10th Brazilian Congress on Computational Intelligence (CBIC'2011), November 8 to 11, 2011, Fortaleza, Ceará Brazil

UM SISTEMA AUTONÔMICO MULTIAGENTE BASEADO EM ONTOLOGIAS PARA SEGURANÇA EM AMBIENTES COMPUTACIONAIS

Wendell C. Veras[1], José Alfredo F. Costa[2], Fábio A. P. de Paiva[3], Leandro A. Pasa[4], André Soffiatti[5], José L. M. Soares[6]

UFRN – Universidade Federal do Rio Grande do Norte[1]
UFRN – Universidade Federal do Rio Grande do Norte[2]
IFRN – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte[3]
UTFPR – Universidade Tecnológica Federal do Paraná[4]
UFRN – Universidade Federal do Rio Grande do Norte[5]
FIP – Faculdades Integradas de Patos[6]

wendellweb@gmail.com[1], jafcosta@gmail.com[2], fabio.procopio@ifrn.edu.br[3], pasaleandro@yahoo.com.br[4], soffiatti.andre@gmail.com[5], leotabira@gmail.com[6]

Abstract – Computational security has become a worry among corporations. Indeed, the use of specifics tools, procedures and policies that cover requirements to keep computational and IT infrastructure protected of malicious agents has been proved necessary. In order to address the problem of protecting computational resources, in this work we propose an autonomic model called AutoCore, constituted by a Muilt-Agent system, an Intelligent Interface (CoreEditor) and a formal ontologies (CoreSec). This proposal has been implemented, tested and validated in real scenarios to assist the safety activities and to minimize the administrator complexity.

Keywords – Information of Security, Autonomic Computing, Intelligent Agents, Ontologies.

1 Introdução

Os diversos ambientes de atuação humana, sejam eles militares, corporativos ou acadêmicos, necessitam de meios transparentes para planejar e gerenciar problemas e informações relacionados à segurança computacional, especificamente à segurança da informação. Os problemas com a segurança da informação passaram a ser uma questão estratégica para os negócios. A IBM (*International Business Machines*) apresentou um manifesto [11] alertando a indústria de Tecnologia da Informação (TI) para uma nova crise. De acordo com este, os sistemas computacionais têm evoluído consideravelmente desde que surgiram, ao mesmo tempo em que tornaram-se mais complexos de serem gerenciados. Alguns motivos que contribuem para esse fato são a segurança de sistemas de missão crítica, a necessidade crescente de interconectividade, a integração de tecnologias heterogêneas e a alocação satisfatória dos recursos computacionais [14].

Ainda segundo esse manifesto, essa complexidade é incompatível para a grande maioria dos profissionais de TI, o que gera custos elevadíssimos e prejudica a qualidade dos serviços oferecidos [12]. Há, portanto, um aumento significativo na complexidade de projetar e planejar segurança, necessitando que meios de manipulação, interpretação e processamento de informações sejam adotados.

Por outro lado, o conjunto de tecnologias que formam a Web Semântica, juntamente com sistemas baseados em Agentes Inteligentes e Computação Autonômica, permitem que sistemas computacionais manipulem, interpretem e processem essas informações através de ontologias, tendo sido aplicadas em diversos contextos que englobam compartilhamento, organização e uso de informação semântica. Ontologias têm contribuído para facilitar a comunicação e o processamento da informação semântica, tanto entre sistemas baseados em agentes quanto entre seres humanos, promovendo assim, interoperabilidade entre sistemas ao representarem dados compartilhados por diversas aplicações [21]. Com uma base de conhecimento a respeito de aspectos relacionados à segurança, as corporações poderão desenvolver e implantar mais facilmente mecanismos de proteção, correção e prevenção de acordo com os seus requisitos de segurança, requisitos estes expressos através de Acordos de Níveis de Serviços (SLAs – *Service Level Agreement*) ou de políticas de segurança exigidos para que seus serviços estejam sempre disponíveis, íntegros e confiáveis.

Nesse contexto, foi proposta uma solução baseada no sistema nervoso autonômico humano, denominada de Computação Autonômica [11]. O sistema nervoso autonômico é responsável por, dentre outras coisas, manter o ritmo do batimento cardíaco, controlar a temperatura do corpo e a pressão sanguínea, de forma independente, ou seja, sem que uma intervenção consciente seja realizada. Portanto, assim como o sistema nervoso autonômico, um sistema computacional autonômico deverá realizar tarefas relacionadas à administração de forma automática e transparente, mantendo a sua integridade, havendo assim, pouca ou nenhuma intervenção humana.

Neste trabalho apresentamos o *AutoCore*, um sistema multiagente baseado em Ontologias para apoiar as atividades realizadas pelos responsáveis pela segurança computacional em ambientes corporativos. Sua finalidade é fornecer um modelo formal que reflita os aspectos de segurança referentes a esse tipo de ambiente, permitindo uma comunicação segura entre os diversos elementos do sistema autonômico e possibilitando um processo de inferência acerca dos diversos eventos que possam colocar a segurança dos sistemas em risco.

O restante deste artigo está organizado da seguinte forma: na Seção 2 são apresentados os trabalhos relacionados. Nas Seções 3 e 4 são apresentados um referencial teórico necessário para o entendimento geral da teoria acerca do trabalho proposto: Computação Autonômica e Ontologias respectivamente. Já na Seção 5, é detalhada a proposta, sua arquitetura, os componentes que a formam e suas funcionalidades. Na Seção 6, são apresentados e discutidos os resultados obtidos nos experimentos realizados com o uso do *AutoCore*. Por fim, as conclusões e os trabalhos futuros são delineados na Seção 7.

