SECURITY POLICY INSTANTIATION TO REACT TO NETWORK ATTACKS

An Ontology-based Approach using OWL and SWRL

Jorge E. López de Vergara
Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid
Francisco Tomás y Valiente, 11, E-28049 Madrid, Spain

Enrique Vázquez and Javier Guerra
Departamento de Ingeniería de Sistemas Telemáticos, E.T.S.I. de Telecomunicación, Universidad Politécnica de Madrid
Av. Complutense, s/n, E-28040 Madrid, Spain

Keywords: Attack reaction, policy instantiation, ontology, OrBAC, IDMEF, OWL, SWRL.

Abstract: A quick and efficient reaction to an attack is important to address the evolution of security incidents in current communication networks. The ReD (Reaction after Detection) project's aim is to design solutions that enhance the detection/reaction security process. This will improve the overall resilience of IP networks to attacks, helping telecommunication and service providers to maintain sufficient quality of service to comply with service level agreements. A main component within this project is in charge of instantiating new security policies that counteract the network attacks. This paper proposes an ontology-based methodology for the instantiation of these security policies. This approach provides a way to map alerts into attack contexts, which are later used to identify the policies to be applied in the network to solve the threat. For this, ontologies to describe alerts and policies are defined, using inference rules to perform such mappings.

1 INTRODUCTION

Currently, Internet has become an attractive mean for service delivery. Network operators and providers are supplying new services such as voice or television based on the Internet Protocol (IP). Unfortunately, this has also increased the number of IP-based attacks to network systems. To address the evolution of security incidents in current communication networks it is important to react quickly and efficiently to an attack. This reaction can range from blocking the traffic to defining new security policies that solve the problem if they are applied as long as the attack is happening.

The ReD (Reaction after Detection) project (http://projects.celtic-initiative.org/red/) is defining and designing solutions to enhance the detection/reaction process, improving the overall resilience of IP networks to attacks and help telecommunication and service providers to maintain sufficient quality of service and respect service level agreements. This project has defined the architecture of a ReD node, in charge of finding the best way to react to an attack, both in the short and long terms. Within this node, a main component (named policy instantiation engine) is devoted to the instantiation of new security policies that counteract the network attacks. This node has to deal with the mapping of the alerts about attacks in the network into these security policies, providing the most suitable policy to reduce or even eliminate the problems caused in the network.

This paper proposes an ontology-based approach to instantiate these security policies. This technology provides a way to map alerts into attack contexts, which are used to identify the policies to be applied in the network to solve the threat. For this, ontologies to describe alerts and policies are defined, using inference rules to perform such mappings.

The use of ontologies for defining policies is not new. For instance, KAoS (Bradshaw *et al.*, 2003) and Rei (Kagal *et al.*, 2003) are well known policy frameworks based on ontologies. On the other side, attack ontologies have also been described before. Examples of them can be found in (Undercoffer *et*

E. López de Vergara J., Vázquez E. and Guerra J. (2008).
SECURITY POLICY INSTANTIATION TO REACT TO NETWORK ATTACKS - An Ontology-based Approach using OWL and SWRL.
In *Proceedings of the International Conference on Security and Cryptography*, pages 78-83
DOI: 10.5220/0001929300780083

al. 2003) and (Geneiatakis *et al.*, 2007), providing a formal modelling of network attacks. However, although we also use ontologies, the approach proposed in this paper is focused on a new problem: the mapping from attack alerts to security policies. For this, we adapt in this scope the translation process presented in (Guerrero *et al*., 2006). One interesting property of ontologies is the ability to deal with several information models, allowing an easy integration of them. The work presented here exploits this capacity to integrate in the same view alert and policy instances, making the mapping process feasible.

The rest of this paper is structured as follows. First of all, the technologies used in ReD project are presented. Then, the mapping steps to perform the policy instantiation are defined. Next, the ontology-based representation of OrBAC (Organization Based Access Control) and IDMEF (Intrusion Detection Message Exchange Format) are provided, as well as mapping rules to derve information from IDMEF to OrBAC. Finally, some conclusions are given.

