UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

FERNANDO AUGUSTO PESTANA JÚNIOR

PROPOSTA DE ATUALIZAÇÃO AUTOMÁTICA DOS SISTEMAS DE DETECÇÃO DE INTRUSÃO POR MEIO DE *WEB SERVICES*

São Luís
2005

FERNANDO AUGUSTO PESTANA JÚNIOR

PROPOSTA DE ATUALIZAÇÃO AUTOMÁTICA DOS SISTEMAS DE DETECÇÃO DE INTRUSÃO POR MEIO DE *WEB SERVICES*

Dissertação de Mestrado submetida à Coordenação do Curso de Pós-graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como parte dos requisitos para obtenção do título de Mestre em Engenharia de Eletricidade, Área de Concentração: Ciência da Computação.

Orientador: Prof. Dr. Zair Abdelouahab

São Luís
2005

FERNANDO AUGUSTO PESTANA JÚNIOR

PROPOSTA DE ATUALIZAÇÃO AUTOMÁTICA DOS SISTEMAS DE DETECÇÃO DE INTRUSÃO POR MEIO DE *WEB SERVICES*

Dissertação de Mestrado submetida à Coordenação do Curso de Pós-graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como parte dos requisitos para obtenção do título de Mestre em Engenharia de Eletricidade, Área de Concentração: Ciência da Computação.

Aprovada em: ___ / ___ / _____

BANCA EXAMINADORA

_____

Prof. Dr. Zair Abdelouahab
Universidade Federal do Maranhão
(Orientador)

_____

Prof. Dr. Aristófanes Corrêa Silva
Universidade Federal do Maranhão

_____

Prof. Dr. Antônio Jorge Gomes Abelém
Universidade Federal do Pará

A minha amada esposa Andréa.
Aos meus pais Fernando e Maria da Graça.
Aos meus sogros Luis e Celeste.
A minha irmã Adriana e a minha sobrinha Letícia.

AGRADECIMENTOS

A minha grande e estimada família pela compreensão nas ausências em nossos almoços de domingo;

Aos meus colegas de projeto Christiane, Rômulo, Glenda e Alfredo pela contribuição inestimável para o desenvolvimento deste trabalho;

Ao Prof. Edson Nascimento pela seu incentivo e apoio que me deram as condições necessárias para realizar este trabalho;

Ao meu orientador Prof. Zair pela confiança depositada em mim;

Aos meus grandes companheiros do mestrado, Antônio e Anderson, pelas incansáveis horas de estudo;

Ao Núcleo de Tecnologia da Informação e à Biblioteca Central da UFMA pela compreensão e estímulo que me foram dados durante a elaboração deste trabalho.

RESUMO

A segurança das redes de computadores é imprescindível para seus administradores. Os Sistemas de Detecção de Intrusão (SDIs) podem torná-las mais seguras, entretanto é imperativo que esses sistemas estejam em constante atualização para desempenhar sua função de forma satisfatória.

Esta dissertação propõe um modelo de compartilhamento de informações entre Grupos de Resposta a Incidentes de Segurança em Computadores, geralmente conhecidos como CSIRTs (do inglês "Computer Security Incident Response Team") e SDIs objetivando a atualização automática do conjunto de ações de resposta a intrusões dos SDIs. Essa atualização será feita com base nas medidas restritivas de curto prazo sugeridas nos alertas de segurança emitidos pelos CSIRTs. Esse modelo é baseado nas tecnologias de *Web services* e da *Extensible Markup Language* (XML).

Também é proposta uma arquitetura multiagente, baseada nas técnicas de recuperação e filtragem de informação, que tem como objetivo manter o mecanismo de detecção de ataques dos SDIs atualizados de forma automática.

Palavras-chave: Segurança de Redes, Detecção de Intrusos, *Web services*, Recuperação e Filtragem de Informação

ABSTRACT

The security of computer networks is indispensable for its administrators. Intrusion Detection Systems (IDSs) can increase their security but is very important to update them constantly.

This research work proposes a model for sharing information between Computer Security Incident Response Teams (CSIRTs) and IDSs, aiming the achievement of an automatic update of the IDSs response actions data base, based on restrictive short-term measures suggested in security alerts issued by CSIRTs. This model is based on Web services and Extensible Markup Language (XML) technologies.

It is also presented a multiagent architecture based on information retrieval and filtering techniques, designed to automatically maintain the IDSs attacks signatures mechanism up-to-date.

Keywords: Network Security, Intrusion Detection, Web services, Information Retrieval and Filtering.

LISTA DE APÊNDICES

LISTA DE FIGURAS

LISTA DE SIGLAS

AAFID	*Autonomous Agents for Intrusion Detection*
ACS	Agência Central de Segurança
AID	*Adaptive Intrusion Detection*
ANS	Agente Servidor de Nomes (*Agent Name Server*)
API	*Application Programming Interface*
BA	Agente BAM
BAM	*Binary Association Memory*
CAP	*Common Alerting Protocol*
CFP	Chamada para Propostas (C*all for Proposals*)
CIDF	*Common Intrusion Detection Framework*
CSIRT	Grupo de Resposta a Incidentes de Segurança em Computadores (*Computer Security Incident Response Team*)
CSM	*Cooperating Security Managers*
CVE	*Common Vulnerabilities and Exposures*
CVS	*Concurrent Versions System*
DARPA	*Defense Advanced Research Projects Agency*
DFDB	Base de Dados de Incidentes de Intrusão e Informação Forense
DNS	Arquivo padrão do Servidor de Nomes (*Default Name Server file*)
DoS	Ataques de Negação de Serviço (*Denial of Service*)
EISPP	*European Information Security Promotion Programme*
FTP	*File Transfer Protocol*
GMT	*Greenwich Mean Time*
HNA	Agente Honey Net

HTML	*HyperText Markup Language*
HTTP	*Hypertext Transport Protocol*
IANA	*Internet Assigned Numbers Authority*
IDA	Intrusion Detection Agent
IDMEF	*Intrusion Detection Message Exchange Format*
IETF	*Internet Engineering Task Force*
IIDB	Base de Dados de Padrões de Intrusos e Intrusões
IODEF	*Incident Object Description and Exchange Format*
ISO	*International Organization for Standardization*
LSASS	Serviço do Subsistema de Autoridade de Segurança Local do *Microsoft Windows*
LSIA	Agente de Segurança Local (*Local Security Intelligent Agent*)
MCA	Agente Controlador Principal
MLP	*Multilayer Perceptron*
NIDIA	*Network Intrusion Detection System based on Intelligent Agents*
OKDB	Base de Dados Otimizada de Palavras-chave
RADB	Base de Dados de Ações de Respostas
RFC	*Request For Comments*
SAA	Agente de Avaliação de Severidade
SAARA	Sociedade Atualizada de Agentes de Reconhecimento de Assinaturas
SCA	Agente Controlador de Ações (*System Controller Agent*)
SDI	Sistema de Detecção de Intrusão
SEA	Agente de Avaliação de Segurança do Sistema (*System Security Evaluation Agent*)

SIA	Agente de Integridade do Sistema (*Self-Integrity Agent*)
SMA	Agente de Monitoramento do Sistema (*System Monitoring Agent*)
SOA	Agente Sistema Operacional
SOAP	*Simple Object Access Protocol*
STDB	Base de Dados de Estratégias
SUA	Agente de Atualização do Sistema (*System Update Agent*)
TCP/IP	*Transmission Control Protocol/Internet Protocol*
UDP	*User Datagram Protocol*
UML	*Unified Modeling Language*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
UTC	*Universal Time Coordinated*
XML	*Extensible Markup Language*

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Um aspecto importante da conectividade entre as redes de computadores proporcionada pela Internet é que a segurança de uma rede está diretamente relacionada com a segurança de todas as outras redes.

Os problemas relativos à segurança das redes de computadores são cada vez maiores, diariamente temos notícias de novas vulnerabilidades nos sistemas existentes e de ataques bem sucedidos contra diversos *sites*. No ano de 2004 foram feitas 3.780 (três mil setecentos e oitenta) notificações de vulnerabilidades ao CERT®/CC[1] e no primeiro semestre de 2005 foram 2.874 notificações [16].

Ataques de “vírus”, “cavalos de tróia” e *worms* disseminam-se rapidamente por todo o mundo, como o *worm sasser* que se espalhou rapidamente pela Internet explorando uma vulnerabilidade do serviço do subsistema de autoridade de segurança local (LSASS) do *Microsoft Windows,* atingindo escala mundial em pouco tempo.

Ataques de Negação de Serviço, mais conhecidos como “*Denial of Service* (DoS)” impedem o uso das redes de computadores, sistemas ou aplicações consumindo todos os seus recursos como por exemplo CPU, memória, largura de banda e espaço em disco. Ataques DoS contra *sites Web* bastante conhecidos, que tiveram suas operações interrompidas tem sido freqüentemente reportados pela mídia desde o final da última década.

Outros exemplos de ataques DoS são:

- Envio de pacotes TCP/IP mal formados para um determinado servidor de forma a causar travamento do seu sistema operacional;

[1] O CERT®/CC foi o primeiro grupo de resposta a incidentes de segurança em computadores e é o principal centro para envio de notificações de incidentes e obtenção de informações sobre problemas de segurança da Internet.

- Enviar requisições ilegais para um sistema com o objetivo de interferir no seu funcionamento normal;
- Estabelecer muitas sessões com um servidor impedindo que outros usuários o utilizem.

Os Sistemas de Detecção de Intrusão (SDIs) são utilizados como parte da solução para o problema de segurança das redes de computadores. Segundo BACE & MELL [5], eles são compostos por *software* ou *hardware* que automatizam o processo de monitoramento de eventos que ocorrem em um sistema computacional ou rede, com os objetivos de identificar ameaças e responder a essas ações intrusivas.

Entretanto para o SDI proteger um sistema com maior segurança, ele deve estar em constante atualização, pois freqüentemente são encontradas novas vulnerabilidades nos sistemas existentes ou novas técnicas de ataque são desenvolvidas [28].

1.1 Definição do Problema

Os incidentes de segurança em computadores apresentam taxas crescentes apoiados pelo desenvolvimento, disseminação e utilização de ferramentas automatizadas de ataques, facilmente encontradas com auxílio de qualquer mecanismo de busca da Internet [1].

Atualmente as vulnerabilidades são exploradas a uma velocidade muito maior o que significa que devemos estar mais alertas do que nunca, não só agindo, como também prevenindo possíveis riscos e problemas de segurança.

Para que um SDI possa detectar e responder a novas formas de ataques é necessário que o tempo entre a descoberta de novas vulnerabilidades e a atualização dos seus mecanismos de detecção e de resposta seja o menor possível.

Entretanto ainda não existe um padrão definido para representar e trocar dados relativos à segurança, impedindo que eles sejam processados de forma automática e requerendo a intervenção humana.

Atualmente páginas HTML, correio eletrônico, telefone, fax e conferências são os meios utilizados para compartilhamento de informações sobre incidentes de segurança.

Os Grupos de Resposta a Incidentes de Segurança em Computadores (CSIRTs - do inglês "*Computer Security Incident Response Team*") têm como missão receber, analisar e responder a incidentes de segurança em computadores.

Ao se reportar um incidente de segurança a um grupo de resposta a incidentes, ele terá maior capacidade para analisá-lo e correlacioná-lo a outras notificações enviadas, tendo como objetivo detectar novas ameaças. Em decorrência desse trabalho, soluções poderão ser encontradas e divulgadas mais rapidamente, permitindo que ações preventivas e de resposta a essas ameaças sejam tomadas.

Devido à inexistência de um padrão definido para representar e trocar dados relativos à segurança e tampouco de como o compartilhamento dessas informações será feito, apresenta-se neste trabalho uma proposta para o compartilhamento automático de informações entre os SDIs e os CSIRTs, assim como uma proposta de arquitetura multiagente para a atualização automática do mecanismo de detecção de ataques dos SDIs multiagente.

1.2 Objetivos Geral e Específico

Neste trabalho é proposto um modelo de compartilhamento de informações relativas a segurança entre SDIs e CSIRTs. Essa proposta tem como objetivo atualizar de forma automática o conjunto de ações de resposta a intrusões dos SDIs, com base nas medidas restritivas de curto prazo sugeridas nos alertas de segurança emitidos pelos CSIRTs. Esse modelo é baseado nas tecnologias de sociedade de agentes inteligentes, *Web services* [12] e da *Extensible Markup Language* - XML [14].

Para que um SDI possa tomar alguma ação de resposta, o seu mecanismo de detecção de ataques também deve estar atualizado e para este fim também é proposto neste trabalho uma arquitetura multiagente que utiliza técnicas do campo da recuperação e filtragem de informação e *Web services*.

Este trabalho tem como objetivos específicos:

- Apresentar uma arquitetura para o compartilhamento de informações entre SDIs e CSIRTs, assim como uma proposta de aplicação para essa arquitetura;
- Apresentar uma proposta de formato de dados para mensagens de alertas divulgadas pelos CSIRTs utilizando-se a XML, tendo como objetivo atualizar de forma automática o conjunto de ações de resposta a intrusões dos SDIs;
- Apresentar uma arquitetura multiagente para atualização do mecanismo de detecção de ataques dos SDIs;
- Apresentar um protótipo do modelo apresentado para o compartilhamento de informações entre SDIs e CSIRTs, por meio de uma sociedade de agentes inteligentes, *Web services* e da *XML*.

1.3 Organização do Trabalho

Este trabalho encontra-se organizado em sete capítulos. No primeiro capítulo são apresentados os aspectos da conectividade mundial das redes de computadores, possíveis por meio da Internet, suas conseqüências para a segurança dessas redes e uma sugestão para a prevenção e combate às novas ameaças desse mundo conectado.

No Capítulo 2 apresentam-se informações sobre segurança de redes, formatos propostos para compartilhamento de informações de segurança e as tecnologias que serão utilizadas para o desenvolvimento deste trabalho.

O Capítulo 3 é dedicado a uma proposta de arquitetura para incorporar o compartilhamento de informações aos atuais Sistemas de Detecção de Intrusão e a uma proposta de formato de dados para os alertas de segurança emitidos pelos CSIRTs.

O Capítulo 4 apresenta uma arquitetura multiagente para a atualização automática de mecanismo de detecção de ataques dos SDIs multiagente.

O Capítulo 5 apresenta um protótipo da proposta apresentada para a atualização do conjunto de ações de resposta dos SDIs.

O Capítulo 6 contém resultados das simulações com o protótipo apresentado no Capítulo 5.

E finalmente, o Capítulo 7 apresenta as conclusões, contribuições deste trabalho e sugestões para trabalhos futuros.

2. REFERENCIAL TEÓRICO

Neste capítulo apresentam-se alguns conceitos importantes sobre como lidar com a segurança de redes, projetos relacionados aos SDIs e troca de mensagens de segurança, assim como tecnologias que permitem o compartilhamento de informações por meio da Internet.

2.1 *O Common Intrusion Detection Framework* (CIDF)

O CIDF [19] é um projeto da *Defense Advanced Research Projects Agency* (DARPA) e constituí-se em um esforço para desenvolver protocolos e *Application Programming Interfaces* (APIs) de forma que projetos de pesquisa de detecção de intrusão possam compartilhar informações e recursos. O CIDF também tem como objetivo possibilitar a reutilização dos componentes de detecção de intrusão em outros sistemas.

Um modelo que didaticamente representa o funcionamento de um SDI, mostrando o fluxo de informações e as funcionalidades básicas do CIDF é mostrado na Figura 2.1 [19].

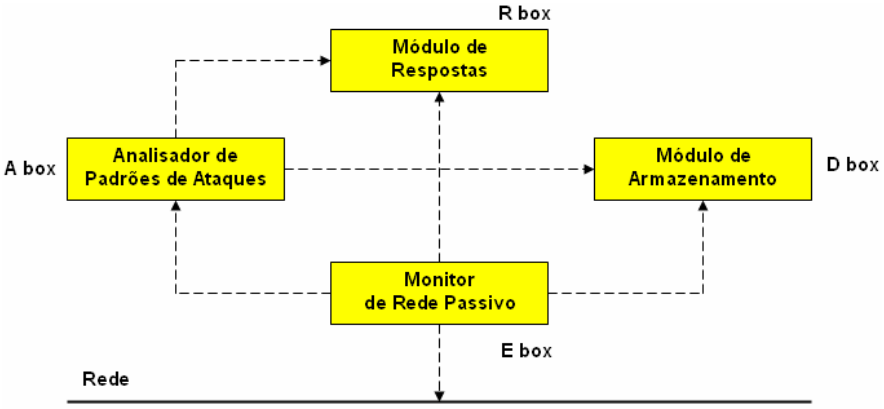


Figura – 2.1. CIDF com capturador de pacotes na rede.

O CIDF é composto por quatro tipos de componentes:

- *E boxes* (*event generators* - geradores de eventos) – são os sensores responsáveis por captar a ocorrência de eventos e fornecer informações sobre os mesmos para o resto do sistema em um formato padrão. Podem ser considerados eventos a presença de um pacote transitando na rede, o registro de alguma atividade no *log*, etc.;
- *A boxes (analysers* - analisadores) – recebem eventos dos outros componentes, os analisam e retornam novos eventos que presumivelmente representam algum tipo de sumário dos eventos analisados. O analisador pode ser uma ferramenta baseada em assinatura que busca por determinados padrões em uma seqüência de eventos;
- *D boxes* (*database components* - mecanismos de armazenamento) – responsáveis por manter informações provenientes dos geradores de eventos e dos mecanismos de análise para que possam ser apreciadas pelos operadores do sistema no futuro, como por exemplo, dados de um atacante reincidente;
- *R boxes (response boxes* - módulo de respostas) – caso um problema de segurança seja detectado, o módulo de respostas é responsável por tomar alguma contramedida. Por exemplo, interromper uma conexão TCP (*Transmission Control Protocol)*.

2.2 O Projeto NIDIA

O *Network Intrusion Detection System based on Intelligent Agents* (NIDIA) [27] constitui-se na proposta de um SDI multiagente [47] que está sendo desenvolvido na Universidade Federal do Maranhão, capaz de gerar um índice de

suspeita de ataque a partir da análise de dados coletados de *logs* de máquinas e de datagramas IP.

A escolha da arquitetura multiagente para um SDI visa obter as seguintes vantagens enumeradas por BALASUBRAMANIYAN [7] e CROSBIE & SPAFFORD [21]:

- Facilidade de manutenção e atualização do sistema com adição e remoção de agentes, mantendo o máximo de disponibilidade do mesmo;
- Possibilidade de atualização dos agentes responsáveis pelo mecanismo de identificação de ataques;
- Capacidade de se obter maior adequação a determinado ambiente, utilizando-se agentes especializados para um computador ou rede em particular;
- Maior tolerância a falhas que sistemas monolíticos que possuem um único ponto crítico de falhas;
- Alto potencial de escalabilidade pela adição de novos agentes para manter a performance exigida em sistemas em expansão.

A arquitetura apresentada para o NIDIA é inspirada no modelo lógico do CIDF, possuindo para este fim agentes com função de geradores de eventos (agentes sensores), mecanismos de análise dos dados (agentes de monitoramento e de avaliação de segurança), mecanismos de armazenamento e um módulo para realização de ações de resposta (agente controlador de ações).

Além disso, existem agentes responsáveis pela integridade do sistema (Agente de Integridade do Sistema), pela coordenação das atividades do SDI como

um todo (Agente de Segurança Local) e pela atualização das bases de conhecimento (Agente de Atualização do Sistema).

O NIDIA e o seu funcionamento são explicados em maiores detalhes no capítulo 5.

2.3 Sistemas de Detecção de Intrusão Multiagente

Nesta seção é feita a análise de alguns SDIs baseados em agentes. São apresentadas as principais características e principalmente o tipo de respostas apresentadas.

2.3.1 *Autonomous Agents for Intrusion Detection* (AAFID)

Um sistema AAFID [44] pode ser distribuído por muitas máquinas em uma rede. Cada Máquina pode conter qualquer número de agentes que monitoram os eventos na máquina. Os agentes podem utilizar filtros para obter dados de uma maneira independente do sistema. Todos os agentes em uma máquina enviam as informações capturadas para um único *transceiver. Transceivers* são entidades que monitoram a operação de todos os agentes em uma máquina. Eles possuem a habilidade de iniciar, parar e enviar comandos de configuração aos agentes. Eles também podem realizar a redução dos dados recebidos dos agentes. Os *transceivers* reportam seus resultados a um ou mais *monitors*. Cada monitor controla a operação de vários *transceivers* e tem acesso a dados de toda a rede, o que os torna capaz de detectar intrusões que envolvam várias máquinas. Os monitores podem ser organizados de forma hierárquica e um *transceiver* pode se reportar a mais de um *monitor* provendo redundância e tolerância à falha ao sistema. O

sistema gera apenas relatórios e não emprega nenhum tipo de reação aos problemas detectados.

2.3.2 *Cooperating Security Managers* (CSM)

O CSM [48] é um sistema de detecção baseado em rede e em máquina, utilizando arquitetura de agentes. O *Command Monitor* captura comandos de usuários e os envia para o *Local IDS*. O *Local IDS* é um sistema de detecção baseado em máquina que trata de intrusões no sistema. Os dados vindos da rede são enviados para o *Security Manager* que examina os dados e informa qual segmento de rede deve ser vigiado. Se uma intrusão for detectada uma mensagem de alerta é gerada e se o administrador tiver habilitado ações de resposta adicionais, elas também serão executadas pelo *Intruder Handler*. O componente final é a interface gráfica que permite aos administradores de segurança comunicarem-se com o CSM para monitorar o estado de segurança do sistema computacional e configurar as diversas opções oferecidas pelo CSM.

2.3.3 *Adaptive Intrusion Detection* (AID)

O AID [43] é uma arquitetura cliente-servidor para sistemas Unix, constituída por agentes instalados em servidores e uma estação de monitoramento central. O monitor central fica localizado na estação, ligado a um sistema especialista, e os agentes localizados nas máquinas para fazer o monitoramento e a coleta de dados. Os agentes transformam os dados para um formato padrão e os enviam para o monitor central que analisa as informações e identifica o ataque baseado na análise de assinaturas. Disponibiliza uma interface gráfica para que o

administrador do sistema tenha acesso às informações do ataque e gera relatórios dos incidentes ocorridos.

2.3.4 Hummingbird

O Hummingbird [24] é um sistema distribuído para detecção de intrusão por anomalia. Ele emprega um conjunto de agentes denominados *Hummer* que podem ser associados a uma única máquina ou a um conjunto de máquinas. Cada *Hummer* interage com outros agentes no sistema por meio de relacionamentos *manager, subordinate* e *peer. Managers* podem transmitir comandos aos subordinados; tais comandos incluem informações para iniciar/parar a captura de dados e iniciar/encerrar o encaminhamento de dados. *Peers* podem enviar requisições para a encaminhar/obter/receber dados de outros *peers. Subordinates* também podem enviar requisições aos *Managers.* Uma interface gráfica permite ao administrador iniciar e parar um conjunto de ferramentas de captura de dados nas máquinas ou na rede. Essas ferramentas permitem coletar importantes informações para a detecção e/ou prevenção de ataques específicos de rede ou para elevar o nível de auditoria quando há suspeita de um ataque em andamento.

2.3.5 Intrusion Detection Agent (IDA)

O IDA [3] é baseado em máquina e faz detecção por abuso. Procura por evidências de intrusões utilizando quatro agentes da arquitetura para a investigação. O *manager* usado para determinar se existe alguma intrusão no sistema. Os *sensors* responsáveis pelo monitoramento local do sistema; procuram por evidencias de ataque e avisam o *manager.* O *trancing agent* que é um agente móvel que tenta descobrir a origem do ataque. O *information gatherers* é o agente acionado pelo

trancing agent para coletar informações no sistema da máquina. O seu sistema de resposta de intrusão gera relatórios e faz investigação automática para a descoberta da origem do ataque.

A Tabela 1.1 apresenta uma análise comparativa relacionada às características relevantes dos SDIs apresentados [41].

Tabela - 1.1. Comparativo entre SDIs multiagente.

Característica / Protótipo	Método de Análise		Tipo de Análise		Tempo de Coleta		Tipo de Resposta		
	Abuso	Anomalia	Rede	Host	Real	Batch	Relatório/ Alarmes	Manual	Automática
AAFID	x		x	x	x		x		
CSM		x	x	x	x				x
AID	x			x	x		x		
Hummingbird	x	x		x	x		x	x	
IDA	x			x	x		x		x

Dentre os SDIs apresentados, apenas o CSM e o IDA apresentam respostas automáticas. O CSM depende da configuração de ações de respostas de forma manual pelo administrador do sistema que incluem encerrar a sessão do usuário e o bloqueio de sua conta de acesso ao sistema.

O IDA apresenta como ação de resposta a uma invasão a investigação automática da origem da ameaça.

A tabela 1.2 apresenta outro comparativo dos SDIs multiagente, referente às suas características de atualização das assinaturas de ataques e das ações de respostas, onde observa-se que nenhum dos sistemas apresentam atualizações automáticas.

Tabela - 1.2. Atualização dos SDIs multiagente.

Característica / Protótipo	Atualização das Assinaturas de Ataques		Atualização das Ações de Respostas	
	Manual	Auto	Manual	Automática
AAFID	x		x	
CSM	x		x	
AID	x		x	
Hummingbird	x		x	
IDA	x		x	

2.4 Formatos de Dados para Troca de Mensagens de Segurança

A falta de um padrão para representar e trocar dados de segurança é um grave empecilho para o compartilhamento dessas informações. Nesta seção são apresentados três formatos de dados para troca de informações de segurança dos computadores e um formato geral para todo tipo de alertas de emergência.

2.4.1 O *Intrusion Detection Message Exchange Format* (IDMEF)

O propósito do IDMEF [46] é definir formatos de dados e procedimentos para o compartilhamento de informações entre sistemas de detecção de intrusão e respostas, e sistemas de gerenciamento que provavelmente necessitem interagir com eles.

Por meio do IDMEF pode-se obter uma base de dados que contenha dados advindos de diferentes SDIs, tornando possível o acompanhamento de incidentes que ocorram entre vários domínios. Isso também facilitaria a tarefa de correlacionar alertas de segurança em busca de padrões de ataques. Os dados no formato IDMEF também podem ser utilizados para enviar informações de incidentes aos CSIRTs.

O IDMEF pode ser utilizado para resolver muitos problemas associados à representação dos dados de um alerta de detecção de intrusão. Dentre eles podemos citar:

- Heterogeneidade das informações dos alertas. Representada pelas diferentes necessidades ao se comunicar um alerta;
- Ambientes e técnicas diferentes para detecção de intrusão;
- Diferentes analisadores de padrões de ataques poderão gerar alertas com diversos níveis de complexidade.

A proposta de implementação do IDMEF utiliza a XML que permite o desenvolvimento de uma linguagem capaz de descrever alertas de detecção de intrusão. A XML também define um padrão para estender essa linguagem.

A Figura 2.2 mostra possíveis utilizações do IDMEF. Ele pode ser implementado no canal de comunicação entre os sensores e unidade de gerenciamento do SDI que recebe os alertas gerados pelos sensores. Também é possível que esse padrão permita a interoperabilidade entre diferentes SDIs. E finalmente ele pode ser utilizado para a geração de notificações de incidentes a serem enviados aos CSIRTs.

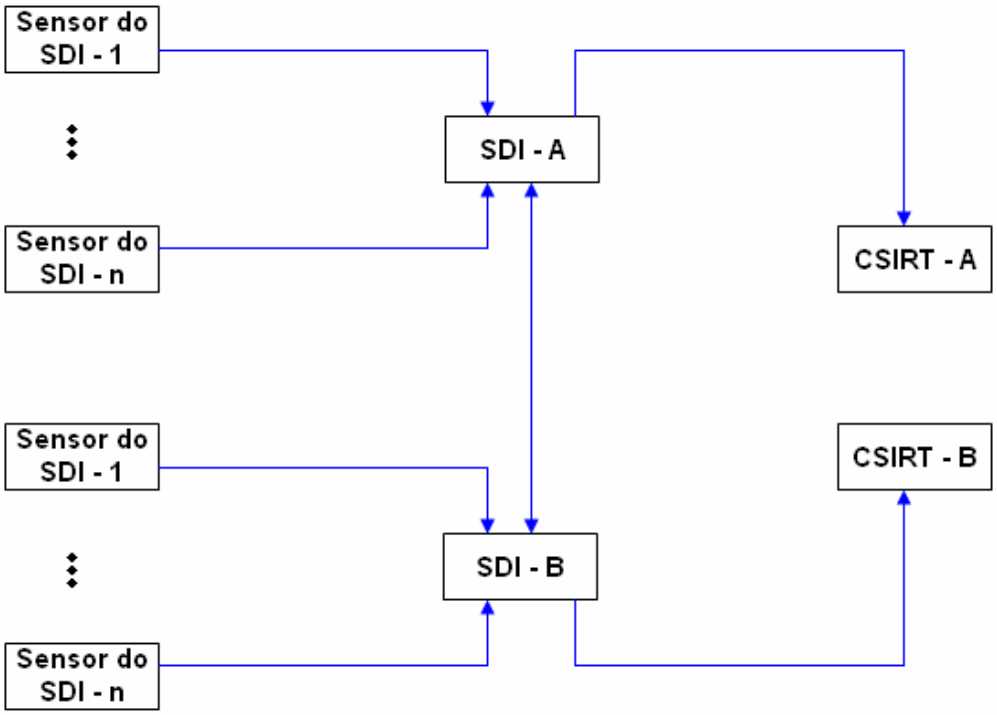


Figura – 2.2. Possíveis utilizações do IDMEF.

O IDMEF ainda é um trabalho em andamento.

2.4.2 O *Incident Object Description and Exchange Format* (IODEF)

A intenção do IODEF [45] é a de definir uma representação de dados que provê um *framework* para o compartilhamento de informações, referentes a incidentes de segurança, comumente trocadas entre grupos de resposta a incidentes de segurança em computadores, como mostra a Figura 2.3.

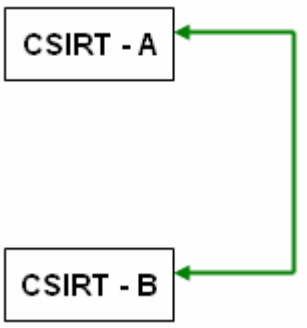


Figura – 2.3. Utilização do IODEF.

O IODEF deve ampliar e melhorar a capacidade operacional dos CSIRTs. A adoção pela comunidade do IODEF poderá fornecer uma maior habilidade para resolver incidentes de segurança por meio da colaboração simplificada e do compartilhamento de dados. O formato estruturado estipulado pelo IODEF permite:

- Uma automação crescente no processamento de dados de incidentes de segurança;
- Menor esforço em normalizar dados similares procedentes de diferentes fontes;
- Um formato comum a partir do qual poderão ser construídas ferramentas interoperáveis para a manipulação de incidentes de segurança.

O IODEF e o IDMEF são formatos complementares. O IDMEF representa dados gerados por um SDI e esses dados são geralmente utilizados por um CSIRT como a base para um relatório de incidente que será representado por meio do IODEF.

Assim como o IDMEF, o IODEF ainda é um trabalho em andamento e também tem proposta de implementação utilizando-se a XML.

2.4.3 O *Common Advisory Format Description*

O *Common Advisory Format Description* [13] é um formato para alertas de segurança que são publicados pelos CSIRTs e que tem como objetivo fornecer aos administradores de sistemas informações atualizadas sobre novas vulnerabilidades e o que pode ser feito para corrigi-las.

Esse formato é proposto pelo *European Information Security Promotion Programme* (EISPP) no intuito de impulsionar a expansão do comércio eletrônico por

parte das pequenas e médias empresas, fornecendo-lhes serviços de segurança da informação em que provavelmente a emissão dos alertas de segurança seja o mais importante.

Para a especificação desse formato, o EISPP parte da premissa de que sistemas de tecnologia da informação podem ser mantidos seguros se eles forem regularmente atualizados ou corrigidos e para isso os seus administradores devem receber automaticamente informações precisas de como realizar essas tarefas de forma adequada.

O *Common Advisory Format Description* foi projetado para ser consumido por administradores e não por sistemas automatizados, embora seja definido por meio da XML.

2.4.4 O *Common Alerting Protocol* (CAP)

O CAP [34] é um formato simples e abrangente para a troca de todo tipo de alertas de emergência. Esses alertas de emergência têm por finalidade informar ao público sobre possíveis perigos ou ameaças que possam causar algum tipo de dano. O CAP também pode ser utilizado sobre todos os tipos de redes de comunicação, inclusive TCP/IP (*Transmission Control Protocol/Internet Protocol)*.

Outra característica do CAP é facilitar a detecção de padrões emergentes nos diversos alertas emitidos ao público que podem indicar uma ameaça não detectada ou atos hostis.

O CAP é um formato aberto e não proprietário de mensagens digitais para todos os tipos de alertas e notificações, sendo compatível com *Web services* e com suporte a criptografia e assinatura digital.

A função principal das mensagens de alerta no formato CAP é prover uma única mensagem que tenha a capacidade de ativar todos os tipos de sistemas de alerta. Uma função secundária dessas mensagens é padronizar os alertas emitidos por diversas fontes de tal forma que eles possam ser agregados e comparados auxiliando na detecção de padrões.

2.5 Tópicos de Segurança

Esta seção contém conceitos acerca da segurança das redes que são importantes para a compreensão dos incidentes de segurança e de como eles devem ser tratados.

2.5.1 Políticas de segurança

Uma política de segurança atribui direitos e responsabilidades às pessoas que lidam com os recursos computacionais de uma instituição e com as informações neles armazenados. Ela também define as atribuições de cada um em relação à segurança dos recursos com os quais trabalham.

Uma política de segurança também deve prever o que pode ou não ser feito na rede da instituição e o que será considerado inaceitável. Tudo o que descumprir a política de segurança é considerado um incidente de segurança.

Na política de segurança também são definidas as penalidades às quais estão sujeitos aqueles que não a cumprem. Infelizmente, segundo o CERT®/CC, muitas instituições apenas desenvolvem uma política de segurança após terem sido alvo de um ataque bem sucedido.

2.5.2 Notificações de incidentes de segurança

Existem muitas definições para um incidente de segurança e para os objetivos deste trabalho adotaremos a definição do CERT®/CC, onde cada organização deve definir o que é um incidente de segurança em computadores, em relação ao seu *site* e de acordo com a sua política de segurança.

Exemplos de incidentes incluem atividades como:

- Tentativas (com ou sem sucesso) de acesso não autorizado a sistemas ou a seus dados;
- Interrupção indesejada ou negação de serviço;
- Uso não autorizado de um sistema para processamento ou armazenamento de dados;
- Modificações nas características de *hardware*, *firmware* ou *software* de um sistema, sem o conhecimento, instruções ou consentimento prévio do responsável pelo sistema.

Para que uma notificação de incidente seja feita é necessário consultar a política de segurança local para determinar a quem ela deve ser enviada. Entre as opções mais comuns encontram-se o coordenador de segurança do *site*, o CSIRT privativo da organização, o CERT®/CC, outros *sites* envolvidos no incidente e órgãos de segurança pública.

O fato de notificar uma tentativa de ataque é uma ação benéfica para todos os envolvidos, porque pode permitir ao administrador da rede atacante identificar se seu ambiente foi também comprometido e estava sendo usado para atacar terceiros [38].

Uma notificação de incidente deve conter informações suficientes para que o CSIRT que a receba seja capaz de entendê-la e respondê-la. Basicamente, ela deve conter:

- Informações de contato de quem a emitiu;
- Informações das máquinas envolvidas no incidente (tanto as que foram utilizadas como fonte quanto aquelas que foram alvo dos ataques);
- Uma descrição das atividades dos invasores, informações dos logs que mostrem as atividades dos invasores, fuso horário relativo ao GMT (*Greenwich Mean Time*) ou UTC (*Universal Time Coordinated*);
- Quais medidas foram tomadas;
- Qual a intenção dessa notificação, se apenas para informação ou se uma determinada ação por parte do CSIRT é desejada.

As notificações de incidentes devem ser feitas assim que um incidente seja detectado e as informações relevantes estejam disponíveis. Para que isto aconteça, os administradores devem monitorar sua rede continuamente em busca de sinais que indicam alguma atividade suspeita, recolher e analisar informações de *logs* e caso confirmem a existência de um incidente de segurança ele deve reunir esses dados e enviá-los a um CSIRT.

2.5.3 Grupos de resposta a incidentes de segurança em computadores

Um Grupo de Resposta a Incidentes de Segurança em Computadores (CSIRTs - do inglês "*Computer Security Incident Response Team)* é uma organização responsável por receber, analisar e responder a notificações e atividades relacionadas a incidentes de segurança em computadores.

Um CSIRT normalmente presta serviços para uma comunidade bem definida, que pode ser a entidade que o mantém, como uma empresa, um órgão governamental ou uma organização acadêmica. Um CSIRT também pode prestar serviços para uma comunidade maior, como um país, uma rede de pesquisa ou clientes que pagam por seus serviços [15].

Como os usuários são a principal fonte de informação, é importante que todos enviem notificações de incidentes ocorridos ou vulnerabilidades encontradas em algum sistema. É por meio dessas notificações de incidentes que os membros dos CSIRTs poderão auxiliar os domínios com problemas de segurança.

Atualmente, uma notificação de incidente ou de vulnerabilidade em sistema pode ser feita utilizando-se formulários próprios, disponibilizados pelos CSIRTs, que devem ser enviados por correio eletrônico, fax ou pelo correio, além de um número telefônico para emergências. Também é possível fazer uma notificação de incidentes utilizando-se um sistema que guia o usuário por uma série de questões sobre um incidente de segurança.