2 Trabalhos Relacionados

Nos últimos anos, vários esforços têm sido despendidos para obterem-se modelos autonômicos que permitam auxiliar a complexidade de se proteger sistemas computacionais e auxiliar a segurança em TI, com objetivo de fornecer diferentes níveis de proteção aos ativos de uma corporação. Embora com diferentes focos e níveis de detalhamento, alguns trabalhos da literatura tratam de sistemas autonômicos no domínio de segurança a fim de gerenciar, avaliar e especificar segurança, seja da informação ou não, nos mais diversos ambientes. Dentre esses trabalhos que ajudaram no amadurecimento e suporte no desenvolvimento da proposta aqui descrita, destacam-se:

O trabalho de [1] propõe o uso de uma ontologia para gestão de segurança da informação em ambientes computacionais autonômicos denominada OSACA. Apesar de meritosa, a OSACA é um trabalho prematuro. A ontologia não foi implementada utilizando linguagens da Web Semântica, como *Resource Description Framework* (RDF) [18] ou *Web Ontology Language* (OWL) [16], não possui axiomas nem restrições em DL (Lógica de Descrição) que são os padrões de facto estabelecidos. O trabalho não apresenta resultados, não garantindo a sua utilidade prática.

No trabalho de [15] foi desenvolvida uma ontologia denominada *OntoSec*, representando de maneira padronizada as informações sobre incidentes de segurança, facilitando compartilhamento e reuso das informações, gerenciamento e geração de conhecimento sobre incidentes e uma ferramenta de auxílio aos responsáveis pela segurança, os CSO - *Chief Security Officers*, onde são realizadas consultas para obtenção de respostas a possíveis incidentes. A principal diferença entre a *OntoSec* e o *AutoCore*, proposto neste trabalho é que a primeira é uma ontologia de aplicação e não do domínio de segurança como a *CoreSec* do *AutoCore*, e o sistema desenvolvido para auxiliar os CSO não é baseado em agentes, não possuindo autonomia para realizar correções preventivas nem analisar ataques em tempo real.

Já o *AutoCore* é um sistema autonômico e possui características superiores à *OntoSec* e seu sistema de auxílio, sendo composto por:

- uma ontologia de domínio com conceitos mais genéricos e abstratos do domínio de segurança servindo como base para construção de ontologias de domínio específicos de segurança denominada *CoreSec* e por

- um sistema chamado *CoreEditor* que possui mais funcionalidades que o do trabalho comparado. Este sistema baseia-se em agentes inteligentes que tomam decisões independentes, acerca de otimização, configuração e correção de um determinado recurso monitorado por eles.

Pode-se citar ainda o sistema *eAutomation* [19] desenvolvido focando um sistema de correlação de mesmo nome, mas não trata especificamente a questão da segurança da informação, além de descrever um conjunto muito reduzido de classes na ontologia que propõe. A abordagem aqui apresentada propõe melhorias significativas em relação aos trabalhos citados nesta seção. Nas próximas seções são apresentadas as teorias (Computação Autonômica e Ontologias) necessárias ao entendimento da proposta descrita.

3 Computação Autonômica

Em 2001, a IBM [11] apresentou ao mercado uma proposta de solução ao problema da excessiva complexidade de software corporativo e a denominou de Computação Autonômica. A idéia segue uma abordagem comum dentro das diversas ramificações da engenharia que é a de buscar inspiração no sistema nervoso autonômico humano, o qual é capaz de manter as funções vitais a fim de compreender e resolver os mais diversos tipos de problemas sem qualquer ou pouca iniciativa, participação ou intervenção direta do ser humano.

Um sistema de computação autonômica pode ser definido como sendo aquele capaz de se gerenciar de acordo com objetivos de alto nível definidos pelo administrador [14]. Uma das metas principais desta computação é a capacidade de autogerenciamento, livrando o administrador de sistemas da preocupação com detalhes operacionais do sistema e possibilitando ao usuário empregar as máquinas com melhor desempenho durante todo o tempo.

Uma vez que os sistemas atualmente têm um grande dinamismo, com mudanças constantes, é fundamental que estes possam se adaptar às mudanças decorrentes de novas demandas, políticas de negócio e restrições de segurança à medida em que forem ocorrendo. Para atingir este objetivo e ser considerado um sistema de computação autonômica é necessário atender a quatro serviços básicos de acordo com [17]:

- Auto-Configuração (Self-Configuring): capacidade do sistema de se configurar em tempo de execução, com quase ou nenhuma intervenção humana. Dessa forma, o ambiente de TI pode se acomodar a novas configurações dinamicamente seja pela inclusão, alteração ou remoção de componentes;
- Auto-Otimização (Self-Optimizing): capacidade do sistema em otimizar a alocação dos recursos existentes, de tal forma que as necessidades dos usuários sejam atendidas sem comprometimento dos demais recursos. A idéia é que, por exemplo, questões relacionadas com desempenho e qualidade de serviço possam ser tratadas de acordo com políticas de alto nível e de forma transparente;
- Auto-Cura (Self-Healing): habilidade do sistema de identificar falhas e executar ações corretivas, sem que para isso seja necessário paralisar o funcionamento dos serviços. Sendo assim, o sistema poderá, após a ocorrência de uma falha, voltar a um estado estável sem a necessidade de intervenção do administrador;
- Auto-Proteção (Self-Protecting): habilidade do sistema em detectar comportamentos maliciosos ou hostis e tomar decisões eficientes e eficazes de forma transparente sobre qual é a ação mais adequada para impedir ou minimizar um ataque.

