Towards a Semantic Web Security Infrastructure

Ronald Ashri[1], **Terry Payne**[1], **Darren Marvin**[2], **Mike Surridge**[2], **Steve Taylor**[2]
[1]University of Southampton, Southampton, SO17 1BJ, United Kingdom
{ra,trp}@ecs.soton.ac.uk
[2]IT Innovation, Chilworth Science Park, Southampton, SO16 7NP, United Kingdom
{djm,ms,sjt}@it-innovation.soton.ac.uk

Abstract

The move towards supporting more autonomous systems, where decisions are made without direct user intervention, and more complex operating scenarios, where services from multiple organisations form temporary ties to solve particular problems, creates new security challenges. This paper argues that the answers should combine the use of conventional security solutions, such as cryptographic mechanisms, with the ability to reason about security at the *semantic level*, using appropriate descriptions of security policies and the required tasks. Such reasoning can enable software entities aiming to interact to determine whether their respective security requirements and capabilities will allow them to proceed. Furthermore, it can support the enforcement of security policies based on the context of the interactions. We motivate the need for such reasoning about security through an example and discuss a set of requirements to support the implementation of a specialised security device, termed a *Semantic Firewall*.

Introduction

In recent years there have been significant advances in the development of infrastructure suitable for supporting dynamic interactions between clients and service providers over the Internet. Enabling technological standards, such as Web Services (W3C), and OWL Web Ontology Language (Mcguinness & van Harmelen), combined with emerging technologies, such as OWL-S (Ankolenkar *et al.* 2001) go some way towards providing viable solutions to the problems of communication, discovery, interoperation, and reasoning. In addition, a crucial aspect of the required infrastructure for the effective deployment of *large-scale* and *open* systems is the creation of a *secure* environment. In part, the solution involves the use of "conventional" security technologies such as PKI and X.509 for user authentication, or SSL for secure communication channels. However, as we are moving towards supporting more autonomous systems, where decisions are made without direct user intervention, and more complex operating scenarios, where services from multiple organisations form temporary ties to solve particular problems (Foster & Kesselman 1998), conventional techniques on their own are not sufficient. There is an increasing need to be able to *describe* and *reason* about security requirements at the semantic level, so as to more effectively automate the security response in such situations. By combining conventional security technologies with semantic reasoning methods, we may be able to provide dynamic, adaptive network security.

Towards this end, a central challenge is the ability to deal with the security implications of supporting complex, dynamic relationships between service providers and clients that operate from within different domains, where different security policies may hold and different security capabilities exist. More specifically, we investigate the security implications that arise due to interactions between service providers that are initiated by a client requiring that the service providers cooperate and coordinate so as to achieve its desired goal. Such situations are characteristic of open, service-oriented environments in which clients need to use of a number of different service providers, which must operate in conjunction, in order to achieve their goals.

In order to deal with these issues, we propose the deployment of a security device that makes use of Semantic Web technologies to reason about the security implications of interactions between a *protected* entity and other entities. This device would act as a *Semantic Firewall*, reasoning about whether the interacting entities are able to support the required security policies and whether the interactions that take place are those expected given the aims of the interaction. In order to perform such reasoning the device requires knowledge of what are the security policies of the protected site, (*site policies*) and what are the expected interactions for a give task (*user-defined workflow or process policies*). The purpose of this paper is to present a set of requirements for such a device through an illustrative scenario and an examination of existing work. As such, the paper aims to act as a point of departure for a discussion of how Semantic Web technologies can be used to improve security in service-oriented, open heterogeneous environments and what challenges must be met.

