UNIVERSIDADE DE CAXIAS DO SUL
Centro de Computação e Tecnologia da Informação
Curso de Bacharelado em Ciência da Computação ou
Sistemas de Informação

Vinícius Borghetti Kerwald

SISTEMA DE DETECÇÃO DE INTRUSÃO DISTRIBUÍDO COM MONITORAMENTO MULTIAGENTES

Caxias do Sul

2010

Vinícius Borghetti Kerwald

SISTEMA DE DETECÇÃO DE INTRUSÃO DISTRIBUÍDO
COM MONITORAMENTO MULTIAGENTES

Trabalho de Conclusão de Curso para obtenção do Grau de Bacharel em Ciência da Computação ou Sistemas de Informação da Universidade de Caxias do Sul.

João Luís Tavares da Silva
Orientador

Caxias do Sul

2010

Dedico este trabalho especialmente à minha família e também às pessoas que de uma forma ou de outra contribuíram para que eu conseguisse chegar até aqui.

AGRADECIMENTOS

Agradeço aos meus pais, Sônia e Carlos, incansáveis em fazer tudo pela felicidade de minha família.

Um agradecimento especial aos meus amigos por entenderem a minha ausência nos períodos em que estive empenhado no desenvolvimento deste trabalho e a muitos por me darem forças para que conseguisse concluí-lo.

Ao meu orientador João Luís Tavares da Silva, ao coorientador Alex Pellin e ao professor Vânius Gava por contribuírem com seus conhecimentos que foram imprescindíveis para a conclusão deste trabalho.

À todas as pessoas do meu convívio que acreditaram em mim e contribuíram de alguma forma para a conclusão deste curso.

RESUMO

Neste trabalho é apresentado o conceito de detecção colaborativa em uma arquitetura para um Sistema de Detecção de Intrusão que utiliza do conceito de sistemas multiagentes. O objetivo deste recurso é fazer com que o SDI possua uma base de conhecimento das tentativas de intrusões já tratadas para que o sistema não cometa redundâncias nas tratativas dos ataques ao tratar uma intrusão já ocorrida. Se o sistema identificar novamente a mesma intrusão, é muito provável que haja falhas em sua tratativa pelo SDI. Através disso, é possível identificar e corrigir pontos falhos do sistema para torná-lo ainda melhor.

Para a elaboração deste protocolo, é apresentado de forma resumida os tipos de SDI existentes, bem como o funcionamento de três SDI implementados. Além disso, é explicado a arquitetura do SDI baseado em sistemas multiagentes no qual a detecção colaborativa será implementada. Após, introduzido os benefícios que a colaboração acrescenta ao SDI, uma revisão da atual arquitetura do SDI, o protocolo de colaboração desenvolvido, o ambiente utilizado nos testes e os cenários dos resultados obtidos.

Palavras-chaves: Detecção Colaborativa, Base de Conhecimento, Protocolo de Colaboração, Redundância

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

Sigla	Significado em Português	Significado em Inglês
DF	Facilitador de Diretório	Directory Facilitador
IA-DIDS	Agentes Inteligentes para Sistema de Detecção de Intrusão Distribuído	Intelligent Agents for Distributed Intrusion
IDA	Sistema de Agentes para Detecção de Intrusão	Intrusion Detection Agent System
MADIF	*Framework* Distribuído para Detecção de Intrusão usando Agentes Móveis	Distributed Intrusion Detection Framework by Using Mobile Agents
SDI	Sistema de Detecção de Intrusão	
SDID	Sistema de Detecção de Intrusão Distribuído	
SMA	Sistemas Multiagentes	

1 INTRODUÇÃO

Devido à popularização da Internet, tanto para uso pessoal quanto profissional, o volume de informações confidenciais armazenadas nos computadores teve um aumento significativo. Devido a esse fato, existe um relevante investimento em segurança destas informações, principalmente por empresas e órgãos governamentais.

A tecnologia atual proporciona diversas formas de prover segurança em nível computacional. Estes meios vão desde a prevenção até a detecção de invasões. Pode-se dizer que atualmente a prevenção é a principal forma utilizada.

Um modo importante de proteger as informações digitais é a detecção de quando estas estão sendo acessadas de forma indesejada. Para tornar isso possível, foram criados os chamados Sistemas de Detecção de Intrusão (SDI), que tem a função de monitorar os sistemas computacionais que emitem alertas quando eventos de interesse sejam detectados (BACE, 00).

O monitoramento de uma rede de computadores consiste em analisar os pacotes de rede que trafegam em um segmento de rede ou por um *switch,* com o objetivo de detectar sinais de intrusão em qualquer dispositivo que está localizado nesta rede ou segmento.

Os SDI, inicialmente criados de forma centralizada, possuíam um ponto central de gargalo que facilitava um ataque. Posteriormente, propostas distribuídas dividiam as funções do sistema em partes para que cada uma monitore eventos específicos. Porém, esta proposta também apresenta pontos falhos, já que da forma como foram projetados, possuem um ponto central, que é a interface de dados monitorados pelo SDI, e novamente, este poderá ser um ponto de ataque (RIBEIRO, 02).

Atualmente, formas alternativas para contornar esse problema utilizam SDI baseados em sistemas multiagentes (SMA), que possuem uma arquitetura distribuída, não existindo um ponto central vulnerável dentro do sistema. Um agente é considerado um processo autônomo, distribuído e inteligente. A combinação e cooperação destes formam um SMA, que tem por função resolver um problema complexo que está além da capacidade de resolução de um único agente (GIRARDI, 02). Essa arquitetura é uma forma dos agentes se comunicarem e tomarem decisões em conjunto, para que consigam chegar ao sucesso na detecção e no alerta dentro do esperado (GORODETSKI, 03).

Uma grande vantagem proveniente da utilização deste tipo de monitoramento é a possibilidade de acompanhar eventos que ocorrem em uma grande rede de computadores, devido a facilidade de distribuir os agentes monitores na rede. Outra vantagem que pode ser citada é que, do ponto de vista do atacante, o monitoramento da rede pode ser invisível, já que a distribuição dos monitores em diversos pontos da rede torna maior a dificuldade de identificar que um possível monitoramento esteja sendo realizado nesta rede.

Em arquiteturas distribuídas de SDI os módulos trocam informações através de mensagens cooperando para que todo o sistema funcione corretamente e as possibilidades de falhas sejam minimizadas, já que pelo próprio formato da arquitetura distribuída, já existe uma tolerância a falhas dos módulos do sistema.

Neste trabalho, trataremos de uma extensão ao trabalho de Júnior (JUNIOR, 09) que propôs uma arquitetura multiagentes para um SDI distribuído (SDID) que apresenta um funcionamento pró-ativo. Além de detectar uma intrusão, o sistema realiza ações para que a intrusão não ocorra por completo ou cause o menor impacto possível. O trabalho de Júnior se concentrou na arquitetura e no modelo de interoperabilidade de componentes de monitoramento e detecção de intrusão para que os dados gerados pelo sistema tenham uma maior confiabilidade na geração de um número reduzido de falsos positivos e negativos. A validação deste trabalho deu-se através de um agente de monitoramento analisando *logs* gerados pelo detector SNORT (SNORT, 08).

1.1 Objetivos

A extensão proposta no presente trabalho refere-se à capacidade de cooperação dos agentes para uma maior confiabilidade do sistema de detecção. Uma única fonte de monitoramento mostra-se incapaz de analisar ataques reais, porém a arquitetura multiagentes visa também facilitar a extensão do SDI.

Com base nisso, propomos a extensão do comportamento do agente de monitoramento e análise para interagir com os agentes já existentes no SDI. Este agente deve ser capaz de analisar uma possível intrusão e decidir se houve realmente uma tentativa de intrusão. Caso uma intrusão seja detectada, através de um protocolo de cooperação, este agente será capaz de cooperar e comunicar-se com os outros agentes do sistema para que o sistema tome as providências cabíveis para impedir a intrusão.

Além disso, será necessário desenvolver um protocolo de cooperação baseado na ação destes agentes. Este protocolo consiste em um padrão de comunicação entre os agentes desse sistema, de modo que um agente interpreta o que outro quer lhe dizer através de regras pré-estabelecidas de análise de detecção.

Em resumo, este trabalho trata da implementação de capacidades de colaboração e análise do agente de monitoramento e análise do SDID proposto em (JUNIOR, 09), um estudo para replicação destes agentes na arquitetura existente e na definição e implementação de um protocolo de cooperação baseado na análise dos dados monitorados.

1.2 Metodologia

Para a elaboração desta proposta, primeiramente será estudado e detalhado o funcionamento e as peculiaridades da arquitetura multiagentes SDID de (JUNIOR, 09). Após, descreveremos alguns Sistemas de Detecção de Intrusão alternativos que também usam agentes para contextualização do problema da tolerância à falhas (SIQUEIRA, 06). Em seguida, estudaremos sistemas de detecção adicionais ao SNORT para poder ser integrado ao sistema e as formas de análise colaborativa na detecção de tentativas de ataque à rede. Para isto, um protocolo de cooperação será definido para estabelecer um padrão de troca de conhecimento entre os agentes. Finalmente, será definido o agente e os seus componentes.

No Capítulo 2 apresentamos o conceito de sistemas detecção de intrusão, bem como sua arquitetura, estratégias de monitoramento e disposições. Além disso, é dado ênfase aos sistemas de detecção distribuídos, apresentando os principais trabalhos relacionados e considerações.

No Capítulo 3 é detalhado o trabalho de JUNIOR (JUNIOR, 09), que desenvolveu um SDI distribuído baseado em sistemas multiagentes. É apresentado o funcionamento do sistema, sua arquitetura, os agentes e seus papéis no sistema, além das tecnologias necessárias e utilizadas.

No Capítulo 4 é apresentado o conceito de um SDID colaborativo, recurso acrescentado ao trabalho de Junior (JUNIOR, 09). Nele é apresentado o novo modelo de agentes e o protocolo de colaboração, a nova arquitetura, além dos cenários de colaboração e a análise dos experimentos.

As conclusões e perspectivas de trabalhos futuros são apresentados no Capítulo 5.

2 SISTEMAS DE DETECÇÃO DE INTRUSÃO

A detecção da intrusão é baseada na suposição de que o comportamento do intruso difere daquele de um usuário legítimo de maneira que podem ser quantificadas. Naturalmente, não podemos esperar que haverá uma distinção clara, exata, entre um ataque de um intruso e o uso normal dos recursos por um usuário autorizado. Em vez disso, devemos esperar que haja uma sobreposição (STALLINGS, 08).

Para se conseguir detectar uma intrusão em uma rede de computadores ou em um sistema de computador, deve ser feita uma análise, de forma manual ou automatizada, nos fatos ocorridos nestes meios.

Os Sistemas de Detecção de Intrusão (SDI) tem como função o monitoramento e análise de eventos, de forma automática, que venham a ocorrer com o objetivo de informar o acontecimento de uma intrusão ou agir de forma pró-ativa para que a intrusão não tenha sucesso para o invasor (BACE, 00).

Segundo Stallings (STALLINGS, 08), esse interesse por esse tipo de sistema é motivado por uma série de considerações, incluindo as seguintes:

- se uma intrusão for detectada com rapidez suficiente, o intruso poderá ser identificado e expulso do sistema antes que seja feito qualquer dano ou qualquer dado seja comprometido. Mesmo que a detecção não seja suficientemente oportuna para impedir o intruso, quanto mais cedo a intrusão for detectada, menor serão os danos e mais rapidamente a recuperação poderá ser obtida;
- um sistema de detecção de intrusão eficaz pode servir como um elemento desencorajador, atuando para impedir intrusões;
- a detecção de intrusão permite a coleta de informações sobre as técnicas de intrusão, o que pode ser usado para fortalecer a estrutura de prevenção de intrusão.

