LEONARDO CAVALLARI MILITELLI

Proposta de um agente de aplicação para detecção, prevenção e contenção de ataques em ambientes computacionais.

Dissertação de Mestrado apresentada à Escola Politécnica da Universidade de São Paulo como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.

São Paulo
2006

LEONARDO CAVALLARI MILITELLI

Proposta de um agente de aplicação para detecção, prevenção e contenção de ataques em ambientes computacionais.

Dissertação de Mestrado apresentada à Escola Politécnica da Universidade de São Paulo como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.

Área de Concentração:
Sistemas Eletrônicos

Orientador: Prof. Dr. João Antônio Zuffo

São Paulo
2006

FICHA CATALOGRÁFICA

Militelli, Leonardo Cavallari
Proposta de um agente de aplicação para detecção, prevenção e contenção de ataques em ambientes computacionais / L. C. Militelli. --São Paulo, 2006.
p. 73

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1.Segurança de redes 2.Segurança de computadores 3.Vírus e antivírus de computador I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II.t.

Agradecimentos

Em primeiro lugar, agradeço minha família, em especial minha mãe Luciana, por todo carinho, apoio e incentivo ao longo dos anos.

Ao meu orientador Prof. Dr. João Antônio Zuffo e co-orientador Prof. Dr. Volnys Borges Bernal pela oportunidade, orientação direcionada e por toda ajuda oferecida.

Aos meus amigos Artur, Gabrielle, Leandro, Milciades, Tamaris e Veronica que estiveram junto ao longo do desenvolvimento do trabalho e contribuiram com uma parcela imensurável para a concretização deste, mesmo sem terem tanta noção disto.

Aos grandes amigos Fábio, Murilo e Renato que acompanham e participam da mesma trajetória desde a graduação, com os quais pretendo enfrentar novos desafios.

Aos integrantes e amigos do grupo NSRAV, em particular, Adílson, Fernando e Matteo, pelas discussões técnicas, cooperação e colaboração no trabalho.

Resumo

Canais seguros, como os gerados pelos protocolos SSL e TLS, são cada vez mais utilizados nos serviços de rede para propiciar autenticação de parceiro, integridade e sigilo dos dados. Porém, sua utilização impede que um sistema de detecção de intrusão de rede possa observar o conteúdo dos pacotes, impossibilitando a análise das mensagens.

Como alternativa de contorno deste problema é proposta a arquitetura de um agente de detecção, prevenção e contenção de ataques baseado em aplicação, que possibilite interceptar fluxos de mensagens diretamente na aplicação, inserido no contexto de uma arquitetura de detecção distribuída e padronizada. O ADACA (Agente de Detecção, Análise e Contenção de Ataques) é um agente IDS (*Intrusion Detection System)* híbrido capaz de operar tanto no modo ativo quanto passivo. Dessa forma, permite realizar a análise do conteúdo de mensagens que estejam protegidos por protocolos seguros, como o SSL e TLS, e adotar uma medida pré-definida antes que a aplicação alvo processe um conteúdo malicioso.

Além disso, o padrão de formato de mensagens de alertas IDMEF (*Intrusion Detection Message Exchange Format*), proposto pelo IDWG, é adotado para notificação de eventos do agente ADACA a um IDS central.

Os resultados obtidos mostraram a viabilidade da utilização de agentes de aplicação, acoplados diretamente à aplicação, como complemento aos sistemas IDS de rede.

Abstract

Secure channel, as the one generated by protocols like SSL and TLS, has been used on network services to provide partner authentication, integrity and confidentiality. However, its utilization prevents a network intrusion detection system to observe and analyze packets content.

As an alternative to circumvent this problem, the present work proposes an agent-based intrusion detection, prevention and containment architecture capable to capture messages flows directly at the host application and introduce it on a distributed intrusion detection framework. The ADACA (Attack Detection, Analysis and Containment Agent) is a hybrid agent that can operate on active and passive mode. In this context, it is able to detect attacks where the application payload is encrypted by secure protocols, like SSL and TLS, and take some predefined measure before the host application process a malicious content.

Further that, Intrusion Detection Message Exchange Format (IDMEF) standard proposed by IDWG is considered to send alerts between agent ADACA and an IDS central.

The results shown that is practicable to use an application agent attached to an application as a complement of network intrusion detection systems.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

API	*Application Program Interface*
BEEP	*Block Extensible Exchange Protocol*
CAM	Controle de Acesso Mandatário
CERT.br	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
CIDF	*Commom Intrusion Detection Framework*
CISL	*Common Intrusion Specification Language*
DoS	Negação de Serviço em inglês *Denial of Service*
DTD	*Document Type Definition*
HIDS	Sistema de Detecção de Intruso baseado em Máquina em inglês *Host-based Intrusion Detection System*
HTTP	Protocolo de Transferência de HiperTexto em inglês *HyperText Transfer Protocol*
HTTPS	Protocolo de Transferência de HiperTexto Seguro em inglês *HyperText Transfer Protocol Secure*
IAP	*Intrusion Alert Protocol*
IDMEF	*Intrusion Detection Message Exchange Format*
IDREF	*Intrusion Detection Response Exchange Format*
IDP	*IDMEF Communication Protocol*
IDS	Sistema de Detecção de Intruso em inglês *Intrusion Detection System*
IDWG	*Intrusion Detection Exchange Format Working Group*
IDXP	*Intrusion Detection Exchange Protocol*
IESG	*Internet Engineering Steering Group*
IETF	*Internet Engineering Task Force*
IPS	Sistema de Prevenção de Intrusos em inglês *Intrusion Prevention System*
NIDS	Sistema de Detecção de Intruso baseado em Rede em inglês *Network-based Intrusion Detection System*
RFC	*Request for Comments*
SCXP	*Secure Components Exchange Protocol*
SQL	Linguagem de consulta estruturada, em inglês *Structured Query Language*
SSL	Camada de Soquetes Seguro em inglês *Secure Socket Layer*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
URL	Localizador Uniforme de Recursos, em inglês *Uniform Resource Locator*
VPN	Rede Privada Virtual em inglês *Virtual Private Network*

XML *Extensible Markup Language*

XSS *Cross-Site Scripting*

Capítulo 1. Introdução

1.1 A problemática da detecção de intrusos

Com base no estudo realizado nos trabalhos (ABBES; BOUHOULA; RUSINOWITCH, 2004) e (DASGUPTA et al., 2005), percebe-se que grandes esforços estão sendo feitos na área de detecção de intrusão de forma a tornar viável a identificação de atividades maliciosas com eficácia, procurando reduzir a quantidade de falso-positivos e falso-negativos. Porém, neste contexto, ainda é possível observar diversas vertentes com grande carência de pesquisas.

Uma dessas vertentes trata da identificação de ataques quando o alvo destes é um serviço ou aplicação que utiliza mecanismos de criptografia na comunicação. Tais mecanismos são utilizados com o intuito de garantir a confidencialidade dos dados e autenticação de parceiros. Porém, ao realizar a criptografia dos dados, estes se tornam legíveis apenas para a entidade a que se destinam, tornando inviável a análise do conteúdo por parte de dispositivos de segurança existentes no ambiente, como sistema de detecção de intrusos e *firewall* com funcionalidade de inspeção de conteúdo.

Desta forma, um atacante poderia fazer uso de um mecanismo adotado para aumentar a segurança da aplicação e camuflar suas atividades ilícitas, sem que seja facilmente detectado. Como exemplo, pode-se citar a utilização do protocolo HTTPS como um canal seguro para varrer sistemas WEB a procura por falhas ou explorar determinada vulnerabilidade de um serviço ou aplicação WEB.

Exemplo desta situação pode ser ilustrada pela vulnerabilidade UNICODE do servidor IIS da Microsoft descoberta em outubro de 2000. Por meio de uma falha no processo de decodificação de requisições, era possível executar comandos do sistema operacional e acessar o conteúdo das unidades lógicas, conforme reportado em (MICROSOFT, 2000). Enquanto as correções não eram divulgadas pelo fabricante, a detecção desta falha somente era possível por meio de sistemas IDS. No entanto, as conexões realizadas sobre um canal seguro de comunicação impossibilitavam a análise dos pacotes.

Para contornar esta situação, a empresa McAfee lançou em meados de 2005 a solução *Intrushield SSL Traffic Inspection and Prevention* (MCAFEE, 2005), um sistema de detecção de intrusos no qual a chave privada do serviço seguro a ser protegido é conhecida e utilizada para decifrar toda a comunicação realizada entre o cliente e servidor. Posteriormente, a solução *BreachView SSL* (BREACH, 2005), voltada para sistemas Linux, foi apresentada pela empresa Breach, partindo do mesmo princípio.

Estas duas soluções, apesar de viabilizar a detecção de intrusos para situações de tráfego criptografado, compartilham a chave privada dos serviços seguros. Esta prática pode não ser aceita em muitas organizações, visto que o compartilhamento da chave privada pode infringir a política de segurança adotada.

De acordo com o recente estudo de PALLER (2005), o crescente aumento dos ataques focados na camada de aplicação, como SQL *Injection* (SPI, 2002) e *Cross-Site Scripting* (XSS) (ZUCHLINSKI, 2003), somado aos inúmeros sistemas de comércio eletrônico, permite concluir que, apesar de todos os produtos de segurança disponíveis, ainda existe carência de soluções específicas.

1.2 Motivação

Na tentativa de sanar a deficiência existente nos sistemas de detecção atuais algumas soluções foram propostas visando detectar ataques em canais cifrados na camada de transporte e na camada de aplicação da pilha TCP/IP, sem haver necessidade do conhecimento da chave privada. Os trabalhos nos quais foram reportadas alternativas para identificação de ataques em protocolos de segurança, baseiam-se na detecção de anomalias por perfis de tráfego. Com base em um perfil criado a partir de uma fase de treinamento utilizando requisições lícitas, todos os pacotes que ultrapassarem o limite imposto pelo perfil pré-definido são classificados, conforme (YASINSAC, 2002) e (LECKIE; YASINSAC, 2004).

Uma alternativa para os ataques direcionados à camada de aplicação é apresentada por Joglekar e Tate (2004), na qual os autores propõem uma solução baseada no acoplamento de um monitor junto a uma aplicação específica, como um servidor SMTP, ou até mesmo ao módulo de criptografia de um servidor WEB. Esse monitor observa as mensagens de controle do protocolo que são transmitidas entre cliente e servidor, o que torna possível detectar ataques baseado no comportamento anômalo do protocolo que é monitorado, possibilitando, ainda, conter certos tipos de ataques específicos. Neste trabalho, Joglekar e Tate convergem os estudos propostos primeiramente em (SEKAR et al., 2002) e (YASINSAC, 2002) e acrescem a funcionalidade de redução da efetividade dos ataques em nível de aplicação por meio da inserção de intervalos entre o processamento das requisições entrantes. A contenção de ataques por meio do gerenciamento dos recursos do servidor é uma prática plausível de ser utilizada, conforme apresentado em (MEYLAN, 2004), onde técnicas de qualidade de serviço (QoS – *Quality of Service*) são empregadas para diminuir a quantidade de banda disponível ao endereço IP ofensivo, principalmente para ataques de DoS (*Denial of Service*).

PLAGGEMEIER e TÖLLE (2002) apresentaram o sistema de detecção de intrusos por meio do agente procurador (*proxy*) criptográfico EPIDS (*Encrypted Proxy Intrusion Detection System*), no qual todas as conexões provenientes de um cliente SSH são intermediadas por um *proxy* transparente capaz de extrair a informação do pacote cifrado, analisar e validar a existência de conteúdo malicioso e, na seqüência, repassar as requisições ao serviço. A técnica adotada parte do princípio do ataque de *Man-in-the-middle* (BURKHOLDER, 2002), o qual possibilita a interceptação e manipulação de pacotes criptografados por meio da interceptação e troca da chave pública original do servidor por uma chave pública falsa.

1.3 Objetivo

Este trabalho apresenta a proposta de uma arquitetura de detecção, prevenção e contenção de ataques baseada em agente de aplicação. O agente de aplicação é acoplado a uma aplicação por meio de uma API genérica. Este agente, denominado **ADACA** (Agente de Detecção, Análise e Contenção de Ataques) é capaz de realizar análise das mensagens recebidas e transmitidas pelo servidor e de identificar dados considerados anômalos ou maliciosos. Esta proposta inclui, não somente a definição da arquitetura do agente de aplicação, mas também sua inserção em uma arquitetura de detecção distribuída e padronizada. Como estudo de caso, será apresentado um protótipo de agente para a situação de um servidor WEB seguro (HTTPS) e a troca de mensagens entre o agente e um IDS central.

Diferente da abordagem inserida em (SEKAR et al., 2002) e utilizada em (JOGLEKAR; TATE, 2004), que se baseiam na comparação das mensagens do protocolo com os perfis pré-definidos (por exemplo, mensagens *ClientHello* e *ClientKeyExchange* provindas

de uma sessão SSL ou troca de mensagens de um servidor SMTP), a presente proposta pretende realizar a detecção de ataques por meio da análise das mensagens do protocolo de aplicação imediatamente antes de seu processamento pela aplicação, no caso de uma requisição, ou antes do envio de sua resposta ao cliente, permitindo inserir medidas de prevenção e contenção aos ataques.

1.4 Organização do Trabalho

Os capítulos deste trabalho estão organizados da seguinte forma.

O capítulo 2 apresenta os conceitos atualmente empregados a respeito de sistemas IDS/IPS e os padrões e protocolos existentes para a intercomunicação entre esses sistemas, como o IDMEF e IDXP. O capítulo 3 é reservado para descrever sobre trabalhos relacionados, apresentando uma breve introdução ao projeto IDREF (SILVA, 2004), Protomon (JOGLEKAR; TATE, 2004) e AppArmor (NOVELL, 2005).