2 RED DESCRIPTION

Following sections present the ReD architecture, as well as some technologies used in it. This information will be valuable to understand the solution addressed in this paper.

2.1 ReD Architecture

To provide the detection and reaction functionalities, ReD proposes a node containing a set of elements, depicted in Figure 1:

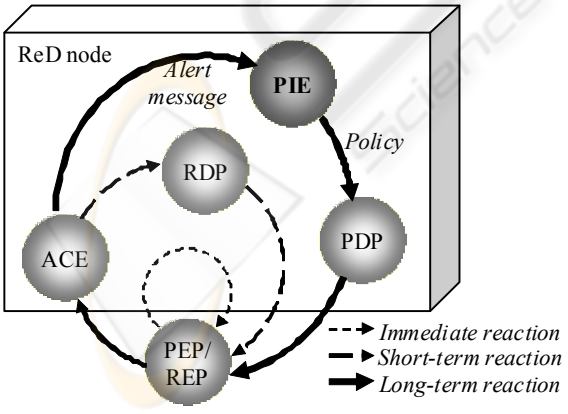

Figure 1: ReD architecture.

- ACE (Alert Correlation Engine): this element is in charge of receiving alerts from network nodes, and enhances the detection of attacks by combining several diagnosis combinations.
- PIE (Policy Instantiation Engine): this element receives the information about attacks from the ACE and instantiates new security policies to react to the attack in a long-term reaction loop. This paper is focused on this element.
- PDP (Policy Decision Point): this element receives the new security policies defined by the PIE and deploy them in the enforcement points.
- RDP (Reaction Decision Point): this element receives the information about attacks from the ACE and decides a reaction in a short-term reaction loop.
- PEP/REP (Policy Enforcement Point/Reaction Enforcement Point): This component, outside the ReD node, enforces the security policies provided by the PDP and reaction provided by the RDP. It also performs an immediate reaction.

As stated when defining the RED components, three type of reactions have been defined, based on the time when they are applied: immediate reactions are directly decided by the PEP/REP, short-term reactions are decided by the RDP based on the information provided by the ACE and without instantiating new security policies, and finally, long-term reactions are decided by the PIE, generating new security policies based on the ACE alerts that are passed to the PDP to deploy them in the PEPs.

2.2 Alerts and Policies in ReD

To propagate alerts from the PEP to the ACE, and from the ACE to the REP and PIE, messages are sent using the Intrusion Detection Message Exchange Format (IDMEF) (Debar *et al*., 2007). This format, based on XML, is defined to report alerts about events in an Intrusion Detection System, and it is composed of a set of tags that describe the different properties that an alert can contain, such as timestamps, source and target of the attack, and its classification. At this point, it is important to remark that the vulnerability exploited is commonly used to classify the attacks, using CVE (Common Vulnerabilities and Exposures) identifiers (Martin, 2001).

Several languages can be used to define the security policy, including Policy Core Information Model (PCIM) (Moore *et al.*, 2001), Ponder (Damianou *et al*. 2001), or Organization Based Access Control (OrBAC) (Abou-El-kalam *et al.*, 2003). After an analysis of existing policy languages, OrBAC was selected as the policy language to be used

in ReD because it is expressive enough to specify a large variety of security requirements. In fact, it has been successfully applied to specify network access control policies and translation mechanisms have been defined to automatically generate firewall configuration rules that are free of inconsistency and redundancy.

In OrBAC, security policies are specified as sets of security requirements that correspond to permissions, prohibitions or obligations. Moreover, in OrBAC, these requirements may depend on contexts that express temporal, location or provisional conditions, which is very useful to specify new security requirements to be triggered in reaction to an intrusion. Contexts are used in ReD to express also an attack condition, this is, when an attack happens, a context about that attack is activated. Then, given a context attack from a source to a target, and about an action, an OrBAC condition is held, activating new security rules.

