An Ontological Approach to Situational Awareness Applied to Information Security

Diórgenes Yuri da Rosa, Ricardo Almeida, Roger Machado, Adenauer Yamin, and Ana Marilza Pernas
Universidade Federal de Pelotas
Email: {diorgenes,rbalmeida,rdsmachado,adenauer, marilza}@inf.ufpel.edu.br

***Abstract*—The typical infrastructures of Ubiquitous Computing assume characteristics of flexibility regarding the connectivity in the environments. Aiming security in these scenarios, several solutions are deployed in its own syntax languages, providing events in different formats. In this sense, Situation Awareness, as a strategy capable of integrating events from different sources, becomes a requirement for the controls implementation. This work proposes an ontological approach to SA applied to the information security domain, called EXEHDA-SO. The proposal was evaluated based on a real infrastructure, showing itself capable of handling heterogeneous events from different contextual source.**

***Index Terms*—Situational Awareness, Information Security, Ontology**

I. Introdução

Para atender as prerrogativas de Computação Ubíqua (Ubicomp) [1], tarefas e funcionalidades cotidianas geralmente contam com algum tipo de conectividade, impondo um ambiente flexível e atento a constantes mudanças. Em se tratando de redes de computadores, o foco nestas premissas remete a uma infraestrutura permissiva, isto é, com o menor número de bloqueios e controles possível. Entretanto, reduzir o número de controles pode expor as informações que trafegam em rede. Estes riscos potencializam-se pelo aumento de conexões simultâneas e conseguinte elevação do tráfego de rede, bem como devido ao aumento na complexidade das conexões com a utilização de diversas portas de conexão, expondo o ambiente a ocorrência de ciberataques.

Em atenção a estes problemas verifica-se um uso recorrente de soluções de propósito específico no domínio de Segurança da Informação (SI), utilizando diferentes formatos para eventos. Além disto, dada a característica flexível e mutável dos ambientes de Ubicomp novas soluções de segurança podem ser necessárias. O emprego destas diferentes soluções de SI trazem empecilhos quanto a integração de diferentes contextos, dificultando uma visão unificada do ambiente. E as atuais soluções usadas para correlação de eventos em SI costumam utilizar linguagens com sintaxes próprias, o que dificulta sua reutilização e não propicia aproveitamento compartilhado das constantes evoluções nas regras de correlação.

Considerando este panorama, a Ciência de Situação (CS) traz consigo estratégias para tratamento deste problema, pois é construída sobre uma visão ampla a respeito do contexto do ambiente. A CS fomenta a identificação de conjunturas complexas, determinando assim auxílio importante aos analistas de segurança nas tomadas de decisões para proteção da infraestrutura. O trabalho [2], define que um sistema é ciente de contexto se este utiliza contexto para fornecer informações ou serviços relevantes para o usuário, onde a relevância depende da tarefa do usuário.

Posto este alinhamento, o presente trabalho propõe uma solução para CS por intermédio de uma abordagem baseada no uso de ontologias. Estas estruturas proveem entendimento compartilhado e processável sobre um domínio de conhecimento, podendo especificar relações semânticas entre diferentes situações e identificando cenários de alto nível. O modelo ontológico concebido, denominado EXEHDA-SO *Execution Environment for Highly Distributed Applications - Security Ontology*, contempla conceitos de SI e informações sobre a arquitetura e os ativos que suportam as funcionalidades de um ambiente típico em UbiComp. Este foi testado e avaliado em ambiente simulado, alusivo à infraestrutura computacional da Universidade Federal de Pelotas. No desenvolvimento, foram integrados conceitos já consolidados pelos trabalhos prévios do grupo de pesquisa LUPS (Laboratory of Ubiquitous and Parallel Systems) [3] e [4].

Este artigo estrutura-se por intermédio de x seções. Na seção II são destacados alguns aspectos fundamentais sobre a teoria de CS, bem como o uso de ontologias enquanto estratégia de CS. A seção III aborda a proposta ontológica desenvolvida e detalha seus fragmentos. Já na sessão IV apresenta-se os estudos de caso e na seção V os trabalhos relacionados. O artigo é concluído na seção VI.

II. Ciência de Situação

Uma situação é caracterizada por um conjunto de elementos contextuais de interesse instanciados e relacionados de forma a prover alguma informação válida em um intervalo de tempo específico. Tendo definido o termo situação, a CS consiste das etapas de: (i) Percepção envolvendo os processos de detecção, reconhecimento e monitoramento, que levam a consciência de múltiplos elementos situacionais (objetos, eventos, pessoas, sistemas, fatores ambientais) e seus estados atuais (locais, condições, formas, ações); (ii) Compreensão que busca integrar informações correlacionando elementos de contexto desconexos para propiciar um entendimento acerca de como se dá o impacto para com as metas e objetivos dos usuários e sistemas; e (iii) Projeção cuja principal característica consiste na antecipação de ocorrências futuras tendo como base a compreensão dos elementos contextuais atuais.

Esta definição de CS leva a um fluxo onde, na etapa de percepção, os eventos são inicialmente registrados por sensores distribuídos, coletados e, na sequência, contextualizados. Na etapa de compreensão, estes eventos contextualizados são processados buscando identificar possíveis situações de interesse. Finalmente, na projeção as situações detectadas fornecem subsídio para a tomada de decisão, podendo realizar atuações diretamente no ambiente monitorado.

A área de segurança computacional, a qual concentra-se este trabalho, tem se voltado para ontologias como subterfúgio para a etapa de Compreensão de CS. Ao considerar que SI permeia diversos níveis tecnológicos é possível destacar o uso de ontologias atendendo os mais diversos aspectos deste domínio. O trabalho [5], já identificava a potencialidade de sua proposta ontológica (Telos) para o desenvolvimento de modelos de especificações de segurança. O uso de ontologias nos domínios de SI é defendido por prover: (i) um entendimento compartilhado da informação estruturada que pode ser fundamentada e analisada automaticamente entre seres humanos e agentes de software; (ii) reuso de conhecimento facilitando a evolução das soluções; (iii) melhora na questão da interoperabilidade, pois diferentes aplicações podem valer-se dos conceitos e relacionamentos definidos nas ontologias e (iv) a habilidade de especificar várias relações semânticas entre diferentes conceitos [6] [7].

III. EXEHDA-SO

A EXEHDA-SO consiste em uma abordagem ontológica, capaz de considerar informações de diversas fontes no intuito de gerar inferências que possam auxiliar na detecção de situações de interesse para aprimoramento da segurança do ambiente ubíquo monitorado. Este trabalho constitui uma colaboração ao *middleware* EXEHDA [3] e derivado trabalho voltado à SI desenvolvido em [4], integrando as funcionalidades de identificação de situações de interesse, focada em ampliar as possibilidades da camada de compreensão de CS.

