Uma arquitetura de interação entre sistemas de detecção de intrusão utilizando a extensão Fault-Tolerant CORBA

Osmar Marchi dos Santos
Departamento de Informática – UNIFRA
97010-032 Santa Maria-RS
osmarmarchi@hotmail.com

Rafael Saldanha Campello
GSeg – Grupo de Segurança
Instituto de Informática – UFRGS
91501-970 Porto Alegre-RS
campello@inf.ufrgs.br

Resumo: *Este trabalho trata de um estudo sobre o uso de padrões abertos para sistemas distribuídos na tentativa de resolver alguns dos atuais problemas dos sistemas de detecção de intrusão (IDSs). O principal objetivo foi o de avaliar a utilização da especificação FT-CORBA (Fault-Tolerant CORBA) visando às necessidades de confiabilidade e disponibilidade dos atuais IDSs. Foram analisadas características como a aplicabilidade e a adaptação dos mecanismos propostos pelo FT-CORBA à área de detecção de intrusão.*

Palavras-chave: *sistemas de detecção de intrusão - segurança - tolerância a falhas - sistemas distribuídos.*

Abstract: *This work is about the use of open patterns for distributed systems in an attempt to solve some problems related to today's intrusion detection systems (IDSs). The main goal was to evaluate the utilization of the FT-CORBA (Fault-Tolerant CORBA) specification aiming at the disponibility and confiability of nowadays IDSs. Characteristics such as aplicability and adaptability of the proposed mechanisms inside the FT-CORBA were analyzed focusing on the needs of the intrusion detection area.*

Keywords: *intrusion detection systems - security - fault tolerance - distributed systems.*

1. Introdução

Como pode ser verificado em CERT (2001), a rede mundial de computadores vem apresentando grande aumento no número de ataques, i.e., tentativas de comprometer a integridade e a confiabilidade dos sistemas, alterando ou corrompendo dados. Pelo fato da Internet ser um negócio lucrativo para muitas empresas, o preço pago por um ataque bem sucedido é difícil de ser calculado, tanto pelo tempo gasto no processo de auditoria como na reconfiguração dos sistemas afetados. Estes são alguns dos fatores que cada vez mais impulsionam pesquisas na área de detecção de intrusão. Os sistemas de detecção de intrusão (*Intrusion Detection Systems* - IDSs) têm como principal objetivo identificar indivíduos que tentam utilizar um determinado sistema de forma não autorizada ou que desejam abusar de privilégios concedidos aos mesmos.

Apesar de toda pesquisa direcionada à área de detecção de intrusão, inúmeros são os problemas enfrentados pelos IDSs. Um bom exemplo dessas dificuldades é a crescente sofisticação dos ataques e das ferramentas utilizadas pelos atacantes, obrigando as ferramentas de detecção de intrusão a constantes atualizações e aumentando a sua complexidade. Outro exemplo dos problemas ainda enfrentados pelos atuais IDSs é a falta de padrões amplamente aceitos pela área, o que dificulta as interações entre sistemas existentes e a correlação dos resultados. Entretanto, um dos pontos mais preocupantes está relacionado à confiabilidade e disponibilidade dos próprios IDSs, o que os torna vulneráveis a vários tipos de ataque (ALLEN et al., 1999).

Seja devido a falhas causadas por agentes intencionais ou não, é possível aplicar conceitos da área de tolerância a falhas de modo a incrementar esses fatores, levando o IDS a um estado mais seguro e resistente a falhas. Existem, atualmente, inúmeros esforços de

pesquisa nesse sentido, demonstrando a importância do uso dessas técnicas em mecanismos de segurança. Como esforço de pesquisa em comum, todos esses projetos deparam-se com a necessidade de recriar toda uma nova base de comunicação voltada para o emprego de técnicas de tolerância a falhas em sistemas distribuídos e adaptada às necessidades de segurança que o contexto impõe.

Poucos trabalhos, no entanto, têm optado pela utilização de padrões de comunicação já existentes, evidenciando futuros problemas de interconexão e visíveis dificuldades de desenvolvimento. Ao empregar como base de comunicação padrões abertos e já difundidos, como é o caso de CORBA, o processo de criação de qualquer sistema distribuído torna-se mais rápido, deixando para o desenvolvedor apenas as preocupações inerentes à aplicação, o que é perfeitamente válido para a aplicação de técnicas de tolerância a falhas em sistemas de detecção de intrusão. Apesar dessas vantagens, características inerentes a essa classe de aplicações, onde questões de segurança e, principalmente, de desempenho são extremamente relevantes, podem impor resistência ao uso de padrões abertos de comunicação nesses casos.