The next section presents the motivating scenario that illustrates the relevant security concerns. Subsequently, we examine how existing work, which employs Semantic Web technologies, can aid in addressing certain aspects of the problem and identify the shortcomings. Based on this, we present a set of requirements for security infrastructure that

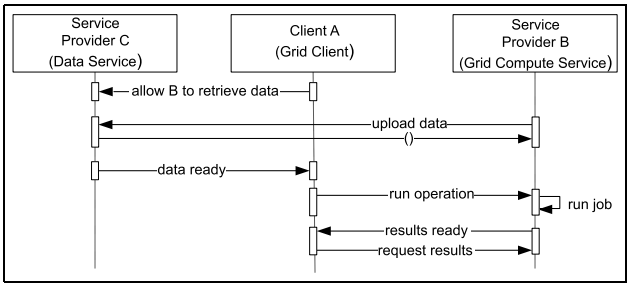

Figure 1: Basic Interaction Scenario

exploits Semantic Web technologies and discuss what challenges such requirements raise. We then discuss the implementations issues for a *Semantic Firewall* device and what technologies can be employed. The paper ends with a brief discussion on other issues that can be considered once a basic Semantic Web security infrastructure is in place.

Motivating Scenario

The example describes a set of basic interactions between two service providers and a client. It is drawn directly from experience with the development of infrastructure and applications for a Grid environment, making use of Web Services technologies (Taylor, Surridge, & Marvin 2004). The scenario aims to highlight the difficulties arising once even just two service providers need to interact to achieve a client's goal and the associated security issues that must be addressed. We illustrate these interactions in Figure 1, through the use of an interaction diagram. In this example, the Grid Client (Client A) needs to make use of a Grid Compute Service (Service Provider B), in order to perform a calculation, but requires that the data for the operation is provided to B by a Data Service (Service Provider C). In practical terms, such a situation could arise when Client A wishes to model the operation of a device (e.g. a heart valve), where Client A provides the model for the device, Service Provider C provides data (e.g. information relating to actual flows of blood through the heart), and Service Provider B provides the computational power and the ability to apply the data to an appropriate model. Each party belongs to a different organisation and as a result may have different security requirements and capabilities. Finally, B and C have no prior knowledge of the existence of each other.

Required Interactions

Our interaction scenario begins from the point where A has identified B and C as the service providers it wishes to use in order to achieve its task. Furthermore, we do not discuss every single step that would be required in a real-world situation but only those core steps that are directly related to the achievement of the task. These are described below.

1. We begin with A requesting from C to allow B to retrieve the relevant data.
2. Subsequently, C notifies A that the data is at B. Alternatively, B could potentially notify A that the data has arrived and that it is ready to perform the calculations.
3. A can now request from B to run the calculations on the data. Alternatively, A could have agreed with B that once the data had been sent from B the operations on the data could take place.
4. A is notified by B that the calculations have finished and that the results are ready.

Discussion

This very simple scenario conflicts with existing conventional network security policies and technologies that system administrators use to protect their systems. In particular a notification step such as the one initiated by B and directed towards the client are only possible if the existing firewall at the client site allows it. This is unlikely to be the case since system administrators enable internally initiated requests for web resources to pass through firewalls they administer but do not allow outside initiated requests against internal user workstations. This type of notification is a prominent requirement for clients invoking long-running computation services on the Grid. In some cases the notification might act more like a call-back implying that some client side handler code is executed. [1]. This suggests that the supporting infrastructure should facilitate such interactions through knowledge of the context of the task.

In addition to such practical concerns, the composition of services at run-time through autonomous or semi-autonomous processes raises another set of higher-level issues that must also be addressed.

- There must be a clear understanding between all parties about who is responsible for *coordinating* the series of actions required. B must request the data from C, after C has accepted to allow such a request, and A must be notified about that before requesting that the operation is performed by B. Knowledge of what is the appropriate *ordering* of actions, or which is the party responsible to define it, is required for the security infrastructure to correctly determine the current context.
- B must accept to store and use data provided by C. An important issue in this respect is determining who will compensate B for the overheads of storing the data and who is responsible for the costs of transferring the data. These decision are complicated by the fact that it is Client A who is initiating these interactions. Although such *payment* issues can be considered beyond the scope of a security device they may have implications on the ordering of actions.
- B must accept that Client A will request that B performs an operation on data that was provided by C and the results should be send to A.
- C must accept to send its data to B and may need guarantees that the data will not be misused or passed on to other parties. For example, although it may accept that A sees the results of the operation it may not want A to see the actual data used.