Após análise das notificações de incidentes, os CSIRTs podem emitir alertas que contém informações sobre novas vulnerabilidades descobertas e que podem auxiliar os administradores a manter suas redes mais seguras. Atualmente, esses alertas são tornados públicos por meio de páginas HTML.

2.5.4 *Common Vulnerabilities and Exposures (CVE)*

O CVE [18] é uma lista ou dicionário que provê nomes comuns para vulnerabilidades e *exposures*[2] publicamente conhecidos. Utilizando nomes comuns é

[2] O termo *exposure* refere-se a fatos relativos à segurança que podem não ser considerados como vulnerabilidades por todas as pessoas.

possível compartilhar dados entre bases de dados e ferramentas diferentes que até então não são facilmente integráveis.

Muitas ferramentas de segurança da informação incluem uma base de dados de vulnerabilidades e *exposures,* entretanto existe uma grande variação entre elas e não há uma forma eficaz de determinar quando bases de dados diferentes estão se referindo ao mesmo problema. As conseqüências são potenciais falhas na segurança e na interoperabilidade entre essas bases de dados e ferramentas.

Desta forma, a utilização de uma nomenclatura comum para vulnerabilidades e *exposures* trará conseqüências benéficas quando se deseja o compartilhamento de informações de segurança e a interoperabilidade de diferentes ferramentas.

2.6 Recuperação e Filtragem de Informação

O acesso à informação precisa de mecanismos que possibilitem localizar e recuperar informações relevantes aos usuários de forma eficiente. Neste trabalho os conceitos da recuperação e filtragem de informações são utilizados para atualizar de forma automática o mecanismo de detecção de ataques dos SDIs multiagente.

Recuperação de informação não é a mesma coisa que acesso a uma base de dados. Os sistemas de gerenciamento de bases de dados fornecem respostas que coincidem exatamente com uma consulta feita. Entretanto os sistemas de recuperação de informações podem incluir na resposta itens que podem ou não coincidir exatamente com a consulta que lhe foi submetida [6].

O acesso a grandes coleções de dados desestruturadas é inevitável porque nem sempre é viável economicamente estruturar essas coleções .

Se a necessidade do usuário for pontual, ele elabora uma consulta e o sistema de recuperação [8] deve acessar uma base indexada, comparar a consulta com o conteúdo dessa base e entregar ao usuário aqueles itens de informação considerados relevantes.

Indexação, estratégia de busca, processo de recuperação e a avaliação dos itens recuperados são as funções principais de um sistema de recuperação de informação.

O conjunto de termos de índice associados a um item de informação por meio do processo de indexação constitui a representação interna do item de informação. Ainda é possível atribuir pesos aos termos de índice para refletir a sua importância relativa ao contexto ao qual o item de informação está inserido.

A estratégia de busca estabelece os mecanismos para extrair os termos da consulta e mapeá-los em uma representação interna, por um processo similar ao da indexação.

A comparação das representações internas dos itens de informação e das consultas elaboradas é parte do processo de recuperação que seleciona uma lista de itens que fornecem uma resposta exata ou aproximada para a consulta feita.

De posse da lista de resultados é possível avaliar os itens recuperados, computando uma medida de similaridade que estima o quanto um item de informação satisfaz a necessidade de informação.

Quando o usuário possui uma necessidade de informação a longo prazo, o sistema realiza apropriadamente a filtragem [9], um processo que consiste em monitorar constantemente fontes de informação em busca de novos itens, compará-los com os interesses do usuário, representados por perfis, e apresentação dos itens relevantes ao usuário.

2.7 *Web Services* e XML

Web services são serviços oferecidos por uma aplicação para outras aplicações por meio da *World Wide Web* comumente utilizando o *Simple Object Access Protocol* (SOAP) [30] sobre o *Hypertext Transport Protocol* (HTTP).

O serviço recebe a requisição do cliente, a processa e devolve a resposta. *Web services* e seus usuários geralmente são aplicações e um provedor de *Web service* pode ser também um consumidor de outro *Web service*.

Essa habilidade para troca de informações entre entidades que provavelmente utilizam sistemas diferentes é obtida utilizando-se a XML, uma linguagem de marcação que torna dados portáveis.

A XML contém dados entre *tags,* assim como a HTML, entretanto essas *tags* estão relacionadas ao significado do texto incluído em cada uma delas. Além disso, a XML é extensível o que significa que novas *tags* podem ser criadas para descrever o conteúdo de um documento em particular.

Para se descrever a estrutura de um determinado tipo de documento XML pode-se criar um arquivo chamado esquema*.* O esquema fornece a portabilidade de dados a XML. Se uma aplicação recebe um documento em formato XML e ela possui o esquema desse arquivo ela pode processar o arquivo de acordo com as regras especificadas*.*

A XML é escrita em formato texto possibilitando a leitura dos documentos tanto por humanos quanto por programas e ela não possui informações de formatação, podendo ser exibida de várias formas.

Para a utilização de *Web services* e da XML é necessário que as partes acordem sobre certas condições [2]:

- Qual será o conjunto de elementos que elas usarão e o que esses elementos significam, ou seja, qual o formato padrão para a representação dos alertas de segurança;
- Quais métodos dos *Web services* elas utilizarão, o que esses métodos fazem e a ordem que esses métodos serão invocados, quando mais de um método for necessário.

2.8 Conclusões

Neste capítulo foi apresentado o NIDIA que é um SDI multiagente baseado na arquitetura CIDF e vários formatos de mensagens para alertas de segurança tanto relacionados à segurança dos computadores como o IDMEF, IODEF e o *Common Advisory Format Description*, assim como o CAP que é um formato para a troca de todos os tipos de alertas de emergência.

Também foram vistos tópicos relacionados à segurança dos computadores, entre eles destaca-se o papel dos CSIRTs que são responsáveis por analisar notificações de incidentes de segurança e emitir alertas contendo informações sobre novas vulnerabilidades.

Constatou-se que não há um formato de mensagens para os alertas emitidos pelos CSIRTs. Esses alertas contêm informações que podem ser utilizadas para responder a um ataque, mas devido a utilização do formato HTML por parte dos CSIRTs, torna-se extremamente difícil analisá-los por meio de um sistema automatizado. Desta forma, apresenta-se no capítulo 3 uma proposta para o compartilhamento de informações entre os CSIRTs e os SDIs que utiliza um formato padrão de mensagens de alerta em XML e *Web services*.

3. PROPOSTA DE MODELO DE COMPARTILHAMENTO DE INFORMAÇÕES ENTRE CSIRTs E SDIs.

Este capítulo apresenta um modelo de compartilhamento de informações entre CSIRTs e SDIs, tendo como objetivo atualizar de forma automática o conjunto de ações de resposta a intrusões dos SDIs, com base nas medidas restritivas de curto prazo sugeridas nos alertas de segurança emitidos pelos CSIRTs utilizando-se as tecnologias de *Web services* e XML [35].

3.1 Módulo de Compartilhamento de Informações para SDIs.

De acordo com o *Common Intrusion Detection Framework* (CIDF), a habilidade dos SDIs e seus componentes de compartilhar informações sobre incidentes de segurança é muito importante e permitiria que um sistema alertasse outro SDI a respeito de possíveis ataques iminentes [19].

Entretanto não seria seguro permitir o acesso direto aos componentes de um SDI por uma entidade externa. Desta forma, é proposto neste trabalho a introdução de um módulo de compartilhamento de informações à arquitetura CIDF que eliminaria este acesso direto e poderia fornecer as seguintes funções de segurança necessárias a este tipo de aplicação: criptografia e assinatura digital. O detalhamento dessas funções de segurança está fora do escopo deste trabalho, sendo objeto de estudo de outra dissertação em andamento.

A Figura 3.1 exibe a proposta de adição do módulo de compartilhamento de informações (eXchange – box) à arquitetura CIDF.

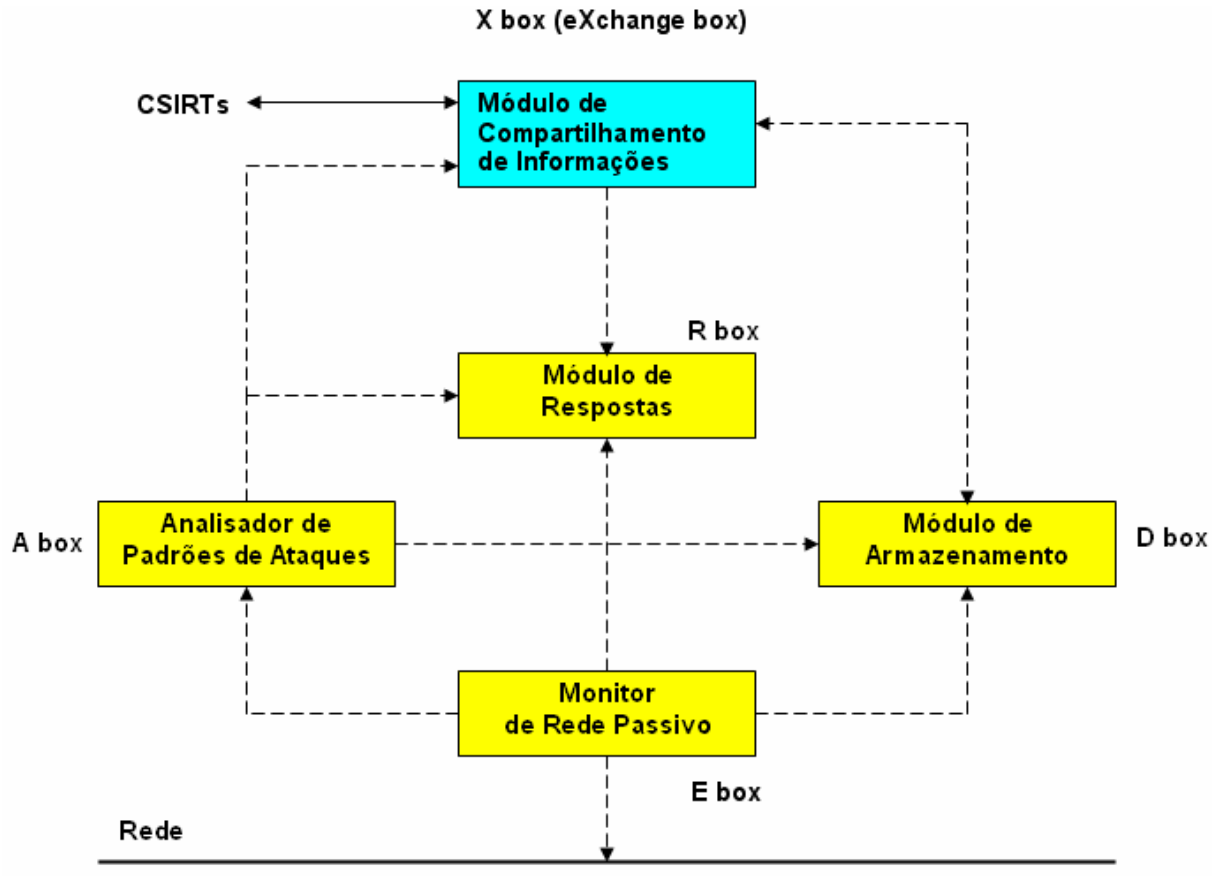


Figura – 3.1. Módulo de Compartilhamento de Informações.

É importante que a troca de informações, entre os SDIs e as entidades com as quais eles interagem, ocorra com o mínimo de interferência nas atividades normais de cada parte envolvida. Outro requisito para a viabilidade da introdução do Módulo de Compartilhamento é a preservação dos sistemas e bases de dados existentes

A Figura 3.2 apresenta um conjunto de domínios monitorados por diferentes SDIs em que o compartilhamento de informações é adicionado aos SDIs existentes, em forma de um novo módulo. Assim que este módulo for ativado o domínio poderá compartilhar informações de segurança com os CSIRTs.

O papel dos CSIRTs é de analisar informações de incidentes de segurança enviadas pelos diversos domínios em busca de padrões de ataques ou de novas vulnerabilidades. Se uma ameaça for detectada com base na análise das

informações dos incidentes, o CSIRT irá gerar um alerta de segurança que será disponibilizado a todos os domínios.

Um alerta de segurança é basicamente constituído pela identificação do CSIRT que o emitiu, o nome e descrição da vulnerabilidade por ele descrita, possíveis impactos, sistemas afetados e soluções. Entre as soluções encontram-se a indicação de correções para os sistemas ou *patches* e também ações de resposta que podem ser tomadas quando um ataque que explore a vulnerabilidade descrita é detectado.

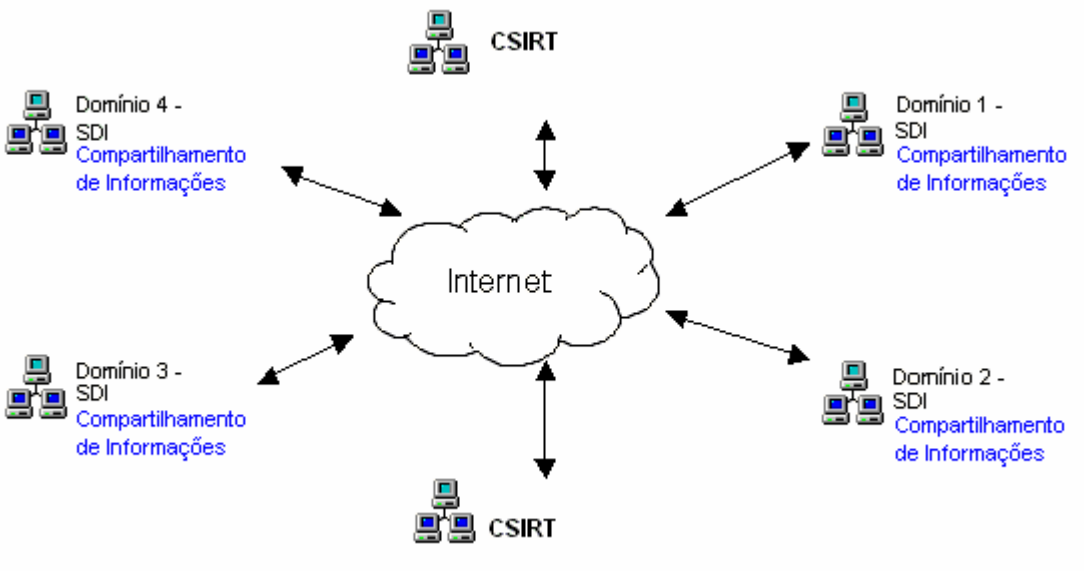


Figura - 3.2. Arquitetura do Sistema.

Caso um novo alerta de segurança seja emitido por um CSIRT, os SDIs deverão recuperá-lo e a partir das informações disponibilizadas poderão atualizar seu conjunto de ações de resposta a intrusões. Também será possível tomar medidas preventivas como por exemplo baixar e instalar uma nova correção para os sistemas instalados. No futuro quando um ataque for detectado, o SDI poderá responder de acordo com as informações do seu conjunto de ações de resposta.

3.2 As Ações de Resposta a Intrusões

Uma resposta pode ser definida como um conjunto de ações que um SDI pode executar quando ele detecta uma intrusão [4]. A seguir são listadas as principais técnicas de respostas [41]:

- Gerar Relatório **-** Deve ser gerado um relatório de todo comportamento intrusivo detectado para que administrador de sistema acompanhe o andamento dos incidentes. Esses relatórios irão fornecer uma visão geral do que ocorreu com o sistema e permitir a análise segura dos ataques ao longo do tempo;
- Gerar Alarmes **-** Alarmes notificam o administrador da detecção de um ataque ou de uma tentativa de ataque contra o sistema. As mensagens de alerta são enviadas por *e-mail*, janela de notificação *pop up*, da ativação de mensagem de *pager* e ainda envio de mensagem para telefones celulares;
- Suspender *Jobs* e encerrar a sessão do Usuário - quando há indicação de atividades intrusivas durante o uso do sistema por um usuário normal, pode-se suspender os *jobs* desse usuário, encerrar a sua sessão e bloquear sua senha antes que atividades válidas sejam corrompidas e que sejam causados danos;
- Investigação de Suspeitos - devido ao grande número de falsos positivos deve-se evitar medidas severas. Havendo suspeita de um comportamento intrusivo, pode-se bloquear a execução de certos tipos de comandos sem excluir totalmente o usuário do sistema, monitorar suas atividades para adquirir informações adicionais e assim classificar definitivamente o comportamento do usuário. Essas medidas viabilizam

a coleta de informações adicionais necessárias para a classificação de um ataque ou atacante e reduzem a possibilidade do intruso causar danos ao sistema antes que ele seja excluído;

- Bloquear endereço IP **-** quando há um ataque de rede e a sua origem é descoberta, como resposta imediata, pode ser obstruído no roteador todo o seu tráfego. Mas essa solução é provisória e só permite que se tenha mais tempo para reagir. Um atacante que tem seu endereço IP bloqueado pode utilizar outro endereço e atacar novamente;
- Derrubando a Máquina **-** a única forma de proteger um sistema comprometido contra um ataque em estágio avançado é desligando a máquina. Apesar de ser uma medida extrema, essa é a única forma de protegê-la de maiores danos;
- Usar uma ferramenta de detecção de intruso adicional **-** devido à deficiência das ferramentas de detecção de intrusão e o alto consumo de recursos do sistema, sistemas de resposta de intrusão podem solicitar ajuda de outros meios de detecção de intrusão para a definição do estado de uma intrusão. Neste caso, ferramentas de detecção de intrusão são empregadas como indicadores adicionais de que um comportamento intrusivo foi detectado;
- Desabilitar portas e serviços afetados **-** quando um ataque usar um serviço ou uma porta conhecida para atingir um sistema, é possível parar efetivamente o ataque, bloqueando os recursos atingidos, sem afetar qualquer outro serviço do sistema;
- Investigar a conexão **-** a investigação de uma conexão de rede para tentar identificar um atacante é uma reposta viável, porque além de

descobrir a identidade do intruso, ele pode detectar que está sendo investigado, se intimidar e desistir do ataque;

- Criar cópias de segurança **-** ataques contra a integridade do sistema podem ter suas conseqüências reduzidas se forem criadas cópias de segurança. Cópias atualizadas permitem a restauração do sistema e a comparação de arquivos. Apesar de ser impossível manter cópias atualizadas em tempo real, pode-se encurtar o intervalo de tempo entre a execução de novas cópias reduzindo a perda de dados;
- Utilizar arquivo temporário de proteção - arquivo temporário de proteção serve como um mecanismo para assegurar a integridade do sistema enquanto estiver sendo vítima de um ataque. Uma cópia do arquivo original é criada e codificada sempre que alguém tentar modificar arquivos críticos do sistema, todas as modificações são salvas em um segundo arquivo e o original permanece intacto.

3.3 Proposta de Atualização Automática do Conjunto de Ações de Resposta dos SDIs com Base nos Alertas Emitidos pelos CSIRTs

Conforme mostra a Figura 3.3, o módulo de compartilhamento de informações do SDI, será responsável por coletar alertas de segurança emitidos por diversos CSIRTs que conterão informações sobre novas vulnerabilidades detectadas e que serão disponibilizados em um formato padronizado.

De posse das informações contidas nos alertas de segurança, o módulo de compartilhamento de informações deverá atualizar o conjunto de ações de resposta a intrusões do SDI, que será constituído por uma Base de Dados de Ações de Respostas (RADB).

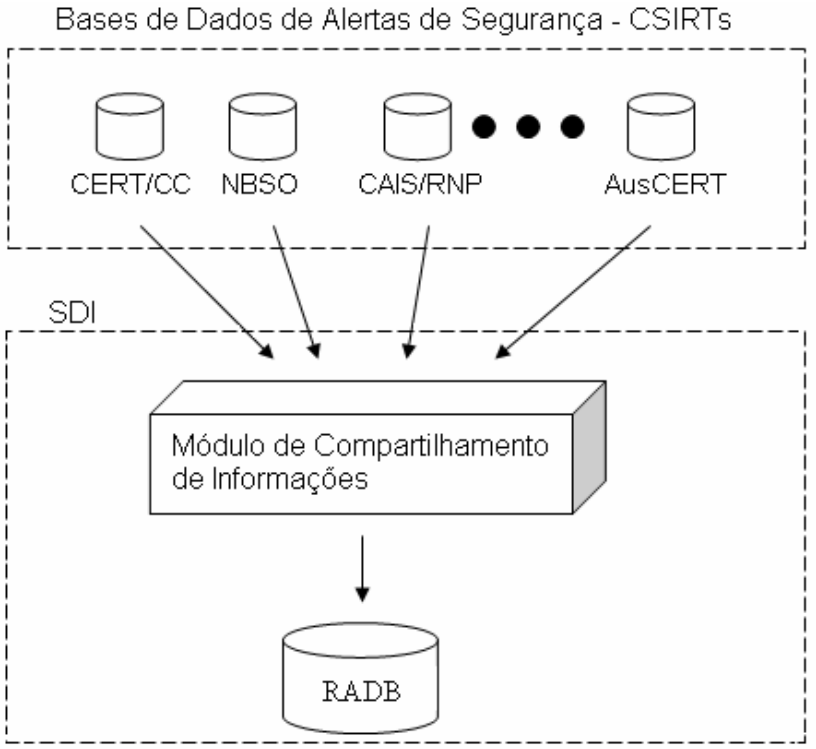


Figura – 3.3. Modelo de atualização automática da base de dados de ações de respostas.

Atualmente os alertas de segurança são publicados utilizando-se a *HyperText Markup Language* (HTML), e devem ser lidos e interpretados por um especialista humano, dificultando ou mesmo impossibilitando o seu processamento de forma automática.

Com os alertas sendo publicados em formato HTML, o administrador de segurança deve ler todos os alertas emitidos e verificar quais vulnerabilidades publicadas aplicam-se ao seu ambiente e tomar as medidas sugeridas.

Para que as informações disponibilizadas nos alertas de segurança sejam processadas de forma automática pelos SDIs necessitam-se de meios que possibilitem a comunicação entre um SDI e os CSIRTs por intermédio da Internet e de um formato padronizado para esses alertas.

Desta forma, propõe-se a criação de um *Web service* nos CSIRTs, isto é, uma aplicação que pode disponibilizar os alertas de segurança para seus clientes, neste caso os SDIs, por meio da Internet.

Como parte dessa solução, o módulo de compartilhamento de informações será um cliente do *Web service* dos CSIRTs e poderá enviar requisições buscando por novos alertas emitidos. O serviço recebe a requisição do módulo de compartilhamento de informações, a processa e busca no banco de dados do CSIRT por novos alertas. Caso novos alertas sejam encontrados, um documento XML para cada alerta é criado e anexado à resposta que é enviada de volta ao módulo de compartilhamento de informações.

Como provavelmente os SDIs e os CSIRTs possuem diferentes sistemas de informação, a habilidade de comunicação entre eles é muito importante e a XML é uma tecnologia que provê a portabilidade de dados necessária.

Sempre que o módulo de compartilhamento de informações receber um novo alerta emitido por um CSIRT, ele será processado e suas informações serão armazenadas na RADB.

Na Figura 3.4 apresenta-se um detalhamento do modelo proposto para a atualização automática da base de dados de ações de respostas exibido na Figura 3.3.

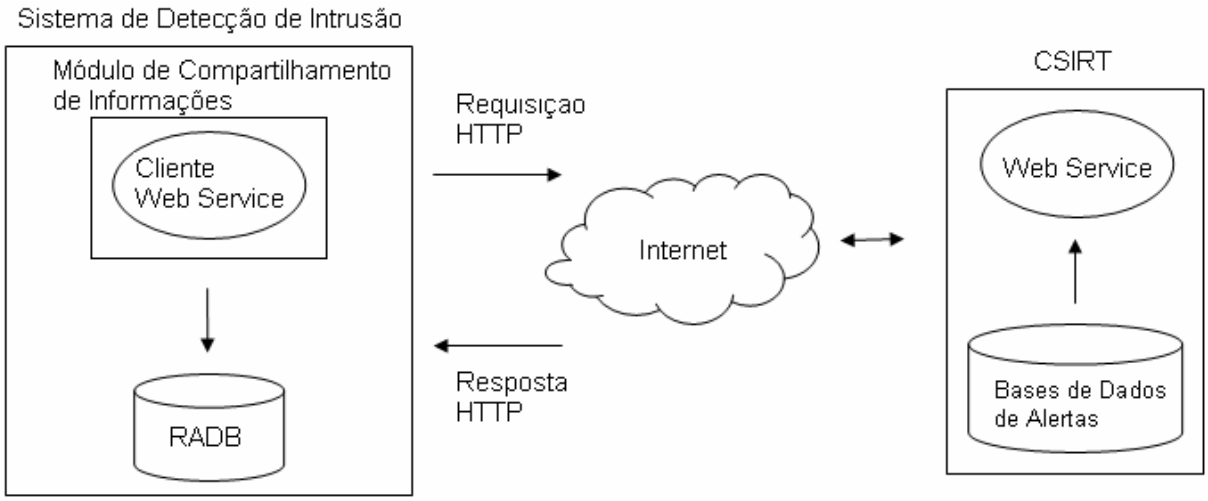


Figura – 3.4. Detalhamento do módulo de compartilhamento de informações.

A Tabela 3.1 mostra o número de notas de vulnerabilidades publicadas pelos CSIRTs CERT®/CC e AusCERT no período entre 2000 e setembro de 2005.

Tabela - 3.1. Notas de vulnerabilidades publicadas.

CSIRT/Ano	2000	2001	2002	2003	2004	2005
CERT®/CC	47	326	375	255	341	189
AusCERT	404	547	724	872	827	700

Pode-se observar que a média diária da publicação dessas notas é inferior a três. Desta forma, haverá uma baixíssima taxa diária de envio de requisições entre o módulo de compartilhamento de informações e os CSIRTs e ainda assim a RADB poderá ser mantida atualizada de forma segura. Evidentemente o módulo de compartilhamento de informações também não terá um elevado nível de processamento para traduzir os alertas, em formato XML, e atualizar a RADB.

Tem-se então um procedimento de atualização da RADB semelhante ao empregado pelos antivírus, entretanto não há uma dependência dos SDIs em relação a seus fabricantes para que tal atualização seja feita.

A Figura 3.5 mostra um alerta emitido pelo CERT®/CC que se constitui de uma página no formato HTML. Algumas partes foram retiradas para melhor entendimento.

No alerta da Figura 3.5, o administrador de segurança após verificar que o seu servidor *Concurrent Versions System* (CVS) está vulnerável, ele deve aplicar as correções fornecidas pelo fabricante. Pode-se ainda observar na seção “Solution - Solução” que medidas restritivas de curto prazo do tipo “Disable CVS Server - Desabilite um serviço” ou “Block or Restrict Access - Bloquear ou Restringir o Acesso” podem ser utilizadas para interromper o acesso de um provável invasor aos sistemas comprometidos, limitar a extensão de uma intrusão ou impedir que esse intruso cause danos adicionais [26].

São exatamente essas medidas restritivas de curto prazo que podem alimentar a RADB, mantendo o mecanismo de respostas do SDI atualizado e apto a responder a incidentes de segurança. Para que a resposta seja possível é necessário que o mecanismo de detecção de ataques também esteja atualizado. A atualização do mecanismo de detecção de ataques é descrita no capítulo 4.

CVS Heap Overflow Vulnerability

Original release date: May 26, 2004
Last revised: --
Source: US-CERT

Systems Affected

- Concurrent Versions System (CVS) versions prior to 1.11.16
- CVS Features versions prior to 1.12.8

Overview

A heap overflow vulnerability in the Concurrent Versions System (CVS) could allow a remote attacker to execute arbitrary code on a vulnerable system.

I. Description

CVS is a source code maintenance system that is widely used by open-source software development projects. There is a heap memory overflow vulnerability in the way CVS handles the insertion of modified and unchanged flags within entry lines.

...

US-CERT is tracking this issue as VU#192038. This reference number corresponds to CVE candidate CAN-2004-0396.

II. Impact

An authenticated client could exploit this vulnerability to execute arbitrary code on the vulnerable system with the privileges of the CVS server process. It is possible for an anonymous user with read-only access to exploit a vulnerable server as they are authenticated through the cvspserver process.

...

Solution

Apply Patch or Upgrade

Apply the appropriate patch or upgrade as specified by your vendor. For vendor specific responses, please see your vendor's website or Vulnerability Note VU#192038.

This issue has been resolved in Stable CVS Version 1.11.16 and CVS Feature Version 1.12.8.

Disable CVS Server

Until a patch or upgrade can be applied, consider disabling the CVS server.

Block or Restrict Access

- Block or restrict access to the CVS server from untrusted hosts and networks. The CVS server typically listens on 2401/tcp, but may use another port or protocol.

...

Figura – 3.5. Alerta emitido pelo CERT®/CC em HTML.

A padronização dos alertas de segurança é muito importante para que um SDI não tenha que entender e decodificar vários formatos. Desta forma, propõe-se o formato de representação dos alertas como uma extensão do *Common Alerting Protocol* (CAP).

No capítulo 2 outros formatos de padronização de compartilhamento de informações de segurança foram apresentados, como é o caso do IODEF e do IDMEF que são propostas do *Internet Engineering Task Force* (IETF). Entretanto a proposta aqui apresentada possui um objetivo diferente das propostas do IETF.

A proposta deste trabalho é de um formato padrão para os alertas emitidos pelos CSIRTs. Esses alertas são destinados ao público em geral e constituem-se basicamente de informações a respeito de potenciais ameaças e de como evitá-las, minimizá-las ou recuperar-se de danos por elas causados.

As propostas do IETF são formatos de dados para alertas possivelmente gerados por um sistema automatizado de detecção de intrusão. Esses alertas têm como objetivo informar ao administrador de rede sobre possíveis incidentes de segurança e também fornecê-lhe dados a serem enviados a um Grupo de Resposta a Incidentes de Segurança em Computadores.

Outro padrão descrito foi o *Common Advisory Format Description* que visa automatizar a troca de informações entre os CSIRTs e a comunidade do EISPP. Entretanto seu formato não permite a atualização automática do conjunto de ações de resposta dos SDIs.

Desta forma, a extensão do CAP evita a utilização de um padrão exclusivo para alertas referentes à segurança dos computadores e propicia um formato que permite a automação da tomada de ações corretivas.

A Figura 3.6 mostra onde os diversos formatos para compartilhamento de informações de segurança são utilizados.

A extensão do CAP foi feita por meio da adição de elementos que possibilitam a identificação de sistemas vulneráveis e das ações de resposta. Para que tal conjunto de novos elementos fosse construído foram analisados diversos

alertas de segurança emitidos pelo CERT©/CC no ano de 2004 e a partir dessa análise foi possível definir 06 (seis) elementos que representam todas as ações de resposta sugeridas nos alertas: <*applyPatch*>, <*removePatch*>, <*blockAccess*>, <*disableFeature*>, <*modifyFile*> e <*permissionChange*>.

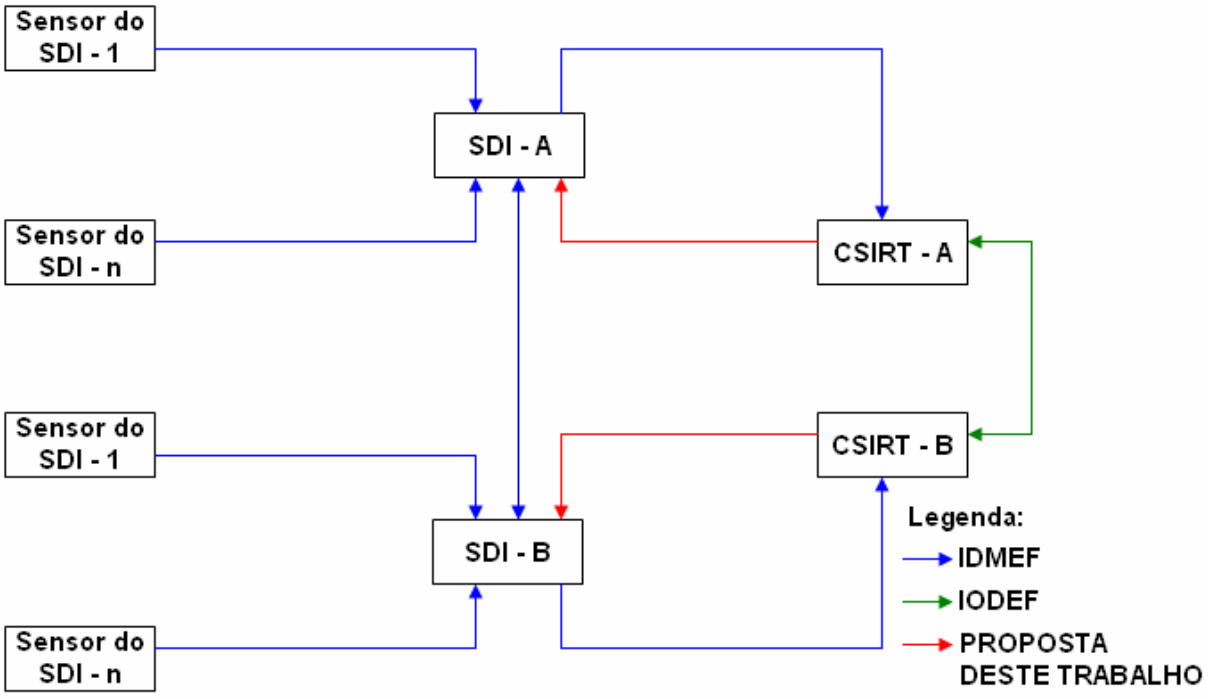


Figura – 3.6. Utilização dos formatos de mensagens para compartilhamento de informações de segurança.

A estrutura dessa mensagem de alerta acerca de problemas de segurança da Internet baseada na estrutura da mensagem de alerta CAP consiste de um segmento <*alert*> que pode conter um ou mais segmentos <*restriction*> e <*info*>. Os segmentos <*info*> conterão um segmento <*instruction*>, e esse segmento <*instruction*> conterá os novos elementos dessa proposta de extensão do format CAP. Essa estrutura é mostrada na Figura 3.7 utilizando-se a UML.

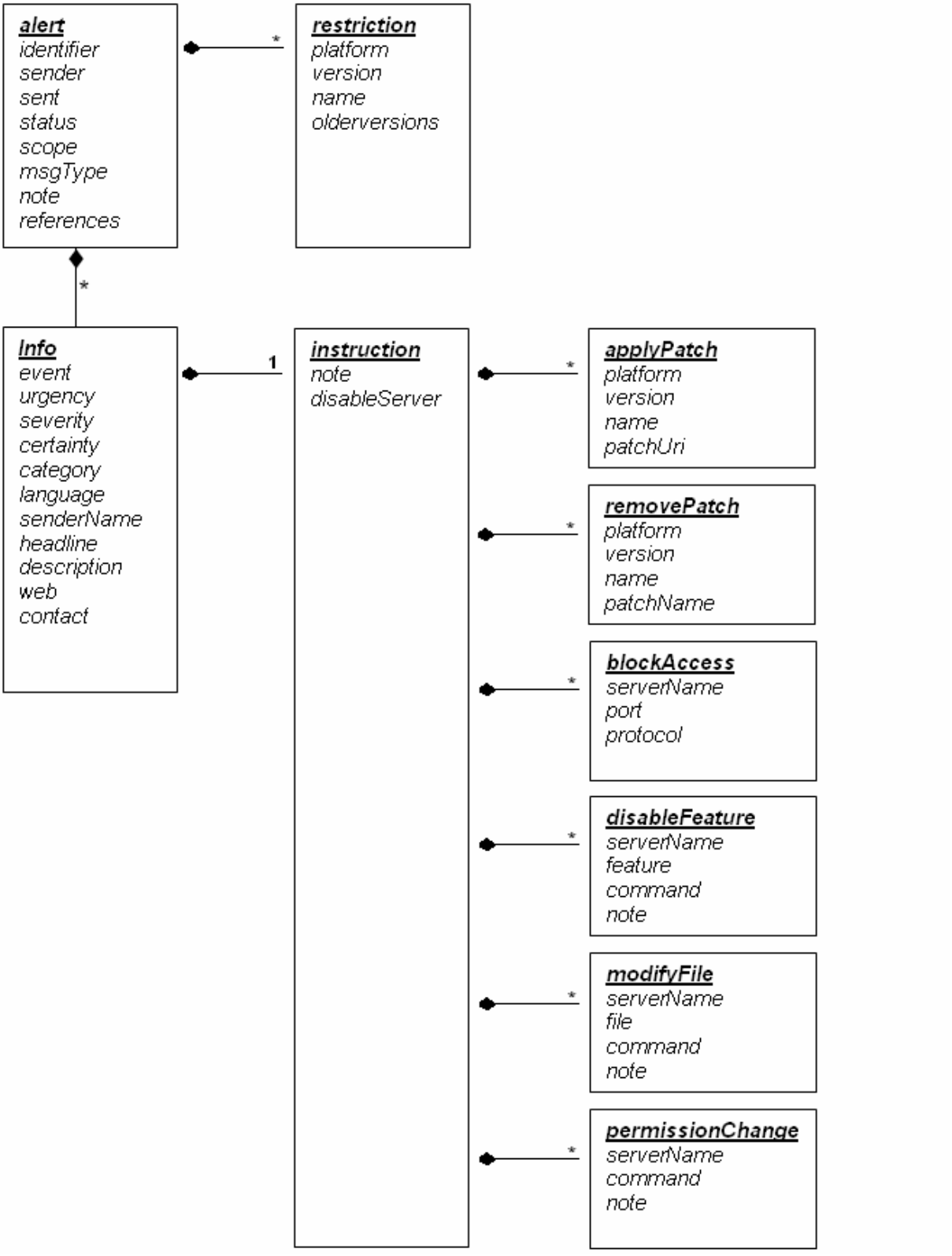


Figura – 3.7. Estrutura da mensagem de alerta de problemas de segurança da Internet.

