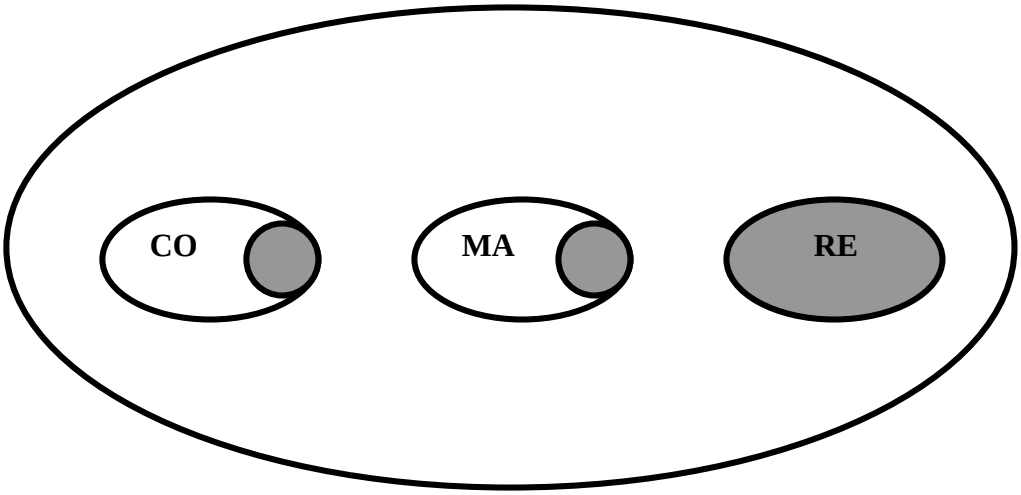

Figura 15: Abrangência das características na taxonomia de Debar

A taxonomia de Axelsson pode ser vista atuando em:

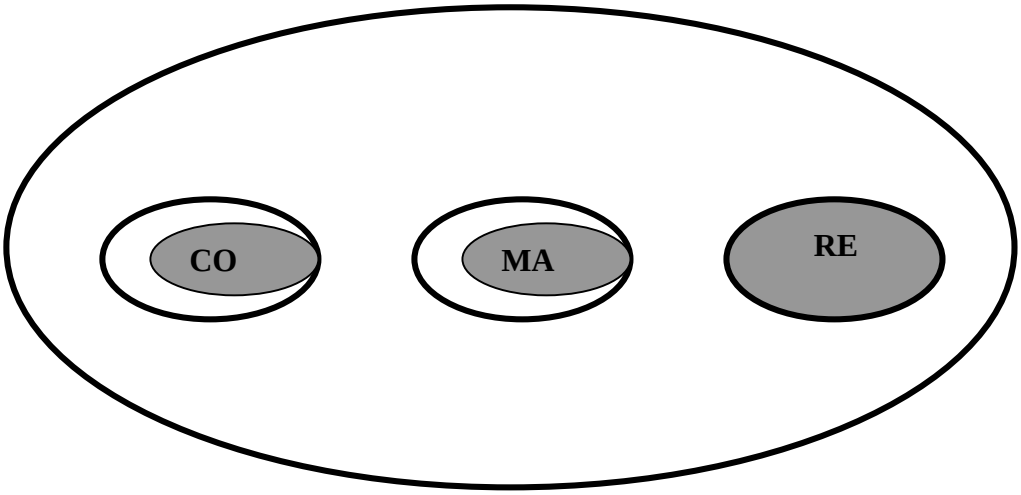

Figura 16: Abrangência das características na taxonomia de Axelsson.

A nova taxonomia pode ser vista atuando em:

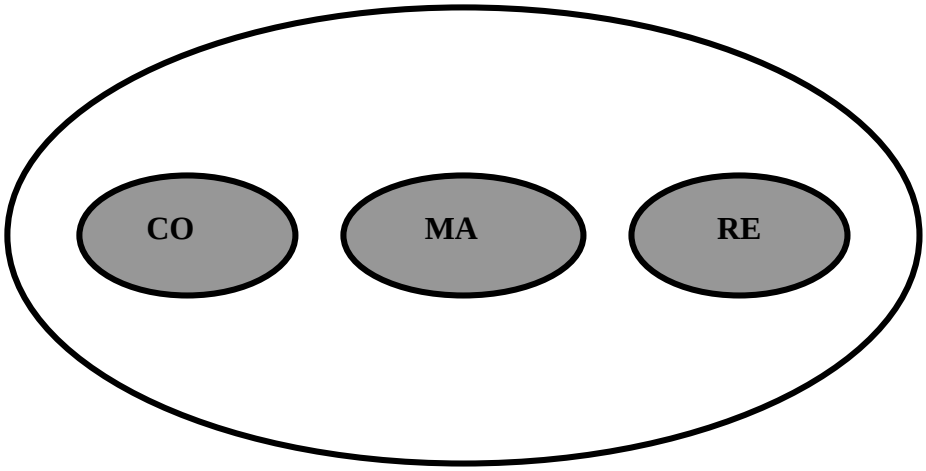

Figura 17: Abrangência das características na nova taxonomia proposta.

3.5 Conclusão

Os três componentes da ID citados por [Bace,2000] foram identificados e utilizados em uma taxonomia mais completa, abrangendo as três fases: coleta, análise e resposta. Com isto foi possível classificar mais completamente os 20 IDS tomados como exemplo.

Ao primeiro componente “coleta de dados” foi incluído o critério “mecanismo de coleta”, pois é importante que o grande número de dados de auditoria sejam reduzidos na fase de coleta, para tanto, deve ser determinado o mecanismo que será usado nesta fase, isto direciona a escolha de

um IDS, identificando também características implícitas como armazenamento de dados coletados. Os IDS que utilizam coletores têm a desvantagem de carregar um volume de dados maior que os IDS que trabalham com sensores, pois estes selecionam e obtêm somente a informação necessária, tendo como resultado, uma menor quantidade de dados gerada, podendo monitorar os dados e somente produzir resultados quando eventos relevantes são detectados. Já os IDS que utilizam sensores têm a desvantagem de serem de implementação mais complexa, pois não fazem só coleta de dados e o mecanismo de monitoramento tem que ser projetado de maneira mais específica para o componente que vai monitorar e, para o tipo de informação que irá gerar. Adicionalmente, se forem vistos com relação ao analisador, os sensores não têm inteligência semelhante, podendo não coletar dados suficientes para o analisador tomar sua decisão.

Também foi acrescentada a característica “híbrido” ao critério “local da informação de origem”. É um tipo de IDS que tem a vantagem de examinar padrões no tráfego de rede, que não é feito em modo promíscuo, monitorando somente os pacotes destinados ao *host* em questão e determinam, em tempo real, se um ataque está sendo executado, podendo, então, responder em tempo real. No entanto apresentam o problema de escalabilidade, pois um IDS híbrido deve ser instalado em cada equipamento.

O segundo componente “análise”, foi aprofundado em mais detalhes quanto à técnica utilizada no engenho de detecção. Algumas técnicas permitem agregar conhecimento e utilizam tecnologias de inteligência computacional que agora aparecem de forma mais evidente quando o IDS é tratado na taxonomia proposta. Também foi levado em consideração o método composto que, embora mais complexo, talvez detecte, com maior rapidez, uma intrusão, porque trabalha com modelo intrusivo e de comportamento normal.

A arquitetura de um IDS foi vista através de outra abordagem. Um exemplo evidente é o caso do IDS ESP, que pelas taxonomias de Debar e de Axelsson não podem ser classificados. No entanto, na taxonomia proposta aparece classificado como arquitetura altamente distribuída, aquela que a análise é desempenhada no mesmo número de pontos de coleta, ou seja, tão próxima no tempo e no espaço quanto possível.

A taxonomia proposta é limitada no grau de aprofundamento do terceiro componente de um IDS “mecanismo de resposta”. Independentemente do modo de resposta dos IDS, de respostas reativas ou passivas, nos dois casos a visualização dos resultados é importante para se conseguir manusear

os resultados advindos dos IDS, referentes a ataques ou intrusões, para a interação do componente alarme e o Administrador de Segurança - AS. Também ajuda a examinar, através do resultado alcançado, a reação tomada ou interagir com o sistema, com a intenção de decidir por alternativas. Embora a visualização dos resultados não pareça fazer parte de um IDS, sendo apenas uma fase posterior à resposta, ela não só complementa, de maneira valiosa, o mecanismo de resposta como também traz benefícios, levando-se em consideração que, na resposta reativa, apesar do AS não interferir nos mecanismos de resposta, pode ser útil no caso de uma ação de recuperação das configurações do sistema. Na resposta passiva, as notificações feitas à autoridade, através de *email* ou gravações nos *logs* de segurança, precisam ser analisadas para possíveis respostas manuais.

Observa-se que, apesar da tendência de ter IDS distribuídos, a questão da apresentação dos resultados para visualização pelo AS continua sendo centralizada. Os alertas referentes a rede e a *host* devem ser apresentados ao AS de maneira integrada para que o diagnóstico seja feito o mais acurado possível. IDS que permitem a visualização dos resultados consolidados, mantendo o poder do AS entender, analisar e intervir nos mínimos detalhes são bem vistos.

O ideal seria que um IDS conseguisse reunir eficiência nos mecanismos de coleta de dados, de análise e de resposta. Enquanto isto não é plenamente encontrado num IDS, uma característica, também relacionada a seus componentes principais como a sua arquitetura pode, de maneira considerável, ajudar na sua eficiência. A incorporação de novos métodos de coleta e análise e a disponibilidade dos resultados da análise para interferência do AS, podem, por exemplo, ser derivada de sua arquitetura.

Capítulo 4

Uma proposta de um modelo de IDS

A proposta de um modelo foi possível através:

- do estudo da Detecção de Intrusão e identificação de suas características, vantagens e limitações;
- da estruturação de uma nova taxonomia com definição de critérios e identificação de características que levam a uma clara visualização dos 3 (três) principais componentes de um IDS.

O modelo proposto tem como objetivo uma nova abordagem direcionada aos 3 (três) componentes funcionais de um IDS, coleta, análise e resposta. Esta abordagem é mostrada por meio:

- do modelo conceitual;
- do diagrama de fluxo de dados,
- da arquitetura de um IDS, que será explicada por meio do diagrama de classes [18]da *Unified Modeling Language* – UML [Furlan, 1998].

[18] Apêndice B

4.1 Modelo Conceitual

O modelo de um IDS, baseado no modelo conceitual do I-DREAM [Silva e outros, 1997], conforme a figura 18, tem como objetivo descrever a funcionalidade do sistema, abstraindo-se das características de implementação. Pode ser visto da seguinte forma:

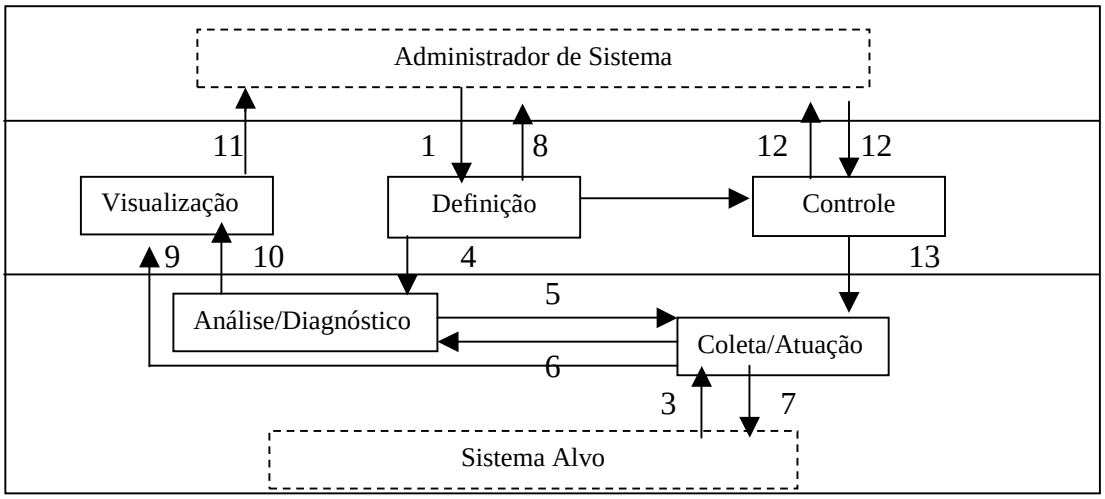

Figura 18: Modelo de um IDS

Em um trabalho anterior, [Franklin, 1997] apresentou este modelo e um protótipo de um Sistema de Monitoração de Recursos e Aplicações Baseados em Intranet, ambos referenciados por I-DREAM. A metodologia apresentada especifica um modelo conceitual que pode ser aplicado a outros problemas. Este trabalho se utilizou deste modelo conceitual e propôs uma arquitetura aplicada ao problema de detecção de intrusão.

A funcionalidade de cada subsistema é descrita abaixo:

- Visualização: É o subsistema que fornece, ao AS, as informações sobre os resultados das análises realizadas pelo subsistema Análise/Diagnóstico. Também fornece as informações contidas na Definição.

- Definição: É o subsistema que mantém as informações necessárias para as monitorações. Contém dados de configuração (como e onde coletar os dados de auditoria, como responder as intrusões, etc) e dados de referência (*profiles* de comportamento esperado do sistema alvo e de seus usuários, históricos de monitoramento, estatísticas e assinaturas de ataques conhecidas). O AS fornece estas definições. As informações contidas na Definição são usadas pelo Controle para configurar as ferramentas de monitoração. Estas informações podem ser, também, usadas pelo subsistema Análise/Diagnóstico para comparar as informações coletadas do sistema alvo e de seus usuários com o comportamento esperado e com as assinaturas, ambos armazenados na Definição. As definições podem ser visualizadas e alteradas pelo AS. Uma outra funcionalidade é a capacidade de adicionar ou remover *hosts* monitorados, por meio de uma ordem do AS.

- Controle: É o subsistema responsável pela ativação e desativação dos *hosts* a monitorar e pela utilização das definições para determinar o comportamento das ferramentas de monitoração e interagir com o subsistema Coleta/Atuação para ativar e desativar suas entidades (monitores, transceptores e agentes). Ferramentas existentes podem ser removidas e modificadas, ativadas e desativadas, além do que, novas ferramentas podem ser adicionadas.

- Análise/Diagnóstico: É o subsistema que usa informações coletadas do sistema alvo e dados sobre o comportamento esperado do sistema/usuários, assinaturas, regras e ações fornecidas pela Definição para detectar problemas no comportamento do sistema/usuários e identificar ações conhecidas de ataques. Este subsistema pode decidir por duas ações: diretamente ativar a Atuação (do subsistema Coleta/Atuação) para corrigir problemas ou ativar alarmes para alertar o AS. Em ambos os casos, este subsistema armazena a informação para que esta seja visualizada posteriormente.

- Coleta/Atuação: É o subsistema que coleta informações no sistema alvo. As informações coletadas são usadas pelo subsistema Análise/Diagnóstico para formar uma análise do comportamento do sistema e compará-las com assinaturas conhecidas de ataque. Pode, também, agir no sistema alvo para modificar seu comportamento por uma ordem do Controle ou, resolver um problema detectado pelo subsistema Análise/Diagnóstico.

Os subsistemas são organizados em duas camadas:

- Camada Superior, à qual pertencem Controle, Definição e Visualização;
- Camada Inferior, Análise/Diagnóstico e Coleta/Atuação.

4.1.1 Diagrama de Fluxo de Dados

Como o modelo conceitual visa demonstrar a funcionalidade do IDS, um Diagrama de Fluxo de Dados foi utilizado para especificar a funcionalidade dos subsistemas e os relacionamentos existentes entre eles. A funcionalidade dos cinco subsistemas já foi mostrada, restando explicar o fluxo das informações entre os diversos subsistemas.

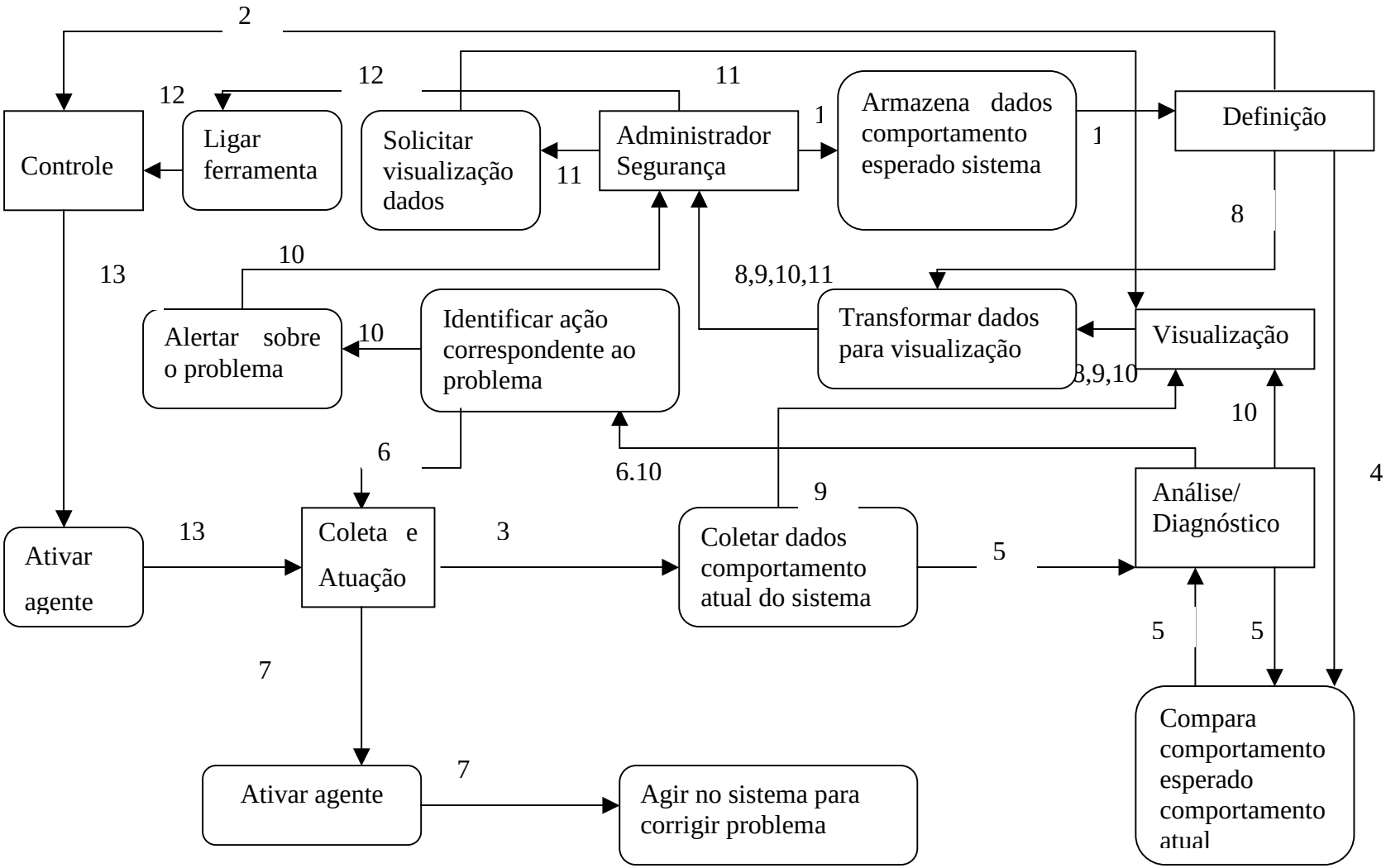

Figura 19: Diagrama de Fluxo de Dados.

Fluxo de Informação:

(1) Mensagem induzida do AS para mudança na base de dados do subsistema Definição. Exemplo: Inserção de novo host na rede, mudança de horários de acesso de usuários na rede (seguindo certas normas da política de segurança).

(2) Mensagem passada sob demanda do subsistema Controle vinda do subsistema Definição, para configurar a ferramenta. Este fluxo também representa uma mensagem induzida pelo subsistema Definição, representando alguma mudança realizada pelo AS na base de dados do subsistema Definição. Exemplo: Inserir uma nova técnica de detecção para ser utilizada como ferramenta ou atualização de assinatura de ataque para ser usada por um agente.

(3) Mensagem sob demanda do subsistema Coleta/Atuação, para coletar dados do comportamento atual do sistema alvo. Também neste mesmo fluxo, representa uma mensagem para armazenar dados do comportamento atual do sistema alvo. Exemplo: Checagem da permissão (400 ou 200 ou 100) no arquivo *rhosts* no sistema alvo, tráfego da rede, utilização da CPU, acesso a discos,etc.

(4) e (5) Mensagem sob demanda do subsistema Análise/Diagnóstico vinda do subsistema Coleta/Atuação e Definição, para fornecer dados do comportamento esperado do sistema alvo. Exemplo: Comparação do comportamento atual com o esperado do espaço em disco no *host* monitorado, *logon* de usuários, etc.

(6) Mensagem induzida do subsistema Análise/Diagnóstico para o subsistema Coleta/Atuação. Exemplo: Detecção de alteração no sistema alvo, ativa um agente para corrigir o problema do tipo “matar” um processo que está ocupando tempo de CPU.

(7) Mensagem induzida pelo subsistema Coleta/Atuação para o sistema alvo. Exemplo: “Matar” um processo.

(8) Mensagem sob demanda do AS vinda do subsistema Definição. Exemplo: Visualizar *hosts* que estão disponíveis para monitoração.

(9) Mensagem sob demanda do subsistema Visualização vinda do subsistema Coleta/Atuação. Exemplo: Visualização do resultado do monitoramento do comportamento atual do sistema alvo.
(10) Mensagem sob demanda do subsistema Visualização vinda do subsistema Análise/Diagnóstico. Exemplo: Visualizar o diagnóstico da análise.

(10) Mensagem induzida do subsistema Análise/Diagnóstico para AS. Exemplo: Detecção de um problema do tipo mudança no comportamento do sistema alvo, por exemplo quando o servidor de DNS não está rodando, deve ser passado o alerta ao AS.

(11) Mensagem induzida do subsistema Visualização para o AS. Exemplo: Alerta sobre o problema detectado do tipo alteração no arquivo etc/passwd, como um sintoma de um ataque.

(11) Mensagem sob demanda do AS vinda do subsistema Visualização. Exemplo: AS deseja ver o resultado da comparação do número de *logins* falhos, de determinado usuário, no *host* monitorado, com o número de *logins* falhos esperados/permitidos.

(12) Mensagem sob demanda do AS vinda do subsistema Controle e vice-versa (mensagem sob demanda do subsistema Controle para o AS). Exemplo: AS ativa agente de verificação de assinatura de ataque do tipo string para o UNIX, que pode ser *´´cat´´++``>/.rhosts``*, em determinados *hosts.* No segundo caso, o AS pode visualizar dados sobre as ferramentas de monitoração configuradas pelo subsistema Controle.

(13) Mensagem induzida do subsistema Controle para o subsistema Coleta/Atuação. Exemplo: Controle solicita que seja ativado o agente que monitora conexões para múltiplas portas em determinados *hosts*, como também a utilização da CPU.

4.2 Uso de agentes inteligentes em IDS

Abordagens baseadas em agentes para detecção de intrusão usam entidades de software que desempenham certas funções de monitoramento de segurança nos *hosts*, como monitoramento da coleta, análise ou resposta. Elas funcionam automaticamente, isto é, são controladas somente pelo sistema operacional e não por outros processos. Também rodam continuamente e cuja operação permite que os agentes aprendam por experiência, tão bem como comunicam e cooperam com outros agentes de construção similar.

Como visto, a abordagem de agentes é muito poderosa, devido às capacidades que podem ser embutidas nele. Seu projeto interno pode ser baseado numa variedade de paradigmas para análise de dados, como o casamento de padrões, o sistema baseado em regras e aprendizagem usando programação genética. Pode ser extremamente simples, contando o número de vezes que um comando particular é invocado num intervalo de tempo. Ou pode ser complexo, procurando evidências de ataques num ambiente particular. Pode ser utilizado como mecanismo de resposta ativa, agindo diretamente no sistema alvo, bem como, ser utilizado nas respostas passivas, à medida que o AS recebe os alertas, executando uma monitoração no *host*, por exemplo, interrompendo a sessão do usuário.

A utilização de agentes autônomos tem sido proposta por alguns autores como forma de construir sistemas de detecção de intrusão não-monolíticos. Foram identificadas algumas vantagens nos sistemas baseados em agentes autônomos em comparação com sistemas monolíticos (Crosbie & Spafford 1995a; 1995b):

- fácil configuração, uma vez que é possível se ter uma série de pequenos agentes especializados em tarefas específicas de detecção, o sistema de detecção pode ser configurado da forma mais adequada para cada caso;
- eficiência, pois agentes podem ser treinados previamente e otimizados para realizar suas tarefas da maneira a gerar a menor sobrecarga possível no sistema;
- capacidade de extensão, porque um sistema de agentes pode ser facilmente modificado para operar em rede e permitir migração para rastrear comportamentos anômalos através da rede, ou, mover-se para máquinas onde eles podem ser mais úteis;
- escalabilidade, para atuar em sistemas maiores, bastando adicionar mais agentes e aumentar sua diversidade;

- flexibilidade e robustez fornecida aos IDS. É flexível na maneira em que agentes podem ser inseridos, removidos e atualizados sem provocar um impacto significante na operação do sistema em tempo real. É robusto, de modo que, se isolado, ou se grupos de agentes são comprometidos, ou falham de alguma maneira, a operação do sistema maior não é necessariamente arriscada.