[1] For a detailed discussion of security concerns in a Grid environment the reader can refer to (Surridge 2002)

- There must be a clear understanding about who is responsible for compensating the various parties for services provided. For example, does C compensate B for the storage of data and then charge A or does A compensate B for the storage as well.

Now, the way these issues are resolved is to a great extent influenced by the particular security policies in practice at each party's domain. For example, the domain within which B operates may *demand* that for all operations on data supplied by parties outside the domain, the provider of the data and the party requesting the operation are the same entity, or that the the requesting party has been delegated authority to request the operation on the data by the provider of the data. The challenge for system developers is to represent such policies for the disparate domains, reason about them and enforce them through appropriate mechanisms. In particular, the different domains will need to dynamically update their policies as more information about the various parties involved in the task is gained. For example, while the basic policy for Service Provider B may be not to accept to perform operations on data from parties that are not owners of the data it may update that policy to represent the situation where the owner has delegated such authority to Client A. Crucially, Client A can only initiate the entire set of operations once it is sure that the various security requirements have been satisfied.

In the next section we discuss some of the current work in relationship to security and Semantic Web technologies.

Related Work

Recent work (Denker *et al.* 2003), has addressed the issue of annotating service descriptions with information relating to their security requirements and capabilities. This information can then be used during the matchmaking process (Paolucci *et al.* 2002), to ensure that clients and service providers meet each others' security requirements, in addition to usual core service requirements. Such a matchmaking capability is a useful means of introducing security considerations and the ability to reason about them at the semantic level. Within the context of the scenario described above, Client A could take advantage of knowledge of the various security requirements in order to establish with which service providers it can cooperate with. For example, if Service Provider B demands that all communication is done using SSL then Client A would have to ensure that it supports that protocol. However, currently the work of Denker et al. focuses on describing conventional security requirements. It does not deal with how more complex information relating to security policies that interacting parties should follow are made known to potential clients, so that they can better guide the discovery process. For example, if Client A was informed that Service Provider B required the authorisation of C from A it may have chosen a different provider that presented a more "relaxed" policy. Such work needs to be taken forward to address not just the description of conventional security requirements but also the description the related security capabilities and requirements in complex scenarios with several interacting parties and possible delegations of security capabilities or rights between services.

Work on policies, based on Semantic Web languages, provides several of the required expressive constructs for defining authorisations and obligations and their delegation (Suri *et al.* 2003). Such work also takes into account some of the issues relating to conflicting policies between different domains, and provides means for resolving them (Uszok *et al.* 2003b). However, although such work takes into account the existence of different policy domains, the resolution of conflicts is centrally managed and relies on basic resolution rules rather than supporting negotiation over how the conflicts can be resolved. In a centrally managed scenario this does not present a problem since the interacting parties do not need to signal their agreement with how the conflicts in policy have been resolved. However, in a scenario where each party belongs to a different organisation any resolution of conflicts should meet the approval of each interacting party. Furthermore, the deployment models suggested for policy enforcement (Suri *et al.* 2003; Dulay *et al.* 2001) may not be suitable for complex, open and dynamic environments where the interaction parties need to reason about and dynamically modify policies.

Kagal et al. (Kagal, Finin, & Joshi 2003), attempt to address some of the shortcomings identified above. They follows a more decentralised and adaptive model. The dynamic modification of policies is supported using speech acts and the suggested deployment models for this work examine different scenarios, such as FIPA-compliant agent platforms [2], web pages and web services. However, they do not take into consideration dynamic adaption of policies within the context of particular interaction scenarios to deal, for example, with notification as discussed in the example.

In conclusion, each of the efforts discussed here makes a significant contribution towards dealing with security issues in distributed, heterogeneous environments by taking advantage of Semantic Web technologies. Nevertheless, there are still several issues that need to be addressed to deal even with a basic interaction scenario as the one described in our motivating scenario. In the next section we set out the outline of the required security infrastructure that combines some of the contributions of the work discussed here and identifies how such work should be extended to better address the issues identified in our motivating scenario.

Semantic Web Security Infrastructure

Our goal is to enhance security in a services-oriented environment by satisfying three overarching *aims*, described below.