A proposta da arquitetura é contemplada no capítulo 4, no qual é conceituado e introduzido o Agente de Detecção, Análise e Contenção de Ataques (ADACA), bem como a arquitetura de detecção e prevenção de ataques, a arquitetura e interfaces do agente com os outros elementos participantes, como o IDS Central e IDS Local. Os detalhes da implementação da solução, as peculiaridades das interfaces com os outros dispositivos e os testes realizados são mostrados no capítulo 5. O capítulo 6 apresenta os resultados obtidos para os diferentes modos de operação, um estudo de viabilidade e as limitações da implementação. Por fim, o capítulo 7 conclui o trabalho e apresenta propostas que deverão ser aprofundadas futuramente.

Capítulo 2. Conceitos e padrões da área de detecção de intrusos

Segundo (BACE; MELL, 2001), IDS é um sistema baseado em *software* ou *hardware*, normalmente interconectado com outros elementos, capaz de monitorar um canal de transmissão de dados e arquivos de registros de servidores em busca de tráfego anômalo e ações consideradas maliciosas, por meio de diferentes técnicas de detecção. O agente IDS, também conhecido como sensor, é o elemento responsável por capturar as informações que serão remetidas para análise.

O IDS pode ser um sistema centralizado, contendo apenas um agente, ou distribuído, de forma que todos os agentes reportem seus registros e alertas à base central de eventos. Neste contexto, sistemas de filtragem (*firewalls)* também podem ser considerados agentes IDS quando enviarem notificações à base central.

A utilização de tal sistema tem se mostrado bastante útil frente ao aumento incessante de ataques partindo da Internet e, até mesmo, internamente aos ambientes corporativos. De acordo com os dados apresentados pelo Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br), é notável o crescimento exponencial do número de incidentes reportados. Uma breve análise baseada na quantidade de incidentes reportados até o ano de 2005, permite concluir que houve um aumento de aproximadamente 51% com relação ao ano de 2004 (CERT.br, 2005). A Figura 2.1, apresenta a estatística dos incidentes reportados nos últimos anos, exceto pelas ameaças do tipo *worm.*

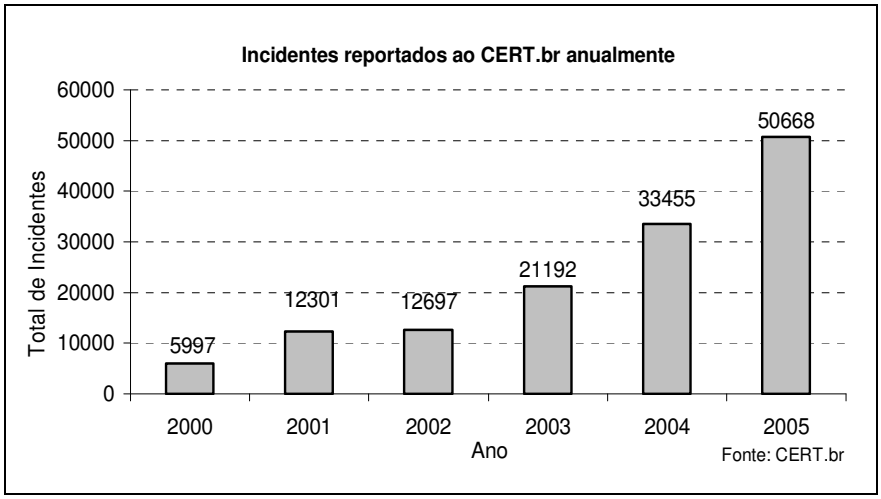

Figura 1. Estatística sobre a quantidade total de incidentes reportados ao CERT.br

2.1 Agentes

Para um melhor entendimento sobre as funcionalidades da arquitetura proposta, é necessário apresentar as diferenças entre as classes de agentes existentes. Os agentes são classificados em relação a sua localização e atividade.

2.1.1 Classificação de agentes em relação à localização

De acordo com o NIST (1999), existem três principais tipos de agentes de detecção de intrusão, sendo:

- Agente baseado em rede;
- Agente baseado em máquina;
- Agente baseado em aplicação.

2.1.1.1 Agente baseado em rede (NIDS)

Agente baseado em rede (*Network-based Intrusion Detection System* - NIDS), comumente conhecido como sensor de rede, é o tipo de IDS mais adotado pelas equipes de segurança devido ao baixo custo do investimento e mínimo impacto sobre a topologia de rede. Sua principal funcionalidade é detectar a presença de tráfego anômalo ou pacotes com dados maliciosos por meio da escuta digital de pacotes e classificá-los de acordo com o conteúdo, gerando os alertas. Em seguida, os alertas podem ser reportados ao analisador central para serem correlacionados e, opcionalmente, alguma contramedida pode ser definida para conter futuras conexões maliciosas provindas do endereço ofensivo ou pacotes que contenham a mesma informação.

Apesar das vantagens existentes na utilização deste tipo de agente, como a monitoração de enlaces de alta velocidade, este tipo de agente é ineficaz na análise do conteúdo de fluxos de dados cifrados, visto que os pacotes somente poderão ser abertos por meio da chave privada armazenada na outra extremidade da comunicação.

2.1.1.2 Agente baseado em máquina (HIDS)

O agente baseado em máquina (*Host-based Intrusion Detection System* - HIDS) tem como objetivo monitorar as atividades executadas no sistema operacional local. Entre elas, pode-se citar a monitoração de processos ativos, trilhas de auditoria, usuários, entre outros. Esta forma de detecção consome diversos recursos do servidor, como memória, processamento e banda,

devido ao constante monitoramento de arquivos e troca de informações com o servidor central do IDS, visto que parte das soluções existentes analisam os dados em uma máquina remota.

2.1.1.3 Agente baseado em aplicação

O agente baseado em aplicação é um subconjunto do HIDS, isso porque sua atuação tem grande semelhança, sendo distinguido apenas pela monitoração direcionada apenas a uma aplicação, como um antivírus em servidor de correio eletrônico. Neste contexto, o IDS de aplicação, como é chamado, é capaz de validar toda a interação entre o usuário, os dados e a aplicação (BACE; MELL, 2001).

Os agentes baseados em aplicação são diferenciados dos agentes baseados em máquina por um limiar bastante estreito. POSEY (2005) ressalta que apesar do escopo de atuação ser diferente, ambos consomem uma série de recursos locais, como processamento e memória, que podem causar atraso no funcionamento da aplicação.

Mesmo tendo grande eficácia no monitoramento da aplicação a ele designada, não provê qualquer outra funcionalidade ao servidor, como intervenção de atividades ilícitas, podendo ser classificado como um agente passivo.

2.1.2 Classificação dos agentes em relação à atividade

Atualmente, existem dois principais modos de operação para os agentes acima citados, sendo classificados em passivo e ativo. Além disso, existe um outro tipo de sistema denominado IPS (Sistema de Prevenção de Intrusos) que tem capacidade de bloquear conteúdo considerado

malicioso. Assim sendo, para um melhor entendimento das diferenças entre as formas operacionais e características específicas, será apresentado um resumo de cada um desses modos.

2.1.2.1 Agente Passivo

O agente passivo é um componente de ação limitada, responsável por detectar ataques e reportar possíveis alertas ao IDS Central. Este tipo de agente não possui qualquer funcionalidade de contramedida aos alertas gerados, cabendo ao IDS Central interagir com os dispositivos do ambiente, *firewalls* e servidores, para contenção de ataques. Atualmente, essa interação é encontrada em soluções proprietárias, as quais possibilitam interações com diversas marcas e tipos de equipamentos.

Já para soluções abertas como o Snort (ROESCH, 2005), existem componentes adicionais, como o SnortSam (KNOBB, 2006) e o Guardian (STEVENS, 2004), que permitem acionar uma regra de filtragem junto a alguns modelos de *firewalls*. Porém, a compatibilidade é bastante reduzida e, portanto, nem sempre viável.

2.1.2.2 Agente Ativo

O agente ativo introduz o conceito mais aproximado ao *Intrusion Prevention System* (IPS) (ZHANG; LI; ZHENG, 2004) no qual o agente tem as funcionalidades de prevenir o sucesso da exploração de ataques ou minimizar o impacto de um ataque bem sucedido, por meio da intervenção direta na receptação dos pacotes considerados maliciosos. A tabela a seguir resume as principais diferenças entre os dois agentes.

Tabela 1. Diferença entre os modos de operação de agentes

Modo de Operação	Funcionalidade	Conceito-Base
Passivo	Detecção	IDS
Ativo	Detecção e Prevenção	IPS

2.1.2.3 IPS – Sistema de Prevenção de Intrusos

De acordo com ENDORF, SCHULTZ e MELLANDER (2004), o sistema de prevenção de intrusos (IPS) tem funcionamento análogo ao IDS com relação à captura e análise de pacotes, porém diferenciam-se pelo fato do IPS ser capaz de interceptar e bloquear requisições que não se adequarem dentre as regras de tráfego legítimo pré-estabelecidas.

A maior discussão a respeito do IPS é sobre a intervenção de conexões nas quais as requisições são classificadas erroneamente, determinando um falso-positivo. Neste caso, o sistema iria bloquear requisições provenientes de clientes legítimos.

Uma breve comparação conceitual entre as características do IDS e IPS pode ser observada na tabela a seguir.

Tabela 2. Características básicas de IDS e IPS

IDS	IPS
Instalado em segmentos de rede e máquina	Instalado em segmentos de rede e máquina
Observa a rede passivamente	Observa e intercepta o tráfego
Ineficaz para fluxos criptografados	Melhor atuação sobre aplicações
Controle de gerenciamento centralizado	Controle de gerenciamento centralizado
Melhor para detecção de ataques em geral	Ideal para bloquear pichações de página
Ação reativa	Ação preventiva

2.2 Padronização de sistema de detecção de intrusos

Na grande maioria das vezes, o sistema IDS é implementado de maneira distribuída, sendo composto por diversos agentes além da base central de alertas e gerenciamento. Para que este sistema não sofra qualquer problema de interoperabilidade durante a comunicação de alertas e mensagens de controle entre os componentes existentes, ou ainda, para tornar possível a utilização dos alertas gerados pelos agentes de rede, máquina e aplicação na correlação de eventos para o rastreamento, intervenção e/ou prevenção de certa atividade maliciosa, se faz necessário adotar um padrão que viabilize a transmissão de mensagens uniformemente (KAHN et al, 1998). Assim, algumas propostas que visam padronizar o formato e conteúdo das mensagens e do protocolo de comunicação serão apresentadas a seguir.

2.2.1 CIDF

A arquitetura comum de detecção de intrusos CIDF (*Commom Intrusion Detection Framework*) idealizada em (KAHN et al, 1998) e (STANIFORD-CHEN; TUNG; SCHNACKENBERG, 1998) foi a tentativa inicial de padronização de sistemas IDS, cujo objetivo era prover a intercomunicação entre dispositivos de identificação de intrusos e os sistemas de resposta, como os *firewalls*. Foi especificada uma linguagem chamada de CISL (*Common Intrusion Specification Language)* que formaliza um protocolo para troca de informações de alertas, além da elaboração de um modelo simplificado dos requisitos, componentes e da arquitetura que compõe o sistema. O CIDF prevê em sua arquitetura funcional, os seguintes elementos:

- E-Box (Unidade de Eventos): responsável por gerar eventos de segurança que poderão se tornar alertas, a partir da informação proveniente de uma fonte de dados. No caso de um

meio físico, este módulo é encarregado por reconstruir o pacote de dados e repassar para análise;

- A-Box (Analisador de eventos): neste é onde ocorre toda a análise e correlacionamento dos eventos, além da interação direta com o módulo de resposta;
- R-Box (Unidade de resposta): componente designado para realizar as atividades de respostas no sistema IDS, sejam elas ativas (contramedida) ou passivas (informativa);
- D-Box (Base de eventos): armazena o histórico dos eventos conforme a ocorrência.

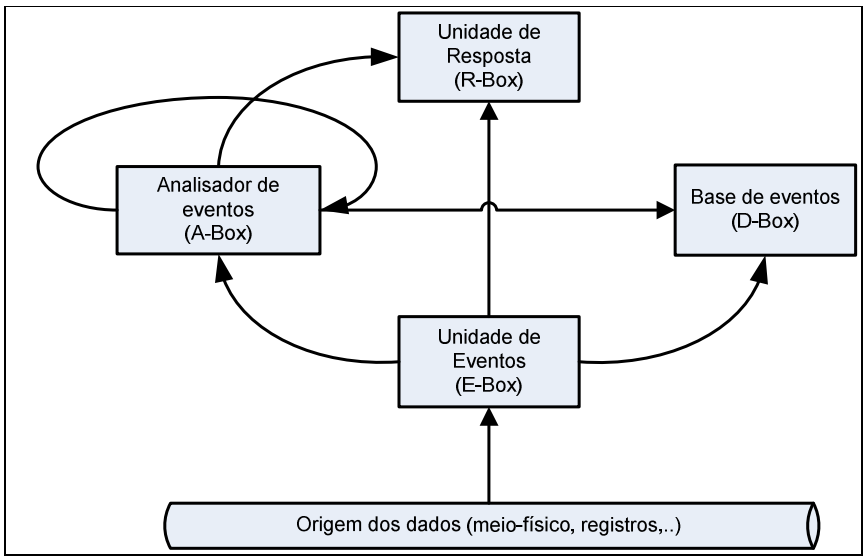

Figura 2. Arquitetura funcional CIDF

Apesar de o esforço inicial ter apresentado pontos positivos com relação principalmente à arquitetura, no mesmo ano de 1998 deu-se início ao grupo IDWG, o qual aplicou algumas das idéias e conceitos introduzidos na especificação CIDF.