[Balasubramaniyan et al, 1998] liberou a segunda implementação de um ambiente, denominado *Autonomous Agents for Intrusion Detection* (AAFID2), utilizando-se diversos recursos de administração de sistemas e segurança comuns em ambiente Unix. Algumas entidades deste sistema foram testadas em laboratório, como parte prática desta dissertação e desempenham monitoramento de coleta e análise de dados.

Uma abordagem alternativa e promissora baseia-se em agentes inteligentes capazes de tomar decisões com mais autonomia e de migrarem entre dispositivos, possibilitando uma forma de monitoramento mais distribuída [Macedo, 2001]. Isto os diferencia das demais tecnologias cliente-servidor, já que não estão confinados no sistema onde começa sua execução. Agentes móveis são livres para serem executados em qualquer máquina do ambiente no qual estão inseridos. Ao chegar em um ponto remoto, o agente pode tomar a decisão (autonomia) de migrar para um novo computador, seguindo sua lista de itinerários. Agentes móveis possuem a habilidade de transportar a si próprios, como código, estado de execução, itinerário, etc, de um sistema para outro numa rede de computadores. Esta habilidade de viajar permite, a um agente móvel, movimentar-se para o sistema de agente que contém o objeto com o qual deseja interagir.

Embora as técnicas de inteligência artificial pareçam promissoras na área de detecção de intrusão, sua efetiva aplicação em IDS tem sido discreta.

4.3 Arquitetura do IDS

A arquitetura modular para um IDS descreve os módulos de implementação da ferramenta. É desenvolvida por meio do mapeamento no modelo conceitual já mostrado.

A arquitetura é composta de quatro módulos descritos abaixo e mostrada na figura 20:

1) Sociedade de Entidades: É responsável por implementar a funcionalidade dos subsistemas Controle, Coleta/Atuação e Análise/Diagnóstico descritos no modelo conceitual. Utiliza as ferramentas para efetuar testes de detecção e utiliza a base de dados para recolher informações de configuração e dados de referência (comportamento esperado do sistema). O monitoramento é desempenhado em três camadas separadamente. Os monitores deste módulo controlam os tranceptores e estes, por sua vez, os agentes.

    a) Monitores. São as entidades mais complexas com respeito à capacidade de comunicação. Um monitor deve ter todas as funcionalidades de um transceptor, pois também precisa ser capaz de controlar entidades locais e coordenar as atividades entre os transceptores. Deve dar início e controlar entidades remotas. Deve também ser capaz de dar início a novos transceptores, ou monitores em *hosts*, que não tenham um rodando, bem como enviar mensagens para eles. O monitor somente se comunica com monitores e transceptores, não o fazendo com agentes que possam estar rodando no *host.* O monitor deve ser capaz de escutar conexões de entidades remotas, porque pode ter iniciado um transceptor, ou outro monitor em um *host* precisar ser controlado por um monitor já existente. A nova entidade deve ser capaz de entrar em contato com o monitor e registrar-se enviando uma mensagem. Uma vez que a entidade remota é iniciada, ou um pedido de conexão foi processado, o monitor deve ser capaz de se comunicar com ela da mesma maneira que com entidades locais.

    b) Transceptores. Residem nos *hosts*. São responsáveis por receber os dados dos agentes, que rodam no mesmo *host* e controlar suas atividades, como iniciação ou terminação. Eles não precisam de capacidade de comunicação remota *downstream*, mas, de se comunicar com um grande número de entidades locais. Agentes podem fornecer dados e comandos, por exemplo, para o pedido de um módulo adicional necessário para execução.

Transceptores podem necessitar enviar um comando para um agente, por exemplo, configurar um valor de parâmetro. Também precisam ser capazes de responder a comandos vindos de monitores e fornecer, aos mesmos, dados. Se um transceptor recebe uma mensagem de um agente, a qual não é capaz de interpretar, a mesma deve ser enviada para uma entidade pai do transceptor para processamento adicional. Portanto, consolidam relatos e os remetem para os sistemas monitores. Cada transceptor controla todas as entidades em um *host*, mas não exercem influência sobre outros agentes em outros *hosts*. Existem técnicas para fazer isto, mas o sistema não as implementa.

c) Agentes. Agentes estáticos de softwares, que com a ajuda dos filtros (implementados como *threads* nos agentes), coletam informações do sistema onde estão rodando. Monitoram aspectos específicos de atividades nos *hosts* e relatam, para os seus transceptores, um comportamento anormal, que pode indicar problema de segurança. Algumas tarefas dos agentes são fornecidas pela infra-estrutura para criação de agentes para permitir novas entidades no menor tempo possível. Outras como inicialização, checagem e *cleanup* são específicas para cada agente. Eles são escritos off-line e realizam tarefas específicas, por exemplo, procurando por assinaturas de ataques específicas escritas também off-line e acrescentadas no SGBD. Exemplos de agentes que poderiam rodar no sistema utilizando diversos recursos de administração de sistemas e segurança comuns em ambiente Unix:

i) *Login* de usuários em horas estranhas;
ii) *Login* de usuários a partir de sites não familiares da rede.
iii) Tentativas falhas de *login* com senhas fracas, especialmente se ocorrerem múltiplas falhas;
iv) *Reboots* inexplicáveis ou, mudanças nos relógios do sistema;
v) Mensagens de erro não usuais de correios, *daemons* ou outros serviços;
vi) Uso não autorizado do comando *su*, para ganhar acesso de *root*;
vii) Espaço ocupado no diretório /tmp;
viii) Processo /etc/Shell;
ix) Espaço ocupado em disco;
x) Seqüência de processos;
xi) Permissões no arquivo rhosts, só aceita 400,200 e 100;
xii) Conexões para múltiplas portas (portscan);

xiii) Conexões para mesmo host de um mesmo serviço;

xiv) Carregamento CPU;

xv) Configuração FTP;

xvi) Conexões TCP.

2) Interface: Faz a interação entre o AS e os módulos SGBD, Caixa de Ferramentas e Sociedade de Entidades. É através da interação desta interface com o Monitor do Módulo de Sociedade de Entidades que se ativa e desativa os Transceptores e por sua vez os Agentes. As informações da definição do sistema podem ser acrescentadas, apagadas, modificadas ou visualizadas do SGBD, bem como as regras, assinaturas e as ações de respostas.

3) Caixa de Ferramentas: A ferramenta pode ser qualquer código ou técnica que ajuda as Entidades a executarem suas atividades. Esta Caixa atua como repositório para todas as informações que as entidades possam precisar. Isto inclui, principalmente, código e módulos de suporte. Por exemplo, um Transceptor pode ter sido iniciado com poucos recursos locais e, quando ele precisar de um módulo cujo código não está presente no local, será solicitado ao seu Monitor. Este deve ser capaz de definir o código necessário e solicitá-lo a Caixa de Ferramentas. O mesmo acontece se um Agente necessitar de um código adicional. Deve ser possível acoplar outras ferramentas a este módulo.

4) SGBD: O SGBD, além de armazenar a definição das configurações, dados de referência, regras, assinaturas e ações definidas pelo AS, mantêm também informações coletadas pelos agentes e resultados sobre o comportamento atual da rede.

A arquitetura é mostrada na figura 20:

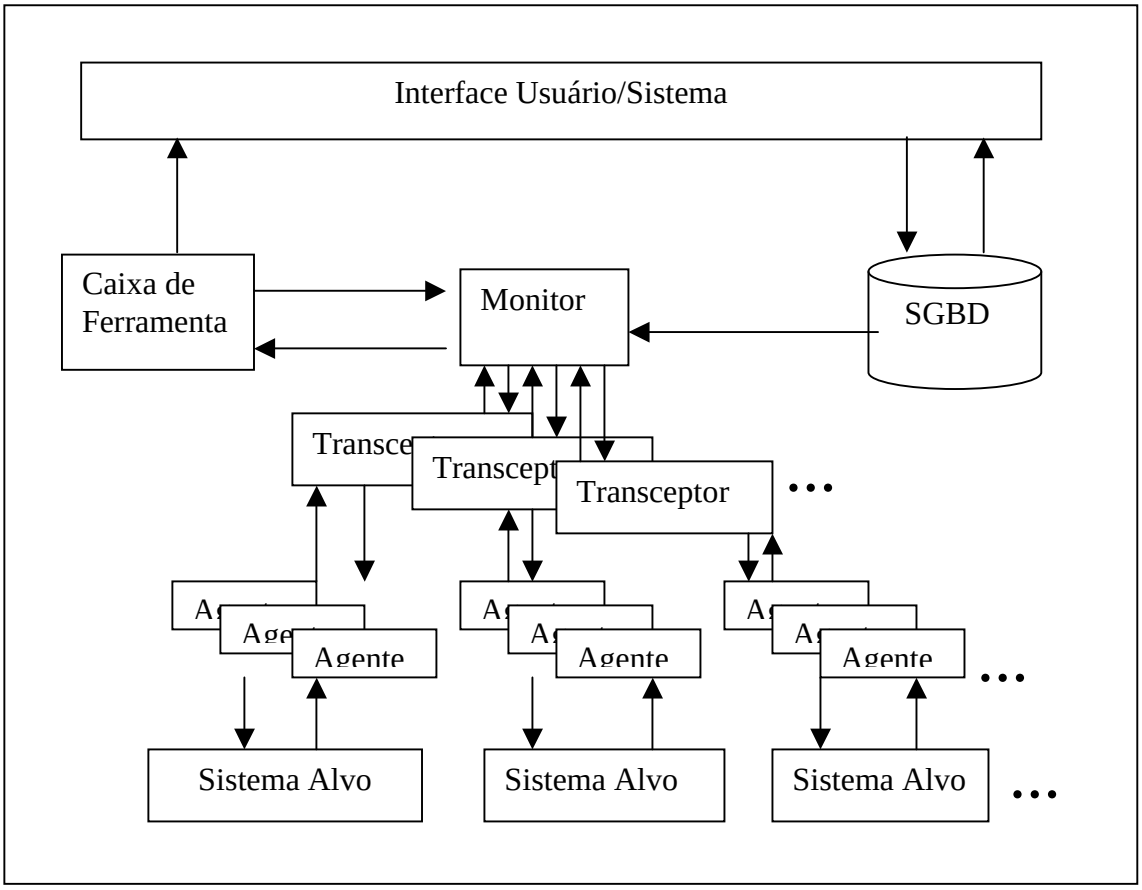

Figura 20: Arquitetura do IDS

A arquitetura foi derivada do modelo conceitual da seguinte maneira:

- A funcionalidade da interface do usuário, encontrada nos subsistemas Definição, Visualização e Controle, é implementada pelo módulo IUS.
- A funcionalidade do subsistema Controle é implementada pelos módulos, Monitor e Transceptor.
- A funcionalidade do Banco de Dados, requisitada pelos subsistemas Definição, Coleta/Atuação e Análise/Diagnóstico, é implementada pelo módulo SGBD.
- A funcionalidade de diagnóstico, identificação de solução e correção de problemas do subsistema Análise/Diagnóstico é implementada pelos módulos, Monitor, Transceptor e Caixa de Ferramentas.
- A funcionalidade da coleta de informações do subsistema Coleta/Atuação é implementada pelos módulos, Monitor, Transceptor, Agente e Caixa de Ferramentas.
- A funcionalidade de realização de ações do subsistema Coleta/Atuação é implementada pelos módulos, Monitor, Transceptor e Agente.

4.4 O que se espera deste modelo

Na ausência de uma tecnologia única que detecte ou resolva os problemas encontrados, a idéia é a existência de uma arquitetura com características de flexibilidade e extensibilidade, visando permitir a alteração ou a inclusão de novos métodos de detecção.

O módulo Sociedade de Entidades é construído de maneira hierárquica, composto de entidades que se referem a qualquer dos componentes definidos na mesma. Inclui agentes, transceptores e monitores. Cada entidade é um objeto separado, que conserva seu próprio estado interno e comportamento de acordo com sua função no sistema. São programas que rodam de forma independente.

O módulo Interface, como um programa que interage com a Sociedade de Entidades, também pode ser considerado entidade superior na hierarquia.

O módulo Caixa de Ferramentas pode implementar técnicas mais aprimoradas, tais como:

- Programação genética, no reconhecimento da dinamicidade do comportamento do usuário.
- Técnica para reter o estado entre sessões. Estados isolados podem não representar um ataque, no entanto, quando vistos em seqüência, podem permitir que sejam detectados ataques a longo prazo ou, mudanças no comportamento do usuário/sistema;
- Técnica de redução de dados, para lidar com volume de dados menor por questões de segurança.
- Rede Neural Artificial, para reconhecer leves variações nas assinaturas de ataques.
- Imunologia Computacional.

Monitores podem detectar eventos não notificados por transceptores, podendo também controlar outros Monitores, localizados em outros *hosts*. No entanto, se implementados de maneira hierárquica, podem sofrer com o problema de consistência da informação. Mesmo esta distribuição de controle do IDS, ainda mantém-se o ponto de falha do sistema num nível mais elevado.

Filtros podem ser implementados como *threads* nos agentes para ajudar na coleta de dados. Agentes diferentes podem necessitar que sejam coletados dados, de uma mesma origem, simultaneamente. Neste caso, os filtros ajudam na passagem dos dados para os agentes.

O arranjo hierárquico observado na arquitetura, onde entidades num nível mais baixo desempenham parte do trabalho e transmitem seus resultados para entidades mais altas na hierarquia, talvez seja a única maneira de manter os sistemas escaláveis.

O modelo proposto contempla 3 (três) abordagens que podem ser encontradas na taxonomia proposta:

- Coletas não podem ser feitas de uma única origem. Deve-se dar o suporte para que o IDS atue em origens diferentes;
- Não existe um método único de detecção que analise todos os problemas. Deve-se dar o suporte ao IDS para que o mesmo permita a inclusão de novos métodos de análise de dados para detecção de intrusão;
- A existência de um outro meio de resposta à intrusão que não utiliza somente respostas automáticas. Um mecanismo que permita que AS programe suas respostas para atuarem através de ferramentas.

Um IDS não deve somente se preocupar em como a intrusão será detectada, nas técnicas utilizadas na análise, como objetivo principal de um IDS. A tarefa de detecção é baseada também em decisões como [Zamboni, 2000]:

- O que será detectado?
- Como os dados serão coletados?
- O que será feito com o resultado da análise?

A maioria dos modelos de IDS estudados, que compõe o *survey*, apresenta coleta de uma única origem, não emprega agentes na coleta de dados nem apresentam visualização dos resultados da análise. A arquitetura proposta se preocupou igualmente com a coleta de diversas origens, com a diversidade de técnicas de análise e a visualização dos resultados da análise.

Na prática, no entanto, o projeto de um IDS pode não contemplar todas as funcionalidades apresentadas, devido a restrições impostas pelo ambiente no qual o IDS vai operar ou pela

política de segurança implantada. Um projeto de um IDS pode ser dirigido aos dados disponíveis para coleta, isto vai influenciar ou determinar as técnicas de análise e as capacidades de detecção. Ou pode ser dirigido a uma técnica de análise específica, neste caso os dados necessários para coleta serão determinados e suas respectivas origens indicadas.

4.5 Análise do modelo proposto numa organização

O modelo proposto poderia ser analisado levando-se em consideração a localização dos IDS numa organização. A base conceitual para análise é a taxonomia, os critérios nela definidos e suas respectivas características compõem uma sólida estrutura para escolha de um IDS a ser aplicado numa organização. Foi tomada como base uma rede com seus recursos públicos, privados e internos bem como os mecanismos de segurança implementados como na figura 21 [Geus, 2002]. Diante do posicionamento dos 5 (cinco) IDS nos cinco segmentos de rede da organização, eles seriam vistos segundo:

- a localização da informação de origem;
- o mecanismo de coleta;
- o método de detecção;
- a aprendizagem;
- a técnica utilizada;
- o mecanismo de resposta;
- a arquitetura do IDS.

Os 5 IDS baseados em rede especificados como: IDS1, IDS2, IDS3, IDS4 e IDS5, aparecem na figura 21 respectivamente como: 1, 2, 3, 4 e 5. Os 6 IDS baseados em host especificados como: IDSA, IDSB, IDSC, IDSD, IDSE e IDSF aparecem também na figura 21, ao lado do servidor no qual serão instalados como: A, B, C, D, E e F respectivamente.

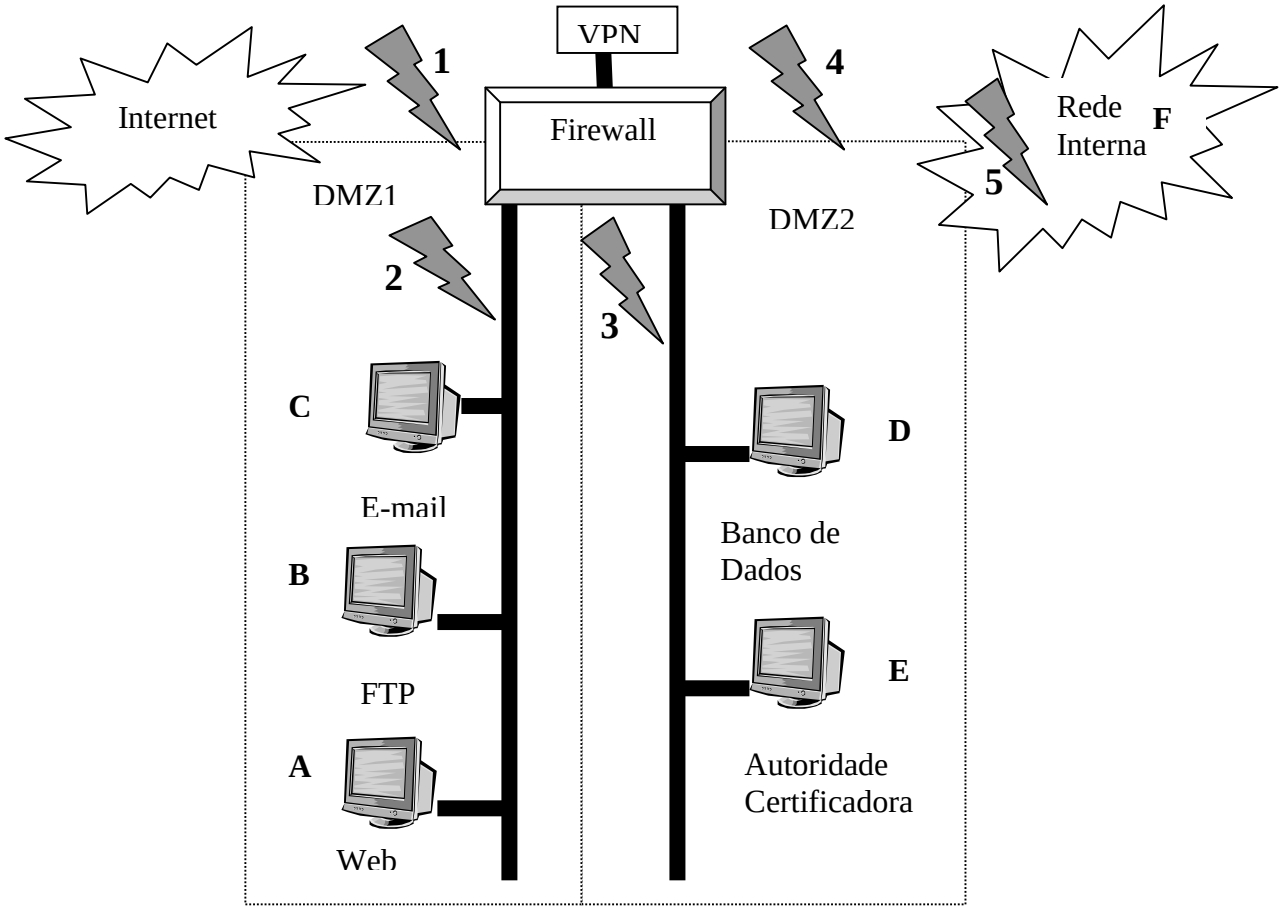

Figura 21: Localização de mecanismos de segurança numa organização[19].

IDS1 com o objetivo de detectar todas as tentativas de ataque contra a rede da organização, até mesmo as tentativas que não teriam efeito algum, como ataques a servidores web inexistentes. Localizado no segmento de rede antes do firewall, seria implementado como:

- com origem da informação na rede; com sensores de rede que possam filtrar por strings, ou protocolos específicos, baseado em conhecimento que utilizam a modelagem de regras à procura de assinaturas de ataques, com aprendizagem programada.

IDS2: com o objetivo de detectar tentativas de ataque contra recursos públicos localizados na zona desmilitarizada (DMZ1), como servidores de *e-mail*, *ftp*, *web*; que passam pelo *firewall*,

[19] Foi utilizado um único componente de firewall, com 5 (cinco) interfaces de rede e utilização da VPN em uma interface dedicada do firewall.

mas, no entanto não implica no comprometimento da rede da organização; seria implementado nos padrões do IDS1.

IDS3: com o objetivo de detectar tentativas de ataque contra recursos privados localizados numa segunda zona desmilitarizada (DMZ2), por exemplo servidor de banco de dados, que passam pelo *firewall*, pela VPN e por algum serviço da primeira zona desmilitarizada, como por exemplo servidor *web*. Localizado no segmento de rede atrás do *firewall* e seria implementado como:

- com origem da informação na rede; com sensores de rede, baseado em comportamento (por ser um acesso mais privado nesta DMZ2, pode-se trabalhar em cima do comportamento do sistema/usuário) que utiliza a modelagem de estado, com aprendizagem programada.

IDS4 com o objetivo de detectar tentativas de ataque contra recursos internos da organização, que passam pelo *firewall* e que podem vir pela VPN, seria implementado nos padrões do IDS3.

IDS5: com o objetivo de detectar tentativas de ataque contra recursos internos da organização seria implementado como:

- com origem da informação na rede, com sensores de rede, baseado em comportamento que utilizam técnicas de inteligência artificial, aprendizagem automática.

É necessário, também, o posicionamento de IDS baseado em *host* na organização, para aumentar a segurança de *hosts* específicos, como por exemplo:

A. Servidor *web*;
B. Servidor FTP;
C. Servidor e-mail;
D. Servidor de banco de dados;
E. Servidor autoridade certificadora;
F. Servidor de aplicativos.

Os IDS A, B, C, D, E e F seriam implementados como:

- Origem da informação baseada em *host*, sensores de *host*, baseado em comportamento que utilizam técnicas de IA, aprendizagem automática.

Estes IDS descritos nos cinco segmentos de rede e localizados na organização seriam implementados no modelo como entidades de IDS, da seguinte forma:

Uso de uma máquina no segmento 5 de rede como a Interface do Usuário do Sistema. O monitor é ativado nesta máquina e este ativa os transceptores para os segmentos de rede 1, 2, 3 e 4 e para o próprio segmento. O transceptor, por sua vez, ativa os agentes específicos para detectar os ataques nos respectivos segmentos. Estas entidades podem ser implementadas nos mesmos padrões dos IDS1, IDS2, IDS3, IDS4 e IDS5. O monitor controla o transceptor da própria máquina bem como aqueles ativados remotamente. Os transceptores apenas controlam os agentes que foram ativados por ele, ou seja, no local.

O monitor também ativa transceptores para trabalharem nos servidores localizados nas duas DMZ e em *hosts* instalados na rede interna, que queiram detectar intrusões. Os transceptores ativam agentes específicos para detectar intrusões nos hosts no qual estão instalados. Estas entidades podem ser implementadas, também, nos mesmos padrões dos IDS descritos como IDSA, IDSB, IDSC, IDSD, IDSE e IDSF.

O mecanismo de resposta do modelo pode ser ativo nas implementações que utilizam a origem de dados baseada em rede e passivo quando utilizam o *host* como origem de dados.

A arquitetura é distribuída, tanto na coleta de dados como na análise.

4.6 Conclusão

A idéia em torno deste modelo é a visão de:

- Transceptores em cada *host* monitorado como um IDS baseado em *host*, inclusive detectando intrusões ao nível de aplicação, com a capacidade de controle e processamento. Podem ser vistos como interface externa de comunicação do *host*;
- Transceptores como IDS baseados em rede monitorando todos os pacotes direcionados aos *hosts* localizados em segmentos da rede;
- Os Monitores com capacidade de controle e processamento, podendo correlacionar a informação dos Transceptores. Estando em um nível mais elevado, percebe ataques não detectados em um nível mais baixo da hierarquia.

- Agentes estão nas folhas da árvore, portanto, somente enviam mensagens para cima. A entidade localizada na raiz da árvore é sempre um monitor. Mas este pode aparecer também, em níveis intermediários. A única entidade que pode estar acima do monitor localizado na raiz é a interface do usuário, ou algum outro programa que controle todo o IDS.
- As ferramentas da Caixa de Ferramentas podem ser inseridas, removidas ou alteradas, dando modularidade à arquitetura.

O ponto de demarcação que, tipicamente, existe entre o monitoramento de detecção de intrusão e ambiente alvo é transparente neste modelo. Transceptores e Agentes podem trabalhar localmente monitorando o *host* local como seu sistema alvo bem como serem ativados em *host* remoto, em um ponto de estrangulamento da rede, monitorando todos os pacotes direcionados aos *hosts* desta rede.

O modelo distribuído da Sociedade de Entidades e o módulo Interface utilizam uma arquitetura hierárquica que provê facilidade de escalabilidade e manutenibilidade em oposição a arquitetura centralizada apresentada na arquitetura do I-DREAM.