As seguintes extensões foram introduzidas na estrutura da mensagem de alerta CAP:

- O elemento obrigatório <*scope*> da mensagem CAP, contido no elemento <*alert*>, deverá ter o seu valor definido como "*Restricted*", indicando que um determinado alerta é destinado a usuários de sistemas específicos;
- O elemento opcional <*restriction*>, contido no elemento <*alert*>, foi redefinido como um segmento obrigatório que irá indicar a quais sistemas um alerta se refere;
- O elemento opcional <*instruction*>, contido no elemento <*info*>, foi redefinido como um segmento obrigatório, e os seguintes elementos são sugeridos para identificar as ações de resposta a serem tomadas: <*applyPatch*>, <*removePatch*>, <*blockAccess*>, <*disableFeature*>, <*modifyFile*> e <*permissionChange*>;
- Como o CAP não prevê um código para problemas de segurança na Internet, criou-se um valor de código para o sub-elemento <*category*> do elemento <*info*>. Esse novo código é "Internet" e indicará que uma mensagem de alerta refere-se a problemas de segurança na Internet.

O apêndice A contém o esquema da RADB que é uma representação do formato de codificação proposto para os alertas emitidos pelos CSIRTs.

3.4 O Dicionário de Dados

Nesta seção descreve-se os elementos que compõem o CAP e as extensões propostas. Essas extensões têm como objetivo a obtenção de um formato de mensagens para alertas de segurança emitidos pelos CSIRTs que possibilite uma atualização automática do conjunto de ações de respostas dos SDIs.

Os nomes dos novos elementos encontram-se em negrito.

Nome do Elemento	**Definição e (opção)**	**Notas ou Valor de Domínio.**
Elemento <*alert*> e sub-elementos		
alert	*Container* para todas as partes componentes da mensagem de alerta. (obrigatório)	(1) Circunda os sub-elementos da mensagem de alerta CAP. (2) Deve incluir o atributo xmlns que referencia a URI CAP como o espaço de nome. (3) Adicionalmente aos sub-elementos especificados contém obrigatoriamente um ou mais elementos <*restriction*> e pode conter opcionalmente um ou mais elementos <*info*> .
identifier	O identificador de uma mensagem de alerta. (obrigatório)	(1) Identifica uma vulnerabilidade de forma única. (2) Nomes de vulnerabilidades de acordo com o *Common Vulnerabilities and Exposures* (CVE) que provê uma lista ou dicionário para nomes de vulnerabilidades.
sender	O identificador do emissor da mensagem de alerta. (obrigatório)	(1) Identifica a entidade que gerou o alerta. Deve ser um identificador globalmente único. Pode ser baseado em nomes de domínios Internet. (2)Sem espaços ou caracteres restritos(< e &).
sent	A hora e data de geração da mensagem de alerta. (obrigatório)	A data e hora são representados no formato ISO 8601.
status	Código que indica o correto manuseio da mensagem de alerta. (obrigatório)	Valores dos Códigos: "*Actual*" - As informações e procedimentos contidos na mensagem de alerta devem ser observados por todos. "*Exercise*" - As informações e procedimentos contidos na mensagem de alerta devem ser observados apenas pelos participantes de um

		exercício; o identificador do exercício deve estar incluído no elemento <note> . “*System*” - Utilizado em mensagens que dão suporte a funções internas da rede de alerta. “*Test*” - Apenas para testes técnicos. Deve ser desconsierada por todos.
scope	Código que denota o tipo de distribuição pretendida para uma mensagem de alerta. (obrigatório)	Valores dos Códigos: “*Public*”- Para disseminação geral a todo tipo de audiência. “*Restricted*” - Para disseminação apenas a usuários que possuam um requerimento operacional conhecido (ver elemento <*restriction*>). “*Private*” - Para disseminação apenas a endereços especificados (ver elemento <*addresses*> abaixo).
msgType	Código que denota a natureza de uma mensagem de alerta. (obrigatório)	Valores dos Códigos: “*Alert*” - Informação inicial que requer atenção por parte dos destinatários. “*Update*” - Atualiza e substitui uma ou mais mensagens anteriores identificadas pelo elemento <*references*>. “*Cancel*” - Cancela uma ou mais mensagens anteriores identificadas pelo elemento <references>. “*Ack*” - Confirma o recebimento e a aceitação de uma ou mais mensagens identificadas pelo elemento <*references*>. “*Error*” - Indica a rejeição de uma ou mais mensagens identificadas pelo elemento <*references*>; uma explicação pode estar no elemento <*note*>.
addresses	Grupo que lista os destinatários de uma mensagem de alerta privada. (condicional)	(1) Utilizado quando o valor do elemento <*scope*> é "*Private*". (2) Cada destinatário pode ser identificado por um identificador ou um endereço. (3) Múltiplos endereços delimitados por espaços podem ser utilizados. Endereços que incluem espaços em branco devem estar entre aspas duplas.
note	Texto que descreve o propósito ou significado de uma mensagem de alerta. (opcional)	Notas de mensagens são destinadas para uso com as mensagens de alerta do tipo “*Cancel”* e “*Error”*.
references	Uma lista de mensagens anteriores referenciadas por	(1) Identificador estendido de mensagem (na forma <*identifier*>/<*sender*>) de uma ou mais mensagens anteriores referenciadas por um alerta.

	uma mensagem de alerta. (opcional)	(2) Se múltiplas mensagens são referenciadas, elas são separadas por espaços em branco.

Elemento <*restriction*> e sub-elementos

restriction	*Container* para todas as partes componentes do sub-elemento <*restriction*> da mensagem de alerta. (obrigatório)	(1) Múltiplas ocorrências são permitidas em um único elemento <*alert*>. (2) Como se definiu o valor de <*scope*> como "*Restricted*", este elemento sempre será utilizado. (3)Contém a identificação dos sistemas que possuem algum tipo de vulnerabilidade.
platform	Código especificando o nome da plataforma do sistema ao qual a mensagem de alerta se refere. (opcional)	Valores dos Códigos: "*Sparc*" – Plataforma *Sparc Sun.* "*x86*" – Plataforma *Intel.* "*Windows*" – *Microsoft Windows.* "*Solaris*" – Plataforma *Solaris* da *Sun.* "Linux" – Plataforma Linux.
version	Identificador da versão do sistema ao qual a mensagem de alerta se refere. (opcional)	(1) Um número ou "*string*" que identifica unicamente a versão de um sistema que é atribuído pelo fabricante. (2)Sem espaços ou caracteres restritos(< e &).
name	Identificador do nome do sistema ao qual a mensagem de alerta se refere. (obrigatório)	(1) Uma "*string*" que identifica unicamente o nome de um sistema que é atribuído pelo fabricante. (2)Sem espaços ou caracteres restritos(< e &).
olderVersions	Código que especifica se versões anteriores do sistema também são afetadas por um alerta. (opcional)	Valores dos Códigos: "*Y*" – Todas as versões anteriores do sistema são afetadas por um mesmo alerta. "*N*" – As versões anteriores do sistema não são afetadas por um mesmo alerta.

Elemento <info> e sub-elementos

info	*Container* para todas as partes componentes do sub-elemento <*info*> da mensagem de alerta. (opcional)	(1) Múltiplas ocorrências são permitidas em um único <*alert*>. Se múltiplos elementos <*info*> do mesmo idioma se sobrepõem, a informação nos elementos posteriores podem ampliar mas possivelmente não anulam os valores correspondentes dos elementos anteriores. Cada conjunto de elementos <*info*> contendo o mesmo identificador de idioma deve ser tratado como uma seqüência separada.
event	Texto que significa o tipo de assunto de uma mensagem de alerta. (obrigatório)	O texto poderá utilizar uma determinada nomenclatura caso disponível.
urgency	Código especificando a urgência do assunto de uma mensagem de alerta. (obrigatório)	(1) O conjunto dos elementos <*urgency*>, <*severity*>, <*certainty*> pode distinguir mensagens menos enfáticas das mais enfáticas. (2) Valores dos Códigos: "*Immediate*"- Ações de resposta devem ser tomadas imediatamente. "*Expected*" - Ações de resposta devem ser tomadas em pouco tempo (dentro da próxima hora). "*Future*" - Ações de resposta devem ser tomadas no futuro próximo. "*Past*" - Ações de resposta não são mais necessárias. "*Unknown*" - Urgência desconhecida.
severity	Código especificando a severidade do assunto de uma mensagem de alerta. (obrigatório)	(1) O conjunto dos elementos <*urgency*>, <*severity*>, <*certainty*> pode distinguir mensagens menos enfáticas das mais enfáticas. (2) Valores dos Códigos: "*Extreme*" - Ameaça extraordinária a vida ou a propriedade. "*Severe*" - Ameaça significante a vida ou a propriedade. "*Moderate*" - Possível ameaça a vida ou a propriedade. "*Minor*" - Ameaça mínima a vida ou a propriedade. "*Unknown*" - Severidade desconhecida.
certainty	Código especificando a possibilidade de um fato.	(1) O conjunto dos elementos <*urgency*>, <*severity*>, <*certainty*> pode distinguir mensagens menos enfáticas das mais enfáticas.

	(obrigatório)	(2) Valores dos Códigos: “*Very Likely*” - Altamente provável (p >~85%) ou certeza. “*Likely*” - Provável (p >~50%). “*Possible*” - Possível mas não provável (p <=~50%). “*Unlikely*” - Não esperado que ocorra (p~0). “*Unknown*” - Certeza desconhecida.
category	Código especificando a categoria do assunto de uma mensagem de alerta. (opcional)	(1) Como o CAP não prevê um código para problemas de segurança na Internet, criou-se um valor para essa categoria: “Internet” - Problemas de segurança na Internet (2) Múltiplas ocorrências são permitidas em um único elemento <*info*>.
language	Código especificando o idioma do sub-elemento <*info*> de uma mensagem de alerta. (opcional)	(1) Valores dos Códigos: identificadores de linguagem natural pela RFC 1766. (2) Se não estiver presente, o valor assumido será “en-US”.
senderName	Texto que nomeia o criador de uma mensagem de alerta. (opcional)	Nome da agência ou autoridade que emitiu o alerta.
headline	Título da mensagem de alerta. (opcional)	Um título breve.
description	Texto descrevendo o assunto de uma mensagem de alerta. (opcional)	
web	Identificador do *hyperlink* que apresenta informações adicionais relacionadas a mensagem de alerta. (opcional)	Um URI completo para uma página HTML ou outro recurso de texto com informações adicionais ou de referência relacionados a mensagem de alerta.
contact	Texto descrevendo o contato para acompanhamento e confirmação de uma mensagem de alerta. (opcional)	

Elemento <*instruction*> e sub-elementos

instruction	*Container* para todas as partes componentes do sub-elemento <*instruction*> do sub-elemento <*info*> da mensagem de alerta. (obrigatório)	(1) Uma única ocorrência é permitida em um único <info>. (2) Além dos sub-elementos especificados, pode conter um ou mais blocos <*applyPatch*>, <*removePatch*>, <*blockAccess*>, <*disableFeature*>, <*modifyFile*>, <*permissionChange*>.
note	Texto que descreve as ações recomendadas que devem ser tomadas pelos destinatários da mensagem. (obrigatório)	
disableServer	Grupo que lista os nomes de servidores que devem ser desativados até que uma correção para o problema descrito no alerta esteja disponível. (opcional)	(1) Caso haja múltiplas instâncias, elas devem ser separadas por espaços em branco. Nomes de servidores que possuem espaços em branco devem estar entre aspas duplas. (2) O nome do servidor deve ser um identificador único atribuído pelo fabricante.

Elemento <*applyPatch*> e sub-elementos

applyPatch	*Container* para todas as partes componentes do sub-elemento <*applyPatch*> do sub-elemento <*instruction*> do sub-elemento <*info*> da mensagem de alerta. (opcional)	(1) Refere-se ao arquivo que contém o programa de correção para o sistema descrito no alerta. (2) Múltiplas ocorrências são permitidas em um único <*instruction*>.
platform	Código especificando o nome da plataforma do sistema ao qual a mensagem de alerta se refere. (opcional)	Valores dos Códigos: "*Sparc*" – Plataforma *Sparc Sun.* "x86" – Plataforma *Intel.* "*Windows*" – *Microsoft Windows.* "*Solaris*" – Plataforma *Solaris* da *Sun.* "Linux" – Plataforma Linux.
version	Identificador da versão do sistema ao qual a mensagem de	(1) Um número ou "*string*" que identifica unicamente a versão de um sistema que é atribuído pelo fabricante.

	alerta se refere. (opcional)	(2)Sem espaços ou caracteres restritos(< e &).
name	Identificador do nome do sistema ao qual a mensagem de alerta se refere. (obrigatório)	(1) Uma "*string*" que identifica unicamente o nome de um sistema que é atribuído pelo fabricante. (2)Sem espaços ou caracteres restritos(< e &).
patchUri	Identificador do *hyperlink* para o arquivo de correção. (obrigatório)	URI absoluto para o arquivo de correção do sistema afetado pelo alerta. Geralmente uma URL que pode ser utilizada para recuperar o arquivo de correção por meio da Internet.

Elemento <*removePatch*> e sub-elementos

removePatch	*Container* para todas as partes componentes do sub-elemento <*removePatch*> do sub-elemento <*instruction*> do sub-elemento <*info*> da mensagem de alerta. (opcional)	(1) Refere-se a correção que foi anteriormente aplicada ao sistema e que deve ser removida antes que uma nova correção seja aplicada. (2) Múltiplas ocorrências são permitidas em um único bloco <*instruction*>.
platform	Código especificando o nome da plataforma do sistema ao qual a mensagem de alerta se refere. (opcional)	Valores dos Códigos: "*Sparc*" – Plataforma *Sparc Sun.* "x86" – Plataforma *Intel.* "*Windows*" – Microsoft Windows. "*Solaris*" – Plataforma *Solaris* da *Sun.* "Linux" – Plataforma Linux.
version	Identificador da versão do sistema ao qual a mensagem de alerta se refere. (opcional)	(1) Um número ou "*string*" que identifica unicamente a versão de um sistema que é atribuído pelo seu fabricante. (2)Sem espaços ou caracteres restritos(< e &).
name	Identificador do nome do sistema ao qual a mensagem de alerta se refere. (obrigatório)	(1) Uma "*string*" que identifica unicamente o nome de um sistema que é atribuído pelo fabricante. (2)Sem espaços ou caracteres restritos(< e &).
patchName	Identificador da correção do sistema que deve ser desinstalada. (obrigatório)	(1) Uma "*string*" que identifica unicamente o nome da correção do sistema que é atribuído pelo fabricante. (2)Sem espaços ou caracteres restritos(< e &).

Elemento <*blockAccess*> e sub-elementos

blockAccess	*Container* para todas as partes componentes do sub-elemento <*blockAccess*> do sub-elemento <*instruction*> do sub-elemento <*info*> da mensagem de alerta. (opcional)	(1) Refere-se às portas e protocolos que devem ser bloqueados, possivelmente nos servidores que hospedam os serviços afetados ou nos *gateways*. (2) Múltiplas ocorrências são permitidas em um único bloco <*instruction*>.
serverName	Identificador do nome do servidor onde portas ou protocolos deverão ser bloqueados. (obrigatório)	(1) Uma "*string*" que identifica unicamente o nome de um servidor que é atribuído pelo fabricante. (2)Sem espaços ou caracteres restritos(< e &).
port	Número inteiro entre 0 e 65535 que indica o número da porta que deve ser bloqueada. (obrigatório)	Números de portas de acordo com a especificação da IANA (*Internet Assigned Numbers Authority*).
protocol	Código especificando o nome do protocolo associado à porta que deve ser bloqueada. (obrigatório)	Valores dos Códigos: "Tcp" – Protocolo TCP. "Udp" – Protocolo UDP.

Elemento <*disableFeature*> e sub-elementos

disableFeature	*Container* para todas as partes componentes do sub-elemento <*disableFeature*> do sub-elemento <*instruction*> do sub-elemento <*info*> da mensagem de alerta (opcional)	(1) Refere-se a um serviço ou a uma parte desse serviço que deverá ser desabilitada. (2) Múltiplas ocorrências são permitidas em um único bloco <*instruction*>.
serverName	Identificador do nome do servidor que terá um de seus serviços ou parte deles desabilitada. (obrigatório)	(1) Uma “string” que identifica unicamente o nome de um servidor que é atribuído pelo fabricante. (2)Sem espaços ou caracteres restritos(< e &).
Feature	Identificador do nome do serviço que será desabilitado. (obrigatório)	(1) Uma “*string*” que identifica unicamente o nome de um serviço ou parte dele que será desabilitada atribuída pelo fabricante. (2)Sem espaços ou caracteres restritos(< e &).
command	Texto contendo o comando e suas opções para desabilitar um serviço ou parte dele. (opcional)	
note	Texto explicativo do serviço ou parte dele a ser desabilitada e possíveis efeitos para o sistema caso essa ação seja executada. (opcional)	

Elemento <*modifyFile*> e sub-elementos

modifyFile	*Container* para todas as partes componentes do sub-elemento <*modifyFile*> do sub-elemento <*instruction*> do sub-elemento <*info*> da mensagem de alerta. (opcional)	(1) Refere-se ao caminho completo e ao nome de um arquivo que deverá ser modificado, possivelmente alterando a configuração de um serviço. (2) Múltiplas ocorrências são permitidas em um único bloco <*instruction*>.
serverName	Identificador do nome do servidor que terá um ou mais de seus arquivos modificados. (obrigatório)	(1) Uma "*string*" que identifica unicamente o nome de um servidor que é atribuído pelo fabricante. (2)Sem espaços ou caracteres restritos(< e &).
file	Nome do arquivo a ser modificado contendo o seu caminho completo. (obrigatório)	
command	Texto contendo o comando necessário para alterar um arquivo. (opcional)	Múltiplas ocorrências são permitidas em um único <*modifyFile*>.
note	Texto explicativo das alterações e suas prováveis conseqüências. (opcional)	

Elemento <*permissionChange*> e sub-elementos

permissionChange	*Container* para todas as partes componentes do sub-elemento <*permissionChange*> do sub-elemento <*instruction*> do sub-elemento <*info*> da mensagem de alerta (opcional)	(1) Refere-se a alterações de permissões de acesso a arquivos e/ou diretórios. (2) Múltiplas ocorrências são permitidas em um único <*instruction*>.
serverName	Identificador do nome do servidor que terá as permissões de acesso a um ou mais de seus arquivos e/ou diretórios modificadas. (obrigatório)	(1) Uma "*string*" que identifica unicamente o nome de um servidor que é atribuído pelo fabricante. (2)Sem espaços ou caracteres restritos(< e &).
command	Texto contendo o comando necessário para alterar as permissões de acesso a arquivos e/ou diretórios. (obrigatório)	Múltiplas ocorrências são permitidas em um único <*permissionChange*>.
note	Texto explicativo das alterações e suas prováveis conseqüências. (opcional)	

A Figura 3.8 mostra o alerta da Figura 3.5 que foi convertido para o formato XML, de acordo com a estrutura do CAP e as extensões propostas neste trabalho. O Apêndice B contém diversos exemplos de alertas emitidos pelo CERT®/CC que foram escritos de acordo com o formato proposto neste trabalho.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!--  UNIVERSIDADE FEDERAL DO MARANHÃO  -->
<!--  Projeto NIDIA  -->
<!--  A XML alert file  -->
<!--  AUTOR: Fernando Augusto Pestana Junior  -->
- <alert xmlns:cap="http://www.incident.com/cap/1.0">
    <identifier>CAN-2004-0396</identifier>
    <sender>cert@cert.org</sender>
    <sent>2004-05-19T15:37:21</sent>
    <status>Actual</status>
    <scope>Restricted</scope>
    <restriction name="CVS Feature" version="1.12.7" olderVersions="Y" />
    <restriction name="CVS Stable" version="1.11.15" olderVersions="Y" />
    <msgtype>Alert</msgtype>
  - <info>
      <category>Internet</category>
      <event>US-CERT Vulnerability Note</event>
      <urgency>Immediate</urgency>
      <severity>Severe</severity>
      <certainty>Very Likely</certainty>
      <senderName>US-CERT United States Computer Emergency Readiness Team</senderName>
      <headline>CVS contains a heap overflow in the handling of flag insertion</headline>
      <description>CVS is a source code maintenance system that is widely used by open-source
        software development projects. There is a heap memory overflow vulnerability in the
        way CVS handles the insertion of modified and unchanged flags within entry lines. When
        processing an entry line, an additional byte of memory is allocated to flag the entry as
        modified or unchanged. There is a failure to check if a byte has been previously allocated
        for the flag, which creates an off-by-one buffer overflow. By calling a vulnerable function
        several times and inserting specific characters into the entry lines, a remote attacker
        could overwrite multiple blocks of memory. The CVS server process is commonly started
        by the Internet services daemon (inetd) and run with root privileges. According to the e-
        matters security advisory, the following versions are affected: CVS feature release <=
        1.12.7 CVS stable release <= 1.11.15</description>
    - <instruction>
        <note>Apply the appropriate patch or upgrade as specified by your vendor. This issue
          has been resolved in Stable CVS Version 1.11.16 and CVS Feature Version 1.12.8.
          Until patches are available and can be applied, consider disabling the CVS
          server.Block or restrict access to the CVS server from untrusted hosts and networks.
          The CVS server typically listens on 2401/tcp, but it may use another port or
          protocol.Configure CVS server to run in a restricted (chroot) environment. Run CVS
          servers with the minimum set of privileges required on the host file system. Provide
          separate systems for development (write) and public/anonymous (read-only) CVS
          access. Host public/anonymous CVS servers on single-purpose, secured systems.
          Note that some of these workarounds will only limit the scope and impact of possible
          attacks.</note>
        <disableServer>"CVS Server"</disableServer>
        <applyPatch name="CVS Feature" version="1.12.7"
          patchUri="http://ccvs.cvshome.org/servlets/NewsItemView?newsitemID=108" />
        <applyPatch name="CVS Stable" Version="1.11.15"
          patchUri="https://ccvs.cvshome.org/servlets/NewsItemView?newsitemID=107" />
        <blockAccess serverName="CVS Server" port="2401" protocol="tcp" />
      </instruction>
      <web>http://www.kb.cert.org/vuls/id/192038</web>
      <contact>mailto:cert@cert.org?Subject=VU#192038 Feedback</contact>
    </info>
</alert>
```

Figura – 3.8. Alerta emitido pelo CERT®/CC que foi traduzido para o formato XML proposto neste trabalho.

3.5 XML *Schema* da Mensagem de Alerta CAP Estendida

O XML *Schema* descreve a estrutura do documento XML que conterá os alertas de segurança emitidos pelos CSIRTs, e que estará de acordo como formato CAP e as extensões propostas neste trabalho.

Os novos elementos encontram-se em negrito.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<schema xmlns = "http://www.w3.org/2001/XMLSchema"
      targetNamespace = "http://www.incident.com/cap/1.0"
      xmlns:cap = "http://www.incident.com/cap/1.0"
      xmlns:xs = "http://www.w3.org/2001/XMLSchema"
      elementFormDefault = "qualified"
      attributeFormDefault = "unqualified">
      <element name = "alert">
            <annotation>
                  <documentation>CAP Alert Message (version 1.0)</documentation>
            </annotation>
            <complexType>
                  <sequence>
                        <element name = "identifier" type = "string"/>
                        <element name = "sender" type = "string"/>
                        <element name = "sent" type = "dateTime"/>
                        <element name = "status">
                              <simpleType>
                                    <restriction base = "string">
                                          <enumeration value = "Actual"/>
                                          <enumeration value = "Exercise"/>
                                          <enumeration value = "System"/>
                                          <enumeration value = "Test"/>
                                    </restriction>
                              </simpleType>
                        </element>
                        <element name = "msgType">
                                    <simpleType>
                                          <restriction base = "string">
                                                <enumeration value = "Alert"/>
                                                <enumeration value = "Update"/>
                                                <enumeration value = "Cancel"/>
                                                <enumeration value = "Ack"/>
                                                <enumeration value = "Error"/>
                                          </restriction>
                                    </simpleType>
                        </element>
                        <element name = "scope" minOccurs = "0">
                                  <simpleType>
                                          <restriction base = "string">
                                                <enumeration value = "Public"/>
                                                <enumeration value = "Restricted"/>
                                                <enumeration value = "Private"/>
                                          </restriction>
                                  </simpleType>
                        </element>
```

```
<element name = "references" minOccurs = "0">
        <simpleType>
                <list itemType = "string"/>
        </simpleType>
</element>
<element name = "restriction" minOccurs = "0" maxOccurs = "unbounded >
         <complexType>
              <sequence>
                   <element name = platform  type = “string” minOccurs = "0">
                   <element name = version type = “string” minOccurs = "0">
                   <element name = name  type = “string” minOccurs = "0">
                   <element name = "olderVersions">
                        <simpleType>
                                <restriction base = "integer">
                                        <enumeration value = "1"/>
                                        <enumeration value = "0"/>
                                </restriction>
                         </simpleType>
                   </element>
              </sequence>
          </complexType>
</element>
<element name = "info" minOccurs = "0" maxOccurs = "unbounded">
      <complexType>
            <sequence>
                  <element name = "language" type = "language" default = "en-US"
                  minOccurs = "0"/>
                  <element name = "category" minOccurs = "0" maxOccurs =
                  "unbounded">
                        <simpleType>
                              <restriction base = "string">
                                    <enumeration value = "Geo"/>
                                    <enumeration value = "Met"/>
                                    <enumeration value = "Safety"/>
                                    <enumeration value = "Security"/>
                                    <enumeration value = "Rescue"/>
                                    <enumeration value = "Fire"/>
                                    <enumeration value = "Health"/>
                                    <enumeration value = "Env"/>
                                    <enumeration value = "Transport"/>
                                    <enumeration value = "Infra"/>
                                    <enumeration value = "Other"/>
                              </restriction>
                        </simpleType>
                  </element>
                  <element name = "event" type = "string"/>
                  <element name = "urgency">
                        <simpleType>
                              <restriction base = "string">
                                    <enumeration value = "Immediate"/>
                                    <enumeration value = "Expected"/>
                                    <enumeration value = "Future"/>
                                    <enumeration value = "Past"/>
                                    <enumeration value = "Unknown"/>
                              </restriction>
                        </simpleType>
                  </element>
```

```
<element name = "severity">
        <simpleType>
                <restriction base = "string">
                        <enumeration value = "Extreme"/>
                        <enumeration value = "Severe"/>
                        <enumeration value = "Moderate"/>
                        <enumeration value = "Minor"/>
                        <enumeration value = "Unknown"/>
                </restriction>
        </simpleType>
</element>
<element name = "certainty">
        <simpleType>
                <restriction base = "string">
                        <enumeration value = "Very Likely"/>
                        <enumeration value = "Likely"/>
                        <enumeration value = "Possible"/>
                        <enumeration value = "Unlikely"/>
                        <enumeration value = "Unknown"/>
                </restriction>
        </simpleType>
</element>
<element name = "senderName" type = "string" minOccurs = "0"/>
<element name = "headline" type = "string" minOccurs = "0"/>
<element name = "description" type = "string" minOccurs = "0"/>
<element name = "web" type = "anyURI" minOccurs = "0"/>
<element name = "contact" type = "string" minOccurs = "0"/>
<element name = "instruction" minOccurs = "0">
    <complexType>
        <sequence>
            <element name = "note" type = "string"/>
            <element name = "disableServer" minOccurs = "0"
            maxOccurs = "unbounded">
                <simpleType>
                    <list itemType = "string" />
                </simpleType>
            </element>
            <element name = "applyPatch" minOccurs = "0"
            maxOccurs = "unbounded">
                <complexType>
                    <sequence>
                        <element name = "platform" type =
                        "string" minOccurs = "0"/>
                        <element name = "version" type =
                        "string" minOccurs = "0"/>
                        <element name = "name" type = "string"
                        minOccurs = "0"/>
                        <element name = "patchUri" type = "
                        anyURI " minOccurs = "0"/>
                    </sequence>
                </complextype>
            </element>
```

```
<element name = "removePatch" minOccurs = "0"
maxOccurs = "unbounded">
    <complexType>
        <sequence>
            <element name = "platform" type =
            "string" minOccurs = "0"/>
            <element name = "version" type =
            "string" minOccurs = "0"/>
            <element name = "name" type = "string"
            minOccurs = "0"/>
            <element name = "patchName" type =
            "string" minOccurs = "0"/>
         </sequence>
    </complextype>
</element>
<element name = "blockAccess” minOccurs = "0"
maxOccurs = "unbounded">
    <complexType>
        <sequence>
            <element name = "serverName" type =
            "string"  minOccurs = "0"/>
            <element name = "port" type =
            "integer"/>
            <element name = "protocol">
                <simpleType>
                    <restriction base = "string">
                        <enumeration value
                        = "TCP”>
                        <enumeration value
                        = "UDP"/>
                    </restriction>
                </simpleType>
            </element>
         </sequence>
    </complextype>
</element>
<element name = "disableFeature” minOccurs = "0"
maxOccurs = "unbounded">
    <complexType>
        <sequence>
            <element name = "serverName" type =
            "string"/>
            <element name = "feature" type =
            "string"/>
            <element name = "command" type =
            "string" minOccurs=”0”  maxOccurs =
            "unbounded"/>
            <element name = "note" type = "string"/>
          </sequence>
    </complextype>
</element>
```

```
                                        <element name = "modifyFile” minOccurs = "0"
                                        maxOccurs = "unbounded">
                                            <complexType>
                                                <sequence>
                                                    <element name = "serverName" type =
                                                    "string"/>
                                                    <element name = "file" type = "string"/>
                                                    <element name = "command" type =
                                                    "string" minOccurs = "0" maxOccurs =
                                                    "unbounded"/>
                                                    <element name = "note" type = "string"
                                                    minOccurs = "0" />
                                                </sequence>
                                            </complextype>
                                        </element>
                                        <element name = "permissionChange” minOccurs =
                                        "0" maxOccurs = "unbounded">
                                            <complexType>
                                                <sequence>
                                                    <element name = "serverName" type =
                                                    "string"/>
                                                    <element name = "command" type =
                                                    "string" maxOccurs = "unbounded"/>
                                                    <element name = "note" type = "string"
                                                    minOccurs = "0" />
                                                </sequence>
                                            </complextype>
                                        </element>
                                    </sequence>
                                </complexType>
                            </element>
                        </sequence>
                    </complexType>
                </element>
            </sequence>
        </complexType>
    </element>
</schema>
```

3.6 Conclusões

Apresentou-se neste capítulo uma proposta para o compartilhamento de informações entre os SDIs e os CSIRTs por meio de *Web services* e da XML. Esse compartilhamento teve como objetivo a atualização automática do conjunto de ações de respostas dos SDIs.

Após análise de diversos alertas publicados pelo CERT®/CC identificou-se 06 (seis) elementos para representar as ações sugeridas nesses alertas, os quais

foram adicionados ao CAP, obtendo-se um formato de mensagens de alertas que pode ser utilizado para problemas relacionados à segurança dos computadores e que possibilite a atualização automática do conjunto de ações de respostas dos SDIs.

4. ARQUITETURA MULTIAGENTE PARA ATUALIZAÇÃO AUTOMÁTICA DO MECANISMO DE DETECÇÃO DE ATAQUES DOS SDIs MULTIAGENTE.

No Capítulo 3, apresentou-se uma proposta para a atualização automática do conjunto de ações de respostas dos SDIs, utilizando-se as informações dos alertas de segurança emitidos pelos CSIRTs. Entretanto para que o SDI seja capaz de responder a novas ameaças é necessário que o seu mecanismo de detecção de ataques também esteja atualizado.

Neste capítulo, apresenta-se uma proposta de arquitetura multiagente que será responsável por coletar informações sobre ataques de rede contidas em diversas bases de dados na Internet e com base nessas informações, ela irá atualizar o mecanismo de detecção de ataques dos SDIs multiagente [36].

Na figura 4.1 observa-se o Modulo de Compartilhamento de Informações comunicando-se com os CSIRTs em busca de alertas de segurança tendo como objetivo atualizar a Base de Dados de Ações de Respostas dos SDIs, conforme apresentado no Capítulo 3.

Também na Figura 4.1, tem-se a proposta deste capítulo, uma arquitetura multiagente para atualização automática do mecanismo de detecção de ataques dos SDIs multiagente. Essa arquitetura está representada por uma Agência Central de Segurança (ACS) e será detalhada na seção 4.1.

A ACS produzirá uma minisociedade de agentes (Sociedade Atualizada de Agentes de Reconhecimento de Assinaturas - SAARA) [23] responsável por desempenhar o papel do mecanismo de detecção de ataques para SDIs multiagente.

A Figura 4.1 mostra a interação entre as fontes de dados de ataques da Internet, a ACS e os SDIs multiagente. A comunicação entre essas entidades, por meio da Internet, é feita utilizando-se *Web services* e XML de forma semelhante ao proposto, no capítulo 3, para o modelo de compartilhamento de informações entre CSIRTs e SDIs.

Como potenciais entidades participantes deste modelo, tem-se a base de dados da *Defense Advanced Research Projects Agency* (DARPA) [29] e o SDI multiagente NIDIA que será apresentado no Capítulo 5.

A ACS enviará requisições ao *Web service* das fontes de dados de ataques em busca de novos arquivos e os SDIs multiagente enviarão requisições a ACS em busca de novas SAARAs.

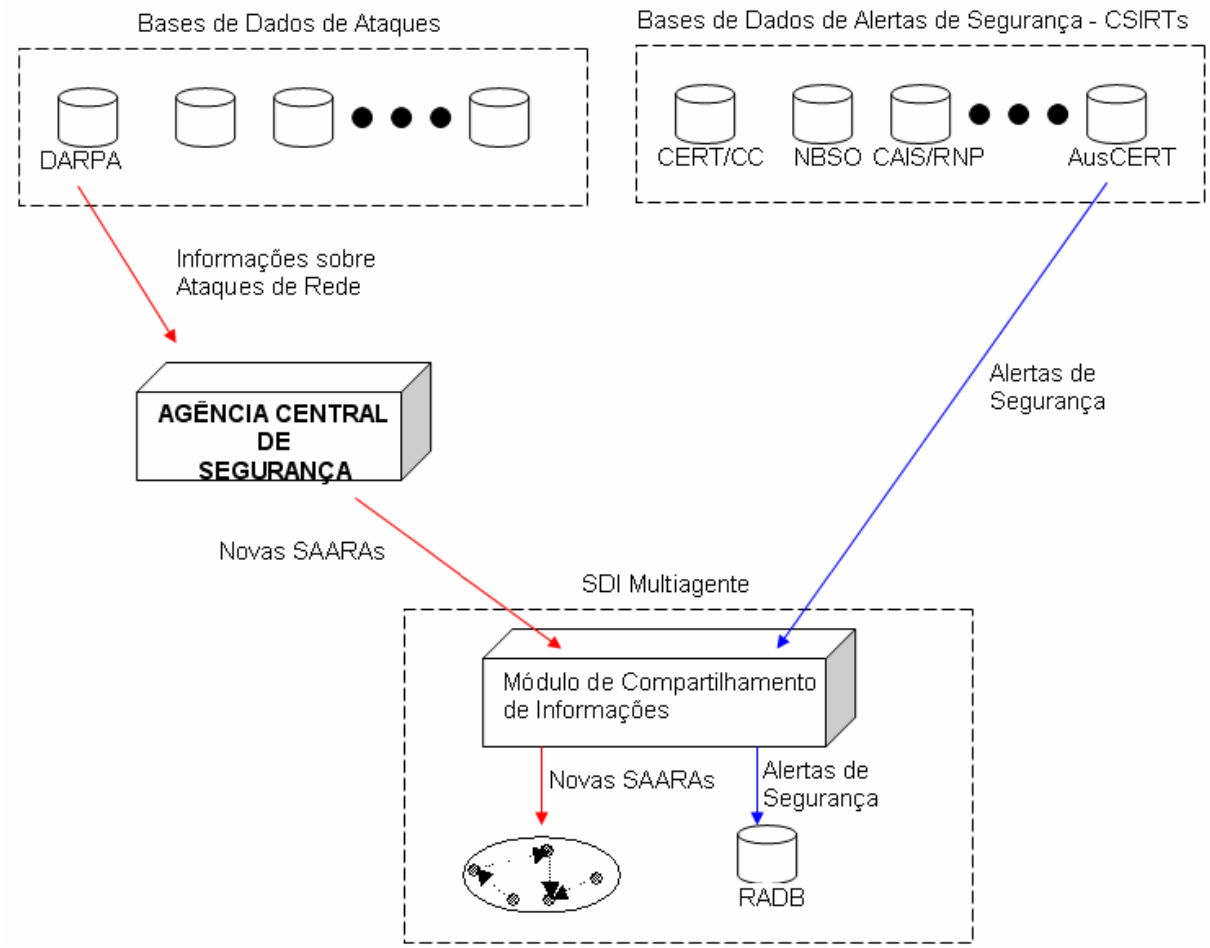


Figura – 4.1. Interação entre ACS, fontes de dados e SDIs multiagente.