Para atingir o objetivo do autogerenciamento é necessário, portanto, que estes quatros requisitos funcionem de forma sinérgica. Por exemplo, durante o processo de inserção automática de um novo elemento no ambiente, a parte responsável pela auto-configuração deverá configurá-lo e comunicar à parte responsável pela auto-proteção sobre este evento a fim de que sejam tomadas as medidas necessárias para resguardar o nível de segurança.

4 Ontologias

Desde o século XVII, o termo "ontologia" tem sido utilizado para denominar a disciplina de metafísica geral, dentro da tradição da "primeira Filosofia" de Aristóteles, como sendo a ciência do ser no papel de ser. É, muitas vezes, encarada como um complemento à idéia de Epistemologia (ciência do conhecimento) [6].

Diversas definições têm surgido a fim de descrever o que é uma ontologia dentro do ramo de informática. A mais conhecida é "uma especificação formal e explícita de uma conceitualização compartilhada" [9], onde:

- formal implica em ser declarativamente definida, portanto, compreensível para agentes e sistemas;

- explícita significa que os elementos e suas restrições estão claramente definidos;

- conceitualização trata de um modelo abstrato de uma área de conhecimento ou de um universo limitado de discurso;

- compartilhada, indica um conhecimento consensual, seja uma terminologia comum da área modelada, ou acordada entre os desenvolvedores dos agentes que se comunicam.

Sendo assim, ontologias, em um nível de abstração mais alto, estabelecem uma terminologia comum e não-ambígua para o domínio em questão.

Para [8], ontologias são um artefato computacional composto de um vocabulário de conceitos, suas definições e suas possíveis propriedades, um modelo gráfico mostrando todas as possíveis relações entre os conceitos e um conjunto de axiomas formais que restringem a interpretação dos conceitos e relações, representando de maneira clara e não ambígua o conhecimento do domínio. Inúmeras vantagens têm sido apresentadas na literatura para a adoção de ontologias. Dentre elas, ressaltam-se as seguintes [7]:

- Oportunidade aos desenvolvedores reusarem conhecimento, mesmo com adaptações e extensões. Isso explica-se pelo fato de que a construção de bases de conhecimento é uma das tarefas mais caras, complexas e demoradas de um sistema especialista e/ou agentes. Portanto, reusar ontologias promove um ganho significativo em termos de esforços e de investimentos;

- A grande disponibilidade de "ontologias de prateleira", prontas para uso, reuso e comunicação entre agentes, podendo estas serem estendidas e complementadas com conceitos de domínios específicos;

- O acesso on-line a servidores de ontologias, capazes de armazenar milhares de classes e instâncias, servindo a empresas ou grupos de pesquisa, funcionando como ferramentas para manter a integridade do conhecimento compartilhado entre elas e garantindo um vocabulário uniforme;

Recentemente o uso de ontologias tem se popularizado através de diversas outras subáreas da Ciência da Computação, tais como: Engenharia de Software, Banco de Dados e Sistemas de Informação, motivados pela Web Semântica, que é uma consequência direta do uso de ontologias [20].

5 Proposta

Neste trabalho apresentamos o *AutoCore*, um sistema multiagente autonômico baseado em ontologias tendo como objetivo principal prover um sistema flexível, adaptativo aos ambientes ao qual está inserido e atuando em tempo real para apoiar a segurança computacional.

Considera-se que a segurança computacional de uma corporação será mais eficiente se esta for baseada em modelos autonômicos, tal como o *AutoCore*. O sistema aqui apresentado pretende ser a base para possíveis soluções de segurança e auxiliar os responsáveis pela Gestão de Segurança da Informação em suas atividades. A arquitetura, componentes e funcionalidades do sistema proposto serão apresentadas nas subseções a seguir.

5.1 *AutoCore*

O *AutoCore* foi implementado utilizando o JADE[1] – *Java Agent Development Framework*. Os agentes foram organizados em *containers* de agentes, onde cada *container* é composto por um conjunto de agentes modelados de acordo com os serviços básicos inerentes a um sistema autonômico (Auto-Otimização, Cura, Configuração e Proteção) explicados na Seção 3 deste artigo. O *AutoCore* faz uso de uma ontologia como base de conhecimento denominada *CoreSec*, sendo uma ontologia de domínio para segurança da informação com conceitos de mais alto nível com intuito de auxiliar a Governança de TI, definindo assim, uma base de informações comum, um aumento na capacidade de tratamento e utilização da informação e uma visão explicita entre os envolvidos nas atividades de segurança computacional.

Por ser uma ontologia de domínio, a *CoreSec* possibilita a criação, de sub-ontologias de domínios mais específicos como ontologias de ataques, vulnerabilidades e incidentes de segurança computacional.

Na Figura 1, observa-se a arquitetura do *AutoCore* e os componentes que o formam. O sistema é composto por um Agente Gerenciador, responsável por gerenciar as colônias de agentes verificadores, avaliadores, acionadores e visualizadores, definindo os objetivos e as tarefas que deverão ser executadas por cada uma delas. O Agente Gerenciador possibilita a Auto-Configuração, visto propiciar às colônias, em tempo de execução, as regras a serem seguidas.