Construir sistemas com processamento distribuído talvez seja a única maneira de torná-los escaláveis. Esta descentralização de processamento permite que mais de uma ferramenta ou mais de um analisador seja usado de maneira distribuída.

Mesmo com o processamento distribuído ainda se mantém o AS (Administrador de Segurança do Sistema), para receber e atuar nos alarmes. A noção de que detecção de intrusão pode tornar-se completamente automática é controvertida. Certamente os IDS exigem extensivo suporte humano, particularmente em termos de correlação e interpretação de informação, mesmo que os sistemas ganhem funcionalidade.

As entidades do módulo Sociedade de Entidades, da arquitetura proposta, podem ser implementadas como programas separados. Uma proposta seria a implementação dos Agentes como *threads* nos Transceptores. Este mecanismo pode acelerar a execução da detecção, eliminando boa parte da sobrecarga envolvida na execução dos Agentes.

Os módulos, Sociedade de Entidades e Interface, apresentam-se de maneira hierárquica. Assim, concede, naturalmente, hierarquia a um modelo de comunicação. Os níveis que estão mais próximos da raiz da hierarquia são identificados como *upstream* e, os que estão mais próximos das folhas da árvore, como *downstream*. Uma entidade está acima da outra se pode ser alcançada pelos caminhos de comunicação *upstream* e, abaixo, se pode ser alcançada pelos caminhos de comunicação *downstream.* Uma entidade somente pode ter comunicação direta com entidades que estejam imediatamente acima e abaixo dela própria.

Uma entidade pode precisar de uma ferramenta, que esteja faltando, para poder desempenhar suas funções. Neste caso a entidade envia uma mensagem para cima especificando a ferramenta que falta. O módulo Caixa de ferramentas disponibilizará a ferramenta necessária para o trabalho da entidade. Este mecanismo traz flexibilidade à arquitetura e leveza das entidades. Também permite o módulo Sociedade de entidades ser limitado a ferramentas essenciais e, então, ter todas as outras, desdobradas quando necessárias.

O protocolo utilizado de comunicação entre *hosts* é o TCP. O único mecanismo de segurança de comunicações corrente previsto para ser utilizado no modelo é o *ssh* para monitorar o *host* remotamente. Contudo, nenhuma autenticação ou criptografia é feita depois deste ponto.

Capítulo 5

Considerações Finais

Esta dissertação apresentou uma proposta de uma nova taxonomia dos IDS, quanto às máquinas de coleta, análise e resposta. Também é apresentado um *survey* de vinte (20) IDS, escolhidos pela diversidade de mecanismos neles implementados. A comparação dos sistemas foi feita baseando-se nas taxonomias apresentadas. Por último e não menos importante, a partir do estudo taxonômico, foi mostrada uma nova abordagem para um modelo de um IDS, que utiliza um modelo conceitual já validado por aplicação na área de Administração de Sistemas.

5.1 A Detecção de Intrusão

A pesquisa e o desenvolvimento da tecnologia de detecção de intrusão vêm se desenvolvendo desde 1980 [Anderson, 1980], proporcionando estratégias de solução para aperfeiçoar seus objetivos, bem como restrições para suportar. Fica claro que tecnologia de detecção de intrusão é uma parte de um programa completo de segurança computacional.

Identificou-se a importância de um extenso conhecimento nas tecnologias existentes para, através de suas vantagens e limitações, identificar sua funcionalidade, aplicação e possível tomada de decisão. Ainda existem muitos desafios a alcançar, principalmente no que diz respeito a:

- Educação do usuário. Muitos, se não a maioria dos ataques, ocorrem através de engenharia social.

- A noção de que detecção de intrusão pode tornar-se completamente automática é controvertida. Certamente os IDS exigem extensivo suporte humano, particularmente em termos de correlação, interpretação de informação e tratamento de alarmes, mesmo que os sistemas ganhem funcionalidade.

5.2 Contribuições

Resumidamente, este trabalho teve como contribuições:

- Apresentação dos conceitos básicos sobre detecção de intrusão, sua finalidade, as formas de ataques às redes computacionais, as vantagens e limitações no emprego de mecanismos de detecção de intrusão e identificação de seus principais componentes funcionais;
- Apresentação de uma taxonomia dos IDS diante das principais características encontradas nos 3 (três) módulos principais de um IDS: coleta de dados, análise e componente de resposta.
- Apresentação do Estado da Arte em termos de Sistemas de Detecção de Intrusão, abordando uma grande variedade de mecanismos conhecidos até a presente data;
- Apresentação de um *Survey* dos IDS levando-se em consideração a nova taxonomia proposta. Uma tabela comparativa dos IDS, mostrando características ainda não analisadas em outros *surveys* de IDS e também importantes como, método de coleta, aprendizagem e arquitetura;
- Identificação das vantagens e limitações dos mecanismos empregados na detecção. Tanto mecanismos de coleta como mecanismos de análise e resposta.
- Definição de uma arquitetura extensível, hierárquica e escalável, facilitada pela capacidade de fácil inserção de novos agentes e de novas assinaturas de ataque, à medida que novas vulnerabilidades são conhecidas.

Os testes com o protótipo AAFID2 demonstraram que um ambiente com agentes autônomos é possível, desempenhando o nível mais baixo da hierarquia. Também demonstrou que podem trabalhar independente do sistema operacional do sistema alvo e de suas interfaces com os dispositivos de rede. Uma contribuição desta dissertação, ao projeto COAST do Departamento de Ciência da Computação da Universidade de *Purdue*, foi o retorno dos resultados obtidos nas máquinas com sistema operacional AIX, onde nunca tinham sido testadas.

A observação de testes com o SNORT mostraram que, na prática, um IDS que implementa regras, que filtram assinaturas de ataques, é melhor em termos de:

- Alarmes são relativos ao que é realmente conhecido ser intrusivo, em oposição aquele alerta ao que é comportamento supostamente intrusivo.
- Codificar regras é mais fácil que codificar exemplos de intrusão e também não necessita de período de treinamento do IDS.

No entanto exige extensivo suporte humano de Administrador de Sistemas experiente, particularmente em termos de interpretação de *logs*, identificação de assinaturas, criação de filtros e atualização da base de regras do IDS. A experiência e o treinamento adequado de um Administrador de Sistemas podem ser decisivos para uma interpretação correta das informações derivadas dos IDS, para que possam tomar ações apropriadas quando uma intrusão é identificada, assegurando um resultado positivo.

A contribuição dada ao I-DREAM [Silva e outros, 1997], na definição de uma arquitetura distribuída para o Gerente e um arranjo hierárquico para a Sociedade de Agentes, provê facilidade de escalabilidade e manutenibilidade em oposição àquela arquitetura, apresentada anteriormente, centralizada no Gerente e a Sociedade de Agentes num único nível.

5.3 Trabalhos Futuros

Como trabalho futuro para esta pesquisa é proposto:

- Validar o modelo proposto segundo os requisitos de [Barrus, 1997] apresentados no capítulo 1.
- Implementar um protótipo funcional que possa incluir o desenvolvimento de ferramentas, como módulos independentes, provendo cenários avançados de detecção de intrusão, como por exemplo:

    - introduzindo o conceito de algoritmos genéticos para avaliação do comportamento do usuário/sistema;

  - com tecnologia de redes neurais como um sistema de respostas com tratamento de alarmes;
  - utilizando tecnologia de Imunologia computacional para cuidar da segurança do próprio IDS;
  - agentes com características de mobilidade, como forma de construir arquiteturas distribuídas.

- Um estudo mais aprofundado da taxonomia, no qual se inclua a análise de dependências entre as características que aparecem na mesma. Por exemplo, características relacionadas a um determinado mecanismo de coleta tendem a aparecer em IDS, quando a localização de coleta apresenta determinada característica. Ou ainda, uma técnica específica possui uma maior tendência de atuar em IDS que implementa determinada aprendizagem ou método específico.

Referências Bibliográficas

[Allen et al., 1998] John Kochmar, Julia Allen, Christopher Alberts, Cory Cohen, Gary Ford, Barbara Fraser, Suresh Konda, Klaus-Peter Kossakowski and Derek Simmel. *Preparing to Detecting Signs of Intrusion*. CMU/SEI, Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1998. http://www.cert.org/security-improvement/modules/m05.html.

[Allen et al., 1999a] Julia Allen, Alan Christie, William Fithen, John McHugh, Jed Pickel, Ed Stoner. *State of the Practice of Intrusion Detection Technologies*, CMU/SEI 1999, TR 028. www.sei.cmu.edu/publications/documents/99.reports/ 99tr028/99tr028abstract.html

[Allen et al., 1999b] Klaus-Peter Kossakowski, Julia Allen, Christopher Alberts, Cory Cohen, Gary Ford, Barbara Fraser, Eric Hayes, John Kochmar, Suresh Konda, and William Wilson. *Responding to Intrusion*. CMU/SEI, Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. http://www.cert.org/security-improvement/modules/m06.html.

[Amoroso, 1999] Edward G. Amoroso. *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps and Response*. Intrusion.Net Books, 1999.

[Anderson, 1980] James P. Anderson. *Computer Security Threat Monitoring and Surveillance*. Fort Washington, PA, April 1980.

[Asaka, 1999] Asaka, M., Taguchi, A., and Goto, S. *The Implementation of IDA: An Intrusion Detection Agent System*. In Proceedings of the 11th Annual FIRST Conference on Computer Security Incident Handling and Response (FIRST'99). www.ipa.go.jp/STC/IDA/paper/ida-install-e.pdf .

[Axelsson, 1999] Stefan Axelsson. *Research in Intrusion Detection Systems: A Survey*. 1999.

[Axelsson, 2000] Stefan Axelsson. *Intrusion detection systems: A Survey and Taxonomy*. 2000.

[Babbie, 1999] Babbie, Earl, *Métodos de pesquisa de survey*, Editora UFMG, 1999.

[Bace, 2000] Rebecca Gurley Bace. *Intrusion Detection*. Macmillan Technical Publishing, 2000.

[Balajinath and Raghavan, 2000] B. Balajinath, S. V. Raghavan. *Intrusion Detection Through Learning Behavior Model*. Computer Communications 24, páginas 1202-1212, 2001. www.netlab.cs.iitm.ernet.in/publications/ refereed-journal.html.

[Barber, 2001] Richard Barber. *The Evolution of Intrusion Detection Systems* – The next Step. Computers & Security 20, 132-145, 2001.

[Barrus, 1997] Joseph D. Barrus. *Intrusion Detection in Real Time in a Multi-Node Multi-Host Environment*. Master's Thesis, Naval Postgraduate School, Monterey, CA, september 1997.

[Barrus, 1998] Joseph D. Barrus and Neil C. Rowe. *A Distributed Autonomous-Agent Network-Intrusion Detection and Response System.* Command and Control Research and Technology Symposium, Monterey CA, June-July 1998.

[Bernardes, 1999] Mauro Cesar Bernardes. *Avaliação do Uso de Agentes Móveis em Segurança Computacional*. Dissertação de Mestrado. Instituto de Ciências Matemática e de Computação - ICMC/USP, novembro 1999.

[Borchardt e Mazieiro, 2001] Mauro A. Borchardt e Carlos Mazieiro, *Verificação da Integridade de Arquivos pelo Kernel do Sistema Operacional*, 2001.

[Cannady and Harrell, 1996] James Cannady, Jay Harrell. *A comparative analysis of current intrusion detection technologies*, 1996.

[Cannady, 1997] James Cannady. *Artificial neural networks for misuse detection*. 1997.

[Cansian, 1997a] Adriano Mauro Cansian. *Desenvolvimento de um Sistema Adaptativo de Detecção de Intrusos em Redes de Computadores*. Tese de doutorado, Instituto de Física de São Carlos/USP, 1997.

[Cansian, 1997b] *ACME! Advanced Counter Measures Environment, Um mecanismo de Captura e Análise de Pacotes para Aplicação em Detecção de Assinaturas de Ataque*. http://www.acme-ids.org/.

[Cansian, 1999] Adriano Mauro Cansian. *Detecção de Intrusão em Redes de Computadores*. UNESP – Universidade Estadual Paulista. Simpósio sobre Segurança da Informação, ITA, CTA, 1999.

[Cansian, 2000] Adriano Mauro Cansian. *Crime na Internet: Conhecendo e Observando o Inimigo*. SSI, 2000.

[Carnut e outros, 2001] Cristiano Lincoln Mattos, Evandro Curvelo Hora e Marco Antonio Carnut. *Desvendando Vulnerabilidades de Rede*. Cesar/Tempest Security Technologies, Cin/UFPE, 2001.

[Cheswick and Bellovin 94] William Cheswick and Steven Bellovin. *Firewalls and Internet Security*. Addison-Wesley, 1994.

[Cunningham and Lippmann, 2000] Richard P. Lippmann and Robert K. Cunningham. *Improving Intrusion Detection Performance Using Keyword Selection and Neural Networks*. Computer Networks 34, 597-603, 2000.

[Debar et al, 1992] Debar, Hervé, Beckerand Monique, Siboni, Didier. *Hiperview: An Intelligent security supervisor*. In Proceedings of the 2nd International Conference on Intelligence in Networks. Bordeaux. França, march 1992.

[Debar et al., 1999] Hervé Debar, Marc Dacier, Andreas Wesp. *Towards a Taxonomy of Intrusion Detection Systems*. Computer Networks 31, páginas 805-822, 1999.

[Denning, 1987] Denning, D., *An Intrusion-Detection Model*. IEEE Transaction on Software Engineering, Vol. SE-13, No. 2, 1987.

[Denning, 1999] Dorothy Elizabeth Denning. *Information Warfare and Security*. Addison-Wesley Longman, Inc., 1999.

[Endler, 1998] Markus Endler. *Novos Paradigmas de Interação Usando Agentes Móveis*. Departamento de Ciência da Computação, IME – USP. SBRC, 1998.

[Filho, 1997] Elson Felix Mendes Filho. *Algoritmos Genéticos*. LABIC-ICMC, USP São Carlos http://www.icmsc.sc.usp.br/~prico/gene1.html.

[Firth et al., 1997] Robert Firth, Gary Ford, Barbara Fraser, John Kochmar, Suresh Konda, John Richael, Derek Simmel. *Detecting Signs of Intrusion. Secutity Improvement Module*, CERT Coordination Center. http://www.cert.org/security-improvement/modules/m01.html.

[Flood and Carson, 1993] Robert Flood and Ewart Carson. *Dealing with complexity – An introduction to the theory and application of system science*, chapter 8, pages 151-158. Plenum Press, New York, second edition, 1993.

[Forrest, 1997] Anil Somayaji, Steven A. Hofmeyr, Stephanie Forrest. *Computer Immunology*, Communications of the ACM 40 (10) 88-96, 1997.

[Forrest, 1999] Steven A. Hofmeyr, Stephanie Forrest. *Architecture for an Artificial Immune System*, Department of Computer Science, UNM, Albuquerque, NM, Santa fe Institute. http://www.cs.unm.edu/~judd/lisys/.

[Franklin, 1997] Danielle Mattos Franklin. *I-DREAM Um Sistema de Monitoração de Recursos e Aplicações Baseado em Intranet*, Dissertação de Mestrado CIn/UFPE, 1997.

[Frincke and Huang, 2000] Deborah A. Frincke and Ming-Yuh Huang. *Recent Advances in Intrusion Detection Systems*. Computer Networks 34, 541-545, 2000.

[Frisch, 1995] Aeleen Frisch. *Essential System Administration*. O'Reilly & Associates, Inc, 1995.

[Furlan, 1998] José Davi Furlan. *Modelagem de Objetos através da UML*, 1998.

[Geus, 1999a] Paulo Lício de Geus. *Componentes da Segurança: Aspectos de Segurança do Linux2.2.*, SSI1999.

[Geus, 1999b] Paulo Lício de Geus. *Comparação entre Filtros de Pacotes com Estados e Tecnologias Tradicionais de Firewall*, 1999 – SSI1999.

[Geus, 2000] Paulo Lício de Geus.*Teias de Confiança*, 2000. SSI2000.

[Geus, 2002] Emílio Tissato Nakamura e Paulo Lício de Geus. *Segurança de redes em ambientes cooperativos* – Editora Berkeley, 2002.

[Heady, 1990] R. Heady, G. Luger, A. Maccabe, M. Servilla, *The Architecture of a Network Level Intrusion Detection System*, 1990.

[Hora, 1999] Evandro Curvelo Hora. *Sobre a Percepção Remota de Sniffers para Detectores de Intrusão em Redes TCP/IP*, Dissertação de Mestrado CIn/UFPE, 1999.

[Huang et al., 1999] Ming-Yuh Huang, Robert J. Jasper, Thomas M. Wicks. *A Large Scale Distributed Intrusion Detection Framework based on Attack Strategy Analysis*. Computer Networks 31, 2465-2475, 1999.

[Ingram, 1999] Dennis J. Ingram. *Autonomous agents for distributed intrusion detection in a multi-host environment*. Dissertação de mestrado da Naval Postgraduate School, 1999.

[Javitz, 1991] H. S. Javits and Valdes. *The SRI IDES Statistical Anomaly Detector*, 1991. http://www.sdl.sri.com/projects/nides/index5.html.

[Javitz, 1993] Javitz, H. S., and A. Valdes. *The NIDES Statistical Components Description and Justification*, 1993. http://www.sdl.sri.com/projects/nides/index5.html.

[Jeremy, 1994] Frank Jeremy. *Artificial Intelligence and Intrusion Detection: Current and Future Directions*. University of California at Davis, 1994.

[Jonsson and Lundin, 2000] Emile Lundin, Erland Jonsson. *Anomaly-based Intrusion Detection: Privacy Concerns and other Problems*. Computer Network 34, 623-640, 2000.

[Lee and Stolfo, 1998] Wenke Lee and Salvatore Stolfo. *Data Mining Approaches for Intrusion Detection*. Columbia University, New York, 1998. www.cs.columbia.edu/~sal/hpapers/USENIX/usenix.html.

[Lunt, 1988] Teresa F. Lunt. Automated *Audit Trail Analysis and Intrusion Detection: A Survey*. 11th National Computer Security Conference, 1988.

[Macedo, 2001] Hendrik Teixeira Macedo. *Mobilidade autonomia e distribuição de agentes para o gerenciamento corporativo de sistemas.* Dissertação de Mestrado CIn/UFPE, 2001.

[Mé, 1998] *GASSATA A Genetic Algorithm as an Alternative Tool for Security Audit Trail Analysis*. First International Workshop on the Recent Advances in Intrusion Detection, Belgium, 1998. www.supelec-rennes.fr/rennes/si/ equipe/lme/PUBLI/raid98.pdf.

[Mukherjee et al, 1990] Biswanath Muwherjee, L. Todd Heberlein, Karl N. Levitt, Gihan V. Diaz, Jeff Wood and David Wolber. *A Network Security Monitor*. IEEE Network, pp. 296 – 304, 1990. http://seclab.cs.ucdavis.edu/papers/pdfs/th-gd-90.pdf.

[Mukherjee et al., 1991] J Brentano, G. Dias, T. Goan, T. Heberlein, C. Ho, K. Levitt, B. Mukherjee, S. Smaha, T. Grance, D. Teal and D. Mansur. *DIDS Distributed Intrusion Detection System – Motivation, Architecture and an Early Prototype*. Proceeding of the Fourteenth National Computer Security Conference, Washington, 1991. seclab.cs.ucdavis.edu/papers/DIDS.ncsc91.pdf.

[Mukherjee et al., 1994] Biswanath Muwherjee, L. Todd Heberlein and Karl N. Levitt. *Network Intrusion Detection*. IEEE Network, vol. 8-3, pp. 26-41, 1994.

[Neumann, 1985] Neumann, P. G., *Audit Trail Analysis and Usage Collection and Processing*. Technical Report Project 5910, SRI International.

[Northcutt, 1999] Stephen Northcutt. *Network Intrusion Detection: An Analyst's Handbook*. New Riders Press, 1999.

[Northcutt, 2001] Stephen Northcutt, Judy Novak, Donald McLachlan. *Segurança e prevenção em redes* – Editora Berkeley, 2001.

[Nilsson, 1996] Nils J. Nilsson. *Introduction to Machine Learning*, 1996.

[Porras and Neumann, 1997] P.G. Neumann and Phillip A. Porras. *EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances*. In Proceedings of the First USENIX Workshop on Intrusion Detection and Network Monitoring, pages 73-80, Berkeley, CA, 1997. www.csl.sri.com/emerald/index.html

http://www.nfr.com/products/NID/docs/NID_Technical_Overview.pdf.

[Reami, 1998] E. R. Reami. *Especificação e Prototipagem de um Ambiente de Gerenciamento de Segurança Apoiado por Agentes Móveis*. Dissertação de Mestrado, ICMC/USP, 1998.

[Richards, 1999] Kevin Richards. *Network based Intrusion Detection: A Review of Technologies*. Computers & Security 18, 671-682, 1999.

[Roesh, 1997] M Roesh. *Snort – Ligtweight Intrusion Detection for Networks*, 1997. http://www.snort.org

[Schneier, 2000] Bruce Schneier. *Secrets & Lies - Digital Security in Networked World*. Wiley Computer Publishing, 2000.

[Schonlau and Theus, 2000] Matthias Schonlau, Martin Theus. *Detection Masquerades in Intrusion Detection based on Unpopular Commands*. Information Processing Letters 76, 33-38, 2000.

[Silva e outros, 1997] Fabio Silva, Danielle Franklin, Luciana Varejão, Fabiana Marins, André Santos. *I-DREAM: An Intranet-baseD REsource and Application Monitoring System.* Revista de Informática Teórica e Aplicada, vol. 4, n° 2, pp. 49-60, 12/1997.

[Sodirey et al., 1996] M. Sobirey, B. Richter and H. Konig. *The Intrusion Detection System – AID: Architecture and Experiences in Automated Audit Analysis*. Proceeding of

Internacional Conference on Comunications and Multimedia Security, Germany, 1996. http://www-rnks.informatik.tu-cottbus.de/~sobirey/aid.e.html.

[Sommer, 1999] Peter Sommer. *Intrusion Detection Systems as Evidence*. Computer Networks 31, 2477-2487, 1999.

[Spafford, 1993] Gene H. Kim and Eugene H. Spafford. *The design and implementation of Tripware: A file system integrity checker*, Purdue Technical Report CSD – TR – 93-071, November 1993.

[Spafford, 1994] Farmer, D., and E. Spafford The *COPS Security Checker Systems*, Purdue Technical Reports CSD-TR-999, January, 1994.

[Spafford and Kumar, 1995] Sandeep Kumar, Eugene H. Spafford. *A Software Architecture to Support Misuse Intrusion Detection*. COAST technical report 95/009, Purdue University, West Lafayette, IN, March 1995.

[Spafford and Crosbie, 1995] Eugene H. Spafford and Mark Crosbie. *Active Defense of a Computer System Using Autonomous Agents*. Technical report 95/008, Purdue University, West Lafayette, IN, 1995.

[Spafford, 1996] Garfinkel, S., Spafford, G., *Pratical UNIX and Internet Security*, Second Edition. Sebastopol, CA: O`Reilly & Associates, Inc., 1996.

[Spafford et al., 1996] Mark Crosbie, B Dole, T Ellis, Ivan Krsul and Eugene H. Spafford. *IDIOT – Users Guide*. Technical report 96/050, Purdue University, COAST Laboratory, West Lafayette, IN, 1996. http://citeseer.nj.nec.com/crosbie96idiot.html

[Spafford et al, 1998a] Jai Sundar Balasubramaniyan, Jose Omar Garcia-Fernandez, David Isacoff, Eugene Spafford and Diego Zamboni. *An Architecture for Intrusion Detection Using Autonomous Agents*. COAST technical report 98/05, Purdue University, W. Lafayette, IN, june 1998. [20] ftp://coast.cs.purdue.edu/pub/COAST/tools/AAFID/

[Spafford et al, 1998b] Ivan Krsul, Eugene Spafford and Mahesh Tripunitara. *Computer Vulnerability Analysis*. COAST Laboratory, Purdue University, West Lafayette, IN, May 1998.

[Spafford and Zamboni, 2000] Eugene H. Spafford, Diego Zamboni. *Intrusion Detection Using Autonomous Agents*. Computer Network 34, 547-570, 2000.

[Stallings, 1998] William Stallings. *Cryptografhy and Network Security: Principles and Practice*. Prentice Hall, Inc. New Jersey, EUA,1998.

[Stephenson, 1999] Peter Stephenson. *Investigating Computer-Related Crime: Handbook for Corporate Investigators*. CRC Press LLC, 1999.

[Valdes et al, 1995] D. Anderson, Frivold, Tamaru, Valdes. *Next Generation Intrusion Detection Expert System (NIDES). Software Design, Product Specification, and Version Description document*, Project 3131, July 1994. http://www.sdl.sri.com/projects/nides/index5.html.

[Zamboni, 2000] Diego Zamboni, *Doing Intrusion Detection Using Embedded Sensors* – Thesis proposal. CERIAS technical Report 2000-21, Purdue University, W. Lafayette, IN, october 18, 2000. http://www.cerias.purdue.edu/projects/esp/.