2.2.2 Intrusion Detection Exchange Format Working Group (IDWG)

No encontro do IETF de agosto de 1998, foi criado o *Intrusion Detection Exchange Format Working Group*, o qual desde então objetiva a padronização do formato de dados e procedimentos para viabilizar o intercâmbio das informações dos alertas.

Dentre os documentos elaborados pelo grupo destaca-se o documento dos requisitos para a troca de mensagens de detecção de intrusos (WOOD; ERLINGER, 2002). Neste documento é especificado o padrão do formato das mensagens (IDMEF – *Intrusion Detection Message Exchange Format)* e os requisitos necessários para a implementação deste, além da definição do protocolo de comunicação IDP (*IDMEF Communication Protocol*). Para este último, foram apresentados diversos requisitos fundamentais para permitir a transmissão de mensagens entre as entidades participantes. Alguns desses requisitos são:

- Autenticação mútua entre sensores e console de gerenciamento;
- Transmissão segura de mensagens por meio de mecanismos que garantam a confidencialidade e integridade do conteúdo das mensagens durante o processo de comunicação, devendo ser utilizado algoritmos de criptografia já existentes;
- Resistência a ataques de negação de serviço (*Denial of Service* - DoS);
- Proteção contra ataques do tipo "*replay*", baseado no envio malicioso de mensagens duplicadas.

Em seguida, foi proposto o modelo de dados para representação da informação dos alertas a serem transmitidos a um IDS Central (DEBAR; CURRY; FEINSTEIN, 2005), utilizando para tal um DTD (*Document Type Definition)* que descreve o formato dos dados IDMEF por meio de documentos XML.

Posteriormente a estas publicações, o grupo desenvolveu a especificação de dois protocolos interoperáveis para comunicação das mensagens IDMEF entre os dispositivos envolvidos no sistema de detecção de intrusos: o *Intrusion Alert Protocol* (IAP) (GUPTA et al., 2001), e o *Intrusion Detection Exchange Protocol* (IDXP) (FEINSTEIN; MATTHEWS; WHITE, 2002).

Algumas destas especificações foram submetidas ao *Internet Engineering Steering Group* (IESG) e encontram-se em fase final de rascunho (*draft)* de padrão.

2.2.2.1 Protocolo de comunicação interoperável

Com base no documento de requisitos do IDP, duas especificações de protocolos foram desenvolvidas pelo IDWG: IAP e IDXP.

O protocolo IAP foi a primeira tentativa do grupo IDWG em conceituar uma solução capaz de cumprir todos os requisitos previamente definidos. Por meio do documento *"Intrusion Alert Protocol"* (GUPTA et al., 2001) foi proposto um protocolo em nível de aplicação para o intercâmbio de dados de alertas entre as entidades do sistema de detecção, semelhante ao protocolo HTTP com alguns conceitos de *proxy*, para autenticação e controle. Porém, no 50º encontro do IETF foi definido que o desenvolvimento sobre o IAP deveria ser desconsiderado e que os esforços fossem concentrados em um novo protocolo: o IDXP.

Herdando algumas características do IAP, o IDXP atende a todos os requisitos necessários para se adequar ao modelo proposto pelo IDWG como protocolo de transporte de mensagens IDMEF.

A implementação do IDXP é concretizada utilizando um arcabouço de desenvolvimento de protocolos para camada de aplicação conhecido por *Block Extensible Exchange Protocol* (BEEP) (ROSE, 2001) (BUCHHEIM et al., 2001), o qual contempla uma série de funcionalidades e mecanismos de controle que garantem integridade, confidencialidade, autenticação de parceiros, entre outros quesitos. Neste contexto, o protocolo BEEP funciona analogamente a uma rede privada virtual (VPN) em nível de aplicação, estabelecendo um túnel seguro e confiável entre dois pontos sob os quais são criados canais de comunicação baseados em perfis[1], que definem a sintaxe e semântica das informações trocadas. A Figura 3 representa um túnel BEEP e o estabelecimento de canais padronizados por meio do perfil do IDXP e Syslog (NEW; ROSE, 2001). Vale ressaltar que a forma de comunicação dentro do túnel é assíncrona, ou seja, não há limitações para a quantidade de canais e perfis utilizados simultaneamente entre duas entidades.

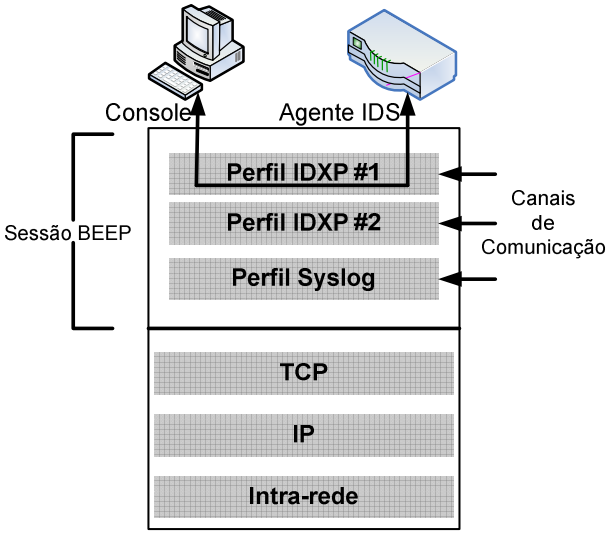

Figura 3. Canais de comunicação criados com diferentes perfis em uma única sessão BEEP

Além da fácil extensibilidade, uma das maiores vantagens da utilização do BEEP está relacionada ao baixo custo de segurança envolvido, pois o processo de autenticação e

[1] A documentação referente ao BEEP e os perfis existentes podem ser encontrados em: http://www.beepcore.org

negociação pode ser realizado uma única vez para o estabelecimento de um túnel de comunicação seguro e confiável. Dentro deste túnel são criados os canais de comunicação definidos pelos perfis, sem que haja necessidade de outros procedimentos de segurança, como a autenticação.

2.2.3 SCXP - Secure Components Exchange Protocol

No sentido de tornar o escopo do IDXP mais abrangente, a proposta do protocolo SCXP, inserida por (YANG; CHANG; CHU, 2003), tem como objetivo prover a integração entre todos os componentes de segurança presentes em uma rede por meio de um protocolo único de comunicação. Como exemplo pode-se citar o envio assíncrono de notificações (alertas) detectadas por um *firewall* ao analisador e, até mesmo, o envio de mensagens de controle (atuação) do IDS Central ao *firewall,* designando as ações a serem tomadas no caso de um ataque.

Esta proposta não pertence aos esforços realizados pelo grupo IDWG e, desde sua publicação, não foram encontrados outros trabalhos relacionados.

2.3 Situações de mensagens fracionadas

Dentre os protocolos de comunicação, existem alguns que realizam a comunicação em pequenos segmentos, como é o caso do protocolo TELNET (RFC 318) e SSH (RFC 4251). Estes protocolos enviam dados conforme a digitação do indivíduo que está fazendo uso de tal serviço. De acordo com a implementação, os dados são enviados *byte*-a-*byte* ou em pequenos

grupos de bytes ao sistema remoto, sendo que a instrução só será interpretada quando o servidor receber o sinal de execução. Se fizermos uma captura digital de pacotes no canal de comunicação durante uma sessão de telnet, observa-se a seguinte situação (Figura 4).

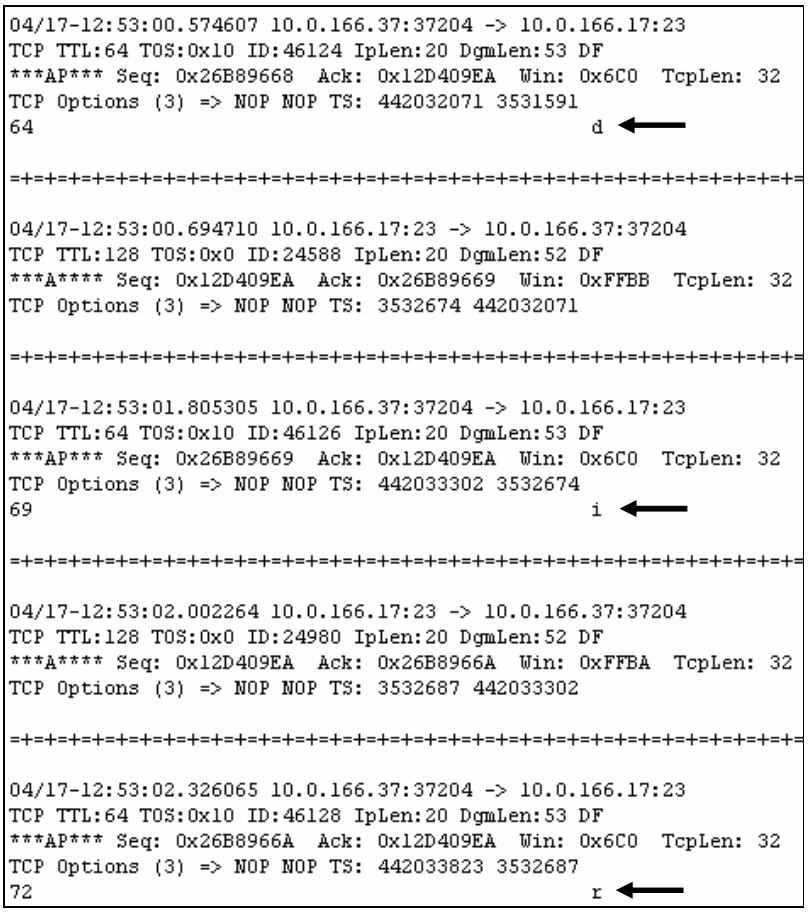

Figura 4. Captura de comando "dir" em conexão telnet

As situações de tráfego fracionado fazem com que os sistemas de detecção de intrusos necessitem armazenar o estado de cada um dos bytes enviados de forma a construir a requisição completa e, somente então, ser capaz de identificar uma possível ação maliciosa.

Porém, tais eventos de mensagens fracionadas podem ocorrer de outras formas, como é o caso do processo de evasão de IDS, no qual se tem por objetivo burlar sistemas de detecção.

2.3.1 Evasão de IDS

Evasão de IDS é um processo no qual um atacante manipula maliciosamente as requisições destinadas a um determinado serviço de forma que possam passar despercebidas pelo processo de detecção de intrusos ou subverter o sistema de análise, diminuindo sua eficácia. Para obter sucesso neste processo de camuflagem podem ser aplicadas diversas técnicas, como a fragmentação e segmentação.

2.3.1.1 Fragmentação

Fragmentação de pacotes é uma técnica adotada no padrão do protocolo TCP/IP a fim de viabilizar a transmissão de pacotes entre diferentes redes. O datagrama IP pode ter um tamanho máximo de 64Kbits, porém muitas redes não têm capacidade de aceitar pacotes desta dimensão e necessitam que estes sejam partidos em uma unidade menor para que todos os elementos de rede presente entre as duas extremidades da comunicação possam trafegá-los. Uma vez que todos os fragmentos tenham alcançado seu destino, sem necessidade de obedecer a ordem de chegada, o sistema operacional remoto é encarregado em remontar o pacote.

As técnicas de evasão por fragmentação apresentadas inicialmente por PTACEK e NEWSHAM (1998) aproveitam este recurso do protocolo IP para fragmentar as requisições maliciosas, na tentativa de dificultar o processo de detecção. Uma delas baseia na fragmentação dos pacotes em unidades mínimas de informação, na tentativa de impossibilitar o armazenamento e remontagem dos fragmentos por parte do sistema de detecção, visto o grande volume de informação trafegado no enlace. Porém, esta técnica já não é mais eficiente devido aos pré-processadores dos sistemas de detecção que tem capacidade de armazenar o estado desses pacotes e identificar requisições maliciosas.

2.3.1.2 Segmentação

A técnica de evasão por segmentação de pacotes é bastante similar à de fragmentação, diferenciada pela atuação na camada de aplicação do protocolo TCP/IP.

Dentre as diversas técnicas de segmentação de pacotes, pode-se destacar a sobreposição de fragmentos. Esta técnica visa transpassar o sistema de detecção devido à subversão na remontagem de pacotes. Seu funcionamento é baseado no envio de requisições segmentadas à aplicação, sendo que a quantidade de *bytes* a serem interpretados pelo sistema vítima é controlado pelo número de seqüência do próximo segmento a ser recebido. Ou seja, o último *byte* do segmento 1 é o primeiro *byte* do segmento 2. De acordo com a Figura 5, no caso de um ataque realizado pela requisição HTTP "**GET /teste.idq**" a troca de pacotes entre atacante e vítima se caracterizaria da seguinte forma.