2.3 Ontologies: OWL and SWRL

An ontology is defined in (Gruber, 1993) as "a formal specification of a shared conceptualization". In practical terms, an ontology is a hierarchy of concepts with attributes and relations that defines a terminology to define in consensus semantic networks of inter-related information units. An ontology provides a vocabulary of classes and relations to describe a domain, stressing knowledge sharing and knowledge representation.

With the advent of the Semantic Web, OWL (Smith *et al*., 2004), the Web Ontology Language has gained relevance. It is a general purpose ontology language defined for the Semantic Web that contains all the necessary constructors to formally describe classes and properties, with hierarchies, and range and domain restrictions. An OWL ontology contains a sequence of axioms and facts. It includes several types of axioms, such as subclass axioms, equivalent Class axioms and property constraints.

The Semantic Web Rule Language (SWRL) (Horrocks *et al*., 2004) extends OWL with rule axioms. In the "human-readable" syntax of SWRL, a rule axiom has the form: antecedent → consequent. Informally, a rule may be read as meaning that if the antecedent holds, then the consequent must also hold. Using this syntax, a rule asserting that the composition of parent and sister properties implies the aunt property would be written:

$$Father(?x,?y) \wedge Sister(?y,?z) \rightarrow Aunt(?x,?z)$$

In this work, OWL will be used as the ontology language to describe both IDMEF and OrBAC concepts, and SWRL will be used to map information from IDMEF to OrBAC.

3 POLICY INSTANTIATION ENGINE PROCESS

Once the ReD main concepts have been introduced, the goal of this section is to describe the way IDMEF alerts are mapped into OrBAC contexts. For this, three steps have been defined:

1. First of all, it is necessary to define ontologies both for the IDMEF alerts (AO, Alert Ontology) and OrBAC (OO, OrBAC Ontology).
2. Then, it is also necessary to establish relationships between AO and OO with OWL properties and SWRL rules. OWL relationships are defined to link an attack to a context. SWRL rules can be used to map from one ontology to the other thanks to the established relationships. SWRL rules can then be considered as meta-policies, because they allow the creation of policies. The SWRL rules used for mapping will look like:
   $AOproperty1(?classAindividual) \wedge interoperablityrelationship1(?classOindividual, ?classAindividual) \rightarrow OOProperty1(?classOindividual)$
3. Finally, an inference engine can be used to derive the necessary information mappings in order to add data to OO based on the information of the AO.

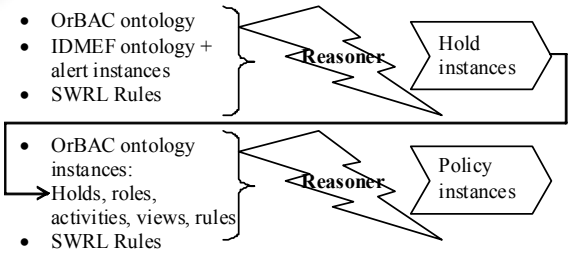

Figure 2: Policy instantiation with ontologies.

Figure 2 shows the information involved in the policy instantiation when ontologies are used. An inference engine needs the OrBAC and the Alerts ontology, new alert instances as well as SWRL rules to map those alerts into OrBAC hold instances, which are selected thanks to the reasoning process (upper side of the figure). These OrBAC hold instances are then used to obtain the security policy rules to be activated (lower side of the figure). It should be noted that this last operation can be ob-

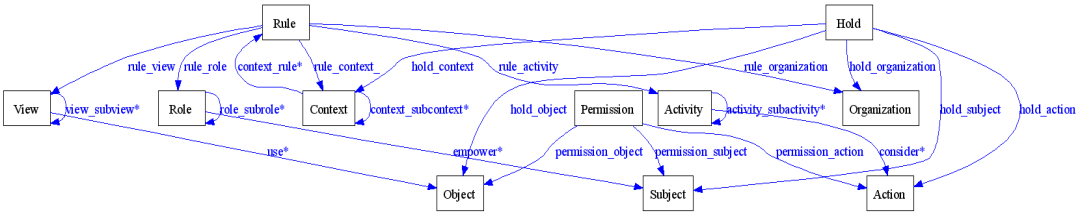

Figure 3: OrBAC ontology.

tained directly using MotOrBAC (the OrBAC engine implemented at Telecom Bretagne, http://motorbac.sourceforge.net/), but our approach is complete in the sense that given an alert message, policy rules are generated by using the set of ontology classes and instances, as well as the SWRL rules.