Firstly, we wish to ensure that only *appropriate* interactions between parties are allowed to take place. An *appropriate* interaction, in this context, is one that fullfils the following requirements.

- It is *expected* given the agreed upon aims of the interaction between the parties and the interactions that have already taken place.

[2] http://www.fipa.org/

- It satisfies any security requirements associated with that interaction.

Secondly, before entering into a set of interactions for the achievement of a task, the parties under question should be able to verify whether their security requirements and capabilities can be met by the other parties. If conflicts exist, then there should be appropriate mechanisms in place to resolve such conflicts.

Finally, such a supporting infrastructure should address both the needs of users, which typically demand more flexibility in order to execute their applications, as well as the needs of system administrators, which typically prefer more restrictive policies in order to better protect the organisation.

Requirements

In order to satisfy the aims outlined above, we identify the following set of *requirements* that should be met by the security infrastructure. These requirements are divided into *description capabilities* (what we should be able to describe using Semantic Web technologies), *reasoning capabilities* (what type of reasoning we should be able to perform, given those descriptions), and *infrastructure capabilities* (what the infrastructure should be able to do given the descriptions and reasoning over them).[3]

Description Capabilities

Here we discuss some of the basic capabilities that the security infrastructure should have with respect to the kinds of information we need to be able to describe.

Context-dependent security requirements: Interacting parties should be able to describe both context-independent security requirements (e.g. mandating the use of SSL for all communications), as well as context-dependent requirements that arise due to the participation of certain types of parties (e.g. a data provider service acting on behalf of a client) or the execution of certain types of actions (e.g. the transfer and storage of data to use for a calculation). This will allow us to deal with the security implications of interactions between a number of parties that have come into cooperation dynamically at run-time.

"Unconventional" security requirements: Security descriptions should include basic conventional security requirements such as the supported authentication mechanisms. However, they must also describe wider requirements relating, for example, to acceptable delegations of rights or issues relating to payment.

Describing interactions: In order to reason about whether a set of interactions a client wishes to perform are acceptable, both to the other parties and to the system administrator of the domain from where the client operates, we need to be able to describe such interaction scenarios.

The first two requirements refer to the need to describe *site policies*, the regulations that determine what is permitted or not when interacting with a particular site. The third requirement refers to the need to describe *user-defined workflows* or *process policies*. These are the steps that are to be followed by the different interacting parties in order to achieve a task.

Reasoning Capabilities

Given appropriate descriptions, we discuss below how they can be exploited.

Identifying conflicts with site policies: Once a protected service decides to interact with outside parties, the domain within which the service operates should be able to reason about whether the proposed interactions are acceptable. In order to do this it must be able to compare the proposed interactions against the site policies and detect conflicts.

Identifying conflicts with interacting services policies: Provided that the domain of the client accepts the proposed interactions, the next step is to identify whether the interacting services and their respective domains are willing to accept to interact. This will depend both on their own security requirements and capabilities as well as those of the other interacting services.

Resolving conflicts: The resolution of conflicts can also be divided into the resolution of conflicts with system-wide policies and with interacting services policies. The mechanisms for such resolution can vary widely, from the use of pre-established conflict resolution rules to negotiation over the points in conflict. A crucial issue is that any resulting solution must be accepted by all the parties, so resolution must be followed by a re-examination of the policies by each party.

Reasoning over the interaction process: In order to ensure that the *appropriate* interactions are taking place, the security infrastructure should be able to reason about the current context of the process and what are the allowed interactions given that context.

Infrastructure Capabilities

Here we discuss what capabilities the infrastructure should have, both as a result of the ability to perform the reasoning described above, as well as a result of wider requirements relating to implementation and interaction with users.

Decoupling of security from* core *services: The infrastructure should take into account the possibility that not *all* services will be able to individually reason about security requirements, although the domains within which they operate will define relevant security policies. For example, we cannot expect every individual Grid Compute Service to reason about security, its *core* capability is to perform calculations. Such services should be supported by other components able to reason about security. This requires a *decoupling* of the capability to reason about security from the core service provision capability along with mechanisms to enable interactions between the two.