Como a ACS deverá coletar informações sobre ataques contidas em diversas bases de dados na Internet, sugere-se que essas informações estejam contidas em arquivos no formato padrão *libpcap* que é uma *Application Programming Interface* (API) usada para capturar tráfego de redes. Devido ao seu amplo uso, é importante reconhecê-la como um padrão. Diversos sistemas de detecção de intrusão baseados em rede, *sniffers* de rede e ferramentas de processamento de dados de rede podem operar em conjunto simplesmente por utilizarem o formato de arquivo *libpcap* [37].

4.1 A Agência Central de Segurança (ACS)

A ACS estará monitorando continuamente as bases de dados de ataques à procura de novos arquivos. Assim que um novo arquivo for disponibilizado ele será recuperado pela ACS que irá atualizar a sua base de dados de assinaturas de ataques e produzir uma nova SAARA.

Os SDIs multiagente deverão monitorar constantemente a ACS em busca de novas SAARAs garantindo uma atualização constante e de forma automática.

Desta forma, existe uma necessidade de informação a longo prazo por parte dos SDIs multiagente, isto é, a necessidade de obter novas SAARAs. Na proposta apresentada neste capítulo, essa necessidade dos SDIs multiagente foi suprida utilizando-se, com algumas alterações, o *framework* para a filtragem de informação personalizada, descrito por OLIVEIRA [33].

Esse *framework* é baseado na tecnologia de agentes e fornece soluções para organizá-los de maneira que possam satisfazer as necessidades dos SDIs multiagente. O padrão Camadas multiagente para sistemas adaptativos/adaptáveis [42] foi utilizado para a construção do *framework* e foram considerados apenas os

agentes que fazem parte da filtragem de informações, ou seja, os agentes que colaboram para satisfazer a necessidade de informação a longo prazo dos SDIs multiagente por novas SAARAs.

A ACS será uma sociedade de agentes inteligentes composta por cinco agentes: Interface, Gerador de Conexões, Surrogate, Construtor de SAARA e Monitor.

Os agentes foram agrupados em camadas de acordo com suas funcionalidades, como mostra a Figura 4.2:

- Camada de Interação com o Usuário: camada que faz a intermediação entre a ACS e seus usuários (SDIs Multiagente), recebendo solicitações e fornecendo novas SAARAs. Ela é composta pelo Agente Interface.

- Camada de Tratamento e Armazenamento de Informação: camada que engloba o tratamento de informações para construção e armazenamento de novas SAARAs. Ela é composta pelos agentes Gerador de Conexões, Surrogate e Construtor de SAARA. Ela envia para a camada de interação com usuário informações sobre novas SAARAs produzidas.

- Camada de Interação com as Fontes de Informação: camada que faz a ligação do sistema com as fontes de informação da Internet, capturando novos arquivos de informações de ataques de rede. Ela é composta pelo Agente Monitor que monitora e envia para a camada de tratamento e armazenamento de informação os arquivos atualizados capturados das fontes de informação.

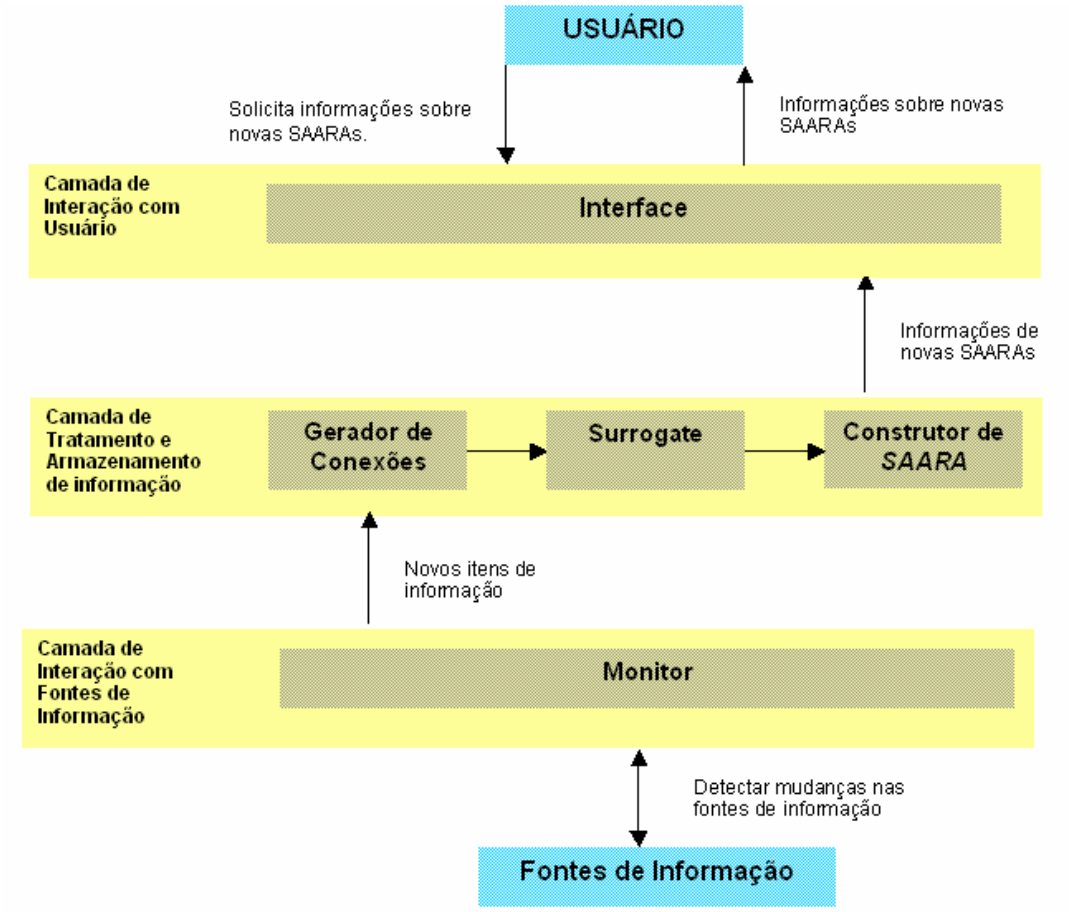


Figura – 4.2. Visão em camadas do modelo.

4.2 Agentes do Projeto

4.2.1 Agente Interface

O Agente Interface é um agente reativo [39] que serve como intermediador entre os usuários e o resto do sistema. Ele tem como função receber requisições dos SDIs multiagente e informá-los sobre a existência de novas SAARAs.

O SDI multiagente, por intermédio do seu módulo de compartilhamento de informações, envia uma requisição à ACS em busca de novas SAARAs. Caso uma nova SAARA esteja disponível ele poderá recuperá-la, atualizando de forma automática o seu mecanismo de reconhecimento de ataques. A Figura 4.3 mostra a interação entre os SDIs multiagente e a ACS.

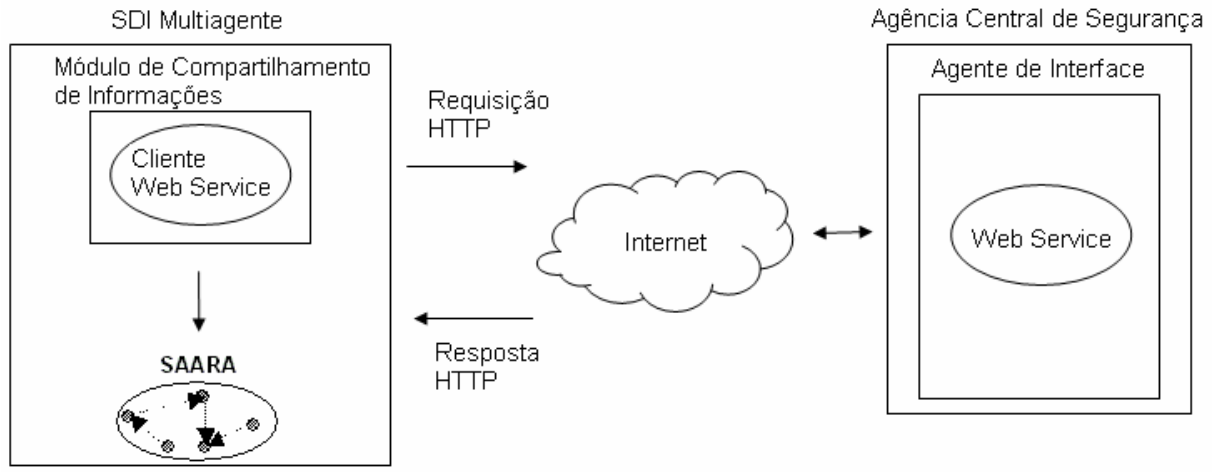


Figura – 4.3. Interação entre os SDIs multiagente e a ACS.

Após o processo de criação de uma nova SAARA, o Agente Interface recebe as informações desta nova SAARA do Agente Construtor de SAARA e as disponibiliza por meio de um *Web service* que poderá receber as requisições dos SDIs multiagente.

A Tabela 4.1 mostra detalhadamente como o Agente Interface reage aos eventos do sistema.

Tabela - 4.1. Reações do Agente Interface a eventos do sistema.

Evento	Reação do Agente Interface
Usuário envia uma requisição em busca de novas SAARAs.	Processa a requisição e caso haja uma nova SAARA disponível envia as informações necessárias para que o usuário possa recuperá-la.
Agente Interface recebe Informações de novas SAARAs do Agente Construtor de SAARA.	Atualiza as suas informações sobre uma nova SAARA, a versão e a URL onde ela pode ser recuperada pelos SDIs multiagente.

4.2.2 Agente Monitor

O Agente Monitor é um agente do tipo reativo, ou seja, age em função das mudanças na fonte de informação, sendo que esta é limitada ao consórcio de instituições responsáveis por manter diferentes bases de dados de ataques de rede. É de extrema importância que essas fontes sejam reconhecidamente confiáveis, pois

a partir de suas informações serão construídos novos mecanismos de reconhecimento de assinaturas de ataques.

Enviando constantemente requisições em busca de novos arquivos de tráfego de rede, o Agente Monitor agirá sempre que um novo arquivo for disponibilizado, recuperando-o e enviando-o ao Agente Gerador de Conexões.

A Figura 4.4 mostra um exemplo de arquivo de ataque de rede, onde se observa um datagrama IP de uma conexão entre as máquinas identificadas pelos endereços IP de origem 206.186.80.111 e de destino 172.16.112.50.

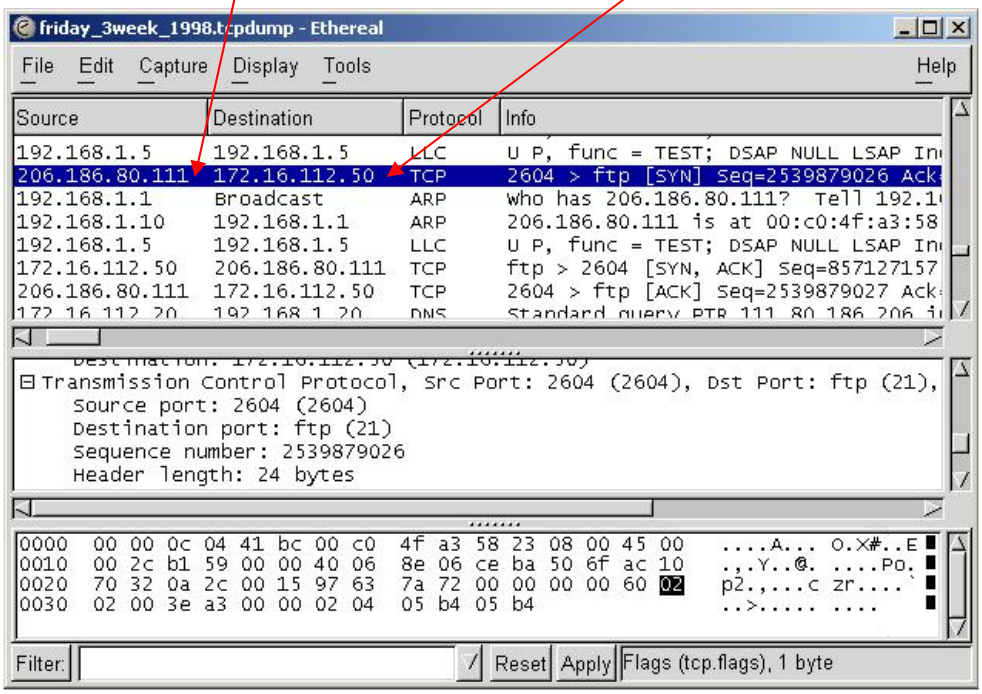


Figura – 4.4. Arquivo de tráfego de rede.

A Figura 4.5 mostra a interação entre a ACS e as bases de dados de tráfego de rede.

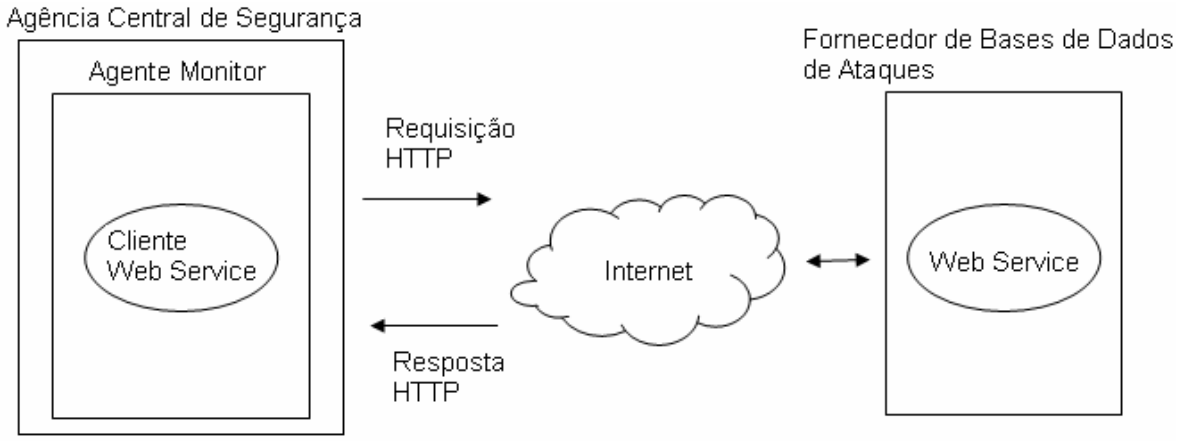


Figura – 4.5. Interação entre a ACS e as bases de dados de tráfego de rede.

4.2.3 Agente Gerador de Conexões

Esse agente será responsável por construir arquivos texto que serão o espelho das conexões presentes nos arquivos de ataques de rede assim como arquivos de índice para conexões de ataque e arquivos de índice para conexões normais.

Um arquivo espelho de conexão é um arquivo texto com o conteúdo da parte de dados dessa conexão. Na Figura 4.6 encontra-se o arquivo texto de nome 172.16.112.50_206.186.80.111_21_2604_1.txt que é o arquivo espelho da conexão FTP, no sentido servidor-cliente, entre as máquinas de origem 206.186.80.111 (cliente) e de destino 172.16.112.50 (servidor). Pode-se ver no conteúdo do arquivo 172.16.112.50_206.186.80.111_21_2604_1.txt as respostas enviadas pelo servidor ao cliente.

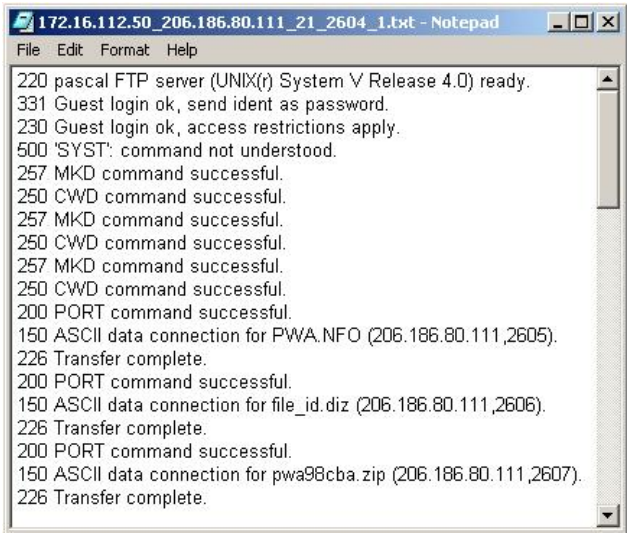


Figura – 4.6. Arquivo espelho de uma conexão FTP.

Para que uma nova SAARA seja construída e a base de dados de assinaturas de ataques seja atualizada é necessário identificar as conexões consideradas como normais e as conexões que contém ataques. Isso será feito por meio da criação de um arquivo de índice para conexões normais e outro para as conexões com ataques. Cada linha desses arquivos contém o nome do arquivo espelho da conexão, um indicador binário de presença de ataque e o nome do ataque, separados por um espaço em branco.

O indicador de presença de ataque será igual a 1 (um) se houver ataque na conexão e igual a 0 (zero) caso contrário.

A Figura 4.7 mostra um arquivo de índice para conexões de ataque.

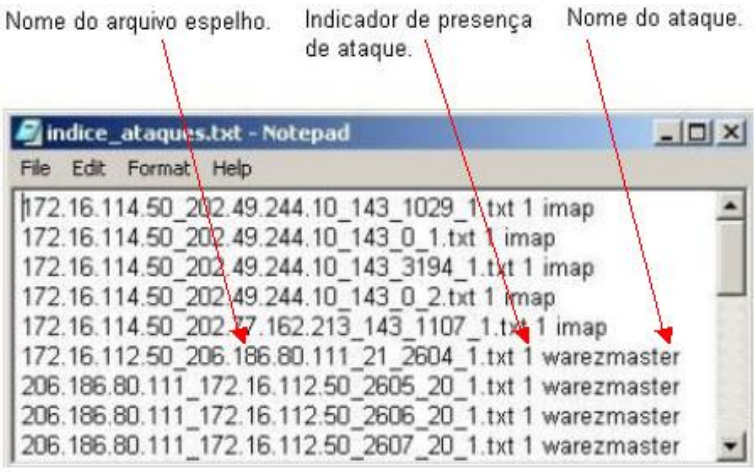


Figura – 4.7. Exemplo de arquivo de índice para conexões de ataque.

4.2.4 Agente Surrogate

Esse agente tem como responsabilidade criar representações dos arquivos texto que são o espelho de conexões de ataques e de conexões normais. Essas representações são obtidas utilizando-se as técnicas do campo da Recuperação de Informação.

No contexto da recuperação da informação, *surrogate* [8] é uma forma como são representados internamente ao sistema os elementos de informação, que no caso deste trabalho são os arquivos espelho de conexões.

O modelo de recuperação de informação escolhido para ser utilizado no projeto do Agente Surrogate é o modelo do espaço vetorial [40]. De acordo com esse modelo, os arquivos espelho de conexões serão representados por vetores de palavras-chave.

O conjunto de palavras-chave a ser utilizado poderá ser obtido utilizando-se uma rede neural PCA (*Principal Component Analysis*) [25] devido à sua característica inerente de extrair quais são os principais elementos que representam as peculiaridades em um determinado domínio [23].

Como existe o arquivo de índice que relaciona uma conexão de rede a um determinado ataque, a base de dados de assinaturas de ataques será constituída pelo nome do ataque e a representação interna do arquivo espelho para a respectiva conexão.

Na Figura 4.8 apresenta-se um exemplo de como as informações relativas ao ataque *warezmaster*, presente na conexão entre as máquinas identificados pelos endereços IP de origem 206.186.80.111 e de destino 172.16.112.50 (Figura 4.4) foram inseridas na base de dados de assinaturas de ataques. Nesse exemplo, o arquivo espelho da conexão, mostrado na Figura 4.6, foi representado por um vetor de palavras-chave, em que cada posição desse vetor indica o número de ocorrências de uma determinada palavra-chave no arquivo espelho da conexão.

O ataque *warezmaster* ocorre por meio de uma conexão anônima FTP copiado-se *Warez* (cópias ilegais de *software* protegido por direitos autorais) para um servidor FTP.

Nome do ataque	cat\s*>	/cgi-bin/phf	guest\: login\s	chmod\s	...
warezmaster	0	0	2	0	

Figura – 4.8. Exemplo da Base de Dados de Assinaturas de Ataques.

Após a atualização da base de dados de assinaturas de ataque o Agente Surrogate comunica-se com o Agente Construtor de SAARA enviando informações sobre os novos arquivos criados.

4.2.5 Agente Construtor de SAARA

O Agente Construtor de SAARA tem a responsabilidade de construir novas SAARAs de acordo com a metodologia descrita por DIAS & NASCIMENTO [23] na qual os vetores de contagem de palavras-chave, tanto de conexões suspeitas quanto de conexões normais, serão utilizados para o treinamento

supervisionado de uma rede neural MLP (*Multilayer Perceptron*) [25] que será responsável por emitir alertas indicando o grau de suspeita de determinada conexão.

Após a fase de treinamento, a rede neural é serializada para compor a SAARA que será enviada a um cliente. O processo de serialização de um objeto permite gravá-lo e assim salvar os seus atributos. No caso da rede neural é possível salvá-la após a fase de treinamento para então utilizá-la posteriormente.

O Construtor de SAARA se relaciona com o Agente de Interface a fim de informar a criação de novas SAARAs e com o Agente Surrogate com o objetivo de capturar novos itens de informação para a construção de novas SAARAs.

4.3 Responsabilidades, Atividades e Protocolos dos Agentes

A Tabela 4.2 mostra a responsabilidade, as atividades e os protocolos de interação de cada um destes agentes.

Tabela - 4.2. Especificação dos Agentes.

Agente	Responsabilidade	Atividade	Protocolo
Interface	Intermediar a interação usuário-sistema.	▪ Receber requisições dos usuários; ▪ Receber informações de novas SAARAs.	▪ Informar dados de novas SAARAs aos SDIs multiagente.
Gerador de Conexões	Gerar Arquivos de Conexões.	▪ Receber novos Arquivos de tráfego de rede do Agente Monitor; ▪ Criar arquivos de conexões de ataques e arquivos de conexões normais; ▪ Criar arquivo de índice de conexões de ataques e arquivo de índice de conexões normais.	▪ Enviar ao Agente Surrogate informações sobre novos arquivos produzidos.
Surrogate	Construir representação interna dos arquivos.	▪ Receber informações sobre novos arquivos do Agente Gerador de Conexões;	▪ Enviar Informações de novas representações

		▪ Construir a representação interna dos arquivos de conexões de ataques e dos arquivos de conexões normais; ▪ Atualizar a base de dados de assinaturas de ataques com a representação interna dos arquivos de conexões de ataques.	internas dos arquivos de conexões de ataques e dos arquivos de conexões normais ao Agente Construtor de SAARA.
Construtor de *SAARA*	Construir *novas SAARAs.*	▪ Receber novos itens de informação do Agente Surrogate; ▪ Construir novas SAARAs; ▪ Informar ao Agente Interface a existência de nova SAARA.	▪ Enviar informações de novas SAARAs ao Agente Interface.
Monitor	Monitorar fontes de informação dinâmicas.	▪ Detectar mudanças nas fontes de informação dinâmicas; ▪ Recuperar arquivos de tráfego de rede.	▪ Enviar novos itens de informação ao Agente Gerador de Conexões.

As entidades externas que interagem com o sistema são: usuário (SDIs multiagente), que utiliza o sistema para satisfazer suas necessidades de informação a longo prazo e as fontes de informação na Internet.

4.4 Funcionamento do Sistema

A Figura 4.9 mostra o diagrama de seqüência em notação UML [11] dos agentes como uma forma de auxílio para a sua explicação.

O Agente Monitor envia constantemente requisições às bases de dados de arquivos de ataques em busca de novos itens de informação. Quando ele recebe a informação de que um novo arquivo está disponível, ele o recupera e o envia ao Agente Gerador de Conexões. Essa comunicação será feita por meio da Internet e utilizando-se *Web services*.

O Agente Gerador de Conexões irá criar arquivos texto com o conteúdo das conexões de ataques, assim como um arquivo de índice correspondente. Da mesma forma serão criados arquivos para as conexões normais e o arquivo de índice. Após a criação destes arquivos, o Agente Gerador de Conexões informa ao Agente Surrogate sobre a disponibilidade destes novos arquivos.

O Agente Surrogate cria representações internas dos arquivos de conexões, atualiza a base de dados de assinaturas de ataques com essas representações e informa ao Agente Construtor de SAARA que novas representações internas foram produzidas.

O Agente Construtor de SAARA cria então uma nova SAARA que contém as informações atualizadas sobre ataques. Após a criação da nova SAARA, o Agente Interface é informado e passa a disponibilizar informações atualizadas aos SDIs multiagente de como obter essa nova SAARA.

Os SDIs multiagente periodicamente enviam requisições ao Agente Interface em busca de novas SAARAs e assim que recebam uma resposta positiva, recuperam a nova SAARA produzida atualizando de forma automática o seu mecanismo de reconhecimento de ataque. A comunicação entre os SDIs e o Agente Interface também será estabelecida por meio da Internet e utilizando-se *Web services*.

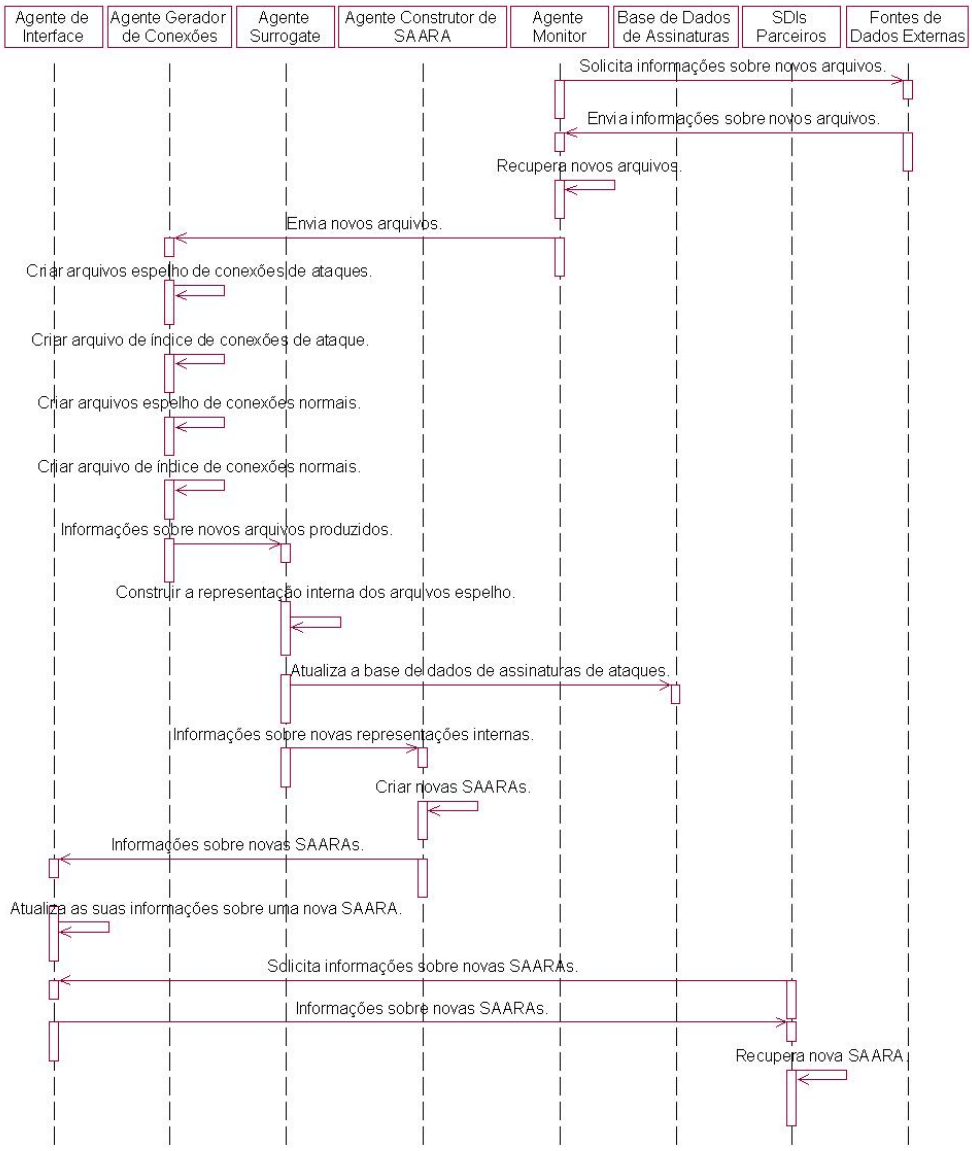


Figura – 4.9. Funcionamento do Sistema

4.5 Conclusões

Para que um SDI possa responder a uma intrusão, além de ter o seu conjunto de ações de respostas atualizado, é necessário que o seu mecanismo de reconhecimento de ataques também esteja ciente das novas ameaças ou de variações das existentes.

Neste capítulo, apresentou-se uma proposta de arquitetura multiagente, baseada em um framework para a filtragem e recuperação de informação, que pode interagir com fontes de dados de ataques de redes da Internet por meio de *Web services*. Essa arquitetura é responsável pela produção de uma sociedade de atualizada de agentes de reconhecimento que desempenha o papel do mecanismo de reconhecimento de ataques para SDIs multiagente.

Desta forma, para um SDI multiagente manter o seu mecanismo de detecção de ataques atualizado bastaria que ele recuperasse e utilizasse uma nova SAARA.

5. APLICAÇÃO DO MODELO DE COMPARTILHAMENTO DE INFORMAÇÕES ENTRE CSIRTs E SDIs AO PROJETO NIDIA

Como o modelo apresentado no capítulo 3 para a atualização do conjunto de ações de respostas constitui-se de uma proposta genérica de mecanismo de atualização para SDIs descreve-se, nesta seção, como ele foi aplicado ao modelo de SDI multiagente NIDIA.

5.1 O Projeto NIDIA

A arquitetura do NIDIA é mostrada na Figura 5.1.

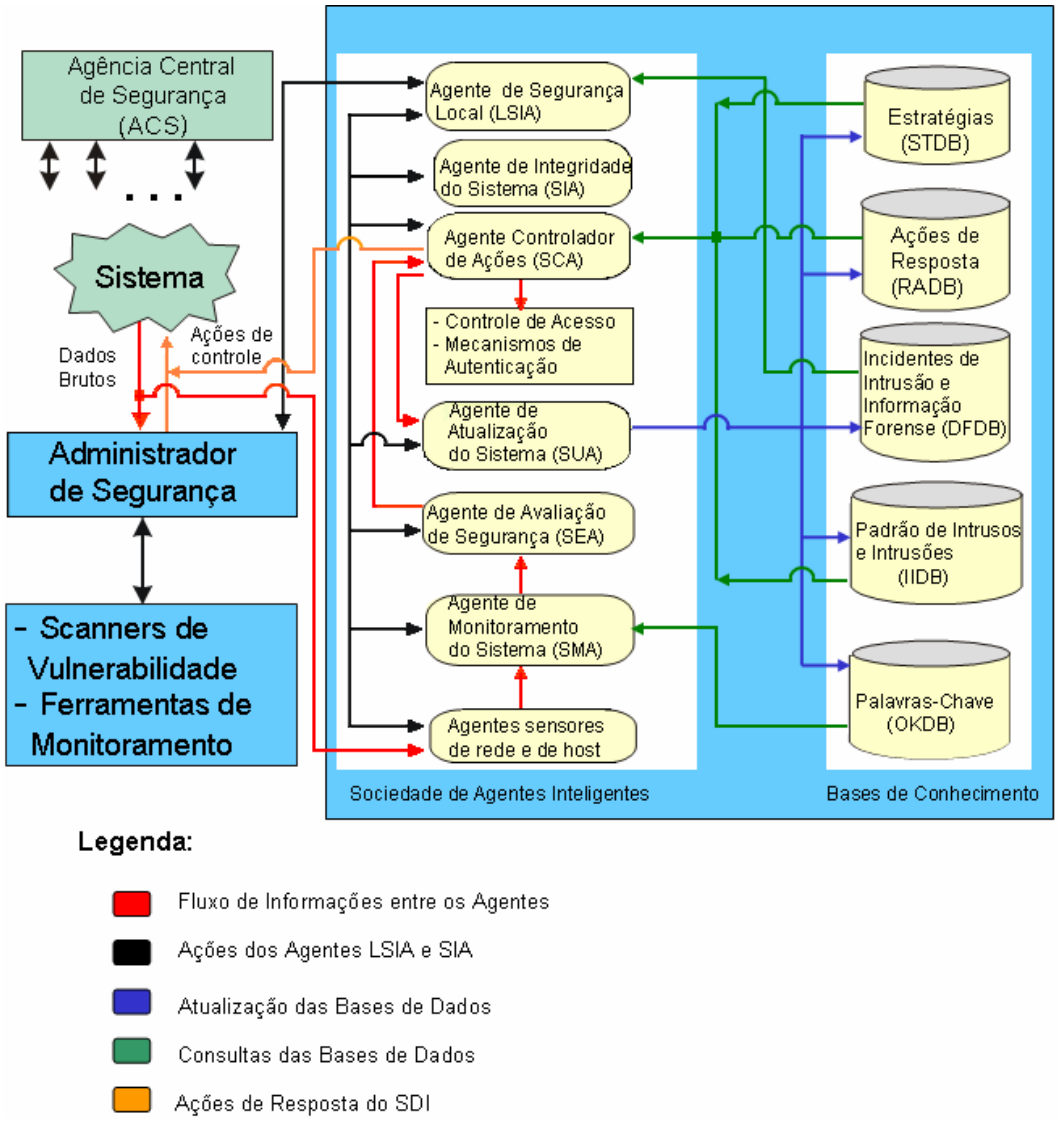


Figura – 5.1. Arquitetura do NIDIA.

- Agente de Monitoramento de Sistema (*System Monitoring Agent* - SMA): é responsável por organizar e formatar os eventos ou conjunto de eventos coletados, de forma que possam ser identificados padrões de ataques e comportamentos anormais na rede e nos servidores monitorados, de acordo com a base de dados otimizada de palavras-chave (OKDB);
- Agente de Avaliação de Segurança do Sistema (*System Security Evaluation Agent* - SEA): é o agente responsável por verificar se o evento corresponde, de fato, a uma tentativa de invasão ou trata-se apenas de um falso positivo. Como saída, esse agente gera um nível de alerta associado à severidade do evento;
- Agente de Atualização do Sistema (*System Update Agent* - SUA): é responsável por manter atualizadas as bases de dados de incidentes de intrusão (DFDB), assinaturas de intrusões (IIDB), ações de resposta (RADB) e estratégias (STDB);
- Agente Controlador de Ações (*System Controller Agent* - SCA): esse agente é responsável pelo controle das ações que o sistema deve tomar em caso de uma tentativa de invasão. Para a tomada de decisão, são utilizadas as bases de dados de estratégia (STDB) e ações (RADB);
- Agente de Integridade do Sistema (*Self-Integrity Agent* - SIA): esse agente é responsável por garantir a integridade do SDI. O Agente de Integridade busca por eventos não esperados ou diferentes do perfil normal dos agentes ativos do sistema;

- Agente de Segurança Local (*Local Security Intelligent Agent* - LSIA): esse agente é responsável pelo gerenciamento da sociedade de agentes e pela interface entre o SDI com o administrador de segurança. É por intermédio desse agente que o administrador gerencia o estado e a configuração dos agentes, a atualização da base de dados de estratégias (STDB), o registro das ocorrências detectadas e as ações tomadas pelo sistema;
- Agente Administrador de Segurança: é o agente humano que interage com o SDI com o auxílio do Agente de Segurança Local para realizar diversas tarefas, tais como: configurar a política de segurança do ambiente computacional (base de dados de estratégias), ativar e desativar agentes, gerenciar o estado e a configuração do sistema, além de conhecer a situação atual do ambiente monitorado.

O NIDIA dispõe de cinco repositórios para armazenar as informações relevantes à detecção de intrusão:

- A Base de Dados de Estratégias (STDB) é responsável por registrar as estratégias adotadas por uma organização qualquer em relação à sua política de segurança. Ela é importante para garantir a adaptabilidade do SDI aos mais diversos casos;
- Na Base de Dados de Ações de Respostas (RADB) estão contidas as informações referentes ao conjunto de ações de resposta a intrusões que devem ser tomadas de acordo com a severidade do ataque detectado;
- A Base de Dados de Intrusos e Intrusões (IIDB) guarda as assinaturas de intrusão que serão utilizadas para a detecção de atividades

suspeitas. Ela deve ser constantemente atualizada para garantir que novas técnicas de ataque possam ser detectadas;

- A Base de Dados de Palavras-Chave (OKDB) armazena as palavras-chave que serão utilizadas na formatação de informações recebidas dos sensores;
- E por fim, tem-se a Base de Dados de Incidentes de Intrusão e Informação Forense (DFDB), que registra os danos causados por ataques bem-sucedidos, tentativas de ataques e as ações de resposta que foram tomadas pelo NIDIA.

O funcionamento do NIDIA é explicado a seguir com auxílio de diagramas de colaboração, em notação UML.

Quando o sistema é iniciado, o administrador informa ao SDI, utilizando o LSIA, quantos e quais agentes sensores (de rede ou de máquina) serão ativados, tomando como base a arquitetura de rede da organização. O LSIA identifica e ativa os agentes sensores, recebe uma confirmação de ativação dos agentes e informa ao administrador de segurança quais agentes foram ativados.

Os agentes sensores de máquina fazem a leitura dos logs de segurança de servidores específicos e repassam para o Agente SMA de máquina para que o mesmo formate os dados. Posteriormente tais informações são enviadas para o Agente SEA de máquina para que seja feita a avaliação dos dados coletados, que foram previamente formatados.