O presente artigo apresenta os resultados de um estudo realizado na área de sistemas distribuídos e tolerância a falhas aplicados à interligação de sistemas de detecção de intrusão. O principal objetivo desse estudo foi avaliar a utilização da especificação FT-CORBA (*Fault-Tolerant* CORBA), que apresenta mecanismos incorporados de tolerância a falhas para sistemas distribuídos, visando às necessidades de confiabilidade e disponibilidade dos atuais IDSs. Foram analisadas características como a aplicabilidade e a adaptação dos mecanismos propostos pelo FT-CORBA à área de detecção de intrusão. Com base nisto foi implementada uma arquitetura de interação entre IDSs utilizando o sistema de detecção de intrusão Snort e a extensão FT-CORBA.

2. Trabalhos Relacionados

Vários trabalhos vêm tentando criar uma base de comunicação mais confiável e disponível através da implementação de conceitos da área de tolerância a falhas aplicada a sistemas de detecção de intrusão e a outros *softwares* de segurança em geral. Dentre estes, destacam-se os projetos MAFTIA (*Malicious-and Accidental-Fault Tolerance for Internet Applications*) e Asgaard. O primeiro se trata de uma plataforma ao nível de *middleware* destinada a aplicações de segurança em geral, enquanto o último consiste em um sistema de detecção de intrusão completamente distribuído.

O projeto MAFTIA (MAFTIA, 2001) tem como objetivo criar uma arquitetura aberta na forma de *middleware* tolerante a falhas, tratando tanto de falhas acidentais quanto maliciosas. Por ser um trabalho direcionado à área de segurança computacional, especialmente para transações operacionais da Internet, ele apresenta um modelo híbrido de falhas (FIGURA 1) onde são identificadas, no mínimo, três classes de falhas relevantes: vulnerabilidades, ataques e intrusões (VERÍSSIMO, 2000).

Através deste modelo são retirados os princípios básicos do MAFTIA, ou seja, o uso recursivo de prevenção de falhas (prevenção de ataques, vulnerabilidade e intrusão) e, caso não seja o suficiente para deter algum tipo de defeito no sistema, a utilização de tolerância a falhas na forma de detecção (atividades do intruso), recuperação (interceptação e recuperação) e mascaramento (votação entre vários componentes). Logo, é possível ter cada componente do sistema apresentando um comportamento controlado (devido à prevenção de falhas) em face de falhas maliciosas.

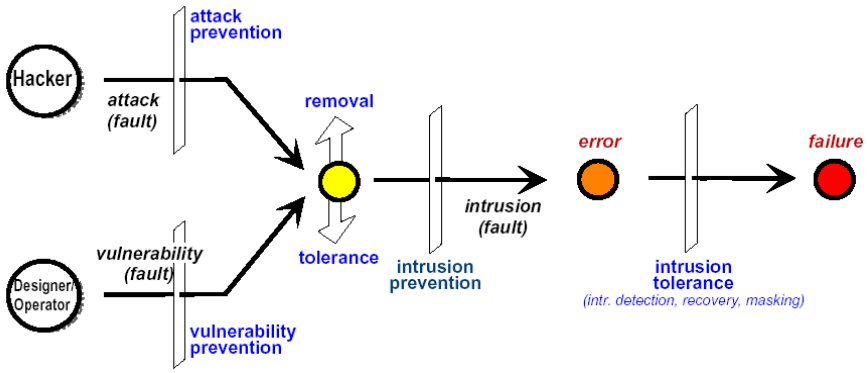

Figura 1 - Modelo de falhas do projeto MAFTIA (Fonte: VERÍSSIMO, 2000).