Os Agentes Verificadores ficam em constante análise dos recursos a que estão ligados, sendo responsáveis pela coleta e análise do tráfego de rede e possuindo o objetivo de detectar comportamentos hostis e classificá-los. Para cada categoria de ataque existe um conjunto de agentes especializados, possibilitando a detecção de possíveis atividades maliciosas. No momento em que são instanciados, os Agentes Verificadores consultam uma sub-ontologia, criada a partir da *CoreSec*, e mantêm em memória interna todas as configurações atuais para detecção de

anomalias associadas aos recursos. Caso seja detectada alguma alteração não configurada, uma nova inserção é realizada à ontologia através do *CoreEditor*. Os Agentes Verificadores têm a tarefa de enviar aos Agentes Avaliadores as informações dos *status* obtidos. Juntos caracterizam a Auto-Proteção, pois enquanto um agente detecta um mau funcionamento, o outro toma decisões eficientes e eficazes de como solucionar determinado problema encontrado.

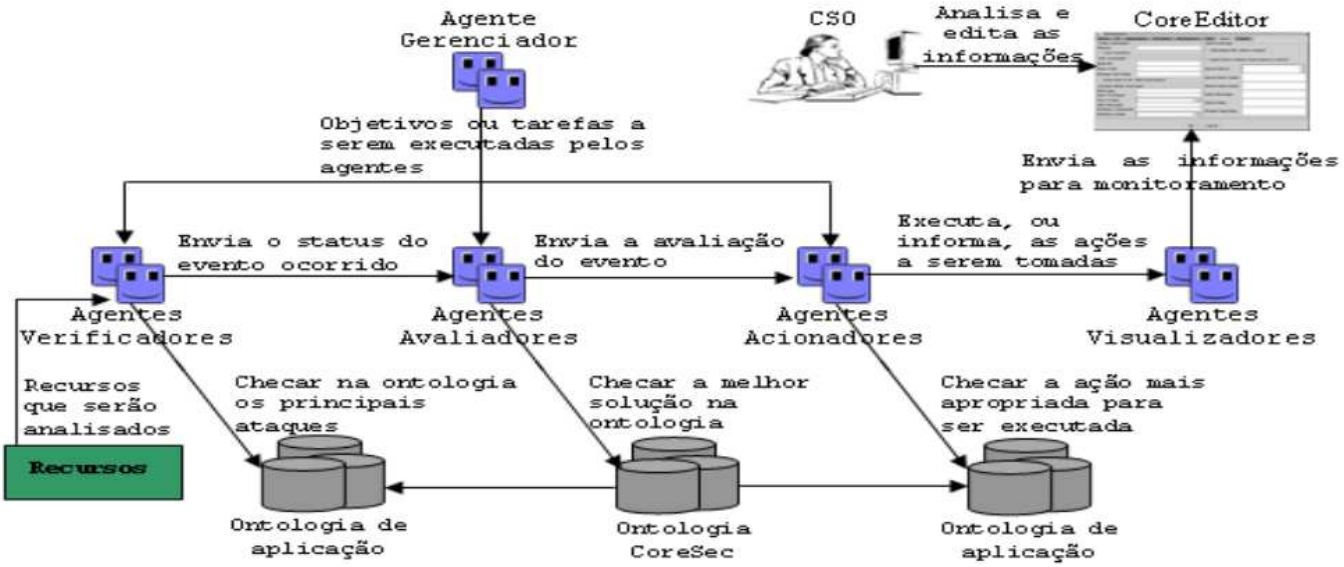

Figura 1 – Arquitetura do AutoCore.

Após receberem as informações do *status* do ambiente, os Agentes Avaliadores consultam a *CoreSec* para descobrir as melhores medidas a serem tomadas, enviando as novas informações aos Agentes Acionadores. As consultas realizadas pelos Agentes Avaliadores possibilitam uma segunda caracterização, a Auto-Otimização, visto possibilitarem a otimização dos recursos sem comprometerem os demais recursos. Os Agentes Avaliadores constroem diversas soluções baseadas em métricas para a correta resolução das vulnerabilidades detectadas, enviando aos Agentes Acionadores alguns indicadores como: Impacto nos Negócios, Tempo de Resolução, Eficácia e Eficiência, Qualidade e Custo das Resoluções.

Os Agentes Acionadores são os responsáveis em receber as avaliações e com base nestas informações executarem as melhores ações a serem tomadas. Toda ação realizada, mesmo que não seja de execução automática de alguma medida de prevenção, é enviada aos Agentes Visualizadores, e estes disponibilizam para o CSO através do *CoreEditor*. Caso seja necessária uma intervenção na decisão tomada pelos Agentes Acionadores, o CSO utiliza o *CoreEditor* para tal ação. Os Agentes Acionadores realizam também Auto-Cura, visto que após a identificação das falhas, executam ações corretivas e preventivas sem paralisarem o funcionamento dos serviços, possibilitando retornar a um estado estável sem intervenções humanas. Esses agentes também consultam uma sub-ontologia, criada a partir da *CoreSec*, sempre antes de executar uma ação.

Os Agentes Visualizadores são responsáveis por exibir gráficos, relatórios informativos e de execução através do *CoreEditor*, de maneira a manter os responsáveis pela segurança da informação cientes a respeito da situação atual de todo o sistema, inclusive, facilitando as possíveis intervenções nas resoluções em momentos críticos de riscos aos sistemas.