2.1 Arquitetura

Bace (BACE, 00) define que, de forma geral, um SDI é composto por três componentes básicos:

- uma fonte de informações de eventos ocorridos dentro do meio de atuação do SDI;

- um componente que busca por sinais de intrusão nestes eventos;
- um componente que tem como responsabilidade de reagir quando uma intrusão é detectada pelo componente anterior.

A arquitetura de um SDI precisa ser segura, pois ele pode ser o alvo de um ataque. O invasor que explorar uma possível vulnerabilidade do SDI tem uma enorme vantagem, pois nenhum sistema de detecção estaria atuando sobre o seu ataque. Uma forma de proteção seria separar a informação que será analisada do sistema alvo do monitoramento, pois se essa separação não existir, o invasor altera estes dados, tornando o ataque imperceptível.

2.2 Estratégias de Monitoramento

A atividade de monitoramento de um SDI depende do alvo monitorado pelo sistema que, segundo Bace (BACE, 2001), podem ser de três tipos: monitoramento em redes, monitoramento de computadores e monitoramento baseado em aplicações.

O monitoramento em redes de computadores analisa os pacotes da rede que trafegam em um segmento de rede. Também pode ser feito através de um *switch,* que detecta sinais de uma intrusão em algum dispositivo da rede ou de seus segmentos. É realizado distribuindo-se, nos diversos computadores da rede, agentes que analisam o tráfego de rede que passam por eles e tentam identificar pacotes de rede que contenham dados que identifiquem que uma invasão está acontecendo.

Devido a facilidade de distribuir os agentes monitores pela rede, torna-se vantajoso utilizar este tipo de monitoramento em grandes redes de computadores. Do ponto de vista do atacante, o monitoramento da rede torna-se imperceptível, uma vez que a distribuição dos agentes monitores em diversos pontos da rede aumenta a dificuldade em identificar um monitoramento nesta rede.

Por outro lado, em momentos que a rede esteja com tráfego intenso, pode não ser possível realizar a análise de todos os pacotes. Com isso, pode ocorrer que justamente dos pacotes que identificam a tentativa de intrusão não sejam monitorados. Outra desvantagem é a impossibilidade de um agente analisar pacotes que tenham dados criptografados. Atualmente, devido à necessidade de segurança dos dados que trafegam nas redes, este tipo de pacote é comum de ser encontrado.

Outro tipo de estratégia, segundo Bace (BACE, 2001), é o monitoramento de computadores. Este tipo de monitoramento analisa eventos que ocorrem em um ou mais

sistemas de um computador. Na maioria das vezes, dados do sistema operacional são analisados, como por exemplo, *logs* do sistema de informações disponibilizados pelo *kernel*.

Os sistemas baseados neste tipo de monitoramento estão mais vulneráveis a ataques como, por exemplo, ataques de negação de serviço. Uma maior facilidade é gerada ao intruso por possuírem um ponto único de monitoramento. Outra desvantagem é a grande quantidade de informação gerada por um sistema operacional, pois em computadores que geram *logs* frequentemente, essas informações podem ser difíceis de serem analisadas, dificultando a identificação de possíveis sinais de ataque. Além disso, deve ser levado em consideração a diminuição de desempenho do computador monitorado, devido ao fato do monitoramento.

A utilização deste tipo de monitoramento tem suas vantagens. Ele possibilita a detecção ataques que não conseguem ser detectados através de uma rede de computadores, quando outros sistemas que rodam no computador possuem brechas que são exploradas por intrusos e dados de informação destes sistemas são gerados pelo sistema operacional da máquina.

O terceiro tipo de estratégia é um SDI baseado em aplicações. Normalmente, esse tipo de sistema monitora dados disponibilizados por aplicações que, comumente, consistem em *logs* transações realizadas através da aplicação. Seu intuito é localizar e identificar usuários que excedem suas permissões de acesso aos dados do sistema ou ainda acesso não autorizado a dados do sistema.

Uma vez que este tipo acontece através do monitoramento da interação entre o usuário e a aplicação, uma grande vantagem que se pode obter é a identificação de acessos não autorizados.

Em contrapartida, uma desvantagem é que os arquivos de *log,* que contém as informações que são monitoradas pelo SDI, estão desprotegidos pelo sistema operacional. Esta falta de proteção compromete a eficácia do sistema, pois possibilita que estes dados sejam apagados ou adulterados.

2.3 Disposição de Sistema de Monitoramento

Os componentes de um SDI podem estar dispostos, segundo Bace (BACE, 2001), de três formas: centralizados, distribuídos ou organizados de forma hierárquica. Cada uma destas formas apresenta vantagens e desvantagens, que são explicadas a seguir.

2.3.1 SDI com Monitoramento Centralizado

Neste tipo de arquitetura, os dados monitorados pelos sensores são enviados para o console do SDI, cuja função é identificar as intrusões e tomar decisões sobre os alarmes que devem ser gerados no momento em que uma intrusão for identificada.

Este tipo de sistema é de fácil instalação e configuração. Em comparação a um sistema distribuído que necessita de um maior tempo de planejamento, do ponto de vista do desenvolvimento este sistema é mais simples. Entretanto, este tipo de SDI se torna ineficiente quando diversidade e tamanho das arquiteturas de redes existentes no ambiente é grande.

2.3.2 SDI com Monitoramento Distribuído

Nesta abordagem, os módulos comunicam-se através de mensagens, cooperando para que todo o sistema funcione corretamente e sejam minimizadas as possibilidades de falhas.

Nesta arquitetura, os dados não são enviados para um ponto central. Os sensores de monitoramento estão distribuídos em diversos pontos da rede obtendo dados, analisando possíveis intrusões e se necessário, reportam em um determinado formato (protocolo) o acontecimento de uma intrusão.

Um ponto negativo que contrasta com uma implementação centralizada é a necessidade de implementação elevada para garantir a segurança da troca de informações entre os módulos.

2.3.3 SDI com Monitoramento Híbrido

Em um SDI que possui este tipo de monitoramento, existe uma hierarquia onde os sensores se reportam ao uma console específica e estas reportam os dados recebidos dos diversos sensores a uma console principal.

Esta arquitetura possui o mesmo problema existente na arquitetura centralizada: existem pontos que se estiverem vulneráveis a falhas ou a ataques, podem causa a indisponibilidade do sistema. A vantagem desta arquitetura é herdada das duas anteriores. Pode ser citada, principalmente, a facilidade de implementação. Além disso, algumas características provenientes da arquitetura distribuída e a tolerância a falhas nos módulos distribuídos continuam existindo.

2.4 Sistemas de detecção de intrusão distribuídos

Até recentemente, o trabalho sobre sistemas de detecção de intrusão focalizava as estruturas isoladas em um único sistema. Uma organização típica, porém, precisa defender um conjunto distribuído de *hosts* que compõem uma LAN ou inter-rede. Embora seja possível montar uma defesa usando sistemas de detecção de intrusão isolados em cada *host*, uma defesa mais eficaz pode ser obtida pela coordenação e cooperação entre sistemas de detecção de intrusão na rede (STALLINGS, 08).

Porras (PORRAS, 1992) aponta os seguintes problemas principais no projeto do sistema de detecção de intrusão distribuído:

- um sistema de detecção de intrusão distribuído pode ter de lidar com diferentes formatos de auditoria. Em um ambiente heterogêneo, sistemas diferentes empregarão diversos sistemas de coleta de auditoria nativos e, se estiverem usando a detecção de intrusão, podem empregar formatos variados para registros de auditoria relacionados à segurança;
- um ou mais nós na rede servirão como pontos de coleta a análise dos dados dos sistemas na rede. Assim, dados brutos de auditoria ou dados resumidos devem ser transmitidos pela rede. Portanto, existe um requisito para garantir a integridade e a confidencialidade desses dados. A integridade é exigida para impedir que um intruso mascare suas atividades alterando as informações de auditoria transmitida. A confidencialidade é exigida porque as informações de auditoria transmitidas podem ser valiosas;
- pode ser usada uma arquitetura centralizada ou descentralizada. Com uma arquitetura centralizada, existe um ponto de coleta e análise central único de todos os dados de auditoria. Isso facilita a tarefa de correlacionar os relatórios que chegam, mas cria um gargalo em potencial e um ponto único de falha. Com a arquitetura descentralizada, existe mais de um centro de análise, mas estes precisam coordenar suas atividades a troca de informações.

2.5 Trabalhos relacionados

No desenvolvimento de um SDI, uma técnica bastante usual é a utilização de agentes móveis. Agentes móveis são agentes que trafegam entre diversos *hosts* da rede com o intuito de identificar intrusões no sistema. O *framework* MADIF (*Distributed Intrusion Detection*

Framework by Using Mobile Agents) propõe uma arquitetura de componentes para auxiliar no desenvolvimento de um SDI que utiliza esta técnica. Este *framework* é composto por seis tipos de agentes (YE, 08): Agente de Monitoramento, Agente de Análise, Agente Executivo, Agente Gerente, Agente de Obtenção e Agente de Resultado. Os agentes estão dispostos em dois grupos. O primeiro grupo é formado por quatro agentes que não se locomovem pela rede, ou seja, são agentes estáticos. O outro grupo é formado por dois agentes trafegam pela rede, ou seja, são agentes móveis.

Outro conceito utilizado na implementação de um SDI é o de comunidade de agentes. Proposta por Bennatou e Tamine (BENNATOU, 05), IA-DIDS (*Intelligent Agents for Distributed Intrusion*) é uma arquitetura baseada neste conceito, onde estes agentes são especializados e distribuídos. Para identificação de ataques distribuídos, esta arquitetura considera eventos já conhecidos para serem correlacionados com eventos capturados. Um agente denominado Agente Local Especializado (*Specialized Local Agent*) é o ponto central da arquitetura. Sua função é analisar e combinar vários tipos de ataques aos dados de referência.

Uma outra arquitetura utiliza o conceito de implementação centralizada, que normalmente, para indicar o acontecimento de uma intrusão, baseia-se na análise de *logs*. Proposto por (ASAKA, 99), o IDA (*Intrusion Detection Agent System*) é uma implementação que utiliza este conceito. Em um sistema de grande escala, esse tipo de implementação produz um grande tráfego de dados, causando uma grande sobrecarga nos serviços que já são executados na rede. Para solucionar este problema, o IDA propõe o desenvolvimento de agentes móveis que irão se locomover até o sistema alvo, diminuindo assim a quantidade de dados que é trafegado pela rede. Além disso, há mais um aspecto que o diferencia dos sistemas atuais: a detecção da intrusão. Ataques ou intrusões já conhecidos mostram que existem passos que são realizados pelos intrusos até que ele consiga alcançar seu objetivo. Supondo que a quantidade de tipos de marcas deixadas pelo invasor é finita, a detecção de uma intrusão é feita monitorando-se as marcas deixadas pelo invasor no sistema alvo. Uma marca deixada por um invasor, por exemplo, pode ser um arquivo de usuários modificado. Alguns tipos de marcas podem acontecer através de acesso autorizado, gerando uma grande quantidade de falsos positivos. Para que isso não ocorra, um alarme de intrusão não será disparado somente através dessa verificação.