O sistema de detecção de intrusão Asgaard tem como objetivo principal criar um sistema completamente distribuído que consiga resolver os seguintes problemas (CAMPELLO et al, 2001):

- Nível de ataques: os programas utilizados em ataques estão se tornando cada dia mais complexos e, conseqüentemente, mais eficientes na descoberta de novas vulnerabilidades;
- Dimensão e complexidade das redes: o rápido crescimento na complexidade e dimensão das redes torna os IDSs menos eficazes, principalmente em redes de larga escala;
- Tolerância a falhas: garantir a confiabilidade e disponibilidade do sistema de detecção de intrusão empregando técnicas de tolerância a falhas é vital para um correto funcionamento;
- Autenticação e privacidade: devem existir mecanismos que provenham autenticação ao sistema e dêem garantias quanto à privacidade das mensagens trocadas;
- Interoperabilidade e padronização: a falta de padrões acarreta a falta de interação entre diferentes ferramentas;
- Arquiteturas distribuídas: a complexidade dos sistemas em geral clama por mecanismos de segurança que trabalhem em conjunto a fim de tornar mais seguro o sistema como um todo.

Com bases nestes problemas o projeto apresenta a proposta de criar um IDS modular, portável e com características de tolerância a falhas. Mais do que um IDS, o objetivo é criar uma plataforma que possibilite, com o passar do tempo, agregar e testar novos módulos e técnicas (CAMPELLO et al, 2001). A estrutura dos módulos Asgaard é dividida em camadas (FIGURA 2). Em relação à camada de tolerância a falhas dos módulos, esta deve conter detectores de defeito ou comunicação de grupo para que módulos falhos sejam detectados em tempo, evitando a degradação do sistema ou mesmo ataques que queiram corromper o serviço. Além disto, esta camada deve atender às solicitações de novos módulos que desejam participar do sistema e manter uma visão geral dos módulos que no momento compõem o sistema.

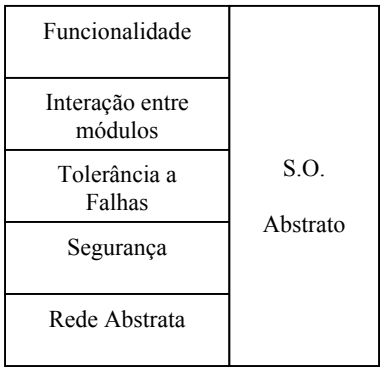

Figura 2 - Camadas de um módulo Asgaard (Fonte: CAMPELLO et al, 2001).

3. O Projeto

Com o objetivo de validar os estudos propostos e, além disso, de tornar esta implementação útil e mais próxima da realidade, optou-se por desenvolver através da infra-estrutura CORBA um *wrapper* (FIGURA 3) para o sistema de detecção de intrusão Snort (SNORT, 2001). O Snort é um IDS centralizado, baseado em rede e que utiliza uma base de assinaturas de ataques para detectar a ocorrência de intrusões. Em uma organização de grande porte, onde há necessidade de se ter vários sensores espalhados pela rede, IDSs desse tipo apresentam sérias restrições, tais como a verificação da disponibilidade dos mesmos e a correlação de alertas.

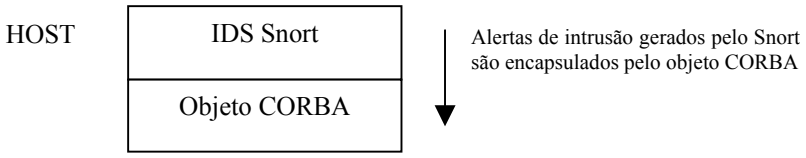

Figura 3 - *Wrapper* para o sistema de detecção de intrusão Snort.

Mesmo sendo um dos IDSs de maior aceitação e utilização da atualidade, o Snort possui as limitações apresentadas e, logo, o *wrapper* criado tem de executar uma função chave que é a de oferecer uma interface através do padrão CORBA pela qual possam ser obtidos os alertas gerados pela instância Snort, encapsulados pelo objeto CORBA do *wrapper*. Estes alertas são obtidos por uma interface com o usuário que consulta periodicamente os *wrappers* e reúne, em um ponto único, todos os alertas gerados pelos mesmos. Uma outra vantagem de encapsular o IDS Snort é a facilidade de distribuição das assinaturas de ataque. Embora esta característica não esteja no escopo do trabalho, ela pode ser facilmente implementada no sistema desenvolvido.

O sistema FT-CORBA é responsável pelo monitoramento do *wrapper* e suas réplicas (FIGURA 4). Na presença de falhas ele tem como obrigação disparar transparentemente uma das réplicas que fazem parte de seu domínio de monitoramento, dando continuidade ao serviço. Toda a comunicação realizada pelo *wrapper* utiliza o padrão CORBA. Assim, além de avaliar o uso do padrão em aplicações de segurança, a opção por CORBA pode facilitar uma possível integração entre esta e outras implementações com o mesmo propósito.

