Network Access Control Interoperation using Semantic Web Techniques

William M. Fitzgerald[1,2], Simon N. Foley[1], and Mícheál Ó Foghlú[2]

[1] Department of Computer Science, University College Cork, Ireland.
s.foley@cs.ucc.ie
[2] Telecommunications Software & Systems Group, Waterford Institute of Technology, Ireland
{wfitzgerald,mofoghlu}@tssg.org

Abstract. Network Access Control requirements are typically implemented in practice as a series of heterogeneous security-mechanism-centric policies that span system services and application domains. For example, a Network Access Control (NAC) policy might be configured in terms of firewall, proxy, intrusion prevention and user-access policies. While defined separately, these policies may interoperate in the sense that the access requirements of one may conflict and/or be redundant with respect to the access requirements of another policy. Thus, managing a large number of distinct policies becomes a major challenge in terms of deploying and maintaining a meaningful and consistent configuration. It is argued that the Semantic Web—an architecture that supports the formal representation, reasoning and sharing of heterogeneous domain knowledge—provides a natural solution to this challenge. A risk-based approach to configuring interoperating policies is described. Each NAC mechanism has an ontology that is used to represent its configuration. This heterogeneous and interoperating policy knowledge is unified with higher-level business (risk) rules, providing a single (extensible) ontology that supports reasoning across the different *NAC* policy configurations.

1 Introduction

While application-level services may provide their own access controls, it is standard practice to deploy additional 'layers' of security, such as firewalls, proxies, and so forth. In practice, security requirements are implemented as a series of independent security-mechanism-centric policies that span multiple system services and application domains. As a consequence, proper application operation is dependent on each security policy. An overly-restrictive policy configuration may prevent normal interaction of services with the resulting failure of the application. An overly-permissive policy configuration, while permitting normal operation of the application, may render the layers of security ineffective; a service hosted on a subnet is at greater risk of compromise if the firewall is configured with an open-access policy.

In this paper we are interested in Network Access Control (NAC) policies. Configuration of NAC policies can be complex, for example, a firewall policy may run to many thousands of rules and are typically maintained on an ad-hoc basis [1, 2]. New access control rules are often added to access control policies with little regard to how

they interoperate with existing rules and likely resulting in an overly-restrictive and/or overly-permissive configuration. Similarly, changes to the policy of one NAC mechanism (for example, a firewall) may indirectly impact the intent of a policy of another NAC mechanism (for example, a proxy). The ideal NAC configuration provides for consistent interoperating NAC policies that support valid service traffic, and, preferably, no more and no less.

The vision of the Semantic Web is the ability to express World Wide Web information in a natural and formal language that can be interpreted by intelligent agents, thus permitting them on behalf of the human user to locate, share and integrate information in an automated way. It provides a framework for dynamic, distributed and extensible structured knowledge (ontology) founded on formal logic [3, 4]. An ontology is an explicit specification of a conceptualization using an agreed vocabulary and provides a rich set of constructs to build a more meaningful level of knowledge. Ontologies and their associated reasoners are the building blocks of the Semantic Web initiative.

We argue that the Semantic Web framework provides a natural approach to constructing, reasoning about and managing security policies: a security policy can be regarded as an explicit specification of terminological knowledge regarding security mechanism configuration, that is, an ontology. Separate ontologies can be developed for different security policies that are naturally composable under the open-world assumption, providing a unified view of the enterprise-wide policy configuration. It provides for separation of concerns, whereby security concerns (security policies) and business concerns (business policies) can be separately developed, with reasoning and deployment over their composition. It also means that information about new mechanisms/policies and vulnerabilities can be incorporated as new facts within the existing policy knowledge-base.

This paper describes a risk-based model for NAC policy interoperation. A series of ontologies are developed using *Description Logic* that describe network host-based access-control knowledge at the application-layer and at the systems-layer. Ontologies are described for the Netfilter/iptables firewall [2],TCP-Wrapper [5] and an application-level access policy that reflects high-level business goals.