Apêndice A

AAFID2 - AUTONOMOUS AGENT FOR INTRUSION DETECTION SYSTEM, VERSÃO 2

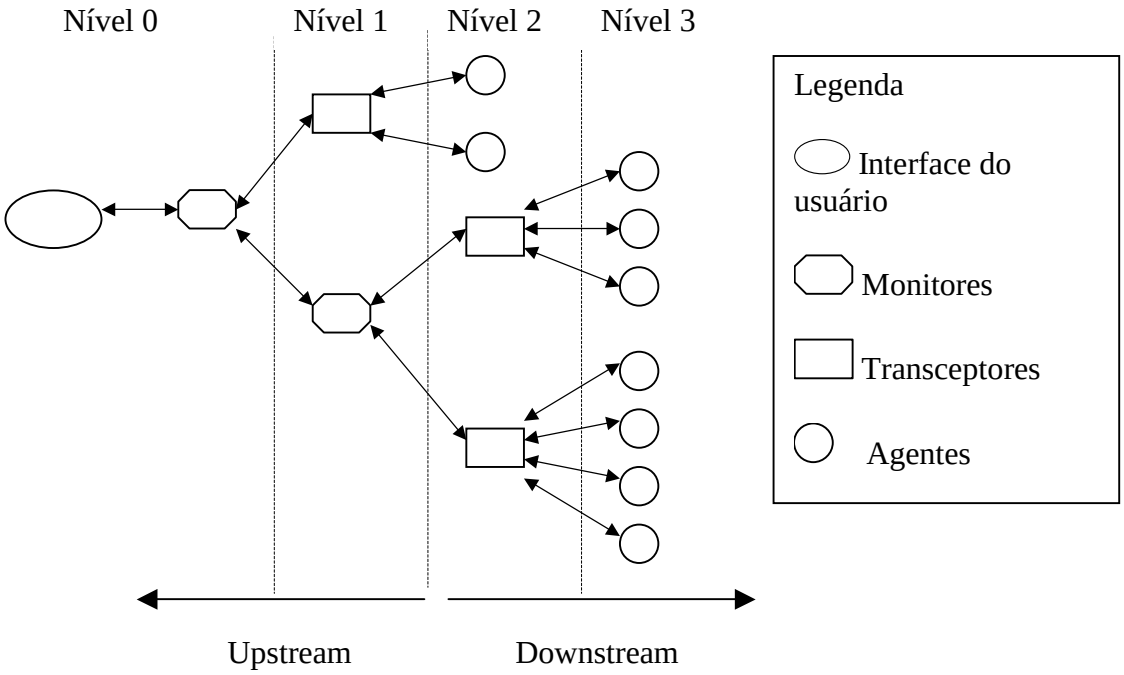

Figura: Arquitetura do AAFID2

É um sistema que explora questões de arquitetura associada à detecção de intrusão distribuída. Tenta resolver um dos principais problemas de detecção de intrusão, que é quanta informação conservar e analisar localmente e quanta analisar centralmente. A coleta e análise de dados distribuída, entre muitos *hosts*, concede escalabilidade não encontrada comumente em IDS. Apesar da arquitetura apresentar a desvantagem de se ter os monitores como único ponto de falha.

Mesmo sendo um sistema baseado em *host*, suporta algumas funções de rede. Foi implementado utilizando-se a linguagem de *script*s *Perl* e diversos recursos de administração de sistemas e segurança comuns em ambiente Unix. A arquitetura, construída de maneira hierárquica, é composta de entidades que se referem a qualquer dos componentes definidos na mesma. Inclui

agentes, transceptores, monitores e outros programas que interagem com a arquitetura, tais como interface com o usuário. Cada entidade é um objeto separado, que conserva seu próprio estado interno e comportamento de acordo com sua função no sistema. São programas que rodam de forma independente.

Agentes estáticos e autônomos

Agentes[20] de software que, com a ajuda dos filtros[21], coletam informações do sistema onde estão rodando. Monitoram aspectos específicos de atividades nos *hosts* e relatam, para os *hosts* transceptores, um comportamento anormal, que pode indicar problema de segurança.

As tarefas inicialização, checagem e *cleanup* são específicas para cada agente. Todas as outras funções são fornecidas pela infra-estrutura, para permitir a criação de novas entidades no menor tempo possível.

Tranceptores

Residem nos *hosts*. São responsáveis por receber os dados dos agentes, que rodam no mesmo *host* e controlar suas atividades, como iniciação ou terminação. Eles não precisam de capacidade de comunicação remota *downstream*, mas, de se comunicar com um grande número de entidades locais. Agentes podem fornecer dados, na forma de *status_updates* e comandos, por exemplo, para o pedido de um módulo adicional necessário para execução. Transceptores podem necessitar enviar um comando para um agente, por exemplo, configurar um valor de parâmetro. Também precisam ser capazes de responder a comandos vindos de monitores e fornecer, aos mesmos, dados como *status_updates*. Se um transceptor recebe uma mensagem de um agente, a qual não é capaz de interpretar, a mesma deve ser enviada para uma entidade pai do transceptor para processamento adicional. Portanto, consolidam relatos e os remetem para os sistemas monitores.

Cada transceptor controla todas as entidades em um *host*, mas não exercem influência sobre outros agentes em outros *hosts*. Existem técnicas para fazer isto, mas o sistema não as implementa.

[20] Tipos de agentes implementados no sistema e sua descrição no apêndice B.

[21] Tipos de filtros utilizados no sistema e sua descrição no apêndice B.

Monitores

São as entidades mais complexas com respeito à capacidade de comunicação. Um monitor deve ter todas as funcionalidades de um transceptor, pois também precisa ser capaz de controlar entidades locais e coordenar as atividades entre os transceptores. Deve dar início e controlar entidades remotas. Deve também ser capaz de dar início a novos transceptores, ou monitores em *hosts*, que não tenham um rodando, bem como enviar mensagens para eles. O monitor somente se comunica com monitores e transceptores, não o fazendo com agentes que possam estar rodando no *host.* O monitor deve ser capaz de escutar conexões de entidades remotas, porque pode ter iniciado um transceptor, ou outro monitor em um host e necessitar de ser controlado por um monitor já existente. A nova entidade deve ser capaz de entrar em contato com o monitor e registrar-se enviando uma mensagem CONNECT. Uma vez que a entidade remota é iniciada, ou um pedido de conexão foi processado, o monitor deve ser capaz de se comunicar com ela da mesma maneira que com entidades locais.

Finalmente, monitores têm a responsabilidade de atuar como repositório para todas as informações que suas sub-entidades possam precisar. Isto inclui, principalmente, código para os agentes e módulos de suporte. Por exemplo, um transceptor pode ter sido iniciado com poucos recursos locais e, quando ele precisar de um módulo cujo código não está presente no local, será solicitado ao seu monitor. Este deve ser capaz de localizar o código necessário e fornece-lo ao transceptor. Se o monitor não tem o código solicitado, deve enviar o pedido à sua própria entidade controladora. Se o código eventualmente chegar, o pedido original do transceptor deve ser imediatamente preenchido.

Representação da entidade

No AAFID2, cada entidade-objeto é representada por uma *hash* anônima, que contém os parâmetros para esta entidade específica. Os parâmetros são armazenados indexados por seus nomes. Uma *hash* simples em *Perl* pode armazenar diferentes tipos de elementos simultaneamente. Cada parâmetro pode ser de um tipo diferente sem causar qualquer problema.

Parâmetros são valores de nomes que são armazenados internamente em cada entidade e usados para determinar características de seu comportamento, por exemplo, *CheckPeriod*, é o período de

checagem, status corrente de 0 a 10. Também se faz necessário para que a entidade seja capaz de manter seu estado interno, por exemplo, quantas tentativas de *login* inválido foram observadas.

O uso de parâmetros nomeados, nos quais os parâmetros são armazenados indexados por seus nomes, é devido ao próprio mecanismo orientado a objetos da linguagem *Perl* conceder naturalmente este esquema. É uma maneira conveniente de comunicação da informação entre entidades e armazenamento de seu estado interno.

A detecção de um problema no sistema AAFID2 é indicada por um sistema indicador de *status*, computado a partir do *status* de todas as entidades no IDS. Cada entidade é capaz de guardar seu próprio status e relatar quando solicitada. Todas as entidades devem conservar seu status com um indicador numérico chamado *status* e uma descrição textual dele chamada *message.*

O valor do status deve ser um número entre 0 e 10 inclusive, onde 0 significa tudo normal e 10 significa extremo alarme. Valores intermediários representam diferentes graus de importância do problema detectado. Nenhuma definição formal é dada para os valores do indicador *status*, cada entidade tem o poder de designar.

Mensagens

As mensagens no modelo de comunicação fluem através das arestas da árvore. No nível de arquitetura, não há preocupação com o conteúdo semântico da mensagem. É definido um formato de mensagem genérico que pode ser usado para representar qualquer tipo de informação que uma entidade possa precisar para se comunicar.

Os campos definidos para as mensagens no AAFID2 são:

- Tipo de mensagem: Este campo, implicitamente, também especifica o formato do campo Dado, uma vez que a entidade destino deve saber como interpretar, de acordo, o campo citado.

- Subtipo de mensagem: Este campo pode ser usado para fornecer informação adicional sobre o conteúdo da mensagem, tão bem como indicar a interpretação semântica específica que deve ser dada ao campo Dado.

- Identificador de origem e Identificador de destino: Estes campos devem conter informações que, unicamente, identificam as entidades de origem e destino. O campo destino pode estar vazio se a mensagem é enviada num canal de comunicação *one-to-one*, onde não há incerteza sobre qual é o destino.

- *Time stamp*: Este campo é uma representação única do momento quando a mensagem foi gerada.

- Dado: A estrutura deste campo depende do campo "Tipo de mensagem" e pode conter sub-campos para a organização adicional da informação.

As mensagens são representadas por objetos da classe *Message.* O objetivo principal desta classe é armazenar os campos como definidos, mas, também, para permitir a conversão para ou, de um formato apropriado de string para enviar sobre a rede. Dentro de uma entidade, a mensagem será armazenada como um objeto, mas, antes de enviá-la para outra entidade, será convertida numa linha simples de texto com o seguinte formato:

TYPE SUBTYPE FROM TO TIME DATA

Modelo de comunicação

A arquitetura do sistema AAFID2 é, por natureza, hierárquica. Assim, concede, naturalmente, hierarquia a um modelo de comunicação. Como mostrado na figura, os níveis que estão mais próximos da raiz da hierarquia são identificados como *upstream* e, os que estão mais próximos das folhas da árvore, como *downstream*. Uma entidade está acima da outra se pode ser alcançada pelos caminhos de comunicação *upstream* e, abaixo, se pode ser alcançada pelos caminhos de comunicação *downstream.*

Uma entidade somente pode ter comunicação direta com entidades que estejam imediatamente acima e abaixo dela própria.

Agentes estão nas folhas da árvore, portanto, somente enviam mensagens para cima. A entidade localizada na raiz da árvore é sempre um monitor. Mas este pode aparecer também, em níveis intermediários. A única entidade que pode estar acima do monitor localizado na raiz é a interface do usuário, ou algum outro programa que controle todo o IDS.

Mecanismo de comunicação

Um dos principais requisitos das entidades na comunicação com entidade-Pai é a transparência. O usuário deve ser capaz de desempenhar o papel da entidade-Pai iniciando-a a partir de um comando *shell*, digitando mensagens no teclado e vendo a saída produzida pela entidade no monitor.

Para se conseguir isto, usou-se o mais comum mecanismo de comunicação disponível no UNIX, bem como em outros sistemas operacionais; os arquivos descritores *standart output* e *standart input*. Para cada entidade, as mensagens vindas de suas entidades controladoras partem da entrada padrão e, qualquer escrita, para a saída padrão, irá para a entidade controladora. No UNIX, os arquivos descritores podem ser redirecionados para, ou, de qualquer lugar diferente. Assim, a saída e a entrada padrão de um processo podem ser redirecionadas para acessar um arquivo, um *pipe*, uma conexão de rede ou um terminal.

Se um usuário executa uma entidade diretamente, suas saída e entrada padrão são redirecionadas para tela e teclado respectivamente, como com qualquer outro programa. Isto e o fato de que as mensagens são representadas por *strings* de leitura humana, permite ao usuário interagir diretamente com a entidade, digitando mensagens e, ser capaz de ler a resposta na tela.

A comunicação *intra-host* é feita usando *pipes* UNIX. Para o processo-filho, o *pipe* que vem do pai é conectado à entrada padrão e, a saída padrão, ao *pipe* que vai para o pai. A comunicação entre *hosts* é feita usando-se canais TCP.

Processamento de entrada

As entidades no AAFID2 podem receber informações de múltiplas origens, *pipes*, conexão de rede e teclado. Uma entidade tem que atender a todas as mensagens que chegam por qualquer destes canais. Por exemplo, um monitor tem que escutar seu canal de controle, aquele que leva mensagens para cima, na entrada padrão, pode ter que atender a um número de *pipes* conectando-os a entidade local e, também, a um número de conexões TCP que conectem a um monitor ou transceptores remotos.

No UNIX, todos estes canais de comunicação podem ser vistos como arquivos manuseáveis. Uma vez que o canal esteja aberto, pode ser lido e escrito como se fosse um arquivo normal no disco local. Assim, é fácil monitorar, automaticamente, todos os canais de entrada para mensagem.

Implementação de mensagem e comando

O comportamento de diferentes tipos de mensagens e comandos é implementado através de subrotinas normais *Perl.* Seguindo certas convenções é, portanto, factível implementar uma nova entidade para acrescentar funcionalidade.

Carregamento e execução da entidade

A habilidade de invocar e controlar uma nova entidade local é herdada por monitores e transceptores. A classe **Monitor** estende esta capacidade, permitindo a invocação de entidades remotas. Cada entidade deve ser capaz de rodar como um programa *stand-alone*.

Mecanismo usado por cada modo de execução da entidade:

As entidades devem ser capazes de trocar mensagens com suas entidades controladoras e, todas as mensagens e comandos recebidos devem ser processados logo que possível. A entidade controladora pode estar no mesmo *host*, por exemplo, o transceptor é a entidade controladora dos agentes, ou pode estar em hosts diferentes, por exemplo, o monitor é a entidade controladora do transceptor.

É possível para o usuário iniciar uma entidade a partir da linha de comando e interagir com ela dando comandos e mensagens do teclado. A entidade também pode se inicializada pela sua entidade controladora.Todas devem reagir adequadamente aos seguintes sinais UNIX:

- TERM - Para uma terminação sem problemas. Mesmo comportamento de uma mensagem ou comando STOP.
- HUP - Para inicializar.
- USR1 - Para, temporariamente, pausar suspendendo todas as atividades. Se a entidade já estiver pausada, não faz nada.

- USR2 - Para reinicializar. Se não estiver pausada, não faz nada.

Execução de entidade

Toda entidade deve ter um método chamado **run,** que será chamado quando do ponto de entrada da execução da entidade. Quando o método **run** retorna, a execução da entidade é considerada finalizada e o solicitante estará livre para proceder com outras atividades.

Para inicialização, toda entidade tem um método chamado **init,** que será chamado quando o objeto é criado. Inicialização adicional pode ser feita no método **run**. Para **run** e **init,** tão bem como o código para fazer automaticamente atividades de configuração essenciais, tais como gerar um identificador de identidade, enviando uma mensagem CONNECT para começar e uma mensagem DISCONNECT para terminar. Estas atividades são feitas no método **new** (construtor), definido pela classe **Entity,** devendo ser herdado por todas as outras entidades.

As classes **ControllerEntity** e **Monitor** reconhecem um novo comando chamado START, que recebe um parâmetro chamado *Class*, o qual contém a especificação da entidade que precisa ser iniciada, em uma das seguintes formas:

- "**Classname**" pedido de execução da classe dada de uma entidade local. Reconhecido por ambos **ControllerEntity** e **Monitor**.
- "**Host:Classname**" pedido de execução da classe dada de uma entidade no *host* especificado. Reconhecido somente pelo **Monitor**.

Execução da entidade stand-alone

Para permitir que uma entidade seja carregada como um programa *stand-alone*, existe um mecanismo que, automaticamente, cria uma instância da classe e invoca seu método **run** quando a entidade é executada. Contudo, este código não deve ser executado quando a entidade é carregada a partir de outra.

No AAFID2, o mecanismo usado para resolver este problema é a subrotina chamada **EndOfEntity**, implementada pela classe **Entity**, que deve ser invocada na última linha no arquivo de origem de uma entidade. Esta subrotina checa se o arquivo está sendo carregado por si próprio ou como um módulo a partir de outro programa. No primeiro caso, cria-se uma instância

da entidade, chama-se o método **run** e se sai quando a execução termina. No segundo caso, nada é feito e o valor de 1 é retornado, que é a maneira normal em *Perl* de se sinalizar que um arquivo módulo foi carregado corretamente.

Quando uma entidade é carregada *stand-alone*, suas entrada e saída-padrão não são redirecionadas e o usuário é capaz de digitar mensagens e ver o resultado no terminal.

Execução local de uma entidade a partir de outro programa

Quando uma entidade precisa ser executada a partir de outro programa, por exemplo, um transceptor carregando um agente, os seguintes passos são desempenhados:

- Carregamento da classe **Entity** usando o procedimento **use**.
- Criação de uma nova instância da entidade usando seu método **new**.
- Criação de um novo processo, onde a nova entidade será executada.
- Criação de dois *pipes* entre os processos pai e filho, um para envio de mensagem do pai para filho e o outro, vice versa. No processo filho, a leitura final de um *pipe* é associada com a entrada-padrão e a escrita final do outro é associada com a saída padrão, portanto, estabelecendo o canal *"up"* da entidade. No processo pai, os finais correspondentes dos *pipes* são armazenados em um parâmetro interno para uso futuro e ser capaz de receber mensagens da nova entidade.
- O processo filho executa a nova entidade invocando o método **run.** Quando retorna, o processo termina.

Execução remota de uma entidade

A classe **Monitor** fornece facilidades para ser capaz de pedir a ativação de uma entidade em um *host* remoto. O mecanismo usado é o seguinte:

- Checa se o canal de comunicação para o transceptor ou monitor no *host* requisitado já existe. Em caso positivo, envia uma mensagem para carregar a entidade solicitada. Caso contrário, continua com os passos seguintes.
- Executa o programa Starter, no *host* remoto, com os parâmetros apropriados e informa a partir de qual *host* foi executado. AAFID2 usa *ssh (Secure Shell)* para executar *Starter* no *host* remoto.

- O monitor configura os *flags* apropriados para indicar que está esperando por uma conexão a partir do *host* onde o *Starter* foi executado e continua sua execução normal.
- No *host* remoto, o *Starter* instancia um transceptor (classe **PlainTransceptor**), contata a porta do servidor no monitor, estabelece uma conexão TCP e redireciona sua saída e entrada-padrão para ele.
- O *Starter* roda o transceptor, que se comunicará com o monitor, através de seus padrões de entrada e saída, como redirecionado no passo anterior, para registrar com ele.
- Quando o monitor recebe a mensagem CONNECT a partir do transceptor criado recentemente, identifica-o como uma entidade que tem que ser iniciada e envia o comando apropriado para ele.

As seguintes características são observadas neste processo.

- Todas as entidades são iniciadas no local, por exemplo, o transceptor é iniciado no local pelo *Starter* e a nova entidade, pelo transceptor. É um programa separado chamado *Starter* que estabelece a conexão de rede e faz o redirecionamento apropriado. Por esta razão, a conexão de rede é transparente para ambos, o novo transceptor e a nova entidade. O monitor sabe sobre a conexão de rede, porque o mesmo recebe o pedido de conexão do *Starter* na porta do seu servidor.
- Uma vez que a conexão de rede é estabelecida entre o monitor e o transceptor, é mantida aberta.

Pedido de código que falta

Quando uma entidade controladora recebe um pedido para iniciar uma nova entidade, pode ser que parte de todo o código necessário para carregar esta entidade, por exemplo, sua classe arquivo-origem ou um módulo necessário, não esteja presente no *host* local. Neste caso, a entidade controladora envia uma mensagem para cima, NEEDMODULE, especificando o nome da classe que falta. A entidade que está acima deve achar o código apropriado, provavelmente enviando uma mensagem NEEDMODULE. Então, envia a mensagem para cima numa mensagem NEWMODULE.

Uma vez que a entidade controladora recebe a mensagem NEWMODULE contendo o código de que necessita, o mesmo é salvo num arquivo local e tenta carregá-lo. Se continuar faltando

código, por exemplo, se o código recebido precisa de outro módulo que não esteja presente, então o processo é repetido até que a entidade da classe apropriada possa ser instanciada.

Este mecanismo permite a instalação inicial do AAFID2 no sistema monitorado, ser limitado a classes essenciais e, então, ter todas as outras, tais como agentes, desdobradas quando necessárias.

Mecanismo de segurança para comunicação

O único mecanismo de segurança de comunicação corrente usado no sistema AAFID2 é o uso de *ssh* para rodar o programa *Starter* no *host* remoto. Contudo, nenhuma autenticação ou criptografia é feita depois deste ponto. Isto é importante para um sistema de produção, particularmente em comunicação entre *hosts*. Pode ser viável o uso de mecanismos de comunicação existentes.

Alguns agentes deste sistema foram testados, eles basicamente monitoram:

- comando su;
- diretório /tmp;
- processo /etc/Shell;
- espaço em disco;
- seqüência de processos;
- arquivo rhosts, só aceita 400,200 e 100;
- logins falhos;
- conexões para múltiplas portas (portscan);
- conexões para mesmo host de um mesmo serviço;
- carregamento CPU;
- configuração FTP;
- conexões TCP.

ACME – ADVANCED COUNTER MEASURES ENVIRONMENT

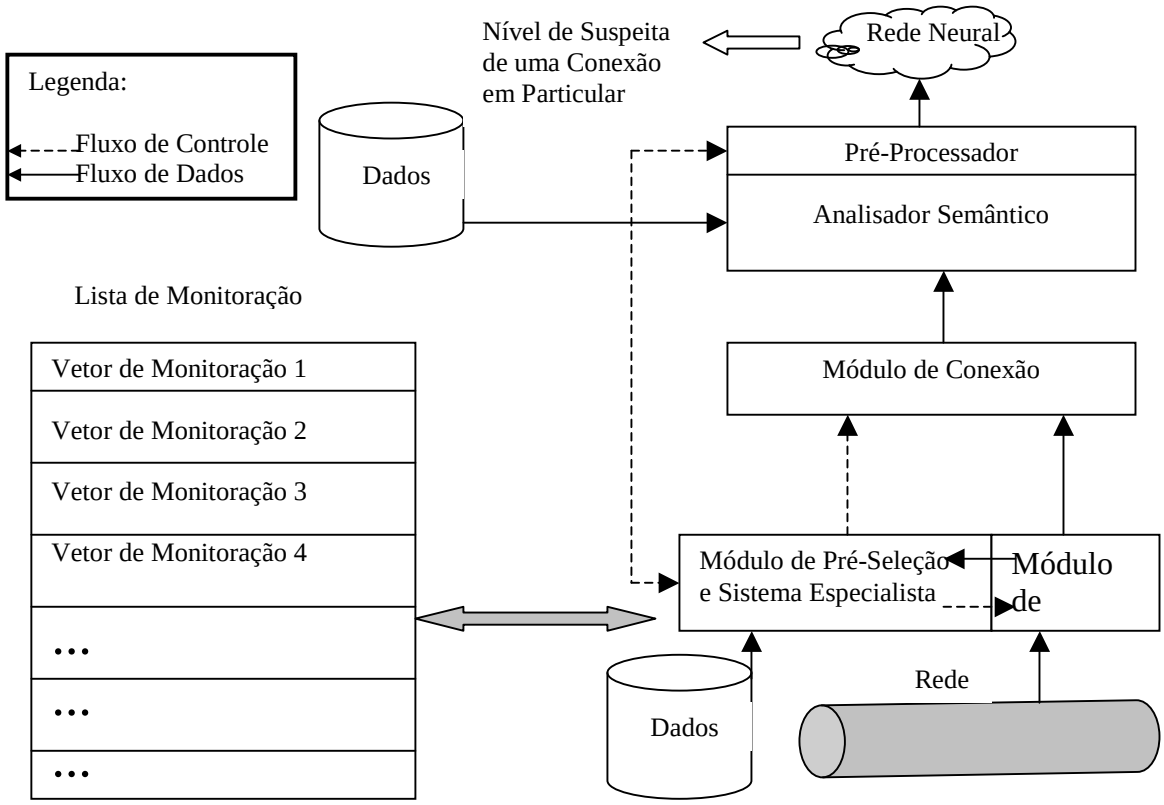

Figura: Arquitetura do ACME

O ACME foi desenvolvido como um IDS baseado em conhecimento. Sua característica primordial é a operação de um sistema de captura de pacotes em conjunto com uma rede neural. A rede neural verifica os padrões de comportamento, previamente codificados em assinaturas e desenvolve uma porcentagem mostrando o quão similar este padrão é de algum ataque que ela conhece. A partir desse ponto, o administrador analisa se é ou não, um ataque. A vantagem deste sistema é a adaptabilidade proporcionada pela rede neural. Ela pode ser retreinada à medida que novos padrões de ataques são descobertos, para que possa identificá-los. Outro fator determinante é sua capacidade de generalização, onde padrões semelhantes podem ser identificados com uma possibilidade intermediária. Assim, o administrador pode determinar uma porcentagem de risco limite para efetuar alguma medida de segurança.

Graças a essa capacidade de generalização, existe a possibilidade de identificação de ataques dissimulados. Ataque desse tipo é aquele onde o intruso procura ocultar suas reais intenções, seguindo um conjunto de ações não correlatas ao ataque. Para que a rede neural possa manter seu critério o mais próximo da situação real, uma base de dados de ataques é freqüentemente atualizada e os ataques são simulados para obtenção de novas assinaturas de ataque.