Layered security support: If an individual service defines and is able to reason about its own security requirements, these may still need to be aligned with the security requirements of the organisation within which the service operates. For example, if we consider a university where

[3] We recognise that to a certain extent the description capabilities are driven by the reasoning capabilities, and these are in turn driven by the required infrastructure capabilities. So the division into categories is done solely to serve presentation rather than indicating that they have been conceived in this order.

different departments, and research groups within departments, offer services in a Grid environment it easy to see how each research group, department and the university as a whole should be able to define the appropriate security policies with relation to access to their services from outside the university and within different domains in the university.

Context-dependent adaption: As the interactions between parties evolve the security infrastructure should be able to adapt to the current context in order to allow the necessary messages, such as event notification. Returning to our motivating scenario, the security infrastructure at Client A should allow for the event notification from Service Provider B to A, once it identifies that such an interaction is acceptable given the current context. This would represent an improvement from having to allow such event-notification for the entire interaction sequence, which could expose the domain to the a bigger danger for a malicious attack.

Informing users on reasons for failure: In order for both system administrators and users to accept any steps taken by the security infrastructure, following reasoning, the infrastructure should be able provide some justifications on why an interaction was accepted or rejected.

In order to provide such capabilities relating to security policies, we propose that a service-oriented infrastructure is supported by semantic-level security services working alongside more conventional supporting infrastructure, such as firewalls, reasoning about the security requirements of both individual services as well as the wider security policy requirements of the domain. The required security capabilities require that we are able at run-time to take decisions about the types of services interacting, the current context and the security policies from a number of different domains. Semantic Web technologies are currently the only viable option for enabling us to perform such reasoning, since they allow for a more fine-grained description and reasoning over the relevant issues in a manner that could also address the challenges placed by the open and heterogeneous nature of the service-oriented environment envisioned. In order to provide a more concrete understanding of how such semantic-level security infrastructure could begin to develop, in the next section we investigate the requirements for a *Semantic Firewall*, that acts on behalf of the protected services to ensure the enforcement of security policies and also provide reasoning capabilities about security policies. We draw direct links between the abstract requirements described above and specific Semantic Web technologies to be used, as well as discuss how a Semantic Firewall could be employed in our motivating scenario.

Semantic Firewall

The Semantic Firewall is envisioned as a service operating alongside a traditional firewall, that reasons about the acceptability of incoming and outgoing messages based on the context under which the messages are being sent or received and the security policies in place within the protected domain. This device is under the control of the system administrator, which is responsible for defining the appropriate policies. In this section we discuss the features that should be built into a Semantic Firewall, based on the division into description, reasoning and infrastructure capabilities as discussed above. Throughout the discussion, we will refer to out motivating scenario and how Semantic Firewall devices could be used to resolve some of the problems raised. In Figure 2 we illustrate how the semantic firewalls would operate "behind" a traditional firewall and the message exchange that would take place.

Description Capabilities: A first basic requirement for the Semantic Firewall is a set of OWL ontologies to represent relevant security concepts. These should range from the conventional security concepts relating to different encryption mechanisms, authorisation mechanisms, and so forth, to more abstract concepts such as acceptable delegations of security rights. Exactly which ontologies should be developed, however, depends to a certain extent on the domain under question and the kinds of issues that the Semantic Firewall needs to deal with. A good starting point for guidance on such issues can obtained from the work of Denker et al. (Denker *et al.* 2003) as well as the various ontologies used by the KaoS (Uszok *et al.* 2003a).

The main challenge at this level, however, is describing what are appropriate process definitions or workflows. The Semantic Firewall needs to have access to workflows that describe the associated parties, the expected series of interactions and the temporal, data and causal dependencies between them. Workflows can also be *annotated* with the related security requirements. For example, returning to our motivating example, the required workflow can, roughly, be broken down into the following steps (which are also illustrated in Figure 2).