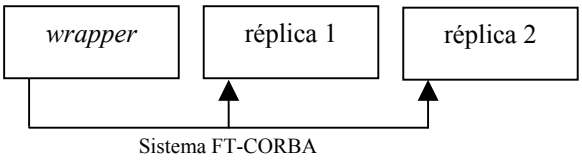

Figura 4 - Monitoração do *wrapper* e suas réplicas pelo sistema FT-CORBA.

Nesse último aspecto, relativo à integração de diferentes sistemas de detecção de intrusão, é positivo ressaltar que o padrão CORBA pode ser comparado, pela forma como está sendo utilizado neste projeto, ao protocolo IDXP (*Intrusion Detection Exchange Protocol*) (IDXP, 2001) criado pelo IDWG (*Intrusion Detection Working Group*) (IDWG, 2001).

O IDXP é um protocolo baseado na existência de clientes e servidores, apresentando características como a autenticação mútua dos participantes da comunicação e a confidencialidade do conteúdo das mensagens, através de criptografia. Ele tem por objetivo ser um canal para a troca de informações entre entidades de detecção de intrusão. A troca de dados entre as entidades utiliza o padrão para troca de mensagens IDMEF (IDMEF, 2001) mas, é possível que dados binários e texto não estruturado possam ser enviados através do canal de comunicação criado pelo IDXP.

No protocolo são definidos analisadores e gerenciadores. Não podem ser criados canais de comunicação entre os analisadores, mas as outras formas de comunicações são possíveis (e.g. analisador -> gerenciador e gerenciador -> gerenciador). Esta arquitetura se assemelha à utilizada neste projeto, sendo que os analisadores condizem com os *wrappers* e o gerenciador com a interface com o usuário.

Outra característica do IDXP é a utilização do modelo *heartbeat*, como definido no padrão IDMEF, para a detecção dos analisadores defeituosos do sistema. Neste modelo os analisadores periodicamente enviam mensagens ao gerenciador que ao analisar a perda do número de mensagens é capaz de verificar quais analisadores estão funcionando corretamente ou não.

Diferentemente do IDXP, a arquitetura proposta neste projeto utiliza o modelo *pull* para monitoração dos *wrappers*. Nesta implementação, a interface com o usuário faz requisições periódicas ao *wrapper* que responde com a devida informação (o número de alertas ou o alerta em si) ou gera um erro por tempo, significando que o objeto está inativo. Outro modelo que poderia ser utilizado é o *push*, em que os *wrappers* assumem o trabalho de contatar a interface com o usuário.

Um problema verificado na arquitetura proposta para os *wrappers* é que somente a interface com o usuário consegue contatar os mesmos. Uma vez que os alertas são a prioridade número um do sistema, seria interessante que o próprio *wrapper* passasse os alertas para a interface com o usuário, assim que esses fossem detectados. Isso poderia ser feito, mas aumentaria a complexidade do sistema com a criação de uma interface adicional. Apesar disto, mesmo que a eficácia do sistema compense tal complexidade, o porte e as exigências computacionais da estrutura FT-CORBA acabaram por determinar a não implementação de tal arquitetura antes que pudesse ser avaliada uma solução mais simples que fornecesse informações relativas à aplicabilidade desse padrão na área de detecção de intrusão.

4. A Implementação

Para o desenvolvimento do sistema foi escolhida a distribuição GroupPac (GROUPPAC, 2001) desenvolvida na UFSC (Universidade Federal de Santa Catarina). O GroupPac é uma implementação baseada nas especificações FT-CORBA, com algumas

modificações para atender requisitos de tolerância a falhas em sistemas de larga escala. Toda a arquitetura de interação, desde a implementação do *wrapper* até a interface com o usuário, foi criada com a linguagem de programação Java.

O primeiro passo na criação do sistema foi definir a interface do *wrapper*, utilizando a linguagem IDL (FIGURA 5). Nela são definidos dois métodos principais:

- *long has_detection()*: o objetivo é indicar o número total de alertas que o Snort monitorado pelo objeto CORBA gerou até o momento;
- *string get_detection(in long id)*: este método retorna o alerta em si, que é identificado pelo valor inteiro passado como argumento.