O modelo de detecção de intrusão é formado por um sistema de captura e tratamento de pacotes, um sistema de rede neural e um gerenciador de comunicações e interface com o usuário. O sistema de captura e o sistema de rede neural podem ou não, residir na mesma máquina. Se o sistema de rede neural for implantado na máquina central, gerenciadora de segurança, devem haver agentes e gerentes de comunicação utilizando protocolos que coordenem a troca de informações entre as máquinas. O sistema de captura e tratamento de pacotes é organizado em módulos, que tratam o fluxo de pacotes e que terminam fornecendo o vetor de estímulo para a rede neural. O nível mais baixo apenas captura um fluxo de dados na rede e passa os pacotes ordenados para o módulo de conexão sob controle do módulo de pré-seleção.

O ACME utiliza algo semelhante a uma ferramenta para captura de pacotes e usa a biblioteca denominada *libpcap*, que implementa o *BSD Packet Filter* ou, simplesmente, *BPF*. Este filtro seleciona pacotes do barramento segundo o protocolo[22], além de oferecer uma gama de opções quanto à filtragem de *hosts*, redes, etc. Para isso, basta passar, para o filtro, instruções sobre o tipo de pacote com sintaxe similar ao *Tcpdump*[23].

O módulo de captura de pacotes (CAP) tem toda a informação na estrutura dos pacotes capturados, necessários à auditoria. Os cabeçalhos e os campos de dados são enviados integralmente para o módulo conexão (CON) e apenas os cabeçalhos para o módulo de pré-seleção e sistema especialista (PSSE).

O módulo PSSE decide a partir de que momento uma conexão é considerada suspeita. É responsável por mais um nível de filtragem, pois o módulo de conexão só registra as atividades de máquinas ou domínios considerados suspeitos que sejam indicados pelo PSSE. Isto é feito por meio do seguinte mecanismo:

- Uma Tabela de Níveis de Segurança (TNS), contendo o nível de segurança dos serviços (por exemplo, *telnet, ftp, finger*) que é consultada de acordo com o serviço usado pela conexão em questão.

[22] IP, TCP, UDP, ICMP entre outros.

[23] TCPDUMP Network Research Group of the Lawrence Berkeley National Laboratory ftp://ftp.ee.lbl.gov/tcpdump.tar.Z

- Uma Tabela de Controle de Conexão (TCC), onde cada entrada contém dados sobre *hosts* origem/destino, um marcador horário (*time stamp)* de conexão, nível de segurança de serviço, porta origem/destino e tempo inicial /final.
- Uma Tabela de Domínios Suspeitos (TDS) mantida quando um *host* atinge um determinado nível de segurança e o módulo de conexão é disparado. É freqüentemente atualizada.

O limiar para o disparo dos procedimentos de monitoração é um valor definido pelo administrador do sistema, o qual vai definir o comportamento do sistema como um todo. Este módulo faz uma seleção dos dados enviados pelo módulo CAP, amenizando o fluxo de dados, o que não acontece normalmente com ferramentas tradicionais.

O módulo CON é responsável pela criação e manutenção de vetores de conexão. Este módulo ignora todos os pacotes até que o módulo PSSE avise qual conexão é considerada suspeita, para que se inicie então o monitoramento. O vetor de conexão consiste na junção de dois vetores de fluxo, que representam a conexão de forma unidirecional. Com a utilização de um par de vetores de fluxo podemos representar uma conexão em particular e os dados que nela transitam. Na prática, o vetor de conexão nada mais é que um arquivo com um nome padronizado representando a conexão. Dentro deste arquivo estão todos os dados que trafegam naquela conexão a partir do momento em que ela foi considerada suspeita.

O Analisador Semântico e Pré-Processador (ASPP), de posse dos dados condensados pelo módulo CON, interpreta e seleciona aquilo que for relevante, como trechos suspeitos que, combinados, caracterizam um ataque (as assinaturas de ataques). Prepara-se, então, para a fase seguinte, onde realiza uma tradução, ou ainda, uma codificação de linguagens, possibilitando a conexão ao módulo de rede neural. O ASPP analisa, em tempo real, os dois sentidos do fluxo (vetor de conexão) com o auxílio de uma base de dados, constituída, principalmente, por duas tabelas de conversão. A primeira associa cada seqüência de caracteres, que seja relevante, a um número inteiro. A segunda, cada número inteiro válido a um código binário reconhecível pela rede neural.

O ponto principal do Módulo de Rede Neural (MRN) é a interface com a rede neural, que é obtida a partir do arquivo de definições da rede, treinada com um aplicativo[24] simulador de redes

[24] Software aplicativo simulador de redes neurais – Stuttgart Neural Network Simulator.

neurais. Para o treinamento, a rede é exposta a um conjunto grande e variado de vetores de estímulo, os quais representam sessões suspeitas, intrusivas e legítimas. É feita uma rígida monitoração durante tal processo, verificando se o aprendizado satisfaz os requisitos impostos pelo sistema e, principalmente, se existe uma melhora na capacidade de generalização de rede frente a novos tipos de conjunto de dados. O MRN pode ser retreinado com o mesmo aplicativo e exposto a novos padrões, escolhidos juntamente com padrões já conhecidos. Amplia-se assim, o nível de conhecimento do MRN e, também, a base de dados utilizada pelo ASPP, o que acarretará uma melhoria significativa no resultado geral do sistema.

A interface recebe todo o conjunto de bits que compõe o vetor de estímulo e, de acordo com o tipo de treinamento ao qual a rede foi submetida, retorna uma porcentagem, a qual indica o grau de suspeita da sessão. A facilidade proporcionada pela utilização desta interface torna altamente factível a criação de sub-módulos de prevenção e contra-medidas, que podem ser desenvolvidos de tal modo a executar tanto ações automáticas quanto interativas em um ambiente *standalone* ou distribuído. Como por exemplo, emissão de vários níveis de alerta ao administrador, disparo de processos de auditoria, ativação de contramedidas para isolar o *host* ou o domínio originador do ataque através do ajuste de, por exemplo, um *firewall.*

AID - Adaptive Intrusion Detection System

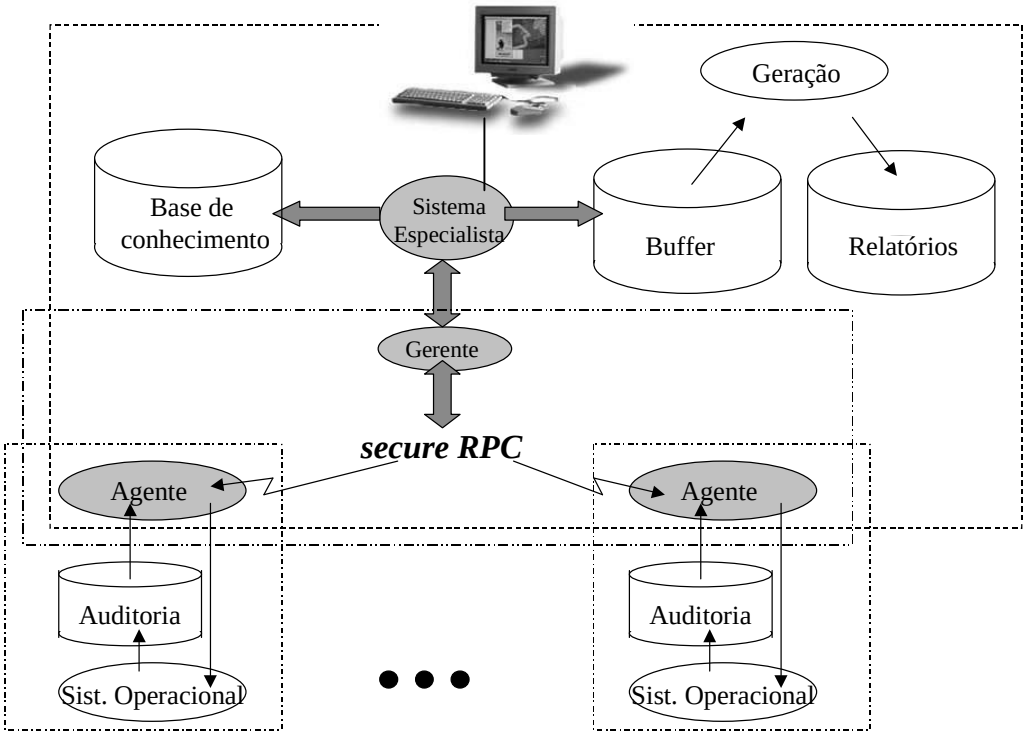

Figura: Arquitetura Cliente – Servidor AID

A arquitetura cliente servidor consiste de agentes residindo nos *hosts* e de uma estação de monitoramento central. As informações são coletadas por agentes nos *hosts* monitorados e convertidos para um formato de dados independentes do sistema operacional. Desta forma, o monitoramento de um ambiente heterogêneo UNIX é suportado. Estes dados auditados são transferidos para uma estação de monitoramento central para análise e processamento. São colocados em *cache* e analisados, em tempo real, por um sistema especialista baseado num *RTWorks*[25].

Na estação central, o gerente fornece funções para administração segura dos *hosts* monitorados, controla as funções auditadas, os pedidos de novos dados auditados e controla o recebimento e encaminha as decisões vindas dos agentes para o sistema especialista. O sistema especialista usa uma base de conhecimento de assinaturas de ataque, que são modeladas por máquinas de estado finito determinísticas e implementadas como seqüência de regras. Tem implementado 100 (cem)

[25] http://www.talarian.com/rtworks.html

regras na sua base de conhecimento e é capaz de detectar 10 (dez) cenários de ataque. O protótipo[26] pode manusear com oito agentes.

O administrador de segurança pode acessar capacidades de monitoramento relevantes via uma GUI. Em adição, o sistema especialista cria relatórios de segurança.

O *Secure RPC* é usado para comunicação entre gerente e agentes.

[26] AID foi testado em dezembro de 1995, num ambiente de rede local, consistindo de estações SPARC rodando Solaris 2.x e TCP/IP.

Ripper - Sistema Baseado em Mineração de Dados

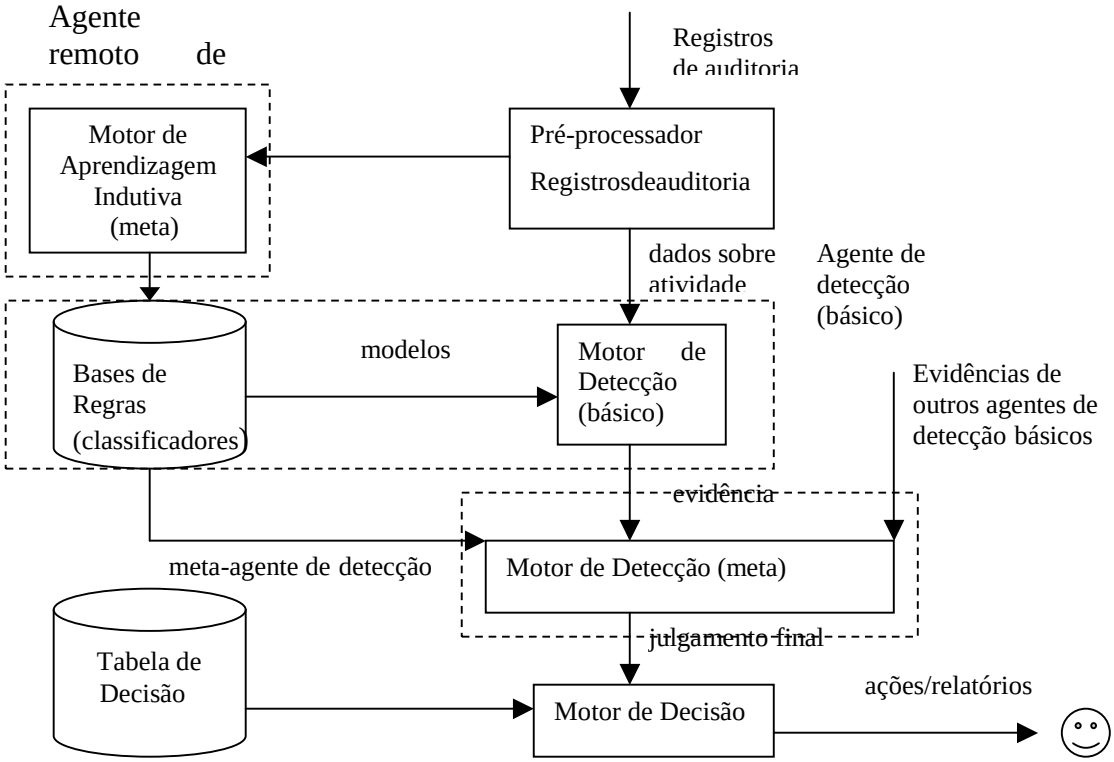

Figura: Arquitetura de um Sistema de Detecção de Intrusão Baseado em Data Mining

Lee e Stolfo apresentam uma abordagem baseada em técnicas de *data mining* para descobrir padrões úteis e consistentes das características do sistema, que descrevem os comportamento de usuários e programas. A partir daí, o conjunto de características relevantes do sistema é usado para computar classificadores (aprendidos por indução) que podem reconhecer anomalias e intrusões.

A arquitetura proposta é hierárquica e inclui dois tipos de agentes inteligentes.

O agente de aprendizado pode residir em uma máquina servidora, pelo seu poder de computação. É responsável pela computação e manutenção do conjunto de regras de programas e usuários. Ele produz os modelos de detecção básicos e os modelos de detecção meta.
O agente de detecção é genérico e extensível. É equipado com um conjunto de regras, oriundo do agente de aprendizado remoto. As regras são aprendidas e periodicamente atualizadas. O motor de detecção executa a classificação nos dados de auditoria e emite evidências de intrusões. A

principal diferença entre agente de detecção básico e agente de detecção meta é que este usa dados auditados pré-processados como entrada, enquanto aquele usa evidências de todos os agentes básicos de detecção. Estes agentes não precisam rodar no mesmo *host*.

Os agentes de detecção são leves e podem funcionar independentemente, de tempo e local, dos agentes de aprendizado e são equipados com conjunto de regras. Podem relatar novas instâncias de intrusões pela transmissão de gravações de auditoria para o agente de aprendizado, que podem computar e atualizar classificadores para detectar tais intrusões e despachar para todos os agentes de detecção.

O sistema constrói modelos de classificação para detecção anômala. O primeiro conjunto de experimentos foi em dados de *sendmail system call*. Utiliza RIPPER, um programa de aprendizagem de regras para treinamento dos dados e classificar os modelos em normal ou anormal. O segundo experimento foi monitorando o tráfego de rede diretamente, utilizando um programa de captura de pacotes, como o *tcpdump*.

Para construir um classificador é necessário colher uma quantidade suficiente de dados para treinamento e identificar um conjunto de características significantes. Dois algoritmos foram usados para guiar o processo de auditoria: *association rules* e *frequence episodes.*

DIDS - DISTRIBUTED INTRUSION DETECTION SYSTEM

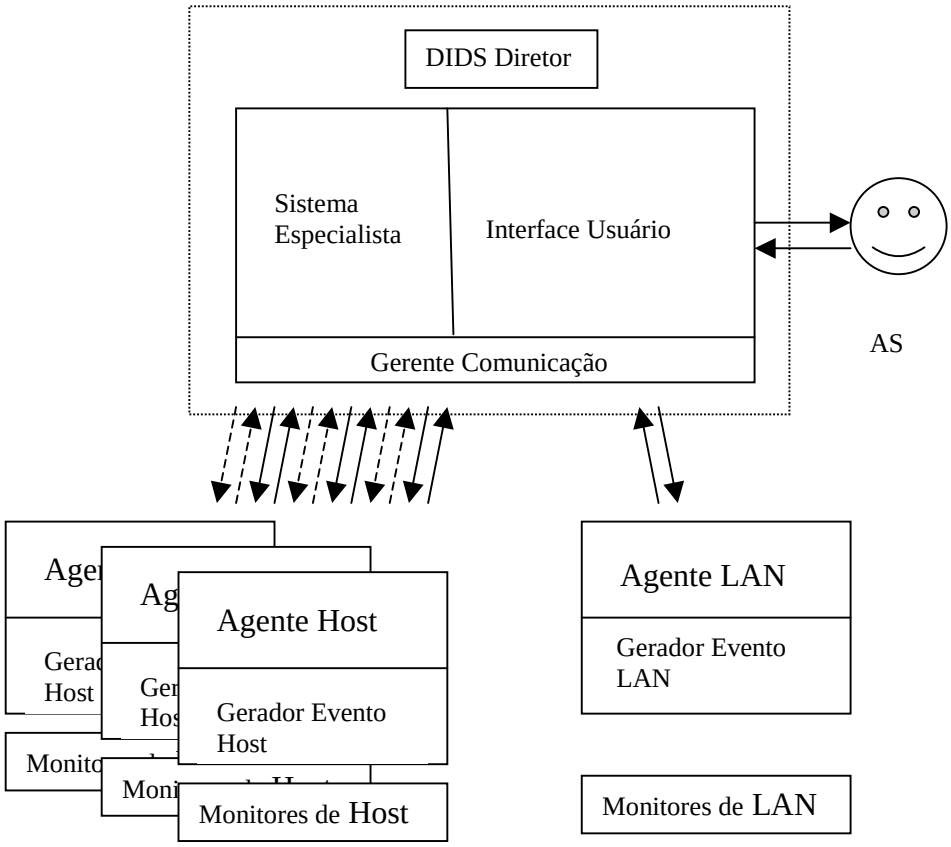

Figura: Arquitetura DIDS

DIDS ilustra a abordagem centralizada para detecção de intrusão em rede. É basicamente uma coleção de múltiplos sistemas de detecção de intrusão rodando em sistemas individuais, que cooperam para detectar intrusões em toda a rede. Os componentes de detecção de intrusão em sistemas individuais são responsáveis pela coleta da informação do sistema e conversão para uma forma homogênea para ser passada para o analisador central.

Sua arquitetura combina coleta distribuída e redução de dados com análise de dados centralizada.

DIDS tenta correlacionar informações sobre monitoramento individual de usuários, com o uso de um *User Network Identifier* (NID), onde é seguida a pista dos movimentos de cada usuário através da rede, assumindo diferentes identidades. Também segue a pista das ações desempenhadas pelos *hosts* via o *LAN monitor*.

Seu projeto é modular e seus componentes são 3 (três).

Monitor de *host* por *host*

Responsável pela coleta de evidência de atividade não autorizada ou suspeita. O *host* monitor usa a auditoria de dados nativa, fornecida pelo sistema operacional, para coleta de dados, Depois é pré processado para uma sintaxe comum chamada *Host Audit Record* (HAR). Esta conversão da informação para uma forma homogênea, para ser passada para o analisador central, torna o DIDS capaz de manusear sistemas individuais heterogêneos com um sistema de detecção de intrusão centralizado. Este primeiro nível de abstração melhora a portabilidade e heterogeneidade das porções restantes do *host* monitor. Gravações redundantes são eliminadas neste ponto, que fornece uma significante redução no numero de gravações que o alto nível do *host* monitor precisa para processar (mais do que uma redução de 4:1)

Certas gravações críticas são enviadas diretamente para o sistema especialista. Outras processadas no local pelo *host* monitor, como os *profiles* do usuário e do comportamento de sistema e as assinaturas de ataques. O objetivo do projeto é empurrar o processamento das operações para os monitores de nível o mais baixo possível. Para fazer isto, o HEG cria um objeto mais abstrato chamado evento, que inclui qualquer dado significante fornecido por uma gravação de auditoria original. Um evento relatado por um *host* monitor é chamado um HAR. De todos os possíveis eventos, somente um subconjunto é encaminhado para o sistema especialista para criação e aplicação do NID.

A comunicação do *host* monitor com o DIDS *director* é feita pelo *host agent.*

Monitor de LAN por segmento de rede.

Responsável pela coleta de evidência de atividade não autorizada ou suspeita. Também consiste de um *LAN Event Generator* (LEG) e de um *LAN Agent.* Usa muitos níveis de análise para capturar a maioria dos eventos significantes. Também retém os dados auditados para análise pelo diretor. Usa e mantém *profiles* do comportamento da rede, que são atualizados periodicamente. *LAN monitor* utiliza heurísticas simples para tentar averiguar se uma conexão particular representa um comportamento intrusivo ou não.

DIDS pode manusear os *hosts* sem os monitores de *host,* desde que o *LAN monitor* possa relatar as atividades de rede de tais *hosts.*

DIDS diretor

É o cérebro do DIDS e responsável pela avaliação. Consiste de três partes principais:

- Gerente de comunicações: Coleta os dados enviados pelo *host* monitor e *pelo LAN monitor* respectivamente e comunica ao sistema especialista para o processamento.
- Sistema Especialista: Faz inferência sobre o estado de segurança do sistema e de cada *host* individual. Agrega a informação para apresentação para o Administrador de Sistemas - AS. É um sistema baseado em regras simples, escrito em CLIPS[27]. As regras são derivadas de um Modelo de Detecção de Intrusão hierárquico (IDM), que descreve a transformação de dados auditados "puros" para hipóteses de alto nível sobre intrusões e sobre segurança geral do ambiente monitorado. Esta visão unificada de sistemas distribuídos simplifica o reconhecimento de comportamento intrusivo, que atravessa *hosts* individuais. O IDM consiste de seis camadas, cada uma representando o resultado de uma transformação desempenhada nos dados (*audit records, events, subjects, events* no contexto, *threats,* valor numérico entre 0 (zero) e 100 que representa o estado de segurança geral da rede). O modelo é o princípio fundamental da base de regra.
- Interface do usuário: no qual o AS pode configurar o sistema e obter conhecimento sobre intrusões suspeitas.

[27] Sistema CLIPS, de domínio público, desenvolvido pela National Aeronautics and Space Administration – NASA.

EMERALD – EVENT MONITORING ENABLING RESPONSES TO ANOMALOUS LIVE DISTURBANCES

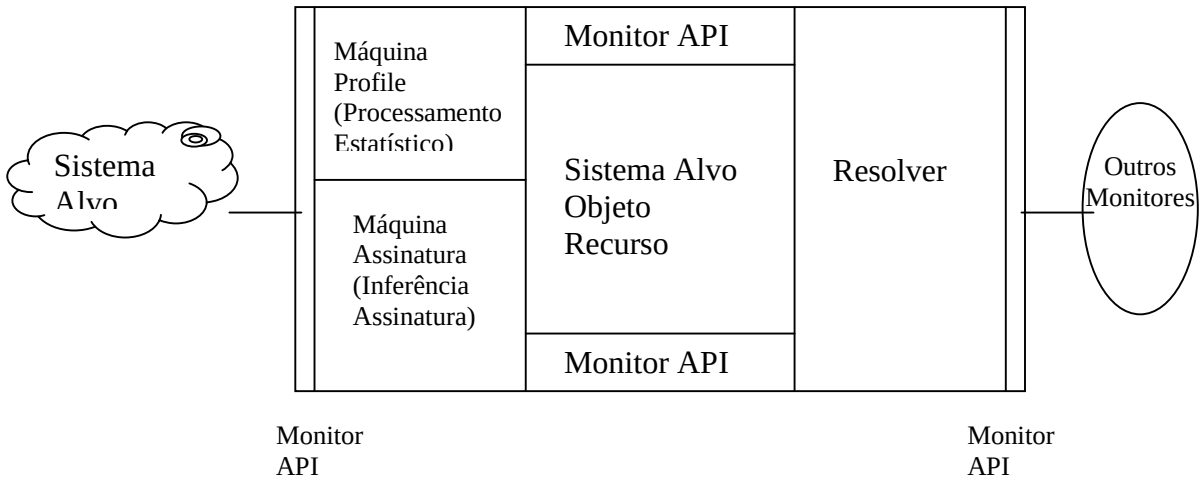

Figura: Arquitetura Monitor Genérico EMERALD

É uma arquitetura que utiliza a abordagem de agente distribuído. Fornecendo a característica de escalabilidade, para monitorar grandes redes distribuídas. E a de flexibilidade, abstraindo-se da funcionalidade para possibilitar adição de ferramentas externas. Emerald tem a habilidade de coletar, assimilar, correlacionar e analisar a informação de diferentes origens em tempo real, sendo este trabalho essencial para detecção de ataques distribuídos/coordenados.

Emerald inclui numerosos monitores locais em um *framework* que suporta distribuição local. Resultando em um *array* global de detectores que, consolidam alarmes e alertas, sem um controlador central.

O monitoramento do sistema é desempenhado em três camadas separadamente, onde cada camada pode ter seu próprio método baseado em comportamento e conhecimento:

- Monitores de serviço. Uma coleção de monitores de serviço que desempenham qualquer função de monitoramento nos *hosts*. São altamente distribuídos, independentes e dinâmicos. É a camada mais baixa na hierarquia.

- Monitores de domínio. Desempenham o monitoramento dentro de um domínio de serviço. São responsáveis em correlacionar informações disponíveis do monitor de serviço, várias configurações e ações relatadas. É a camada intermediária na hierarquia.
- Monitores de rede. Coleção de monitores que correlacionam atividades relatadas através de domínios monitorados. Permite a detecção de ameaças de rede, tais como *worms* e ataques coordenados entre múltiplos domínios. É a camada mais alta na hierarquia.

Para que as três camadas de monitoramento funcionem é empregada uma arquitetura para um componente genérico de monitoramento mostrado na figura acima.

A alimentação da informação segue via *audit trail,* pacotes de rede, tráfego *SNMP* e outros meios. A arquitetura tenta se abstrair do meio no qual a informação alimentada é obtida para produzir um esquema mais geral.