It is also important to remark that for this process it is necessary to convert existing information into ontologies, including OrBAC contexts and rules and IDMEF alerts. With respect to OrBAC, in this approach it is supposed that these contexts and rules are defined using an ontology editor by the manager of the system. However, IDMEF alerts are coming to the system in XML messages. Then, an IDMEF parser has been implemented that translates these alerts into instances of the Alerts ontology. Another issue is to translate OrBAC rules to be deployed in the PEPs, but this problem is independent of using ontologies.

4 REPRESENTING INFORMATION WITH ONTOLOGIES

As stated above, a first step in the policy instantiation process is to define ontologies for both OrBAC and IDMEF alerts. Next sections provide such definitions.

4.1 OrBAC Ontology Representation

An OrBAC context ontology has been presented in (Coma *et al.*, 2007). Nevertheless, the work presented here is different, as it is not focussed on the definition of different types of contexts. In this paper we define a set of classes to describe in an ontology the constructions used in OrBAC (see Figure 3):

- *View*: this class models the views contained in OrBAC. It has a relationship with an object, and a view can be derived from another view.
- *Object*: this class specifies the objects contained in OrBAC.
- *Role*: this class models the roles contained in OrBAC. It has a relationship with a subject, and a role can be derived from another role.
- *Subject*: this class specifies the subjects contained in OrBAC.
- *Activity*: this class specifies the activities contained in OrBAC. It has a relationship with an action, and an activity can be derived from another activity.
- *Action*: this class models the objects contained in OrBAC.
- *Organization*: this class models the views contained in OrBAC.
- *Hold*: this class specifies the OrBAC holds. A hold will have a subject, an object, an action, a context and an organization to be asserted.
- *Context*: this class specifies the contexts contained in OrBAC. A context will have a name. Four auxiliary properties have been defined: the first one, *context_actived*, provides which alerts activate this context, based on the CVE or BugTrack classification of the alert. The second one is used to know if the context is active or not. *Context_granularity* indicates the mapping strategy when receiving an alert, taking the information for *Subject, Object* and/or *Action.* Finally, *context_level* indicates the level of mapping, which can be *system*, *network* or *application*. Depending on such level, the information taken from the IDMEF message in the mapping will be different.
- *Rule*: this class models the rules contained in OrBAC. A rule will have a number, a type (permission, prohibition, obligation), and relationships with a context, a view, a role, an activity and an organization. An auxiliary property has been included to know if a rule is active or not.
- *Permission*: this class models the concrete policies. That is, if an action is permitted, prohibited or obligated for a subject on an object.

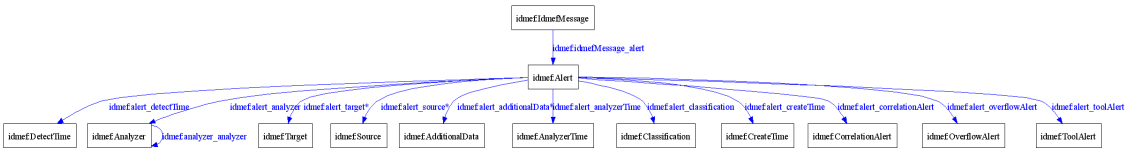

Figure 4: Alert ontology (alert relationships).

4.2 IDMEF Ontology Representation

With respect to the classes of the alerts ontology, our approach tries to reduce as much as possible the translation problems from real IDMEF alerts to the alert ontology. Thus, an alert ontology has been defined that maps the IDMEF alerts structure. This means that the alerts ontology is much more complex than the OrBAC ontology, with many more classes and relationships.