Para ajudar na monitoração e alimentação da base de conhecimento (*CoreSec*) utilizada pelo *AutoCore* foi desenvolvida uma aplicação denominada *CoreEditor* para uso exclusivo do CSO, com intuito de auxiliar na transmissão, geração e distribuição do conhecimento, manipulação, avaliação e utilização da *CoreSec*. Um aspecto a ser considerado é que o *CoreEditor* não é apenas uma ferramenta para manipulação da *CoreSec*, mas também, um *framework* para criação e edição de ontologias, desenvolvido utilizando o *framework* Jena, produzido por Brian McBride da *Hewlett-Packard* para criação e manipulação de grafos RDF [2]. O *CoreEditor* é uma aplicação exclusiva para manipulação da base de conhecimento utilizada quando for necessário uma intervenção humana.

Entre as principais características e funcionalidades do *CoreEditor* estão: utilizar a linguagem OWL para criação de ontologias (Conceitos, Propriedades e Indivíduos) e a máquina de inferência *Pellet* para geração de hipóteses a partir das informações nas bases de conhecimento; manipular e gerar código e Java para aplicações da ontologia desenvolvida, gerar o seu *OntoDoc*, similar ao javadoc do Java; gerar diagramas de classes UML da ontologia desenvolvida e interfaces de consultas utilizando a linguagem SPARQL[2].

6 Experimentos e Resultados

Os experimentos foram realizados para ataques de negação de serviço, DoS (*Denial of Service*) SYN *Flooding*, caracterizados pelo envio de múltiplas solicitações de abertura de conexão a um mesmo *host* em uma mesma porta, em um curto período de tempo. O objetivo de um ataque DoS é tornar a rede indisponível [13]. Esse tipo de ameaça é facilmente identificado, porém, muito difícil de evitar.

Desenvolvemos um simulador para esse tipo de ataque, no qual configuramos a quantidade de requisições ao servidor, a porta atacada e o IP atacado. Observa-se na Figura 2 o simulador em funcionamento.

Figura 2 – Simulador de Ataques DoS.

Nos campos IP e Port são inseridos o endereço IP e a porta alvo do ataque DoS. O campo *Quantity* informa ao simulador quantas requisições deverão ser realizadas. Ao clicar no botão *Execute*, o simulador então inicia o ataque DoS de acordo com os dados inseridos. O botão *Stop* tem a função de finalizar as requisições antes de alcançar a quantidade configurada. O campo *History* apresenta um histórico das requisições realizadas e o botão *ClearAll* apaga as informações desse histórico. A tela inicial do *AutoCore* está ilustrada na Figura 3.

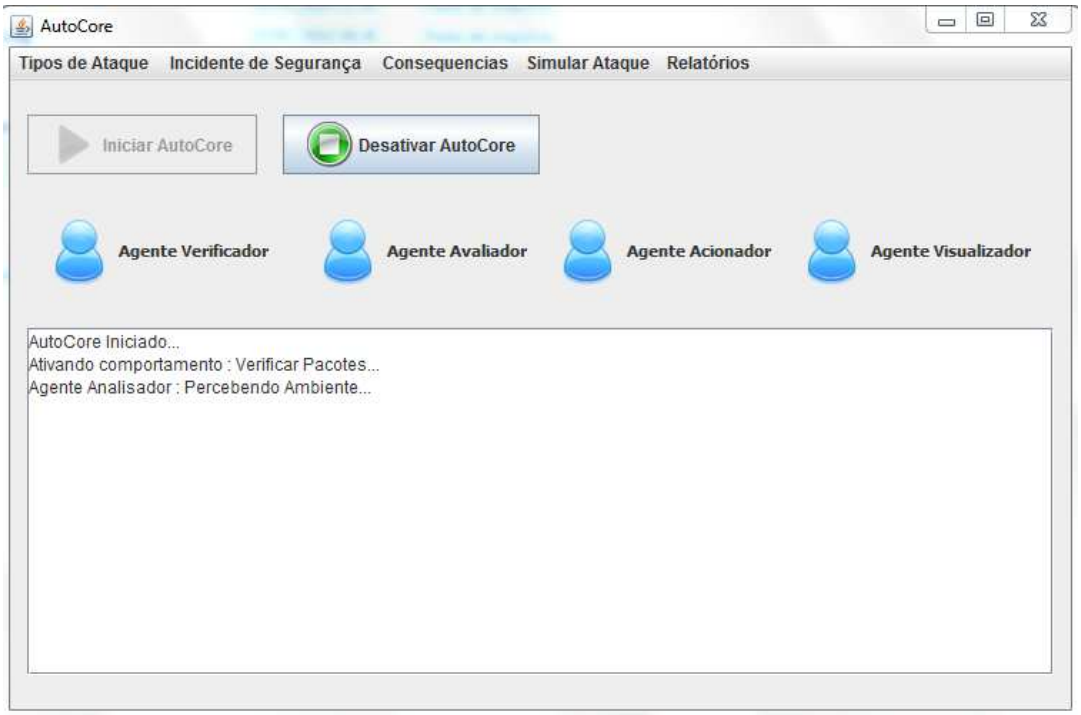

Figura 3 – Tela Inicial do *AutoCore*.