No centro da arquitetura se encontra o objeto de recurso específico do sistema alvo, que fornece um ambiente de processamento independente. É feito através de uma biblioteca de configuração de dados específicos e de métodos. Objetos de recurso são importantes, porque representam os únicos componentes não genéricos da arquitetura.
Objeto de recurso inclui os seguintes componentes:

- Estrutura de eventos configuráveis. Especifica os fluxos de eventos para o sistema alvo. Isto torna o Emerald portável para muitos fluxos diferentes, via esta especificação de entrada.
- Evento coleção de métodos. Processa a filtragem, por meio de rotinas de baixo nível, que manuseiam os fluxos obtidos do sistema alvo.
- Configurações de máquinas. Muitas máquinas de análise podem ser incluídas no objeto recurso. As operações de configuração da estrutura de dados e variáveis nestas máquinas são definidas, quando necessárias, para os fluxos-alvo.
- Configurações de unidade de análises. Os tipos de métodos de processamento de intrusão são especificados nas máquinas de objeto recurso.
- Listas de subscrição. Suporta a infra-estrutura de comunicação do Emerald. Informações sobre subscrições incluindo endereços e chaves públicas são encontradas nesta lista.
- Métodos de resposta. Respostas programadas para o objeto-recurso são incluídas para eventos identificados.

A informação alimentada no sistema alvo é enviada para as máquinas de análise para processamento. A máquina de *profile* desempenha detecção anômala baseada em *profile* estatístico e a máquina de assinatura, que desempenha o casamento de seqüências de assinaturas específicas as quais corresponde a ataques com atividades de sistema alvo.

O componente resolver é utilizado para coordenar os relatórios de análises das máquinas de *profiler* e assinatura e para iniciar atividades de respostas específicas. Também trata com informações fornecidas por outras máquinas de outros monitores de serviço, domínio ou rede, no qual a subscrição está disponível para informação.

O monitor API tem a habilidade de interoperar com a análise do alvo e com outras ferramentas de detecção de intrusão. Fornece uma interface para o monitor administrador, que inicia contramedidas de ataques, tais como terminando processos.

ESP – Embedded Sensors Project

A idéia em torno deste IDS é desempenhar detecção de intrusão usando pequenos sensores embutidos em um sistema computacional, que procuram sinais de intrusões específicas. Estes sensores desempenham monitoramento do alvo por observação do comportamento do sistema diretamente e não através do audit trail ou pacotes da rede. Sendo construídos dentro do código do sistema operacional e de seus programas e somente necessitando usar recursos adicionais do sistema quando fossem executados ou disparados. Sendo assim não impondo um carregamento extra considerável no host monitorado.

Um sensor é definido como uma parte de software que monitora uma variável específica, atividade ou condição de um host e suas observações são relatadas como valores numéricos e podem ser discretas ou contínuas.

O problema de análise de dados que vem de muitas origens diferentes não é novo. É assunto de estudo no campo chamado Fusão de Dados Multisensor. A aplicação de FDM na detecção de intrusão promete criar IDS que não somente seguem certas técnicas de análise nos dados coletados, mas que podem inferir conhecimento sobre seu ambiente e as ameaças que encontram.

A técnica utilizada para análise é *clustering*, o pacote de software Cluster foi utilizado, ele estima parâmetros de um modelo Gaussiano, usando algoritmo EM. Ele também estima o número de clusters, usando a estratégia baseada em Princípio de Comprimento de Descrição Mínima Rissanen

A proposta do trabalho é em direção a validar duas hipóteses:

- É possível construir sensores em um sistema computacional de modo que permita a detecção de intrusões conhecidas. Os sensores podem ter limitada capacidade de processamento e *logging* para evitar distúrbio na operação normal do sistema. Isto significa que cada sensor pode fazer algum processamento de valores que ele observa. Pode conservar também uma limitada quantidade de informação de estado. Suas capacidades de *logging* são limitadas a relatar valores de suas observações, possivelmente depois de uma transformação. O sensor pode também decidir não relatar uma observação específica.

- Um grupo de sensores projetado para detectar intrusões conhecidas pode também ser usado para detectar novas intrusões. Supõe-se que um número suficientemente grande de sensores exista e que tenham sido propriamente projetados.

Se estas hipóteses são válidas suas verificações devem contribuir para o projeto de um mais leve, mais distribuído e mais resistente IDS e para determinar os tipos de dados que necessitam ser coletados para detectar certos tipos de intrusões.

Uma possível fonte de informação sobre ataques é o CVE (*Common Vulnerability and Exposure Enumeration*) do MITRE. Uma vez que um número suficiente de sensores é implementado e testado, novos ataques são feitos contra o sistema para determinar se e como os sensores existentes reagirão a novos ataques. Uma origem de informação de novos ataques pode ser, CERT *Advisories* www.cert.org/advisories/, *Bugtraq* e *SecurityFocus database* www.securityfocus.com/.

Interpretação e análise dos dados produzidos pelos sensores serão feitas para ajudar a mostrar a validade da hipótese proposta, de que novos ataques podem ser detectados usando-se sensores existentes. Espera-se detectar uma percentagem significativa de novas tentativas de ataques contra o sistema. Mesmo que a hipótese não puder ser validada, o resultado será útil para fornecer uma evidência de que não existe tipo geral de dados que possam ser coletados para detectar novos ataques e que para cada novo ataque será necessário o desenvolvimento de sensores especializados para sua detecção.

Todos os sensores serão implementados na mesma plataforma, para reduzir os diferentes sensores que tem que ser testados. Sensores serão projetados para detectar tentativas de ataque e não se os ataques foram bem sucedidos, o ataque não precisa existir numa plataforma de implementação atualmente, e de fato pode ser um ataque para uma plataforma completamente diferente. Neste sentido os sistemas trabalharão como *honeypots* universais, porque eles serão capazes de aceitar e monitorar ataques para diferentes plataformas.

A maioria dos IDS baseados em *host* que existem hoje tem sido projetada em torno de dados que o sistema operacional torna disponível na forma de *audit trail, logs* de processos de contabilidade, etc. A idéia é decidir que dados coletar e os mecanismos que devem ser implementados para coletá-los. Embutir sensores apropriados dentro do SO e dentro dos

programas afetados, de maneira que possam monitorar qualquer atividade no sistema e obter qualquer informação que possa contribuir para a detecção de um ataque. Embutindo os sensores dentro do código afetado é fazer monitoramento do alvo. É diretamente estudar as atividades do objeto, ao invés de indiretamente através do *audit trail*, com a vantagem de reduzir o risco de um intruso modificar a informação antes de chegar no sensor.

A arquitetura AAFID é suficientemente geral para fornecer uma infra-estrutura necessária para implementar diferentes tipos de IDS. O protótipo AAFID será usado como plataforma de suporte para coleta de dados necessários para o trabalho.

O sistema operacional escolhido foi Open BSD por ter origem aberta o que facilita a incorporação de sensores no *kernel* e nos programas do sistema. Sua árvore de origem é gerenciável e distribuída como uma árvore de diretório simples, imita de perto o layout do próprio sistema, facilitando a localização do código para diferentes programas e subsistemas. A maioria dos problemas de segurança para os quais os sensores serão implementados já foram fixados no Open BSD. Procurando nos *patches* de segurança e no *log* modificado de cada arquivo pode ser mais fácil localizar as porções do código onde os problemas existem. Isto ajuda a determinar onde os sensores para cada intrusão têm que ser colocados.

Os dados coletados pelos agentes do AAFID foram analisados centralmente usando-se técnicas de *clustering* e os resultados mostraram que atribuições de *clustering* nos dados apontaram mudança durante o período que os ataques foram desempenhados contra as máquinas da rede.

GASSATA – GENETIC ALGORITHM TOLL FOR SIMPLIFIED SECURITY AUDIT TRAIL ANALYSIS

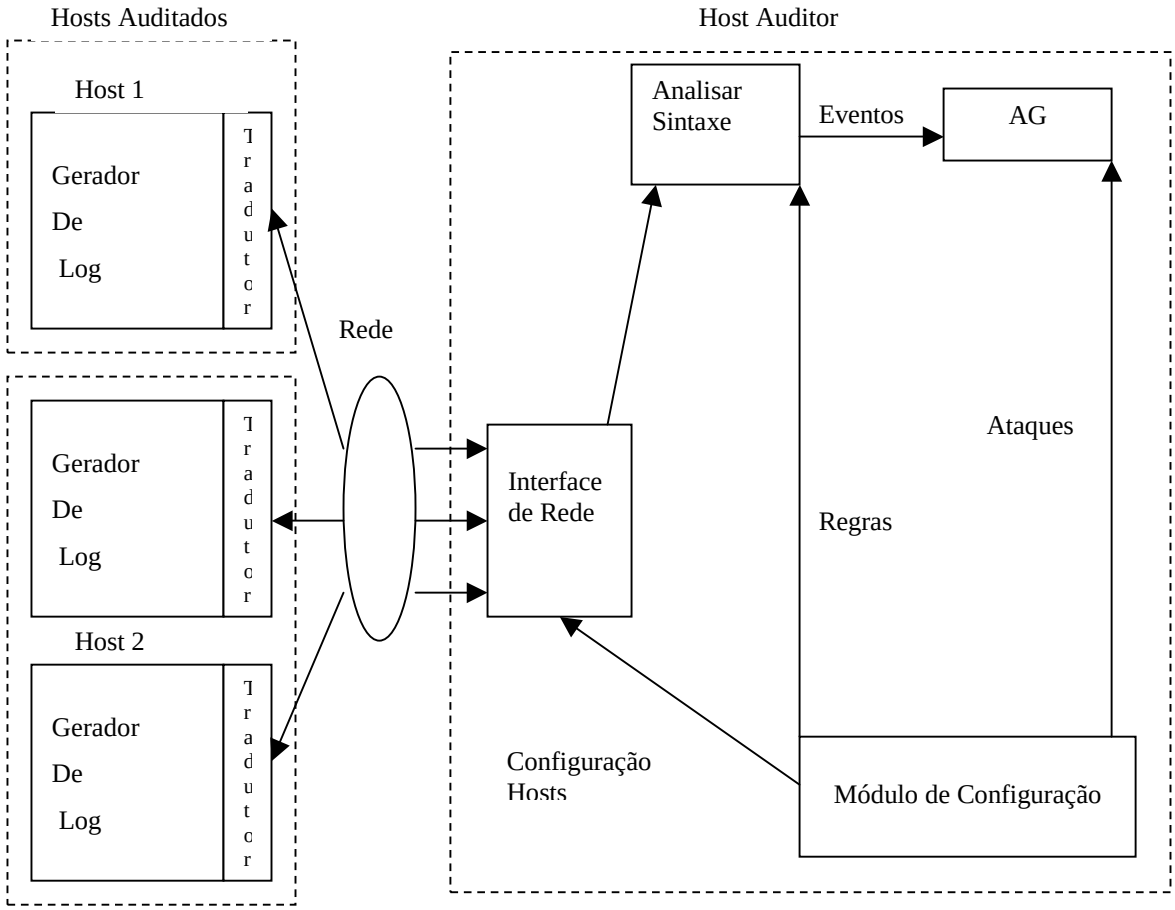

Figura: Arquitetura do GASSATA

O sistema nasceu do problema de segurança da análise do *audit trail.* É um sistema baseado em *host* para detecção baseada em conhecimento. Ele não trabalha em tempo real, divide as gravações de auditoria em segmentos de 30 minutos e busca por assinaturas de ataques usando algoritmos genéticos.

GASSATA pode detectar mais do que 200 ataques, mas exige acima de 10 minutos para fazer isto. Seu sistema de resposta é limitado a relatórios mostrados na interfase do usuário.

Neste sistema, os algoritmos genéticos são aplicados ao problema de classificação de eventos do sistema, pelo uso de um conjunto de vetores hipótese H, um vetor por fluxo de eventos de

interesse, de n dimensões, onde n é o número de ataques conhecidos. Hi é definido como 1 se representa um ataque e como 0 se não.

A função tem duas partes. Na primeira, o risco que um ataque particular representa para o sistema é multiplicado pelo valor do vetor hipótese. O produto é então ajustado para uma função de penalidade quadrática para eliminar hipóteses não realísticas. Este passo melhora a discriminação entre ataques possíveis. O objetivo do processo é otimizar os resultados desta análise, assim a probabilidade de um ataque detectado ser real, sinaliza 1 e a probabilidade de um ataque detectado ser falso, sinaliza 0.

As seguintes desvantagens são observadas para abordagem de detecção de mal uso:

- A forma de expressão binária, para fluxos de eventos individuais, leva o sistema a não detectar múltiplos ataques de forma simultânea. Existe a possibilidade de que algoritmos genéticos não binários possam resolver o problema.
- Se o mesmo evento ou conjunto de eventos é comum para muitos ataques e um atacante usa isto para executar múltiplos ataques simultâneos, o sistema não pode encontrar um vetor de hipótese ótimo.
- Sistema não localiza ataques, precisamente, no *audit trail*. Nenhum sentido de temporalidade ocorre nos resultados de um detector.

GBID Genetic Based Intrusion Detection

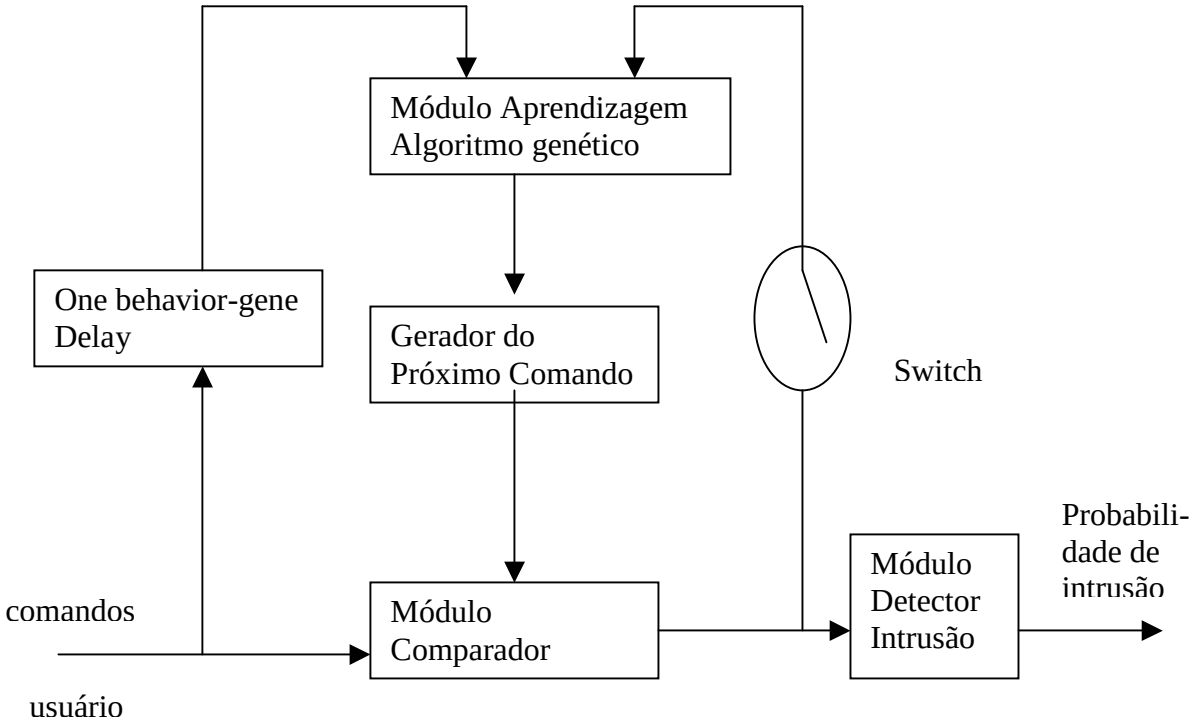

Figura : Modelo de detecção de intrusão baseado em algoritmo genético.

IDS baseado em comportamento, observa anomalia no uso de comandos por usuários individuais. O comportamento do usuário é aprendido continuamente para capturar desvio que resulte em um número menor de alarmes falsos na detecção de intrusão. Todo mundo exibe regularidades em suas vidas diárias. Usuários de sistemas computacionais respondem a uma situação de uma maneira similar, quando a mesma situação ocorre, mostrando regularidades, que podem ser usadas para classificação de usuários no ambiente interativo. Algoritmos genéticos são usados para aprendizagem do comportamento do usuário num sistema computacional, por sua robustez e adaptabilidade a mudanças no ambiente. São algoritmos de busca baseados em mecanismos de seleção natural e genética natural. Os genes são a unidade atômica que representa propriedades no ambiente, sofrem um processo de evolução que pode ser usado com uma técnica heurística para resolver vários problemas em diferentes ambientes. No processo de aprendizagem do comportamento, o primeiro passo é a codificação, vários comandos do usuário são mapeados na forma de gene, num processo de evolução natural. Alfabetos (símbolos únicos) são atribuídos para cada comando usado na sessão do usuário. Comandos da sessão do usuário são divididos dentro de string de alfabetos de tamanho n, chamado *behavior-gene*. Usou-se 4353 alfabetos para

representar todos os possíveis comandos no desenvolvimento de *behavior-gene*. A população aleatória de *behavior-genes* é gerada a partir de alfabetos vistos assim distantes.

A função *fitness* é calculada por uma coleção de *behavior-genes*. *Behavior-genes* com a mais alta aptidão na geração corrente são propagados para a próxima geração para obter o *behavior-gene* mais apto, sua determinação apropriada é importante para melhorar a exatidão na predição. O processo de evolução é mostrado na figura 1(inserir figura). Um parâmetro que interfere na função *fitness* é o índice de entropia no comportamento do usuário. Quando o usuário exibe mudança freqüente no comportamento resulta em grande valor de entropia e indica que o próximo comportamento do usuário não pode ser predito com exatidão e vice-versa.

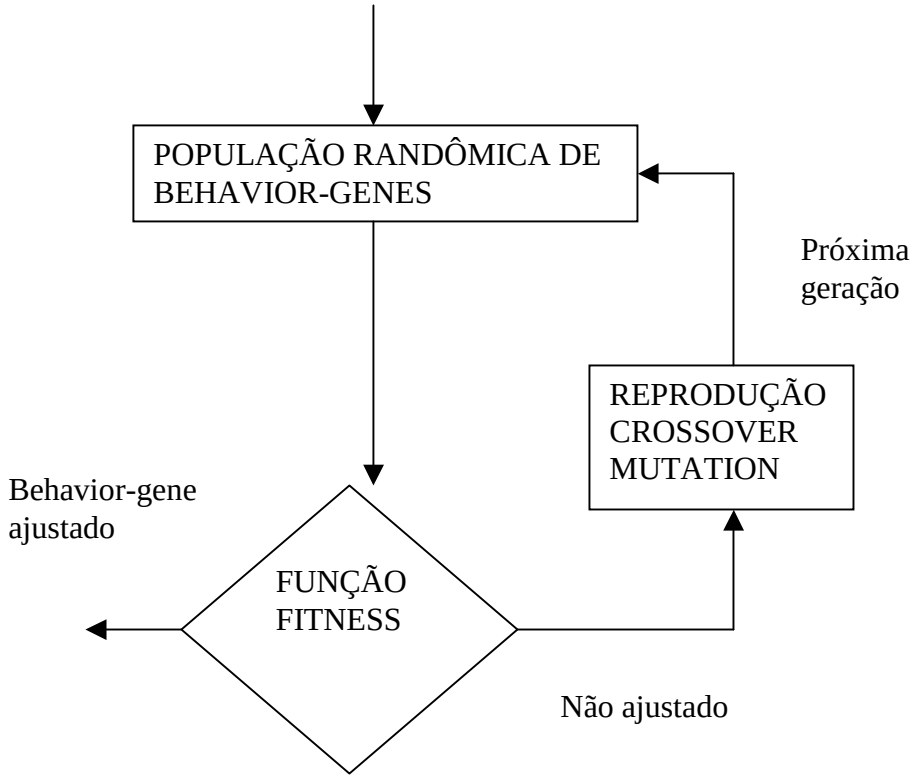

Figura: Algoritmo genético na aprendizagem do comportamento.

Na abordagem GBID três operadores genéticos são aplicados na população aleatória de *behavior-genes*:

- Reprodução: seleciona *behavior-genes* com relativamente mais alto valor de aptidão a partir da população aleatória inicial para gerar novos *behavior genes* para aplicar *crossover* e *Mutation*: Esta operação é desempenhada em *behavior-genes* na geração corrente, antes de aplicar outros operadores genéticos;

- *Crossover*: Duas diferentes partes de *behavior-gene* na geração corrente são concatenadas para formar novos *behavior-genes*. Baseados no número de pontos da concatenação, a operação é chamada *single-point crossover* ou *mult-point crossover;*
- *Mutation*: Alguns dos símbolos no *behavior-genes* são trocados aleatoriamente para que evolua um novo *behavior-gene*. Este processo é aplicado quando operações de *crossover* não estão gerando suficientes *behavior-genes* novos.

O comportamento é caracterizado na abordagem por 3 (três) *tuple:*

- Match *index*: é um parâmetro que mede a regularidade de comportamento do usuário;
- Índice de entropia: É a medida do comportamento dinâmico do usuário, também a medida da distribuição de comandos na amostra de comandos;
- *Newness index*: É a medida do número de novos comandos, que não tinham ocorrido antes.

O bloco semântico do modelo GBID é mostrado na figura a seguir. Alfabetos mapeados de comandos de usuário são usados como fluxo de entrada para o modelo.

Comandos do usuário	**Alfabetos (code)**
ls	1
clear	41
ps	8
clear	41
cc <arquivo>	14
ls	1
cc <arquivo>	14
...	...

Figura: Mapeamento dos comandos do usuário para alfabetos.

O módulo de aprendizagem aprende a tendência do valor da entropia pelos comandos do usuário através de algoritmos genéticos. A exatidão do *behavior-gene* predito decide o valor de 3-tuple que por sua vez afeta a exatidão da detecção de intrusão.

Gerador do próximo comando gera os próximos comandos do usuário que ocorrem a partir do *behavior-gene* predito.

O módulo comparador calcula o valor do 3-tuple a partir dos comandos ocorridos realmente e comandos preditos. O valor do 3-tuple do comportamento normal esperado é comparado com o valor do 3-tuple calculado para calcular o desvio no comportamento do usuário. Um *switch* é usado para controlar o *feed back* para o módulo de aprendizagem. Quando o número de alarmes falsos é alto, o *switch* é fechado para adaptar a mudança no comportamento do usuário.

O módulo detector de intrusão calcula a probabilidade da amostra do comando corrente ser intrusiva a partir do desvio no comportamento do usuário.

O modelo proposto tem as seguintes características:

- Conhecimento de novos ataques e vulnerabilidades do sistema não é necessário, desde que a detecção é feita pela observação de desvio nos padrões de comportamento do usuário;
- Detecção de intrusão pode ser feita em tempo real. Neste modelo a quantidade de logs de dados processada para detecção intrusão é baixa;
- Tem baixa taxa de alarmes falsos. O *switch* é fechado para aprender o comportamento do usuário, quando este está no modo exploratório;
- O comportamento do usuário é continuamente aprendido, para captar o rumo (tendência) do comportamento do usuário;
- GBID opera de uma maneira descentralizada. O algoritmo GBID roda em cada nó de maneira independente.

A exatidão da predição no modelo de aprendizagem, quase permanecem constante (com uma pequena mudança de 0,01 ou menos) para diferentes tamanhos de história (intervalo observado), para diferentes usuários. Este fenômeno pode ser explicado pelo fato do próximo comando que ocorre ser mais dependente do comando próximo passado do que toda história.

O tamanho do período inicial de observação determina o período durante o qual o GBID aprende o comportamento do usuário, antes de começar a detecção de intrusão. A predição dos próximos

comandos do usuário ocorrerem é baseada nos comandos próximos passados do usuário. Se mais números de novos comandos não vistos ocorrem então a exatidão da predição do modelo de aprendizagem decresce. A predição pode ser começada quando o número de novos comandos, no conjunto de comandos do usuário, torna-se constante para alcançar grande exatidão na detecção de intrusão. Por observação, no modelo, a detecção de intrusão pode ser começada depois dos iniciais 500 comandos ocorridos, para se obter a máxima exatidão na predição.

No algoritmo usado no GBID, todo o intervalo de comandos observados, na sessão do usuário, é dividido em blocos de tamanho fixo chamados de amostras de comando. Os valores da 3-tuple para as amostras de comando na sessão do usuário são calculados e formam o conjunto S. O valor da 3-tuple da amostra de comando é comparada com o valor da 3-tuple inicial de entrada, para remover amostras de comando do conjunto S, que estão desviando dos valores de entrada. O processo de remoção de amostras de comando é repetido até as amostras de comando na sessão corrente do usuário seja removida pelo aumento dos valores de entrada pelo fator de entrada, no próximo ciclo. O número de amostras de comando restantes no conjunto, quando a amostra de comando corrente estando eliminada do conjunto S determina a probabilidade da sessão corrente ser intrusiva.

A performance do sistema GBID na detecção de intrusão é avaliada através de dois parâmetros:

- Exatidão da intrusão: Em IDS existentes o calculo da exatidão da detecção de intrusão varia dependendo do domínio e definição da intrusão. Na abordagem GBID, a intrusão é um conjunto de ações que desviam do comportamento normal do usuário. A probabilidade de uma amostra de comando ser intrusiva é a mesma que exatidão da detecção na abordagem GBID;
- Taxa de alarme falso: É a medida da contagem dos exemplos no qual uma amostra de comando do usuário autêntica (genuína) é classificada como uma amostra intrusiva. É a probabilidade da amostra de comando corrente ser classificada como intrusiva, embora seja não intrusiva.