Através destes trabalhos, observa-se que cada implementação tem uma forma particular, seja através de agentes móveis ou comunidade de agentes ou da forma de detecção da intrusão, todos tentam solucionar um mesmo problema, o de se detectar e tratar uma intrusão de forma eficaz.

Embora todos estes trabalhos tenham sua implementação baseada em sistemas multiagentes, nenhum deles trata de uma abordagem explícita de colaboração entre os agentes do sistema.

3 UM SISTEMA DE DETECÇÃO DE INTRUSÕES BASEADO EM AGENTES

Em (JUNIOR, 09) foi proposto um Sistema de Detecção de Intrusões baseado em Sistemas Multiagentes. Este trabalho partiu de um estudo teórico feito em (MEIRA, 01) e propôs a implementação de um sistema de detecção de intrusão usando a tecnologia de agentes distribuídos para evitar a vulnerabilidade de sistemas centralizados.

Um SDI necessita de segurança, integridade, disponibilidade e autonomia, devido a grande quantidade de dados envolvida. Um SDI que tenha funções centralizadas em um único ponto está muito vulnerável a falhas e a ataques. Por este motivo, um sistema de detecção de intrusão baseado em sistema multiagentes tem como objetivo incorporar ao SDI os benefícios dos SMAs, ou seja, descentralização da decisão e execução, autonomia e redundância de dados e execução.

No trabalho de (JUNIOR, 09) os componentes básicos do SDI estão implementados sobre uma arquitetura de agentes, de forma que não existe um único ponto centralizado de funções, evitando-se assim, falhas no sistema. E para garantir a autonomia do sistema, os agentes estão implementados em um modelo bem definido de interação. Através destas interações, é realizada a coordenação interna de todos os agentes envolvidos no sistema.

Além disso, o trabalho acrescenta duas importantes qualidades a um SDI. A primeira qualidade é o fato de possuir diversos tipos de componentes que realizam tarefas específicas. Desta forma, a arquitetura ganha um grande aumento de desempenho e um pequeno impacto nos sistemas que já rodam sobre a rede. A segunda é a ideia do modelo de interoperabilidade de componentes de monitoramento e detecção de intrusão. Através deste modelo, é possível assegurar que os dados gerados pelo sistema tenham uma maior confiabilidade, ou seja, um baixo número de falsos positivos e falsos negativos.

O SDI de (JUNIOR, 09) monitora *hosts* e segmentos de rede de forma pró-ativa, isto é, além de detectar uma intrusão, ele realiza ações para que a intrusão não ocorra por completo ou se ocorrer, cause o menor impacto possível.

3.1 ARQUITETURA

A arquitetura do sistema utiliza agentes em diversos pontos de um segmento de rede. Cada um destes agentes desempenha uma função específica, entretanto, atua sempre em função do objetivo principal. Através da comunicação entre agentes, é possível identificar o acontecimento de intrusões.

O sistema funciona com sete tipos de agentes (Figura 1): *Agentes de Integridade, Agentes de Monitoramento e Análise, Agentes de Reação, Agentes de Retorno, Agentes de Log, Agentes de Dados e Agentes de Informação*.

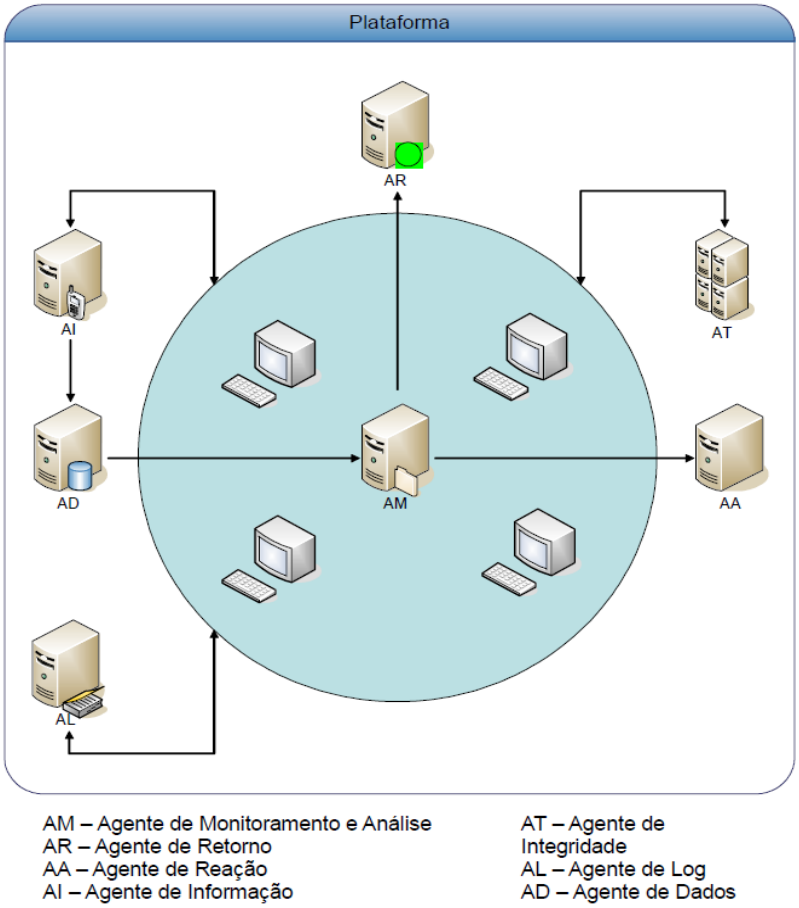


Figura 1: Tipos de Agentes e Interações entre os Agentes.

3.1.1 Agentes do sistema

Os agentes do sistema dividem-se em dois grupos, de acordo com o objetivo de suas funcionalidades. Um grupo realiza a detecção das intrusões. O outro grupo auxilia os demais, provendo informações e garantindo a disponibilidade do sistema.

Em um sistema deste tipo, tolerância a falhas e a auto-recuperação são capacidades de extrema importância para garantir sua confiabilidade. No SDID em questão, a garantia destas capacidades é dada por duas características dos agentes: a autonomia de cada agente e a capacidade de colaboração entre seus agentes.

Para assegurar a tolerância a falhas, o sistema trabalha com agentes que possuam funções repetidas. Esta repetição visa assegurar que no caso de um tipo de agente falhar, outro agente que possua a mesma função estará em funcionamento. Explica também que para assegurar a auto-recuperação, este sistema faz com que no momento da falha de um determinado agente, seja iniciado outro agente para realizar a mesma função daquele que falhou.

O **agente de integridade (*AT*)** garante a disponibilidade do SDID. Ele verifica se os agentes que se registram no sistema estão ativos e se estão funcionando corretamente.

Esta verificação destes agentes é realizada através da troca de mensagens entre esse tipo de agente com os demais agentes que integram o sistema. Um problema no sistema é constatado pelo agente de integridade quando este recebe de uma mensagem não esperada ou o não recebe uma mensagem de confirmação solicitada a outro agente.

O **agente de informação (*AI*)** é usado para registro dos outros agentes no SDID. Esse registro viabiliza a comunicação entre os agentes do sistema, que é fundamental para a auto-recuperação.

Além disso, as informações do registro, tais como a localização dos agentes e se estes estão ativos, são repassadas aos agentes de dados para que estes façam a persistência no banco de dados. Estes dados são armazenados, pois podem ser necessários podem ser utilizados posteriormente.

O **agente de monitoramento e análise (*AM*)** procura identificar o acontecimento de uma intrusão no sistema. Esta identificação é realizada através do monitoramento e análise de eventos específicos nos diversos tipos de serviços providos pelo sistema alvo.

Quando uma intrusão é detectada, estes agentes agrupam-se para realizar uma verificação mais precisa. Esse recurso garante a interoperabilidade ao SDID. A interoperabilidade é definida pelo administrador, durante a configuração do sistema, que indica para quais tipos de intrusões é necessário que haja uma união de dois ou mais agentes.

O **agente de retorno (*AR*)** recebe alarmes de intrusões gerados pelos agentes de monitoramento. Dependendo da informação recebida, este agente pode ativar um agente de reação.

Estes alarmes são repassados ao administrador do sistema e podem ser visualizados de diversas formas, conforme sua configuração. As formas disponíveis são: a exibição na interface do sistema, o envio por e-mail e o envio de mensagens SMS.

O **agente de dados (*AD*)** recebe informações que precisam ser armazenadas e as persistem em um banco de dados. Além de armazená-las, ele é capaz de recuperá-las. Estas informações são para utilização dos demais agentes do sistema.

O **agente de log (*AL*)** tem a responsabilidade de armazenar e analisar *logs* no sistema. Um agente de *log* pode ser requisitado pelos demais agentes para registrar ou recuperar uma informação. Para que se torne uma informação segura, elas são armazenadas em mídias de dados não voláteis.

Estas informações são exibidas na interface do sistema e podem ser requisitadas no momento que administrador do sistema precisar.

Por fim, o **agente de reação (*AA*)** é ativado pelo agente de retorno e quando ativado, todas as suas ações têm como finalidade tentar impedir que o intruso tenha sucesso na invasão do sistema.

3.2 Infraestrutura do sistema

A infraestrutura do SDID em análise utiliza um sistema de banco de dados distribuído (utilizado pelos agentes de dados), uma ferramenta para realizar a detecção de determinados tipos de intrusões (que será encapsulada em um agente de monitoramento) e uma plataforma que implementa agentes.

3.2.1 Sistema de banco de dados distribuído – MySQL Cluster

O sistema de banco de dados é responsável pela gravação e recuperação dos dados que precisam ser persistidos de forma a não se perderem, mesmo que o sistema não esteja em uso.

Segundo (JUNIOR, 09), o banco de dados do SDID não pode ser um ponto central sujeito a falha. Assim, é necessário garantir que o sistema não pare, mesmo que um problema afete o banco de dados. Essa importante característica do SDID é chamada de alta disponibilidade.

Para garantir a alta disponibilidade do sistema, todos os dados do banco de dados são replicados pelo menos uma vez. O sistema utiliza o banco de dados

MySQL por ser de uso gratuito e por implementar a replicação de dados. Atualmente, o MySQL é um SGBD muito utilizado, devido a integração nativa com muitas ferramentas de programação e de ser fácil utilização, podendo ser comparado com diversos SGBD's comerciais.

3.2.2 SNORT

No SDID em questão, a ferramenta escolhida que identifica uma intrusão, através da interceptação e análise dos pacotes que trafegam na rede é o SNORT (SNORT, 08). O SNORT é uma ferramenta de detecção de intrusão de rede, baseado em regras, capaz de monitorar pequenas redes TCP/IP, detectando tráfego suspeito e ataques externos, fornecendo dados para as decisões dos administradores.

O Snort é um software livre completo que desempenha esta função. A facilidade de configuração e utilização são pontos fortes para adesão a esse sistema. Ele evoluiu bastante devido o apoio de grandes empresas e da comunidade de software livre.

Neste SDID, utiliza-se o Snort para gerar os alertas de eventos em um arquivo de *log*, que posteriormente são lidos pelos agentes de monitoramento e análise. A cada alteração deste arquivo, este agente analisa se foram incluídas neste arquivo informações que possam indicar uma intrusão.