Pode-se visualizar no campo de informações do *AutoCore* o monitoramento dos pacotes transmitidos pela rede com objetivo de detectar anomalias. Esse monitoramento do ambiente é realizado pelo agente Analisador (Verificador). No momento em que o *AutoCore* é inicializado são criados os agentes, conforme Figura 4. Para explicar esta figura, utilizou-se a ferramenta de monitoramento do processo de comunicação dos pacotes (*sniffer*) do JADE.

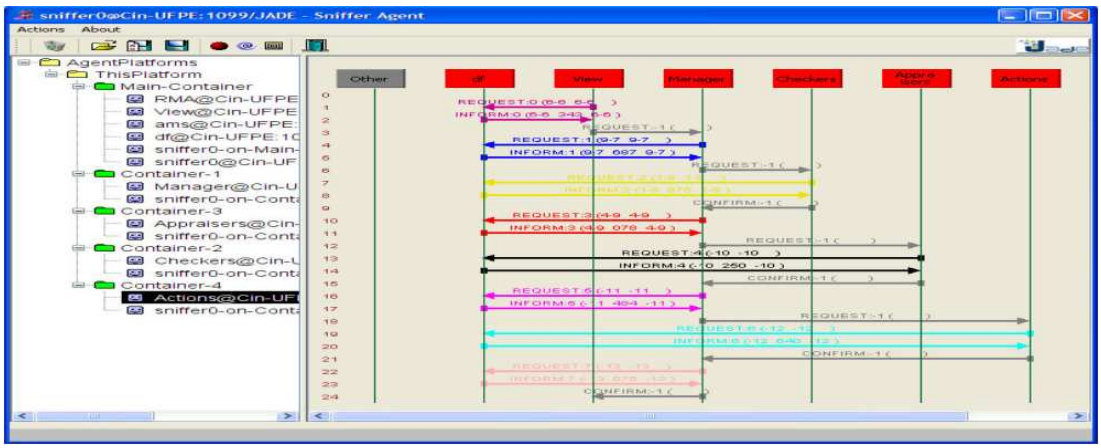

Figura 4 – Inicialização e Criação dos Agentes.

O Agente Visualizador (*Viewer*) processa o evento de requisição de inicialização do sistema, visto ser o responsável por monitorar os eventos da interface entre o sistema e o usuário. Logo, ele envia ao Agente Gerenciador (*Manager*) a solicitação de *start* do sistema e este por sua vez, inicializa os Agentes Verificadores (*Checkers*), Avaliadores (*Appraisers*) e Acionadores (*Actions*). É possível notar que antes da conexão com cada agente, é realizada uma consulta ao Facilitador de Diretórios (DF – *Directory Facilitator*), sendo este responsável por informar o identificador (ID) dos agentes, possibilitando, portanto, a comunicação e troca de mensagens. As mensagens são codificadas utilizando o protocolo de interação *contract-net* [FIPA, 2002], e usam como vocabulário a ontologia *CoreSec*. Após receber a confirmação da inicialização de cada agente, o Agente *Manager* confirma essa informação ao Agente *Viewer*, finalizando o processo de inicialização do sistema. Na figura 5 ilustramos a tela inicial do *AutoCore* detectando um ataque:

Figura 5 – Tela Inicial do AutoCore Detectando um Ataque.

Podemos ver no campo de informações todo o acompanhamento da execução do *AutoCore* na detecção de uma ataque, a troca de mensagens entre os agentes, o tipo de ataque que foi detectado e a persistência dos dados na *CoreSec*. É apresentado na Figura 6 o processo de comunicação entre os agentes que compõem o *AutoCore* no momento da simulação de um ataque DoS utilizando o simulador descrito anteriormente.

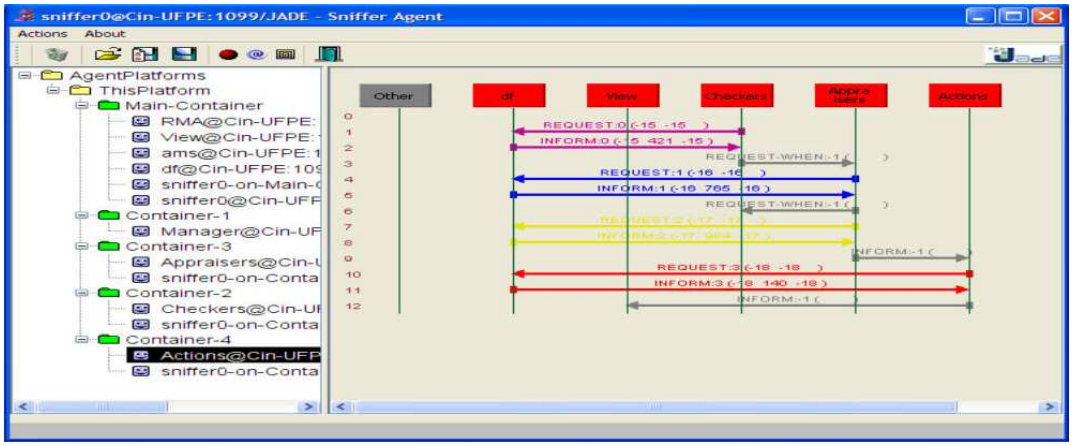

Figura 5 – Tela Inicial do AutoCore Detectando um Ataque.