A avaliação experimental do sistema GBID mostrou que a abordagem é capaz de detectar intrusão com uma exatidão de 96,8% e uma taxa de alarme falso de 3,2 %, na vida real da história de comandos UNIX de 25 usuários.

Hyperview

O Hyperview é um sistema com dois componentes importantes, o primeiro um SE que monitora *audit trail* por sinais de intrusões conhecidas e o segundo uma rede neural baseada em componentes que aprendem o comportamento do usuário adaptativamente e levantam um alarme quando o *audit trail* desvia do comportamento aprendido. O emprego da RN para funções de detecção baseada em comportamento estatístico se firmou na hipótese de que os dados de auditoria poderiam conter. A hipótese era de que continham séries temporais multivariadas, onde o usuário constitui um processo dinâmico que emite uma seqüência ordenada de eventos. A gravação de auditoria que representa tal evento consiste de variáveis de dois tipos. Os valores dos primeiros vêm de um conjunto finito de valores, por exemplo, nome do terminal no qual o comando foi emitido. O segundo tem um valor contínuo, por exemplo, uso da CPU. A proposta era mapeamento de séries temporais para as entradas da RN, reconhecendo assim que isto faria um modelo simples que poderia ser facilmente treinado. No entanto devido a uma série de problemas, decidiu-se por um caminho diferente. O emprego de uma rede recorrente RR, onde parte da saída da rede era conectada a entrada da mesma, formando a entrada para o próximo estágio. Isto cria uma memória interna na rede, a proposta é a mesma, dotar a rede com a percepção do passado. É interessante notar que a RR tem memória longa sobre os parâmetros dos processos, na forma de peso das conexões na rede e memória curta sobre a seqüência sob estudo na forma de ativações dos neurônios.

O projeto do sistema como um todo é complexo, a RN é conectada a dois SE. Um sistema monitora a operação e o treinamento da rede, para prevenir a rede de aprender comportamento anômalo, por exemplo e avaliar sua saída. O outro SE varre os dados de auditoria a procura de padrões conhecidos de abuso e junto com a saída do primeiro SE (e, portanto da RN) forma uma opinião se dispara um alarme ou não. A decisão do SE também fornece a RNA com dados "situação de consciência", dados que o *audit trail* não contém, simples data e hora corrente, a um complexo estado de alerta ou estado de perigo do sistema, definido pelo Administrador de Segurança.

IDA – Intrusion Detection Agent system

O método trabalha usando marcas deixadas por intrusos suspeitos (MLSIs – *Marks Left by Suspected Intruders*). Recuperação da informação e pistas das rotas de intrusão são feitas pelo uso de agentes móveis. Como intrusos podem atacar fora de seu *host* base, a proposta é seguir a pista de origem de uma intrusão.

O protótipo conta com duas funções completas:

- Detecção de ataque local;
- Traçar a pista de intrusão numa LAN.

A próxima função é detectar ataques remotos, seguir a pista de intrusões na internet, e estender a aplicabilidade do sistema para redes de grande escala.

IDA trabalha observando eventos que possam relacionar com intrusões, ao invés de analisar todas as atividades dos usuários. Se uma marca suspeita é achada, a informação é colhida, analisada e decide se ocorreu a intrusão ou não. Por exemplo, ele monitora se arquivos críticos relacionados com a segurança do sistema foram modificados. No entanto, usuários legítimos também podem fazer alterações nos arquivos e o sistema não pode somente se basear na modificação de arquivos, para uma intrusão ter ocorrido. Assim IDA colhe mais informações relacionadas a modificações no arquivo, antes de decidir se a intrusão ocorreu.

Passos executados pelo IDS:

1. Cada sensor busca marcas suspeitas no *log* do sistema;
2. Se o sensor detecta a marca, ele a reporta para o gerente;
3. O gerente dispara um agente que segue a pista para um sistema alvo onde a marca foi detectada;
4. O agente chega no sistema alvo e ativa um agente de colheita de informação;
5. O agente então coleta a informação relacionada as marcas suspeitas no sistema alvo;
6. Depois de ativar o agente de colheita, o agente que segue a pista investiga o ponto de origem das marcas suspeitas e se esforça para identificar o *site* remoto do usuário. Este agente pode acumular dados sobre conexões de rede, processos rodando no sistema;
7. Depois de coletar a informação, o agente de coleta, independente do agente de seguir a pista, retorna ao gerente e incorpora a informação no *bulletin board*;

8. O agente que segue a pista move para o próximo sistema alvo, na rota traçada e ativa um novo agente de coleta de informação;
9. Se o agente que segue a pista chega na origem da rota ou não pode mover para lugar nenhum. Ou se outros agentes que seguem a pista perseguiram a rota, ele retorna para o gerente.

O sistema IDA consiste de:

- Gerente. Analisa a informação colhida pelos agentes de coleta - IA e detecta intrusões. Ele gerencia os agentes móveis e *bulletin board* – BB e fornece uma interface entre o AS e o sistema. O gerente acumula e pesa a informação entregue pelo agente móvel no BB e se o peso excede a um conjunto de entrada, o gerente conclui que uma intrusão ocorreu. Um gerente reside em cada segmento de rede.
- Sensores. Presentes em cada sistema alvo, monitora os *logs* do sistema em busca de marcas suspeitas. Se o sensor acha uma marca ele relata ao gerente, como também o tipo de marca.
- Agente segue a pista. Este agente segue a pista de uma intrusão e identifica seu ponto de origem. O lugar no qual o usuário deixou sua marca remotamente no *host* alvo. No percurso da rota o agente pode achar qualquer nó intermediário comprometido. O agente não pode migrar para um sistema no qual o IDA não esteja instalado.
- Agente coleta. Este agente que é móvel recolhe informações relacionadas a marcas no sistema alvo. Cada vez que o agente que segue a pista, na perseguição de um intruso, é disparado no sistema alvo, ele ativa um agente de coleta naquele sistema. Então, dependendo do tipo de marca, o agente de coleta recolhe a informação, retorna ao gerente e relata.
- *Bulletin Board* – BB e *Message Board* -MB. São áreas de uso comum que podem ser acessadas pelo agente que segue a pista e agente de coleta e também um meio de troca de informações.

Existe um MB em cada sistema alvo, usado por agentes que segue a pista para troca de informações. Qualquer agente que segue a pista pode conhecer se uma pista sob sua investigação já tem sido traçada por outro agente e pode usar esta informação e decidir aonde ir. O BB está na

máquina gerente e é usado para gravar informações colhidas dos sistemas alvo pelos agentes de coleta, também para integrar a informação colhida sobre toda rota traçada.

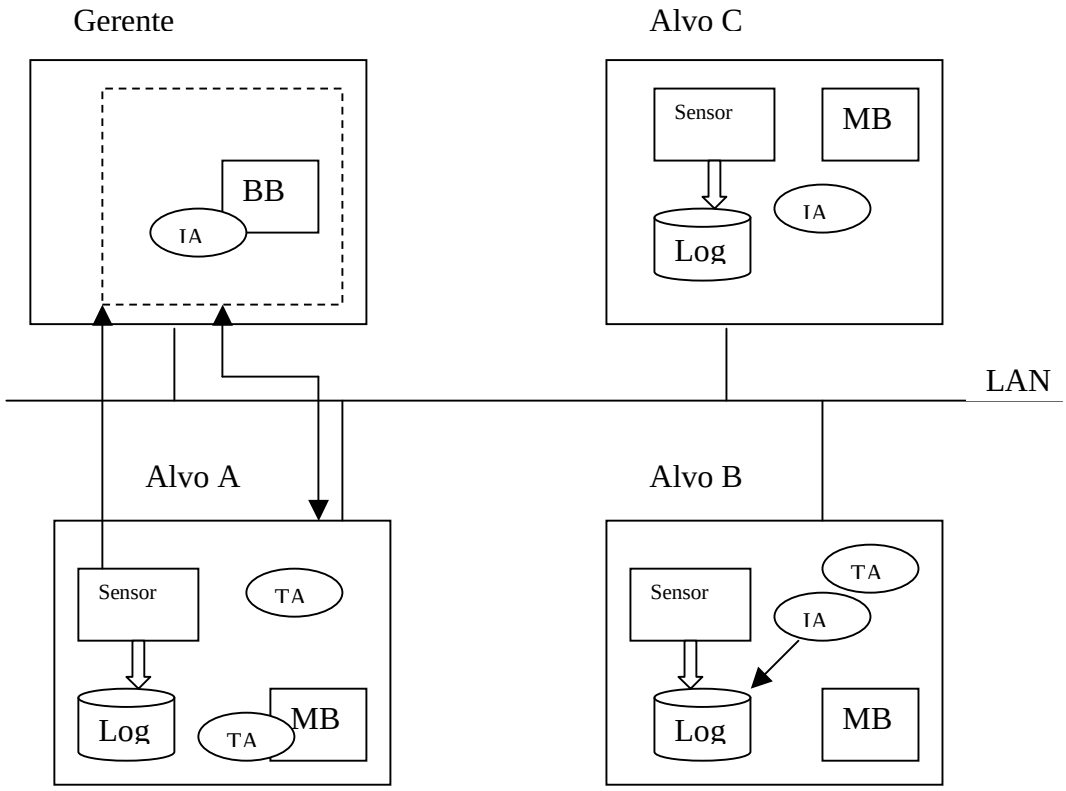

Figura : Arquitetura do sistema IDA.

IDES - INTRUSION DETECTION EXPERT SYSTEM

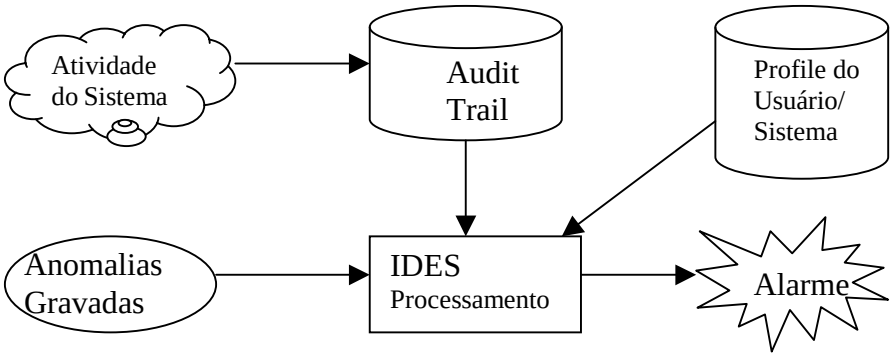

Figura: Modelo IDES

Inicialmente foi projetado como simples sistema baseado em regras, usando cenários de intrusão descritos por um conjunto de regras. O componente baseado em regras foi baseado no mesmo *Production-Based Expert System Toolset (P-BEST)* que MIDAS[28] usou. Embora este sistema baseado em regras seja útil, apresenta obstáculos como a falta de suporte para o desenvolvimento de cenários de intrusão. É difícil se determinar a relação entre as regras.

Para superar esta dificuldade, o conceito de detecção de intrusão baseado em modelo foi desenvolvido conjuntamente. Cada cenário de intrusão foi separadamente modelado. Assim, o número de regras necessárias para mudanças é de um tamanho mais manejável. Esta abordagem organiza as regras para o cenário de intrusão, sendo que somente as regras usadas para checar os passos iniciais da intrusão são disparadas, enquanto as outras continuam dormentes. Quando o cenário de intrusão se inicia, regras adicionais para detecção de passos subseqüentes de intrusão podem ser acrescentadas ao conjunto de regras que devem ser avaliadas. Na abordagem baseada em regras, nenhuma das regras permanece dormente. Todas são constantemente avaliadas. Abordagem baseada em modelo ganha em eficiência e manutenção.

[28] Multics Intrusion Detection and Alerting System, desenvolvido pela National Computer Security Center (NCSC), disponibilizado em 1989 [Bace, 2000].

Informações do *audit trail* são coletadas para um *log* protegido, sendo possível estatísticas baseadas em *profile*. Outros tipos de análise também o são usando-se técnicas e ferramentas off-line.

IDS, segundo Denning, são modelados como os seguintes objetos matemáticos.

<subjects, objects, profiles, audit records, anomaly records, alarms>

Subjects correspondem aos processos básicos que iniciam as atividades.

Objects correspondem aos arquivos e diretórios usados para armazenar a informação.

Profiles são características de comportamento de usuários, grupos de usuários, *hosts* remotos e sistemas alvo. Estes *profiles* são atualizados para refletir novos comportamentos uma vez ao dia, depois da vida útil [29]do *profile* original.

No caso do usuário ser novo, ainda não conhecido pelo sistema, IDES usa um *profile default* para dar início ao monitoramento daquele usuário.
Gravações de auditoria são incluídas para modelar as estruturas de dados usadas para capturar o comportamento observado no sistema. A segurança do sistema operacional envolve o uso de *audit trail*, que são compostos de seqüência de *audit records*. Cada gravação pretende contribuir para a visão temporal do sistema ou atividade do usuário no ambiente alvo.

As gravações de anomalia são incluídas para modelar as estruturas de dados usadas para capturar decisões de programas baseadas na análise de intrusões.

Alarme é a maneira na qual problemas são relatados. Correspondem a ações de respostas tomadas depois de uma anomalia ter sido detectada.

[29] 30 dias

IDIOT

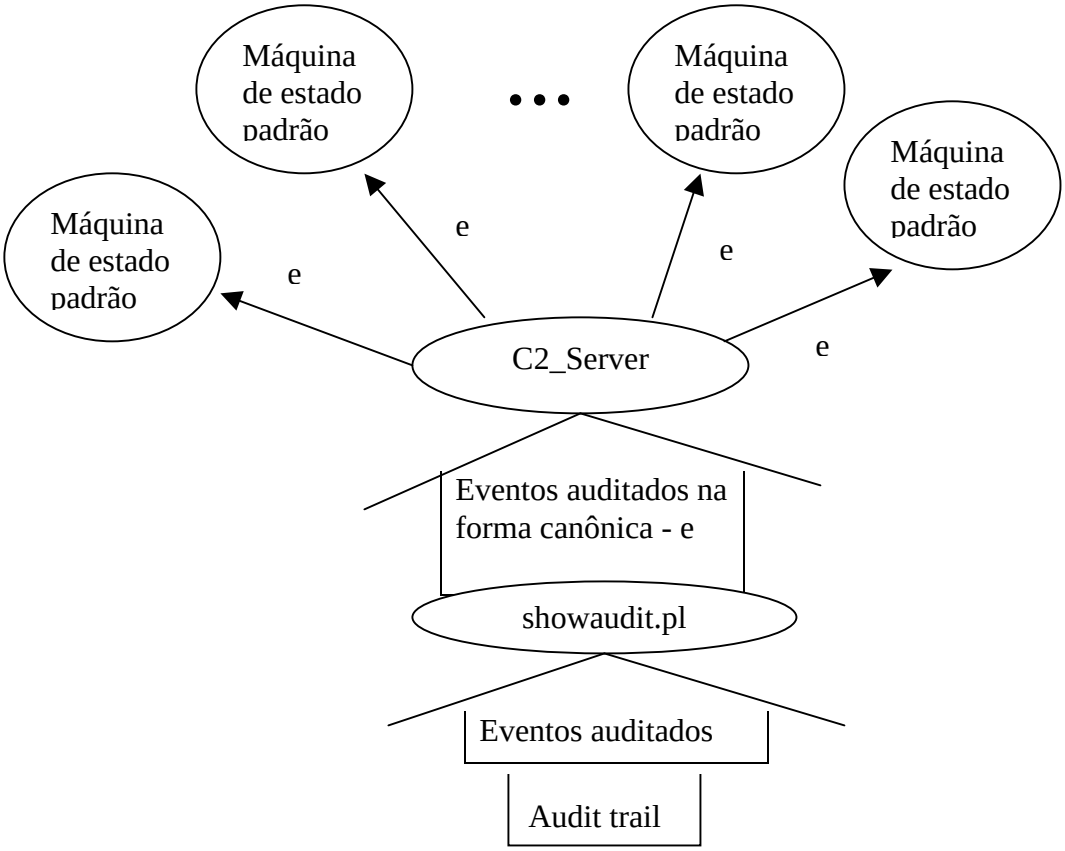

Figura : Estrutura do sistema IDIOT.

É um sistema que o princípio básico é empregar redes de Petri coloridas -RPC, para detecção de intrusão baseada em conhecimento. A abordagem em camadas é sugerida quando técnicas baseadas em assinaturas são aplicadas ao problema de DI. É uma técnica que permite o casamento condicional de padrões e se prestam a uma representação gráfica. Os padrões desempenham um papel principal e são escritos em uma linguagem textual simples e então analisados, resultando em uma nova máquina de casamento de padrão. Esta nova máquina pode ser dinamicamente acrescentada a uma instância já rodando do IDIOT, via a interface do usuário. Além disso, o usuário pode estender o IDIOT a reconhecer novos eventos auditados.

O sistema IDIOT consiste de 4 (quatro) componentes:

- Audit trail (Solaris 2.4 e Sun BSM). O sistema transforma dados do *audit trail* para a forma canônica.

- Showaudit.pl É um *script perl* que converte o *audit trail*, por exemplo Sun BSM para o formato canônico necessário ao IDIOT. Se o IDIOT for utilizado em outra plataforma, este componente deve ser reescrito, para o novo formato do *audit trail*.
- C2_Server. É o centro do sistema, consiste de uma classe C++ com muitos métodos associados a ela.
- Padrões descritivos. São escritos numa linguagem que descreve um método conhecido de ataque como um conjunto de estados e transições entre eles. Devem ser traduzidos para C++, compilados e ligados ao objeto C2_Server.

Audit trail e *showaudit.pl* são máquinas dependentes, enquanto o C2_Server e padrões descritivos são portáveis. Um quinto componente C2_appl fornece uma interface para o usuário interativa, também portável.

LISYS – Lightweight intrusion detection System

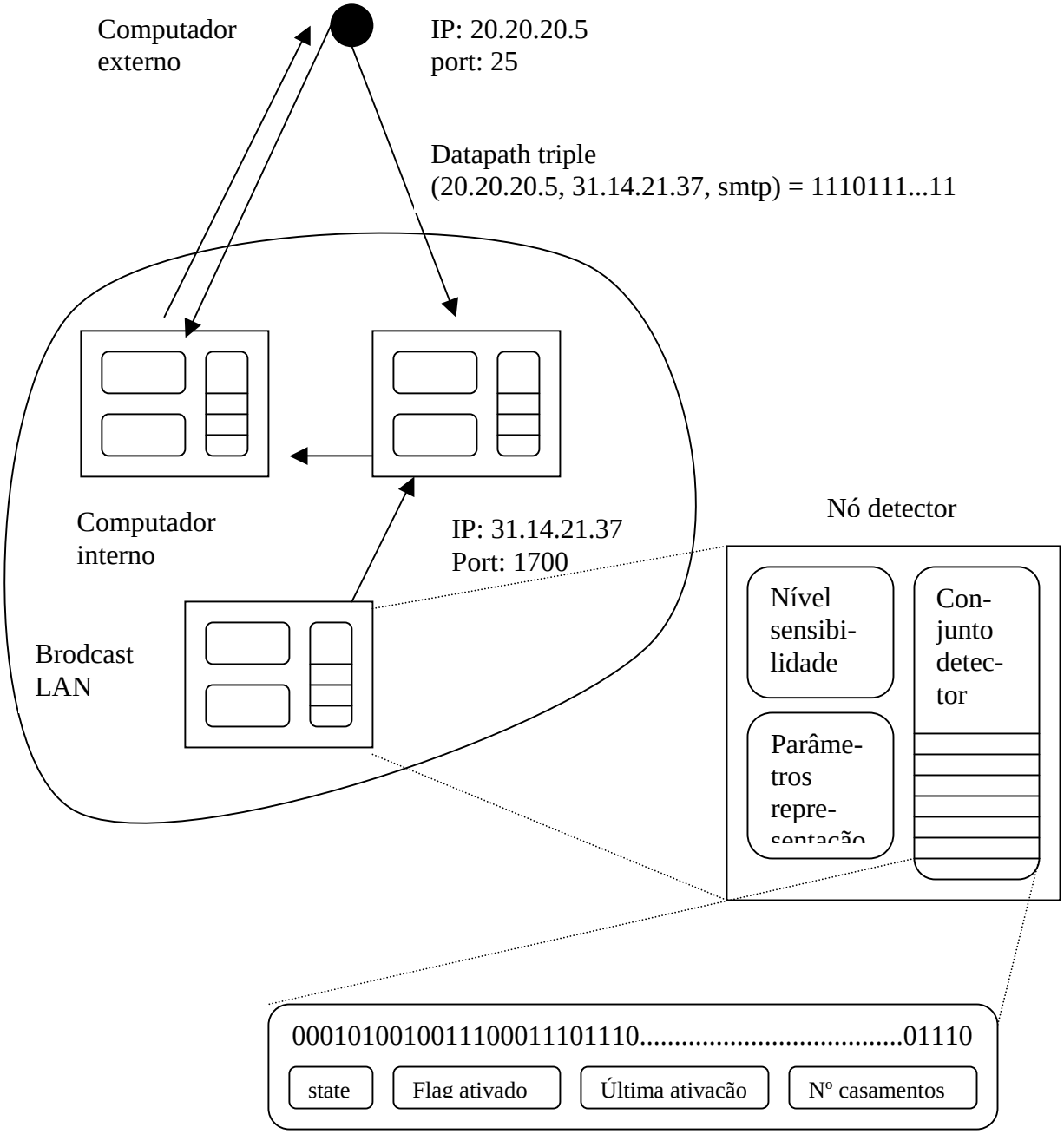

Figura : Arquitetura do Lisys.

Forrest e Hofmeyr apresentam um método de detecção de intrusão, baseado no sistema imunológico humano. A definição de *self* é feita em termos de seqüências curtas de chamadas ao sistema executadas por processos privilegiados, provendo uma assinatura compacta do comportamento normal dos processos. O método é baseado na construção de uma base de dados do comportamento normal de cada programa de interesse. Cada base de dados é específica para uma determinada arquitetura, versão e configuração de software, política de administração e

padrões de uso. Depois de construída, a base de dados é utilizada para monitorar o comportamento do programa relacionado. As seqüências de chamadas ao sistema da base de dados formam o conjunto de padrões normais do processo, e as seqüências não encontradas na base de dados indicam anomalias no comportamento do processo. O método proposto apresenta dois estágios. No primeiro estágio, são extraídos traços de comportamento normal dos processos e construídas as bases de dados. No segundo estágio, as seqüências de chamadas ao sistema executadas pelos processos são comparadas com os padrões armazenados em suas bases de dados. Se uma seqüência não é encontrada na base de dados do processo, é reportado um *mismatch*, representando uma possível invasão. Testes foram feitos com programas como, *sendmail* e *lpr*.

Eles também apresentam um sistema imunológico artificial, que detecta conexões TCP não usuais com a rede protegida pelo sistema. A definição de *self* é feita em termos de conexões TCP, determinadas pelos endereços IP de origem e destino, e pela porta TCP de serviço. Cada conexão é representada por uma string de 49 bits, sendo que o *self* é o conjunto das conexões que normalmente são observadas na rede. O sistema apresenta um único tipo de detector, que combina propriedades de várias células do sistema imunológico, podendo assumir diferentes estados. Cada detector é representado por uma string de 49 bits e um conjunto de estados.

A técnica descrita tenta descobrir novas intrusões, mas o sucesso somente tem sido em escala experimental. O uso de uma analogia imunológica é, em princípio, muita poderosa.

MIDAS

É construído ao redor da idéia de heurísticas de DI, onde o AS analisa seus *logs* de auditoria manualmente a procura de evidência de comportamento intrusivo. MIDAS aplica um conjunto de ferramentas de SE baseado em produção para DI. Este conjunto de ferramentas, P-BEST's é escrito em *Lisp* e produz um código que pode ser compilado e rodar em uma máquina *Lisp* simbólica dedicada. A compilação do código do SE dentro do código do objeto fornece uma execução eficiente do Shell do SE.

A base de regas é feita em três categorias distintas e sua estrutura é de duas camadas enfileiradas. A primeira camada mais baixa deduz imediatamente certos tipos de eventos como número de *logins* ruins e afirma se uma entrada particular suspeita foi alcançada, quando os eventos dispararam. Estas suspeitas são então processadas pelas regras da segunda camada, que decide se dispara um alarme baseado nos fatos suspeitos afirmados pelas regras da camada mais abaixo. Por exemplo: este usuário é um intruso porque digitou 40 comandos errados nesta sessão, tentou comando inválido *suid* e *login* em horário não usual. Considerados em conjunto, é uma forte indicação de que alguma coisa foi confirmada pelas regas do segundo nível.