A Figura 1 apresenta uma visão abrangente da solução proposta. Nesta, trabalhos prévios do grupo estão refletidos, sendo proposto o tratamento de contexto em um Repositório Híbrido de Informações Contextuais [8] e o processamento de eventos por meio de regras [4]. O EXEHDA-SO propõem o Processamento Híbrido de Eventos, a partir do desenvolvimento de um modelo ontológico capaz de prover semântica, interoperação e flexibilidade ao método previamente proposto.

Para concepção da EXEHDA-SO foram definidos três fragmentos ontológicos (Core, Scope Analiser e InterCell Analiser), cujas principais classes podem ser visualizadas na Figura 2. As próximas subseções descrevem cada um dos fragmentos.

A. EXEHDA-SO: Core

O fragmento ontológico denominado *Core* visa representar conceitos gerais de SI e do próprio *middleware* EXEHDA, no sentido de transversalizar uma base de conhecimento em apoio as funcionalidades de CS tratadas ao longo do modelo como um todo. Este fragmento, bem como os demais, consiste em uma ontologia tipificada como Leve. Não busca-se representar a completude de conceitos estabelecidos em bibliotecas do domínio de SI, refletindo sim apenas a abstração necessária para as demandas do projeto.

Este fragmento conta com a classe *Fundamentals* que remetem aos 5 aspectos teóricos centrais de SI destacados pela ISO 27002[1]: *Confidenciality*, *Integrity*, *Availability*, *Authenticity*, *Non-repudiation*. Por intermédio da propriedade *aimstoEnsure* pontua-se a classe *Control* que visa assegurar os fundamentos de SI. De forma geral, os cenários de interesse são caracterizados no momento em que confirma-se ou existe a suspeita de que estão ocorrendo eventos voltados a perda de algum destes aspectos de SI.

Os ativos que compõem o ambiente ubíquo, desde o mais básico sensor até servidores de aplicação, são passíveis de ataques e são representados no modelo pela classe *EXEHDANode*. Como subclasses desta representação tem-se (i) *EXEHDAmobNode* englobando dispositivos móveis, equipamentos específicos de IoT ou sensores diversos, (ii) *Station* para as estações de trabalho não móveis, (iii) *Border Server* para servidores distinguindo instalações nativas e virtuais. Por intermédio das propriedades de dados das instâncias da classe *EXEHDANode* é possível estabelecer um perfil de execução que dita as regras pelas quais aquele ativo é submetido em termos de conectividade com a rede interna e externa, sistema operacional, hardware disponível, entre outros.

A classe *Control* representa toda a contramedida que pode ser adotada em prevenção ou correção a um determinado cenário de ataque identificado. Esta classe descreve requisições de mudança no ambiente como sendo: (i) *Preventive*, ao estabelecer controle voltado a alguma atuação que possivelmente pode ocorrer no ambiente observando comportamentos passados; (ii) *Reductive*, quando mesmo sem garantir a não ocorrência de determinada ação há uma diminuição das situações, minimizando as falhas decorrentes; (iii) *Repressive*, que são medidas que neutralizam as ocorrências maliciosas; e (iv) *Corrective* quando são necessários ajustes estratégicos para a recuperação de falhas. Considerando que a aplicação destes controles pode impactar o ambiente de alguma maneira, são estipuladas ações ativas e passivas. Por ação ativa entende-se toda a ação voltada a proteção ao ambiente executada de forma automática, quando a criticidade do cenário impõe isto. O bloqueio de uma porta de comunicação por intermédio de um *script* personalizado que interage com o *Firewall* exemplifica um controle ativo. Já os controles passivos são aqueles que apenas instrumentalizam a análise de um gestor de redes. O envio de um e-mail é considerado uma ação passiva, pois não interage diretamente com os perfis de execução do ambiente de forma automática.

Para a classe *Risk* são avaliados tanto o impacto que a exploração de determinado ativo pode representar quanto a probabilidade com que esta exploração ocorra. Cenários percebidos tem seus riscos determinados por dados contextuais que podem ser incluídos inclusive na fase de pré-processamento, como por exemplo (i) severidade da situação (ii) prioridade do

[1]ISO 27002: boas práticas para a gestão de segurança da informação

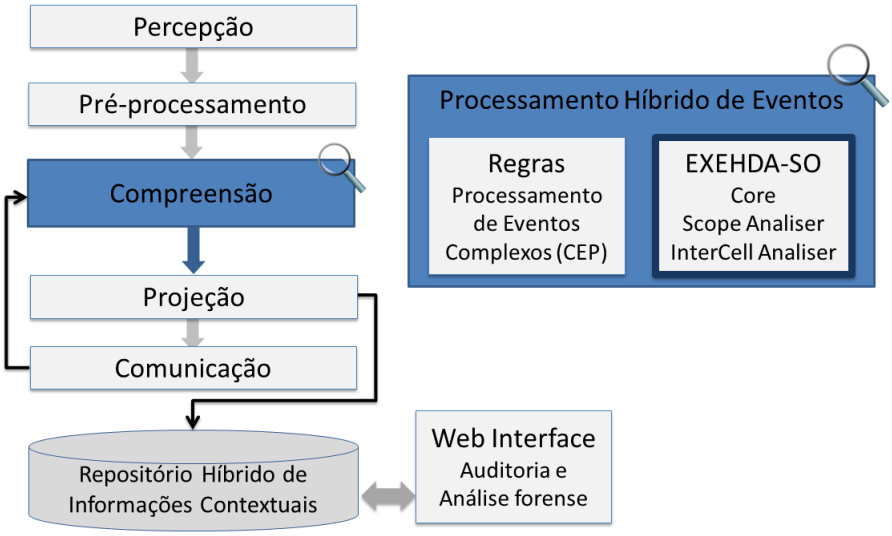

Figura 1. EXEHDA-SO integrando a etapa de Compreensão em CS

evento e (iii) criticidade do ativo, sendo tratados como propriedades de dados. Como subclasses de *Risk* foram definidas: *Human Actions*, *System and Technology Failures*, *External Factors*.

Todo o evento percebido que possa impactar algum dos fundamentos básicos de SI estabelecidos no modelo é representado pela classe denominada *Events*. Esta é uma das classes principais do modelo pois a partir dos eventos, geralmente estruturados em logs de serviços ou soluções de segurança, encaminha posterior identificação de situações de interesse. Cada instância desta classe trás consigo uma série de predicados, materializando dados contextuais que definem o evento segundo aspectos técnicos relevantes, como: endereçamentos IP's e *HostName* de origem e destino do evento, o serviço alvo, por exemplo. Por intermédio dos predicados das instâncias desta classe é possível verificar o método utilizado pelos atacantes, sua estratégia e intenção. Identifica-se assim se o evento refere-se a, por exemplo, um escaneamento de portas (atividade tipicamente preliminar à ataques), tentativa de entrega de *malware's*, tentativas de acesso ilegítimo, entre outros. Conforme definido pela relação *exploits*, os eventos são ações maliciosas destinadas a *EXEHDANode's*, sendo um vocábulo estipulado para ataques ou ações tipicamente preliminares à ataques.