Os Agentes *Checkers* são os responsáveis por verificar o tráfego da rede. No momento em que percebem uma alta frequência de pacotes vindos de um mesmo *host*, considerando ainda o tamanho e o tipo do pacote em um dado período de tempo, detectam estar havendo algo anormal, baseado nas regras da base de conhecimento de uma instância da *CoreSec* carregadas durante a inicialização. Após consultarem o DF e saberem para quais agentes devem enviar essas informações, fazem uma requisição para os Agentes *Appraisers* enviando as informações de *status* da anomalia da rede.

Os Agentes *Appraisers*, baseados nas descrições informadas pelos Agentes *Checkers*, consultam a ontologia *CoreSec* com o objetivo de avaliar a anomalia. Após consultarem o DF, fazem uma requisição aos Agentes *Actions*. Estes devem tomar decisões para amenizar o problema e para isso consultam uma instância da *CoreSec*, procurando informações específicas para resolução da anomalia em questão. A Auto-Cura caracteriza-se, então, pela execução da melhor solução baseada nas métricas recebidas dos Agentes *Appraisers*. Após o *AutoCore* detectar um possível ataque DoS, as seguintes ações são tomadas pelos agentes: encaminhar os pacotes forjados para uma falsa aplicação, com o objetivo de alterar o fluxo do ataque liberando a aplicação real para uso e informar o CSO sobre a ocorrência do ataque, apresentando-lhe a origem do emissor dos pacotes forjados, facilitando a criação de regras concisas para configuração do *firewall*.

Após consultarem o DF, os Agentes *Actions* enviam uma mensagem para os Agentes *Views*, que interagem com o *CoreEditor* informado a situação atual, o que foi realizado ou não e as possíveis outras correções ao CSO para, caso seja necessário, intervir no sistema.

Tabela 1 – Resultado das Consultas Utilizando o *CoreEditor*.

Atacante	Ataque	Vulnerabilidade	Recurso	Probabilidade Ocorrência	Impacto Negócio	Nível Tolerância	Contra-Medida
Hacker	*UPD Packet Storm*	*Distributed Denial of Service*	Servidores	*Hard*	*Hard*	*Not Acceptance*	- Analisar tráfego da rede (Pacotes TCP e UDP); - Utilizar IDS.
Hacker	*SYN Flooding*	*Denial of Service*	Servidores	*Hard*	*Hard*	*Not Acceptance*	- Analisar tráfego da rede (Pacotes TCP e UDP); - Utilizar IDS.
Nature	*Fire*	Falta de proteção contra incêndios	Sala dos Servidores	*Low*	*Hard*	*Acceptance*	- Cofres à prova de fogo; - Salas à prova de fogo; - Duplicação dos servidores em locais remotos.

Como o *AutoCore* é composto pelo *CoreEditor* na camada de visualização, pode-se então utilizá-lo para monitoramento e apoio à Gestão de Segurança e Riscos, mantendo assim, os serviços oferecidos pelas organizações disponíveis sem perder competitividade no mercado. Algumas consultas foram baseadas nas seguintes questões de competência que a *CoreSec* deve ser capaz de responder: se um agente mal intencionado explorar uma vulnerabilidade e realizar um ataque do tipo DoS ou DDoS – *Distributed Denial of Service*, por exemplo, qual a conseqüência destes ataques para a organização? Qual a probabilidade de ocorrência desse ataque? Qual o impacto nos negócios? Quais ativos são atingidos? Qual nível de tolerância é permitido? Quais controles utilizar para mitigar a exploração de tal vulnerabilidade? Qual o nível de risco aceitável?

As instâncias utilizadas para alimentar a *CoreSec* foram obtidas do NIST4 (Instituto Nacional de Padronização e Tecnologia) dos Estados Unidos. Trata-se de uma base de dados internacional de vulnerabilidades, um repositório de acesso gratuito, contendo informações sobre cada vulnerabilidade e como corrigi-las. Utilizando o módulo de consulta do *CoreEditor* algumas consultas foram realizadas e seus resultados são apresentados na Tabela 1. As consultas que respondem às questões de competências foram definidas de acordo com o que os especialistas necessitam responder para mitigar riscos reais em suas corporações.

De acordo com o resultado das consultas, os responsáveis pela segurança tomarão decisões estratégicas para mitigar os riscos e manter os serviços oferecidos pelas organizações disponíveis sem perder competitividade no mercado.

Observando a Tabela 1, seria dada prioridade aos ataques de DDoS e DoS por terem grande probabilidade de ocorrência, alto impacto nos negócios e por possuir um nível de tolerância não aceitável para a organização. Ao ataque *Fire* não seria dada prioridade já que a probabilidade de ocorrência é baixa e o risco nesse caso é aceitável, porém, os impactos e valores de segurança são altos. Os responsáveis deverão analisar questões de níveis de aceitação para tomar decisões de controle do risco. Algumas classes não foram exibidas na tabela de resultados por questões de espaço. Sem a utilização da *AutoCore* as decisões são tomadas de forma manual sem apoio de uma base de conhecimento ou ferramenta computacional inteligente comprometendo assim a tomada de decisões sobre segurança.

7 Conclusões e Trabalhos Futuros

O *AutoCore* cumpre o papel ao qual se propõe, constituindo-se em uma ferramenta computacional que poderá ser utilizada para o tratamento e utilização da informação a respeito de riscos e segurança da informação, possibilitando aos responsáveis pela Gestão de Riscos e Gestão de Segurança da Informação tomar decisões estratégicas de alinhamento de TI e Segurança aos processos de negócios das organizações fornecendo uma visão de alto nível entre os envolvidos.