NFR – NETWORK FLIGHT RECORDER

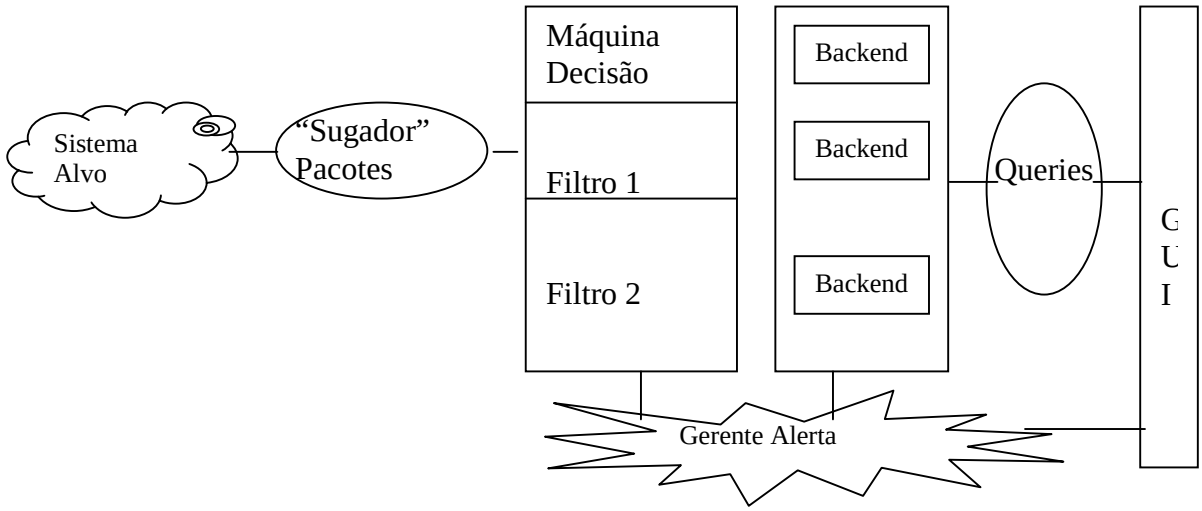

Figura: Configuração de Alto Nível da Arquitetura do NFR

Não é uma ferramenta de detecção de intrusão baseada em rede [Bace, 2000], mas um monitor de rede genérico com APIs para suportar o acréscimo de analisadores de intrusão. É, essencialmente, um kit de ferramentas para construção de analisadores de tráfego, com abordagem de detecção baseada em conhecimento que usa assinaturas. E gravação de eventos estatísticos, com abordagem de detecção baseada em comportamento que usa *profile* de comportamento normal.

A configuração geral do sistema NFR em alto nível é esquematizada na figura acima. Alguns aspectos arquiteturais e de segurança incluem:

- Os "sugadores" de pacotes, que operam para capturar[30] o tráfego de rede e encaminhar cópias dos pacotes para máquinas de decisão. A questão é como armazenar grandes volumes de dados usando dispositivos de *drivers*. A idéia é proporcionar facilidades de captura de pacotes baseadas em dispositivos de hardware. O casamento de um software flexível NFR e uma rápida captura de pacotes baseados em hardware é algo benéfico para a área de detecção de intrusão.

[30] NFR usa uma versão modificada do libpcap para promiscuamente e passivamente extrair parte principal de pacotes da rede.

- A máquina de decisão é baseada numa coleção de filtros escritos numa linguagem de programação especial chamada N[31], onde são compilados e interpretados. Casamento de padrão é desempenhado para permitir que os pacotes sejam reagrupados. Todos os filtros são aplicados nos pacotes.

- A extração de dados dos filtros, depois da operação de filtragem, é feita através de uma primitiva *alert*, que passa os alarmes para um sistema designado para gerenciamento de alarmes e, de uma primitiva *Record*, que passa a estrutura do dado para o *backend* [32]para desempenhar o processamento de intrusão. Esta modularidade é útil, pois permite o uso de rotinas terceirizadas para desempenhar o gerenciamento de alarmes ou a gravação.

- O processamento no *backend* é feito via uma pequena coleção de programas de propósito geral, que fornecem histogramas de dados, o que facilita a captura de dados numa matriz multi dimensional. A totalidade de categorias relevantes é acumulada em células da matriz. Alertas podem ser gerados baseados nos números absolutos ou nas freqüências relativas dentro das células.Também fornece listas cronológicas de gravações de informação, proporcionando um nível de detalhamento não fornecido pela função histograma. Também apresentam modularidade.

- Um conjunto de *queries* é usado para analisar os dados. A capacidade do *query* é fornecida por uma base de dados externa acessada com uma linguagem estruturada SQL.

- Uma Interface Gráfica do Usuário (GUI) é fornecida e operam independentemente. Isto permite análise flexível de diferentes tipos de informação de diferentes *backends.*

- Um subsistema gerente de fila de alertas é usado para fornecê-los. Um *daemon* chamado *alertd* é usado para priorizar e definir rotas de alertas. Os tipos de alerta que o NFR usa incluem mecanismos de entrega tais como, impressão, email e fax.

[31] Marcus Ranum alega ter evoluído de uma linguagem chamada UberMUD

[32] Um grupo de assinaturas que juntas podem detectar múltiplos ataques.

A natureza *on-the-fly* do NFR para detecção baseada em comportamento é complementada pelo seu *backend*, que pode ser mais bem entendido como um método de processamento de *audit trail* off-line.

NFR é mais flexível e completo que o Snort[33]. Foi o primeiro produto que preocupou em vencer os ataques contra o próprio IDS em redes.

Tanto o NFR quanto o Snort operam com uma política de assinaturas fonte aberta.

[33] Descrito neste capítulo.

NIDES - NEXT GENERATION INTRUSION DETECTION EXPERT SYSTEM

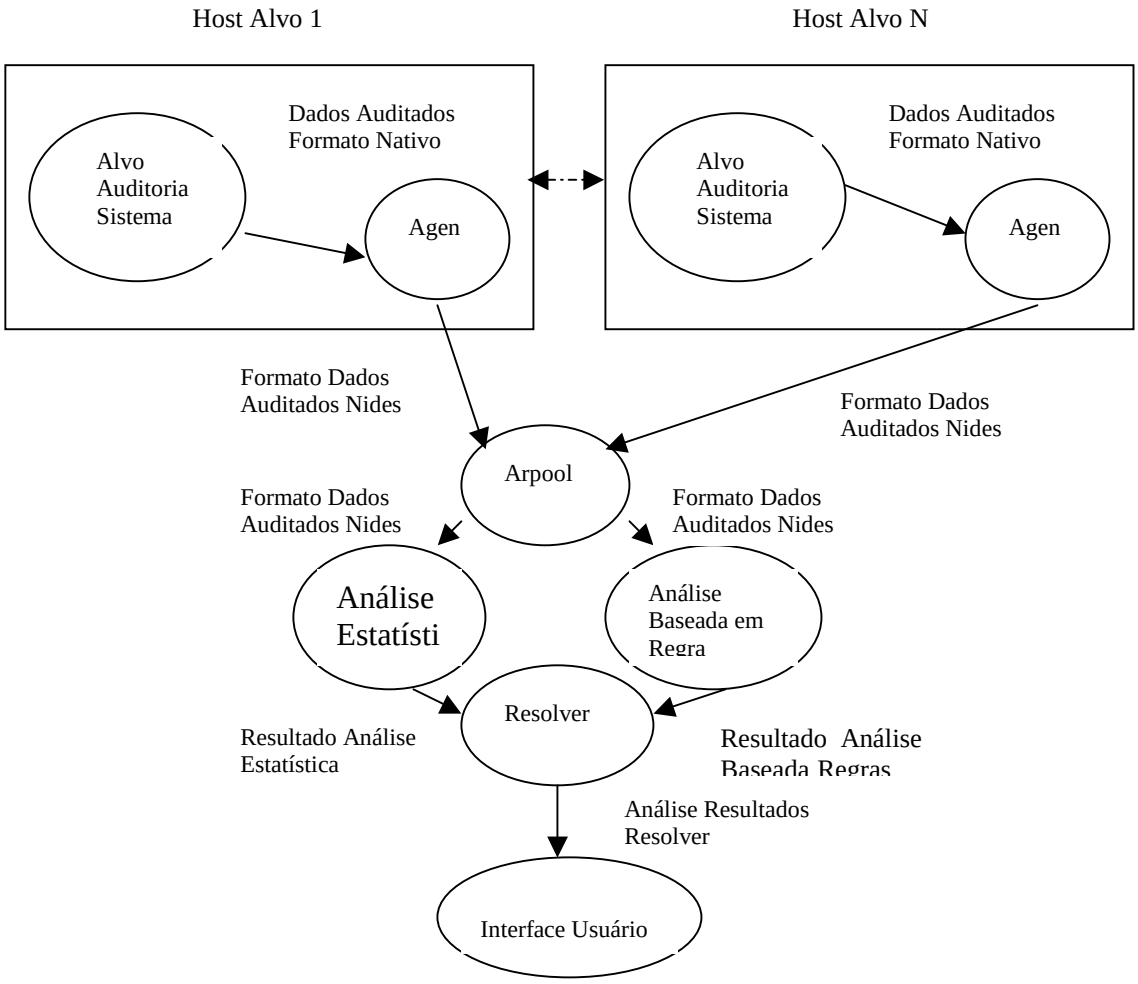

Figura : Arquitetura do NIDES

O *Next Generation Intrusion Detection Expert System* emprega o método baseado em comportamento é utiliza a técnica estatística. Constrói *profiles* estatísticos através do monitoramento de entidades que podem ser usuários, grupos de usuários, estações de rede, *hosts* remotos ou programas de aplicações. É complementado com um componente baseado em conhecimento que utiliza um sistema especialista baseado em assinatura, implementado usando *P-Best*[34].

O desenvolvimento do NIDES se deu graças ao bem documentado IDES, pois foi importante para dar continuidade à pesquisa sobre sistemas de detecção de intrusão. A mesma funcionalidade foi

[34] Utilizado no SDI MIDAS, como no IDES.

mantida do IDES, mas a arquitetura foi alterada. NIDES é modular, com interfaces bem definidas entre os componentes. E construído numa arquitetura cliente-servidor.
Tem análise centralizada num *host* específico, chamado NIDES *host*, que coleta dados de vários *hosts* na rede. Estes *hosts-alvo* coletam os dados auditados, de vários *logs* baseados em *host*[35], *logs* de tráfego de rede baseado em *host*. O componente *Agen* converte os dados auditados. O processo *Agend* é responsável por parar e iniciar o *agen* quando instruído pela interface do usuário[36].
O processo *Arpool*, no NIDES *host*, coleta os dados auditados que vieram do *agend* e os fornece para os componentes de análise estatística e de análise baseada em regra[37] sob demanda.
São, na verdade, quatro sistemas diferentes. Cada um construído no topo do seu antecessor:

- O componente *Resolver* é responsável por avaliar e agir sobre os dados recebidos dos módulos de análise agregá-los e fazer uma decisão composta.
- O componente *Archiver* é responsável em armazenar gravações de auditoria e resultados de análise e alertas.
- O componente de análise *Batch* permite ao SSO experimentar novas configurações a partir de dados auditados conhecidos, em paralelo, rodando o NIDES em produção.
- A interface do usuário, no NIDES *host*, é responsável por comunicar violações de segurança suspeitas para o SSO, bem como o status geral de processamento. Somente uma instância da interface pode ser ativada de cada vez.

A funcionalidade do método baseado em comportamento foi alterada para permitir ao NIDES não só trabalhar com distribuição parametrizada simples, mas também, com distribuição multi modal. Por exemplo, um usuário que desempenha duas tarefas completamente diferentes, desenvolvendo um software e escrevendo a documentação para ele. Em outro dia, o usuário está fazendo um relatório do software. Padrões de uso como este exigem um modelo estatístico com diferentes modelos de usuário numa conta.

Três mudanças foram feitas em favor da performance:

[35] Existe uma previsão de utilizar TCP WRAPPER.

[36] É implementado usando RPC.

[37] Baseada em assinatura.

- O componente de análise estatística armazena informações sobre arquivos e diretórios acessados numa lista. Como esta lista cresce muito e para aliviar este problema, o algoritmo de análise foi alterado. O processamento de gravação de auditoria foi movido para um estágio de geração de *profile*.
- Uma característica foi acrescentada para dar, ao usuário, a chance de ter os *profiles* atualizados em tempo real.
- Um *cache* para *profile* de configuração de usuário foi adicionado, para aumentar a velocidade de processamento, no módulo baseado em comportamento.

Outras características foram acrescentadas:

- Estrutura de armazenamento de *profile* otimizada para o armazenamento de *profile* de curto e de longo prazo.
- Análise de configuração do NIDES em tempo real quando está rodando. Para detecção em tempo real e em modo *batch*, deu ao SSO a possibilidade de configurar aspectos da análise de componentes.
- Um relatório de status expandido relata o status e a configuração de cada *host* monitorado. O status e o resumo dos alertas são relatados periodicamente, quando da análise de dados auditados em modo *batch*.
- Facilidade de gerenciamento de dados. Esta facilidade permite o AS alcançar e recuperar os dados auditados e os dados resultantes do processamento.
- Base de regras expandida[38].
- Introdução de um *script perl* para converter o *audit trail* do *host* específico para a forma canônica do NIDES[39], por ser mais disponível a um número de plataformas diferentes.
- Um *agen* monitor em modo promíscuo para detectar *sniffer* na rede.
- Um modelo expandido de *facto* de *audit record*, permite considerar todos os campos disponíveis numa gravação de auditoria para detecção.

[38] De 21 para 39 regras.

[39] Anteriormente escrito em C, limitado e difícil de portar.

NSM – Network Security Monitor

Foi o primeiro sistema a usar tráfego de rede diretamente como origem de dados de auditoria. Ele escuta passivamente todo o tráfego de rede que passa através de uma LAN broadcast a procura de comportamento intrusivo a partir desta entrada. O NSM pode monitorar uma rede de *hosts* heterogêneos sem ter de converter os formatos de *audit trail* para uma forma canônica.

Ele segue uma abordagem em camadas. A camada de conexão é responsável por estudar os dados da rede e tentar formar pares de canais de comunicação bi-direcional entre conjunto de *hosts*. Estas conexões são condensadas em um vetor conexão, podando para fora alguns dos dados vindos das camadas mais baixas. No sistema descrito, somente os vetores *host* e os vetores conexão são usados como entrada para um SE simples, que analisa os dados a procura de comportamento intrusivo. Estes *profiles* consistem de caminhos de dados esperados que descrevem que sistema é esperado comunicar com qual, usando quais protocolos. Outro tipo de *profile* é construído para cada tipo de protocolo de alto nível, por exemplo, o que uma sessão *telnet* normal parece. Outras entradas são conhecidas pelas várias capacidades dos protocolos e de como eles autenticam seus pedidos. Por exemplo, *telnet* é um protocolo poderoso que permite o usuário desempenhar uma variedade de tarefas e autentica seus pedidos. Enquanto o *sendmail* pede identificação, mas não autentica esta identificação. O dado destas origens é combinado para tomar uma decisão sobre a probabilidade que uma conexão particular representa comportamento intrusivo, baseado em reações anômalas. Isto é combinado dentro de um conceito de estado de segurança da conexão.

A apresentação normal dos dados ao Administrador de Sistemas - AS tem a forma de lista classificada, onde cada fileira na lista consiste de um vetor conexão e o nível de suspeita computado. Os resultados são também armazenados numa base de dados, permitindo o AS selecionar eventos específicos para serem estudados mais de perto.

SNORT – LIGHTWEIGHT INTRUSION DETECTION FOR NETWORK

Snort é um IDS baseado em rede, com abordagem em detecção baseada em conhecimento. É leve e pode monitorar pequenas redes TCP/IP. Detecta uma grande variedade de tráfego de rede suspeito e ataques. Snort direciona sua arquitetura na simplicidade, no desempenho e na flexibilidade.

Snort, comparado ao *tcpdump*, está mais voltado para a inspeção dos dados do pacote e tem a vantagem de mostrar a saída de uma maneira mais amigável.

Snort pode utilizar um conjunto de regras flexíveis para desempenhar funções adicionais, tal como procurar e gravar somente aqueles pacotes que tenham seus *flags TCP* sinalizados num modo particular, ou contendo pedidos da *web* com provas de vulnerabilidades CGI.

Existem três subsistemas primários que fazem o Snort:

- Pacote decodificador. A máquina de decodificar é organizada ao redor das camadas da pilha de protocolos presentes, suportada pelo link de dados e definições do protocolo TCP/IP. Cada sub-rotina no decodificador impõe ordem nos dados do pacote, revestindo a estrutura de dados de tráfego de rede “puro”. Estas rotinas são chamadas em ordem através da pilha de protocolos, a partir da camada link de dados, a camada de transporte até a camada de aplicação. A maior funcionalidade do decodificador consiste em configurar ponteiros dentro dos pacotes de dados para análise posterior, pela máquina de detecção.
- Máquina de detecção. O Snort mantém as regras de detecção numa lista de duas dimensões que são denominados Cadeia de cabeçalhos e Cadeia de opções. Estas listas de regras são condensadas para uma lista de atributos comuns na Cadeia de cabeçalhos, com as opções variadas de detecção contidas na Cadeia de Opções. Por exemplo, se quarenta e cinco regras de detecção de provas de CGI-BIN são especificadas em um dado arquivo de biblioteca de detecção do Snort, elas compartilham todas, geralmente, um endereço IP de origem e destino e portas comuns. Para acelerar o processo de detecção, essas são condensadas dentro de uma simples Cadeia de cabeçalhos e então assinaturas de detecção individuais são conservadas em estruturas de Cadeias de Opções. Essas Cadeias de regras são buscadas recursivamente para cada pacote, em ambas as direções. A ação específica

na definição da regra é disparada, assim que a primeira regra atenda o casamento de padrão na máquina de detecção e retorna.

- logging and alerting subsystem. É selecionado na linha de comando, quando em funcionamento. Existem três (3) opções de *logging* e cinco (5) de alerta. As opções de *logging* podem ser colocadas nos pacotes de *log* no seu *decode*, formato de leitura humana para uma estrutura de diretório baseado em IP, ou em formato binário *tcpdump* num arquivo de *log* simples. O formato decodificado de *logging* permite análise rápida dos dados coletados pelo sistema. O formato *tcpdump* é mais rápido para gravar no disco e deve ser usado em instâncias onde alto desempenho é exigido. *Logging* pode ser desligado completamente, deixando as alertas habilitadas para maior desempenho. Alertas podem ser enviadas para *syslog*, colocadas num arquivo texto de alerta em dois formatos diferentes. As alertas do *syslog* são enviadas como mensagens segura/autorização que são facilmente monitoradas com ferramentas tais como *Swatch*[40]. Existem duas opções para enviar as alertas para um arquivo texto plano, alerta completa e rápida. A primeira escreve a mensagem alerta e a informação do cabeçalho do pacote através da camada de transporte. A segunda opção escreve um subconjunto condensado da informação do cabeçalho para o arquivo alerta, permitindo maior desempenho do que no modo completo. Existe uma quinta opção para desabilitar completamente as alertas, que é útil quando as alertas são desnecessárias ou inapropriadas, tais quando testes de penetração na rede estão sendo desempenhados.

Estes subsistemas ficam no topo da biblioteca *libpcap* de *sniffing* de pacotes em modo promíscuo, que fornece um *sniffing* de pacotes portável e capacidade de filtragem. Configuração do programa, análise de regras e geração de estrutura de dados tomam lugar antes da seção de *sniffing* ser inicializada, conservando a quantidade de processamento por pacote a um mínimo requerido para alcançar a funcionalidade do programa base.

Snort é bem empregado na segurança da rede, quando um novo ataque aparece e a ferramenta instalada não liberou a nova assinatura de reconhecimento de ataque.

O desenvolvimento de novas regras de detecção de *exploits* é feito através da simples escrita de tais regras, que são poderosas o suficiente para detectar uma grande variedade de hostilidades ou

[40] Esta ferramenta está disponível em ftp://coast.cs.purdue.edu/tools/unix/chrootuid.

tráfego de rede suspeito. Existem três ações básicas diretivas que o Snort pode usar quando um pacote casa com uma regra de padrão específico:

- passar as regras, simplesmente pulando os pacotes.
- colocar as regras em *log*, escrevendo todo o pacote para uma rotina de *logs* que tenha sido selecionada pelo usuário no *run-time.*
- regras de alertas que geram uma notificação de evento usando um método especificado pelo usuário na linha de comando e então colocar em *log* o pacote todo, usando mecanismo de *log* para permitir análise futura.

O método geral para desenvolvimento consiste de obter o *exploit* de interesse, tal como um novo *buffer overflow*, rodar o *exploit* em uma rede teste com o Snort gravando todos o tráfego entre o alvo e o *host* atacado e então analisar os dados para uma assinatura única e condensar esta assinatura em uma regra.

Snort tem capacidade de alerta em tempo real, sendo enviada para o *syslog*. Pode prover administradores com dados suficientes para tomar decisão das ações em face das atividades suspeitas.

W&S – Wisdom & Sense

É o único na sua abordagem para detecção de intrusão baseada em comportamento, ele estuda o histórico de dados auditados para produzir várias regras que descrevem o comportamento normal, formando o "*Wisdom*". Estas regras são então alimentadas para um Sistema Especialista que avalia dados auditados recentes, por violações das regras e alerta o administrador de sistemas, quando as regras indicam desvio de comportamento, formando o "*sense*". W&S lê as gravações de auditoria históricas a partir de um arquivo. O elemento "*sense*" lê gravações de auditoria, avalia contra uma base de regras e dispara um alarme se o número suficientemente de regras relatam uma discrepância bastante alta com o *profile*. Ele relata uma anomalia se a contagem excede um operador entrada pré-definido. O elemento *sense* então lê a base de regra, dicionário e novas gravações de auditoria, tanto em batch ou na medida que se tornam disponíveis. A máquina de inferência processa cada gravação de auditoria, achando as regras que aplicam e computa suas transações. Fazendo isto a máquina de inferência basicamente computa todas as contribuições a partir das diferentes regras falhas, tendo em mente que se procura por anomalias, e as regras que descrevem comportamento normal fazem parte das gravações de auditoria.

Apêndice B

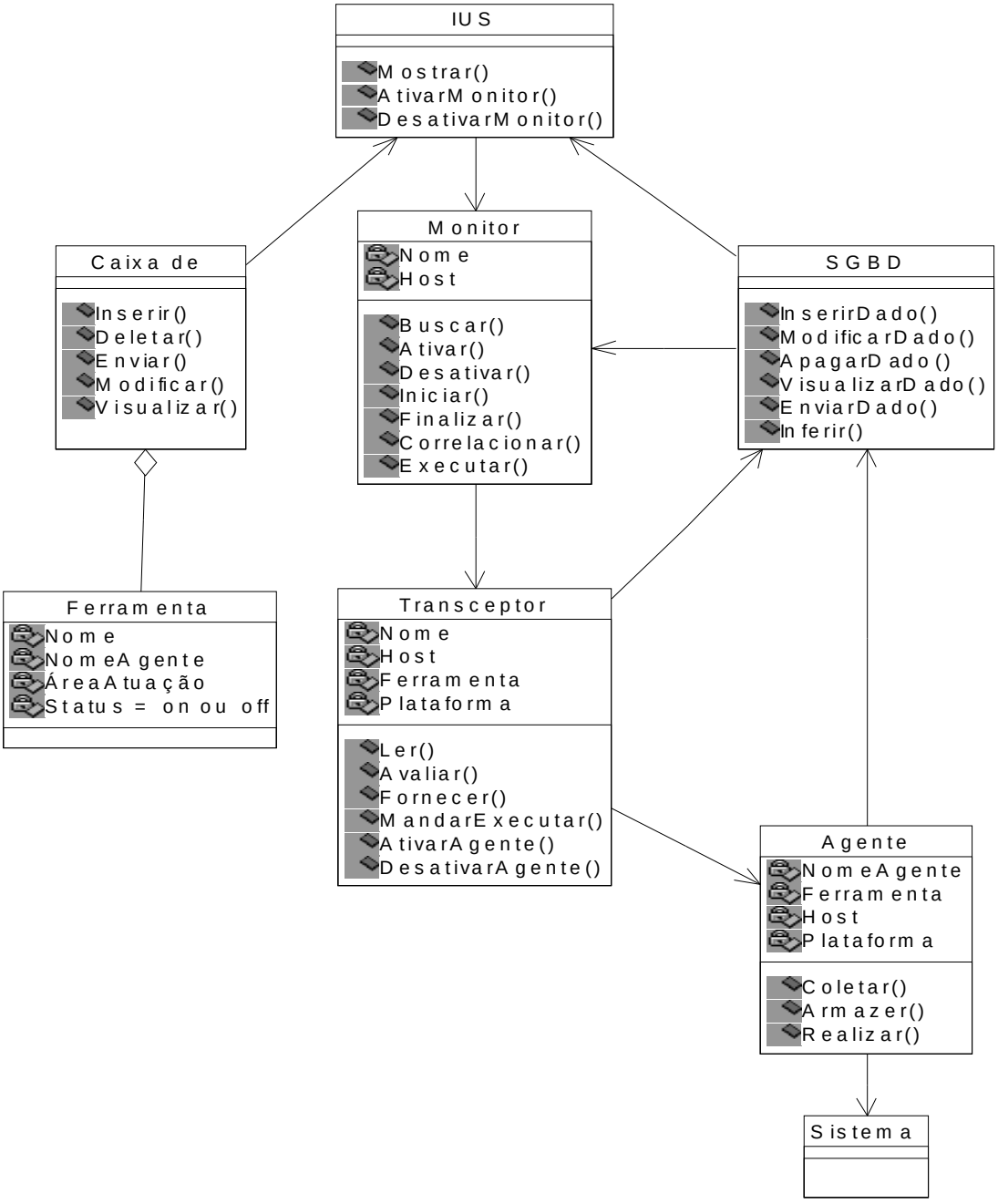

Figura : Diagrama de classes (UML)