Os agentes sensores de rede capturam datagramas IP e realizam duas tarefas: a) enviam o cabeçalho desses datagramas para os agentes SMA de rede específicos para formatar esse tipo de informação e b) constroem as conexões TCP entre servidor-cliente a partir do conteúdo dos datagramas recolhidos e as enviam

para os agentes SMA de rede específicos para formatar esse tipo de informação. A Figura 5.2 apresenta o diagrama de colaboração do Agente Sensor de Rede em notação UML.

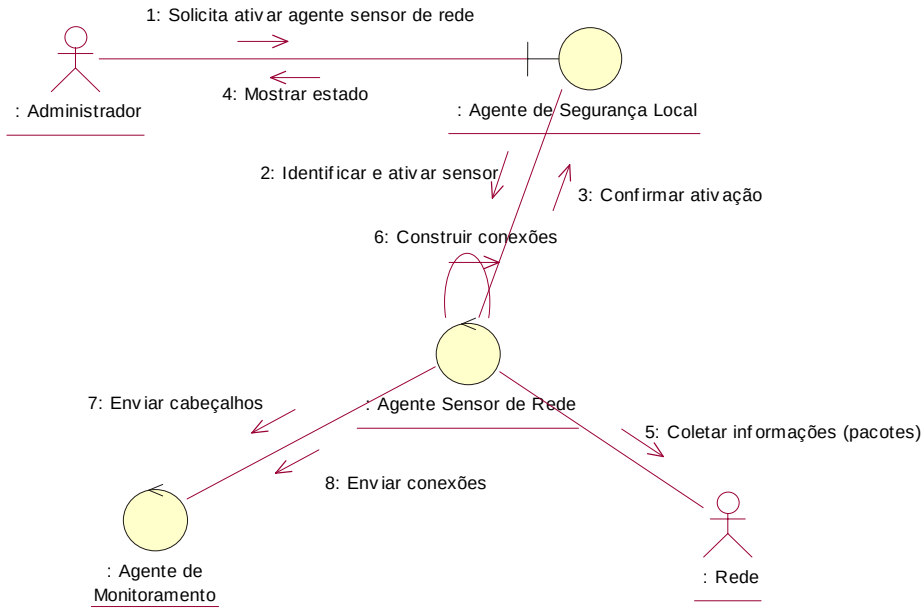


Figura – 5.2. Diagrama de colaboração do Agente Sensor de Rede (notação UML).

Os agentes SMA realizam a tarefa de formatar os dados recebidos. No caso específico do tratamento das conexões montadas pelo agente sensor de rede o agente SMA especializado consulta uma base de dados de palavras-chaves para a contagem de ocorrência das mesmas. O diagrama de colaboração, em notação UML, para o SMA encontra-se na Figura 5.3.

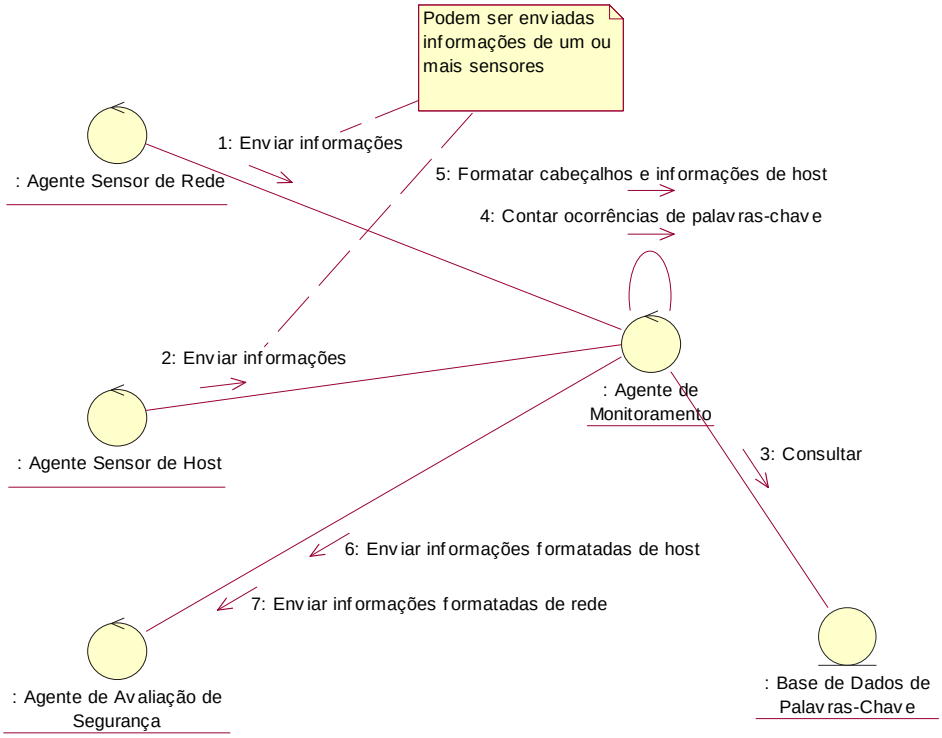


Figura – 5.3. Diagrama de colaboração do Agente de Monitoramento - SMA (notação UML).

Posteriormente os agentes SMA enviam os dados pré-processados para os agentes SEA avaliarem. Desta forma, existem agentes SEA especializados em tratar o cabeçalho dos pacotes por meio de filtros, ou possivelmente algum sistema baseado em regras, e existem agentes SEA especializados em inspecionar o conteúdo das conexões TCP.

O produto gerado pelos agentes SEA é o grau de suspeita dos dados verificados que é enviado juntamente com as informações formatadas de rede *e de logs* de servidores e para o SCA para tomar alguma contramedida. Na Figura 5.4 encontra-se o diagrama de colaboração do Agente de Avaliação de Segurança – SEA em notação UML.

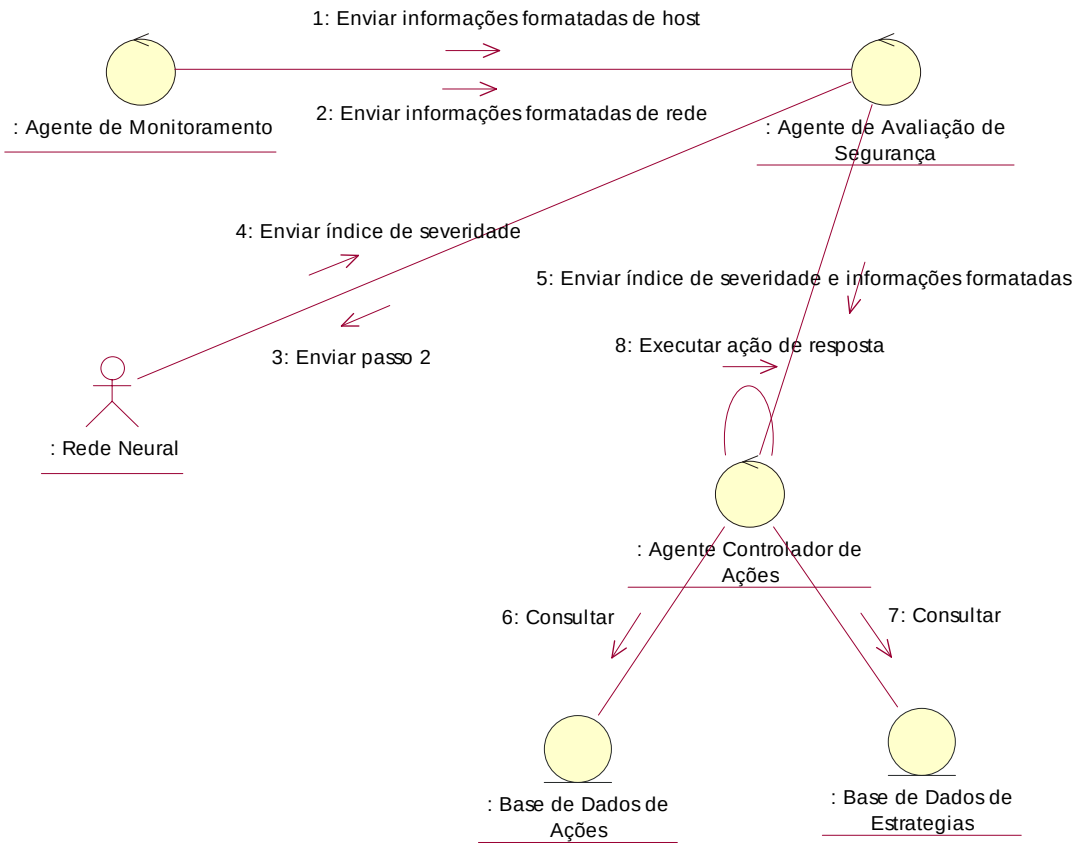


Figura – 5.4. Diagrama de colaboração do Agente de Avaliação de Segurança – SEA (notação UML).

O Agente Controlador de Ações (SCA) foi detalhado por SANTOS & NASCIMENTO [41], sendo composto por uma sociedade de agentes de acordo com a Figura 5.5.

A seguir estão relacionados os agentes e suas atribuições:

- Agente Controlador Principal (MCA): é responsável pelo controle e monitoramento das atividades dos outros agentes fornecendo as condições necessárias para uma resposta automática. Ele recebe do SEA o índice de severidade e as informações formatadas de rede e de *logs* de servidores;
- Agente BAM (BA): é responsável pela identificação do tipo de ataque. Ele recebe do MCA as informações formatadas de rede e de *logs* de

servidores e, identifica o tipo de ataque por meio de uma rede neural BAM (*Binary Association Memory*), e retorna o resultado ao MCA;

- Agente de Avaliação de Severidade (SAA): A função desse agente é medir o nível de perigo que uma intrusão está oferecendo para o sistema. O SEA, por meio de uma rede neural MLP, gera um grau de severidade, entre 0 e 1, e quanto mais próximo de 1 maior a suspeita de ataque. A partir deste índice, que é enviado pelo MCA, o SAA irá gerar um nível fuzzy de segurança indicando o estado do sistema (normal, alerta ou emergência);
- Agente *Honey Net* (HNA): após a sua definição inicial por SANTOS & NASCIMENTO [41], OLIVEIRA, NASCIMENTO & ABDELOUAHAB [32] propôs uma arquitetura baseada em agentes inteligentes associada a estratégias de implantação de *honeypots* para o Agente *Honey Net*. Ele é responsável pela investigação de atos suspeitos de potenciais atacantes e pela configuração dinâmica de *firewalls* e roteadores;
- Agente Sistema Operacional (SOA) : Agente que irá eliminar anomalias no sistema. Executando rotinas de interação com o sistema operacional irá realizar modificações nas configurações do sistema e no conjunto de prioridade dos usuários.

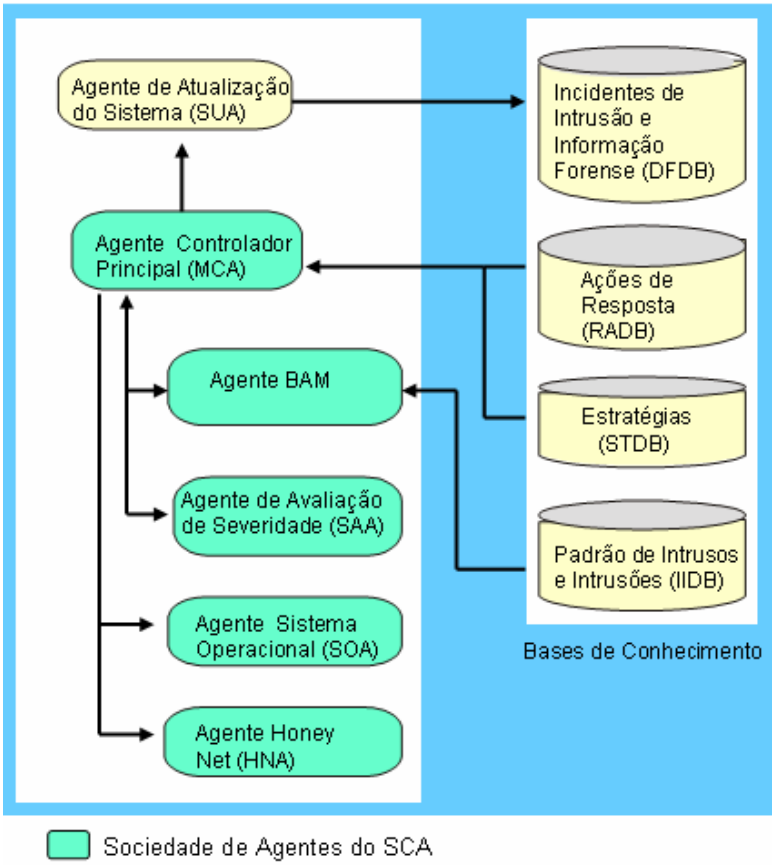


Figura – 5.5. Detalhamento do Agente Controlador de Ações - SCA.

5.2 O Mecanismo de Atualização da Base de Dados de Ações de Resposta (RADB)

A arquitetura proposta para o NIDIA prevê a capacidade do sistema responder ativamente a atividades intrusivas e nesta seção será apresentado como a RADB foi mantida atualizada e como NIDIA pode responder a intrusões com base nas informações dessa base de dados.

Tendo-se como objetivo manter a RADB atualizada, propõe-se o detalhamento do Agente de Atualização do Sistema (SUA) que agora será composto por uma sociedade de agentes inteligentes que cooperam entre si, executam de forma independente e serão responsáveis por implementar as funcionalidades do

módulo de compartilhamento de informações descritas na seção 3.3 e detalhadas na Figura 3.4.

A Figura 5.6 mostra o detalhamento do SUA, no qual o Agente Monitor será responsável pela comunicação do NIDIA com os CSIRTs, isto é, ele é um cliente de *Web service* e o Agente Analisador de XML é responsável por analisar os alertas de segurança recebidos pelo Agente Monitor e atualizar a RADB com as informações desses alertas.

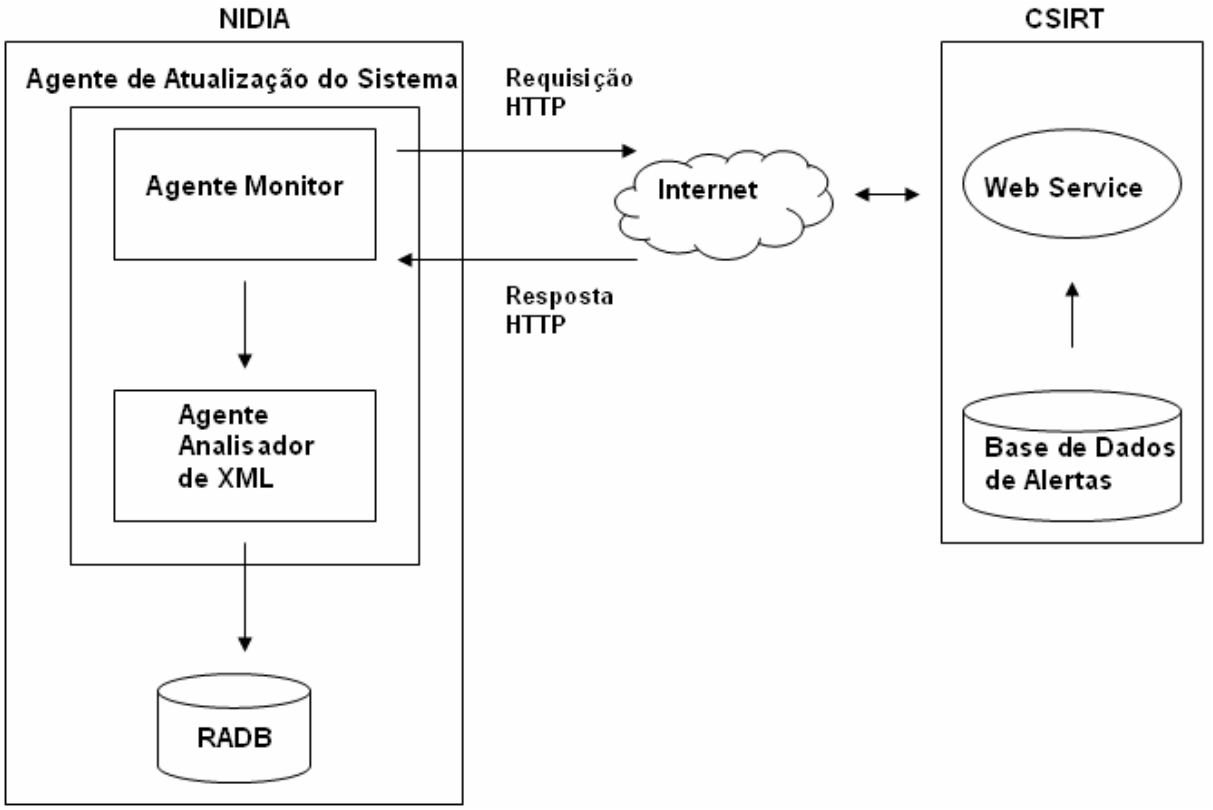


Figura – 5.6. Detalhamento do Agente de Atualização Sistema - SUA.

O Agente Monitor, que executa uma aplicação cliente do *Web service*, envia uma requisição ao *Web service* do CSIRT contendo a identificação do alerta de segurança mais recente presente na base de dados de ações de respostas. Quando uma resposta é recebida, os alertas de segurança contidos em documentos XML são enviados ao Agente Analisador de XML que será o responsável pela atualização da RADB.

A Figura 5.7 mostra, com auxílio de um diagrama de colaboração em notação UML, o funcionamento da proposta de detalhamento para o SUA.

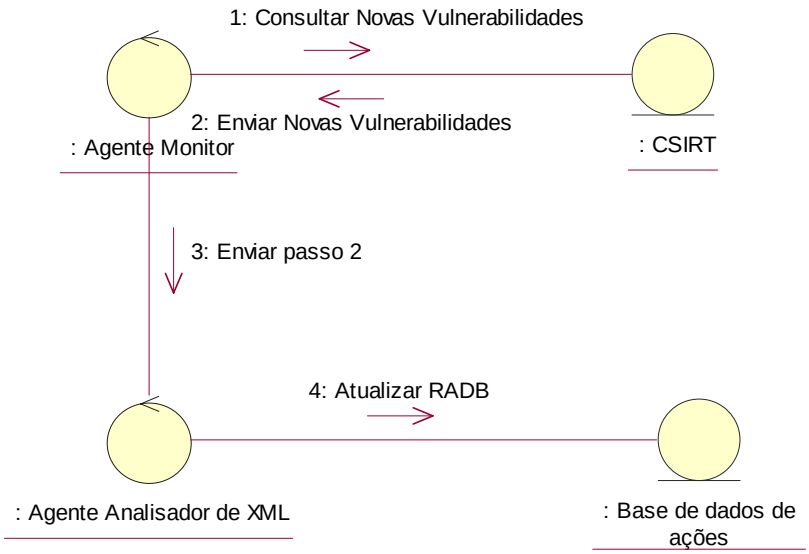


Figura – 5.7. Diagrama de colaboração do Agente de Atualização do Sistema - SUA (notação UML).

5.3 Protótipo do CSIRT

Para a realização de testes foi implementado um protótipo, representando um CSIRT, que disponibiliza vários alertas de segurança por meio de um *Web service que* é executado em um *container* Tomcat [17].

O cenário para este *Web service* constitui-se de um SDI que envia uma requisição para o CSIRT em uma dada URL utilizando o protocolo SOAP sobre HTTP. Essa requisição contém a identificação do alerta mais recente que o SDI possui. O *Web service* recebe a requisição e busca em sua base de dados por alertas mais recentes, que serão enviados de volta ao SDI na forma de arquivos XML anexados à resposta.

O CSIRT disponibiliza alertas de segurança sobre vulnerabilidades e possíveis ações de resposta contra essas vulnerabilidades na Internet. O SDI se

comunica com o servidor do CSIRT, composto por um Java *Servlet*, para obter os alertas de segurança.

Neste protótipo, a troca de mensagens entre o SDI e o CSIRT é feita por meio do sistema de mensagens requisição-resposta compatível com as especificações SOAP 1.1 e SOAP *with Attachments* [31], tendo sido estabelecido previamente as condições necessárias para a troca de documentos XML. A implementação do protótipo foi feita utilizando-se a especificação SOAP *with Attachments API for Java* (SAAJ) 1.2 que permite a troca de mensagens XML na plataforma Java possibilitando a criação, envio e o consumo dessas mensagens XML pela Internet.

As mensagens SOAP podem possuir ou não anexos. A Figura 5.8 mostra a estrutura de uma mensagem SOAP sem anexos e com exceção do cabeçalho SOAP, todas as outras partes são obrigatórias.

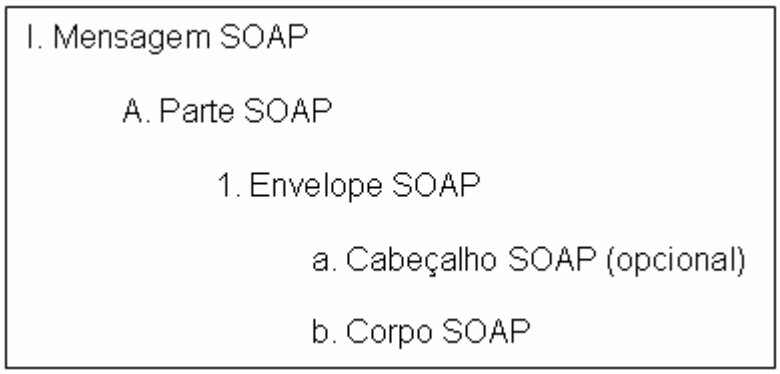


Figura – 5.8. Estrutura de uma mensagem SOAP sem anexos.

Uma mensagem SOAP pode conter uma ou mais partes de anexos e neste protótipo os anexos utilizados são constituídos por documentos XML. Com a utilização de anexos, a parte de cabeçalho torna-se obrigatória para indicar o tipo de dados contidos nos anexos.

Todas as mensagens SOAP são enviadas e recebidas por meio de uma conexão ponto-a-ponto que vai diretamente do emitente ao destinatário.

As condições necessárias para a troca de mensagens incluem o tipo e a forma das mensagens, além do sistema de mensagens a ser utilizado. Quanto ao tipo tem-se:

- O SDI envia um tipo de mensagem: Requisição por novos alertas;
- O CSIRT responde com um tipo de mensagem: Novos alertas.

A forma das mensagens trocadas é especificada pelo padrão XML *Schema*, contido nos arquivos requisicao.schema (enviada pelo SDI) e resposta.schema. O arquivo requisicao.schema encontra-se no Apêndice C e o arquivo resposta.schema é o esquema da mensagem de alerta CAP estendida apresentado na seção 3.5.

As mensagens são trocadas utilizando o sistema requisição-resposta no qual o cliente envia uma requisição e bloqueia até receber uma resposta que é enviada obrigatoriamente para cada mensagem recebida pelo servidor. Esse sistema foi escolhido pela facilidade de implementação sendo apenas necessário a especificação de uma URL para a qual a mensagem de requisição será enviada.

5.3.1 O *Web service* do CSIRT

O *Web service* do CSIRT é responsável por responder às requisições enviadas pelo SDI, e é composto por um *servlet*, *CSIRTServlet,* que possui três métodos principais : *init, doPost e onMessage.* O *CSIRTServlet* possui ainda os métodos *getHeaders* e *putHeaders* que não são métodos padrão em um *servlet*.

Os métodos *init* e *doPost* configuram a mensagem e o método *onMessage* cria o seu conteúdo.

Sendo parte de uma aplicação *Web* o *CSIRTServlet* estende a classe *HttpServlet*. Inicialmente ele cria um objeto estático *MessageFactory* que será

utilizado para criar um objeto *SOAPMessage* a ser retornado ao SDI, como observado na Figura 5.9.

```
public class CSIRTServlet extends HttpServlet {
     static MessageFactory fac = null;

     static {
          try {

               fac = MessageFactory.newInstance ();

          } catch (Exception ex) {
               ex.printStackTrace ();
          }
     };
```

Figura – 5.9. Classe CSIRTServlet.

O método *init* inicializa o *servlet* com informações de configuração recebidas do *Tomcat*, conforme mostra a Figura 5.10.

O método *doPost* recebe do *Tomcat* dois argumentos:

- O primeiro é o objeto *req* do tipo *HttpServletRequest* que contém o conteúdo da mensagem enviada. Esse conteúdo é então posto em um objeto *msg* do tipo *SOAPMessage* que será enviado ao método *onMessage*;
- O último argumento é o objeto *resp* do tipo *HttpServletResponse* que conterá a mensagem gerada após a execução do método *onMessage*.

Em seguida, o método *doPost* chama o método *getHeaders* para obter o cabeçalho da requisição *req* e escrevê-lo na resposta *resp* e obtém o conteúdo da requisição *req* que será passado juntamente com o cabeçalho ao método *MessageFactory.createMessage* obtendo-se o objeto *msg*.

Para construir a resposta, o método *doPost* declara um objeto *reply* do tipo *SOAPMessage* e o popula chamando o método *onMessage*, enviando como parâmetro a mensagem recebida do SDI (o objeto *msg)*.

Se o conteúdo do objeto *reply* não é nulo, ele é salvo, e o estado do objeto *resp* é configurado para "OK". Em seguida o cabeçalho e o conteúdo de *reply* são escritos no objeto *resp*. No caso do objeto *reply* ter um conteúdo vazio, o estado do objeto *resp* é configurado para indicar que a requisição foi recebida com sucesso mas não há nova informação a ser retornada. A Figura 5.10 também exibe o código para o método *doPost*.

```
public void init(ServletConfig servletConfig)
                 throws ServletException {
          super.init(servletConfig);
}
public void doPost(HttpServletRequest req,
                    HttpServletResponse resp)
                   throws ServletException, IOException {
          try {
               // Obtém todos os cabeçalhos da requisição HTTP
               MimeHeaders headers = getHeaders(req);
               // Obtém o corpo da requisição HTTP
               InputStream is = req.getInputStream();
               // Internaliza o conteúdo da requisição HTTP
               // e cria uma SOAPMessage
               SOAPMessage msg = fac.createMessage(headers,is);
               // Chama o método onMessage.
               SOAPMessage reply = null;
               reply = onMessage(msg);

               if (reply != null) {
                    if (reply.saveRequired()) {
                         reply.saveChanges();
                    }
                    resp.setStatus(HttpServletResponse.SC_OK);
                    putHeaders(reply.getMimeHeaders(), resp);

                    // Escreve a mensagem na stream de resposta
                    OutputStream os = resp.getOutputStream();
                    reply.writeTo(os);
                    os.flush();
               } else {
                    resp.setStatus(
                    HttpServletResponse.SC_NO_CONTENT);
               }
          } catch (Exception ex) {
               throw new ServletException("SAAJ POST failed: "
                                          + ex.getMessage());
          }
}
```

Figura – 5.10. Métodos init e doPost da classe CSIRTServlet.

O método *doPost* chama o método *getHeaders* passando o objeto *req* e recebe um objeto do tipo *MimeHeaders* populado com o cabeçalho de *req*. O método *putHeaders* também é chamado pelo *doPost* e tem como responsabilidade escrever o cabeçalho da resposta que será envida ao cliente do *Web service*.

No método *onMessage* encontra-se o código da aplicação responsável por criar a mensagem de resposta que será enviada ao cliente do *Web service.* Ele utiliza o objeto *msg,* passado para ele pelo método *doPost,* para obter a identificação do alerta de segurança mais recente que o SDI cliente possui.

Em seguida, ele cria a mensagem de resposta e lê um arquivo texto (que simula um banco de dados) contendo a relação de alertas disponíveis, seleciona aqueles que possuem um número de identificação maior que o enviado pelo SDI e cria uma parte anexa à mensagem de resposta para cada arquivo selecionado.

A Figura 5.11 mostra o código do método *onMessage*.

```
public SOAPMessage onMessage(SOAPMessage msg) {
      try {
      // Recuperar id do alerta mais recente que o SDI possui.
      SOAPBody sentSB =
                msg.getSOAPPart().getEnvelope().getBody();
      Iterator sentIt = sentSB.getChildElements();
      SOAPBodyElement sentSBE = (SOAPBodyElement)
      sentIt.next();
      Iterator sentIt2 = sentSBE.getChildElements();
      SOAPElement sentSE = (SOAPElement) sentIt2.next();
      String sentID = sentSE.getValue();

      // Criar a mensagem de resposta
      confirmation = fac.createMessage();

      // Obter o cabeçalho e o corpo da mensagem SOAP
      SOAPHeader header = confirmation.getSOAPHeader();
      SOAPBody body = confirmation.getSOAPBody();

      // Arquivo de Alertas do CSIRT
      fr = new FileReader(new File("alertas.txt"));
      br = new BufferedReader(fr);

      // Ler a primeira linha do Arquivo de Alertas do CSIRT
      strAlerta = br.readLine();
      int i = 0;
      while (strAlerta != null) {

        if (Integer.parseInt(strAlerta) >
                  Integer.parseInt(sentID)) {
            // Criar o anexo para o arquivo do alerta
            URL url = new URL ("file: CAP_VU" + strAlerta +
                              ".txt");

            DataHandler dataHandler = new DataHandler(url);
            attachment[i] =
                  confirmation.createAttachmentPart(dataHandler);

            attachment[i].setContentId("CAP_VU" + strAlerta);

            confirmation.addAttachmentPart(attachment[i]);
            i = i + 1;
        }
        strAlerta = br.readLine();
    }
    confirmation.saveChanges();

    } catch (Exception e) {
            e.printStackTrace();
    }
    return confirmation;
}
```

Figura – 5.11. Método onMessage da classe CSIRTServlet.

5.4 Protótipo NIDIA

Com o objetivo de criar a sociedade de agentes de forma mais amigável, utilizou-se a linguagem Java e o Zeus [20] que é um conjunto de ferramentas capaz de dar suporte ao desenvolvimento de sistemas com agentes cooperativos.

O protótipo do NIDIA utilizou a implementação feita por LIMA et al [27] para o agente sensor de rede e as implementações de DIAS & NASCIMENTO [23] para os agentes de Monitoramento do Sistema (SMA) e de Avaliação de Segurança do Sistema (SEA).

O Agente Controlador Principal (MCA) foi parcialmente implementado, como parte deste trabalho, seguindo-se a definição de SANTOS & NASCIMENTO [41] e para o Agente BAM (BA), também sugerido por SANTOS & NASCIMENTO [41], uma nova proposta é apresentada neste trabalho na seção 5.4.6.1.

Foi realizada a integração deste protótipo com o protótipo do Agente *Honey Net* (HNA) desenvolvido por OLIVEIRA, NASCIMENTO & ABDELOUAHAB [32]. Essa integração é explicada na seção 5.4.6.2.

O Agente Monitor e o Agente Analisador de XML foram implementados de acordo com os objetivos de cada agente exposto na seção 5.2. O detalhamento dessas implementações encontra-se nas seções 5.4.6.3 e 5.4.6.4.

O Zeus suporta o gerenciamento e a comunicação de agentes autônomos estáticos em java, de forma que o desenvolvedor preocupa-se apenas com o problema do negócio, pois a infra-estrutura para criação e comunicação de agentes já se encontra pronta no Zeus.

O processo de construção da sociedade de agentes é constituído de uma série de fases e devem obedecer a seguinte ordem de execução:

i) Criação da Ontologia

ii) Criação dos Agentes;

iii) Configuração dos Agentes Utilitários;

iv) Configuração dos Agentes Tarefa;

v) Implementação dos Agentes.

5.4.1 Criação da ontologia

Ontologia são os conhecimentos declarativos que representam conceitos importantes para uma aplicação [20]. A ferramenta utilizada para criar a ontologia é o editor de ontologia (*Ontology Editor*) do Zeus.

Para o Zeus um conceito é significante e deve ser modelado quando a interação entre agentes não puder ser feita sem que ambos conheçam esse conceito. O termo *fact* é utilizado pelo Zeus para descrever um conceito individual.

A Figura 5.12 mostra o editor de ontologias do Zeus onde foram criados os seguintes conceitos ou *fact* segundo a nomenclatura do Zeus:

- ArqConexao: produzido pelo agente sensor de rede, a partir do conteúdo dos datagramas IP recolhidos das conexões TCP entre servidor-cliente;
- ArqRede: de responsabilidade do Agente de Monitoramento do Sistema (SMA) que consulta a base de dados de palavras-chaves e produz um vetor de contagem dessas palavras para cada conexão de rede e os envia ao Agente de Avaliação de Segurança do Sistema (SEA);
- IndiceAtaque: representa na ontologia o índice de severidade produzido pelo SEA;

- NomeAtaque: o agente BAM (BA) de posse do índice de severidade e do vetor de contagem de palavras-chave identifica o nome do ataque que é representado na ontologia por esse conceito;
- AcaoResposta: de posse do índice de severidade e do nome do ataque, o Agente Controlador Principal (MCA) deve identificar as ações de resposta a serem tomadas, que são representadas na ontologia por esse conceito;
- Responder: representa as ações de resposta que serão tomadas pelo Agente *Honey Net* (HNA);
- ArquivoXML: produzido pelo Agente Monitor e contém informações das ações de resposta contra novas vulnerabilidades que foram recebidas do CSIRT.

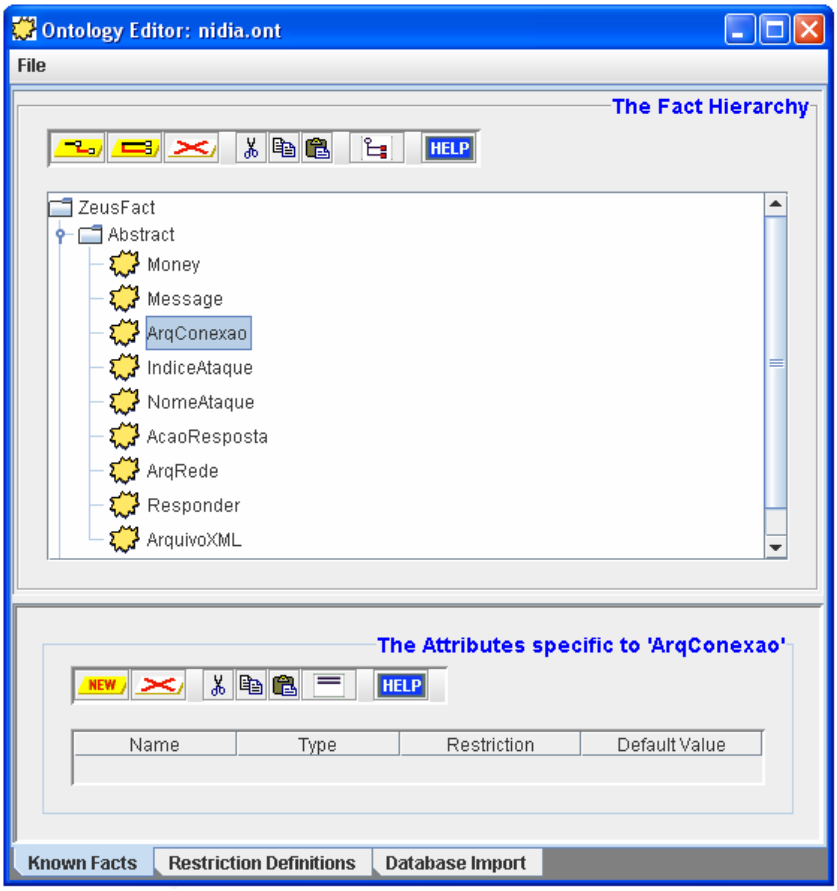


Figura – 5.12. *Ontology Editor* do Zeus.

5.4.2 Criação dos agentes

Os agentes Zeus genéricos são criados e configurados no painal *Agent Options* do Zeus, atribuindo-lhes tarefas da aplicação.

Foram criados os agentes: NetSensor (sensor de rede), SMA (monitoramento), SEA (avaliação de segurança), MCA (controlador principal), BA (BAM), HNA (*honey net*), Monitor e AnalizadorXML.

O Agente de Avaliação de Severidade e o Agente Sistema Operacional não foram implementados.

A Figura 5.13 mostra o painel *Agent Options* do Zeus.

Figura – 5.13. Painel *Agent Options* do Zeus.

Após a criação dos agentes é necessário configurá-los, atribuindo-lhes tarefas, fornecê-lhes informações sobre os outros agentes e das habilidades desses outros agentes e finalmente equipá-los com protocolos e conhecimentos de coordenação necessários para a interação com outros agentes.

Portanto para cada agente foi criada e associada uma tarefa que ele é capaz de executar. Para cada tarefa é necessário especificar as suas pré-condições e os seus efeitos o que é feito no painel *Task Options* do Zeus exibido na Figura 5.14. Pré-condições são os recursos necessários para a execução de uma tarefa e os efeitos são os recursos produzidos pela execução dessa tarefa.

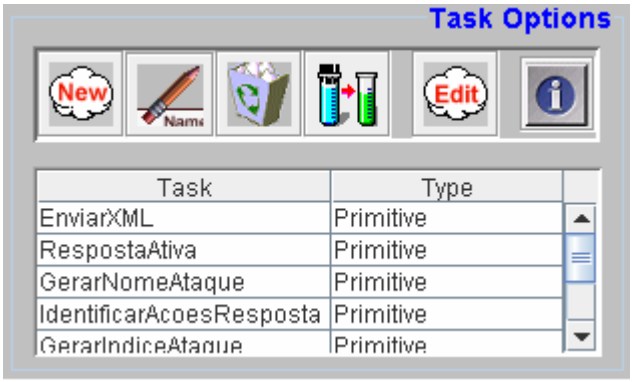


Figura – 5.14. Painel *Task Options* do Zeus.

Na Figura 5.15 tem-se o detalhamento da tarefa "GerarNomeAtaque" que produzirá "NomeAtaque" e tem como pré-condição "IndiceAtaque".