B. EXEHDA-SO: Scope Analyzer

A concepção do fragmento ontológico *Scope Analyzer* define as representações necessárias às observações das situações de interesse do ambiente ubíquo no que diz respeito as primeiras fases de processamento ontológico. O seu foco central está no reconhecimento de contextos específicos de segurança na célula de execução atribuída a ele.

O processamento dos cenários já nas células aproxima a correção dos nodos envolvidos no que tange a conectividade, isto é, a correção é aplicada sem percorrer outros níveis da arquitetura. Esta alternativa também auxilia a questão de escalabilidade no tratamento dos eventos ao distribuir o processamento entre setores distintos do ambiente.

As relações entre as demais classes auxiliam a definição de vulnerabilidades, representada pela classe *Vulnerability* do modelo. Neste sentido destaca-se que um controle (classe *Control*) tem por intenção a minimização de uma vulnerabilidade, podendo inclusive eliminá-la protegendo assim os *EXEHDANode's* alvo que eventualmente possuam a vulnerabilidade em questão.

Relaciona-se com *EXEHDANode* a classe denominada *Profile*. Esta representa as configurações vigentes empregadas aos recursos computacionais, caracterizando regras de conectividade na subclasse *Network* e especificando quais serviços estão ativados em *Services*. Mais especificamente, a subclasse *Network* disponibiliza padrões permissivos ou restritivos, observada a relevância e criticidade do equipamento em questão.

A classe *Events*, descrita no fragmento *Core*, possui considerável relevância neste fragmento, uma vez que são aceitos dados oriundos de fontes distribuídas e heterogêneas. Com isso, ao identificar padrões de interesse com base nas propriedades de dados são então configuradas as situações, representadas pela classe *Situation*, as quais por sua vez determinam o uso de um determinado controle. A classe *Situation* pode ser compreendida como um repositório de cenários de interesse.

C. EXEHDA-SO: InterCell Analyzer

Observando a larga conectividade e distribuição dos ambientes ubíquos pode-se considerar que uma análise de eventos individualizada das células de execução não é capaz de gerar, por si só, uma visão global do ambiente quanto a situações de SI. Ou seja, é necessário avaliar cenários de múltiplos níveis da arquitetura valendo-se assim do cruzamento de informações entre células distintas. Desta forma, a intenção do fragmento *InterCell Analyzer* é prover uma visão unificada do ambiente, recebendo ocorrências situacionais de múltiplas células e inferindo controles, considerando que a ação pode

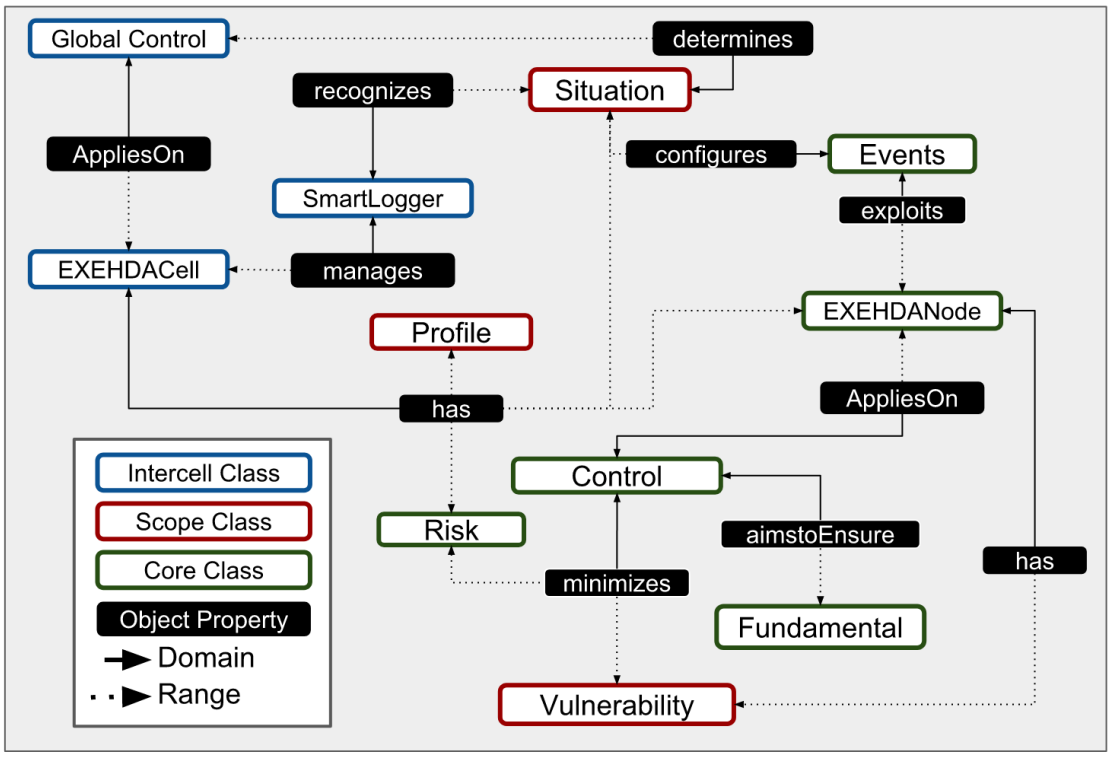

Figura 2. Principais classes da EXEHDA-SO

ser efetuada em todo o ambiente. No caso, o *Intercell* recebe os dados tratados no fragmento *Scope Analyzer* adicionando alguns conceitos necessários a ações entre células.

A classe *EXEHDACell* é responsável pelo conhecimento referente às células que compõem o ambiente ubíquo, em seus diversos níveis. Esta classe é instanciada por informações sobre os ativos presentes naquele parque computacional por intermédio da sua relação com a classe *EXEHDANode*. É então obtido um panorama geral do funcionamento das células por intermédio dos equipamentos que regem as conexões. Já na subclasse *level*, específica desta classe, é delimitada a abrangência daquela célula no que se refere a questões organizacionais e até mesmo disposição geográfica.

O SmartLogger, componente arquitetural responsável pela primeira frase de processamento ontológico, é representado por uma classe de mesma nomenclatura. Assim a *InterCell* reconhece a gestão deste componente considerando subclasses que trazem dados sobre as técnicas de coleta utilizadas, quais são os sensores disponíveis, quais são os *scripts* personalizados existentes, entre outros. Esta classe também indica as *EXEHDACell's* nas quais o *SmartLogger* efetua o gerenciamento, podendo ser em um ou mais níveis da arquitetura.