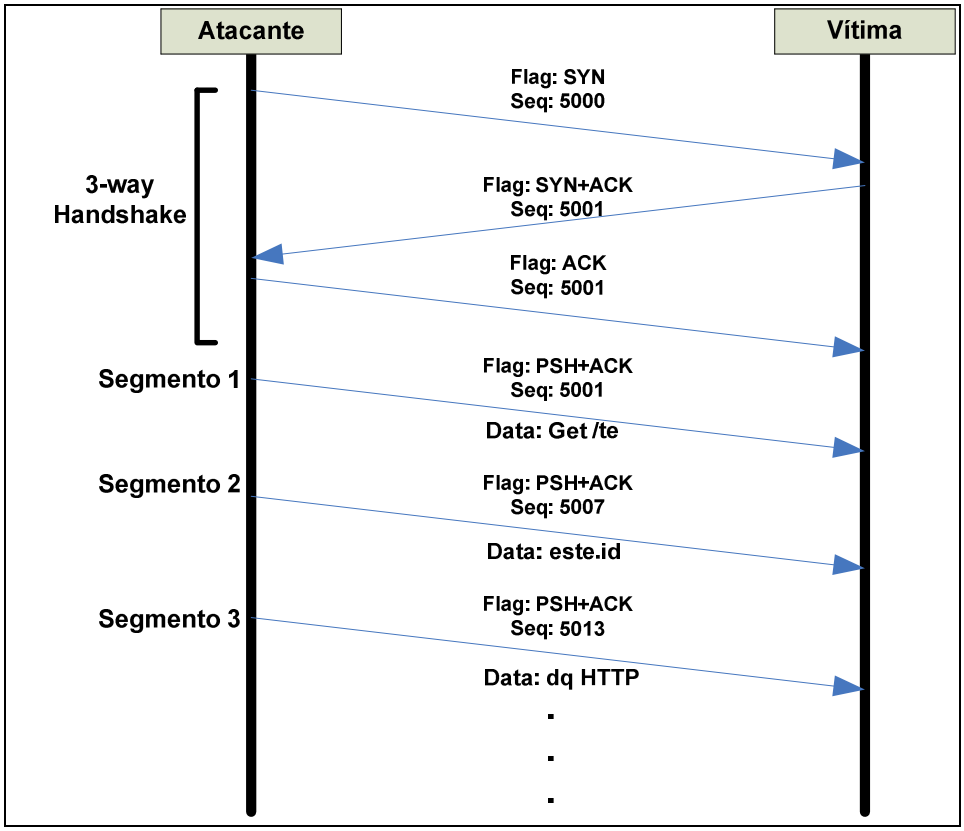

Figura 5. Evasão por sobreposição de fragmentos

Além da fragmentação por sobreposição, existem outras formas como fragmentação por sobreescrição de *bytes* e por expiração do tempo de armazenamento no *buffer* de memória do sistema de detecção.

Capítulo 3. Trabalhos relacionados

Durante a etapa de pesquisa, foram encontrados projetos que se relacionam com o presente trabalho. O projeto mais próximo desta proposta é o *Intrusion Detection Response Exchange Format* - IDREF (SILVA, 2004), que estende a utilização do padrão IDMEF com a funcionalidade de envio de contramedidas aos ataques detectados. Outro projeto é o ProtoMon (JOGLEKAR; TATE, 2004), abreviação de *Protocol Monitor,* no qual é apresentada uma arquitetura de detecção baseada em comportamento anômalo de protocolos. Mais recentemente foi encontrado o projeto AppArmor (NOVELL, 2005), suportado pela equipe de desenvolvimento do sistema operacional SUSE da empresa Novell.

3.1 IDREF

Com o intuito de estender a utilização e acrescer funcionalidades ao modelo de dados IDMEF, descrito anteriormente, o IDREF tem por objetivo adotar mecanismos que possibilitem a um operador enviar respostas aos alertas detectados pelo sistema IDS.

Com base no modelo IDMEF, foram realizadas modificações somente sobre o elemento resposta da arquitetura original (Figura 6), adicionando relacionamentos e funcionalidades mais específicas, conforme pode-se observar na Figura 7.

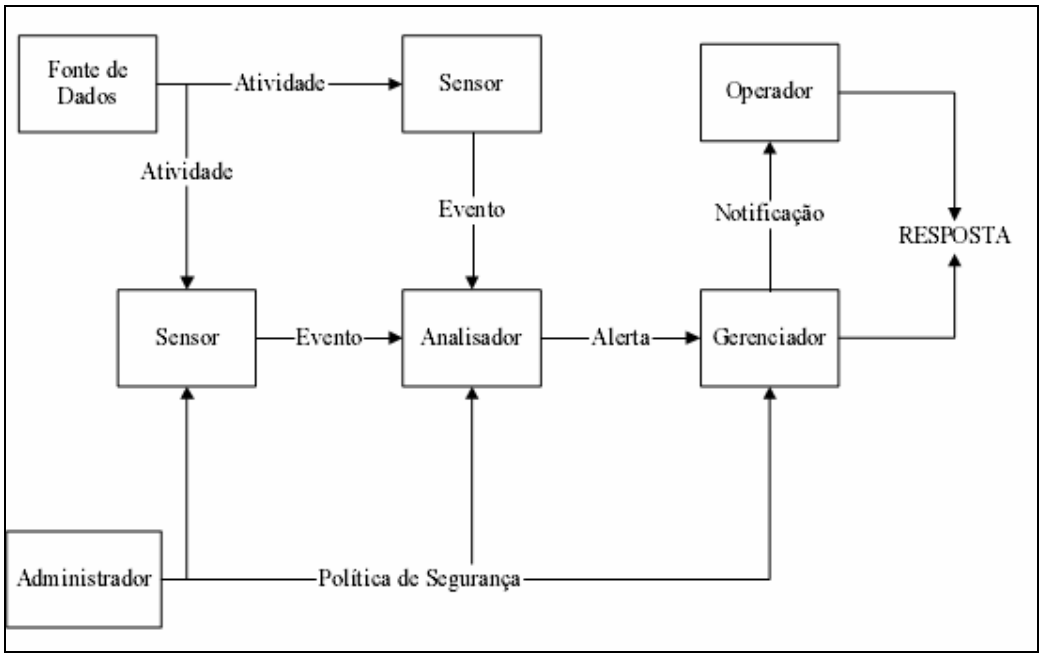

Figura 6. Arquitetura IDS conforme definição do IDWG (SILVA, 2004).

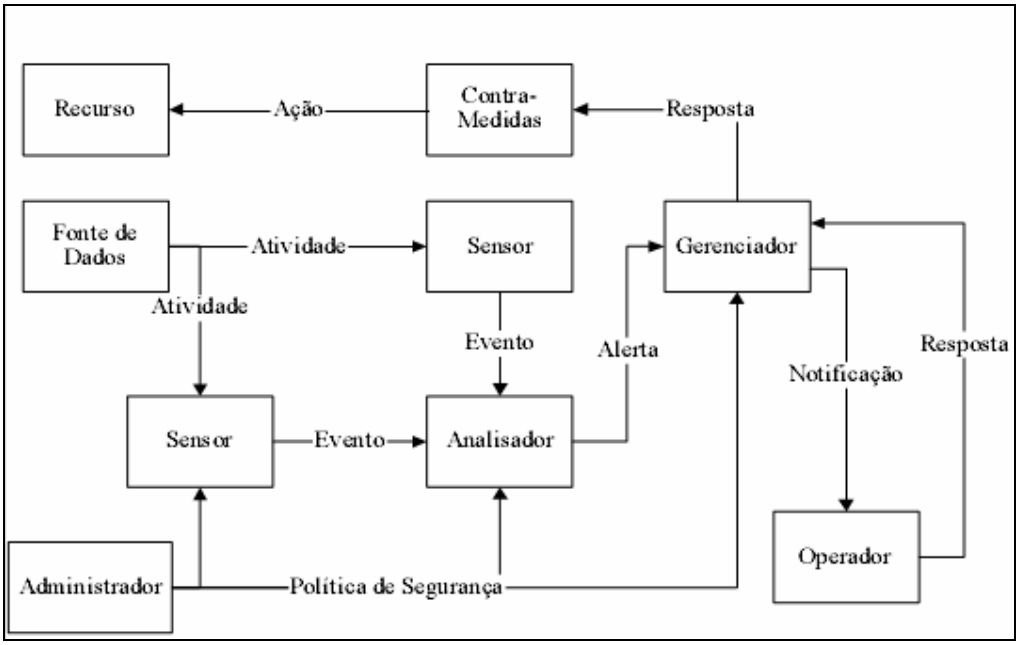

Figura 7. Arquitetura IDS adaptada ao IDREF (SILVA, 2004).

Portanto, nota-se que os componentes adicionados na arquitetura do IDREF refletem somente sobre as ações de resposta. Agora, o gerenciador, responsável por centralizar os alertas ocorridos no sistema de detecção e apresentar as informações por meio da console ao operador, é responsável também por controlar as ações de contenção pré-estabelecidas e selecionadas pelo operador quando em situação de ataque, atuando sobre o componente contramedida.

Também foram estipulados os possíveis tipos de resposta a serem selecionados pelo operador, sendo divididos em três categorias:

- *Response:* permite enviar ocorrências de ataques aos operadores por meio de telefone, *pager* e e-mail;
- *React:* define as ações de contenção que podem ser atribuídas a certos recursos como bloqueio de arquivos, fechamento de conexões e finalização de processos do sistema operacional;
- *Config:* permite alterar configurações dos recursos, como alteração de permissão de usuário ou arquivo, reconfiguração de regras de *firewall*, entre outros.

Toda esta arquitetura se baseia nos alertas gerados por um IDS de rede, ou seja, mesmo havendo as funcionalidades de resposta e comunicação, esta arquitetura não é capaz de identificar ataques quando o protocolo de comunicação está criptografado, sendo este um fator limitante.

A implementação do projeto limita-se na troca de mensagens entre os componentes de detecção, análise e gerência. Não foi abordada a forma como são realizadas as ações de resposta, reação e configuração junto aos recursos do ambiente.

3.2 ProtoMon

O ProtoMon, proposto por Joglekar e Tate, tem como principal objetivo monitorar protocolos de comunicação, a fim de identificar requisições maliciosamente manipuladas. Baseando-se em perfis pré-definidos, realiza-se a identificação de anomalias nas requisições transacionadas entre cliente e servidor,.

Os autores apresentam três modos de operação para os agentes: aprendizado, detecção e prevenção. O modo de aprendizado é a forma como se determinam os perfis de acesso pré-definidos. Por meio da análise do comportamento de conexões legítimas e da análise estatística dos estados de transição de um protocolo, o ProtoMon permite definir o tipo de requisição e resposta esperado. Ou seja, os agentes analisam todo o fluxo de mensagens trocadas entre cliente e servidor desde o estabelecimento da conexão até a finalização da mesma, observando a ordem e freqüência da aparição de mensagens do protocolo analisado. Dessa forma, o ProtoMon viabiliza a identificação de ataques que tem por objetivo explorar falhas em protocolos seguros, como o SSL. A Figura 8 apresenta as etapas utilizadas no estabelecimento de uma conexão segura, as quais estariam sendo analisadas constantemente pelo monitor do protocolo.

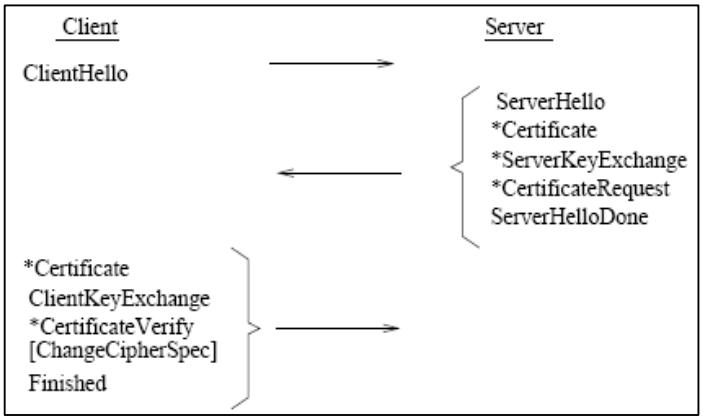

Figura 8. Troca de estados de uma conexão SSL

Já o modo de detecção faz uso do perfil criado pelo modo de aprendizado como padrão de comportamento esperado. É definido certo nível de tolerância para o tráfego não completamente condizente ao padrão, de forma que o mesmo não seja classificado como anômalo. Após observada a presença de alguma conexão que esteja ultrapassando o limite estabelecido, um alerta é gerado e o modo de prevenção pode ser acionado.

A única ação pró-ativa capaz de ser acionada consiste no aumento do tempo de resposta entre as requisições recebidas pelo servidor, a fim de evitar o sucesso do ataque. Esta contramedida permite a contenção de ataques à protocolos seguros do tipo "*timing*" (KOCHER, 96) e "*side-channel*" (KESLEY et al., 2000), visto que esses ataques são baseados no estabelecimento de milhares de conexões com o intuito de deduzir a chave privada (BRUMLEY; BONEH, 2003).

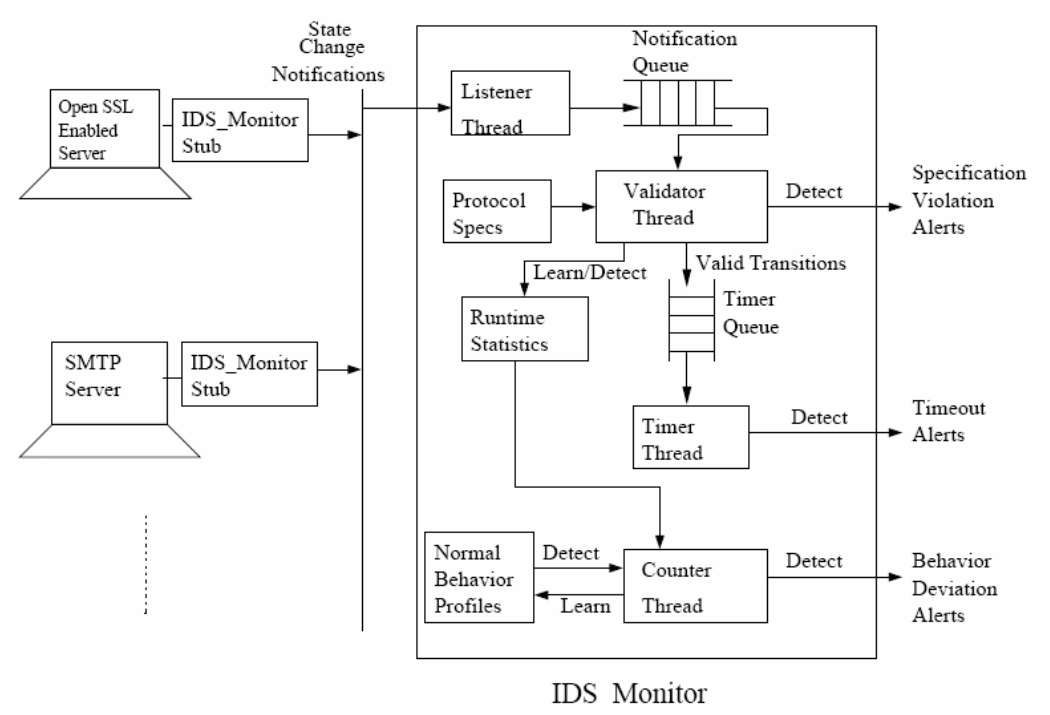

Figura 9. Arquitetura genérica do ProtoMon (Fonte: (JOGLEKAR; TATE, 2004)).