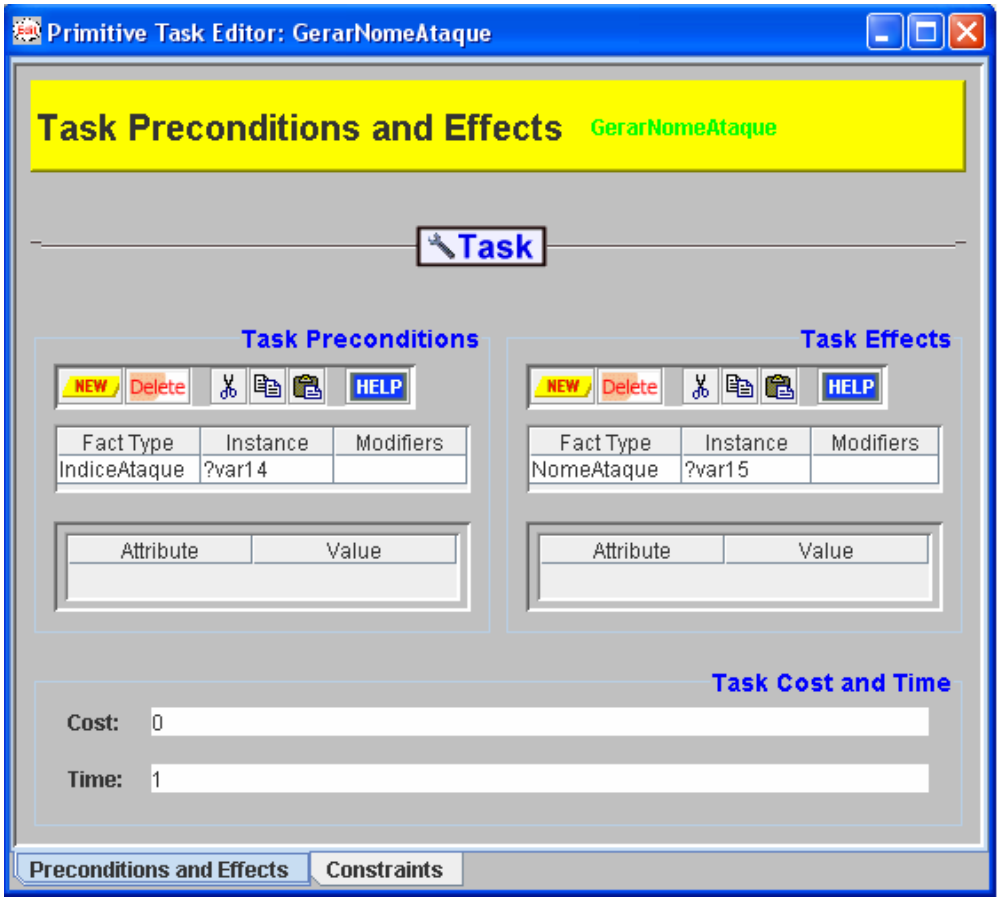


Figura – 5.15. *Primitive Task Editor* do Zeus.

A tabela 5.1, lista os agentes, suas respectivas tarefas com pré-condições e efeitos que são os conceitos especificados no editor de ontologias do Zeus. De acordo com essa tabela o agente NetSensor executará a tarefa GerarArqConexão

que não possui uma pré-condição para ser executada e que terá como efeito a geração de ArqConexão.

Tabela - 5.1. Agentes, pré-condições e efeitos de suas tarefas.

Agente	Tarefa	Pré-Condição	Efeito
NetSensor	GerarArqConexao	-	ArqConexao
Monitoramento do Sistema (SMA)	GerarArqRede	ArqConexao	ArqRede
Avaliação de Segurança do Sistema (SEA)	GerarIndiceAtaque	ArqRede	IndiceAtaque
Controlador Principal (MCA)	IdentificarAcoesResposta	IndiceAtaque NomeAtaque	AcaoResposta
BA	GerarNomeAtaque	IndiceAtaque	NomeAtaque
Honey Net (HNA)	RespostaAtiva	AcaoResposta	RespostaAtiva
Monitor	EnviarXML	-	ArquivoXML
AnalizadorXML	AtualizarRADB	ArquivoXML	AtualizarRADB

Após a configuração dos agentes e das tarefas que eles são capazes de executar é necessário que eles tenham conhecimento dos outros agentes e das habilidades desses outros agentes. Isso pode ser atingido por meio de um serviço de diretório ou de conhecimento prévio sobre os outros agentes.

Existem quatro tipos de relacionamentos entre agentes:

- Equivalentes: Relacionamento padrão, não havendo suposição sobre a interação dos agentes;
- Superior: O agente possui autoridade superior e pode emitir ordens aos outros agentes;
- Subordinado: Agente de menos autoridade que deve obedecer a ordens de agentes superiores;

- *Co-worker* : Agentes pertencentes a mesma “comunidade” e serão consultados antes dos agentes “Equivalentes” quando algum recurso for requisitado.

Para os objetivos deste trabalho foram estabelecidos os seguintes relacionamentos entre os agentes:

- NetSensor: Subordinado ao SMA;
- SMA: Superior ao NetSensor e subordinado ao SEA;
- SEA: Superior ao SMA e subordinado ao MCA;
- MCA: Superior ao SEA, BA e HNA;
- BA: Subordinado ao MCA;
- HNA: Subordinado ao MCA;
- Monitor: Subordinado ao AnalisadorXML;
- AnalisadorXML: Superior ao Monitor.

Na etapa de coordenação dos agentes, os mesmos foram equipados com o protocolo de coordenação e estratégias, que implementam vários aspectos da conversação tipo rede de contrato (*contract-net*).

A contratação como um mecanismo de coordenação é simbolizada pelo protocolo de rede contratual clássico [22], onde um agente gerenciador anuncia um contrato, recebe as ofertas de outros agentes interessados e após serem avaliadas, o contrato é entregue ao agente premiado. Esta é a abordagem utilizada pelos agentes Zeus, onde um ou mais "Iniciadores" emitem uma chamada para propostas (*cfp*), e um ou mais "Participantes" respondem à solicitação.

Utilizando-se o painel *Agent Coordination* do editor de agentes do Zeus, definiu-se os agentes NetSensor, BAM (BA), *Honey Net* (HNA) e Monitor como "Participantes", os agentes de Monitoramento do Sistema (SMA) e de Avaliação de

Segurança do Sistema (SEA) como "Iniciadores" e "Participantes" e os agentes Controlador Principal (MCA) e AnalisadorXML como "Iniciadores".

A Figura 5.16 mostra o painel *Agent Coordination* do Agente AnalisadorXML.

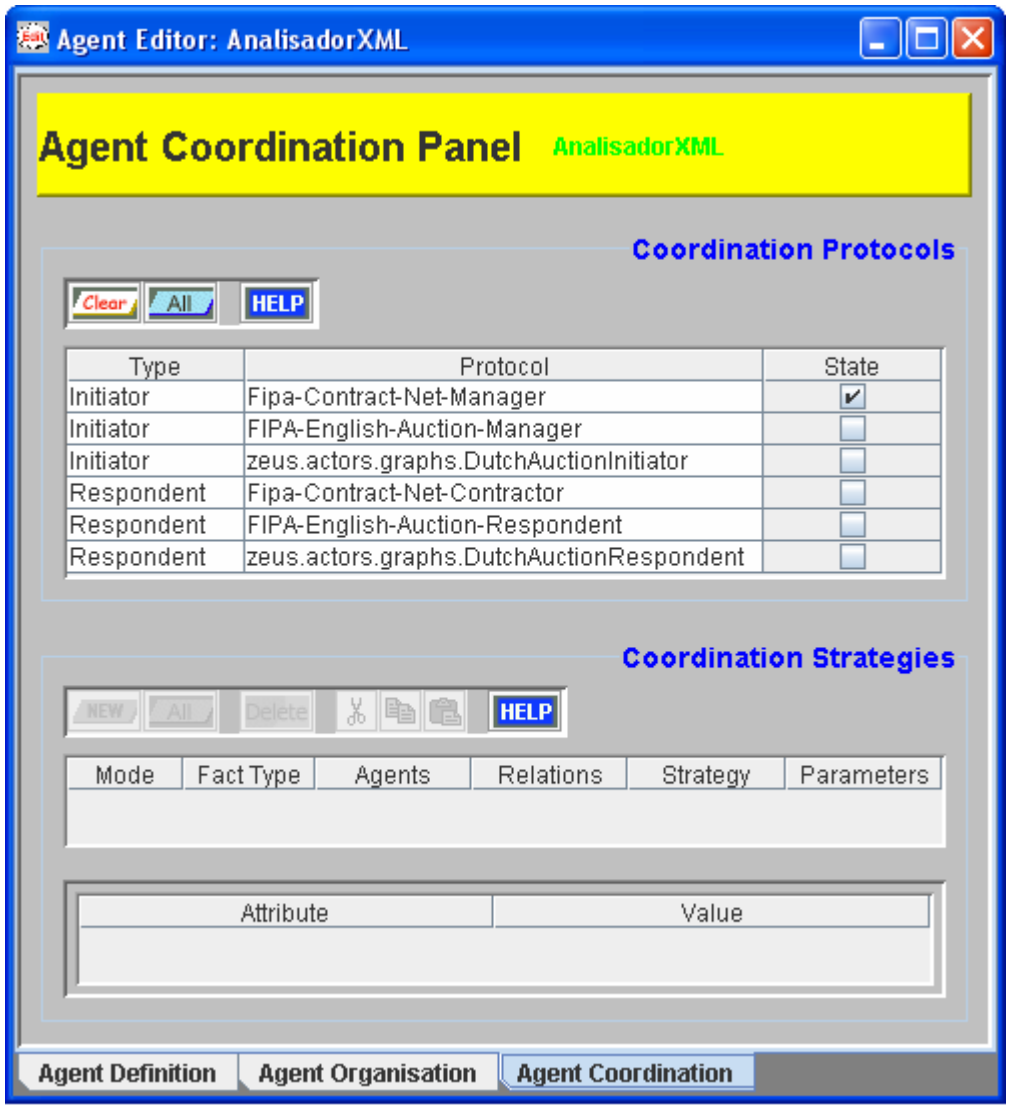


Figura – 5.16. Painel *Agent Coordination* do Zeus.

5.4.3 Configuração dos agentes utilitários

Os Agentes utilitários "Servidor de Nomes", "Facilitador" e "Visualizador" fornecem a infra-estrutura de suporte para os Agentes Zeus e são configurados no painel *Utility Agent* do gerador de código do Zeus que é exibido na Figura 5.17.

Uma sociedade de Agentes deve possuir obrigatoriamente pelo menos um Agente Servidor de Nomes (ANS). O ANS mantém um registro dos Agentes habilitando-os a mapear identidades de Agentes a localizações lógicas em uma rede de computadores. Isso é necessário porque os Agentes apenas conhecem o nome dos outros Agentes, desconhecendo a localização.

O agente "Facilitador" deve ser criado de acordo com a natureza da aplicação e fornece um serviço de diretório para que os agentes possam descobrir os nomes e habilidades dos outros agentes.

O uso do agente "Visualizador" também não é obrigatório, entretanto ele fornece a capacidade de monitorar e analisar a sociedade de agentes tendo sido incluído neste protótipo.

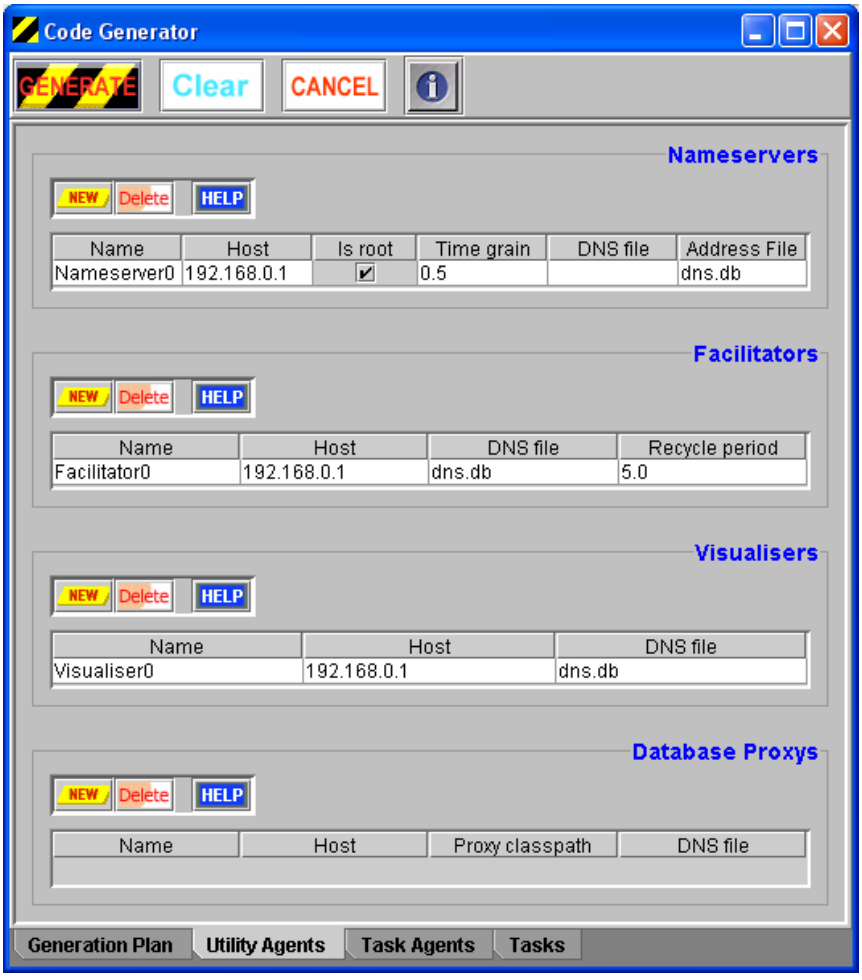


Figura – 5.17. Painel *Utility Agents* do gerador de códigos do Zeus.

5.4.4 Configuração dos agentes tarefa

Agentes Tarefa são os agentes Zeus genéricos criados na seção 5.4.2 e para configurá-los deve-se especificar o endereço IP da máquina onde cada agente irá executar, a localização do arquivo padrão de DNS (dns.db) que é gravado pelo ANS quando ele inicia, os recursos externos que ele poderá acessar para executar suas tarefas e os programas externos com os quais eles irão interagir.

Ainda é possível gerar uma *interface* gráfica para cada agente, utilizando-se a opção *Create GUI?*, que exibe informações detalhadas sobre os componentes internos dos agentes.

Os Agentes Tarefa são configurados no painel *Task Agents* do gerador de códigos do Zeus que é exibido na Figura 5.18.

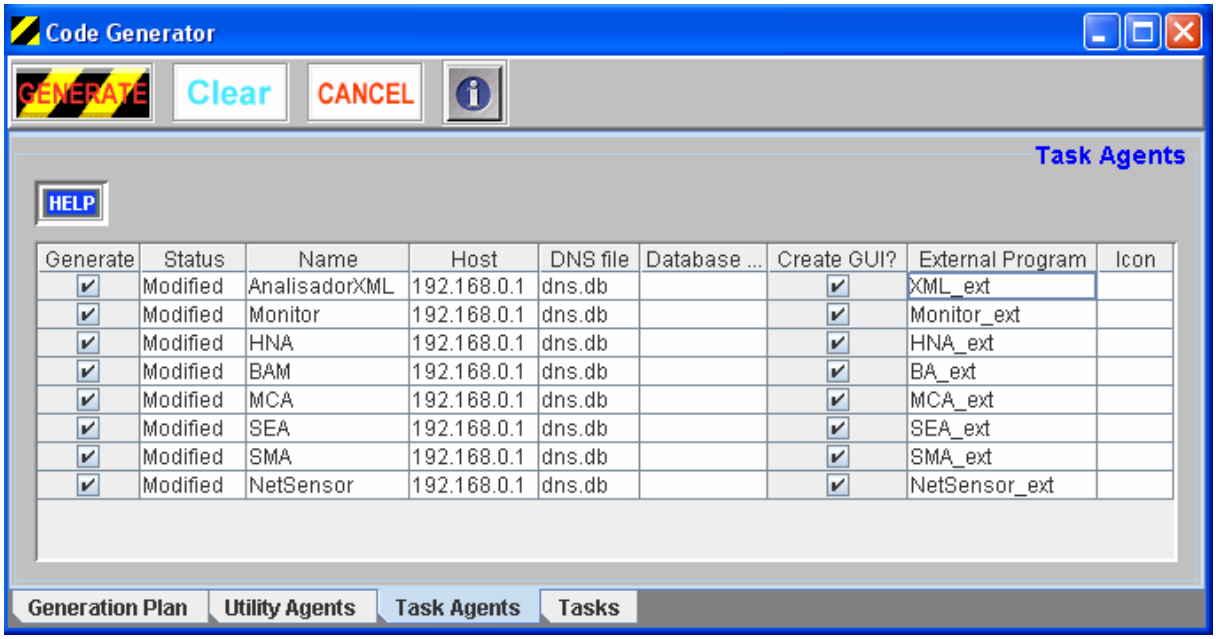


Figura – 5.18. Painel *Task Agents* do gerador de códigos do Zeus.

A próxima seção descreve como os agentes foram implementados, o que envolve as etapas de geração do código fonte dos agentes e de seus programas externos.

5.4.5 Implementação dos agentes

A geração do código fonte dos agentes e de suas tarefas é feita usando-se o painel *Generation Plan* do gerador de código mostrado na Figura 5.19.

Por meio de um procedimento simples informa-se onde o código fonte será gerado e em qual sistema operacional os agentes serão utilizados. Ao final do processo de geração, os seguintes arquivos são criados: NetSensor.java, SMA.java, SEA.java, MCA.java, BA.java, HNA.java, Monitor.java, AnalisadorXML.java, GerarArqConexao.java, GerarArqRede.java, GerarIndiceAtaque.java, IdentificarAcoesResposta.java, GerarNomeAtaque.java, RespostaAtiva.java, EnviarXML.java e AtualizarRADB.java.

Entretanto não são criados os arquivos que implementam os programas externos. Esses arquivos foram desenvolvidos para executar tarefas específicas da aplicação e serão detalhados na seção 5.4.6.

Figura – 5.19. Painel *Generation Plan* do gerador de códigos do Zeus.

5.4.6 Implementação dos programas externos dos agentes

Os programas externos com os quais cada agente interage foram definidos na seção 5.4.4 e são responsáveis por executar as tarefas de cada agente de acordo com a lógica de negócio da sociedade criada.

A seguir descreve-se os programas externos para os agentes Controlador Principal (MCA) e BAM (BA) (MCA_ext e BA_ext), para o Agente *Honey Net* (HNA) (HNA_ext) e os programas externos para os agentes Monitor (Monitor_ext e AlertRequest) e AnalisadorXML (AnalisadorXML_ext).

5.4.6.1 Implementação dos programas externos para os agentes Controlador Principal (MCA) e BAM (BA)

O MCA foi implementado de forma parcial por meio da classe MCA_ext. Essa classe recebe do Agente de Avaliação de Segurança do Sistema (SEA) o grau de severidade e o vetor de contagem de palavras-chave para cada conexão que ele analisou. Esse vetor de palavras-chave contém a identificação da conexão e o número de ocorrências de cada palavra-chave.

De posse dessas informações, o MCA envia o vetor de contagem de palavras-chave ao BA.

Como a classe BA_ext recebe um vetor de contagem de palavras-chave e não um vetor binário como exposto por SANTOS & NASCIMENTO [41], uma nova proposta para o Agente BA foi desenvolvida.

Propõe-se a utilização do modelo clássico de recuperação de informação denominado espaço vetorial [9], para o processo de comparação do vetor de palavras-chave com as informações da Base de Dados de Padrões de Intrusos e Intrusões (IIDB), cujos dados são gerados de acordo com o exposto na seção 4.2.4.

A técnica do modelo do espaço vetorial foi escolhida pelo fato de ser um dos modelos mais utilizados e simples de implementar. Na IIDB cada ataque está relacionado a um vetor de contagem de palavras-chave, como mostra a tabela 5.2, e

a classe BA_ext irá comparar o vetor recebido do MCA com os existentes na IIDB para identificar o nome do ataque.

Tabela - 5.2. IIDB do Protótipo do NIDIA.

Nome do Ataque	cat\s*>	Jumping to address	/cgi-bin/phf	chmod\s	(?i)login incorrect	(?i)permission denied	...
CAN-2004-0459	1	1	0	0	0	0	
CAN-2004-0396	1	1	0	0	0	0	
CAN-2004-0747	0	0	1	1	0	0	
CAN-2004-0800	0	0	1	1	0	0	
CAN-2004-0203	0	0	0	0	1	0	
CVE-1999-0067	0	0	0	0	0	1	
...							

Para realizar a análise de similaridade entre o vetor recebido do MCA e as informações da IIDB a classe BA_ext utiliza um cálculo de similaridade que determinará o nome do ataque.

Para esse cálculo, utilizou-se a seguinte equação:

$$I_S = \Sigma(P_t * F_t)$$

onde:

I_S = Índice de Similaridade

P = Número de ocorrências de uma palavra-chave no vetor enviado pelo MCA

F = Número de ocorrências de uma palavra-chave na IIDB

O BA envia o nome do ataque ao MCA que irá enviá-lo, juntamente com a identificação da conexão sendo utilizada pelo invasor, ao Agente *Honey Net* (HNA).

5.4.6.2 Implementação do programa externo do agente *Honey Net* (HNA)

A classe HNA_ext implementa o programa externo do HNA e é responsável pela integração com o protótipo do HNA.

Ela recebe informações do Agente Controlador Principal (MCA) contendo o nome do ataque detectado e a identificação da conexão que originou esse ataque, da qual é possível extrair os endereços IP de origem e destino e as portas de origem e destino.

Em seguida, a classe HNA_ext adiciona uma regra no *firewall* bloqueando o acesso do atacante. Para atingir seu objetivo, a classe HNA_ext cria um objeto do tipo "Mensagem" contendo as informações da conexão utilizada pelo atacante e que será utilizado para inserir uma regra no *firewall,* bloqueando o acesso do invasor. O código Java responsável por essas atividades está na Figura 5.20.

```
/*
 * Gera mensagem
 */
Mensagem msg = new Mensagem();
msg.setIpOrigem(IPOrigem);
msg.setIpDestino(IPDestino);
msg.setPortDestino(Integer.parseInt(PortaDestino));
msg.setPortOrigem(Integer.parseInt(PortaOrigem));
msg.setProtocolo("TCP");

/*
 * Insere regra no firewall
 */
DynaRule dynarule = new DynaRule(msg, "192.168.2.106");
dynarule.EnviaMensagemFirewall();
```

Figura – 5.20. Código Java do programa externo (HNA_ext) do agente Honey Net - HNA.

5.4.6.3 Implementação do programa externo do Agente Monitor

Esta seção concentra-se no programa externo para o Agente Monitor que é responsável por interagir com o *Web service* do CSIRT em busca de novos alertas de segurança emitidos e desta forma o seu programa externo implementa um cliente desse *Web service*. O Agente Monitor envia uma requisição ao *Web service* contendo a identificação do alerta mais recente presente na Base de Dados de Ações de Respostas (RADB) e aguarda pela resposta que poderá conter novos alertas emitidos na forma de documentos XML.

Assim que uma resposta é recebida, o Agente Monitor grava os documentos XML anexos na forma de arquivos e envia uma mensagem ao Agente AnalisadorXML informando a identificação dos novos alertas que estão disponíveis.

Este cliente de *Web service* constitui-se das classes Java Monitor_ext e AlertRequest que foram implementadas utilizando-se o *Java Web Services Developer Pack (WSDP)* 1.3 [2].

A classe Monitor_ext invoca a classe AlertRequest que tem duas responsabilidades: enviar uma requisição e extrair o conteúdo da resposta.

A classe AlertRequest inicia criando uma conexão que será utilizada para enviar a requisição e ,em seguida, ela cria o objeto *msg* do tipo SOAPMessage que representa a mensagem a ser enviada. A Figura 5.21 contém o código Java desta etapa.

```
SOAPConnectionFactory scf =
          SOAPConnectionFactory.newInstance();
SOAPConnection        con = scf.createConnection();

MessageFactory mf  = MessageFactory.newInstance();
SOAPMessage    msg = mf.createMessage();
```

Figura – 5.21. Classe AlertRequest criando uma conexão.

Em seguida, o objeto SOAPEnvelope da mensagem é acessado e ele será utilizado para acessar o objeto do tipo *SOAPBody*, ao qual o conteúdo da mensagem será adicionado. A implementação desta etapa está na Figura 5.22.

```
SOAPPart part = msg.getSOAPPart();
SOAPEnvelope envelope = part.getEnvelope();
SOAPBody body = envelope.getBody();
```

Figura – 5.22. Classe AlertRequest acessando o corpo da mensagem.

O arquivo requisicao.schema especifica que o elemento de primeiro nível no conteúdo do elemento *body* é o elemento *request-alerts* e que esse elemento contém o elemento *request*. O nó texto adicionado ao elemento *request* é o texto da requisição a ser enviada e contém a identificação do alerta mais recente que o SDI possui em sua Base de Dados.

Cada novo elemento que é adicionado à mensagem deve obrigatoriamente possuir um objeto do tipo *Name* para identificá-lo, o qual é criado por meio do método *createName* da classe *Envelope*. A criação do conteúdo da mensagem a ser enviada é obtida com o código da Figura 5.23.

```
Name bodyName = envelope.createName("request-alerts",
                "RequestAlerts", "http://nidia.dee.ufma.br");
SOAPBodyElement requestAlerts = body.addBodyElement(bodyName);
Name requestName = envelope.createName("request");
SOAPElement request =
                requestAlerts.addChildElement(requestName);
request.addTextNode("192038");
```

Figura – 5.23. Classe *AlertRequest* adicionando um elemento à mensagem.

Neste ponto, a mensagem encontra-se pronta para ser enviada ao *Web service* do CSIRT por meio do objeto *con* do tipo *SOAPConnection,* conforme mostra a Figura 5.24.

```
URL endpoint = new URL(URLHelper.getSaajURL() +
                        "/getAlertList");
SOAPMessage response = con.call(msg, endpoint);
con.close();
```

Figura – 5.24. Classe AlertRequest enviando uma mensagem.

A segunda tarefa da classe *AlertRequest* é obter o conteúdo da mensagem de resposta que está contido em anexos na forma de documentos XML. Esses anexos encontram-se em os objetos do tipo *AttachmentPart* que são obtidos por meio do método *getAttachments* da classe *SOAPMesage*.

O método *getAttachments* retorna um objeto do tipo *java.util.Iterator,* possibilitando o acesso a todos os anexos na forma de objetos do tipo *AttachmentPart.* Assim que um anexo é obtido o seu conteúdo é gravado como um arquivo XML.

O código da Figura 5.25 executa as ações de acessar os anexos da mensagem e gravar o conteúdo de cada um deles em arquivos XML.

```
Iterator iterator = response.getAttachments();

while (iterator.hasNext()) {

      AttachmentPart attached = (AttachmentPart)
      iterator.next();

      String id = attached.getContentId();
      String type = attached.getContentType();
      Object content = attached.getContent();

      // Criando o arquivo XML contendo o alerta recebido
      file = new FileOutputStream(id + ".xml");
      BufferedOutputStream buffer = new
                                    BufferedOutputStream(file);
      DataOutputStream dados = new DataOutputStream(buffer);

      dados.writeBytes(((String) content));
      dados.flush();

}
```

Figura – 5.25. Classe AlertRequest obtendo o conteúdo da mensagem de resposta.

5.4.6.4 Implementação do programa externo do Agente AnalisadorXML

O Agente AnalisadorXML é responsável por atualizar a Base de Dados de Ações de Respostas (RADB) com as informações contidas nos arquivos XML de alertas. Neste protótipo simulou-se o processo de atualização da RADB por meio de um arquivo texto onde a identificação de cada novo alerta recebido é gravada.

Após receber uma mensagem do Agente Monitor contendo a identificação dos alertas recebidos, o arquivo "RADB.txt" é atualizado.

5.5 Estudo Comparativo do NIDIA com os outros SDIs

Com a construção dos protótipos do CSIRT e do NIDIA mostrou-se como a Base de Dados de Ações de Respostas do NIDIA pode ser atualizada de forma automática. Utilizando-se a proposta de arquitetura multiagente para atualização automática do mecanismo de detecção de ataques dos SDIs multiagente, o NIDIA pode também atualizar de forma automática o seu mecanismo de detecção de ataques baseado em assinaturas.

A tabela 5.3 mostra o comparativo do NIDIA com outros SDIs com enfatizando-se a característica de atualização automática.

Tabela - 5.3. Atualização dos SDIs multiagente e o NIDIA.

Característica / Protótipo	Atualização das Assinaturas de Ataques		Atualização das Ações de Respostas	
	Manual	Auto	Manual	Automática
AAFID	x		x	
CSM	x		x	
AID	x		x	
Hummingbird	x		x	
IDA	x		x	
NIDIA		x		x

5.6 Conclusões

Neste capítulo, foi descrita a construção do protótipo do modelo de atualização automática do conjunto de ações de respostas apresentado no capítulo 3. Esse modelo foi aplicado ao SDI multiagente NIDIA.

Um protótipo de um CSIRT também foi desenvolvido utilizando-se a linguagem Java e as especificações SOAP 1.1 e *SOAP with Attachments* .

O protótipo do NIDIA foi construído com a ferramenta Zeus que fornece toda a infra-estrutura para a criação dos agentes e para a comunicação entre eles e a comunicação do NIDIA com o protótipo do CSIRT pode ser feita utilizando-se *Web services.*

6. RESULTADOS PARCIAIS DE SIMULAÇÕES COM OS PROTÓTIPOS DO NIDIA E DO CSIRT

Este capítulo apresenta os resultados de simulações realizadas com os protótipos do NIDIA e do CSIRT que foram apresentados no capítulo 5.

Essas simulações tiveram como objetivo manter atualizada a Base de Dados de Ações de Respostas (RADB) do NIDIA com os alertas emitidos pelo protótipo do CSIRT. Também será mostrado como o NIDIA foi capaz de responder ativamente a uma tentativa de ataque, alterando as regras do *eTrust Firewall*, bloqueando o acesso da máquina invasora.

Para um melhor entendimento do funcionamento do protótipo, a Figura 6.1 que exibe o fluxo de informações do protótipo do NIDIA, com os agentes, pré-condições e efeitos das tarefas executadas.

O Agente Netsensor executa a tarefa "GerarArqConexão" que tem como efeito arquivos texto (ArqConexao) com o conteúdo das conexões de rede sendo monitoradas. De posse dos arquivos de conexão gerados pelo Netsensor, o Agente de Monitoramento do Sistema (SMA) executa a tarefa "GerarArqRede" que formata os arquivos de conexão como um vetor de contagem de palavras-chave (ArqRede).

O vetor de contagem de palavras-chave é enviado ao Agente de Avaliação de Segurança do Sistema (SEA) que gera um índice de ataque (IndiceAtaque) que será enviado ao Agente Controlador Principal (MCA). De posse do índice de ataque, o MCA solicita ao Agente BAM (BA) a identificação do nome do ataque (NomeAtaque). De posse do nome do ataque, o MCA envia uma mensagem (AcaoResposta) ao agente *Honey Net* que será o responsável pela execução de uma ação de resposta (RespostaAtiva).

O Agente Monitor recupera do CSIRT novos arquivos XML contendo alertas de segurança (ArquivoXML) e os envia ao Agente AnalisadorXML que é o responsável pela atualização da Base de Dados de Ações de Respostas (RADB) do NIDIA.

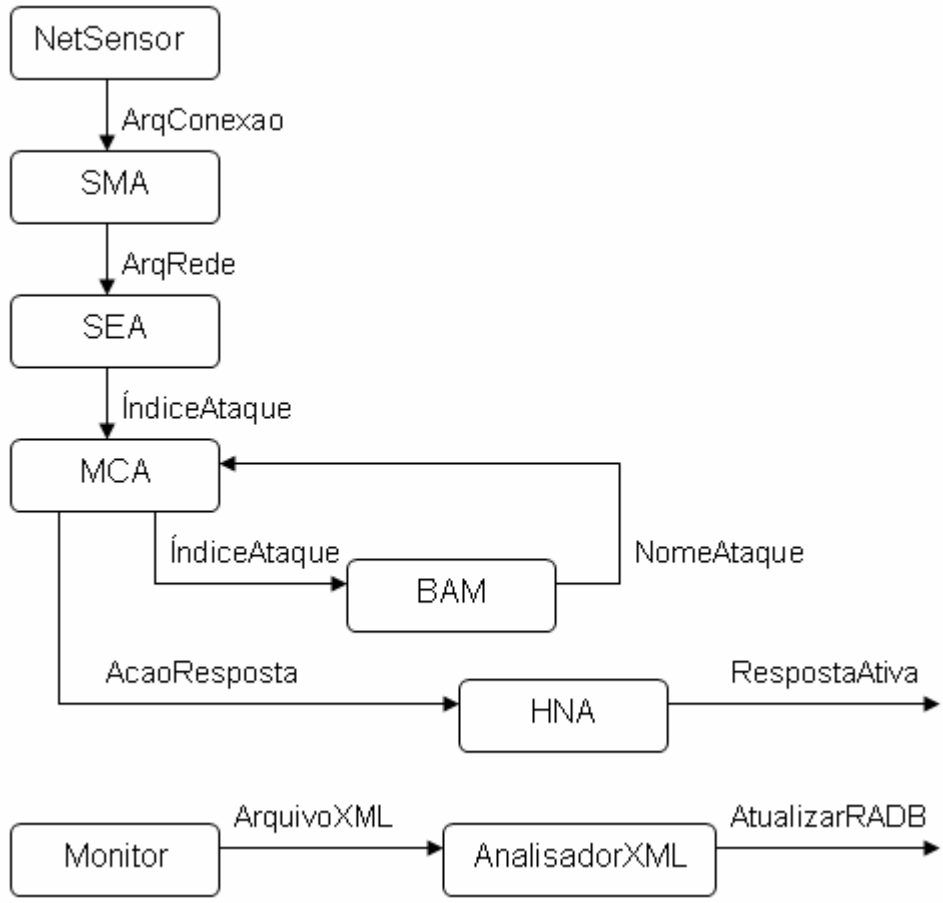


Figura – 6.1. Fluxo de informações do protótipo NIDIA.

6.1 A interação dos Agentes Tarefa com os Agentes Utilitários

Os agentes utilitários fornecem a infra-estrutura de suporte para toda a sociedade de agentes e esta seção mostra a comunicação entre esses agentes e os agentes tarefa.

Quando os agentes são criados, eles possuem apenas o conhecimento das tarefas que devem executar e das pré-condições para a execução dessas tarefas.

Para que as pré-condições sejam satisfeitas, os agentes comunicam-se com o agente "Facilitador" buscando a informação de qual agente poderá supri-lo com os insumos necessários.

O comportamento padrão do agente "Facilitador" é recuperar uma lista dos agentes a partir do Agente Servidor de Nomes, e em seguida enviar uma mensagem a cada agente requisitando detalhes das habilidades que são capazes de executar.

A Figura 6.2 mostra a interface gráfica do Agente Monitor, onde se observa na caixa de entrada (*Mail-In*:1) uma mensagem do agente "Facilitator0" solicitando informações sobre as habilidades do Agente Monitor. Na caixa de saída (*Mail Out*:1) está a resposta enviada em que o Agente Monitor informa que é capaz de produzir um fato do tipo "ArquivoXml". Este processo repete-se para todos os agentes da sociedade.

Deste ponto em diante, os agentes são capazes de descobrir, por intermédio do "Facilitador", quais agentes irão produzir os fatos que servem de pré-condição para a execução de suas tarefas. Entretanto para a comunicação direta entre agentes, eles necessitam conhecer a localização dos seus parceiros, conhecimento que é adquirido utilizando o Agente Servidor de Nomes.