Além disso, é herdada uma interface chamada *PullMonitorable* que faz parte das especificações FT-CORBA. É através desta interface que o objeto CORBA consegue ser monitorado contra falhas.

```
module CORBA {
      module FT {
            interface Wrapper: PullMonitorable {
                  long has_detection();
                  string get_detection(in long id);
            };
      };
};
```

Figura 5 - Interface do *wrapper*.

O *wrapper* é criado como um objeto CORBA que, ao ser rodado, executa uma instância do IDS Snort. À medida que novos alertas são gerados pelo IDS, estes são encapsulados pelo objeto CORBA e se tornam disponíveis através da interface anteriormente definida.

A interface com o usuário é responsável pela visualização dos *wrappers* que estão ativos e inativos e por reunir em um mesmo local os alertas contidos nos diferentes *wrappers*. Ela é composta por vários clientes CORBA (cada um deles é responsável por monitorar um *wrapper*) que, periodicamente, chamam o método *has_detection()* do *wrapper* que estão monitorando. Caso seja evidenciado um novo alerta, esse mesmo cliente CORBA executa o método *get_detection()* com a intenção de obter o(s) alerta(s) e gravá-los em um arquivo de *log*, que é disponibilizado ao administrador da rede.

Toda comunicação utilizada neste trabalho usa o padrão CORBA sem nenhum tipo de segurança para o canal de comunicação. Isto cria sérios problemas relativos a segurança da arquitetura proposta e implementada. Em primeiro lugar, não é provido um meio criptográfico pelo qual os dados transitam, deixando as informações que trafegam entre a interface com o usuário e os *wrappers* bastante vulneráveis. Além disto, não são utilizados mecanismos de autenticação entre a interface e os *wrappers*, o que torna o sistema completamente vulnerável a ataques de personificação, onde um *wrapper* malicioso seja inserido no sistema e assuma as funções de um outro legítimo. Outra forma de explorar a falta de autenticação do sistema é substituir o código Snort a ser executado por um código alterado, inoperante e voltado a ocultar as ações de um dado atacante.

Com relação à extensão FT-CORBA, esta não dá garantias de que as réplicas a serem executadas em caso de falha de algum *wrapper* monitorado estão adulteradas ou não. Caso um atacante obtenha acesso ao sistema de arquivos onde a réplica se encontra, a mesma pode ser modificada e até mesmo substituída, sendo que o sistema FT-CORBA a executará sem maiores problemas. Ferramentas para verificação de integridade de arquivos estão atualmente disponíveis (TRIPWIRE, 2001; SERAFIM & WEBER, 2001) e poderiam ser usadas para minimizar esse tipo de problema.

Outra forma de garantir maiores características de segurança ao sistema é aplicar o modelo CORBA de segurança (CORBAsec) (CORBAsec, 2001), que prevê aspectos de segurança como autenticação, segurança na comunicação entre os objetos e autorização e controle de acesso. O CORBAsec não foi utilizado neste trabalho porque não havia, no momento da criação do projeto, uma implementação CORBA que apresentasse ambas as extensões FT-CORBA e CORBAsec em conjunto.

5. Considerações Finais

Nota-se que, mesmo sistemas voltados à área de segurança apresentam sérios riscos ao não proverem mecanismos de proteção à própria aplicação. Estes mecanismos tolerantes a falhas são inseridos no sistema com o objetivo de deixar a aplicação em um estado mais seguro e resistente a falhas. Por CORBA ser um padrão aberto e amplamente difundido, sua utilização na resolução de alguns dos atuais problemas encontrados nos IDSs seria bastante proveitosa, pois maior velocidade seria dada ao processo de criação da base de comunicação, assim como um maior grau de dedicação poderia ser direcionado a pesquisas na própria área de detecção de intrusão.

Com a implementação da arquitetura de interação entre IDSs proposta, tornaram-se visíveis algumas das vantagens e desvantagens da extensão FT-CORBA para a área de detecção de intrusão. Dentre as vantagens observadas, podem ser destacadas a facilidade de criação do sistema e as garantias da extensão FT-CORBA de que as falhas serão tratadas transparentemente para a aplicação, caso exista alguma réplica disponível.