Para a simulação do agente de monitoramento, (JUNIOR, 09) implementou regras para a detecção dos seguintes acontecimentos no *log* do Snort:

- *NETBIOS SMB-DS Session Setup AndX request Unicode username overflow attempt:* uma ocorrência deste evento identifica a tentativa de execução de código malicioso nos sistemas RealSecure e BlackICE que possuem vulnerabilidade no módulo SMB.
- *NETBIOS DCERPC Remote Activation bind attempt:* a ocorrência deste evento identifica uma tentativa de execução de código malicioso em sistemas Windows que possuem uma falha na interface RPC.
- *DDOS mstream client to handler:* a ocorrência deste evento identifica uma tentativa de efetuar uma intrusão através de um ataque de negação de serviço distribuído.

- *ICMP L3retriever Ping:* a ocorrência deste evento identifica a tentativa de obter informações de alvo através do protocolo ICMP.

3.2.3 Plataforma de agentes – JADE

Java (JAVA, 95) é uma linguagem de programação que possui características conhecidas e recomendadas, tais como: orientação a objetos, robustez, segurança, independência de plataforma, entre outras. Devido a essas características, o SDID proposto é implementado utilizando a linguagem Java.

Mesmo utilizando as facilidades propostas por esta linguagem, o desenvolvimento de um SMA necessita de uma série de componentes básicos que fazem parte de sua infraestrutura. O *framework* chamado Jade (*Java Agent Development Framework*) (JADE, 08) foi utilizado neste trabalho.

As principais características oferecidas por este *framework* que facilitam o desenvolvimento de um SMA, são:

- Plataforma: O sistema pode ser distribuído em diversas máquinas que possuam uma JVM (*Java Virtual Machine*) instalada. A comunicação é realizada através de RMI.
- Interface gráfica: O Jade disponibiliza uma interface gráfica para facilitar a administração das diversas plataformas e dos diversos agentes que compõem o sistema.
- Comunicação: Seguindo especificações da FIPA-ACL, o *framework* disponibiliza ferramentas que transportam e traduzem as mensagens trocadas pelos agentes.
- Serviços descritos pela FIPA: disponibiliza dois serviços: o AMS (*Agent Management System*), que gerencia a plataforma e armazena as informações dos agentes ativos e o DF (*Directory Facilitador)*, que é o serviço de páginas amarelas.

3.3 Funcionamento do SDID

O SDID foi testado em uma rede de computadores independente, montada somente para essa finalidade. Ela é formada somente por computadores dedicados ao sistema ou a algum recurso utilizado por este sistema.

Os computadores utilizados no teste executam o sistema operacional *Linux*, distribuição *Ubuntu Server*, e estão configurados com os serviços básicos necessários para o exercício de sua função no sistema. O esquema do ambiente de teste proposto é mostrado na Figura 2.

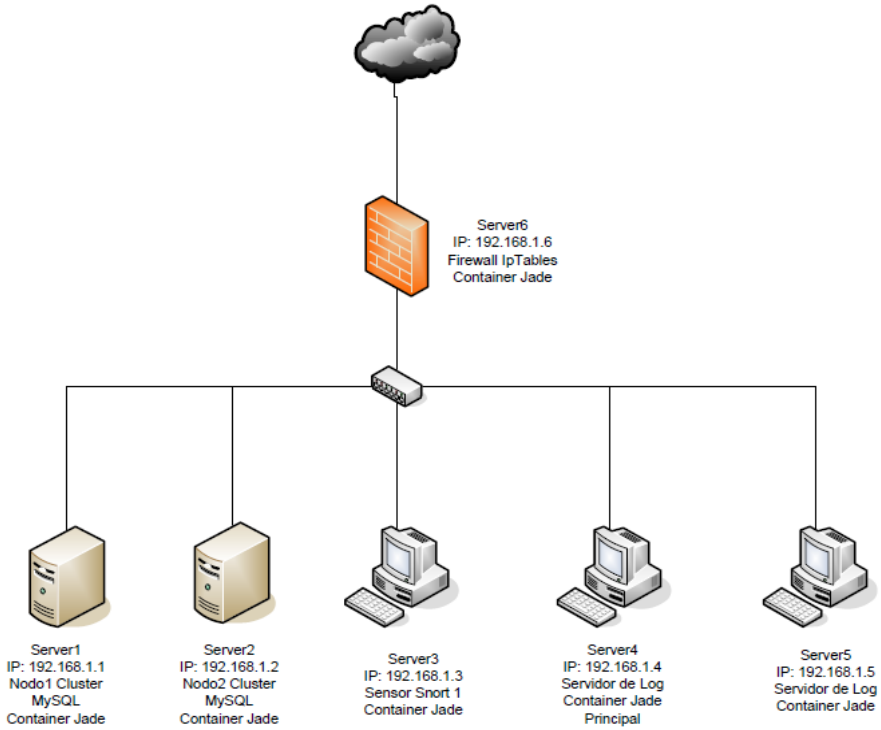


Figura 2: Esquema do ambiente de testes (JUNIOR, 09).

Os serviços básicos são: um *cluster MySQL* formado por 2 computadores, um *firewall* que utiliza o *IpTables* (IPTABLES, 10) no computador que possui acesso a rede externa (internet), o serviço do SNORT para detecção de intrusão em outros dois computadores e um serviço configurado para receber chamadas de *log* remotamente em outras duas máquinas.

Para instanciar o sistema, é preciso informar em qual dos contêineres Jade vai ser iniciado o agente de integridade principal, a classe deste agente e um parâmetro para inicializar o sistema. Este agente de integridade instancia, em seu próprio contêiner, um agente de dados para fazer a busca da configuração do sistema no banco de dados.

Nesta configuração, o agente de integridade encontra quais tipos de agentes devem ser instanciados e suas respectivas quantidades. Além disso, essa configuração contém o nome dos dois computadores que contém um sensor Snort ativo, para que o agente de integridade instancie neles os agentes de monitoramento e análise.

Cada agente instanciado no sistema é distribuído aleatoriamente pelos computadores da rede, desde que não dependa de um serviço específico rodado em uma determinada máquina, como no caso do agente de monitoramento e análise.

Para simular uma tentativa de intrusão, foram retiradas as restrições de portas do *firewall* e para que fossem gerados alertas nos *logs* do SNORT, foi utilizado o software NESSUS (NESSUS, 09). A utilização do NESSUS gerou eventos no *log* do SNORT e estas informações foram lidas e analisadas pelo agente de monitoramento e análise. A análise do *log* resultou na identificação de uma tentativa de intrusão.

Identificada a invasão, o agente de análise e monitoramento enviou uma mensagem ao agente de *log*, que alertou o agente de reação e retorno para que decidisse, conforme parametrização no banco de dados, se alguma ação seria tomada. Nesta simulação, o sistema estava configurado para envio de e-mail de qualquer evento que fosse detectado e reconhecido, independente da classificação, e para criar uma regra no *firewall* que impeça a conexão do endereço de origem dos eventos. A Figura 3 ilustra a iteração entre os agentes no sistema:

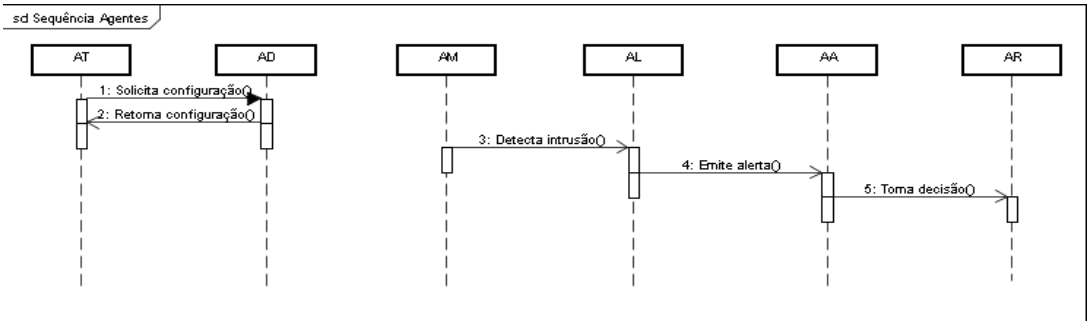


Figura 3: Iteração entre os agentes do sistema.

A indisponibilidade de um agente do sistema foi simulada desligando-se um dos computadores. Logo, o agente de integridade agiu inicializando em um novo contêiner todos os agentes que pertenciam ao contêiner perdido. A regra de não inicializar mais de um agente do mesmo tipo em um contêiner foi respeitada.

3.4 Considerações finais

O atual SDID baseado em agentes previne a indisponibilidade de um tipo de agente, por falha no sistema ou por intrusão, através de um agente de integridade que inicia um novo agente do mesmo tipo para suprir a funcionalidade necessária. Agentes de mesmo tipo podem coexistir no ambiente, conforme parametrização do administrador, reforçando técnicas de tolerância a falhas.

No entanto, as funcionalidades de um SMA podem ser melhor exploradas no contexto deste trabalho, principalmente no que diz respeito à capacidade de colaboração entre os agentes. Por exemplo, a replicação de agentes de monitoramento e análise pode ser também explorada através de diferentes tipos de análise nos *logs* do Snort. Agentes de monitoramento e análise distribuídos na rede podem compartilhar informações parciais, monitorar diversos ataques simultâneos e ainda trocar informações sobre ações tomadas por outros agentes em outros ataques.

A proposta deste trabalho é a continuação do SDID de (JUNIOR, 09) para estender a funcionalidade de colaboração entre os agentes, a qualidade da informação de monitoramento e análise executada por vários agentes em várias máquinas.

4 SDID COLABORATIVO – IMPLEMENTAÇÃO E EXPERIMENTOS

As funcionalidades de um SMA podem ser melhor exploradas no contexto deste trabalho, principalmente no que diz respeito à capacidade de colaboração entre os agentes. Por exemplo, a replicação de agentes de monitoramento e análise pode ser também explorada através de diferentes tipos de análise nos *logs* do Snort. Agentes de monitoramento e análise distribuídos na rede podem compartilhar informações parciais, monitorar diversos ataques simultâneos e ainda trocar informações sobre ações tomadas por outros agentes em outros ataques.

4.1 Definição de requisitos para a colaboração multiagente no SDID

Toda uma infraestrutura multiagentes foi desenvolvida para a construção de SDIs distribuídos plenamente funcional e parametrizável em termos da quantidade de agentes, máquinas e sistemas de detecção instalados. Podemos explorar a partir deste trabalho, as possibilidades da própria arquitetura distribuída e dos sistemas multiagentes através da sua capacidade de colaboração. Por exemplo, o SDI atual, na forma como se apresenta, não é capaz ainda de gerenciar os *logs* gerados pelos agentes de forma distribuída e manter sua persistência no sistema. Atualmente, os agentes de *log* (AL) apenas mantém um registro local de suas varreduras. Em consequência, as detecções analisadas no *log* do SNORT pelos agentes de monitoramento e análise (AM) podem ainda sofrer redundância quanto a reação do sistema. Neste momento, o sistema ainda não se preocupa com a troca de conhecimento entre os agentes, o que habilita aos agentes de reação tomarem decisões já estabelecidas no sistema ou determinarem várias ações para as mesmas intrusões.

Levando estas observações em consideração, este trabalho pretende continuar o projeto anterior dotando o atual SDI baseado em agentes desta capacidade colaborativa. A hipótese inicial é a de que os agentes de monitoramento e análise podem trocar informações sobre potenciais intrusões detectadas no *log* e decidir, caso ocorram múltiplas detecções para a mesma intrusão, qual agente deve continuar o processo. Outra atividade diz respeito à persistência dos *logs*. O gerenciamento dos *logs* do sistema através do *syslog* do sistema Linux pode resolver a questão da persistência dos *logs*, enquanto a troca de conhecimento entre estes agentes pode impedir a escolha de uma ação já executada anteriormente, por

exemplo, não há necessidade de duplicar uma mesma regra no *firewall*. No contexto desta proposta, será limitado a persistir a informação dos *logs* no conhecimento interno de cada agente de análise e monitoramento.