1. (a) In preparation for retrieving data from the data store service (Service Provider C), the client (Client A) signals the data store service that the compute service (Service Provider B) will require access to some data owned by the client. This step instructs the data store service that it has a role to play in the process and defines what authorisations the data store should open at its site, through the conventional firewall.
   (b) In keeping with the process definition Client A requests execution of some long running service from the Service Provider B. Client A does not block all other execution waiting for completion of the service, but instead B commits to notifying A when execution is complete.
2. Service Provider B obtains the input data from C. It is authorised to do so because C has been informed in step 1a that a request will take place.
3. The job is executed normally by Service Provider B.
4. On job completion the compute service notifies the client. The notification step is included within the process definition and hence the Semantic Firewall instance at the client site can enable authorisation for the notification message sent from the compute service.

The client could then obtain any output from the execution by connecting to the service provider, however, for brevity this is not discussed here. A system administrator will have to enable incoming requests over their existing

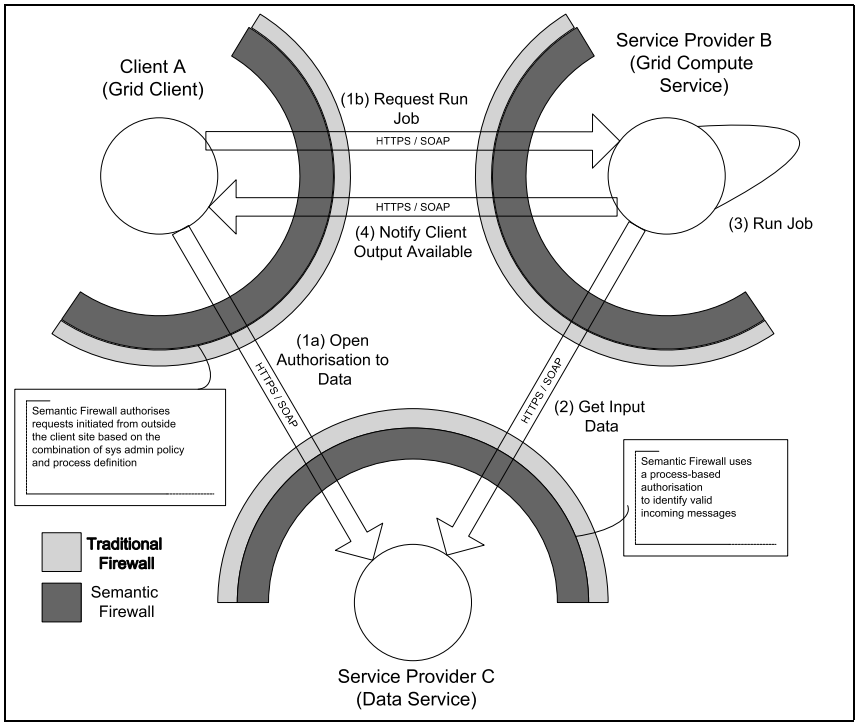

Figure 2: Basic Interaction Scenario

firewall but they do so on the basis that the Semantic Firewall will block requests that are not part of the established Grid workflow. A candidate technology for describing such a workflow could be the Process Model ontology defined by OWL-S (Ankolenkar *et al.* 2001). Despite the fact that the OWL-S Process Model was designed to represent the interactions between components of an individual service as opposed to the interactions between services it is expected that substituting the notion of service with individual components should not pose significant problems. Furthermore, the notions of preconditions and effects, supported by the OWL-S Process Model, could be especially useful in enabling the identification of how a particular interaction may be relevant to a policy at the Semantic Firewall, since they could be used to describe aspects of the environment that are affected as a result of a message exchange. For example, we could state that a precondition of Service Provider B obtaining data form Service Provider C would be to delegate to Service Provider C the appropriate authorisation rights.

Conversational policies represent an alternative way of conceptualising a workflow (and may not necessarily) exclude the use of the OWL-S Process Model. A conversational policy can be seen as both a restriction on the types of messages parties can exchange as well as defining a state transition diagram where the message exchanges indicate changes in state (Smith *et al.* 1998). The benefit of this approach is that it can provide some more flexibility on the kinds of workflows that are eventually enacted. This approach is currently used by the KAoS system (Uszok *et al.* 2003b), although not under the security context which we suggest for the Semantic Firewall.