Neste fragmento novamente a classe *Situation* está presente, contudo desta vez incorporando informações contextuais sobre a localização onde a situação ocorreu. Observar as situações instanciadas por fragmentos ontológicos dispostos em setores periféricos da arquitetura no *InterCell Analyzer* serve como gatilho preventivo a situações identificadas em outras células. Desta forma, identifica-se a propriedade de objetos *determines*, ligando estas situações aos *Global Controls*. Diferente da classe *Control* do fragmento *Scope*, esta classe provê contramedidas observando todas as células. Com esta ação transversalizada no ambiente, protegem-se serviços e funcionalidades passíveis dos mesmos cenários, mesmo que as ocorrências ainda não tenham explorado àquele ambiente.

IV. Cenário de Uso

A Universidade Federal de Pelotas (UFPel) apresenta em seu conjunto de recursos tecnológicos uma série de características comuns aos conceitos de UbiComp. Dentre estas características destaca-se a elevada distribuição na qual os recursos tecnológicos estão distribuídos entre os diversos campus. De forma análoga a este cenário foram instalados os servidores, conforme demonstra a figura 3:

- Servidores Web: nestes servidores são hospedados sites para os quais, de forma geral, não há mais manutenções ou atualizações. São projetos antigos que ainda detém grande importância institucional, contudo por questões técnicas (versões de serviços, por exemplo) e de gestão, não têm mais investimentos.
  - *webserver1* (WS_1): sistemas gerenciadores de conteúdo, do inglês *Content Management System*. O endereço IP deste ativo é 192.168.0.1.
  - *webserver2* (WS_2): hospedagem de sites com painel de controle *open source* para gerenciamento dos ambientes web. O endereço IP deste ativo é 192.168.0.2.

  - *webserver3* (WS_3): Sites descontinuados que ainda precisam estar acessíveis para consulta, entre outros sistemas legados. 192,168.0.3.

- *Vulnerability Analysis* (VA): servidores que foram distribuídos no ambientes simulados do Campus Campão do Leão (CCL) e do Campus Anglo (CA), os quais possuem um conjunto de soluções sem custo e de código aberto para análise de vulnerabilidades incluindo OpenVAS[2] (Open Vulnerability Assessment System), Nikto[3], wapiti[4], sqlmap[5], entre outros, e *scripts* personalizados para automatização de determinadas tarefas.
- *Network Intrusion Detection System* (NIDS): o suricata-ccl e o suricata-ca são responsáveis por realizar a análise do tráfego de rede da DMZ (*demilitarized zone*) nos dois campus, sendo baseados no IDS suricata[6].
- *Host Intrusion Detection System* (HIDS): Sistema de Detecção de Intrusão Baseado em *Host*, configurado em cada *webserver*, monitora o comportamento ou os estados dinâmicos de uma máquina verificando eventos registrados em log ou os dados de auditoria e, portanto, detecta e avalia as mudanças no sistema de arquivos, controle de acesso de usuários, o comportamento dos processos do sistema e o uso de recursos, entre outros.
- *Firewall*: foi prevista a existência de um Firewall para a DMZ de cada datacenter. Este firewall realiza a filtragem de pacotes seguindo uma política restritiva de acesso. O mesmo considera apenas o cabeçalho das camadas de rede e de transporte, camada 3 e 4 do modelo OSI (*Open System Interconnection*) respectivamente. Ou seja, as regras são formadas indicando os endereços de rede (de origem ou destino) e as portas TCP/IP envolvidas na conexão.

Proteger o ambiente a partir da análise de origem e destino dos eventos detectados é um dos eixos centrais do estudo de caso. Verificar se os eventos de ataque são oriundos de apenas um ou mais *hosts* trás indícios da abrangência necessária das contramedidas. Isto é, se a origem dos eventos coletados, verificada na ontologia pela propriedade *eventSrcIP*, for recorrente para um ou mais destinos é interessante que o analista considere especificamente o bloqueio daquele agente seguindo indicação de contramedida documentada na classe *Controls* da ontologia. Por outro lado, se os eventos refletem a ação de muitas origens (novamente com base no *eventSrcIP*) em um só destino (considerando o campo *eventDstIP*), a contramedida pode considerar o sugestionamento ao analista de segurança para avaliar a possibilidade de diminuir as possibilidades de ataques no servidor de borda em questão (desativando serviços, limitando o acesso aos mesmos apenas da rede interna, entre outras alternativas que objetivem a mitigação do risco).

[2]http://www.openvas.org/
[3]https://cirt.net/Nikto2
[4]http://wapiti.sourceforge.net/
[5]http://sqlmap.org/
[6]http://suricata-ids.org/

Para demonstrar esta identificação, primeiramente o Colector recebe o evento bruto oriundo do HIDS (*Host-based Intrusion Detection System*) OSSEC *(Open Source Security)* que está configurado em modo *Standalone* no servidor WS_2. Observa-se que um evento similar a este é também registrado no WS_1 diferindo apenas o IP e *hostname* do servidor de destino e o momento da ocorrência.

Este evento reflete um ataque de Força Bruta (conhecidos em inglês como *Brute Force Attacks*) que consiste em uma ação maliciosa na qual é realizada a tentativa de descoberta de credenciais de um determinado sistema ou serviço, geralmente por intermédio de um dicionário *(wordlists)* ou por combinação de caracteres. Para execução do ataque em questão foi utilizada a ferramenta THC-Hydra[7] disponível nativamente na distribuição linux Kali [8].

Os *logs* que caracterizam os eventos de Força Bruta são então tratados pelo componente *Colector* dos servidores. Este estágio de pré-processamento, vale-se inicialmente do Filebeat que lê *logs* oriundos de diversos sensores do ambiente, inclusive do HIDS OSSEC. O Filebeat encaminha então os *logs* recebidos ao Logstash, controlando o fluxo deste encaminhamento para que não ocorra sobrecarga, provendo métodos de retomada do encaminhamento em caso de falha do Logstash e criptografando esta comunicação. No Logstash estes dados são normalizados utilizando uma expressão regular.

O resultado da normalização do Logstash sobre os dados dos eventos são disponibilizados em JSON, conforme demonstrados na figura 4. Ainda nesta figura, observa-se a adição dos campos *category*, *sub_category* e *priority*, os quais são resultantes da contextualização empregada pelo Logstash. A partir deste estágio de pré-processamento já se pode visualizar de maneira simplificada, quando comparado ao evento bruto, algumas das informações centrais para a definição do cenário de interesse como: (i) quem executou a ação maliciosa no campo SRCIP; (ii) o alvo escolhido pelo atacante no campo DSTIP; (iii) quando ocorreram as ações no campo ”date”; (iv) o serviço relacionado e o tipo do ataque no campo “rule_comment”.

Após executadas estas tarefas, o componente Colector encaminha os eventos para o SmartLogger da célula que representa o CCL, onde a ontologia é instanciada na classe *Events* para ataques, conforme as triplas demonstradas na tabela I. Observa-se que nem todos os campos foram encaminhados para a ontologia, sendo utilizados apenas os dados relevantes para o processamento ontológico deste caso. Para efetuar o encaminhamento dos eventos na ontologia foi utilizada a API Java Jena.