For instance, the *Alert* class, depicted in Figure 4, has relationships with *Message*, *CreateTime*, *AnalyzerTime*, *DetectTime*, *Source*, *Target*, *AdditionalData*, *Assessment*, *Analyzer*, *Classification*, *CorrelationAlert*, *OverflowAlert*, and *ToolAlert*. Then, *Source* and *Target* have also some relationships with other classes such as *User*, *Process*, *Service* and *Node*, which has an *Address*.

5 LOGIC RULES DEFINITION

Once the ontologies have been defined, it is also important to define the logic rules to be used in the inference engine. Three types of rules have been defined: rules to infer hierarchy information in the OrBAC model, rules to perform the mapping from IDMEF alerts to OrBAC holds, and rules to obtain the security policies.

With respect to the first type of rules, based on the relationships defined in section 4.1, it is possible to infer information from the hierarchy model of ORBAC. For instance, for *Role* and *Subrole*, the following rule is defined to empower a subject for subrole, given that it empowers its parent role:

orbac:Role(?role) ^ orbac:role_subrole(?role, ?subrole) ^ orbac:empower(?role, ?subject) → orbac:empower(?subrole, ?subject)

Similar rules are defined for *Activity* and *Subactivity* with respect to *Action*, as well as for *View* and *Subview* with respect to *Object*.

To perform the mapping from an IDMEF alert message to an OrBAC hold (see Figure 2), properties *context_actived, context_granularity* and *context_level* are taken into account:

- *context_actived*. This property will be used to select the context from a list of existing contexts, by comparing it with the IdmefMessage. Alert.Classification.Reference.name. It is necessary to populate this property in all context instances. Then, if more than one context is activated with an alert, both contexts will hold and a conflict resolution should be defined.
- *context_granularity* and *context_level*. Different rules will be defined based on the granularity, to map information of the alert to a *Subject*, an *Object*, or/and an *Action*. At the same time, these rules will map a specific part of the IDMEF message based on the level. For instance, if the level is system, then, *Object.name* is obtained from *IdmefMessage. Alert.Target.Process.name*, and *Subject.name* is taken from *IdmefMessage. Alert.Source.User.UserId.name*.
  If the level is network, *Object.address* is derived from *IdmefMessage. Alert.Target.Node.Address.address* and *Subject.address* is obtained from *IdmefMessage.Alert.Source.Node.Address. address*.
  Finally, if the level is application, *Object. name* is taken from *IdmefMessage. Alert.Target.Node.name* and *Subject.name* is derived from *IdmefMessage. Alert.Source.Node.name*

As stated, a set of SWRL rules are defined for the mappings, following the approach proposed in section 2.3. The SWRL rules are more complex in this case because they have to follow the structure of the IDMEF message. The rules are not presented here due to space constraints, but to show a little example of their verbosity, to reference the IdmefMessage.Alert.Classification.Reference.name property the following atoms have to be defined:

idmef:IdmefMessage(?im) ^
idmef:idmefMessage_alert(?im, ?a) ^
idmef:alert_classification(?a, ?c) ^
idmef:classification_reference(?c, ?r) ^
idmef:reference_name(?r, ?rn)

Note that SWRL does not have any built-in function to generate a new instance, which is necessary to generate new *Hold* instances. To solve this prob-

lem the swrlx:makeOWLThing function has been used, as proposed in (Protégé, 2007).

Once a *Hold* is obtained, and based on the defined instances of *Role*, *Activity* and *View*, it is also possible to obtain the policy security rules, indicating permission, prohibition and obligation. For this, SWRL rules are also defined. In this case, their complexity is caused because these rules resemble the behaviour of MotOrBAC.