The contribution of this paper is a NAC policy model ontology, over which, intra-policy and inter-policy interoperation can be reasoned about. At the individual policy level, one can reason about conflicts within a policy configuration; for example, firewall consistency checking can be performed over the Netfilter ontology. One can also reason across different policies; by combining the firewall ontology with the application-level risk access ontology, one can reason about service reachability and access permissiveness. The resulting model also demonstrates the practical effectiveness of using Semantic Web techniques in constructing, reasoning about and managing security policies.

The paper is organized as follows. Section 2 introduces Description Logic (*DL*) [6]. Section 3 outlines the policy architecture. Individual ontologies for Netfilter, TCP-Wrapper proxy and an application-level access policy are described in Section 4. Section 5 defines a model that is used to represent configuration of a network of systems, services, proxies, and so forth. This model provides the basis for configuration reasoning, which is considered in Section 6.

2 Description Logic & Ontologies

Description Logic (*DL*) is a family of logic-based formalisms that are well-suited for the representation of and reasoning about terminological (*T-box*) and assertional (*A-box*) knowledge based on the Open World Assumption (OWA) [6, 7]. *DL* represents a decidable portion of first-order logic and forms part of the W3C recommendation for the Semantic Web [4, 8].

DL uses classes (concepts) to represent sets of individuals (instances) and properties (roles) to represent binary relations applied to individuals. For example, the *DL* assertion:

$$Server \sqsubseteq Node \sqcap \exists hasHost.BusinessService \sqcap \exists hasProtection.ProtectionService$$

specifies that a server node (class) hosts business services (class) and has (property) a protection service (class) protecting them. Note that properties are conventionally prefixed by "has"; for instance, $hasHost$, is the property over the individuals of the class $Server$ (domain) that hosts a business service $BusinessService$ (range).

The Semantic Web Rule Language, (*SWRL*), complements *DL* providing the ability to infer additional information in *DL* ontologies, but at the expense of decidability. *SWRL* rules are Horn-clause like rules written in terms of *DL* concepts, properties and individuals. A *SWRL* rule is composed of an antecedent (body) part and a consequent (head) part, both of which consist of positive conjunctions of atoms [9]. For example, the requirement: *servers hosting ssh based business services protected by a firewall require that firewall to open port 22* is expressed in SWRL as:

$$Server(?n) \wedge hasHost(?n, s) \wedge hasPort(?s, ssh) \wedge hasFirewall(?n, ?f) \rightarrow hasPortOpen(?f, ssh)$$

3 Security Policy Model

It is considered best practice to secure business critical applications (regardless of their own ability to secure themselves at a service level) using a layered approach, in particular, by employing low-level infrastructure access control (firewalls, intrusion prevention and so forth). Figure 1 provides an abstract policy architecture whereby the elements represent *DL* classes that are defined in later sections and arcs represent relationships between the classes. The architecture provides for three security policies: Netfilter, TCP-Wrapper and a business-risk policy. The Netfilter and TCP-Wrapper policies are explicitly enforced by their underlying mechanisms. Section 6 considers how the overall configuration is reasoned over to determine whether the business-risk policy is upheld.

A network is a collection of systems Nodes that may host Business Services. A service Risk policy defines the risk of allowing particular clients (Client Range) having access to the service. For example, a particular range of IP addresses may be considered high-risk to a mail server due to a large volume of spam messages, while

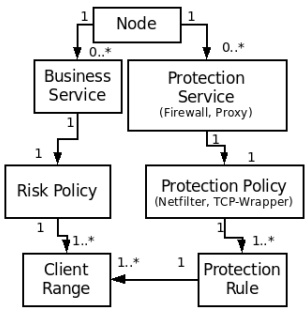

Fig. 1. Ontology Policy Architecture

IP addresses with a service agreement are considered low-risk. Each node is deployed with zero or more Protection Service(s) that provide access control mechanisms such as Firewall(s) and Proxy(s). These protection services are configured in terms of a set of Protection Rule(s) that define its Protection Policy. A protection rule, amongst other components, is defined in terms of a range of IP addresses (Client Range) that may be permitted access.

4 Security Policy Ontology

In this section, we develop *DL* ontologies for the model components described in the previous section. Note that due to space reasons, we provide only fragments of the *DL* model. Complete *DL* definitions, for example; disjoint axioms, data type properties or closure axioms are typically not included. The complete *DL* model has been coded in the W3C Semantic Web languages (*OWL-DL* and *SWRL*) using the Protégé IDE [10].