Neste ponto, com o evento apresentado na tabela I em conjunto com o evento similar produzido no WS_1 já instanciados na classe estipulada *Events*, tem-se o conhecimento de diversos fatores fundamentais de uma ação maliciosa como (i) quais foram os endereçamentos de origem e destino do evento; (ii)

[7]https://www.thc.org/thc-hydra/
[8]https://www.kali.org/

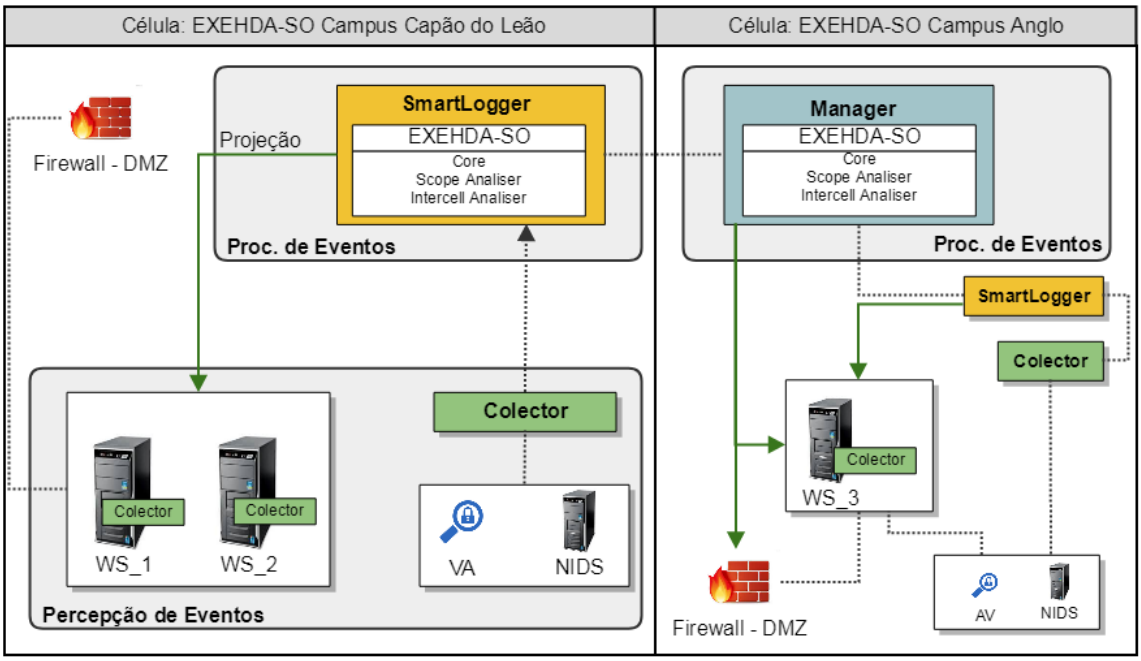

Figura 3. Componentes EXEHDA-SO: disposição de ativos em duas células

```
{_id: ObjectId("5579fbd26e58bc7ac47bb87"),
date: "2017-09-12 15:31:40.000000";
rule_id:"11306",
rule_level:"10",
rule_group:['syslog','pure-ftpd','authentication_failures'],
rule_comment:"FTP brute force (multiple failed logins)",
srcip:"192.168.0.10",
username:"webmaster",
hostname"webserver2",
location:"/var/log/messages",
dstip:"192.168.0.2",
event:"Set 12 15:31:40" webserver2 pure-ftpd:(?@192.168.0.10)
[WARNING]
Authentication failed for user [webmaster]".
category:"authentication",
sub_category: "failed",
priority: "10" }
```

Figura 4. Pré-processamento: normalização de eventos no Logstash

Tabela I
TRIPLAS DA CLASSE *Events*

Sujeito	Predicado	Objeto
Event	eventSrcIP	192.168.0.2
Eventss	eventDstIP	192.168.0.10
Events	eventCat	authentication
Events	eventService	FTPD
Events	eventDate	2017-09-12
Events	eventSUser	webmaster
Events	eventDstHN	webserver2
Events	eventTime	15:31:40

o serviço alvo; (iii) o usuário que está sendo utilizado pelas tentativas de acesso e (iv) a prioridade do evento.

A estratégia para esta identificação de cenário primeiramente separa os eventos por intermédio do predicado destino denominado *eventDstIP* que informa o endereço IP alvo. Assim, na simulação realizada, todos os eventos destinados para o WS1 são encaminhados para a classe que o representa na hierarquia de ativos *(EXEHDANode)*. Esta regra para o WS1 pode ser visualizada na figura 5. A regra repete-se para cada Web Server previsto.

A tarefa de perceber que muitos eventos apresentam uma mesma origem para mais de um destino no ambiente é executada pela regra da figura 6. Nesta regra SWRL executada pelo SmartLogger do CCL, a origem de eventos identificados nos dois servidores web são comparados e caso apresentem a situação de interesse, estes eventos são então instanciados na classe *Situation*, a qual por sua vez determina os controles a serem identificados por meio da classe *Control*.

A partir deste momento, a classe *Situation* é instanciada com os eventos cujas propriedades de dados informam quais endereços IP's estão envolvidos no cenário e a ação, que

```
exehdaso:Events(?x) ^ exehdaso:eventDstIP(?x, ?a) ^ swrlb:equal(?a, “192.168.0.1”) ^ exehdaso:eventName(?x, ?b) ^ swrlb:equal(?b, “bruteforce”) ->exehdaso:BF_WS_1(?x)
```

Figura 5. Regra para encaminhar os eventos para a classe que representa o ativo alvo

```
exehdaso:BF_WS_1(?x) ^ exehdaso:BF_WS_2(?y) ^ exehdaso:eventSrcIP(?x, ?a) ^ exehdaso:eventSrcIP(?y, ?b) ^ exehdaso:equal(?a, ?b) ^ swrlb:eventDstIP(?x, ?c) ^ swrlb:eventDstIP(?y, ?d) ^ swrlb:notEqual(?c, ?d) ->OSND(?x) ^ OSND(?y)
```

Figura 6. Regra para identificação de origens únicas em muitos destinos

implica na necessidade de adaptação do ambiente ubíquo promovendo o bloqueio do IP do atacante. Neste momento, é inferido que a recorrência deste atacante no ambiente indica a possibilidade do mesmo realizar novas tentativas de acesso indevido nesta célula. Considerando isto, o *Control* da EXEHDA-SO Core sugestiona que o bloqueio ocorra no Firewall - DMZ, protegendo desta maneira todos os servidores de borda sob a política da DMZ. Na sequência, ocorre a comunicação via API Jena com o módulo de projeção, o qual é encarregado de aplicar a ação ativa de bloqueio do endereçamento de origem (192.168.0.10) destas ações maliciosas no Firewall - DMZ do CCL. A figura 8 mostra as classes do modelo ontológico voltado para este cenário.