4.2 Arquitetura Revista

Neste trabalho a arquitetura proposta em (JUNIOR, 09) sofreu alterações para implementar o recurso de detecção colaborativa. Para isto, foi adicionado na arquitetura outro agente de monitoramento e análise para realizar a colaboração com o agente já existente, além da interação explícita entre os dois agentes. Na Figura 4 é mostrado o novo esquema desta arquitetura.

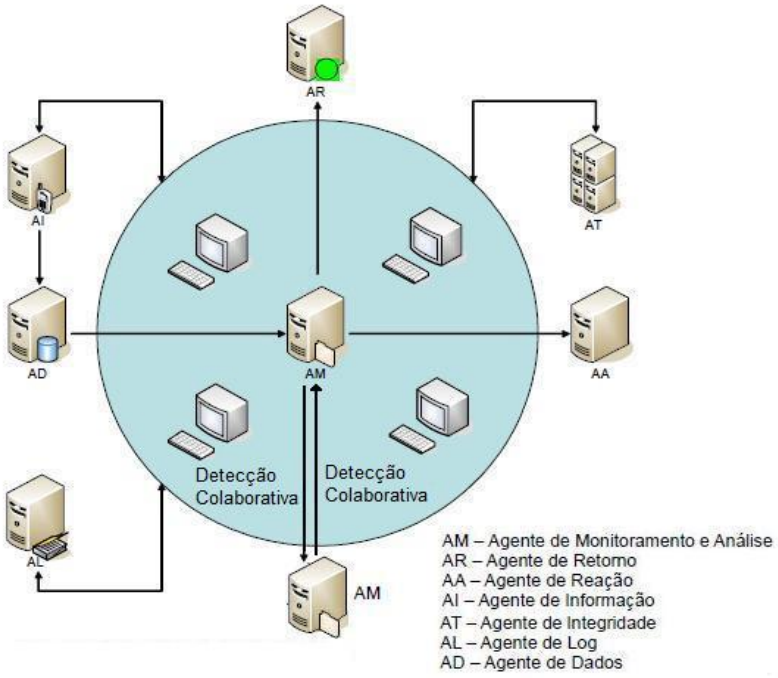


Figura 4: Arquitetura com detecção colaborativa.

A adição do recurso de detecção colaborativa adicionou à arquitetura uma certa inteligência ao evitar a redundância de informações no *firewall* do sistema *IpTables*[1]. A Figura 5 mostra a sequência de execução dos processos dentro da nova arquitetura.

1 *Iptables* é um sistema de controle de filtros para protocolos ipv4 com gestão das regras do *firewall* (IPTABLES, 10).

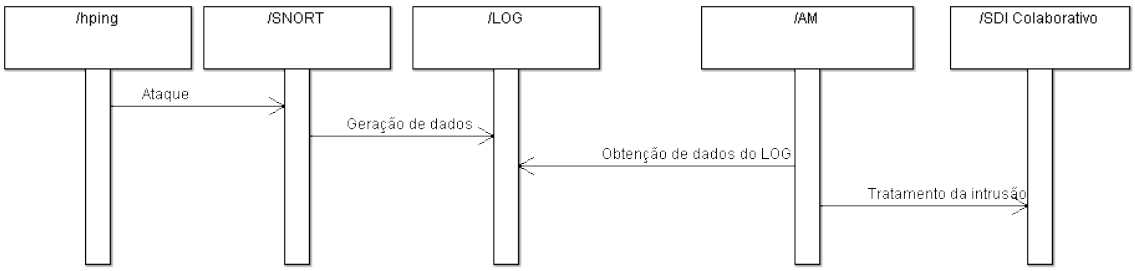


Figura 5: Sequência de execução de processos.

Neste modelo, uma ferramenta dispara pacotes que identificam uma tentativa de intrusão para dentro da rede. Um sensor ativo do Snort (SNORT, 10) captura esses pacotes e grava os eventos em um arquivo de *log*. O agente de monitoramento e análise, que está monitorando as modificações deste arquivo, faz a leitura dos dados gerados e analisa se houve algum pacote que que identifique um ataque. Quando um ataque é identificado por este agente, os agentes de monitoramento trocam informações sobre o evento ocorrido e decidem qual deles irá assumir a responsabilidade de tratar a intrusão.

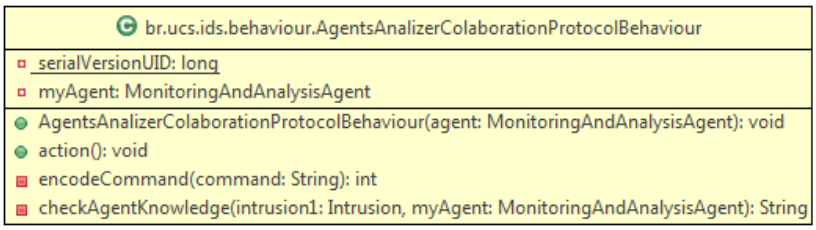
4.3 Modelos dos Agentes

A Tabela 1 mostra e explica os serviços oferecidos pelo agente de monitoramento e análise:

br.ucs.ids.agent.MonitoringAndAnalysisAgent serialVersionUID: long MONITORING_AND_ANALYSIS_AGENT_DF_TYPE: String logLines: Queue<String> myKnowledge: Vector<Intrusion> intrusionDetectionOntology: Ontology logOntology: Ontology codec: Codec setup(): void registerInDirectoryFacilitator(): void getMyKnowledge(): Vector<Intrusion> listarConhecimento(): void	**Serviços básicos do agente:** • *setup*: método da classe jade.core.Agent chamado quando um agente é instanciado; • *registerInDirectoryFacilitator*: método para registar o agente no DF; • *getMyKnowledge*: método que retorna a base de conhecimento do agente; • *listarConhecimento*: método que lista na saída padrão os dados da base de conhecimento.

Tabela 1: Serviços do agente de monitoramento e análise

A Tabela 2 mostra e explica os serviços da classe responsável pelo comportamento colaborativo.



Serviços do comportamento colaborativo:

- *AgentsAnalizerColaborationProtocolBehaviour:* método construtor da classe;
- *action:* método da classe *jade.core.behaviours.CyclicBehaviour;*
- *encondeCommand:*codifica o comando recebido para um código numérico;
- ***checkAgentKnowlegde:*** verifica o conhecimento do agente sobre a intrusão.

Tabela 2: Serviços do comportamento colaborativo

4.4 Arquitetura Geral

A implementação da detecção colaborativa trouxe modificações ao sistema e para isto, algumas classes foram modificadas e outras foram criadas. A *MonitoringAndAnalisysAgent,* classe do agente de monitoramento e análise, foi modificada para adicionar ao agente o comportamento da classe *AgentsAnalizerColaborationProtocolBehaviour.* Além disso, foi acrescentado um atributo do tipo *java.util.Vector* representa a base de conhecimento do agente, composta por objetos da classe *Intrusion*.

O processo da detecção colaborativa é disparado ao detectar dados suspeitos no *log* do Snort, pela classe *SnortLogAnalizeBehaviour* através do método *SendIntrusionMessageForAMAgents*. A classe *ColaborationProtocol* é responsável por implementar o protocolo de comunicação interpretado pelos agentes de monitoramento e análise. A passagem do protocolo de colaboração entre as classes do sistema é realizada utilizando-se a classe *ProtocolMessage.* Esta classe contém um atributo, do tipo *string*, para armazenar a mensagem do protocolo de colaboração. A Figura 6 ilustra a nova arquitetura.

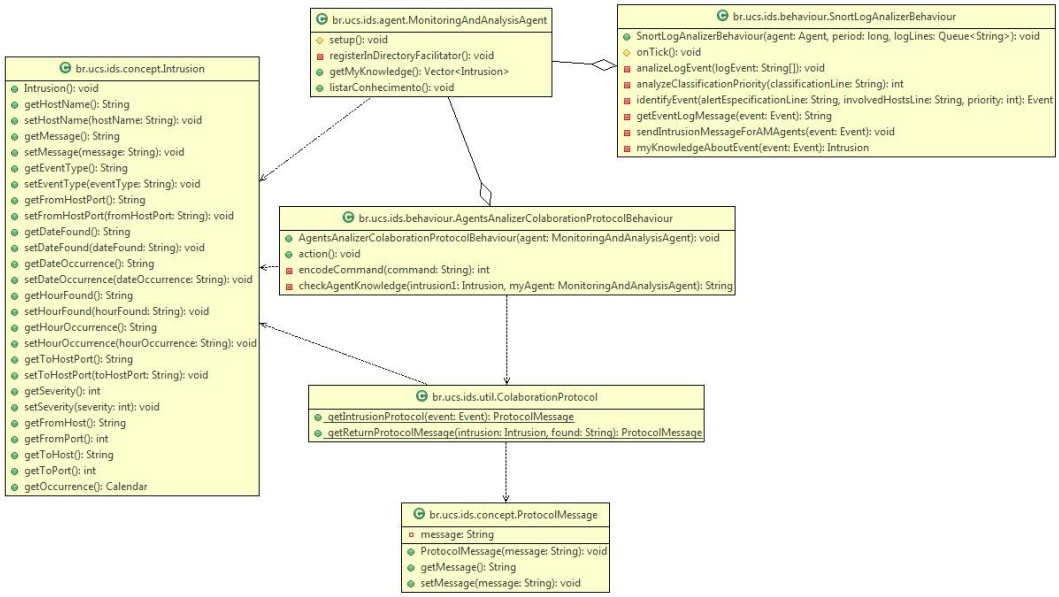


Figura 6: Diagrama de Classes da nova arquitetura (somente modificações)

4.5 Protocolo de Colaboração

As informações contidas no protocolo são, respectivamente: nome do servidor que está enviando a mensagem, data do envio da mensagem; hora do envio da mensagem, ID do evento ocorrido (tipo do ataque), data em que SNORT identificou o evento, hora em que SNORT identificou o evento, IP do *host* de origem, porta do *host* de origem, IP do *host* de destino e porta do *host* de destino.

Quando uma intrusão é detectada, o agente que a identificou busca, em sua base de conhecimento, informações sobre um prévio acontecimento deste mesmo evento. Se este agente possui tal conhecimento, ele assume a responsabilidade da tratativa do evento, uma vez que possui o conhecimento para isso. Caso não possua em sua base tal conhecimento, ele envia uma mensagem contendo a literal “INTRUSION_DETECTED” para os outros agentes de monitoramento registrados no DF. Abaixo, um exemplo desta informação:

```
SERVER7;07/07/2010;20:15:26;INTRUSION_DETECTED;1:228:7;07/07/2010;20:15:27;200.
233.17.190:51959;192.168.1.5:1
```

Nesta mensagem o agente de monitoramento que é executado SERVER7 envia aos outros agentes de monitoramento e análise no dia 07/07/2010 às 20:15:26 uma mensagem

contendo a literal “INTRUSION_DETECTED”. O Snort detectou um tipo de ataque cujo identificador é 1:228:7 em 07/07/2010 às 20:15:27 originado do IP 200.233.17.19 e porta 51959 com destino para o IP 192.168.1.5 e porta 1.