Having described a workflow, the next issue is how the Semantic Firewall can identify whether the proposed workflow is acceptable or not. A possible avenue is to define the entire set of acceptable workflows that would then be checked against the proposed workflows to be enacted by a service with the protected domain. Such workflows can be annotated with associated security requirements, such as required authorisations, relevant to the specific interactions described within the workflow. The disadvantage of this approach is that such a set is necessarily finite, and there may be situations where a service wishes to enact a workflow where it would still be valid given the implicit security concerns of the site administrator, although there may not be a direct correspondence to the set of acceptable workflows.

Furthermore, such workflow descriptions would have to refer to *types* or *roles* of interacting parties rather than specific instances of interacting parties, since those would only be known once the relevant parties have been identified (in our case by Client A). It is hoped that we can define an essential set of roles that can be adopted by the different parties and relationships that can be created between them, based on the roles, which can then act as the building blocks for describing arbitrarily complex scenarios. Nevertheless, knowledge of the exact interacting parties may have further implications that lead to a rejection of the suggested workflow, even though the individual steps described are acceptable. Therefore, a protected entity could present to the Semantic Firewall an "abstract" workflow to check whether it is acceptable and then a workflow with the precise identities of

the interacting parties.[4]

Reasoning Capabilities: The Semantic Firewall needs to incorporate an appropriate description logics reasoning engine, such as RACER (Haarslev & Möller 2001) or JTP (Fikes, Jenkins, & Frank 2003), along with a specialised module that exploits the underlying reasoning engine to address the particular needs of the Semantic Firewall. The "questions" that need to be answered by the Semantic Firewall are those identified in the previous section, so we do not discuss them again here.

A relevant issue at this level is the frequency with which the Semantic Firewall will need to perform reasoning for each suggested workflow. The problem is that semantic reasoning could become a relatively costly operation, and it should not be performed at each step of the workflow. Ideally, the reasoning process should be performed once before the initiation of the workflow, producing a set of checkpoints that the Semantic Firewall should monitor and a more optimised form of the workflow for following the interactions.

In addition, the reasoning capabilities should provide the possibility for the protected parties to address some of the conflicts identified by the Semantic Firewall. Protected parties can then work towards addressing such requirements before initiating a complex series of interactions involving service providers from different domains. Returning to our scenario, the need to authorise Service Provider C may not become apparent until after some initial communication with the Semantic Firewall of A, which could define an appropriate policy for such situation. Therefore, A should be able to amend the proposed workflow to include appropriate actions that would authorise C.

Infrastructure Capabilities: The introduction of a Semantic Firewall has considerable implications to the more conventional Web Services infrastructure. Below, we discuss some of the implementation issues that need to be considered.

- The Semantic Firewall must be capable of manipulating and processing messages. Those messages are most likely to be SOAP-based as a consequence of using web services but might not be, other message formats could include things like MIME or HTML.
- There are a growing number of security related specifications for SOAP-based web sites, and in particular WS-Security [5] seems to be gaining substantial industry support and is currently under development within OASIS. The interactions between such standards efforts and Semantic Web technologies must be taken into account if the a Semantic Firewall can be deployed widely across organisations.
- Protected parties may not be aware of the existence of the Semantic Firewall, and therefore cannot present appropriately described workflows or deal with the rejection of their workflows. In this case, it would be necessary to establish some pre-agreed workflows offline, that such web services are allowed to enact or participate in.
- The rejection of an incoming or outgoing message by the Semantic Firewall, on the grounds that it violated the acceptability criteria, should be sent to the affected parties, using appropriate notification mechanisms. If these parties are not able to directly reason about rejections from the Semantic Firewall, other appropriate error notification mechanisms should be used.
- The interaction between the Semantic Firewall and enactment engines (e.g. BPEL4WS engines) should be carefully considered. A possible avenue is to develop appropriate extensions to an enactment engine so that it can directly interact with the Semantic Firewall [6]. This could be especially attractive since it would allow the Semantic Firewall to exploit the capabilities of an enactment engine, as a tighter integration between the two would allow for a better control over the interactions. For example, prior to any interactions taking place, intended interactions can be checked with the Semantic Firewall for their validity. However, this may not always be feasible since the enactment engine may not be amenable to changes, or it may not even located within the same domain as that within which the Semantic Firewall operates. As a result, the Semantic Firewall should once more be able to provide appropriate error messages to enactment engines when interactions are rejected.
- Mechanisms should be in place to enable a system administrator to have detailed information about the activities of the Semantic Firewall, so that they can better understand why interactions are accepted or reject. This implies the creation of an appropriate interface for interaction with the users of the Semantic Firewall and the implementation of record-keeping facilities.