Porém, há controvérsias com relação a este método de contenção, uma vez que existem ataques pontuais, como ataques do tipo DoS, nos quais poucas requisições seriam suficientes para indisponibilizar ou causar algum dano ao servidor ou serviço alvo, conforme pode ser exemplificado pelos boletins de segurança da US-CERT (2005a)(2005b). Isso sem contar que o sistema proposto por (JOGLEKAR; TATE, 2004) não é capaz de atuar somente sobre as conexões e requisições maliciosas, afetando o desempenho de resposta sobre toda aplicação.

3.3 AppArmor

Uma proposta para delimitar o escopo de execução de determinado processo, baseando-se na utilização de esquemas de controle de acesso mandatários – CAM - (SANDHU, 1993), foi

apresentada pela empresa Novell. A ferramenta AppArmor (NOVELL, 2005), como foi nomeada, tem o propósito de inserir controles de leitura, escrita e gravação por usuário e serviço, o que remete ao conceito “caixa de areia” (*SandBox*). Todos os recursos necessários para execução de determinada aplicação são mapeados e, então, é criado um perfil contendo as permissões necessárias para o correto funcionamento.

Com isso, mesmo que os serviços oferecidos ao usuário, como serviço de correio eletrônico, aplicações WEB e compartilhamento de arquivos, contenham algum tipo de vulnerabilidade, o sistema operacional bloqueia o acesso a qualquer recurso que não esteja explicitamente definido. A figura a seguir define o perfil de execução da página “localtime.php” por meio da atribuição de restrições aos recursos do sistema operacional necessários para exibição da página.

```
/usr/sbin/httpd2-prefork^/cgi-bin
localtime.php
{
/etc/localtime r,
/srv/www/cgi-bin/localtime.php r,
/usr/lib/locale/** r,
}
```

Figura 10. Acesso somente leitura (r) aos recursos acessados pela página localtime.php

Esta solução, mesmo que bastante recente e ainda em desenvolvimento, já está inserida nas distribuições do sistema operacional Suse, em sua versão comercial.

3.4 Conclusão

Os trabalhos apresentados neste capítulo trataram de pontos específicos relacionados a detecção, contenção e prevenção de ataques e, até mesmo, da padronização de mensagens de controle para sistemas IDS.

O IDREF propõe a extensão da utilização do padrão do formato de mensagens IDMEF, de forma a possibilitar a troca de mensagens de controle e envio de respostas a partir do IDS Central de gerenciamento ao agente de detecção. O agente Protomon realiza a análise do protocolo por meio da utilização de máquinas de estados e possibilita maximizar o tempo de processamento das requisições em situações de ataque, porém impactando também nas conexões lícitas. A proposta da empresa Novell permite prevenir a ocorrências de ataques restringindo ao máximo os recursos do servidor que são utilizados pela aplicação.

Embora tais propostas tenham seus pontos fortes, ainda existem lacunas para desenvolvimento de alternativas ainda não abordadas que envolvam, por exemplo, a análise do conteúdo de mensagens de protocolos seguros ou que integrem duas ou mais propostas para realização de um projeto mais homogêneo.

Capítulo 4. Proposta de arquitetura e seus componentes

Este capítulo descreve a arquitetura de detecção, seus componentes e a arquitetura do agente de detecção e prevenção de ataques, objetivo principal deste trabalho.

A proposta do **Agente de Detecção, Análise e Contenção de Ataques** (**ADACA**) faz parte de uma arquitetura de detecção de intrusão que é composta por agentes IDS de rede, agentes ADACA e a central de correlação de alertas e gerenciamento (**IDS Central**), mostrado na Figura 11. O agente de aplicação, foco do trabalho, é integrado diretamente à aplicação, o que permite obter maior controle sobre o fluxo de mensagens transacionadas.

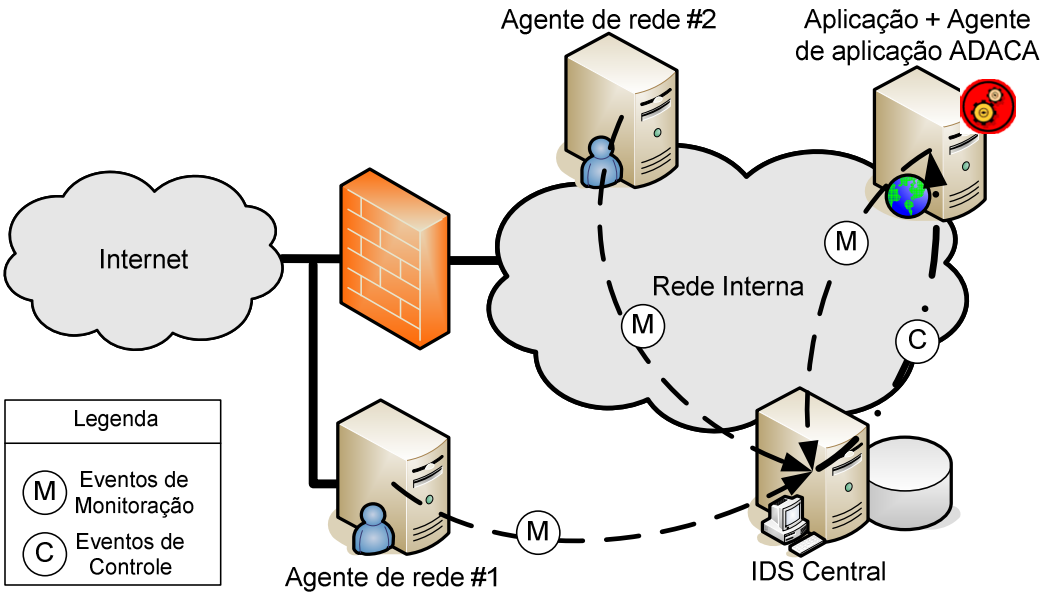

Figura 11. Exemplo de arquitetura de detecção: aplicação com ADACA, sensores de rede e central de gerência e correlação.

A Figura 11 apresenta a topologia de rede adotada em uma solução de IDS completa, estando presente tanto os componentes propostos com o presente trabalho quanto outros

agentes de rede que realizam a detecção de eventos por meio dos dados trafegados no canal de comunicação, contribuindo com o correlacionamento de eventos.

O ADACA é um agente híbrido que pode ser configurado para operar nos modos ativo e passivo, possibilitando realizar a detecção, prevenção e contenção de requisições maliciosas e respostas anômalas ou não autorizadas.

Os atos de prevenção e contenção são obtidos por meio de medidas genéricas, as quais contêm uma ação ou um grupo de ações a serem executadas quando uma medida for instanciada. De acordo com o momento em que for instanciada é que se distingue entre medida de prevenção ou contenção.

O conceito de medida de prevenção é definido por toda interação realizada no sistema operacional pelo ADACA ou mesmo por processamento interno do agente, que tenha por objetivo impedir que determinado ataque ou requisição maliciosa cause qualquer impacto ao sistema alvo. Pode-se citar como exemplo de uma medida de prevenção a inserção de uma medida que visa bloquear determinado endereço IP de origem, o qual tenha sido caracterizado como endereço ofensivo pelo IDS Central.

As medidas de contenção, ou contramedida, são as interações que visam conter ou minimizar o impacto de um ataque quando este já esteja em andamento. Para citar um exemplo pode-se supor a exploração de uma falha que tenha por objetivo obter uma console remota no servidor alvo. Nesta situação, o atacante necessita realizar diversas conexões para explorar e fazer uso da console obtida, as quais poderiam ser bloqueadas por uma medida de contenção.

A prevenção de ataques é realizada em duas diferentes situações. A primeira ocorre quando o agente está configurado em modo ativo, pois a classificação e análise dos dados são realizadas sincronamente com o processamento da requisição pela aplicação. Nesta

configuração, o agente ADACA possibilita o descarte da requisição e até mesmo a inserção de uma medida preventiva para situações nas quais já exista uma medida previamente definida.

O segundo caso é independente do modo de operação do agente. É realizado por meio de mensagens de controle que são configuradas e enviadas por meio da console do IDS Central ao ADACA, nas quais estão contidas as medida preventivas, fruto do correlacionamento de eventos dos agentes distribuídos.

Outra forma de atuação do ADACA é por meio de medidas de contenção, ou contramedidas. Estas medidas somente são aplicadas quando o agente estiver operando no modo passivo, visto que a análise dos dados é realizada assincronamente pelo IDS Local. Assim que houver correspondência de uma medida pré-definida para o resultado da análise e classificação da mensagem, é que se realiza a contenção do ataque.

Para realização da troca de mensagens entre agente e IDS Central foram definidos eventos de monitoração e controle (Figura 11), ambos podendo ser gerados e enviados assincronamente. Os eventos de monitoração são as mensagens nas quais estão contidos os alertas detectados pelos agentes distribuídos ao longo do ambiente de rede e enviados ao IDS Central a fim de informá-lo sobre a ocorrência das atividades anômalas. Já os eventos de controle são mensagens enviadas unidirecionalmente pelo IDS Central aos agentes, quando estes suportam tais mensagens, e neles estão contidos os comandos que deverão ser executados no agente.

Uma mensagem de controle emitida pelo IDS Central ao agente pode conter comando de configuração, inserção de medida ou programação de uma nova medida. Um comando de configuração permite, por exemplo, ativar e desativar o agente, configurar o modo de operação (ativo ou passivo), controlar geração de registros (*logs*), entre outras.

O formato da mensagem e o protocolo de comunicação utilizado para informar o IDS Central a respeito dos eventos ocorridos são baseados no padrão IDMEF proposto em (WOOD; ERLINGER, 2002) e (DEBAR; CURRY; FEINSTEIN, 2005), apresentado na seção 2.2.2.

A arquitetura do ADACA foi desenvolvida levando-se em consideração a portabilidade entre diversos sistemas operacionais e a possibilidade de integração a diferentes aplicações. Para isto, foi definido um núcleo único genérico no qual estão compreendidos os componentes de análise, comunicação e atuação. A API ADACA define a interface deste núcleo com a aplicação. Este núcleo também interage com módulos externos por meio de interfaces padronizadas apresentadas a seguir:

- Interface com acionador: informa a medida a ser adotada pelo acionador;
- Interface com classificador: possibilita a classificação da mensagem;
- Interface com IDS Local: viabiliza a análise da mensagem e detecção de alertas;
- Interface com IDS Central: viabiliza a troca de mensagens de monitoração e controle;

A Figura 12 apresenta os elementos que compõem o ADACA e as interações realizadas entre eles. Posteriormente, neste capítulo, será explanado com maiores detalhes o modo operacional da arquitetura de detecção.

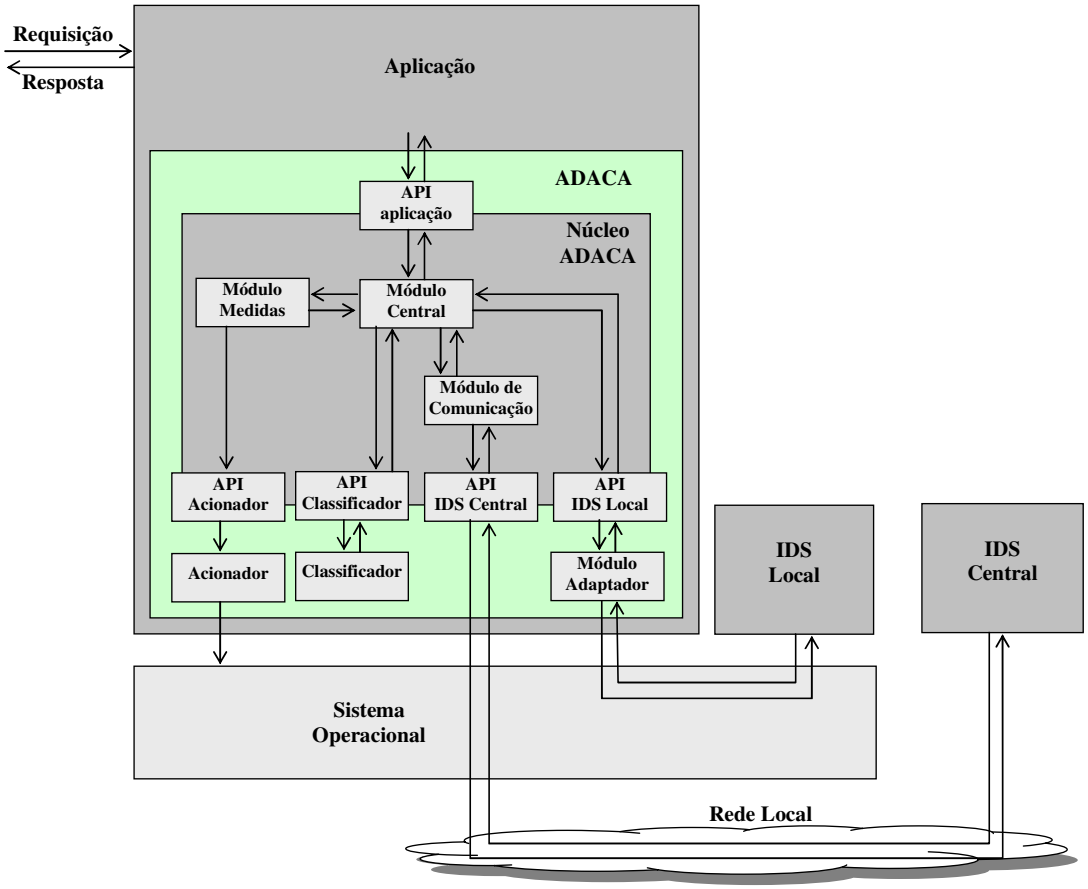

Figura 12. Arquitetura do ADACA e suas interações com outros componentes

4.1 API ADACA

A API ADACA possibilita que a aplicação repasse ao ADACA mensagens para serem analisadas. Desta forma, é possível ao agente observar e analisar as mensagens de requisição e resposta. Na requisição, a aplicação, após receber uma mensagem, deve repassá-la ao ADACA utilizando a função *Analisa_dados( )* da API ADACA. Na resposta, a aplicação, após a formatação da mensagem de resposta, pode submetê-la para análise utilizando esta mesma função.