4.1 System Ontology

Class $Node$ represents the set of system nodes. Nodes host business and protection services and are connected to each other ($hasConnectionTo$) to form a network.

$$Node \sqsubseteq \exists\, hasConnectionTo.Node \sqcap \forall hasProtection.ProtectionService \sqcap \forall hasHost.BusinessService \sqcap \ldots$$

For example, a $Node$ instance $node2$ hosting business service `bs1` that is connected to a gateway node, (`node1`) and is also protected by various access control service mechanisms namely a gateway firewall (`fw1`), a local firewall (`fw2`) and a local proxy (`proxy2`) is illustrated in the following expression. Note, atomic individuals are written in a `typewriter` font, while inferred individuals are given in an *italic* font.

$$Node(node2) \leftarrow hasConnectionTo.(node2, \texttt{node1}) \sqcap hasHost.(node2, \texttt{bs1}) \sqcap hasProtection.(node2, (\texttt{fw1,fw2,proxy2}))$$

The $BusinessService$ class represents the set of business services deployed to nodes within the network.

$$BusinessService \sqsubseteq Service \sqcap \exists hasDeployedTo.Node \sqcap \ldots$$

The class of services, $Service$ is a business service or a protection service.

$$ProtectionService, BusinessService \sqsubseteq Service$$

Access control based services (for example, TCP-Wrapper, Netfilter) are subclasses of $ProtectionService$, where membership of these classes requires protection of at least one node.

$$ProxyService,\ FirewallService \sqsubseteq ProtectionService \sqcap \exists isProtectionOf.Node$$

4.2 Risk Policy Ontology

Security Confidence. Every node has a security confidence; (`high`, `medium` and `low`), that indicates the degree to which the services that it hosts are protected.

$$Node \sqsubseteq \exists_{=1} hasAssurance.SecurityConfidence$$

Thus a $Node$ individual must also have a $hasAssurance$ relationship to an individual of $SecurityConfidence$. This value could be based on an assessment of security of the node itself (for example, Unix or Security Enhanced Linux), the subnet it resides in, and/or the protection services that it hosts. Incorporating the property $hasAssurance.(node2, \texttt{high})$ into the same individual $node2$ described earlier indicates that the `high` security assurance reflects that $node2$ has multiple and combined protection by various access control mechanisms namely a gateway firewall (`fw1`), a local firewall (`fw2`) and a local proxy (`proxy2`).

Each business service has a minimum security confidence requirement for any node on which it is hosted and is defined in the business service definition as:

$$BusinessService \sqsubseteq \exists_{=1} hasAssurReq.SecurityConfidence$$

Risk Metric. Each business service has associated risks ($hasRisk$) of permitting clients (IP address) access to the service. Ideally the sum of the associated risks should not be greater than the maximum acceptatble risk threshold ($hasRiskThreshold$).

$$BusinessService \sqsubseteq \exists hasRisk.RiskPolicy \sqcap \exists_{=1} hasRiskThreshold.Float \sqcap \ldots.$$

While a service will have clients with which it has service agreements (risk value of zero), there can be scenarios where it is expedient to tolerate potential service access

from other client/sources $hasRisk.RiskPolicy$, (notwithstanding the service's internal access controls). For example, it might be considered low to moderate-risk for a mail service to accept packets from addresses that are believed to be spam sources, while accepting packets from addresses that have been blacklisted as botnet sources is considered high risk. The service risk threshold reflects an business/security trade-off decision. Section 6.1 considers how a configuration is tested against this measure.

$$RiskPolicy \sqsubseteq \exists hasRiskIPStartRange.Integer \sqcap \exists hasRiskIPEndRange.Integer \sqcap \exists hasRiskValue.Float \sqcap ...$$

4.3 Netfilter Firewall Ontology

Netfilter is a framework that enables packet filtering, network address translation (NAT) and packet mangling. As a firewall, it is both a stateful and stateless packet filter that is characterised by a sequence of firewall decisions against which all packets traversing the firewall are filtered. Each firewall decision takes the form of a series of conditions representing packet attributes that must be met in order for that decision to be applicable, with a consequent action for the matching packet (accept, drop, log and so forth). An in-depth description for the Linux Netfilter ontology is given in [11].