A alta complexidade da infra-estrutura FT-CORBA composta pelo gerenciador de replicação, detector de falhas e fábrica genérica, acaba por não compensar o esforço computacional necessário para sua utilização na área de detecção de intrusão. Sistemas de detecção de intrusão são aplicações peculiares que despendem uma grande quantidade de processamento, o que inviabiliza sua coexistência com toda uma complexa infra-estrutura. Os mecanismos empregados pelo sistema FT-CORBA tentam generalizar a utilização do padrão para uma grande variedade de áreas, implicando em mecanismos de tolerância a falhas demasiadamente complexos para a área de detecção de intrusão. É importante ressaltar que as constatações a respeito da complexidade da infra-estrutura FT-CORBA resultam da experiência prática com o sistema, mas nenhum tipo de teste ou avaliação de desempenho foi realizado sobre o sistema. Acredita-se que a simplificação de alguns mecanismos de tolerância a falhas existentes no sistema FT-CORBA seria a melhor forma de garantir a confiabilidade e disponibilidade de um IDS sem causar um impacto excessivo no seu desempenho.

Nesse sentido, os próximos passos deste trabalho enfocariam um estudo mais aprofundado dos mecanismos de tolerância a falhas existentes e o desenvolvimento de soluções leves e direcionadas aos sistemas de detecção de intrusão. Quanto à arquitetura apresentada, muitos esforços poderiam ser considerados na tentativa de aperfeiçoá-la e torná-la operacionalmente viável. Um dos primeiros trabalhos nesse sentido, que está em desenvolvimento, é o aperfeiçoamento da interface com o usuário, tornando-a visualmente

mais atraente e interligando-a com um banco de dados para facilitar o armazenamento dos alertas e a correlação de resultados. Outra tarefa importante se trata da implementação do modelo *push*, aumentando o leque de opções da ferramenta e diminuindo, em alguns casos, o tráfego gerado. Incorporar mecanismos de autenticação entre a interface e os *wrappers* participantes é outro ponto fundamental, assim como garantir a cifragem do canal de comunicação.

6. Referências Bibliográficas

ALLEN, Julia et al. 1999. **State of the Practice of Intrusion Detection Technology**. Software Engineering Institute: Carnegie Mellon University.

CAMPELLO, Rafael Saldanha et al. 2001. **O Sistema de Detecção de Intrusão Asgaard**. Anais do I Workshop em Segurança de Sistemas de Computação. Florianópolis: UFSC-DAS.

CERT: EMERGENCE RESPONSE TEAM . 2001. **Official Home Page**. Encontrado em http://www.cert.org. 10/11/2001.

CORBAsec. 2001. **Security Service**. Encontrado em http://www.omg.org/technology/documents/formal/security_service.htm. 10/11/2001.

IDMEF: Intrusion Detection Message Exchange Format. 2001. **Intrusion Detection Message Exchange Format Extensible Markup Language (XML) Document Type Definition**. Encontrado em http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-04.txt. 10/11/2001.

GROUPPAC. 2001. **Página Oficial**. Encontrado em http://grouppac.sourceforge.net/grouppac/br/. 18/04/2002.

IDWG: Intrusion Detection Working Group. 2001. **Intrusion Detection Exchange Format (idwg) Charter**. Encontrado em http://www.ietf.org/html.charters/idwg-charter.html. 10/11/2001.

IDXP: Intrusion Detection Exchange Protocol. 2001. **The Intrusion Detection Exchange Protocol (IDXP)**.Encontrado em http://www.ietf.org/internet-drafts/draft-ietf-idwg-beep-idxp-03.txt. 10/11/2001.

MAFTIA: Malicious-and Accidental-Fault Tolerance for Intenet Applications. 2001. **Official Home Page**. Encontrado em http://www.maftia.org. 10/11/2001.

SERAFIM, Vinícius da Silveira & WEBER, Raul Fernando. 2001. **Um verificador seguro de integridade de arquivos**. 3º Simpósio Segurança em Informática. Campos do Jordão: ITA.

SNORT. 2001. **Official Home Page**. Encontrado em http://www.snort.org. 10/11/2001.

TRIPWIRE**.** 2001. **Tripwire, Inc. – Data and Network Integrity Software**. Encontrado em http://www.tripwiresecurity.com. 10/11/2001.

VERÍSSIMO, Paulo et al. 2000. **The Middleware Architecture of MAFTIA: A Blueprint**. Third Information Survivability Workshop. Boston, Massachusetts.