A função *Analisa_dados( )* necessita, além da mensagem, informações sobre o endereço IP e porta origem, endereço IP e porta destino, sentido e contexto. O contexto é qualquer informação adicional, mantida pela aplicação, que possa ser útil ao agente ADACA. A Figura 13 apresenta a sintaxe desta função e seus argumentos.

```
Estado = Analisa_dados(sentido,IPo, Po, IPd, Pd, Mensagem, Contexto);
```

Figura 13. Sintaxe da função *Analisa_Dados( )* da API ADACA

Como já foi citado, o agente pode operar nos modos ativo e passivo. Esta configuração é obtida de um arquivo de configuração no momento de iniciação do agente, porém pode ser redefinida através de mensagens de controle proveniente do IDS Central ou mesmo por meio da função *Modo_operacao( )* da API ADACA.

Quando o agente estiver configurado para operar no modo ativo, a função *Analisa_dados( )* aguarda o processo de análise ser concluído para retornar o resultado à aplicação. Caso opere no modo passivo, a função retorna imediatamente e o processo de análise ocorre paralelamente ao processamento da mensagem pela aplicação.

Mesmo quando o agente ADACA opera no modo passivo podem ser ativadas medidas de contenção e prevenção.

4.2 Módulo central

O módulo central coordena as atividades de análise e ações realizadas pelo agente ADACA. Quando um pedido de análise é requisitado pela aplicação, o módulo central é invocado. Este, por sua vez, aloca o pedido em uma tabela de controle e, em seguida, ativa a função

Classifica_mensagem( ) da API Classifica. No retorno desta função, o resultado da classificação é registrado na tabela e, em seguida, é requisitado ao IDS Local a análise da mensagem. Terminada a análise, o módulo central registra o resultado na tabela de controle.

Caso o IDS Local tenha detectado alguma mensagem anômala ou maliciosa, é ativado o módulo comunica para submeter o alerta ao IDS Central. Independente do resultado do IDS Local, o módulo de medidas é ativado a fim de verificar se para a mensagem em análise existe alguma medida registrada. O módulo de medidas utiliza como base os parâmetros registrados na tabela de controle do ADACA e na tabela de medidas cadastradas. Caso exista, a medida é invocada e registrada na tabela de incidências. A Tabela 3 apresenta um exemplo da tabela de incidências.

Tabela 3. Tabela-exemplo utilizada para armazenamento das incidências detectadas

#	Alerta	Classific.	IP Orig.	P. Orig.	IP Dest.	P. Dest.	Mensagem	Contexto	Id. medida	Hora Data
1	78	GET	10.0.0.10	4567	10.0.0.2	80	GET /etc/passwd		1	14:30:4567 05/01/06
2	67	GET	10.0.0.6	3205	10.0.0.2	80	GET /inc/global.asa			14:24:0345 05/01/06
3	X	OPTIONS	10.0.0.13	3890	10.0.0.2	443	OPTIONS /		3	14:35:2309 05/01/06
4	54	POST	10.0.0.11	2015	10.0.0.2	443	POST /../../boot.ini			14:29:3678 05/01/06

A tabela de incidências, apresentada na Tabela 3, registra os alertas, eventuais medidas executadas e relaciona, também, as mensagens que geraram o alerta.

4.3 API Classifica

Esta interface realiza a ponte entre o módulo central e o classificador. A principal função definida nesta API é *Classifica_mensagem( )* que possui como parâmetros de entrada o

endereço IP e porta de origem, endereço IP e porta destino e a mensagem a ser classificada. Como resultado da função é obtida a classificação da mensagem.

O classificador é um módulo opcional da arquitetura do ADACA. Sua função principal é classificar a mensagem do protocolo de aplicação. Como exemplo, pode-se citar as mensagens GET, POST e HEAD do protocolo HTTP e as mensagens HELO, RCPT TO, VRFY, MAIL FROM do protocolo SMTP.

Devido à variância das regras de classificação, as quais são dependentes da aplicação, o classificador foi definido como um módulo externo ao núcleo do ADACA. De acordo com o perfil estipulado, o classificador poderia implementar, inclusive, uma máquina de estado para análise das mensagens trocadas com um mesmo parceiro, incrementando funcionalidades ao processo de classificação.

4.4 API IDS Local

O IDS Local, apesar de não fazer parte do agente, é um elemento fundamental para o funcionamento da arquitetura de detecção, uma vez que ele é responsável por analisar a mensagem interceptada junto à aplicação. Para permitir a integração ao ADACA foi definida a API IDS Local. Esta API tem como objetivo definir uma interface padronizada para acoplamento de diferentes tipos de IDS.

Por padrão, a API IDS Local converte os dados para o formato PCAP[2], este suportado por analisadores de pacotes e sistemas de detecção, como o Snort e Cisco NetFlow. De

[2] Maiores informações podem ser obtidas em http://www.tcpdump.org

qualquer forma, é possível definir um módulo adaptador responsável por converter os dados recebidos para o formato específico do sistema de detecção a ser empregado.

Para realizar a troca de informações, foi definida a função *Analisa_IDS( )*. Ao final do processo de análise, a função retorna um dos dois possíveis resultados: normal ou anômalo. No caso de anômalo, a função também informa o alerta gerado pelo IDS Local.

4.5 Tabela de controle

Para cada mensagem submetida ao agente ADACA para análise é criado um registro na tabela de controle. Neste registro são mantidas informações relevantes associadas ao processo de análise desta mensagem, incluindo os parâmetros informados pela aplicação.

As informações relacionadas ao sentido da mensagem, endereço IP origem, porta origem, endereço IP destino, porta destino, mensagem e contexto foram obtidas da chamada da função *Analisa_dados( )* da API ADACA. O instante é determinado e registrado pelo próprio agente ADACA. Os resultados da classificação e análise da mensagem são registrados após o processamento do classificador e análise do IDS Local, respectivamente.

Os próximos módulos a serem apresentados irão utilizar as informações registradas na tabela de controle.

4.6 Módulo Comunica

O Módulo Comunica é responsável por gerenciar toda comunicação realizada entre o ADACA e o IDS Central e adequar os eventos de monitoração ao padrão IDMEF. Já para o recebimento de eventos de controle é utilizado o formato IDREF (SILVA, 2004), apresentado na seção 3.1.

A comunicação entre o agente ADACA e o IDS central é realizada assincronamente, de forma que os eventos de monitoração e de controle sejam independentes. Esses eventos são transmitidos utilizando o protocolo IDXP por meio da API IDS Central.

4.7 API IDS Central

A API IDS Central define a interface de funções entre o agente ADACA e o IDS Central remoto por meio das funções *Enviar_alerta( )* e *Callback_controle( ).* A função *Enviar_alerta( )* destina-se a notificar os eventos de monitoração ao IDS Central. Já a função *Callback_controle( )* permite receber mensagens de controle provenientes do IDS Central. O módulo central do agente é responsável por tomar as devidas providências, de acordo com o valor do parâmetro *tipo_controle* informado na função *Callback_controle( ).* Caso seja uma mensagem de comando de configuração do modo de operação do agente, a função *Modo_operacao( )* da API ADACA é acionada. No caso de criação ou inserção de uma medida é ativada uma função do módulo medidas.

4.8 Módulo Medidas

O módulo medidas tem como objetivos gerenciar as medidas existentes, registrar novas medidas a serem executadas e analisar a existência de alguma medida pré-definida para cada uma das mensagens.

Para melhor entendimento do funcionamento deste módulo será primeiramente apresentado o gerenciamento das medidas existentes. Para realizar o controle das medidas, foi definida uma tabela de medidas que contém a identificação da medida, a condição em que é ativada, o tipo da ação a ser executada e o tempo de execução, conforme mostrado na Tabela 4.

Tabela 4. Exemplo de tabela de medidas.

Id Medida	Classificação	ID Alerta	Ação	Duração (min)
1		78	Bloqueia_ip	10
2		89	Bloqueia_porta	5
3	OPTIONS		Bloqueia_porta	10
4		63	Shutdown	

Com relação ao gerenciamento das medidas, ainda existe a funcionalidade de registrar e remover medidas com base nas informações recebidas pelo módulo central. Este procedimento é realizado por meio da função *n_medida = Incluir_medida(Classificação, ID_Alerta, Ação, Tempo)* e *Excluir_medida(n_medida),* respectivamente.

Com base na tabela de controle, apresentada na seção 4.5, é realizada uma consulta à tabela de medidas a fim de verificar a existência de uma medida para a mensagem. Caso

exista, este módulo torna-se responsável por solicitar ao módulo acionador a inserção desta medida pré-definida e controlar seu tempo de execução.

Para realização desta tarefa foi definida a função *Executar_medida(entrada_tabela_controle).* Como retorno desta função é informado ao módulo central a identificação da medida executada.

Uma outra tabela, denominada tabela de execução de medidas, relaciona todas as medidas executadas pelo agente ADACA. Nesta tabela estão presentes os endereços e portas de origem e destino do ataque, o número da medida adotada e a data e hora. Esta tabela tem como objetivo controlar o tempo de execução de cada uma das medidas de acordo com a especificação da mesma, sendo que ao término do tempo a medida é automaticamente desativada. A Tabela 5 apresenta um exemplo da tabela de medidas executadas.

Tabela 5. Tabela de exemplo de execução de medidas

Id Medida	Ação	Dados	Data Início	Hora Início	Data Término	Hora Término
12	Bloqueia_ip	200.100.20.1	05/01/06	14:30	05/01/06	14:40
8	Bloqueia_porta	200.200.20.1:80	05/01/06	14:35	05/01/06	14:50
2	Desativa_interface	Eth0	06/01/06	00:25		
4	Shutdown		06/01/06	14:00		

Como último propósito, mas não menos significativo, a Tabela 5 serve como repositório de registros (*logs*) a serem utilizados para fins de auditoria.

4.9 API Aciona

A comunicação entre o módulo medidas e o módulo acionador é estabelecida por meio da API Aciona, que relaciona as ações que podem ser realizadas sobre os recursos do sistema

operacional. Como exemplo, pode-se citar a criação de uma regra junto ao *firewall* local, a reiniciação de uma determinada conexão e o redirecionamento do tráfego de um endereço IP origem para outro destino, como, por exemplo, uma *honeynet*[3].

O módulo acionador é a porção do agente responsável por mapear as ações definidas na API Aciona em comandos do sistema operacional.

Foi definido, no presente momento, um conjunto básico de funções na API Aciona para controlar as ações realizadas junto ao sistema operacional, sendo:

- Estado bloqueia_IP (IPo);
- Estado desbloqueia_IP(IPo);
- Estado bloqueia_porta (IPo,Pd);
- Estado desbloqueia_porta(IPo,Pd);
- Estado Shutdown();
- Estado Desativa_interface(interface);
- Estado Ativa_interface(interface);

4.10 IDS Central

[3] Rede simulada para atrair e avaliar a forma conduzida por atacantes. Maiores informações podem ser obtidas no site do projeto Honeynet: http://www.honeynet.org

Este elemento é dispensável para o funcionamento do agente, porém, sem ele não existiria a arquitetura de detecção e somente um sistema de detecção isolado. Como já foi citado, o IDS Central possui função de gerenciador de agentes e repositório unificado de alertas de todos agentes distribuídos ao longo da rede.

Como gerenciador de agentes, o IDS Central possibilita o monitoramento do estado e configuração do modo de operação de cada um dos agentes, a programação de novas medidas e o pedido de inserção de uma medida. Como repositório de alertas, permite a correlação dos eventos que forem acontecendo ao decorrer do tempo em todo ambiente.

Com relação à comunicação, o IDS Central é capaz de receber eventos de monitoração formatados sobre o padrão IDMEF e enviar eventos de controle aos agentes utilizando o formato IDREF. Esses dois tipos de eventos são independentes e ocorrem assincronamente durante a operação do sistema. Os eventos de controle somente serão disparados por meio de interação humana, de forma que seja possível analisar claramente a veracidade das informações correlacionadas e impedir que uma medida de prevenção seja adotada equivocadamente.

4.11 Operação

Para representar o modo de operação do ADACA serão descritos os detalhes das interações entre os módulos e componentes apresentados anteriormente na Figura 12.