Netfilter requires the specification of a *table* (`filter`, `NAT` or `mangle`), a *chain*, the accompanying decision *condition* details and an associated *target* outcome. A table is a set of chains and it defines the global context, while chains define the local context within a table. Our research focuses on the firewalling aspects of Netfilter and hence our current model only incorporates the `filter` table attributes. A chain is a set of firewall decisions and those decisions in a chain are applied to the context defined both by the chain itself and the particular table. A Netfilter firewall decision is composed of exactly one chain, one or more condition filters and a single permission target. This is expressed as the *DL* assertion:

$$NamedFirewallPolicy \equiv NetfilterFirewall \sqcap \exists_{=1} hasChain.Chain \sqcap \exists_{\geq 1} hasCondition.Filter \sqcap \exists_{=1} hasTarget.Target$$

The Netfilter/iptables decision that accepts incoming `ssh` requests from a trusted client IP address `4.3.2.1` to a protected server (`1.2.3.4`) is written as:

```
iptables -t filter -A INPUT -s 4.3.2.1 -d 1.2.3.4
                            -p tcp --dport 22 -j ACCEPT
```

This decision is represented in our ontology by an individual id such that,

$$NamedFirewallPolicy(id) \leftarrow hasChain(id, \texttt{inputChain}) \sqcap hasSrcIP(id, \texttt{ip4.3.2.1}) \sqcap hasDstIP(id, \texttt{ip1.2.3.4}) \sqcap hasProtocol(id, \texttt{tcp}) \sqcap hasDstPort(id, \texttt{portSSH}) \sqcap hasTarget(id, \texttt{acceptTarget})$$

Note that the low-level facts of a firewall configuration are presented as individuals rather than classes on the basis that they are atomic and will not be further decomposed. Using instances (rather than subclasses) allows subsequent reasoning of collections of firewall decisions using *SWRL*, as outlined in [11].

4.4 TCP-Wrapper Proxy Ontology

The Linux/Unix-based TCP-Wrapper service is a host-based transport layer proxy that provisions network access control to local daemons spawned by the Internet services daemon (*inetd*). Under typical circumstances Linux environments use a super server (*inetd*) to invoke TCP/IP based network services, for example the `ssh` service. Instead of invoking the `ssh` daemon directly the *inetd* daemon will invoke the TCP-Wrapper daemon. The TCP-Wrapper proxy will permit or deny access to the requested service daemons it protects based on the requesting client (for e.g. an IP address) as ascertained from the *inetd* network connection. If a decision has concluded with a permit action then the TCP-Wrapper proxy shall invoke the appropriate service daemon.

A TCP-Wrapper policy—a set access control rules to protect host-based services—is specified in terms of access control files *hosts.allow* and *hosts.deny*. Recent versions allow the access-controls to be specified in a single file. Like firewall decisions, the ordering of decisions is vital, thus once a decision has been matched no further decisions are processed. A TCP-Wrapper decision at its simplest level can be described as having the following components; one or more service daemons, one or more requesting client and a exactly one permission target and is represented by the following *DL* assertion:

$$NamedTCPDPolicy \equiv TCPWrapperDomain \sqcap \exists_{\geq 1} hasDaemon.DaemonFilter \sqcap \exists_{\geq 1} hasTCPDClient.ClientFilter \sqcap \exists_{=1} hasAction.Action$$

A TCP-Wrapper decision twd, (an individual of $NamedTCPDPolicy$) that states a trusted client IP address is permitted access to the protected `ssh` daemon is represented by the following:

$$NamedTCPDPolicy(twd) \leftarrow hasDaemon.(twd, \texttt{sshD}) \sqcap hasTCPDClient(twd, \texttt{ip4.3.2.1}) \sqcap hasAction(twd, \texttt{allow})$$

The corresponding TCP-Wrapper syntax detailing the access control of the previous knowledge base individual twd is written as:

```
sshd: 4.3.2.1 : ALLOW
```

5 Systems Configuration

This section defines a model that represents network configuration. Intuitively, a configuration is a collection of nodes, services, proxies, etc. and their relationships, and are represented as collections of instances from the ontologies.