Ainda na perspectiva de análise de endereçamentos, outro cenário avaliado foi a identificação de ataques com múltiplas origens focadas em um único servidor de borda, percebendo alvos recorrentes no ambiente. Para realização deste ataque, foram utilizadas 2 máquinas virtuais com kali linux novamente explorando a ferramenta THC Hydra. Desta vez, como alvo do ataque foi definido o servidor webserver3 (WS_3).

Os dados dos eventos são instanciados na classe *Events*, onde posteriormente são separados por regra semelhante a anteriormente apresentada na figura 5). Na sequência, para a análise deste cenário foi estipulada a regra conforme figura 7, a qual executa no SmartLogger do CA. Como consequência da execução desta regra, ocorre a instanciação da classe *Situation*, a qual possui como controle a ação passiva de sugestionamento

```
exehdaso:BF_WS_3(?x) ^ exehdaso:BF_WS_3(?y) ^ exehdaso:eventSrcIP(?x, ?a) ^ exehdaso:eventSrcIP(?y, ?b) ^ swrlb:notEqual(?a, ?b) ->NSOD(?x)
```

Figura 7. Exemplo de regra para identificação de múltiplas origens em um destinos

da desativação do usuário sob ataque (webmaster). O módulo de projeção, ao comunicar-se via API Jena com a EXEHDA-SO, realiza o envio de e-mail com os dados do controle sugerido, contribuindo para a tomada de decisões do analista. Por fim, estes eventos são salvos em um repositório para posterior análise utilizando base de dados Virtuoso.

Uma vez concluído o processamento ontológico dos eventos no Smartlogger, este envia os seus alertas para o Manager, componente disposto na célula alusiva ao Campus Anglo (CA). Conforme referido anteriormente, com a atuação ontológica em um nível superior da arquitetura, o Manager recebendo as situações detectadas pelos SmartLogger's de todas as células que compõem o ambiente, torna-se capaz de inferir ações de proteção preventiva para o ambiente, considerando as detecções individuais das células. Por exemplo, considerando que ambas as situações de interesse identificadas possuem como endereço IP do atacante 192.168.0.10, é possível inferir que o atacante irá continuar a sua busca por acesso indevido a outros serviços que dão suporte ao ambiente ubíquo. Neste caso, a classe de controles global da InterCell Analyzer pode indicar o processo de bloqueio do endereço em questão no firewall das diferentes células sob coordenação do Manager.

Aplicar o raciocínio sobre as classes, com o vocabulário previamente determinado, torna a busca de padrões mais genérica atribuindo uma amplitude maior quanto a identificação de cenários. Como resultado desta análise busca-se uma otimização na identificação de cenários auxiliando os analistas nas tomadas de decisões dirigidas a proteção das informações trafegadas no ambiente. Observa-se ainda que este fator potencializa a avaliação de contextos oriundos de soluções que não possuem integração nativa e cujo formato dos eventos são heterogêneos.

Para a validação da correlação de contextos independentes foi utilizado o OpenVAS que é uma das soluções empregadas pelo servidor VA para identificação de vulnerabilidades e o NIDS Suricata para identificação das atividades maliciosas no ambiente. Já para a simulação do ataque foi explorado o Nmap[9]. Para além da perspectiva de contextos independentes, a determinação deste cenário é interessante pois muitas vezes as equipes não realizam o tratamento das vulnerabilidades por diversas razões. Adicionalmente, o número de alertas gerados pelo Suricata quando empregando regras de detecção de varreduras de redes, pode ser consideravelmente elevado. Então, notificar um incidente associando a vulnerabilidade com uma tentativa de exploração da mesma fornece maior informação para a priorização das adaptações necessárias, novamente auxiliando a tomada de decisões.

Inicialmente é realizada uma varredura de vulnerabilidades pelo OpenVAS. O componente Colector que opera no servidor VA realiza o pré-processamento e o encaminhamento dos eventos e vulnerabilidades identificadas para o SmartLogger. Por sua vez, é realizada a instanciação das vulnerabilidades na classe *Vulnerability* da EXEHDA-SO.

[9]https://nmap.org/

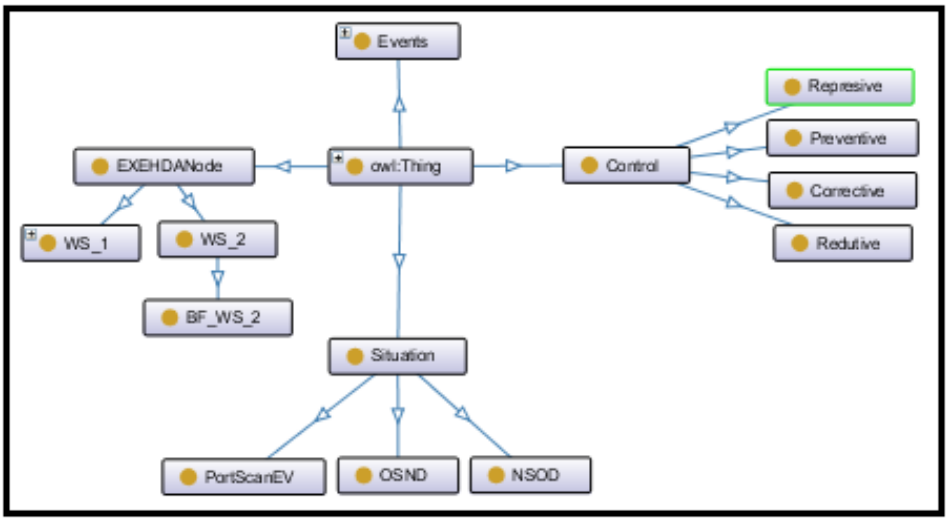

Figura 8. Classes envolvidas no processo de análise de endereçamento

exehdaso:Vulnerability(?y) ^ exehdaso:Events(?x) ^ exehdaso:eventDstIP(?x, ?a) ^ exehdaso? vilIP(?y, ?b) ^ swrlb:equal (?a, ?b) ^ exehdaso:eventDstPort(?x, ?c) ^ exehdaso:vullPort(?y, ?d) ^ swrlb:equal(?y, ?d) ->exehdaso:PortScanEV(?x)

Figura 9. Regra : disposição de ativos em duas células

Na sequência, a ferramenta nmap, comumente empregada na primeira etapa de um teste de intrusão, é utilizada, para levantamento de informações sobre o alvo, no caso, o servidor webserver1 (WS_1). Uma das características dessa ferramenta é que ela fornece dados sobre as versões de serviços que estão em execução no dispositivo alvo. Logo, de posse dessa informação, o atacante pode valer-se das vulnerabilidades documentadas relativas às versões identificadas, especialmente se houver algum serviço desatualizado.