Após o estabelecimento da conexão, seja utilizando um túnel criptografado por meio de protocolo de aplicação segura (HTTPS, SFTP, entre outras) ou em texto plano, inicia-se a troca de mensagens entre cliente e servidor.

Quando uma requisição for recebida ou imediatamente antes de uma resposta ser enviada ao cliente, a aplicação deve submetê-la ao ADACA para análise. Para isto, ativa a função *Analisa_dados( )* da API ADACA que é responsável pelo registro dos parâmetros passados na função em uma nova entrada da tabela de controle. Neste momento, se o agente ADACA estiver operando no modo passivo, a função retorna imediatamente para a aplicação. Caso opere no modo ativo, a função irá retornar somente após finalizada toda análise da mensagem.

Dando continuidade ao processamento, o módulo central do ADACA ativa o módulo classificador, por meio da função *Classifica_mensagem( )* da API Classifica, e, em seguida, ativa a análise da mensagem junto ao IDS Local, o qual utilizará suas técnicas de detecção a fim de determinar a presença de dados maliciosos. Ao final destes processos, os resultados são retornados ao módulo central, bem como o alerta detectado, se esta for a situação. Caso exista um alerta, este é encaminhado ao módulo comunica para ser formatado e enviado ao IDS Central.

Com base na entrada recém registrada na tabela de controle, o módulo medidas é ativado para determinar a presença de alguma medida definida para a mensagem em análise. Caso não haja qualquer relacionamento com a tabela de medidas (Tabela 4), a requisição pode ser considerada lícita e a função *Analisa_Dados( )* da API ADACA retorna informando que não foi detectada anomalia na mensagem e a libera para processamento. Do contrário, as informações sobre a mensagem, classificação e análise são gravadas na tabela de incidências (Tabela 3), juntamente com a medida a ser adotada, se este for o caso.

Neste caso, a medida é instanciada de acordo com as informações constantes da entrada na tabela de controle e sua execução é repassada ao módulo acionador, o qual irá realizar as devidas interações com o sistema operacional para prevenir ou conter alguma ameaça.

No módulo comunica, o alerta é formatado de acordo com o padrão IDMEF e enviado ao IDS Central. Paralelamente a estas atividades, o IDS Central realiza constantemente o correlacionamento de eventos recebidos por todos os agentes.

Quando surgir a necessidade do IDS Central enviar um evento de controle, seja destinado à configuração do modo de funcionamento do agente, inserção ou programação de uma medida de prevenção, a função *Callback_controle( )* da API IDS Central é ativada e o evento é direcionado ao módulo central do ADACA. Este módulo determina a que propósito se destina este evento de controle e interage com o módulo medidas ou com a API ADACA.

Os mesmos procedimentos acima descritos são utilizados para analisar o tráfego de saída da aplicação, sendo que o módulo classificador pode ser mais utilizado que o próprio IDS Local, visto que as condições de classificação seriam mais dinâmicas baseando-se nos perfis adotados.

Capítulo 5. Implementação e ambiente de testes

O objetivo a ser alcançado com a implementação de um protótipo do ADACA dentro do contexto da arquitetura de detecção apresentada é realizar uma prova de conceito que possibilita avaliar a viabilidade de implementação do agente integrado a uma aplicação real.

5.1 Escopo da implementação

Para possibilitar a validação da arquitetura de detecção apresentada no capítulo anterior, foi desenvolvido o agente genérico de detecção em nível de aplicação ADACA, conforme ilustrado pela Figura 12. Porém, foi necessário realizar algumas limitações no escopo do desenvolvimento do protótipo.

O componente IDS Local, indispensável para o funcionamento do ADACA, não é o foco desta pesquisa. Portanto, para esta validação foi considerada a adoção da solução Snort como mecanismo de detecção, uma vez que esta ferramenta possui funcionalidades específicas que satisfazem a necessidade da arquitetura proposta.

5.1.1 Simplificações

O componente IDS Central foi implementado parcialmente. Foi desenvolvida uma central de gerenciamento que possibilita enviar eventos de controle. O mecanismo de correlação de eventos é um componente de alta complexidade sobre o qual existem diversos

trabalhos específicos a respeito (WANG; ABDEL-WAHAB, 2005) (XU; NING, 2004) (WU; FOO; MEI; BAGCHI, 2003), não sendo do escopo do trabalho e, portanto, não considerado para a etapa de implementação.

Com relação à comunicação de eventos de monitoração ao IDS Central, não será utilizado o protocolo de comunicação IDXP, baseado na especificação IDP. Até o momento da etapa de implementação, somente foi encontrada uma biblioteca na linguagem Java que fosse capaz de implementar o protocolo IDXP, dificultando a integração com este projeto. Como o protocolo não é foco do trabalho este recurso foi substituído por um simulador de comunicação na API Comunica do ADACA.

5.2 Definição da aplicação

Antes de ser iniciada a fase de codificação, foi necessário definir uma aplicação representativa para a fase de testes. Neste contexto, a aplicação escolhida para ser utilizada foi o serviço WEB, uma vez que o protocolo utilizado por este serviço contempla a versão em texto-plano (HTTP) e a versão segura (HTTPS), implementada sobre o SSL/TLS.

Uma vez definido o tipo de aplicação que seria utilizado como prova de conceito, realizou-se uma pequena busca acerca das peculiaridades dos servidores WEB mais utilizados, sempre focando as interfaces existentes para a integração com o ADACA. Desta forma, o servidor WEB Apache (“www.apache.org”) foi escolhido pois, além de ser uma solução de código aberto, é o servidor utilizado por aproximadamente 67% dos 15 milhões de sites avaliados, de acordo com a estatística da Netcraft (2006).

Outra vantagem existente com esta escolha é com relação ao suporte a plataformas operacionais distintas, possibilitando a portabilidade do agente de detecção a diversos sistemas operacionais.

Para a prova de conceito foi utiliza a versão 1.3.34 do Apache, atualmente a mais recente da distribuição 1.3. As distribuições 2.0 e 2.2 não foram consideradas, pois realizam o processamento das mensagens em múltiplas linhas de execução (*multithreads*), o que dificultaria a codificação do protótipo.

5.3 Ambiente de desenvolvimento

Para desenvolvimento dos componentes do ADACA foi utilizada a linguagem de programação C. Esta linguagem foi escolhida pela portabilidade e por ser utilizada na implementação do servidor WEB Apache, facilitando a adaptação deste aplicativo ao protótipo.

O protótipo foi desenvolvido no sistema operacional Fedora Core 4 da Red Hat e o compilador utilizado foi o GCC versão 4.0.

5.4 Implementação do agente

Conforme apresentado no capítulo 4, o ADACA foi projetado em módulos específicos e bem definidos, facilitando o desenvolvimento e codificação do protótipo. A seguir, serão apresentados alguns detalhes da fase de implementação.

5.4.1 Integração do ADACA à aplicação

Para tornar possível integração do ADACA, se fez necessário realizar uma análise detalhada do código-fonte e das funções utilizadas pelo Apache, no tratamento de requisições e respostas. Foi identificado que a função "*ap_process_request()*", do arquivo "http_request.c", realiza o processamento das requisições no servidor Apache. Em seguida, foram mapeadas as variáveis que armazenam as informações a respeito da mensagem, como endereços IPs e portas.

Até alcançar a função Analisa_Dados() da API ADACA, inserida antes da função "*ap_process_request()",* o Apache realiza diversas validações da mensagem da requisição entrante com o intuito de verificar a presença de requisições incompletas. Se este for o caso, as informações sobre a mensagem não passam pela API ADACA e é retornada uma página de erro padrão do próprio Apache.

Para realizar a análise da resposta a função Analisa_Dados() da API ADACA é ativada entre a função "*ap_process_request()"* e a função *"ap_bhalfduplex()"*. A função *"ap_bhalfduplex()"* finaliza a formatação da mensagem de resposta antes que sejam gerados os registros (logs) de auditoria e o envio da resposta ao cliente pelo Apache.

5.4.2 Modulo Comunica

Para implementação do Módulo Comunica foi utilizada a biblioteca LibIDMEF (POPPI, S.; MIGUS, A.; MCALERNEY, J., 2005) que possibilita a padronização dos alertas ao formato

IDMEF. Também, foi identificado que o Snort tem como uma de suas funcionalidade a geração de alertas no formato IDMEF, baseado nesta mesma biblioteca.

No entanto, como a arquitetura do ADACA visa possibilitar a utilização de outros mecanismos de detecção como IDS Local, a biblioteca LibIDMEF foi integrada ao módulo comunica para formatação dos alertas, ao invés de obtê-los padronizados a partir do Snort. Apesar de não ser totalmente documentada, a integração ao módulo comunica pôde ser realizada de forma bastante satisfatória, sem haver necessidade de despender muito tempo.

5.4.3 IDS Local

Conforme citado anteriormente, apesar do IDS Local não ser o objeto de pesquisa deste projeto, sua utilização é imprescindível. Portanto, foi definido que a solução Snort deveria ser utilizada para análise das mensagens recebidas pelo ADACA, pois, além de ser uma ferramenta de livre distribuição e portável para diversos sistemas operacionais, possui funcionalidades que facilitam a integração ao ADACA. Entre elas, pode-se destacar a análise de pacotes por meio de arquivo de entrada no formato PCAP e o envio dos alertas detectados por *socket* interno, possibilitando a comunicação entre processos.

5.5 Ambiente de testes

Para realização dos testes se fez necessário definir um ambiente que houvesse as mesmas características de um ambiente real e que possibilitasse a execução dos testes de viabilidade

com a utilização do ADACA como solução de segurança. A arquitetura apresentada na Figura 11 foi ligeiramente simplificada, levando em consideração um servidor WEB com o ADACA integrado à aplicação, a central de gerência e clientes, conforme mostra a Figura 14.

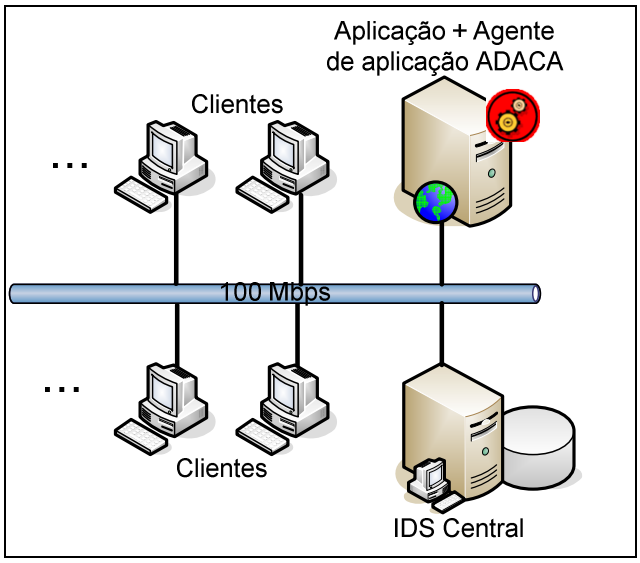

Figura 14. Ambiente estruturado para validação do agente

O servidor WEB e o IDS Central foram alocados em duas máquinas equipadas com processador AMD Atlhon XP 3.0 GHz, 1 Gbyte de memória RAM e sistema operacional Fedora Core 4. Para os clientes foram utilizadas estações com diversas configurações, visto que a simulação de requisições HTTP não demanda alto processamento.

5.6 Testes

A etapa de testes teve como principal objetivo realizar o estudo de viabilidade do ADACA e, consequentemente, medir o tempo gasto por ele para verificar, em cada uma das mensagens, a

presença de dados maliciosos ou respostas não-autorizadas. Alguns testes foram definidos para avaliar as funcionalidades do agente, conforme as etapas de operação descritas na seção 4.11.

Portanto, também se fez necessário avaliar as características dos dois modos de operação do agente, sendo considerada três situações:

- Aplicação WEB sem agente;
- Aplicação WEB e ADACA operando no modo passivo;
- Aplicação WEB e ADACA operando no modo ativo.

Para cada situação, foram disparadas requisições normais e maliciosas com objetivo de diversificar o tipo de tráfego e explorar as funcionalidades implementadas pelo agente. Portanto, foi possível observar cada uma das etapas do processo de classificação, análise e tomada de decisão do agente.

Também, foram enviados eventos de controle contendo mensagens para inserção e programação de medidas, a partir do IDS Central ao ADACA, com o intuito de observar a efetividade desta funcionalidade.

Os resultados dos testes apresentados nesta seção serão discutidos no próximo capítulo.

Capítulo 6. Análise de resultados

Neste capítulo serão apresentados os resultados obtidos com a utilização do ADACA para as situações consideradas no capítulo anterior.

Como primeiro ponto relevante da análise, deve ser observado que o protótipo desenvolvido do agente ADACA operou de forma satisfatória, conforme a proposta inserida no capítulo Capítulo 4. A partir deste resultado inicial é que se pôde dar continuidade dos testes descritos na seção anterior.

Foi, então, realizada a medição do tempo gasto pelo Apache sem acoplamento do agente para executar a função "*ap_process_request( )*", a qual realiza o processamento da requisição no Apache. Em seguida, foram colhidos os resultados computando a sobrecarga adicionada com a inserção do ADACA nos modos passivo e ativo.

A medida do tempo foi realizada por meio da função "*getsystemtime( )*", nativa da linguagem C. Para as medições com o ADACA em operação, esta função foi posicionada antes da chamada à função "*Analisa_dados()*" da API ADACA e logo após a função "*ap_process_request( )*" do Apache. No caso das medições sobre a aplicação original, a função para obtenção do tempo foi posicionada antes e depois da função "*ap_process_request()*". Este ponto foi escolhido, pois reflete somente a sobrecarga introduzida com a utilização do ADACA.