A small to medium enterprise network environment typically deploys a gateway firewall as the initial step in provisioning access control and a succession of locally hosted access control mechanisms may follow. Figure 2 illustrates an example of a network access control architecture whereby internal nodes hosting business applications are protected by low-level infrastructure namely a gateway firewall and may also be further protected by a local firewall and/or local proxy deployed on those same nodes. *SWRL* rules and queries can be defined to infer and discover facts about the current systems policy configurations within the network. For example, assuming that nodes $?n$ and business services $?b$ have been assigned security confidence levels (by instantiating the $hasAssurance$ and $hasAssurReq$ properties) we can reason over the knowledge

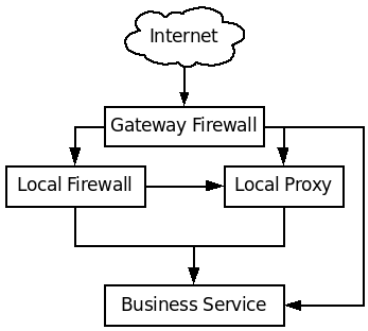

Fig. 2. Example SME Access Control Architecture

base to find suitable nodes on which to host services, that is a hosts confidence is sufficient for the services it may host.

$Node(?n) \wedge BusinessService(?b) \wedge hasAssurance(?n, ?a) \wedge hasAssurReq(?b, ?br) \wedge hasSecureConfidenceValue(?a, ?nrate) \wedge hasSecureConfidenceValue(?br, ?brate) \wedge swrlb : lessThanOrEqual(?brate, ?arate) \rightarrow hasDeployedTo(?b, ?a)$

6 Policy Analysis

6.1 Risk Policy

The ideal security policy configuration is one that is *aligned* with the business service that is, it permits only valid service traffic, and, no more and no less. The following example of a risk-based inter-policy query, tests whether the aggregate risk from clients that are permitted (by the protection services) to reach a service exceeds the risk threshold specified by the service.

$BusinessService(?b) \wedge RiskThreshold(?r) \wedge hasRiskThreshold(?b, ?max) \wedge hasRiskIPStartRange(?r, ?iprs) \wedge hasRiskIPEndRange(?r, ?ipre) \wedge hasRiskValue(?r, ?v) \wedge hasProtectiveProxyDecision(?b, ?t) \wedge hasRisk(?b, ?r) \wedge hasSrcIPStartRange(?t, ?sip) \wedge hasSrcIPEndRange(?t, ?eip) \wedge ipAddress(?sip, ?x) \wedge ipAddress(?eip, ?y) \wedge swrlb : greaterThanOrEqual(?iprs, ?x) \wedge swrlb : lessThanOrEqual(?ipre, ?y) \rightarrow sqwrl : select(?max) \wedge sqwrl : sum(?v)$

This compares the ranges of clients made visible by the proxy decisions ($hasProtectiveProxyDecision(?b, ?t)$) with the associated risks of each client range. Should the sum ($sqwrl : sum$) of individual risks be greater than the maximum acceptable risk threshold then that service is not protected sufficiently by its host TCP-Wrapper proxy policy.

6.2 Policy Conflict Detection

Al-shaer et. al. [1] classify firewall conflicts into four categories: redundancy, shadowing, correlation and generalisation. Due to page constraints, we focus our attention to intra-firewall policy conflicts detected using *SWRL* rules applied across the *DL* knowledge base. Note for reasons of space, we do not consider all four conflict categories in this paper, however they are modeled withinin our ontology. Table 1 provides a fragment of a Linux Netfilter firewall access control configuration.