Quando outro agente de análise e monitoramento recebe esta mensagem, ele verifica se possui em sua base de conhecimento informações relativas ao evento. Se ele possuir, este agente então assume a responsabilidade da tratativa do evento e responde “INSTRUSION_ALREADY_FOUND” ao agente que enviou a mensagem. Abaixo, o exemplo desta mensagem:

```
SERVER5;19/05/2010;22:13:06;INSTRUSION_ALREADY_FOUND;1:249:8;09/03/2010;18:51:5
8;192.168.1.97:2131;192.168.1.4:15104
```

Se este agente não possuir tal conhecimento, ele responde “INSTRUSION_NOT_FOUND” ao agente que enviou, repassando a responsabilidade da tratativa do ataque ao agente que a detectou. Abaixo, um exemplo testa mensagem:

```
SERVER5;19/05/2010;22:13:06;INSTRUSION_NOT_FOUND;1:249:8;09/03/2010;18:51:58;19
2.168.1.97:2131;192.168.1.4:15104
```

O agente de monitoramento e análise responsável por tratar uma intrusão adiciona as informações da intrusão em sua base de conhecimento, caso ainda não a possua.

O controle do envio e recebimento das mensagens é realizado pelo *framework* JADE. As mensagens enviadas nesta plataforma exigem um tipo de performativo[2], isto é, é exigido que se especifique o tipo de mensagem. O performativo utilizado nesse protocolo é do tipo INFORM. Este tipo foi utilizado pois objetiva apenas a troca de mensagens em caráter informativo. A Figura 7, mostra como ocorre a troca de mensagens entre os agentes de Monitoramento e Análise.

2 Enunciados Performativos são uma implementação dos atos perlocucionários de J. L. Austin, onde os enunciados proferidos por um agente na forma afirmativa e na voz ativa realizam uma ação no interlocutor [KQML, 10].

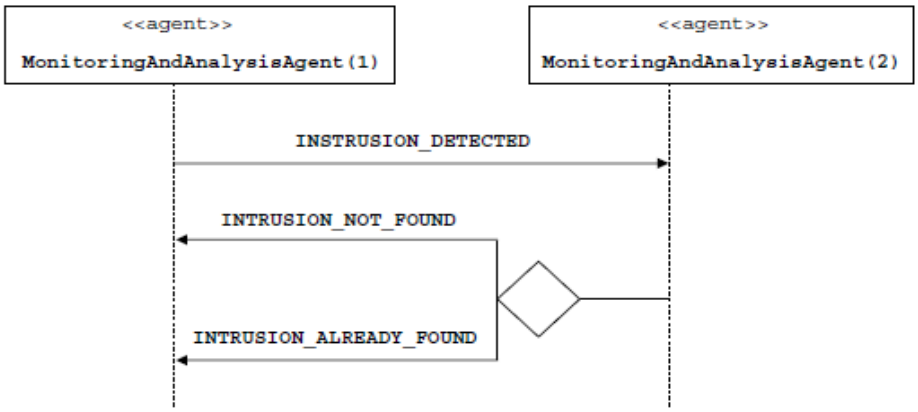


Figura 7: Troca de mensagens entre os agentes de monitoramento e análise

4.6 Implementação

Para a implementação de um protocolo de colaboração entre os agentes de monitoramento e análise do sistema, foram alteradas as classes *MonitoringAndAnalysisAgent* e *SnortLogAnalizerBehaviour* e criadas uma nova classe: a *AgentsAnalizerColaborationProtocolBehaviour.*

As alterações implementadas no trabalho de Junior, (JUNIOR, 09) constituem-se na criação da base de conhecimento dos agentes de monitoramento e análise, procedimentos de pesquisa nessa base de conhecimento e a criação de um comportamento para a troca de mensagens entre agentes. A seguir, são explicadas de forma mais detalhada as alterações implementadas.

A classe *MonitoringAndAnalysisAgent* é a classe que representa o agente de monitoramento e análise. Nela foi adicionada um *array* unidimensional para armazenar as intrusões tratadas por este agente, isto é, a base de conhecimento do agente. A Figura 8 mostra a alteração realizada:

```
public class MonitoringAndAnalysisAgent extends Agent {

    private static final long serialVersionUID = 1L;
    public static final String MONITORING_AND_ANALYSIS_AGENT_DF_TYPE = "AM";
    private Queue<String> logLines;
    private Vector<Intrusion> myKnowledge;
```

Figura 8: Declaração da base de conhecimento do agente

A recuperação da base de conhecimento de um agente é feita através do método *getMyKnowledge(),* como mostra a Figura 9.

```
public Vector<Intrusion> getMyKnowledge() {
    return myKnowledge;
}
```

Figura 9: Método para recuperação da base de conhecimento do agente

A criação da base de conhecimento do agente é feita no método *setup,* que é executado na criação do agente, através do comando *this.myKnowledge = new Vector<Intrusion>()*. Além disso, neste momento é adicionado a este agente um novo comportamento, através do comando *addBehaviour(new AgentsAnalizerColaborationProtocolBehaviour(this))*. Este comportamento dá ao agente de monitoramento e análise o recurso de colaboração com outros agentes do mesmo tipo. A Figura 10 mostra o método com as devidas alterações.

```
protected void setup() {
    // Registro das ontologias utilizadas
    try {
        intrusionDetectionOntology = IntrusionDetectionOntology.getInstance();
        logOntology = LogOntology.getInstance();
    } catch (BeanOntologyException e) {
        e.printStackTrace();
    }

    getContentManager().registerOntology(intrusionDetectionOntology, IntrusionDetectionOntology.NAME);
    getContentManager().registerOntology(logOntology, LogOntology.NAME);

    // Registro das linguagens utilizadas
    codec = new SLCodec();
    getContentManager().registerLanguage(codec);

    // Inicialização de variáveis
    logLines = new LinkedList<String>();

    // Registro dos comportamentos
    addBehaviour(new SnortLogReaderBehaviour(this, 2000, "/var/log/snort/alert", logLines));
    addBehaviour(new SnortLogAnalizerBehaviour(this, 2000, logLines));
    addBehaviour(new AgentsAnalizerColaborationProtocolBehaviour(this));

    // Registro nas páginas amarelas
    registerInDirectoryFacilitator();

    // inicializa conhecimetos
    this.myKnowledge = new Vector<Intrusion>();
    System.out.println("MonitoringAndAnalysisAgent iniciado");
}
```

Figura 10: Adição do comportamento de colaboração na criação do agente

Na classe *SnortLogAnalizerBehaviour* foi alterado o método *analizeLogEvent(),* responsável por analisar o arquivo de *log* do SNORT e identificar uma intrusão. Após a identificação de um evento de intrusão, foi adicionado a lógica a seguir, para verificar na base de conhecimento do agente que capturou o evento se este tem um prévio conhecimento do fato, conforme Figura 11**:**

```
if (event != null) {
    System.out.println(getEventLogMessage(event));
    Logger.logMessage(getEventLogMessage(event), myAgent);

    if (myKnowledgeAboutEvent(event) == null) {
        sendIntrusionMessageForAMAgents(event);
    } else {
        ProtocolMessage plMsg = ColaborationProtocol.getIntrusionProtocol(event);
        ACLMessage aclIntrusionMessage = new ACLMessage(ACLMessage.INFORM);
        aclIntrusionMessage.setSender(myAgent.getAID());
        aclIntrusionMessage.setContent( plMsg.getMessage() );
        aclIntrusionMessage.setProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST);
        aclIntrusionMessage.setLanguage(FIPANames.ContentLanguage.FIPA_SL0);
        aclIntrusionMessage.setOntology(JADEManagementOntology.NAME);
        // envia para si mesmo a mensagem
        aclIntrusionMessage.addReceiver(myAgent.getAID());
        myAgent.send(aclIntrusionMessage);
    }
}
```

Figura 11: Verificação do conhecimento e envio da mensagem de colaboração

Esta lógica verifica se o agente que detectou a intrusão tem conhecimento sobre este evento. Se não possuir tal conhecimento, então ele irá enviar uma mensagem com dados do evento para os outros agentes de monitoramento e análise. Caso contrário, ele manda esta mensagem para si mesmo. O objetivo deste processo é seguir o padrão de colaboração existente entre os agentes de monitoramento e análise, garantindo que o SDID tenha um único ponto para a lógica do processo de tratativa da intrusão, evitando que haja tratativas diferentes em pontos distintos do sistema.

O método *sendIntrusionMessageForAMAgents* descrito na Figura 12, monta a mensagem do protocolo a ser enviada aos outros agentes de monitoramento e análise. Após, procura no DF do JADE para quais os destinatários a mensagem será enviada.

```
private void sendIntrusionMessageForAMAgents(Event event) {
    // monta o objeto ProtocolMessage contendo a mensagem a ser passada aos outros agentes
    ProtocolMessage plMsg = ColaborationProtocol.getIntrusionProtocol(event);
    // System.out.println("(" + myAgent.getAID().getLocalName() + ") MENSAGEM DE COLABORACAO: " + plMsg.getMessage() );

    // monta a mensagem no padrão ACL
    ACLMessage aclIntrusionMessage = new ACLMessage(ACLMessage.INFORM);
    aclIntrusionMessage.setSender(myAgent.getAID());
    aclIntrusionMessage.setContent( plMsg.getMessage() );
    aclIntrusionMessage.setProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST);
    aclIntrusionMessage.setLanguage(FIPANames.ContentLanguage.FIPA_SL0);
    aclIntrusionMessage.setOntology(JADEManagementOntology.NAME);

    DFAgentDescription[] agents =
        Utils.searchAgentsInDFByType(myAgent, MonitoringAndAnalysisAgent.MONITORING_AND_ANALYSIS_AGENT_DF_TYPE);

    if (agents != null && agents.length > 0) {
        for (int i = 0; i < agents.length; i++) {
            if (!agents[i].getName().equals(myAgent.getName())) {
                aclIntrusionMessage.addReceiver(agents[i].getName());
                // System.out.println("Enviando para " + agents[i].getName());
            }
        }
    }

    myAgent.send(aclIntrusionMessage);
}
```

Figura 12: Envio de mensagem de colaboração aos outros agentes de monitoramento e análise

A classe *AgentsAnalizerColaborationProtocolBehaviour* foi criada para adicionar ao agente a funcionalidade de colaboração. Nesta se encontra toda o comportamento de troca de mensagem e as ações que o agente realiza, conforme Figura 13:

```
switch(commandCode) {
    case 1: // INTRUSION_DETECTED
        System.out.println("Recebi: " + aclRequestMessage.getContent());
        intrusion = Utils.parseProtocolToIntrusion(aclRequestMessage.getContent());
        sMessage = checkAgentKnowledge(intrusion, myAgent);
        pmMessage = ColaborationProtocol.getReturnProtocolMessage(intrusion, sMessage);

        aclResponseMessage = new ACLMessage(ACLMessage.INFORM);
        aclResponseMessage.setSender(myAgent.getAID());
        aclResponseMessage.setContent( pmMessage.getMessage() );
        aclResponseMessage.addReceiver(aclRequestMessage.getSender());
        System.out.println("Enviando: " + aclResponseMessage.getContent());
        myAgent.send(aclResponseMessage);
        break;
    case 2: // INTRUSION_NOT_FOUND
        System.out.println("Recebi: " + aclRequestMessage.getContent() );
        intrusion = Utils.parseProtocolToIntrusion(aclRequestMessage.getContent());
        // adiciona a intrusao a base de conhecimento do agente
        myAgent.getMyKnowledge().add(intrusion);
        // trata a intrusão
        myAgent.addBehaviour(new InformIntrusionBehaviour(Utils.parseIntrusionToEvent(intrusion)));
        break;
    case 3: // INTRUSION_ALREADY_FOUND
        System.out.println("Recebi: " + aclRequestMessage.getContent() );
        System.out.println("-----------------------------------------------------------------");
        System.out.println("| AVISO IMPORTANTE:                                             |");
        System.out.println("| O sistema detectou uma instrusão anteriormente detectada!     |");
        System.out.println("| Verifique possíveis problemas no firewall.                    |");
        System.out.println("-----------------------------------------------------------------");
        break;
}
```

Figura 13: Lógica da detecção colaborativa

4.7 Experimentos

Para a realização dos testes, foi montada uma estrutura de rede com sete máquinas, formada somente por computadores dedicados ao sistema de detecção de intrusão ou a algum recurso adicional necessário ao funcionamento do sistema.

