EXEHDA-SO: Uma Abordagem Ontológica para Ciência de Situação Aplicada a Segurança da Informação

Diórgenes Y. L. da Rosa[1], **Roger S. Machado**[1], **Ricardo B. Almeida**[1], **Adenauer C. Yamin**[1], **Ana Marilza Pernas**[1]

[1]Programa de Pós-Graduação em Computação
Universidade Federal de Pelotas (UFPel), Pelotas – RS – Brasil

{diorgenes,rdsmachado,rbalmeida,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.*

Resumo. *As infraestruturas típicas da Computação Ubíqua pressupõem características de flexibilidade quanto à conectividade nos ambientes. Visando segurança nestes cenários, diversas soluções são implantadas em linguagens de sintaxe própria, provendo eventos em formatos distintos. Neste sentido, a Ciência de Situação (CS), enquanto estratégia capaz de integrar eventos de diferentes fontes, torna-se requisito para a implementação de controles. Este trabalho propõe uma abordagem ontológica para CS aplicada ao domínio de segurança da informação, denominada EXEHDA-SO. A proposta foi avaliada com base em uma infraestrutura real, se mostrando capaz de tratar eventos heterogêneos provenientes de diferentes fontes contextuais.*

1. Introdução

Para atender as prerrogativas de Computação Ubíqua [Weiser 1991], 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 possível de bloqueios e controles. 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. O emprego destas diferentes soluções de SI trazem empecilhos

quanto à 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 decisão para proteção da infraestrutura. Dey (1999) afirma 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) [Lopes et al. 2014] e [Almeida 2016].

A próxima seção detalha a proposta em seus fragmentos ontológicos, a seção 3 trás um cenário de uso no qual a proposta foi aplicada, a seção 4 aborda alguns trabalhos relacionados e por fim apresentam-se as conclusões na seção 5

2. 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 colaboração ao *middleware* EXEHDA [Lopes et al. 2014] e derivado trabalho voltado à SI desenvolvido em [Almeida 2016], integrando às funcionalidades de identificação de situações de interesse, focada em ampliar as possibilidades da camada de compreensão de CS. A proposta EXEHDA-SO é apresentada em sua totalidade em [da Rosa 2017].

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 [Machado et al. 2017] e o processamento de eventos por meio de regras [Almeida 2016]. O EXEHDA-SO propõe 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.

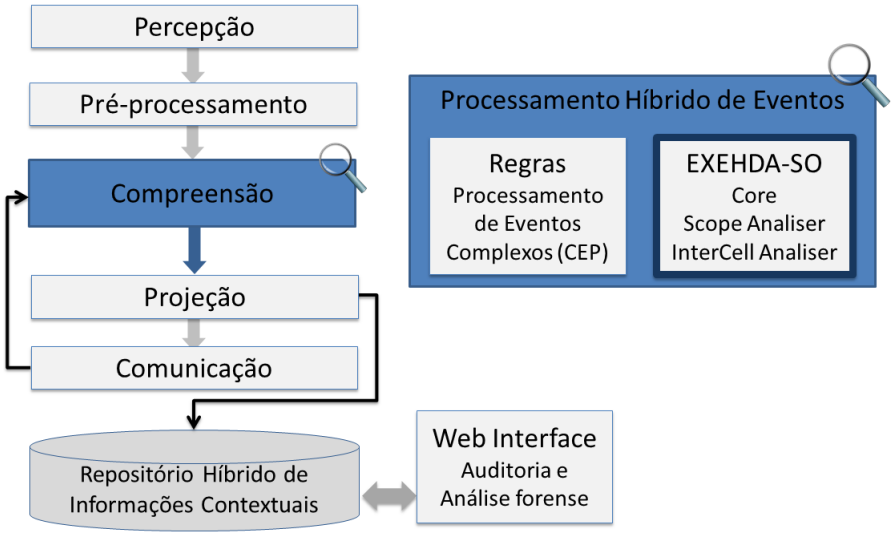

Figura 1. EXEHDA-SO integrando a etapa de Compreensão em CS

2.1. 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 à 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