Table 1. Firewall Decision Policy Example Extract

Decision	Chain	Src IP	Src Port	Dst IP	Dst Port	State	Action	Conflict
1	Forward	*.*.*.*	any	192.168.1.2	80		Drop	
2	Forward	192.168.1.6	any	192.168.1.2	80		Accept	Shadowed by (1)
3	Output	192.168.1.1	any	10.37.2.*	21	Rel	Drop	
4	Output	192.168.1.1	any	10.37.2.*	22	Est	Drop	
5	Output	192.168.1.1	any	10.37.2.3	21,22	Rel,Est	Accept	Shadowed by (3,4)

Shadowing. In general, firewall decisions are activated in sequence starting at decision 1. A shadowed decision is one that is never activated due to previous decisions filtering the same kinds of packets but those decisions having different target actions. Table 1, illustrates that Decision 2 is shadowed by Decision 1. Since Decision 2 is never activated, intended http traffic from a specific host is not permitted.

SWRL rules detect conflicts within a firewall configuration. For example, the following 'partial' *SWRL* rule provides a list of tuples $x \mapsto y$, where x is a decision and y is the decision that shadows x. When executed against the knowledge in Table 1 it returns tuples $2 \mapsto 1$.

$$
\begin{aligned}
&FW1(?fwrule1) \wedge FW1(?fwrule2) \wedge \\
&decisionOrder(?fwrule1, ?order1) \wedge decisionOrder(?fwrule2, ?order2) \wedge \\
&hasSrcIPStartRange(?fwrule1, ?ip1start) \wedge hasSrcIPEndRange(?fwrule1, ?ip1end) \wedge \\
&hasSrcIPStartRange(?fwrule2, ?ip2start) \wedge hasSrcIPEndRange(?fwrule2, ?ip2end) \wedge \\
&ipAddress(?ip1start, ?ip1s) \wedge ipAddress(?ip1end, ?ip1e) \wedge \\
&ipAddress(?ip2start, ?ip2s) \wedge ipAddress(?ip2end, ?ip2e) \wedge \\
&hasDstPortStartRange(?fwrule1, ?dst1ps) \wedge hasDstPortEndRange(?fwrule1, ?dst1pe) \wedge \\
&hasDstPortStartRange(?fwrule2, ?dst2ps) \wedge hasDstPortEndRange(?fwrule2, ?dst2pe) \wedge \\
&portNumber(?dst1ps, ?dst1ns) \wedge portNumber(?dst1pe, ?dst1ne) \wedge \\
&portNumber(?dst2ps, ?dst2ns) \wedge portNumber(?dst2pe, ?dst2ne) \wedge \\
&hasTarget(?fwrule1, ?tar1) \wedge hasTarget(?fwrule2, ?tar2) \wedge \\
&differentFrom(?fwrule1, ?fwrule2) \wedge \\
&swrlb : greaterThanOrEqual(?order2, ?order1) \wedge swrlb : lessThanOrEqual(?ip1s, ?ip2s) \wedge \\
&swrlb : greaterThanOrEqual(?ip1e, ?ip2e) \wedge swrlb : lessThanOrEqual(?dst1ns, ?dst2ns) \wedge \\
&swrlb : greaterThanOrEqual(?dst1ne, ?dst2ne) \wedge \\
&\ldots \rightarrow sqwrl : select(?fwrule1, ?fwrule2)
\end{aligned}
$$

Our DL model not only detects pair-wise conflicts between two firewall decisions (like the approach taken by [1]) but it can readily detect the conjunctions of partial conflicts in a set-wise fashion that may occur across multiple decisions in regard to

a specified decision being analysed. For example, Decisions 3-5 of Table 1 captures the filtering of stateful outward ftp and ssh traffic of the firewall itself towards the 10.37.2.* subnet and 10.37.3.3 node respectively whereby connections are Related or Established. Our model can capture that Decision 5 is 'partially shadowed' by the conjunction of a number of individual proceeding decisions namely 3 & 4 (ports and state).

The same techniques of intra-policy conflict detection are applicable to TCP Wrapper proxy policies. A risk-based inter-policy analysis was discussed in Section 6.1. Inter-policy conflict detection between firewall and proxy configurations to analyze how one may impact on the other can also be reasoned over within the knowledge base by the previous conflict categorisation methods. While individual configurations may be conflict free their interoperation may not be, for example; a netfilter firewall decision may unintentionally 'shadow' an intended TCP-Wrapper decsion.