Os computadores desta rede utilizam o sistema operacional *Linux*, distribuição *Ubuntu Server*, e todos atuam como contêineres JADE. A Figura 14 representa o esquema deste ambiente.

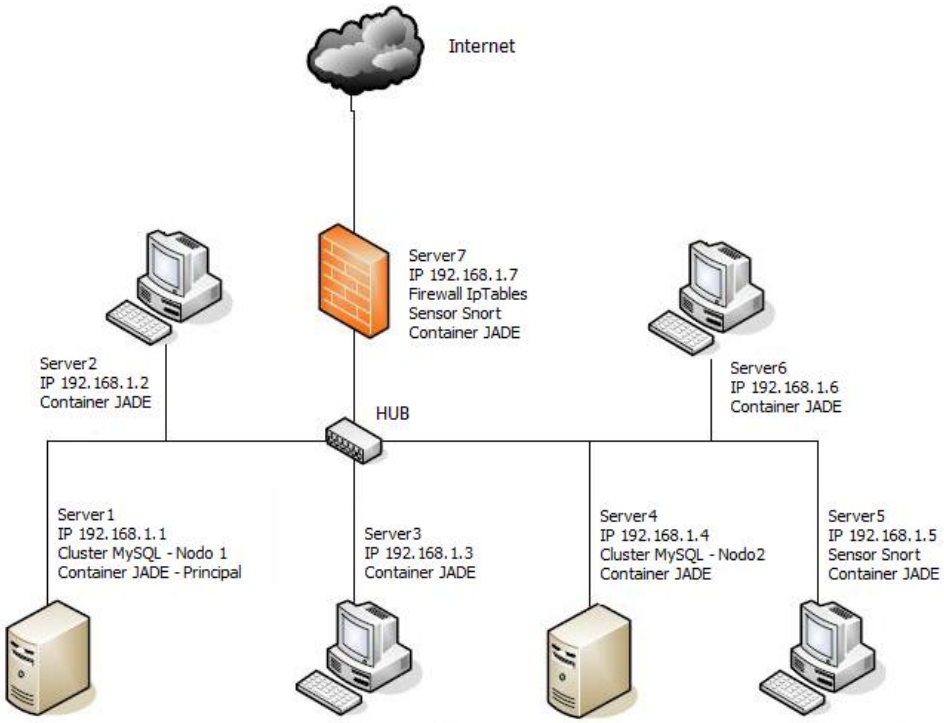


Figura 14: Estrutura de rede utilizada do experimento

4.7.1 Configuração do Ambiente

Neste ambiente, existem máquinas que cumprem funções específicas. Para inicializar o sistema, o JADE requer um contêiner principal. Além de ser o contêiner principal do JADE, o Server1 contém o primeiro nodo do *Cluster MySQL*. O Server4 possui o segundo nodo no *Cluster MySQL*.

O Server5 e Server7 contém particularidades dentro deste ambiente. Devido ao fato dessas máquinas serem os sensores ativos do Snort, os agentes de monitoramento e análise do sistema são executados somente nestas duas máquinas. O restante dos agentes é distribuído de forma aleatório na estrutura da rede.

Para os computadores que exercem a função de *Cluster MySQL*, foram criados *scripts* chamados *mysql.sh* para auxiliar a execução do serviço no sistema operacional: Abaixo, o conteúdo deste *script*:

```
#!/bin/bash
cd /etc/init.d
./mysql stop
./mysql-ndb stop
./mysql-ndb-mgm stop
```

```
./mysql-ndb-mgm start
./mysql-ndb start
./mysql start
```

Neste *script*, os serviços de banco de dados, replicação e gerenciamento são parados e logo em seguida são inicializados. É necessário pará-los antes de iniciar, pois ao ligar o computador, estes serviços são inicializados incorretamente.

Foram criados também, para todos os micros da rede, *scripts* chamados *start.sh* que ao serem executados, inicializam o sistema JADE em cada computador. Abaixo, o conteúdo deste *script*:

No server1:

```
#!/bin/bash
java -cp ids.jar jade.Boot -name IDS -jade_domain_df_autocleanup true -detect-
main false -services
jade.core.replication.MainReplicationService\;jade.core.replication.AddressNoti
ficationService
```

No server2:

```
#!/bin/bash
java -cp ids.jar jade.Boot -container-name server2 -jade_domain_df_autocleanup
true -backupmain -host 192.168.1.1 -name IDS -services
jade.core.replication.MainReplicationService\;jade.core.replication.AddressNoti
ficationService
```

No server3:

```
#!/bin/bash
java -cp ids.jar jade.Boot -container-name server3 -jade_domain_df_autocleanup
true -backupmain -host 192.168.1.1 -name IDS -services
jade.core.replication.MainReplicationService\;jade.core.replication.AddressNoti
ficationService
```

No server4:

```
#!/bin/bash
java -cp ids.jar jade.Boot -container-name server4 -jade_domain_df_autocleanup
true -backupmain -host 192.168.1.1 -name IDS -services
jade.core.replication.MainReplicationService\;jade.core.replication.AddressNoti
ficationService
```

No server5:

```
#!/bin/bash
java -cp ids.jar jade.Boot -container-name server5 -jade_domain_df_autocleanup
true -backupmain -host 192.168.1.1 -name IDS -services
jade.core.replication.MainReplicationService\;jade.core.replication.AddressNoti
ficationService
```

No server6:

```
#!/bin/bash
java -cp ids.jar jade.Boot -container-name server6 -jade_domain_df_autocleanup
true -backupmain -host 192.168.1.1 -name IDS -services
jade.core.replication.MainReplicationService\;jade.core.replication.AddressNoti
ficationService
```

No server7:

```
#!/bin/bash
java -cp ids.jar jade.Boot -container-name server7 -jade_domain_df_autocleanup
true -backupmain -host 192.168.1.1 -name IDS -services
jade.core.replication.MainReplicationService\;jade.core.replication.AddressNoti
ficationService
```

A função destes *scripts* é auxiliar a execução do *framework* JADE na máquina. Para a execução do JADE, é informado no parêmetro *-cp* o nome do arquivo jar que contém o projeto. No parâmetro *-container-name* é informado o nome do *host.* No parâmetro *-host* é informada o nome do host que contém no contêiner principal do JADE. No parâmetro *-name* é informado o nome da aplicação e no parâmetro *-services* é informado quais os serviços do JADE serão utilizados.

Nos próximos quatro tópicos, são mostrados quatro cenários para mostrar como o protocolo de colaboração atua no sistema. Três destes cenários mostram as possíveis situações de colaboração entre agentes de monitoramento e análise. O último cenário mostra o SDI original funcionando sem este recurso.

4.7.2 Cenário1 – Com colaboração entre agentes

O agente de monitoramento e análise que roda no Server7 identifica no *log* do Snort as três linhas abaixo:

```
[**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/07-20:15:26.013677 200.233.17.190:51959 -> 192.168.1.5:1
```

O agente que realizou a leitura verifica em sua base de conhecimento se possui um prévio conhecimento sobre este tipo de intrusão. Neste cenário este agente não possui tal conhecimento. Então um mensagem será enviada pelo Server7 aos outros agentes de monitoramento e análise (no caso, o agente que está rodando no Server5). A mensagem será:

```
SERVER7;07/07/2010;20:15:27;INTRUSION_DETECTED;1:228:7;07/07/2010;20:15:26;200.
233.17.190:51959;192.168.1.5:1
```

O agente do Server5 recebe esta mensagem e irá verificar em sua base de conhecimento se uma intrusão deste tipo já foi anteriormente detectada. Neste cenário, este agente não possui conhecimento sobre este tipo de intrusão. Então o agente responderá ao agente solicitante a seguinte mensagem:

```
SERVER5;07/07/2010;20:15:28;INTRUSION_NOT_FOUND;1:228:7;07/07/2010;20:15:26;200
.233.17.190:51959;192.168.1.5:1
```

Quando o agente que detectou a intrusão recebe esta mensagem, entende que o outro agente também não tem conhecimento sobre um evento deste tipo. Então o agente do Server7 irá adicionar este conhecimento em sua base e irá acionar o agente de reação. Cada agente de reação, neste caso são dois, adicionará uma regra no *IpTables* para bloquear pacotes que venham do endereço do evento e o sistema emite um alerta ao administrador. A Figura 15 mostra as regras adicionadas.

```
Last login: Wed Jun 30 20:45:29 2010
root@server7:~# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  200.233.17.190       0.0.0.0/0
DROP       all  --  200.233.17.190       0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:22
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:465
DROP       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp flags:0x17/0x02

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  200.233.17.190       0.0.0.0/0
DROP       all  --  200.233.17.190       0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@server7:~#
```

Figura 15: Regras do *firewall*

4.7.3 Cenário2 – Com colaboração entre agentes

O agente de monitoramento e análise que roda no Server7 identifica no *log* do Snort as três linhas abaixo:

```
[**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/07-20:15:26.013677 200.233.17.190:51959 -> 192.168.1.5:1
```

O agente que detectou essa leitura verifica em sua base de conhecimento se possui um prévio conhecimento sobre esta intrusão. Neste cenário este agente possui tal conhecimento. Então ele mostrará na tela do console do sistema operacional uma mensagem informando que um mesmo ataque, que anteriormente foi bloqueado, está sendo detectado novamente. O agente não adiciona esse conhecimento a sua base e não adiciona regras no *firewall*, evitando-se assim redundância de informações. Também, nenhum alerta padrão do SDID é enviado ao seu administrador.

4.7.4 Cenário3 – Com colaboração entre agentes

O Snort identifica um ataque e o agente que detectou não tem conhecimento sobre o evento, mas o outro agente possui. O agente de monitoramento e análise que roda no Server5 identifica no *log* do Snort as 3 linhas abaixo:

```
[**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/07-20:15:26.013677 200.233.17.190:51959 -> 192.168.1.5:1
```

O agente que detectou essa leitura verifica em sua base de conhecimento se possui um prévio conhecimento sobre esta intrusão. Neste cenário este agente não possui tal conhecimento. Então a mensagem abaixo será enviada pelo agente do Server5 aos outros agentes de monitoramento e análise (no caso Server7).

```
SERVER5;07/07/2010;20:15:27;INTRUSION_DETECTED;1:2228:7;07/07/2010;20:15:26;200
.233.17.190:51959;192.168.1.5:1
```

O agente de monitoramento e análise do Server7 recebe esta mensagem e irá verificar em sua base de conhecimento se uma intrusão deste tipo já foi anteriormente detectada. Neste cenário, este agente agente possui conhecimento deste tipo de intrusão. Então o agente responderá com a mensagem abaixo:

```
SERVER7;07/07/2010;20:15:28;INTRUSION_ALREADY_FOUND;1:228:7;07/07/2010;20:15:26
;200.233.17.190:51959;192.168.1.5:1
```

Quando o agente que detectou a intrusão recebe esta mensagem do agente do Server7, entende que o outro agente tem conhecimento deste evento e que fará a tratativa da intrusão. A tratativa neste caso será mostrado uma mensagem na tela do console do sistema operacional informando que um mesmo ataque, que foi anteriormente bloqueado, está sendo detectado novamente. Nenhum conhecimento será adicionado a sua base nem será adicionado regras no *firewall*. Também, nenhum alerta padrão do SDID é enviado ao seu administrador.