Por meio de uma amostra de cem requisições a uma mesma URL, a média ponderada do tempo de processamento de cada requisição obtida para o Apache original foi de 101 (± 31) microssegundos. Com base na mesma amostragem, o valor médio obtido para o Apache com o ADACA no modo ativo foi de 270 (±94) microssegundos. Não foram observados

atrasos significativos no processamento de uma mensagem individual com o agente operando no modo passivo. Portanto, observa-se que a utilização do ADACA no modo ativo introduz um aumento da latência de resposta de aproximadamente duas vezes quando comparado à aplicação original e ao modo passivo.

A análise dos resultados obtidos permite concluir que a inserção do ADACA no servidor WEB Apache não acarretou perdas consideráveis de tempo, visto que a latência adicionada por requisição é da ordem de microssegundos.

6.1 Limitações da implementação

Durante a fase de implementação, foi considerada a utilização da proposta IDREF, introduzida por Silva (2004). Esta proposta possibilitaria enviar mensagens de controle ao ADACA a partir do IDS Central, com base no mesmo modelo das mensagens IDMEF. Porém, a biblioteca disponibilizada pelo autor foi construída na linguagem Java e a integração com a linguagem C demandaria muito tempo devido às dificuldades de integração dessas duas linguagens.

Outro fator limitante desta implementação é que o protótipo desenvolvido não é reentrante, impossibilitando realizar análises de mensagens em paralelo.

Capítulo 7. Conclusão

Este trabalho apresentou a arquitetura de detecção usualmente utilizada pelos sistemas IDS e IPS e a proposta de uma arquitetura de agente de aplicação que fosse capaz de analisar mensagens de requisições e respostas. Como principal motivador do trabalho, pode-se destacar a atual incapacidade dos agentes tradicionais de realizar análise de fluxos de mensagem que focam uso de protocolos de comunicação seguros, como, por exemplo, SSL e TLS.

O agente ADACA, da forma como foi proposto, pode ser posicionado nas extremidades do canal cifrado ou, então, utilizado em conjunto com o módulo de decodificação de mensagens que compartilham a chave privada.

Os resultados preliminares obtidos mostraram a capacidade de prevenir o processamento de requisições maliciosas e o envio de conteúdo não-autorizado na resposta. Além disso, foi possível instanciar as medidas de contenção e prevenção por meio do módulo medidas e executá-las com o módulo acionador. No caso de execução de uma medida de contenção (agente no modo passivo), notou-se que, para uma página contendo imagens, o carregamento não se deu por completo, pois a medida surtiu efeito antes mesmo que as requisições para aquisição das imagens fossem processadas.

Com relação ao desempenho, este trabalho apresentou o impacto da sobrecarga do processamento realizado pelo agente nos diferentes modos de operação. Esta sobrecarga é decorrente, principalmente, das análises realizadas pelo IDS local, o qual é dependente de cada aplicação e das técnicas de análise de pacotes por ele utilizada.

A principal desvantagem observada com a arquitetura proposta é a necessidade de modificar a aplicação para integração da API ADACA, tornando necessário realizar um estudo das APIs contidas na aplicação para tal.

A análise dos resultados obtidos com a medição do tempo despendido para análise, classificação e tomada de decisão por parte do ADACA permitiu concluir que a presente proposta é bastante aceitável, visto que a latência adicional inserida em cada mensagem é da ordem de microssegundos quando utilizado o modo ativo e não considerável para o modo passivo.

Finalizando, pode-se citar como contribuições importantes do trabalho a proposta de uniformização da taxonomia de classificação de agentes de detecção e a definição de um modelo de agente de aplicação e seus principais componentes, que pode ser utilizado como referência para trabalhos futuros da área de detecção de intrusos.

7.1 Trabalhos futuros

Algumas propostas surgiram no decorrer deste projeto, as quais são agora apresentadas como possíveis caminhos para continuidade do desenvolvimento do ADACA e da arquitetura de detecção.

No trabalho apresentado, o classificador tem uma função bastante simples, porém fundamental, a qual poderia ser incrementada por meio da implementação de uma máquina de estados que possibilitaria definir novas funcionalidades para este módulo, como a análise das mensagens trocadas com um mesmo parceiro.

As funções definidas para a API Aciona apenas abragem um conjunto mínimo de funções. Portanto, é necessário realizar melhorias nesta API de forma a torná-la genérica, com um conjunto de funções mais completo para as atividades de prevenção e contenção.

A formatação das mensagens de controle utilizando a proposta IDREF e a implementação do protocolo de comunicação IDXP são dois pontos importantes para a padronização e interoperabilidade entre sistemas de detecção de intrusos. Para tal, deve-se criar uma interface de integração entre as linguagens Java e C ou ainda realizar a interpretação do código fonte das bibliotecas disponíveis em Java com a finalidade de reproduzir o mesmo feito na linguagem C.

Referências

ABBES, T; BOUHOULA, A; RUSINOWITCH, M. Protocol Analysis in Intrusion Detection Using Decision Tree. IEEE International Conference on Information Technology: Coding and Computing, v. 1, p. 404-408, 2004.

BACE, R.; MELL, P. Intrusion Detection Systems. NIST - National Institute of Standards and Technology, Special Publication, 2001.Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf>

BURKHOLDER, P. SSL Man-in-the-Middle Attacks. SANS Reading Room, 2002. Disponível em:<http://www.sans.org/rr/whitepapers/threats/480.php>

BREACH. BreachView SSL White Paper. 2005 Disponível em: <http://www.breach.com/downloads/product/BreachView_SSL_White_Paper.pdf>.

BRUMLEY, D.; BONEH, D. Remote Timing Attacks are Practical. 12th USENIX Security Symposium, 2003.

BUCHHEIM, T.; ERLINGER, M.; FEINSTEIN, B.; MATTHEWS, G.; POLLOCK, R.; BETSER, J.; WALTHER, A. Implementing the Intrusion Detection Exchange protocol. Computer Security Applications Conference. Proceedings 17th Annual, p. 32-41, Dezembro, 2001.

CERT.br - Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil. Estatísticas do CERT.br. Disponível em: <http://www.cert.br/stats/incidentes/>. Acesso em: 11 Jan. 2006.

DASGUPTA, D.; GONZALEZ, F.; YALLAPU, K.; GÓMEZ, J.; YARRAMSETTII, R.; DUNLAP, G.; GREVEAS, M. CIDS: An Agent-based Intrusion Detection System. Computers & Security Journal, v. 24, issue 5, p. 387-398, Agosto, 2005

DEBAR, H.; CURRY, D.; FEINSTEIN, B. The Intrusion Detection Message Exchange Format. Internet Draft, IETF, Janeiro 2005. <http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-14.txt>.

ENDORF, C., SCHULTZ, E., MELLANDER, J. Intrusion Detection & Prevention. Ed. McGraw-Hill. 2004.

FEINSTEIN, B.; MATTHEWS, G.; WHITE, J. The Intrusion Detection Exchange Protocol (IDXP). Internet Draft, IETF, Outubro, 2002. Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-idwg-beep-idxp-07.txt>.

GUPTA, D.; BUCHHEIM, T.; FEINSTEIN, B.; MATTHEWS, G.; POLLOCK, T. IAP: Intrusion Alert Protocol. Internet Draft, IETF, Fevereiro, 2001. Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-idwg-iap-05.txt>.

JOGLEKAR, S. P.; TATE, S. R. ProtoMon: Embedded Monitors for Cryptographic Protocol Intrusion Detection and Prevention. IEEE International Conference on Information Technology: Coding and Computing, v. 1, p. 81-88, Abril, 2004.

KAHN, C.; PORRAS, P. A.; STANIFORD-CHEN, S.; TUNG, B., A Common Intrusion Detection Framework. Journal of Computer Security, Julho, 1998.

KELSEY, J; SCHEINER, B.; WAGNER, D.; HALL, C. Side Channel Cryptanalysis of Product Ciphers. Journal of Computer Security, v. 8, n. 2-3, p. 141-158, 2000.

KNOBB, F. SnortSam, 2006. Disponível em: <http://www.snortsam.net>.

KOCHER, P. C. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology, p. 104-113, 1996.

LECKIE, T.; YASINSAC, A. Metadata for Anomaly Based Security Protocol Attack Detection. IEEE Transactions on Knowledge and Data Engineering, Volume 16, Number 10, Outubro, 2004.

MCAFEE. Encrypted Threat Protection: Network IPS for SSL Encrypted Traffic. 2005. Disponível em:<http://www.mcafee.com/us/local_content/white_papers/wp_encr_th_prot.pdf>

MICROSOFT. Microsoft Security Bulletin (MS00-078): Patch Available for 'Web Server Folder Traversal' Vulnerability. Outubro, 2000. Disponível em: <http://www.microsoft.com/technet/security/bulletin/MS00-078.mspx>.

NEW, D.; ROSE, M. Reliable Delivery for Syslog. RFC3195. Novembro, 2001.

NETCRAFT. Web Server Survey Archives. Fevereiro, 2006. Disponível em: < http://news.netcraft.com/archives/web_server_survey.html>

NIST. Acquiring and Deploying Intrusion Detection Systems. ITL Computer Security Bulletins. NIST. Novembro, 1999.

NOVELL. Novell AppArmor Powered by Immunix Administration Guide. Novell (Suse), Outubro, 2005.

PALLER, A. New reports finds significant shifts in software being targeted by attackers. SANS TOP 20 MOST CRITICAL INTERNET VULNERABILITIES REPORT. Novembro, 2005.

PLAGGEMEIER, M.; TÖLLE, J. Secure Shell Proxy Intrusion Detection, Proc. of R.T.O. Information Systems Technology Panel Symposium on Real Time Intrusion Detection, Maio, 2002.

PTACEK, T. H., NEWSHAW, T. N. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Relatório técnico, Secure Networks, Inc. Janeiro, 1998.

POPPI, S. Snort IDMEF Plugin. 2005. Disponível em: <http://sourceforge.net/projects/snort-idmef/>. Acesso ao site: junho 2005.

POPPI, S.; MIGUS, A.; MCALERNEY, J. LibIDMEF. Disponível em: <http://sourceforge.net/projects/libidmef/>. Acesso ao site: junho de 2005.

POSEY, B. M. Choosing an intrusion detection system: Network, host or application-based IDS. Searchwindowssecurity.com, 28 Abril 2005. Disponível em: <http://searchwindowssecurity.techtarget.com/tip/1,289483,sid45_gci1083969,00.html>

ROESCH, M. Snort - the de facto standard for intrusion detection/prevention. Versão 2.4. Disponível em: <http://www.snort.org>. Acesso em: 20 junho 2005.

ROSE, M. The Blocks Extensible Exchange Protocol Core. RFC3080, IETF, Março, 2001.

SANDHU, R. S. Lattice-based access control models. IEEE Computer, Estados Unidos, v. 26, n. 11, p. 9-19, 1993.

SEKAR, R.;GUPTA, A.; FRULLO, J.; SHANBHAG, T.; TIWARI, A.; YANG, H.; ZHOU, S. Specification-based Anomaly Detection: A New Approach for Detecting Network Intrusions. Proceedings of the 9th ACM Conference on Computer and Communications Security, p. 265-274, 2002.

SILVA, P. F. Extensão do modelo IDWG para detecção de intrusão em ambientes computacionais. Dissertação de mestrado, 2004.

SPI DYNAMICS, Inc. SQL Injection – Are Your Web Applications Vulnerable? 2002. Disponível em: <http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf>

STANIFORD-CHEN, S.; TUNG, B.; SCHNACKENBERG, D., The Common Intrusion Detection Framework (CIDF). Information Survivability Workshop, Outubro, 1998.

STEVENS, A. Guardian Active Response for Snort, 2004. Disponível em: <http://www.chaotic.org/guardian/index.html>.

US-CERT: United States Computer Emergency Readiness Team. Microsoft Windows domain controller denial of service in Kerberos message handling. Disponível em: <http://www.kb.cert.org/vuls/id/610133>. Acesso ao site: 16 Ago. 2005.

_____. Microsoft Windows Remote Desktop Protocol service input validation vulnerability. Disponível em:<http://www.kb.cert.org/vuls/id/490628>. Acesso ao site: 16 Ago. 2005.

WANG, Y.; ABDEL-WAHAB, H. A correlative context-based framework for network intrusion detection system. 10th IEEE Symposium on Computers and Communications (ISCC), p. 463-468, 2005.

WOOD, M.; ERLINGER, M. Intrusion Detection Message Exchange Requirements. Internet Draft, IETF, Outubro 2002. Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-idwg-requirements-10>.

WU, Y.; FOO, B.; MEI, Y.; BAGCHI, S. Collaborative intrusion detection system (CIDS): a framework for accurate and efficient IDS. 19th Annual Proceedings on Computer Security Applications Conference, December, 2003.

YANG, Y.; CHANG, J.; CHU, Y. The Security Components Exchange Protocol (SXCP). Internet Draft, Dezembro, 2003.

YASINSAC, A. An Environment for Security Protocol Intrusion Detection. Journal of Computer Security, v. 10, p. 177-188, 2002.

XU, D.; NING, P. Alert correlation through triggering events and common resources. 20th Annual Computer Security Applications Conference, p. 360-369, Dezembro, 2004.

ZHANG, X.; LI, C.; ZHENG, W. Intrusion Prevention System Design. 4th International Conference on Computer and Information Technology, p. 386-390, Setembro 2004

ZUCHLINSKI, G. The anatomy of Cross Site Scripting – Anatomy, Discovery, Attack, Exploitation. Novembro, 2003. Disponível em: <http://www.net-security.org/dl/articles/xss_anatomy.pdf>