7 Related Research

An ontology-based model that can be used to (binary) test the safety of an individual firewall policy with respect to a Semantic Web application policy is described in [11]. This paper builds on the results of [11] by considering how interoperation of multiple NAC policies (involving multiple firewalls and proxies) are influenced by more general network service requirements. With this extended model, it is possible to analyze configurations for intra- & inter-policy conflicts, and to also search for suitable configurations based on partial configuration knowledge. Furthermore, the risk-metric provides a quantitative approach to aligning NAC policy to service requirements.

Previous research in relation to applying ontologies to the security domain had primarily focused on security classifications and tended to go no further than providing abstract taxonomy's [12, 13]. However, our research provides, in conjunction with providing an explicit specification and basic *DL* reasoning from a taxonomy perspective, inferences at a lower level of granularity. Thus, low-level facts of a NAC configuration are presented as individuals rather than classes on the basis that they are atomic and will not be further decomposed. Using instances (rather than subclasses) allows subsequent reasoning of collections of NAC configuration policies using *SWRL* to extend the expressive capabilities of *DL*. Policy specific languages such as KAoS utilize the Semantic Web approach [14]. However, these languages tend to have a number of short comings, for example, KAoS's dependence on Deontic Logic unlike *SWRL* [15]. SWRL provides a more generic form of expression with its horn-like rules that can represent varying degrees of policy configuration (enforcement, conflict analysis), business rules, risk metrics and so forth.

Due to the complexity of NAC policy configurations, being able to perform configuration analysis can greatly improve network configuration management issues. For example, a number of approaches have been proposed for the formal analysis of firewalls [16–20]. For example, model-checking techniques [16, 19] are used to test that a configuration of firewalls uphold a global routing policy that restricts certain data to certain sub-nets. In [18] constraint programming is used as an approach to finding suitable firewall rules from higher level policy constraints. The focus in these approaches

is more on analyzing that firewall rules uphold particular 'correctness' properties, or on synthesising firewall rules from specified 'correctness' properties. While this notion of 'correctness' does, in effect, provide semantics for firewall configuration under a limited number of a priori properties, it is not intended to provide a framework for general knowledge representation about firewalls. The *DL & SWRL* approach, while not as expressive as the logics that underlay [16–20], is intended to allow the knowledge base to be extended and managed in general.

The policy conflict detection aspects of this paper focus on the issues of firewall conflicts classified by [1] as an example. Al-Shaer's research provides a filtering decision tree approach to discover and generate an effective and anomaly-free firewall policy decisions [1, 21]. However, we argue that the practicalities of the Semantic Web approach outlined in this paper greatly extends the research of [1] and other such related research. Utilising this approach provisions the expressive semantics to interpret not just firewall configurations but those of NAC's in general in a much more diverse and expressive way. The provision of reasoning, in particular within the context of OWA, provides our model with flexibility and extendability of incorporating new knowledge.

8 Discussion & Conclusion

This paper outlined an approach to using a DL constrained ontology to construct, reason about and manage NAC policy configurations. We demonstrated how NAC policies (for example; Netfilter, TCP-Wrapper and business-level risk policies) could be represented using Semantic Web techniques. The effectiveness of an access control policy may be limited or compromised by poor configuration and management of policy decisions. The paper describes how reasoning facilitated access control (inter-policy & intra-policy) interoperability and measured the level of risk between protection policies and higher-level business goals. Further investigation is required, for example, extending the risk metric model, scalability issues and how this work might be used in practice.

Typical errors in a security policy configuration range from invalid syntax and inappropriate decision ordering to errors resulting from comprehending the configuration given its scale and complexity. This paper illustrated how Semantic Web techniques could be employed to resolve policy conflicts (Section 6.2).

Future research will investigate developing a prototype autonomic architecture that is based on this ontology and reasoning framework. Existing Semantic Web techniques can be used to provide an agent-based approach to deploying and maintaining security policy configurations in a automated/autonomic manner. Semantic NAC agents are responsible for managing the configuration of access controls (firewalls, proxies, IDS's and so forth). These semantic agents negotiate NAC settings that are constrained by the current knowledge base, which is in turn controlled by the NAC agents and other application agents, managing for example, the business rules. The knowledge base is controlled by adding or deleting facts based on new knowledge and inferences by the agents. For example, a business agent informs a firewall agent of a new service offering whereby the firewall agent must reconfigure (new facts) to enable access.