4.7.5 Cenário4 – Sem colaboração entre agentes

O agente de monitoramento e análise que roda no Server7 identifica no *log* do Snort as três linhas abaixo:

```
[**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/07-20:15:26.013677 200.233.17.190:51959 -> 192.168.1.5:1
```

Então o agente faz a tratativa da intrusão incluindo no *IpTables* a regra para bloquear este ataque. Caso esta intrusão tenha sido anteriormente detectada, a regra seria novamente incluída no *IpTables* e o sistema emite um novo alerta ao administrador. A replicação das regras é mostrada na Figura 16.

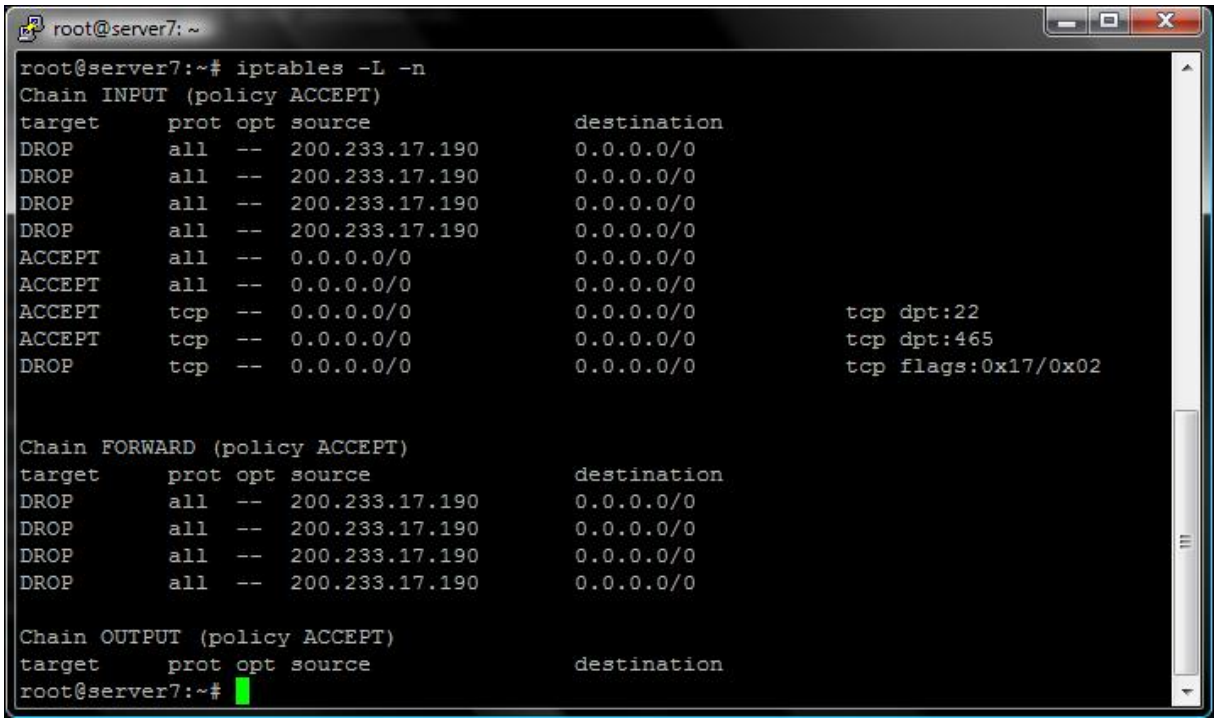


Figura 16: Replicação dos regras no *firewall*

4.8 Análise dos experimentos

Analisando os cenários propostos anteriormente, observamos que a colaboração realizada pelos agentes de monitoramento e análise trouxe benefícios ao sistema. Esses benefícios podem ser interpretados como a adição de inteligência ao sistema.

No cenário 4, sem o recurso de colaboração, o SDID adicionava no *firewall* do sistema operacional regras repetidas, gerando uma redundância desnecessária. Além disso, o SDID emite desnecessariamente mensagens de alerta ao administrador do sistema.

Nos cenários onde há a colaboração entre os agentes de monitoramento e análise, o sistema mostrou-se mais inteligente. Cumpriu sua função sem fazer a redundância de informações, pois não adicionou regras repetidas ao *firewall* nem emitiu alertas desnecessários ao administrador. Ainda, adicionou mais um recurso ao sistema, ou seja, se mais que uma intrusão de mesmo tipo for detectada pelo SDID, é bem provável que a rede esteja desprotegida de ataques, pois há falhas de segurança no *firewall*.

5 CONCLUSÕES E PERSPECTIVAS FUTURAS

O principal objetivo para se usar um sistema de detecção de intrusão é para possuir o conhecimento de que está em andamento ou se ocorreu uma tentativa de invasão. Essa informação é crucial quando se quer proteger a integridade, privacidade e autenticidade de dados em uma rede. Detecção de intrusão é uma tentativa de monitorar estações ou fluxos de rede com o intuito de descobrir ações de intrusos. Mais especificamente, um SDI tenta detectar ataques ou usos impróprios e alertar o responsável pela rede do acontecimento.

Os SDI, inicialmente criados de forma centralizada, possuíam um ponto central de gargalo que facilitava um ataque. Atualmente, formas alternativas para contornar esse problema utilizam SDI baseados em sistemas multiagentes (SMA), que possuem uma arquitetura distribuída, não existindo um ponto central vulnerável dentro do sistema.

Este trabalho adicionou ao SDID de Junior (JUNIOR, 09) o recurso de colaboração entre os agentes de monitoramento e análise e que estes agentes possuam uma base de conhecimento sobre os eventos ocorridos. Agora, estes agentes trocam informações sobre os eventos detectados e são capazes de tomar decisões sobre o tratamento da intrusão.

Com isso, acrescentou inteligência ao sistema, evitando que ações de mesmo efeito fossem realizadas mais que uma vez. Possibilitou também que o administrador da rede identifique saiba se há falhas em seu sistema *firewall,* através da detecção repetida de um ataque de mesmo tipo e com a mesma origem.

Podemos dizer que o processo de colaboração não trouxe apenas benefícios, mas também adicionou ao sistema mais processos que também estão sujeitos a falhas. Acrescentou a quantidade de dados que circulam na rede e um tempo de processamento de mensagens e base de conhecimentos que podem ser preciosos durante uma tentativa de intrusão.

O SDI colaborativo tem muito a ser implementado e muitos aspectos existentes podem ser melhorados. Um exemplo seria a persistência da base de conhecimento dos agentes de análise e monitoramento em um banco de dados faria com que, na reinicialização do SDI, os agentes não perdessem o conhecimento já adquirido. Outra implementação a ser feita para melhorar o sistema seria estender a colaboração realizada nos agentes de monitoramento e análise para os outros agentes do sistema como por exemplo, o agente de reação. Desta forma,

o recurso de colaboração nestes agentes evitaria a inserção de mais regras duplicadas no *firewall*.

6 REFERÊNCIAS

[ASAKA, 99] ASAKA, M.; TAGUCI, A.; GOTO, S. **The implementation of ida: An intrusion detection agent system.** 11th FIRST Conference-Brisbane-Australia, 1999.

[BACE, 00] BACE, G. Rebecca. **Intrusion Detection**. Macmillan Technical Publishing, 2000.

[BACE, 01] BACE, G. Rebecca; MELL, Peter. **Intrusion Detection Systems. Califórnia**: National Institute of Standard and Technology, 2001.

[BENATTOU, 05] BENATTOU, M.; TAMINE, K. **Intelligent Agents for Distributed Intrusion Detection System.** Transactions on Engineering, Computing and Technology, V6, 2005.

[GIRARDI, 02] Ferreira, Steferson L. C.; Girardi, Rosario. **Arquitetura de Software** Baseadas em Agentes: Do Nível Global ao Detalhado. Revista Eletrônica de IniciaçãoCientífica. Universidade Federal do Maranhão, 2002.

[GORODETSKI, 03] Gorodetski, Vladimir; Kotenko, Igor; Karsaev, Oleg. **Multiagent technologies for computer network security: Attack simulation, intrusion detection and intrusion detection learning.** International Journal of Computer Systems Science & 88 Engineering, 2003.

[IPTABLES, 10] Pablo Neira Ayuso. **iptables**, 2010. Disponível em: http://www.netfilter.org/projects/iptables/index.html. Acesso em: jun 2010.

[JADE, 08] Telecom Itália. **Java Agent Development Framework**, 2008. Disponível em: < http://jade.cselt.it/>.Acesso em: nov. 2008.

[JAVA, 95] Oracle Corporation. **Java**, 1995. Disponível em: < http://www.sun.com/java/>. Acesso em: mai. 2010.

[JUNIOR, 09] JÚNIOR, Mário R.K. **Utilização da abordagem multiagentes para desenvolvimento de sistemas de detecção de intrusão**. Trabalho de Conclusão de Curso da Universidade de Caxias do Sul, 2009.

[KQML, 10] Labrou, Yannis; Finin, Tim. KQML Knowledge Query and Manipulation Language. Disponível em: http://www.cs.umbc.edu/research/kqml/. Acesso em: jun 2010.

[MEIRA, 01] MEIRA, Lara. **A utilização de Sistemas multiagentes para a Detecção de Intrusão em Redes**. 2001. Trabalho de Conclusão de Curso da Universidade de Caxias do Sul.

[NESSUS, 09] Tenable Network Security. **Nessus**, 2009. Disponível em: <http://www.tenablesecurity.com/nessus/>. Acesso em: nov. 2009.

[PORRAS, 92] PORRAS, P. Stat. **A state transotion analysis tool for intrusion detection**. Tese de metrado, University of California at Santa Barbara, 1992.

[RIBEIRO, 02] Ribeiro, Alexandre Moretto. **Sistema de Detecção de Intrusão Distribuído Baseado em Sistemas multiagentes.** Caxias do Sul: Universidade de Caxias do Sul, 2002.

[SIQUEIRA, 06] L. and Abdelouahab, Z. **A Fault Tolerance Mechanism for Network Intrusion Detection System based on Intelligent Agents (NIDIA). In Proceedings of the the Fourth IEEE Workshop on Software Technologies For Future Embedded and Ubiquitous Systems, and the Second international Workshop on Collaborative Computing, integration, and Assurance.** Seus-Wccia, 2006.

[SNORT, 08] Sourcefire Inc. **Snort,** 2008.Disponível em: < http://snort.org/>. Acesso em: mar 2010.

[STALLINGS, 08] STALLINGS, William. **Criptografia e segurança de redes**. Pearson Prentice Hall, 2008.

[YE, 08] YE, Dayong; BAI, Quan; ZHANG Minjie; YE, Zhen. **P2P Distributed Intrusion Detections by Using Mobile Agents**. Seventh IEEE/ACIS International Conference on Computer and Information Science, 2008.