Como consequência desta execução, o NIDS Suricata registra os eventos que identificam esta tentativa de escaneamento de portas. Este registro de eventos emprega o formato JSON, o qual facilita a etapa de coleta por ser nativamente suportado pelo Filebeat. Além disso a etapa de pré-processamento é desonerada da normalização. Com isso, após contextualizados, os eventos são instanciados dentro da EXEHDA-SO na classe *Events* com dados que incluem origem e destino de endereçamentos, informações sobre a regra do Suricata que detectou o evento, entre outros.

Contado com estas informações de contextos independentes instanciadas nas classes *Vulnerability* e *Events* é aplicada uma regra apresentada na figura 9 que gera novas instâncias de alertas nas subclasses de *Situation*, que informam sobre a existência de uma situação de levantamento de informações de segurança em um serviço com versão exposta.

Estes alertas são utilizados pela etapa de Projeção, para envio de email para o analista onde é sugerido controle já documentado no campo solução proveniente da varredura executada pelo OpenVAS. Neste caso específico é sugerida a alteração dos parâmetros de configuração do Apache *ServerTokens* para Prod e *serverSignature* para Off.

V. Trabalhos Relacionados

Nesta seção estão destacados trabalhos que utilizam ontologias visando proporcionar algum nível de ciência de contexto, limitando o escopo do levantamento a trabalhos que tenham a SI como artifício ou finalidade.

O trabalho denominado *The Evaluation Process of a Computer Security Incident Ontology* [6] faz uso de ontologias dentro do domínio de SI, onde a denominada OntoSec objetiva o estabelecimento de uma estrutura única para representar informações sobre incidentes de segurança, possibilitando a correlação dos eventos percebidos. A OntoSec deriva diversos de seus conceitos do Projeto CVE *(Common Vulnerabilities and Exposures)*, que é um modelo de referência público de vulnerabilidades conhecidas mantido pelo MITRE[10]. Como fonte de eventos da ontologia são mapeadas automaticamente as detecções originalmente percebidas pelo Snort[11]. Uma vez mapeados os eventos na ontologia, são possíveis consultas na direção de diversos aspectos de interesse.

No trabalho [9] é apresentado um sistema com foco no diagnóstico e na detecção de intrusão, implementando uma correlação de processos híbrida e hierárquica para a detecção de cenários de intrusão. A capacidade de correlação de indícios de ataques é dirigida por uma base de conhecimento ontológica, capturando a relação causal entre as atividades maliciosas preliminarmente detectadas. A ontologia é composta por diversas entidades voltadas ao Monitoramento de Ataques, à

[10]MITRE - corporação americana sem fins lucrativos que coordena diversos centros de pesquisa do governo federal

[11]Snort - sistema *open-source* para detecção de intrusão em redes

Correlação, à Detecção de Intrusão e à Recuperação. Com as informações instanciadas na ontologia, o processamento dos cenários de ataques é efetuado por CEP, correlacionando as situações intermediárias encontradas. Uma vez detectados os cenários de ataque são então encaminhados alertas ao chamado agente remediador, vocábulo alusivo ao analista de sistemas. Os registros utilizados na prototipação são oriundos do IDS *Prelude*.

O trabalho [10] pontua a proposta voltada para a etapa de percepção de situações, sendo concebida uma ontologia. Esta ontologia considera os conceitos estabelecidos pelas taxonomias CVE (*Common Vulnerability Enumeration*), CWE (*Common Weakness Enumeration*), e CAPEC (*Common Attack Pattern Enumeration and Classification*), caracterizando o reuso da proposta. Na ontologia estão representadas as classes: Ator; Ataque; Rede; Vulnerabilidade. Ao receber como entrada as vulnerabilidades oriundas de uma ferramenta externa, a ontologia determina ou percebe o estado da rede, que pode ser "seguro", "vulnerável" ou "inseguro".

No trabalho [11] é proposta a utilização do *middleware Adapter* que tem por finalidade efetuar o mapeamento entre os dados de log armazenados e um modelo ontológico, bem como prover uma integração fácil para com a estrutura de logs por intermédio de consultas SPARQL. A ontologia é composta basicamente por 3 elementos: (i) uma terminologia usada no domínio de *Healthcare*; (ii) uma estrutura de logs oriundos dos sistemas; e (iii) uma política de segurança. A terminologia de *Heathcare* mune a ontologia de classes referentes a conceitos da área médica como "Relatórios de Consulta", "Dados de pacientes", etc. Quanto à estrutura de logs, o estudo de caso trabalha com logs ATNA (*Audit Trail and Node authentication*) especificado pelo padrão IHE (Integrating the Health care Enterprise) como entrada. O modelo de política de segurança, um terceiro elemento que compõe a ontologia, é adaptado de OrBAC [12] e de RBCA [13], modelos genéricos para controle de acesso.

Com intuito de comparar a EXEHDA-SO com os trabalhos relacionados descritos nesta seção, apresenta-se a tabela II. Nos quesitos "formalismo", "consultas" e "interação" foi mantido um alinhamento com os trabalhos relacionados. Entretanto, destaca-se que a EXEHDA-SO não limita-se a instâncias de uma determinada solução, como pode-se ver na coluna Eventos. Neste quesito observa-se que alguns trabalhos relacionados não informam as fontes de seus eventos enquanto outras utilizam apenas uma. A EXEHDA-SO propõe o uso de regras SWRL, as quais são mencionadas em apenas um trabalho relacionado. Registra-se ainda que o uso de CEP explorado no EXEHDA é compatível com a premissa operacional da EXEHDA-SO.

VI. Considerações Finais

No momento em que a informação passa a determinar desde vantagens estratégicas de mercado, direcionamentos políticos e inclusive guerras ao redor do mundo, é natural que a segurança destes dados passe a ser uma preocupação cada vez maior para todos. Este protagonismo da informação no cotidiano social é impulsionado pela materialização da UbiComp, que insere funcionalidades e serviços de forma transparente, atribuindo conectividade inclusive à dispositivos que tradicionalmente não contariam com nenhum tipo de inteligência computacional. Esta circunstância da UbiComp, que explicita o paradigma computacional denominado Internet das Coisas, propicia constantes cenários de exposição e riscos às informações.

Para que as premissas de UbiComp e IoT relacionadas sejam atendidas é requerido que os ambientes (i) garantam conectividade para um elevado número de nodos; (ii) processem e validem contextos de segurança; e (ii) sejam constantemente adaptados considerando as identificações de ações maliciosas. Voltadas para estes desafios, são recorrentemente implementadas diversas soluções focadas na estruturação de linhas de defesa de redes. Entretanto, estas soluções, por não terem finalidades iguais, muitas vezes não apresentam formatos de dados compatíveis entre sí, o que não significa que estas soluções não possam contribuir mutuamente para a construção de contexto, mas que dificulta o seu uso integrado. Sem uma constante avaliação sobre as métricas obtidas por essas soluções e sobre o tráfego que está sendo gerado, os controles implementados tendem a não alcançar o êxito esperado.