6 CONCLUSIONS

This paper has presented a new approach to react to network attacks in which a set of alerts are used to automatically generate new security policies. For this, an ontology-based approach has been used, in which the alert and the policy information models can be combined to help in the instantiation of the policies. The use of OWL and SWRL provides several advantages, including the availability of generic tools for parsing and reasoning. The nature of OWL as a Semantic Web component allows merging distributed ontologies. Moreover, there are many works on mapping different knowledge bases that can be leveraged for our purpose.

Some issues have also been found when doing the experimentation. The expressivity of SWRL is limited, not allowing some logic operators, such as OR or NOT. Moreover, SWRL supports only monotonic inference, which means that SWRL rules cannot be used to modify or remove information in the ontology. This fact reduces the possibility to invalidate OrBAC holds when a new hold is asserted.

Future works include the validation of the implemented prototype in a test scenario, and integrating the PIE with the rest of the components in ReD architecture.

ACKNOWLEDGEMENTS

This paper has been partially funded by the European CELTIC Project ReD (CP3-011). The authors would like to thank the rest of the partners for their opinions and ideas.

REFERENCES

A. Abou-El-kalam, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, R. El-Baida, A. Miège, C. Saurel, G. Trouessin, 2003. Organization based access control. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks (POLICY 2003)*, Lake Como, Italy.

J. M. Bradshaw, A. Uszok, R. Jeffers, N. Suri, P. Hayes, M. H. Burstein, A. Acquisti, B. Benyo, M. R. Breedy, M. Carvalho, D. Diller, M. Johnson, S. Kulkarni, J. Lott, M. Sierhuis, R. Van Hoof, 2003. Representation and reasoning for DAML-based policy and domain services in KAoS and Nomad. In *Proc. Autonomous Agents and Multi-Agent Systems Conference (AAMAS 2003)*, Melbourne, Australia.

C. Coma, N. Cuppens-Boulahia, F. Cuppens, 2007. A context ontology based approach for secure interoperability. In *2007 Workshop of HP Software University Association. HP SUA*. Munich, Germany.

N. Damianou, N. Dulay, E. Lupu, M. Sloman, 2001. The Ponder Policy Specification Language. In *Workshop on Policies for Distributed Systems and Networks (POLICY2001). Lecture Notes in Computer Science*, Vol. 1995.

H. Debar, D. Curry, B. Feinstein, 2007. *The Intrusion Detection Message Exchange Format (IDMEF)*. IETF Request for Comments 4765.

D. Geneiatakis, C. Lambrinoudakis, 2007. An ontology description for SIP security flaws. *Computer Communications*, Vol. 30, Issue 6, pp. 1367-1374

T. R. Gruber, 1993. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, Vol. 5, No. 2, pp. 199-220

A. Guerrero, V. Villagrá, J.E. López de Vergara, A. Sánchez-Macián, J. Berrocal:, 2006. Ontology-based Policy Refinement Using SWRL Rules for Management Information Definitions in OWL. *Lecture Notes in Computer Science*, Vol. 4269, pp. 227-232

I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, 2004. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C Member Submission.

L. Kagal, T. Finin, A. Johshi, 2003. A Policy Language for Pervasive Computing Environment. In *Proceedings of IEEE 4th International Workshop on Policies for Distributed Systems and Networks (POLICY 2003)*, Lake Como, Italy.

R.A. Martin, 2001. Managing Vulnerabilities in Networked Systems. *IEEE Computer Magazine*, Vol. 34, No. 11, pp. 32-38

B. Moore, E. Elleson, J. Strassner, A. Westerinen, 2001. *Policy Core Information Model - Version 1 Specification*. IETF Request For Comments 3060.

Protégé, 2007. *Extensions Built-ins Library*, Stanford University. Available at http://protege.cim3.net/cgi-bin/wiki.pl?SWRLExtensionsBuiltIns.

M. K. Smith, C. Welty, D. L. McGuinness, 2004. *OWL Web Ontology Language Guide*. W3C Recommendation.

J. Undercoffer, A. Joshi, A. Pinkston, 2003. Modeling computer attacks: an ontology for intrusion detection, *Lecture Notes in Computer Science*, Vol. 2820, pp. 113-135.