Os resultados obtidos são encorajadores e indicam sucesso na detecção e correção de ataques DoS e no auxílio a tomada de decisões por parte dos responsáveis pela segurança computacional de uma determinada corporação. Foi possível realizar a avaliação e validação do *AutoCore* resolvendo um problema real que atinge diversas organizações e o sistema proposto se comportou como esperado e de acordo com as características de um sistema autonômico, melhorando assim a complexidade de procedimentos relativos a ferramentas que implementam mecanismos de segurança semelhantes.

Como trabalhos futuros estamos trabalhando na detecção de outros tipos de ataques como o DDoS e variantes do DoS, SQL *Injection*, *Buffer Overflow* entre outros, ampliando o número de estudos de caso realizados, também será realizada a junção do *AutoCore* a *firewalls*. Algumas classes estão sendo adicionadas à *CoreSec* e funcionalidades estão sendo inseridas no *CoreEditor*, como o módulo de relatórios apoiando ainda mais o nível gerencial corporativo.

7 Referências

[1] Almeida, M. J. S. C.; et al. 2007. An Ontology about Information Security Management for Autonomic Computing Environments. In: 2nd Latin American Autonomic Computing Symposium (LAACS 2007), 2007, Petrópolis - RJ. Proceedings of the 2nd Latin American Autonomic Computing Symposium (LAACS 2007), 2007.G. A. Rovithakis, I. Chalkiadakis, M. E. Zervakis, High-order neural network structure selection for function approximation applications using genetic algorithms, **Systems, Man, and Cybernetics, Part B: Cybernetics**, 34(2004), 150 – 158.

[2] Carroll, J, J. Carroll, et al. 2004. Jena: implementing the semantic web recommendations. In WWW (Alternate Track Papers & Posters), 2004.

[3] Chess, D.M., Palmer, C. C. e White, S.R. 2003. Security in an Autonomic Computing Environment. IBM Systems Journal, Volme 42, Issue 1, 2003.

[4] FIPA (2002), FIPA Contract Net Interaction ProtocolSpecification,http://www.fipa.org/specs/fipa00029/SC0029H.pdf.

[5] Fernández, M. A.; Gómez-Pérez, A.; Juristo, N. 1997. Methontology: From ontological art towards ontological engineering. In Proceedings of the AAAI Spring Symposium Series, p. 33-40.

[6] Freitas, F; Schulz, Stefan ; Moraes, Eduardo. 2009. Survey of Current Terminologies and Ontologies in Biology and Medicine. In revista eletrônica de comunicação, informação e inovação em saúde. Vol. 3, p. 1-13.

[7] Freitas, F. 2003. Ontologias e a web semântica. In: Renata Vieira; Fernando Osório. (Org.). Anais do XXIII Congresso da Sociedade Brasileira de Computação. Campinas: SBC. Vol. 8, p. 1-52.

[8] Guarino, N. 1998. Formal Ontologies and Information Systems. In Formal Ontology in Information Systems. Proceedings of FOIS'98, Trento, Itália. 6-8 june. P3-15. IOS Press.

[9] Gruber, T. R. 1995. Toward Principles for the Design of Ontologies used for Knowledgement Sharing. In International Journal of Human-Computer Studies.Vol 43, Inssue 5-6: 907-928.

[10] Guizzardi. G. 2000 "Uma abordagem metodológica de desenvolvimento para e com reuso, baseada em ontologias formais de domínio." Dissertação de Mestrado. Universidade Federal do Espírito Santo.

[11] Horn, P. 2001 "Autonomic Computing: IBM´s Perspective on the State of Information Technology"; http://www1.ibm.com/industries/government/doc/contet/resource/thought/278606109.html. Last access in 6th February 2009.

[12] IBM - International Bussiness Machine. 2006. An Architectural Blueprint for Autonomic Computing. Avaliable inhttp://www 03.ibm.com/autonomic/pdfs/ACBP2_2004 10- 04.pdf. Last access in 15th March 2009.

[13] Khadraoui, D. e Herrmann, F. 2007. Advances in Enterprise Information Technology Security, Books24x7: IGI.

[14] Kephart, Jeffrey O. and Chess, David M. 2003. The Vision of Autonomic Computing. IEEE Computer Society, January, 41-50.

[15] Martimiano, L. A. F. 2006. Sobre a estruturação de informação de segurança computacional: o uso de ontologia. 163 p. Tese (Doutorado em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação – ICMC, Universidade de São Paulo - USP, São Carlos.

[16] OWL. Web ontology language overview - w3c;. [Online]. Disponível: http://www.w3.org/TR/owl-features/ Acessado em: Feb. 2011.

[17] Parashar, M. and Hariri S. 2007. Autonomic Computing: Concepts, Infrastructure, and Applications. CRC Press, USA.

[18] RDF. Web ontology language overview - w3c;. [Online]. Disponível: http://www.w3.org/RDF/ Acessado em: Feb. 2011.

[19] Stojanovic, L. et al. 2004. The Role of Ontologies in Autonomic Computing Systems. IBM Systems Journal, Volume 43, Issue 3.

[20] T. Berners-Lee, O. Lassila, and J. Hendler. 2001. The semantic web. Scientific American, 5:34–43.

[21] Uschold, M, Grüninger, M. 1996. Ontologies: Principles, Methods and Applications. Knowledge Engineering Review. Vol. 11, Nº 02. June.