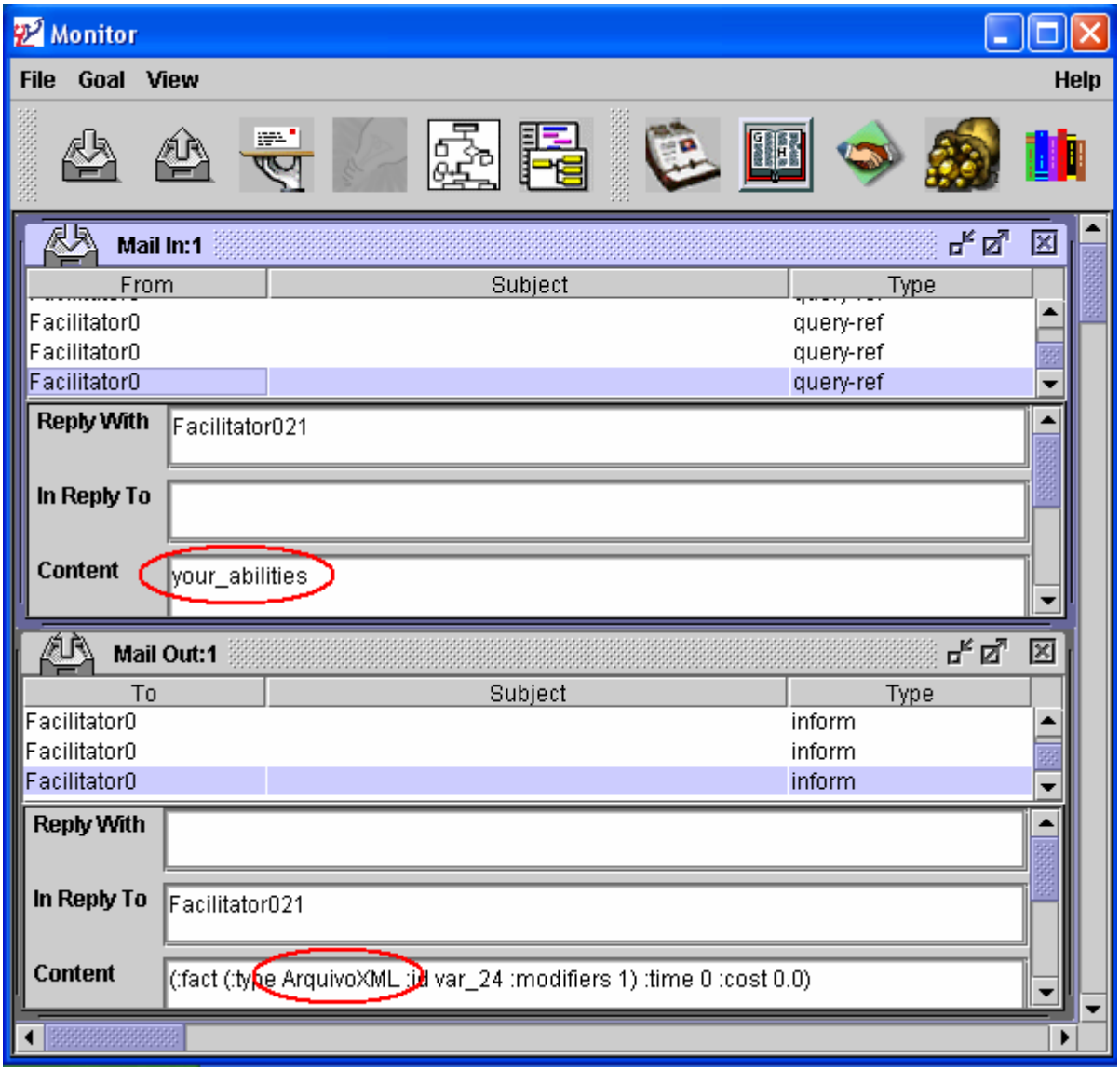


Figura – 6.2. Interface Gráfica do Agente Monitor

Observa-se na Figura 6.3 que o Agente Controlador Principal (MCA) envia uma requisição ao Agente Servidor de Nomes em busca do endereço do agente *Honey Net* (HNA) (caixa de saída *Mail Out*:1). O Agente Servidor de Nomes responde a essa solicitação informando que o endereço IP do HNA é 192.168.0.1 (caixa de entrada *Mail In*:1).

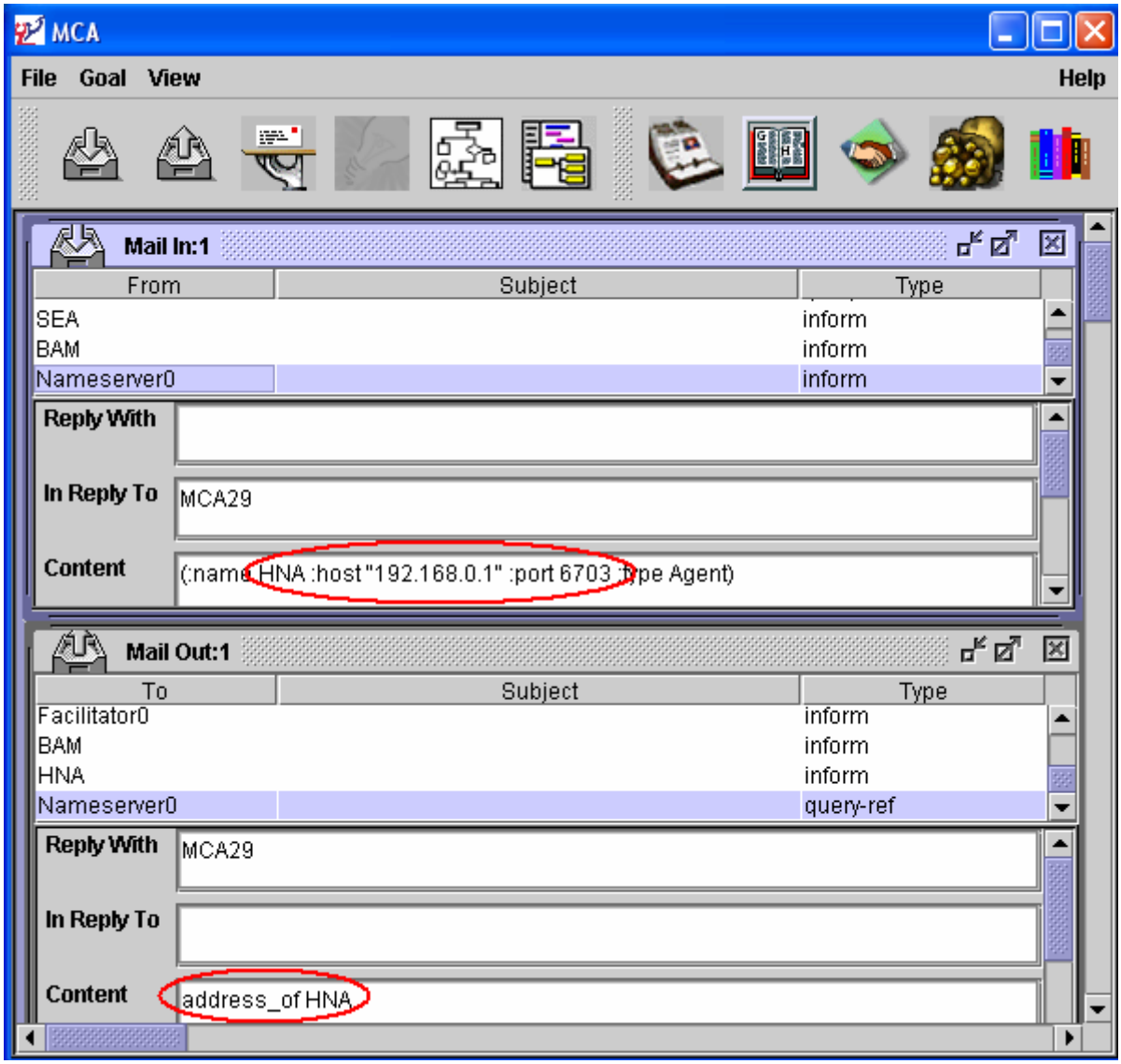


Figura – 6.3. Interface Gráfica do Agente Controlador Principal - MCA.

6.2 A Troca de Informações entre os Agentes do Protótipo

De posse de todo o conhecimento necessário sobre os outros agentes, ou seja, habilidades e localização, a sociedade está pronta para atingir os seus objetivos.

No caso deste protótipo, o arquivo RADB.txt deverá ser atualizado com os alertas mais recentes emitidos pelo CSIRT. Para alcançar esse objetivo, o Agente Monitor ao ser iniciado, envia uma requisição ao *Web service* do CSIRT contendo o número do último alerta recebido.

Inicialmente o arquivo RADB.txt possui apenas uma linha com o número um. O Agente Monitor lê esta linha e envia uma requisição contendo o número um. O *Web service* do CSIRT, recebe essa requisição e compara o seu conteúdo, isto é, o número um, com a identificação dos alertas emitidos que se encontram no arquivo Alertas.txt, como mostra a Figura 6.4.

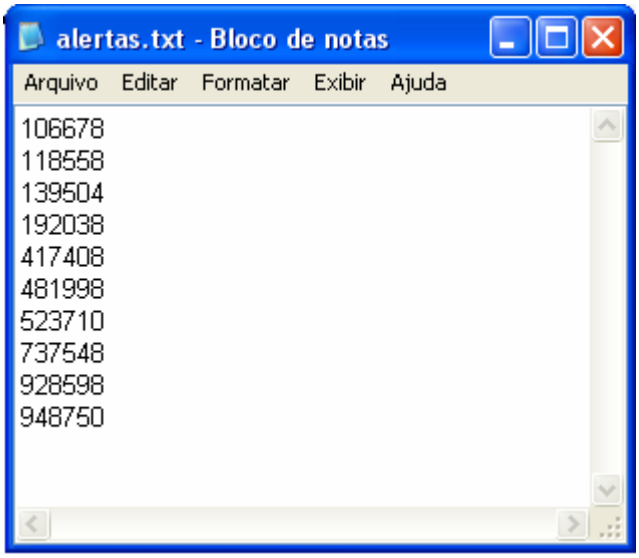


Figura – 6.4. Arquivo de Alertas Emitidos pelo CSIRT.

Como os alertas emitidos pelo CSIRT possuem números de identificação maiores que o presente na requisição enviada pelo Agente Monitor, uma mensagem de resposta é criada e enviada ao Agente Monitor contendo os dez alertas emitidos na forma de arquivos XML.

Ao receber a resposta, o Agente Monitor lê os anexos, grava os arquivos 106678.xml, 118558.xml, 139504.xml, 192038.xml, 417408.xml, 481998.xml, 523710.xml, 737548.xml, 928598.xml e 948750.xml no disco local, e envia uma mensagem ao Agente AnalisadorXML contendo a identificação dos novos alertas recebidos. Ao receber esta mensagem o Agente AnalisadorXML atualiza o arquivo RADB.txt com a identificação dos novos alertas recebidos.

A Figura 6.5 mostra a caixa de envio de mensagens do Agente Monitor, na qual observa-se a mensagem enviada ao AnalisadorXML contendo a identificação dos novos alertas.

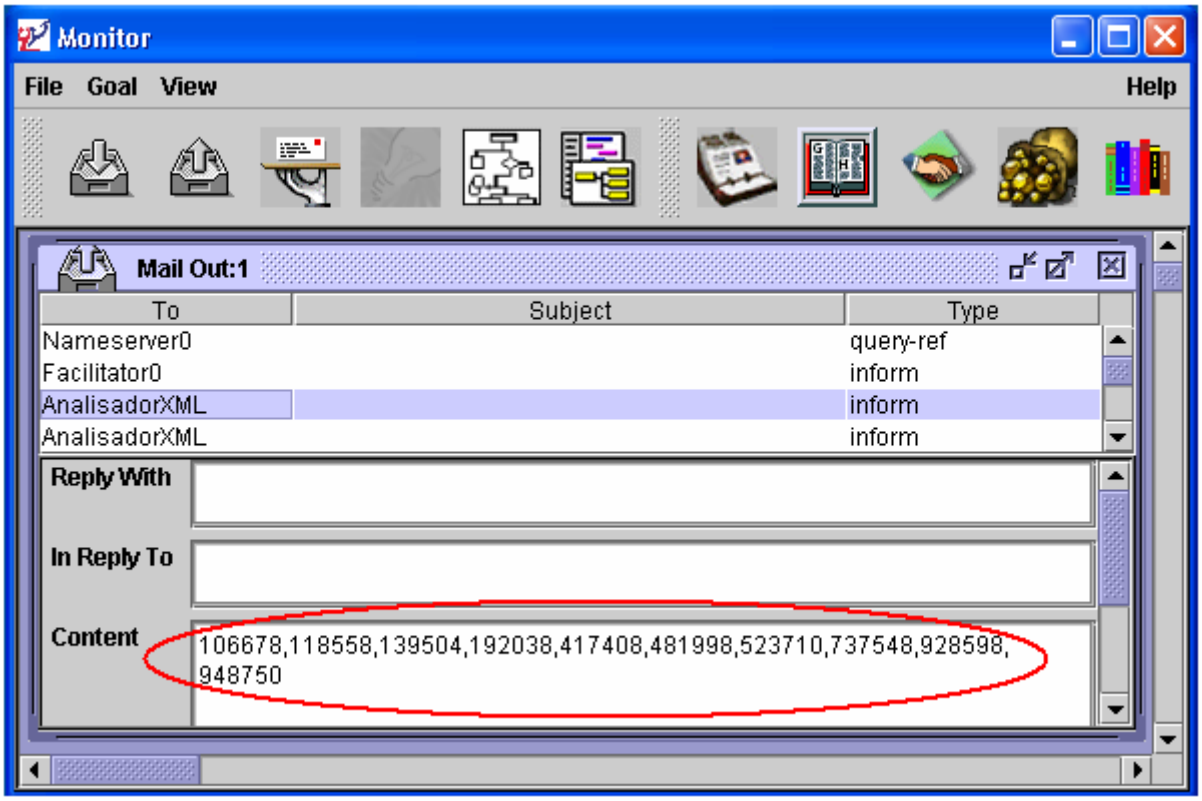


Figura – 6.5. Caixa de Saída (*Mail Out*:1) do Agente Monitor.

Após o recebimento dessa mensagem, o Agente AnalisadorXML atualiza o arquivo RADB.txt, como pode ser observado na Figura 6.6.

Figura – 6.6. Arquivo RADB.txt atualizado.

De posse desses alertas, o protótipo do NIDIA está pronto para responder caso uma tentativa de invasão que explore as vulnerabilidades descritas seja detectada.

A Figura 6.7 mostra a caixa de entrada (*Mail In*:1) do Agente de Monitoramento do Sistema (SMA) onde é possível observar diversas mensagens enviadas pelo Agente NetSensor. Essas mensagens possuem o conteúdo dos datagramas IP capturados na rede monitorada.

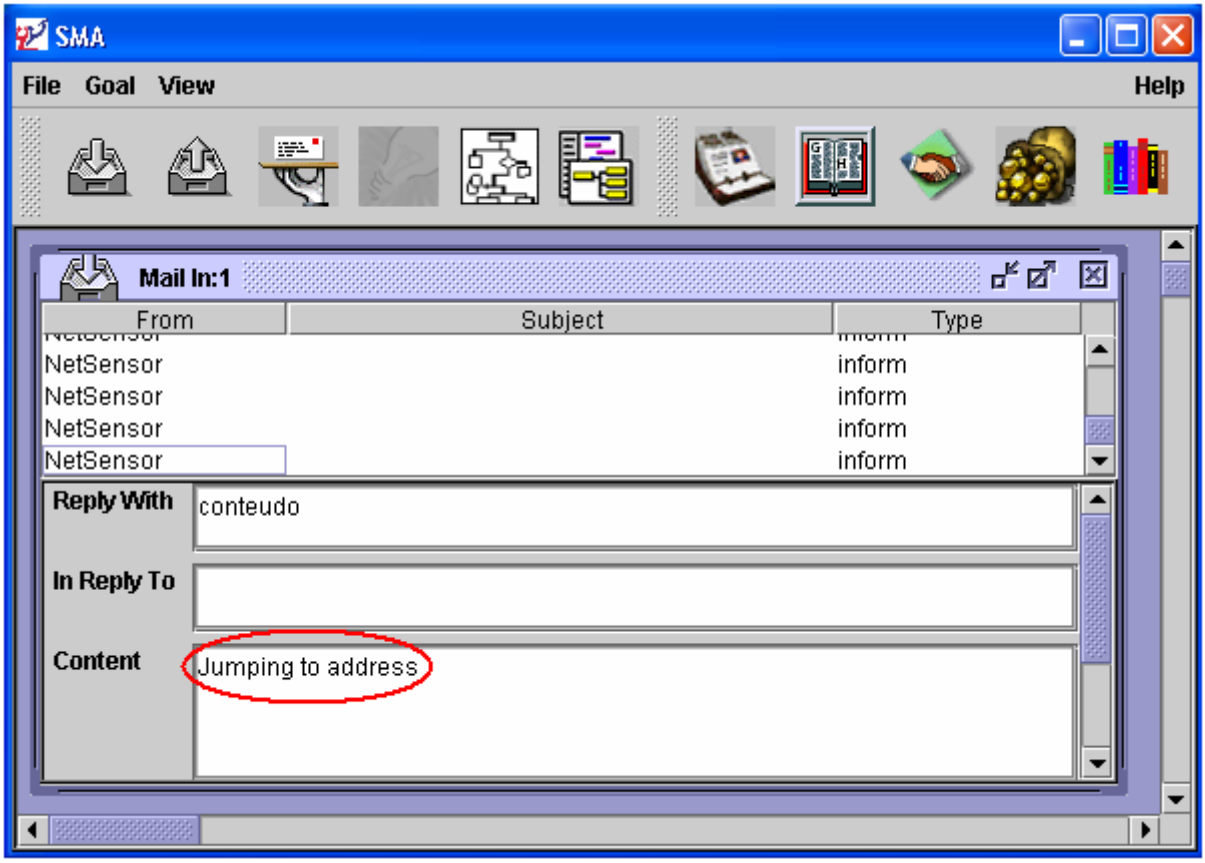


Figura – 6.7. Caixa de Entrada do Agente de Monitoramento - SMA.

O SMA é responsável por formatar os dados recebidos, na forma de um vetor de contagem de palavras-chave, e enviá-los ao Agente de Avaliação de Segurança do Sistema (SEA). A Figura 6.8 mostra a caixa de entrada do SEA com os dados enviados pelo SMA.

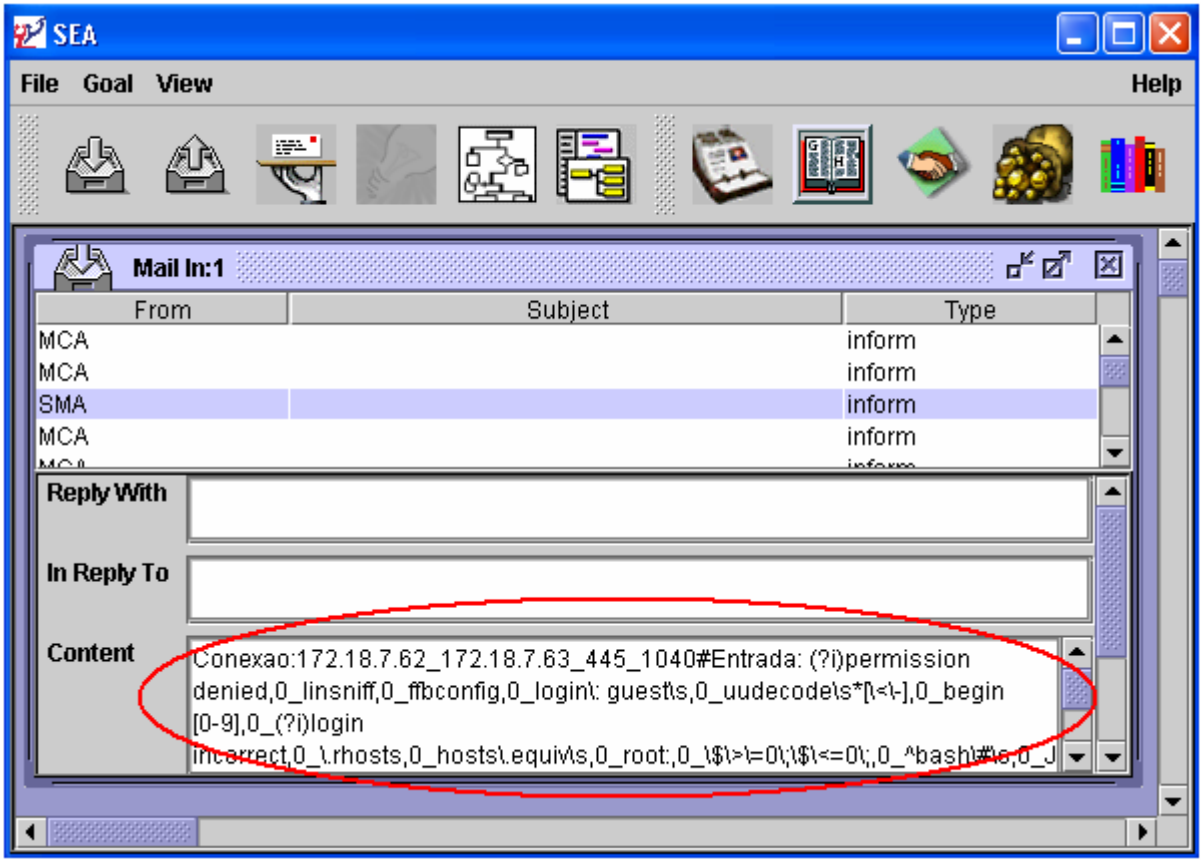


Figura – 6.8. Caixa de Entrada do Agente de Avaliação de Segurança do Sistema - SEA.

O SEA repassa os dados recebidos para uma rede neural em Java que foi baseada na rede *Multilayer Perceptorn* (MLP), com algoritmo de aprendizagem *BackPropagation*, encontrada em [10], previamente treinada e recebe como resultado o grau de severidade apresentado pela conexão.

Em seguida, o SEA envia o grau de severidade fornecido pela rede neural para o Agente Controlador Principal (MCA), que irá requisitar ao Agente BAM (BA) a identificação do nome do ataque. De posse do nome do ataque essas informações são exibidas por meio de uma interface gráfica, como mostra a Figura 6.9. Nessa figura observa-se que a conexão identificada por 172.18.7.62_172.18.7.63_445_1036 possui um grau de severidade igual a 0.98 e foi identificado o ataque “Scripts_de_Ataque_Buffer_Overflow”.

No caso de uma invasão detectada cabe ao MCA consultar a Base de Dados de Ações de Respostas (RADB) e tomar as ações de resposta necessárias [41]. Neste protótipo as ações de resposta foram tomadas por meio do Agente *Honey Net* (HNA) que recebe do MCA as informações da conexão utilizada pelo atacante e altera as regras do *eTrust Firewall* bloqueando o acesso do invasor. A Figura 6.10 mostra inserção da regra identificada por “NIDIA001” que bloqueia o acesso da máquina de endereço IP 192.168.2.32 à máquina de endereço IP 192.168.2.24, porta 2401, protocolo TCP.

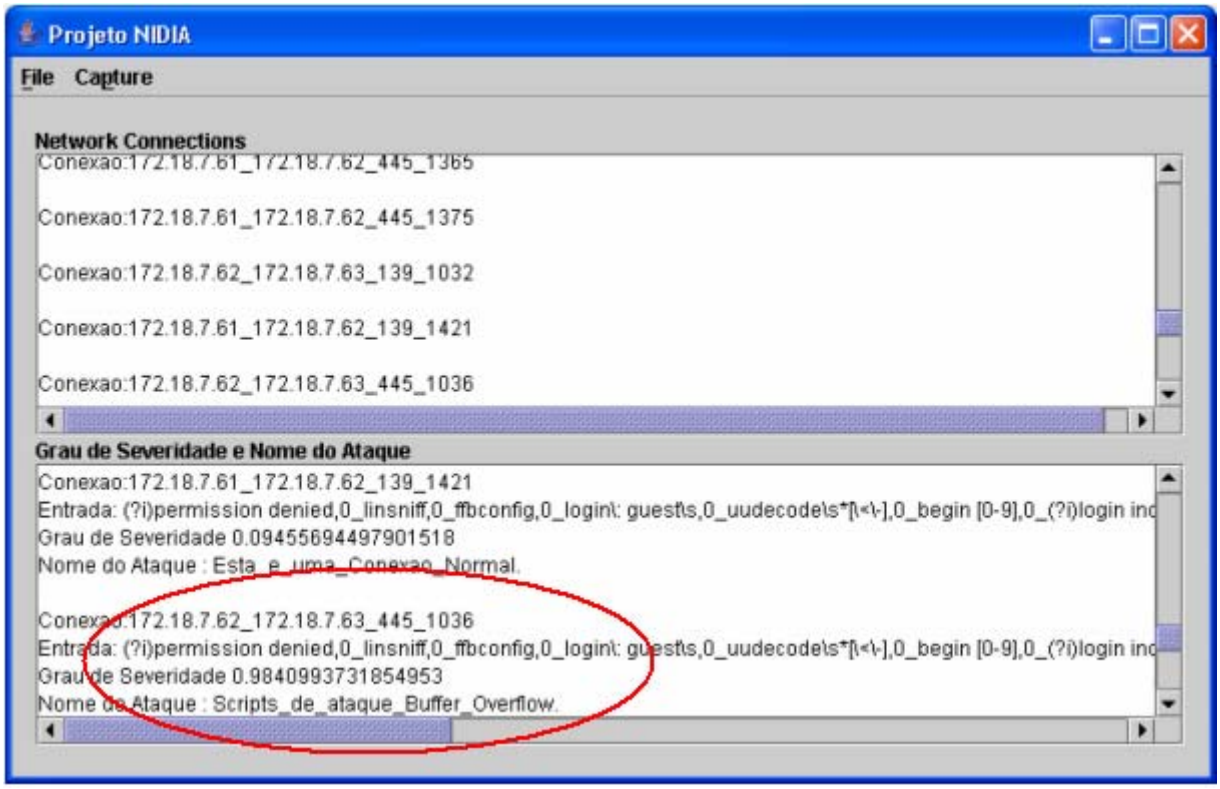


Figura – 6.9. Grau de Severidade de Conexões.

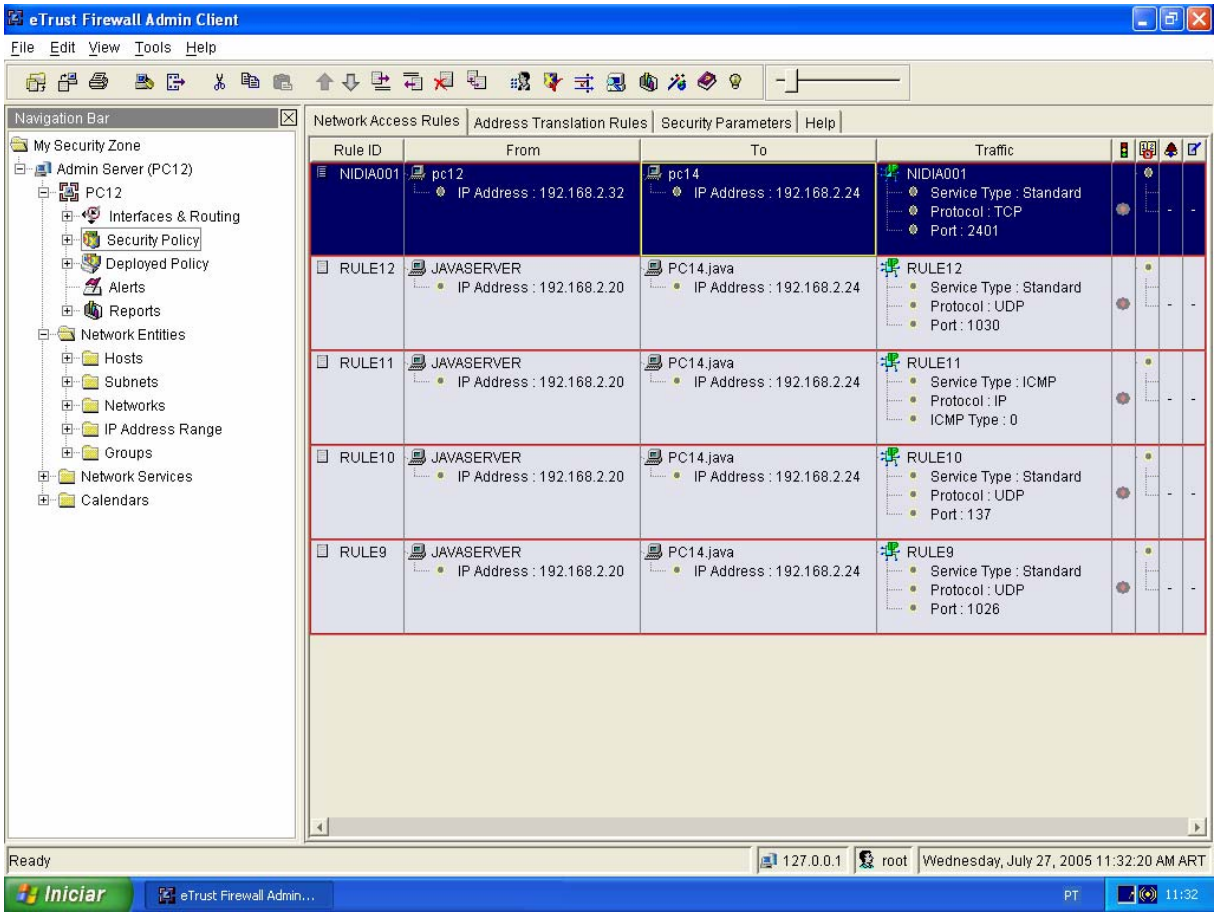


Figura – 6.10. Base de Regras do *eTrust Firewall*.

6.3 Conclusões

Os resultados obtidos com a utilização de *Web services* e da XML para a troca de informações entre os protótipos do SDI e do CSIRT demonstram o potencial dessas tecnologias para a atualização das bases de dados dos SDIs, não apenas por seus fabricantes, mas por toda uma comunidade preocupada em manter os seus sistemas o mais seguro possível.

7. CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Este capítulo apresenta as contribuições deste trabalho, conclusões sobre os resultados alcançados e sugestões para futuras pesquisas.

7.1 Contribuições do Trabalho

Com a descoberta diária de novas ameaças e vulnerabilidades nos sistemas computacionais existentes, a atualização dos SDIs torna-se uma preocupação constante.

Nesse trabalho foi proposto como manter o conjunto do ações de respostas dos SDIs e o mecanismo de detecção de ataques atualizados de forma automática.

No caso da atualização do conjunto do ações de respostas foi proposto um modelo de compartilhamento de dados entre CSIRTs e SDIs, com a adição do módulo de compartilhamento de informações ao CIDF. Outra contribuição foi o formato de dados para os alertas de segurança emitidos pelos CSIRTs baseado no formato Common Alerting Protocol (CAP).

Também foi proposta a utilização do modelo de espaço vetorial para a obtenção dos nomes dos ataques em substituição ao proposto por SANTOS & NASCIMENTO [41].

E finalmente um protótipo foi implementado com a utilização da ferramenta Zeus, *Web services* e XML. Para que esse protótipo pudesse realizar ações de resposta foi feita a sua integração ao protótipo do Agente *Honey Net* (HNA) desenvolvido por OLIVEIRA,NASCIMENTO & ABDELOUAHAB [32].

Para a atualização do mecanismo de detecção de ataques foi proposta uma arquitetura multiagente utilizando a tecnologias do campo da recuperação e filtragem de informação e de *Web services*.

7.2 Considerações Finais

Neste trabalho constatou-se a falta de padrões para o compartilhamento de informações que possibilitem a melhoria da segurança das redes de forma automática. Tanto o processo de envio de notificações de incidentes aos CSIRTs quanto a tomada de ações que previnam ou interrompam novos ataques ainda dependem da intervenção humana e a automação desses processos poderia auxiliar os administradores a manter suas redes mais seguras.

A utilização da plataforma Zeus para o desenvolvimento do protótipo mostrou-se muito simples e eficiente, permitindo que todos os esforços fossem concentrados na implementação das responsabilidades de cada agente. Entretanto o desenvolvimento dessa plataforma foi interrompido sendo necessária a definição de uma nova plataforma para o desenvolvimento dos agentes.

A implementação do *Web service* para a troca de informações na forma de documentos XML, ocorreu de forma rápida e eficiente, fato amplamente facilitado pela farta documentação disponível.

Outras pesquisas estão em andamento e têm como finalidade propor soluções para as questões de tolerância a falhas nos agentes, comunicações seguras com o auxílio de criptografia e assinatura digital, assim como uma proposta para SDIs em ambientes distribuídos.

7.3 Trabalhos Futuros

A seguir, apresenta-se uma série de propostas para trabalhos futuros e para o aperfeiçoamento este trabalho:

- Implementar um protótipo para a atualização do mecanismo de detecção de ataques;
- Implementar no Agente AnalisadorXML a capacidade de analisar os documentos XML e armazenar o seu conteúdo em um banco de dados;
- Utilizar outras plataformas para a construção de *Web services* para verificar a portabilidade entre plataformas;
- Implementar no Agente Controlador Principal (MCA) rotinas que permitam a seleção de ações de respostas de acordo com as informações existentes na Base de Dados de Ações de Respostas (RADB). Também faz-se necessária a implementação da Base de Dados de Estratégias (STDB) para que o MCA possa recomendar ações de resposta aos agentes *Honey Net* (HNA) ou Sistema Operacional (SOA);
- Expandir a capacidade de respostas do HNA e implementar o SOA e o Agente de Avaliação de Severidade (SAA);
- Apresentar uma proposta para as funções de segurança quando da utilização de *Web services*: criptografia e assinatura digital;
- Propor um modelo para a automação do processo de envio de notificações de incidentes de segurança que são a fonte de informação para todo o trabalho desempenhado pelos CSIRTs;

- Utilizar outras plataformas de construção de agentes para a implementação do protótipo, como o *Java Agent Development Framework* (JADE).

REFERÊNCIAS

[1] ALLEN, Julia et al. **State of the Practice of Intrusion Detection Technologies.** Technical Report Tr-99-028, Carnegie Melon University, Pittsburgh, PA, USA , 1999.

[2] ARMSTRONG, Eric et al. **The Java Web Services Tutorial.** Addison-Wesley 2002.

[3] ASAKA, M.; OKAZAWA, S.; TAGUCHI, A. **A Method of Tracing Intruders by Use of Mobile Agent.** Proceedings of the 9th Annual Internetworking Conference (INET`99), San Jose, California, USA, 1999.

[4] BACE, Rebecca Gurley. **Intrusion Detection.** Macmillan Technical Publishing, 1999.

[5] BACE, Rebecca; MELL, Peter. **Intrusion Detection Systems**. National Institute of Standards and Technology Special Publication SP800-31, Gaithersburg, MD, USA, 1999.

[6] BAEEZA-YATES, Ricardo; RIBEIRO-NETO, Berthier. **Modern Information Retrieval.** 1st edition, Addison Wesley. 1999.

[7] BALASUBRAMANIYAN, Jai et al. **An Architecture for Intrusion Detection using Autonomous Agents.** Coast TR 98-05. Department of Computer Sciences, Purdue University, 1998.

[8] BELKIN, N J.; CROFT, W. Bruce. **Information filtering and information retrieval: two sides of the same coin?** Communications of the ACM, Special issue on information filtering, December 1992, Volume 35, Issue 12.

[9] BELKIN, Nicholas J. and Croft, W. Bruce. **Retrieval techniques.** In Annual Review of Information Science and Technology, M.E. Williams, Ed. Chapt. 4, pp. 109-145. Elsevier, 1987.

[10] BIGUS, Joseph P.; BIGUS, Jennifer. **Constructing Intelligent Agents using Java.** 2. ed. Wiley Computer Publishing, 2001.

[11] BOOCH, Grady; RUMBAUGH James; JACOBSON, Ivar. **The Unified Modeling Language Reference Manual.** Addison-Wesley Professional; Bk&CD-Rom edition. 1998.

[12] BOOTH, D. et al. **Web Services Architecture***.* World Wide Web Consortium, 2004.

[13] BOURGEOIS, P. et al. **EISPP Common Advisory Format Description.** European Information Security Promotion Programme, 2004.

[14] BRAY, T. et al. **Extensible Markup Language (XML) 1.1**. World Wide Web Consortium, 2004.

[15] CERT COORDINATION CENTER. **CERT Coordination Center Frequently Asked Questions.** Disponível em <http://www.cert.org/csirts/csirt_faq.html>. Acesso em 15 de março 2005.

[16] CERT COORDINATION CENTER. **CERT Coordination Center Statistics.** Disponível em <http://www.cert.org/stats/cert_stats.html>. Acesso em 10 de setembro 2005.

[17] CHOPRA, V. et al. **Professional Apache Tomcat 5**. Wiley Publishing, Inc., Indianapolis, Indiana, USA, 2004.

[18] CHRISTEY, Steven M. et al. **The Development of a Common Vulnerabilities and Exposures List.** Second International Workshop on Recent Advances in Intrusion Detection, Purdue University, West Lafayette, Indiana, USA, 1999.

[19] CIDF WORKING GROUP. **The Common Intrusion Detection Framework Architecture**. Draft CIDF, 1998.

[20] COLLIS, J.; NDUMU, D. **Zeus Role Modelling Guide Release 1.02.** British Telecommunications, 2000.

[21] CROSBIE, M.; SPAFFORD, E. H. **Defending a Computer System using Autonomous Agents.** Coast TR 95-02. Department of Computer Sciences, Purdue University, 1995.

[22] DAVIS, R.; SMITH, R.G. **Negotiation as a metaphor for distributed problem solving**. Artificial Intelligence 20, p. 63-109, 1983.

[23] DIAS, R. A.; NASCIMENTO, E. **Um Modelo de Atualização Automática do Mecanismo de Detecção de Ataques de Rede para Sistemas de Detecção de Intrusão.** Proceedings of V Simpósio de Segurança em Informática. São José dos Campos, Brasil, 2003.

[24] FRINCKE, D. et al. **A Framework for Cooperative Intrusion Detection.** Proceedings of the 21st National Information Systems Security Conference, pp. 361-373, 1998.

[25] HAYKIN, Simon. **Neural Networks - A Comprehensive Foundation**. Prentice Hall - Upper Saddle River, New Jersey, 2nd ,1999.

[26] KOSSAKOWSKI, Klaus-Peter et al. **Responding to Intrusions.** Carnegie Melon University, Pittsburgh, PA, USA, 1999.

[27] LIMA, C. F. L. et al. **The NIDIA Project Network Intrusion Detection System based on Intelligent Agents.** Proceedings of Tenth Latin-Ibero-American Congress on Operations Research and Systems. Mexico City, Mexico, pp. 212-217, 2000.

[28] LOPES, E. N.; SILVA FILHO, M. V.; PEREIRA, R. S. **Hacker Curso Completo.** Rio de Janeiro, Book Express, 2002.