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

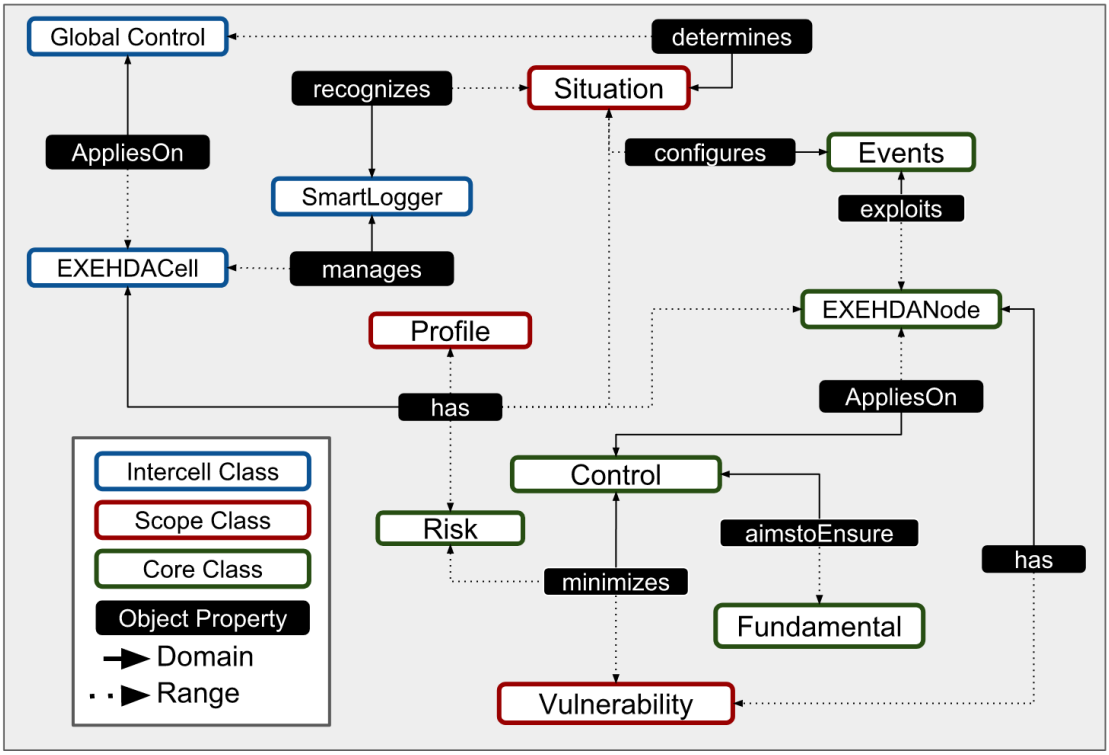

Figura 2. Principais classes da EXEHDA-SO

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 à 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 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, materi-

alizando 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 a 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.

2.2. 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 às 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 à 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.

2.3. 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 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.

3. Cenário de Uso

Proteger o ambiente a partir da análise de origem e destino dos eventos detectados é um dos eixos centrais desse cenário de uso. 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 pela rede interna, entre outras alternativas que objetivem a mitigação do risco).

Para demonstrar esta identificação, primeiramente é realizada a coleta do log oriundo do HIDS (*Host-based Intrusion Detection System*) OSSEC *(Open Source Security)* que está configurado em modo *Standalone* em um servidor, nomeado webserver2. Observa-se que um evento similar é registrado em um segundo servidor, webserver1, diferindo apenas o IP e *hostname* do servidor de destino e o momento da ocorrência.

Este evento reflete um ataque de Força Bruta, 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[2] disponível nativamente na distribuição linux Kali [3].

Os *logs* que caracterizam os eventos de Força Bruta são então tratados pelo componente *Collector* 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 expressão regular.

Após executadas estas tarefas, os eventos são instanciados na classe *Events* da ontologia. Observa-se que nem todos os campos são 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 os eventos 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) o serviço alvo; (iii) o usuário que está sendo utilizado pelas tentativas de acesso; e (iv) a prioridade do evento.