O exercício de adequação a cenários amplamente mutáveis e dinâmicos requer mecanismos voltados a CS onde identifica-se, para além da heterogeneidade das fontes de dados contextuais, oportunidades de contribuições direcionadas à distribuição da computabilidade, à diversidade de formatos nos eventos de SI disponíveis e a multiplicidade de situações que podem ser extraídas ao combinar eventos deste perfil.

Desta forma, em resposta a estas demandas, optou-se pelo uso de ontologias, dentre outras técnicas de processamento de contexto existentes, por proverem: (i) a análise de eventos independentemente dos formatos utilizados pelas soluções de fins específicos de SI; (ii) o reuso de conhecimento, facilitando a evolução das soluções, permitindo aproximação dos vocabulários ao *middleware* para o qual este trabalho traz contribuição e auxiliando na interoperabilidade (diferentes aplicações podem valer-se dos conceitos e relacionamentos definidos nas ontologias); e (iii) a visão unificada das situações identificadas em diferentes cenários de SI.

Por intermédio de ontologias, a EXEHDA-SO integrou-se à fase de processamento de eventos de segurança da EXEHDA-USM, contribuindo para a composição de uma estratégia híbrida voltada a camada de compreensão de CS. A alternativa integra-se tanto ao componente SmartLogger quanto ao Manager, que orquestram uma arquitetura multinível, que é observada em diversas organizações atualmente.

A estruturação da ontologia foi desenvolvida primeiramente relacionando alguns conceitos fundamentais de SI, os quais foram estipulados no fragmento denominado Core. Este fragmento transversaliza os demais, servindo como base para todo o conhecimento que é processado na ontologia. Já o fragmento denominado Scope Analyzer buscou representar o conhecimento necessário tipicamente ao Smartlogger, encaminhando adaptações aos cenários identificados com uma maior

Tabela II
COMPARAÇÃO ENTRE TRABALHOS RELACIONADOS E A EXEHDA-SO

	Formalismo	Consultas	Inferências	Regras	Repositório	Eventos	Interação
6	OWL	SparQL	-	-	MySQL	SNORT	Java Jena
9	-	-	-	CEP	-	Prelude	-
10	OWL	-	Hermit	SWRL	-	-	-
11	RDF/OWL	SparQL	-	-	-	IHE-ATNA	-
EXEHDA-SO	OWL	SparQL	Pellet	CEP e SWRL	RHIC (Virtuoso)	Diversos	Java Jena

proximidade ao trecho da arquitetura em questão. O modelo é então complementado pelo fragmento InterCell Analyzer, que visa relacionar os acontecimentos entre as diversas células que compõem o ambiente.

O estudo de caso buscou validar a estruturação ontológica apresentada no modelo, implementando mecanismos que possibilitaram a identificação de padrões de ataques configurando cenários de risco. Para tal, fez-se uso de técnicas já consolidadas no âmbito das ontologias, como consultas em SparQL e regras SWRL. Os testes do modelo foram executados em ambiente inspirado na Universidade Federal de Pelotas (UFPel), por possuir diversas características consonantes aos desafios de UbiComp, como elevada distribuição e heterogeneidade. Foram processados eventos coletados no ambiente quanto as suas origens e destinos, encaminhando contra-medidas aos cenários identificados, no intuito de demonstrar como as regras ontológicas podem contribuir para soluções mais abrangentes. Logo, cruzaram-se os dados dos eventos com a análise de vulnerabilidades, gerando uma visão sobre a criticidade das ações maliciosas no ambiente, e novamente indicando possíveis ações preventivas.

Destaca-se ainda que este trabalho enseja atenção futura sobre (i) a ampliação dos cenários de uso por intermédio de revisão e possíveis acréscimos ao vocabulário atual, (ii) o aumento no número de regras a serem aplicadas em busca de diferentes contextos e (iii) o desenvolvimento de módulo voltado para adaptação e manutenção da ontologia.

REFERÊNCIAS

[1] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 66–75, January 1991. [Online]. Available: http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

[2] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," in *In HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. Springer-Verlag, 1999, pp. 304–307.

[3] J. Lopes, R. Souza, G. Gadotti, A. Pernas, A. Yamin, and C. Geyer, "An architectural model for situation awareness in ubiquitous computing," *Latin America Transactions, IEEE (Revista IEEE America Latina)*, vol. 12, no. 6, pp. 1113–1119, Sept 2014.

[4] R. B. Almeida, "Exehda-usm: uma arquitetura hierárquica multinível consciente de situação aplicada a segurança da informação," Dissertação de Mestrado em Ciência da Computação, Programa de Pós-Graduação em Computação/UFPel, Pelotas - RS, 2016.

[5] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis, "Telos: Representing knowledge about information systems," *ACM Trans. Inf. Syst.*, vol. 8, no. 4, pp. 325–362, Oct. 1990. [Online]. Available: http://doi.acm.org/10.1145/102675.102676

[6] L. A. F. Martimiano and E. Moreira, "The evaluation process of a computer security incident ontology," in *THE 2ND WORKSHOP ON ONTOLOGIES AND THEIR APPLICATIONS, RIBEIRÃO PRETO*, 2006.

[7] A. Vorobiev and N. Bekmamedova, "An ontology-driven approach applied to information security." *Journal of Research and Practice in Information Technology*, vol. 42, no. 1, pp. 61–76, 2010. [Online]. Available: http://dblp.uni-trier.de/db/journals/acj/acj42.html#VorobievB10

[8] R. S. Machado, R. B. Almeida, D. Y. L. da Rosa, J. L. B. Lopes, A. M. Pernas, and A. C. Yamin, "Exehda-hm: A compositional approach to explore contextual information on hybrid models," *Future Generation Computer Systems*, vol. 73, pp. 1 – 12, 2017.

[9] M. Ficco and L. Romano, "A generic intrusion detection and diagnoser system based on complex event processing," in *Data Compression, Communications and Processing (CCP), 2011 First International Conference on*, June 2011, pp. 275–284.

[10] P. Bhandari and M. Gujral, "Ontology based approach for perception of network security state," in *Engineering and Computational Sciences (RAECS), 2014 Recent Advances in*, March 2014, pp. 1–6.

[11] H. Azkia, N. Cuppens-Boulahia, F. Cuppens, and G. Coatrieux, "Log content extraction engine based on ontology for the purpose of a posteriori access control," *IJKL*, vol. 9, no. 1/2, pp. 23–42, 2014. [Online]. Available: http://dx.doi.org/10.1504/IJKL.2014.067149

[12] N. Cuppens-Boulahia, F. Cuppens, J. de Vergara, E. Vazquez, J. Guerra, and H. Debar, "An ontology-based approach to react to network attacks," in *Risks and Security of Internet and Systems, 2008. CRiSIS '08. Third International Conference on*, Oct 2008, pp. 27–35.

[13] D. Ferraiolo and R. Kuhn, "Role-based access control," in *In 15th NIST-NCSC National Computer Security Conference*, 1992, pp. 554–563.