[29] MASSACHUSETTS INSTITUTE OF TECHNOLOGY. **Data Sets Overview.** Disponível em <http://www.ll.mit.edu/IST/ideval/data/data_index.html>. Acesso em 25 de janeiro 2004.

[30] MITRA, N. **SOAP Version 1.2 Part 0: Primer.** World Wide Web Consortium, 2003.

[31] NIELSEN, Henrik Frystyk; RUELLAN, Hervé. **SOAP 1.2 Attachment Feature.** World Wide Web Consortium, 2004.

[32] OLIVEIRA, Antonio Alfredo Pires ; NASCIMENTO, Edson ; ABDELOUAHAB, Z. **Using Honeypots And Intelligent Agents In Security Incident Responses And Investigation Of Suspicious Actions In Interconnected Computer Systems**. Proceedings of the E-Crime And Computer Evidence Conference Mônaco, 2005.

[33] OLIVEIRA, Ismênia Ribeiro de. **Um Sistema de Padrões Baseados em Agentes para a Modelagem de Usuários e Adaptação de Sistemas**. 2004. 123 f. Dissertação (Mestrado em Engenharia de Eletricidade) – UFMA.

[34] Organization for the Advancement of Structured Information Standards. **Common Alerting Protocol, v. 1.0.** OASIS Standard 200402, 2004.

[35] PESTANA Jr., Fernando A.; ABDELOUAHAB, Zair; NASCIMENTO, Edson. **Proposal of Model and Message Format for Sharing Information between CSIRTs and IDSs.** Proceedings of 14th International Congress on Computing. México Federal District, México, 2005.

[36] PESTANA Jr., Fernando A.; ABDELOUAHAB, Zair; NASCIMENTO, Edson. **Proposta de Arquitetura Multiagente para a Atualização do Mecanismo de Detecção de Ataques de SDIs.** Proceedings of IADIS Conferencia Ibero Americana WWW/Internet 2005, Lisboa, Portugal, 2005.

[37] PICKEL, Jed; DANYLIW, Roman. **Enabling Automated Detection of Security Events that affect Multiple Administrative Domains.** Carnegie Melon University Thesis, 2001.

[38] RUFINO, N. M. de O. **Segurança Nacional: Técnicas e Ferramentas de Ataque e Defesa de Redes de Computadores.** São Paulo: Novatec Editora Ltda., 2002.

[39] RUSSEL, Stuart; NORVIG; Peter. **Artificial Intelligence: A Modern Approach.** Prentice-Hall, Inc. 1995.

[40] SALTON, G. and McGILL, M.J. **Introduction to Modern Information Retrieval**. McGraw-Hill, 1983.

[41] SANTOS, G. de L. F.; NASCIMENTO, E. **An Automated Response Approach for Intrusion Detection Security Enhancement**. Proceedings of VII International Conference on Software Engineering and Applications. Marina Del Rey, California, USA, 2003.

[42] SILVA JUNIOR, Geovanne Bezerra da. **Padrões Arquiteturais para o Desenvolvimento de Aplicações Multiagente.** Dissertação (Mestrado em Ciência da Computação) – Curso de Pós-Graduação em Engenharia de Eletricidade, Universidade Federal do Maranhão.

[43] SOBIREY, M.; RICHTER, B.; KÖNIG, H. **The Intrusion Detection System AID.** Architecture, and experiences in automated audit analysis, in Horster, P. (ed.): Communications and Multimedia Security II, Proc. of the IFIP TC6 / TC11 International Conference on Communications and Multimedia Security, Essen, Germany, Sept. 1996, Chapman & Hall, London, 278 - 290

[44] SPAFFORD, Eugene H; ZAMBONI, Diego. **Intrusion detection using autonomous agents**. Computer Networks, Volume 34, Issue 4, October 2000, Pages 547-570.

[45] THE INTERNET ENGINEERING TASK FORCE. **The Incident Object Description Exchange Format Data Model and XML Implementation.** IETF Internet-Draft, November 2004.

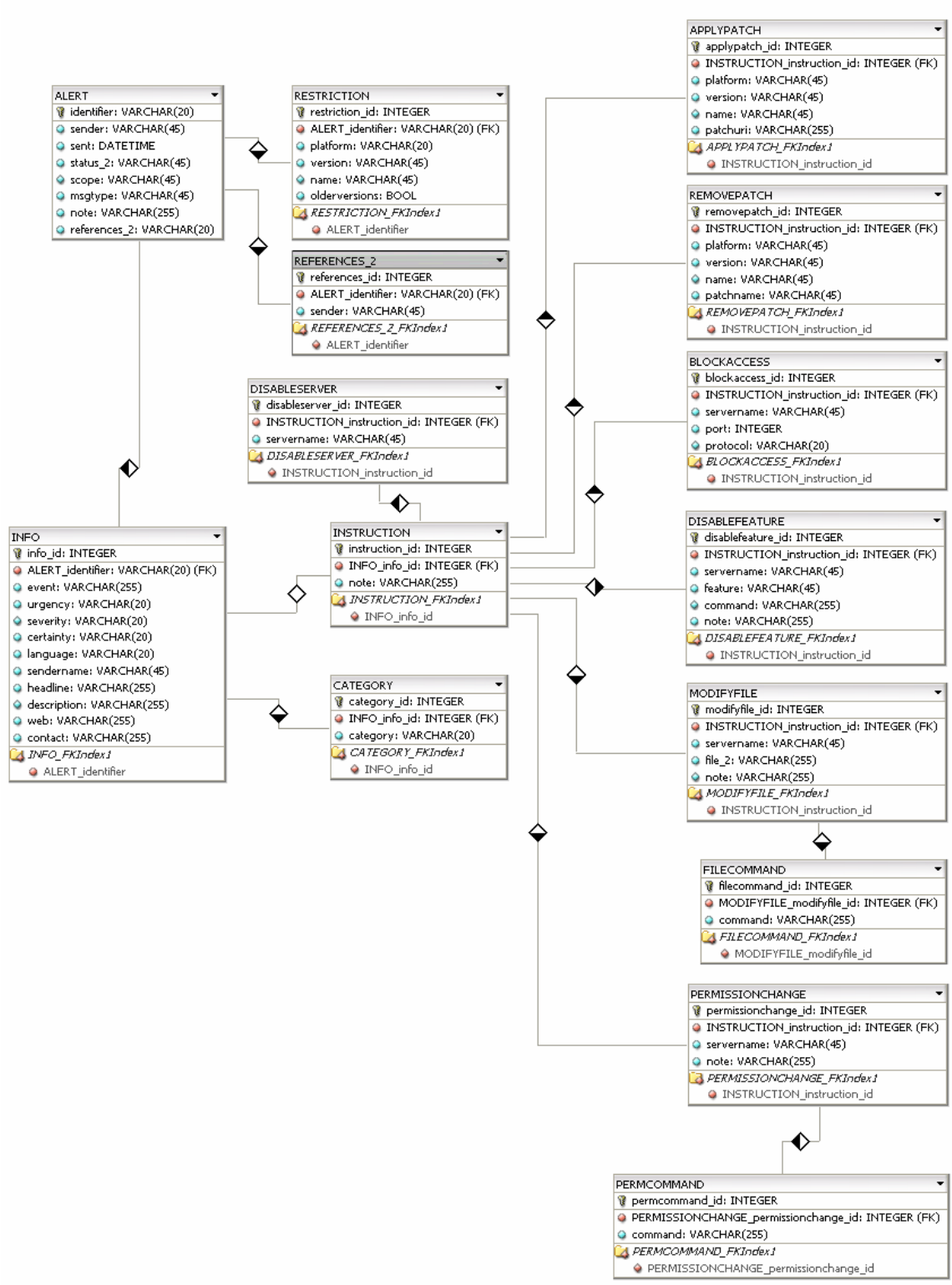
[46] THE INTERNET ENGINEERING TASK FORCE. **The Intrusion Detection Message Exchange Format.** IETF Internet-Draft, January 2005.

[47] WEISS, Gerhard. **Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence.** The MIT Press - London, England, 1999.

[48] WHITE, G. B.; FISH, E. A.; POOCH, U.W. **Cooperating Security Managers: A Peer-based Intrusion Detection System.** IEEE Network, vol 10 (1), 1996, pp. 20-23.

APÊNDICES

APÊNDICE - A ESQUEMA DA BASE DE DADOS DE AÇÕES DE RESPOSTAS - RADB.



APÊNDICE B – EXEMPLOS DE MENSAGENS DE ALERTAS EMITIDAS PELO CERT®/CC QUE FORAM CONVERTIDAS PARA O FORMATO DE DADOS PROPOSTO NESTE TRABALHO

1. Nota de Vulnerabilidade CAN-2004-0459

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!-- UNIVERSIDADE FEDERAL DO MARANHÃO -->
<!-- Projeto NIDIA  -->
<!-- A XML alert file -->
<!-- AUTOR: Fernando Augusto Pestana Junior-->
<alert xmlns:cap="http://www.incident.com/cap/1.0">
      <identifier>CAN-2004-0459</identifier>
      <sender>cert@cert.org</sender>
      <sent>2004-05-13T02:39:32</sent>
      <status>Actual</status>
      <scope>Restricted</scope>
      <restriction name="IEEE 802.11 Wireless Network" />
      <msgtype>Alert</msgtype>
      <info>
            <category>Internet</category>
            <event>US-CERT Vulnerability Note</event>
            <urgency>Immediate</urgency>
            <severity>Severe</severity>
            <certainty>Very Likely</certainty>
            <senderName>US-CERT United States Computer Emergency Readiness
                  Team
            </senderName>
            <headline>IEEE 802.11 wireless network protocol DSSS CCA
                  algorithm vulnerable to denial of service
            </headline>
            <description>IEEE 802.11 wireless network protocols use a Clear
            Channel Assessment (CCA) algorithm to determine whether or not
            the radio frequency (RF) channel is clear so that a device on
            the network can transmit data. The CCA algorithm used in
            conjunction with Direct Sequence Spread Spectrum (DSSS)
            transmission is vulnerable to an attack in which a specially
            crafted RF signal will cause the algorithm to conclude that the
            channel is busy, so that no device in range of the signal will
            transmit data. This type of signal is sometimes called
            "jabber." The attacker must be actively transmitting a signal
            and within range to affect wireless devices.
            This vulnerability is more thoroughly documented in AusCERT
            Advisory AA-2004.02. AusCERT notes that devices that use 802.11
            and DSSS transmission encoding are affected:
            Wireless hardware devices that implement IEEE 802.11 using a
            DSSS physical layer. Includes IEEE 802.11, 802.11b and low-
      speed (below 20Mbps) 802.11g wireless devices. Excludes IEEE
      802.11a and high-speed (above 20Mbps) 802.11g wireless devices.
            This is not an implementation vulnerability; any 802.11 DSSS
            device, including wireless network cards and access points, is
            vulnerable. WEP, WPA, or other WLAN security features will not
            protect vulnerable devices.
            It is worth noting that since 802.11 management frames are
      weakly authenticated (VU#391513), it is possible for an
      attacker to DoS an 802.11 network by sending de-authentication
```

```
      or failed authentication frames using the spoofed MAC and IP
      addresses of an access point. Tools that perform this type of
      attack are publicly available (FATA-jack, airjack, wlan-jack).
      While the CCA attack may be less expensive for an attacker,
      both attacks have similar characteristics (active attacker in
      range using commodity hardware) and impacts (DoS while attacker
      is in range and active). Wireless networks in general are also
      subject to RF interference or jamming. Careful consideration
      should be given to the use of commercial grade wireless
      networks for applications that require high availability.
      </description>
            <instruction>
                  <note>A complete solution is not available for 802.11
                  DSSS devices. As noted by AusCERT, "...a comprehensive
                  solution, in the form of software or firmware upgrade, is
                  not available for retrofit to existing devices.
                  Fundamentally, the issue is inherent in the protocol
                  implementation of IEEE 802.11 DSSS." Sites running
                  wireless networks should consider security and
                  availability requirements, network design, and the
                  workarounds listed below.
                  Use non-DSSS 802.11 protocols
                  802.11 protocols that use frequency hopping spread
                  spectrum (FHSS) or orthogonal frequency division
                  multiplexing (OFDM) are not affected by this
                  vulnerability. 802.11a uses OFDM, 802.11 can use FHSS,
                  and 802.11g can use OFDM. Note that this workaround will
                  not provide protection against management frame spoofing
                  or RF attacks.
                  Constrain wireless networks
                  Depending on the application and site infrastructure, it
                  may be possible to prevent attackers from getting in
                  range of 802.11 networks by using physical barriers
                  (walls, fences, elevation, etc.). In addition, different
                  building materials provide various degrees of shielding.
                  Do not rely on 802.11 for high availability
                  Due to the inherent vulnerabilities in 802.11 (VU#106678,
                  VU#391513, RF interference), do not deploy 802.11
                  networks for applications that require high availability
                  (e.g. safety, critical infrastructure).
                  </note>
            </instruction>
            <web>http://www.kb.cert.org/vuls/id/106678</web>
            <contact>mailto:cert@cert.org?Subject=VU%23106678
                  Feedback
            </contact>
      </info>
</alert>
```

2. Nota de Vulnerabilidade VU#118558

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!-- UNIVERSIDADE FEDERAL DO MARANHÃO -->
<!-- Projeto NIDIA  -->
<!-- A XML alert file -->
<!-- AUTOR: Fernando Augusto Pestana Junior-->
<alert xmlns:cap="http://www.incident.com/cap/1.0">
      <identifier>VU#118558</identifier>
      <sender>cert@cert.org</sender>
      <sent>2004-05-14T15:38:31</sent>
      <status>Actual</status>
      <scope>Restricted</scope>
      <restriction name="Java Runtime Environment"/>
      <msgtype>Alert</msgtype>
      <info>
            <category>Internet</category>
            <event>US-CERT Vulnerability Note</event>
            <urgency>Immediate</urgency>
            <severity>Severe</severity>
            <certainty>Very Likely</certainty>
            <senderName>US-CERT United States Computer Emergency Readiness
                  Team</senderName>
            <headline>Sun Java Runtime Environment vulnerable to
                  DoS</headline>
            <description>The Sun Java Runtime Environment provides the
            libraries and components necessary to run Java-based
            applications. There is a non-specific vulnerability in
            the Java Runtime Environment, which could allow an
            unauthenticated, remote attacker to cause the Java
            Virtual Machine to become unresponsive.
            </description>
            <instruction>
                  <note>According to Sun Security Alert 57555, this issue
                        has been addressed in the following releases:
                        Windows Production Releases
                        SDK and JRE 1.4.2_04 or later 1.4.2 releases
                        Solaris Operating Environment Releases
                        SDK and JRE 1.4.2_04 or later 1.4.2 releases
                        Linux Production Releases
                        SDK and JRE 1.4.2_04 or later 1.4.2 releases
                  </note>
                  <applyPatch platform="WINDOWS" name="SDK 1.4.2_04"
                  patchUri="http://java.sun.com/j2se/1.4.2/download.html"/>
                  <applyPatch platform="WINDOWS" name="JRE 1.4.2_04"
                  patchUri="http://java.sun.com/j2se/1.4.2/download.html"/>
                  <applyPatch platform="Solaris" name="SDK 1.4.2_04"
                  patchUri="http://java.sun.com/j2se/1.4.2/download.html"/>
                  <applyPatch platform="Solaris" name="JRE 1.4.2_04"
                  patchUri="http://java.sun.com/j2se/1.4.2/download.html"/>
                  <applyPatch platform="Linux" name="SDK 1.4.2_04"
                  patchUri="http://java.sun.com/j2se/1.4.2/download.html"/>
                  <applyPatch platform="Linux" name="JRE 1.4.2_04"
                  patchUri="http://java.sun.com/j2se/1.4.2/download.html"/>
            </instruction>
            <web>http://www.kb.cert.org/vuls/id/118558</web>
            <contact>mailto:cert@cert.org?Subject=VU%23118558
                  Feedback</contact>
      </info>
</alert>
```

3. Nota de Vulnerabilidade VU#139504

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!-- UNIVERSIDADE FEDERAL DO MARANHÃO -->
<!-- Projeto NIDIA  -->
<!-- A XML alert file -->
<!-- AUTOR: Fernando Augusto Pestana Junior-->
<alert xmlns:cap="http://www.incident.com/cap/1.0">
      <identifier>VU#139504</identifier>
      <sender>cert@cert.org</sender>
      <sent>2004-08-11T14:38:50</sent>
      <status>Actual</status>
      <scope>Restricted</scope>
      <restriction name="Sun Solaris X Display Manager"/>
      <msgtype>Alert</msgtype>
      <info>
            <category>Internet</category>
            <event>US-CERT Vulnerability Note</event>
            <urgency>Immediate</urgency>
            <severity>Severe</severity>
            <certainty>Very Likely</certainty>
            <senderName>US-CERT United States Computer Emergency Readiness
                  Team</senderName>
            <headline>Sun Solaris X Display Manager does not properly
                  handle invalid XDMCP requests</headline>
            <description>The X Display Manager (xdm(1)) is responsible for
                  managing collections of X displays from local or remote
                  servers using the X Display Manager Control Protocol
                  (XDMCP). The Sun Solaris X Display Manager contains a
                  denial-of-service vulnerability that could be triggered
                  by an invalid XDMCP packet.
            </description>
            <instruction>
                  <note>Apply patch
                        Sun has issued an advisory which addresses this
                        issue. For more information on patches available
                        for your system, please refer to Sun Security Alert
                        57619.
                  </note>
                   <applyPatch platform="SPARC" name="112785-38"
      patchUri="http://sunsolve.sun.com/search/pdownload.pl?target=112785-
      38method=h"/>
                  <applyPatch platform="x86" name="111845-03"
      patchUri="http://sunsolve.sun.com/search/pdownload.pl?target=111845-
      03method=h"/>
            </instruction>
            <web>http://www.kb.cert.org/vuls/id/139504</web>
            <contact>mailto:cert@cert.org?Subject=VU%23139504
            Feedback</contact>
      </info>
</alert>
```

4. Nota de Vulnerabilidade CAN-2004-0396

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!-- UNIVERSIDADE FEDERAL DO MARANHÃO -->
<!-- Projeto NIDIA  -->
<!-- A XML alert file -->
<!-- AUTOR: Fernando Augusto Pestana Junior-->
<alert xmlns:cap="http://www.incident.com/cap/1.0">
      <identifier>CAN-2004-0396</identifier>
      <sender>cert@cert.org</sender>
      <sent>2004-05-19T15:37:21</sent>
      <status>Actual</status>
      <scope>Restricted</scope>
      <restriction name="CVS Feature" version="1.12.7" olderVersions="Y"/>
      <restriction name="CVS Stable" version="1.11.15" olderVersions="Y"/>
      <msgtype>Alert</msgtype>
      <info>
            <category>Internet</category>
            <event>US-CERT Vulnerability Note</event>
            <urgency>Immediate</urgency>
            <severity>Severe</severity>
            <certainty>Very Likely</certainty>
            <senderName>US-CERT United States Computer Emergency Readiness
            Team</senderName>
            <headline>CVS contains a heap overflow in the handling of flag
            insertion</headline>
            <description>CVS is a source code maintenance system that is
            widely used by open-source software development projects.
            There is a heap memory overflow vulnerability in the way CVS
            handles the insertion of modified and unchanged flags within
            entry lines. When processing an entry line, an additional byte
            of memory is allocated to flag the entry as modified or
            unchanged.
            There is a failure to check if a byte has been previously
            allocated for the flag, which creates an off-by-one buffer
      overflow. By calling a vulnerable function several times and
      inserting specific characters into the entry lines, a remote
      attacker could overwrite multiple blocks of memory. The CVS
      server process is commonly started by the Internet services
      daemon (inetd) and run with root privileges. According to the
      e-matters security advisory, the following versions are
      affected: CVS    feature release &lt;= 1.12.7 CVS stable
      release &lt;= 1.11.15 </description>
            <instruction>
                  <note>Apply the appropriate patch or upgrade as specified
                  by your vendor. This issue has been resolved in Stable CVS
                  Version 1.11.16 and CVS Feature Version 1.12.8. Until
                  patches are available and can be applied, consider
                  disabling the CVS server.Block or restrict access to the
                  CVS server from untrusted hosts and networks. The CVS
                  server typically listens on 2401/tcp, but it may use
                  another port or protocol.Configure CVS server to run in a
                  restricted chroot) environment. Run CVS servers with the
                  minimum set of privileges required on the host file
                  system. Provide separate systems for development (write)
                  and public/anonymous (read-only) CVS access. Host
                  public/anonymous CVS servers on single-purpose, secured
                  systems.Note that some of these workarounds will only
                  limit the scope and impact of possible attacks.
                  </note>
```

```
                <disableServer>"CVS Server"</disableServer>
                 <applyPatch name="CVS Feature" version="1.12.7"
patchUri="http://ccvs.cvshome.org/servlets/NewsItemView?newsitemID=108" />
                 <applyPatch name="CVS Stable" Version="1.11.15"
patchUri="https://ccvs.cvshome.org/servlets/NewsItemView?newsitemID=107" />
                 <blockAccess serverName="CVS Server" port="2401"
                       protocol="tcp"/>
            </instruction>
            <web>http://www.kb.cert.org/vuls/id/192038</web>
            <contact>mailto:cert@cert.org?Subject=VU#192038
Feedback</contact>
      </info>
</alert>
```

5. Nota de Vulnerabilidade VU#417408

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!-- UNIVERSIDADE FEDERAL DO MARANHÃO -->
<!-- Projeto NIDIA  -->
<!-- A XML alert file -->
<!-- AUTOR: Fernando Augusto Pestana Junior-->
<alert xmlns:cap="http://www.incident.com/cap/1.0">
      <identifier>VU#417408</identifier>
      <sender>cert@cert.org</sender>
      <sent>2004-08-13T15:39:25</sent>
      <status>Actual</status>
      <scope>Restricted</scope>
      <restriction Version="2.0.8" name="JetboxOne"/>
      <msgtype>Alert</msgtype>
      <info>
            <category>Internet</category>
            <event>US-CERT Vulnerability Note</event>
            <urgency>Immediate</urgency>
            <severity>Severe</severity>
            <certainty>Very Likely</certainty>
            <senderName>US-CERT United States Computer Emergency Readiness
                  Team</senderName>
            <headline>JetboxOne may allow unauthorized users to execute
                  arbitrary code
            </headline>
            <description>JetboxOne, an open-source content management
            system, could allow an attacker with "AUTHOR" privileges to
            upload arbitrary files to the image folder via the upload image
            control. The vulnerability exists because the type of file
      being uploaded is not verified as a valid image file e.g. GIF,
      JPEG. Once uploaded, the attacker is then able to request the
      file, which will be interrpreted by the JetboxOne application.
      Based on the file type this may permit a malicious user to
      execute the arbitrary code on the compromised system.
      Currently, the vulnerability has been demonstrated on version
      2.0.8. However, it may also exist in previous versions, but
      they are as of yet untested. </description>
            <instruction>
                  <note>The CERT/CC is currently unaware of a practical
                  solution to this problem. </note>
            </instruction>
            <web>http://www.kb.cert.org/vuls/id/417408</web>
            <contact>mailto:cert@cert.org?Subject=VU%23417408
            Feedback</contact>
      </info>
</alert>
```

6. Nota de Vulnerabilidade CAN-2004-0747

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!-- UNIVERSIDADE FEDERAL DO MARANHÃO -->
<!-- Projeto NIDIA  -->
<!-- A XML alert file -->
<!-- AUTOR: Fernando Augusto Pestana Junior-->
<alert xmlns:cap="http://www.incident.com/cap/1.0">
      <identifier>CAN-2004-0747</identifier>
      <sender>cert@cert.org</sender>
      <sent>2004-09-17T16:09:21</sent>
      <status>Actual</status>
      <scope>Restricted</scope>
      <restriction version="2.0.50" name="Apache" olderVersions="S"/>
      <msgtype>Alert</msgtype>
      <info>
            <category>Internet</category>
            <event>US-CERT Vulnerability Note</event>
            <urgency>Immediate</urgency>
            <severity>Severe</severity>
            <certainty>Very Likely</certainty>
            <senderName>US-CERT United States Computer Emergency Readiness
            Team</senderName>
            <headline>Apache vulnerable to buffer overflow when expanding
            environment variables</headline>
            <description>The Apache HTTP Server is a freely available web
                  server that runs on a variety of operating systems
                  including Unix, Linux, and Microsoft Windows. The
                  ap_resolve_env() function is responsible for expanding
                  environment variables when parsing configurations files
                  such as .htaccess or httpd.conf. There is a vulnerability
                  in this function that could allow a local user to trigger
                  a buffer overflow.
                  The Apache Software Foundation notes that in order to
                  exploit this vulnerability, a local user would need to
                  install the malicious configuration file on the server
                  and force the server to parse this file.
            </description>
            <instruction>

            <note>Upgrade or apply patch as specified by your vendor. This
            issue is resolved in Apache version 2.0.51.</note>
            <applyPatch version="2.0.50" name="Apache"
      patchUri="http://apache.usp.br/httpd/httpd-2.0.51-win32-src.zip" />
            </instruction>
            <web>http://www.kb.cert.org/vuls/id/481998</web>
            <contact>mailto:cert@cert.org?Subject=VU%23481998
            Feedback</contact>
      </info>
</alert>
```

7. Nota de Vulnerabilidade VU#523710

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!-- UNIVERSIDADE FEDERAL DO MARANHÃO -->
<!-- Projeto NIDIA  -->
<!-- A XML alert file -->
<!-- AUTOR: Fernando Augusto Pestana Junior-->
<alert xmlns:cap="http://www.incident.com/cap/1.0">
      <identifier>VU#523710</identifier>
      <sender>cert@cert.org</sender>
      <sent>2004-06-24T11:43:01</sent>
      <status>Actual</status>
      <scope>Restricted</scope>
      <restriction name="Sun Solaris"/>
      <msgtype>Alert</msgtype>
      <info>
            <category>Internet</category>
            <event>US-CERT Vulnerability Note</event>
            <urgency>Immediate</urgency>
            <severity>Severe</severity>
            <certainty>Very Likely</certainty>
            <senderName>US-CERT United States Computer Emergency Readiness
                  Team</senderName>
            <headline>Sun Solaris patches may cause passwords to be logged
                  in clear text</headline>
            <description>Sun Microsystems released patches 112908-12 and
            115168-03 to address issues in kerberos.  There is a
            vulnerability in these patches that may result in user
            passwords being logged in clear text.According to the Sun
            Security Alert:This issue can occur on a Solaris system
            configured as a kerberos client with patch 112908-12 or
            115168-03 installed and any service using pam_krb5 as an
            "auth" module.
            </description>
            <instruction>
                  <note>Sun has issued an advisory which addresses this
                  issue. For more information on patches
                  available for your system, please refer to Sun Security
                  Alert: 57587.</note>
                  <removePatch platform= "SPARC" name="112908-12"/>
                  <removePatch platform= "x86" name="115168-03"/>
                  <applyPatch platform="SPARC" name="112908-16"
      patchUri="http://sunsolve.sun.com/search/pdownload.pl?target=112908-
16method=f"/>
                  <applyPatch platform="WINDOWS" name="115168-04"
      patchUri="http://sunsolve.sun.com/search/pdownload.pl?target=115168-
04method=f"/>
                  <disableFeature serverName="PAM" Feature="Debug"
                        command="egrep -e '[\\t ]*[^#].*pam_krb5.*debug'
                        /etc/pam.conf"/>
                  <disableFeature serverName="PAM" Feature="logging of
                        LOG_DEBUG level messages"
                        command="egrep -e '\*.debug|daemon.debug'
                        /etc/syslog.conf"/>
            </instruction>
            <web>http://www.kb.cert.org/vuls/id/523710</web>
            <contact>mailto:cert@cert.org?Subject=VU%23523710
                  Feedback</contact>
      </info>
</alert>
```

8. Nota de Vulnerabilidade VU#737548

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!-- UNIVERSIDADE FEDERAL DO MARANHÃO -->
<!-- Projeto NIDIA  -->
<!-- A XML alert file -->
<!-- AUTOR: Fernando Augusto Pestana Junior-->
<alert xmlns:cap="http://www.incident.com/cap/1.0">
      <identifier>VU#737548</identifier>
      <sender>cert@cert.org</sender>
      <sent>2004-04-14T11:22:23</sent>
      <status>Actual</status>
      <scope>Restricted</scope>
      <restriction name="Sun Solaris"/>
      <msgtype>Alert</msgtype>
      <info>
            <category>Internet</category>
            <event>US-CERT Vulnerability Note</event>
            <urgency>Immediate</urgency>
            <severity>Severe</severity>
            <certainty>Very Likely</certainty>
            <senderName>US-CERT United States Computer Emergency Readiness
                  Team</senderName>
            <headline>Sun Solaris SSH Daemon fails to properly log client
                  IP addresses</headline>
            <description>SSH is a program used to provide secure connection
            and communications between client and servers. Upon connecting
            to the service, the client's IP address is logged. There is a
            vulnerability in the Sun Solaris SSH Daemon that may cause it
            to inaccurately log the IP addresses of clients. When the SSH
            Daemon initializes, it reads configuration information from the
            sshd_config file. If this file contains the "ListenAddress"
            keyword configured in a specific way, SSH will fail to properly
            log client IP addresses.According to the Sun Security
            Advisory:A system is only affected by this issue if the sshd
            configuration file (sshd_config(4)) has the "ListenAddress"
            keyword configured as "0.0.0.0" which means to listen on only
            IPv4 (see inet(3SOCKET)) configured interfaces. To determine
            which interfaces on a system are configured to use IPv4 the
            following command can be run:$ ifconfig -a4 lo0:
            flags=1000849&lt;UP, LOOPBACK, RUNNING, MULTICAST, IPv4&gt; mtu
            8232 index 1 inet 127.0.0.1 netmask ff000000 eri0:
            flags=1000843&lt;UP, BROADCAST, RUNNING, MULTICAST, IPv4&gt;
            mtu 1400 index 2 inet 192.168.254.202 netmask ffffff00
            broadcast 192.168.254.255</description>
            <instruction>
                  <note>Sun has released an advisory which addresses this
                        issue. For more information on patches available
                        for your system, please refer to Sun Security
                        Alert: 57538.
                  </note>
                  <applyPatch platform="SPARC" name="113273-05"
      patchUri="http://sunsolve.sun.com/search/pdownload.pl?target=113273-
      05method=f"/>
                  <applyPatch platform="x86" name="114858-04"
      patchUri="http://sunsolve.sun.com/search/pdownload.pl?target=114858-
      04method=f"/>
                  <modifyFile serverName="SSH" file="/etc/ssh/sshd_config">
```

```
                        <command>"grep ^ListenAddress /etc/ssh/sshd_config
                        ListenAddress ::"
                        </command>
                        <command>"pkill -HUP sshd"</command>
                  </modifyFile>
            </instruction>
            <web>http://www.kb.cert.org/vuls/id/737548</web>
            <contact>mailto:cert@cert.org?Subject=VU%23737548
            Feedback</contact>
      </info>
</alert>
```

9. Nota de Vulnerabilidade CAN-2004-0800

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!-- UNIVERSIDADE FEDERAL DO MARANHÃO -->
<!-- Projeto NIDIA  -->
<!-- A XML alert file -->
<!-- AUTOR: Fernando Augusto Pestana Junior-->
<alert xmlns:cap="http://www.incident.com/cap/1.0">
      <identifier>CAN-2004-0800</identifier>
      <sender>cert@cert.org</sender>
      <sent>2004-08-25T14:26:18</sent>
      <status>Actual</status>
      <scope>Restricted</scope>
      <restriction platform="SPARC" version ="Solaris 8" name="CDE 1.4"
            olderVersions="Y"/>
      <restriction platform="SPARC" version ="Solaris 9" name="CDE 1.5"
            olderVersions="Y"/>
      <restriction platform="x86" version ="Solaris 8" name="CDE 1.4"
            olderVersions="Y"/>
      <restriction platform="x86" version ="Solaris 9" name="CDE 1.5"
            olderVersions="Y"/>
      <msgtype>Alert</msgtype>
      <info>
            <category>Internet</category>
            <event>US-CERT Vulnerability Note</event>
            <urgency>Immediate</urgency>
            <severity>Severe</severity>
            <certainty>Very Likely</certainty>
            <senderName>US-CERT United States Computer Emergency Readiness
            Team</senderName>
            <headline>Sun Solaris dtmail contains a format string
            vulnerability</headline>
            <description>The dtmail program is a mail user agent (MUA) for
            the Common Desktop Environment (CDE). It provides a graphical
            user interface for reading, sending, and managing email. There
            is a vulnerability in the way Sun Solaris dtmail handles
            command-line arguments. By supplying a specially crafted
            argv[0] value containing a format string specifier, a local
            user could execute arbitrary code with privileges of the
            vulnerable process.</description>
            <instruction>
                  <note>Sun has issued an advisory which addresses this
                        issue. For more information on patches available
                        for your system, please refer to Sun Security Alert
                        57627(http://www.sunsolve.sun.com/pub-
                        cgi/retrieve.pl?doc=fsalert/57627). Remove the
                        "set-group-ID" bit from dtmail by doing the
                        following: chmod 0555 /usr/dt/bin/dtmail. Note:
                        This may cause users to be unable to read NFS
                        mounted mailboxes.
                  </note>
                  <applyPatch platform="SPARC" version ="Solaris 8"
                        name="CDE 1.4"
      patchUri="http://au.sunsolve.sun.com/search/pdownload.pl?target=10961
      3-07method=h"/>
                  <applyPatch platform="SPARC" version ="Solaris 9"
      name="CDE 1.5"

      patchUri="http://au.sunsolve.sun.com/search/pdownload.pl?target=11281
0-06method=h" />
```

```
                  <applyPatch platform="x86" version ="Solaris 8" name="CDE
                  1.4"
      patchUri="http://au.sunsolve.sun.com/search/pdownload.pl?target=10961
      4-07method=h" />
                  <applyPatch platform="x86" version ="Solaris 9" name="CDE
                  1.5"
      patchUri="http://au.sunsolve.sun.com/search/pdownload.pl?target=11387
      0-05method=h" />
                  <permissionChange serverName="dtmail" note="This may
                        cause users to be unable to read NFS mounted
                        mailboxes.">
                        <command>"chmod 0555 /usr/dt/bin/dtmail"</command>
                  </permissionChange>
            </instruction>
            <web>http://www.kb.cert.org/vuls/id/481998</web>
            <contact>mailto:cert@cert.org?Subject=VU%23481998
            Feedback</contact>
      </info>
</alert>
```

10. Nota de Vulnerabilidade CAN-2004-0203

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!-- UNIVERSIDADE FEDERAL DO MARANHÃO -->
<!-- Projeto NIDIA  -->
<!-- A XML alert file -->
<!-- AUTOR: Fernando Augusto Pestana Junior-->
<alert xmlns:cap="http://www.incident.com/cap/1.0">
      <identifier>CAN-2004-0203</identifier>
      <sender>cert@cert.org</sender>
      <sent>2004-08-11T08:52:30</sent>
      <status>Actual</status>
      <scope>Restricted</scope>
      <restriction version="5.5" name="Microsoft Exchange"/>
      <msgtype>Alert</msgtype>
      <info>
            <category>Internet</category>
            <event>US-CERT Vulnerability Note</event>
            <urgency>Immediate</urgency>
            <severity>Moderate</severity>
            <certainty>Very Likely</certainty>
            <senderName>US-CERT United States Computer Emergency Readiness
            Team</senderName>
            <headline>Microsoft Outlook Web Access contains vulnerability
            in HTML redirection query</headline>
            <description>Outlook Web Access (OWA) is a component of
            Microsoft Exchange. By using OWA, a server that is running
            Exchange Server can also function as a website that lets
            authorized users read or send email messages, manage their
            calendar, or perform other mail functions over the Internet by
            using a web browser. A cross-site scripting vulnerability
            exists in the way OWA validates user input provided to an HTML
            redirection query. By convincing a user to click on a specially
            crafted URL within an email message,an attacker could cause
            arbitrary scripting code to be executed in the victim's
            browser. If executed, the script would have all privileges of
            the OWA user, including access to and manipulation of messages
            and folders on the server.
            </description>
            <instruction>
                  <note>Apply a patch as described in Microsoft
                        Security Bulletin MS04-026.
                  </note>
                  <applyPatch
patchUri="http://download.microsoft.com/download/d/f/6/df625ed0-5475-4df0-
9364-4dfb2a10ab69/Exchange5.5-KB842436-x86-enu.EXE"/>
                  <disableFeature serverName="Microsoft Exchange"
                        Feature="Outlook Web Access"  note="1. Start
                        Exchange Administrator. 2. Expand the Configuration
                        container for the site. 3. Click the Protocols
                        container for the site. 4. Open the properties of
                        the HTTP (Web) Site Settings object. 5. Click to
                        clear the Enable Protocol check box.6. Wait for the
                        change to replicate, and then verify that this
                        change has replicated to each server in the site.
                        To do this, bind to each server in the site with
                        Exchange Administrator, and then view the
                        setting."/>
            </instruction>
```

```
            <web>http://www.kb.cert.org/vuls/id/948750</web>
            <contact>mailto:cert@cert.org?Subject=VU%23948750
            Feedback</contact>
      </info>
</alert>
```

APÊNDICE C - XML *SCHEMA* PARA ENVIO DE MENSAGEM DO SDI PARA O CSIRT.

Arquivo requisicao.schema.

```
<?xml version="1.0" encoding="UTF-8"?>

<element name="request-alerts">

        <complexType>

                <sequence>

                        <element name="request" type="string">

                </sequence>

        </complexType>

</element>
```