A abordagem 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 webserver1 são encaminhados para a classe que o representa na hierarquia de ativos *(EXEHDANode)*. Esta regra para um servidor web pode ser visualizada na Figura 3. A regra repete-se para cada servidor previsto no ambiente.

```
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 3. Regra de encaminhamento dos eventos para a classe que representa o ativo alvo

A tarefa de perceber que muitos eventos apresentam uma mesma origem para mais de um destino no ambiente é executada pela regra da Figura 4. Nesta regra SWRL, a origem dos eventos identificados nos dois servidores web é comparada e apresentaram 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 implica a necessidade de adaptação do ambiente ubíquo, neste caso, promovendo o bloqueio do IP do atacante. Neste momento, é inferido que a recorrência deste atacante

[2] https://www.thc.org/thc-hydra/
[3] https://www.kali.org/

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

Figura 4. Regra para identificação de origens únicas em muitos destinos

no ambiente indica a possibilidade do mesmo realizar novas tentativas de acesso indevido nesta célula. Scripts personalizados são executados para efetuar a ação na fase Projeção de CS.

A Figura 5 mostra as classes do modelo ontológico aplicadas neste cenário. A completude dos testes e cenários abordados pode ser vista em [da Rosa 2017].

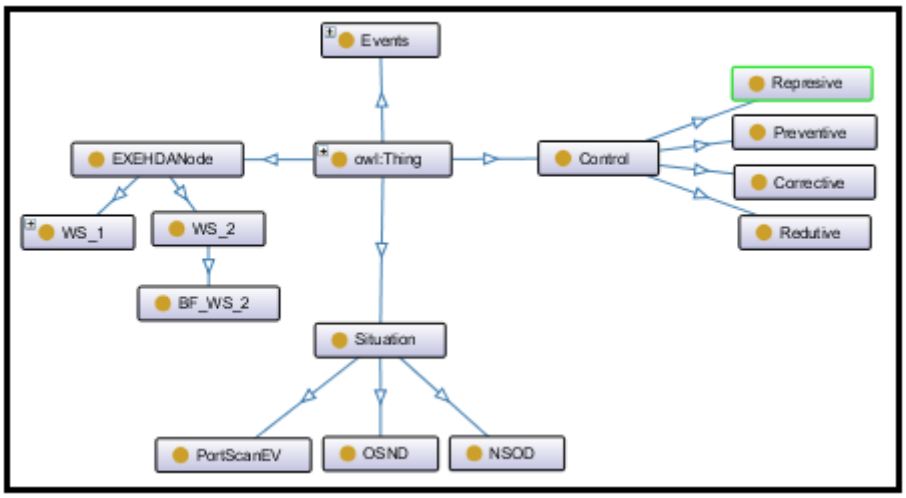

Figura 5. Classes envolvidas no processo de análise de endereçamento

4. 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.

No trabalho [Ficco and Romano 2011] 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 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 [Bhandari and Gujral 2014] 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 [Azkia et al. 2014] 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 *Healthcare* 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 [Cuppens-Boulahia et al. 2008] e de RBCA [Ferraiolo and Kuhn 1992], modelos genéricos para controle de acesso.

Com intuito de comparar a EXEHDA-SO com os trabalhos relacionados descritos nesta seção, apresenta-se a tabela 1. 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.

Tabela 1. Comparação entre trabalhos relacionados e a EXEHDA-SO

	Formalismo	Consultas	Inferências	Regras	Repositório	Eventos	Interação
FICCO; ROMANO, 2011	-	-	-	CEP	-	Prelude	-
BHANDARI; GUJRAL, 2014	OWL	-	Hermit	SWRL	-	-	-
AZKIA et al., 2014	RDF/OWL	SparQL	-	-	-	IHE-ATNA	-
EXEHDA-SO	OWL	SparQL	Pellet	CEP e SWRL	RHIC (Virtuoso)	Diversos	Java Jena

5. Considerações Finais

A adequação a cenários amplamente dinâmicos requer mecanismos voltados a CS, onde contribuições podem ser direcionadas a distribuição da computabilidade, a diversidade de formatos nos eventos de SI e a multiplicidade de situações que podem ser extraídas na combinação de eventos distintos. Por intermédio das ontologias, a EXEHDA-SO contribui para a estratégia de Compreensão de CS por meio do provimento de um vocabulário

que recebe instâncias de dados contextuais de diversas fontes, processando-as de forma descentralizada, unificando formatos e provendo indicações de contra-medidas passivas ou promovendo adaptação do ambiente ubíquo por intermédio de ações ativas.

A continuidade da pesquisa é identificada nos seguintes aspectos: (i) ampliação do número de cenários de uso; (ii) aumento do número de regras aplicadas; (iii) desenvolvimento de um módulo voltado à adaptação e manutenção da ontologia; e (iv) exploração da interação entre as técnicas de CEP e Ontologias.

Agradecimentos

O presente trabalho foi realizado com apoio da CAPES (Programa Nacional de Cooperação Acadêmica - Procad) e da FAPERGS (Programa Pesquisador Gaúcho - PqG).

Referências

Almeida, R. B. (2016). 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.

Azkia, H., Cuppens-Boulahia, N., Cuppens, F., and Coatrieux, G. (2014). Log content extraction engine based on ontology for the purpose of a posteriori access control. *IJKL*, 9(1/2):23–42.

Bhandari, P. and Gujral, M. (2014). Ontology based approach for perception of network security state. In *Engineering and Computational Sciences (RAECS), 2014 Recent Advances in*, pages 1–6.

Cuppens-Boulahia, N., Cuppens, F., de Vergara, J., Vazquez, E., Guerra, J., and Debar, H. (2008). An ontology-based approach to react to network attacks. In *Risks and Security of Internet and Systems, 2008. CRiSIS '08. Third International Conference on*, pages 27–35.

da Rosa, D. Y. L. (2017). EXEHDA-SO: Uma abordagem ontológica para ciência de situação aplicada ao domínio de segurança da informação. Dissertação de mestrado em ciência da computação, Programa de Pós-Graduação em Computação/UFPel.

Dey, A. K. and Abowd, G. D. (1999). Towards a better understanding of context and context-awareness. In *HUC '99*, pages 304–307. Springer-Verlag.

Ferraiolo, D. and Kuhn, R. (1992). Role-based access control. In *In 15th NIST-NCSC National Computer Security Conference*, pages 554–563.

Ficco, M. and Romano, L. (2011). A generic intrusion detection and diagnoser system based on complex event processing. In *Data Compression, Communications and Processing (CCP), 2011 First International Conference on*, pages 275–284.

Lopes, J., Souza, R., Gadotti, G., Pernas, A., Yamin, A., and Geyer, C. (2014). An architectural model for situation awareness in ubiquitous computing. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 12(6):1113–1119.

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

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):66–75.