Acknowledgements. This research has been funded by SFI Autonomic Management of Communications Networks and Services PI Cluster Award: 04/IN3/I404C.

References

1. Al-Shaer, E., Hamed, H., Boutaba, R., Hasan, M.: Conflict Classification and Analysis of Distributed Firewall Policies. In IEEE Journal on Selected Areas in Communications, Volume 1-1 (2005)
2. Gheorghe, L.: Designing and Implementing Linux Firewalls with QoS using netfilter, iproute2, NAT and l7-filter. PACKT Publishing (2006)
3. Alesso, H.P., Smith, C.F.: Thinking on the Web: Berners-Lee, Gdel and Turing. Wiley-Interscience (2006)
4. Smith, M.K., Welty, C., McGuinness, D.L.: OWL Web Ontology Language Guide. (W3C Recommendation, Technical Report)
5. Venema, W.: TCP Wrapper: Network monitoring, access control, and booby traps. Third UNIX Security Symposium (Baltimore, September'92) (1992)
6. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
7. Haarslev, V., Mller, R.: Description Logic Systems with Concrete Domains: Applications for the Semantic Web. In: Proceedings of the International Workshop on Knowledge Representation meets Databases, (KRDB), Hamburg, Germany. (2003)
8. Taniar, D., Rahayu, J.W.: Web Semantics Ontology. Idea Publishing (2006)
9. O'Connor, M.J., Knublauch, H., Tu, S.W., Grossof, B., Dean, M., Grosso, W.E., Musen., M.A.: Supporting Rule System Interoperability on the Semantic Web with SWRL. (Fourth International Semantic Web Conference (ISWC2005)
10. Standford: Protege IDE. (http://protege.stanford.edu/)
11. Foley, S.N., Fitzgerald, W.M.: Semantic Web and Firewall Alignment. First International Workshop on Secure Semantic Web (SSW'08), Cancun, Mexico (2008)
12. Anya Kim, J.L., Kang, M.: Security Ontology for Annotating Resources. 4th International Conference on Ontologies, Databases, and Applications of Semantics, (ODBASE), Agia Napa, Cyprus. (2005)
13. Herzog, A., Shahmehri, N., Duma, C.: An Ontology of Information Security. International Journal of Information Security and Privacy (2007)
14. Uszok, A., Bradshaw, J., Jeffers, R., Johnson, M., Tate, A., Dalton, J., Aitken, S.: KAoS Policy Management for Semantic Web Services. In IEEE Intelligent Systems, Vol. 19, No. 4, (2004)
15. Prez, G.M., Clemente, F.J.G., Blaya, J.A.B., Skarmeta, A.F.G.: Representing Security Policies in Web Information Systems. Policy Management for the Web (PM4W) Workshop in the 14th International World Wide Web (WWW) Conference (2005)
16. Guttman, J.D.: Filtering Postures: Local Enforcement for Global Security Policies. IEEE Symposium on Security and Privacy, Oakland (1997)
17. Mayer, A., Wool, A., Zishind, E.: Fang: A Firewall Analysis Engine. 2000 IEEE Symposium on Security and Privacy, p. 0177 (2000)
18. Eronen, P., Zitting, J.: An Expert System for Analyzing Firewall Rules. (In: In Proceedings of the 6th Nordic Workshop on Secure IT Systems (NordSec 2001), pages 100-107)
19. Hazelhurst, S.: A Proposal for Dynamic Access Lists for TCP/IP Packet Filtering. South African Computer Journal, Vol. 33 (2004)
20. Marmorstein, R., Kearns, P.: A Tool for Automated iptables Firewall Analysis. (USENIX Annual Technical Conference, FREENIX Track)
21. Golnabi, K., Min, R., Khan, L., Al-Shaer, E.: Analysis of Firewall Policy Rule Using Data Mining Techniques. In the 10th IEEE/IFIP Network Operations and Management Symposium, (NOMS) (2006)