Conclusions and Further Discussion

In this paper we described a basic scenario illustrating several of the challenges relating to the establishment of appropriate security infrastructure for an open, dynamic services-oriented environment. Particularly, we discuss the issues raised by having to deal with the interactions between different service providers operating within different domains as a result of a task initiated by a client. Current work on security for the Semantic Web was discussed in relation to this and further issues that must first be addressed have been identified. We propose the development of appropriate security services that can support the definition and reasoning about security policies, which includes the definition of acceptable workflows, annotated with security requirements, from the perspective of the secured domain.

Such security services should act as *Semantic Firewalls*, providing adaptive and dynamic protection to parties within

[4] This implies that the protected entity is aware of the existence of the Semantic Firewall and can communicate with it, which may not always be the case.

[5] www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[6] IT Innovation have been developing an extensible and adaptable enactor engine for several years that already support two process languages and could be adapted to handle BPEL4WS or OWL-S (http://freefluo.sourceforge.net/)

the domain. Such protection should not be based on inflexible rules, but on an understanding of the expected series of interactions between parties within and outside the protected domain. Security support will be crucial for creating the envisaged computing environments where a number of services operating within different domains can be employed in unison to achieve wider goals.

The next steps for the Semantic Firewall will focus on analysing a number of illustrative scenarios within a Grid systems, such as the one described here, so as to inform the development of appropriate models for the required descriptive, reasoning and infrastructural requirements. These models will then guide the implementation of a prototype system that will be tested within the domain of secure access to medical records and services.

Several issues have not been addressed here due to an attempt to contain and focus as much as possible the issues raised. For example, a very relevant issue is the ability for clients and service providers aiming to enter into an interaction to *negotiate* over the required security policies until they arrive to an appropriate set of policies to be enforced for the task at hand. Closely related to this is the notion of establishing *contracts* between interacting parties that clearly define what are the relevant policies which guide their interactions with reference to a particular task. These issues raise a number of difficulties, relating both to the complexity of the resulting interactions and to the development of appropriate protocols for supporting such interactions.

Acknowledgements

The authors would like to acknowledge the valuable contributions of Grit Denker from SRI International, Jeff Bradshaw, Andrzej Uszok, Matthew Johnson, and Gianluca Tonti from IHMC, Claudia Di Napoli from Southampton University, as well as the very useful comments from anonymous reviewers. This research is funded by EPSRC Semantic Firewall project (ref. GR/S45744/01).

References

Ankolenkar, A.; Burstein, M.; Hobbs, J. R.; Lassila, O.; Martin, D. L.; Drew McDermott, S. A. M.; Narayanan, S.; Paolucci, M.; Payne, T. R.; and Sycara, K. 2001. DAML-S: Web Service Description for the Semantic Web. In Cruz, I. F.; Decker, S.; Euzenat, J.; and McGuinness, D. L., eds., *The First Semantic Web Working Symposium*, 411–430. Stanford University, California.

Denker, G.; Kagal, L.; Finin, T.; Paolucci, M.; and Sycara, K. 2003. Security for DAML Services: Annotation and Matchmaking. In Fensel, D.; Sycara, K.; and Mylopoulos, J., eds., *Proceedings of the 2nd International Semantic Web Conference*, volume 2870 of *LNCS*, 335–350. Springer.

Dulay, N.; Lupu, E.; Sloman, M.; and Damianou, N. 2001. A Policy Deployment Model for the Ponder Language. In *IFIP/IEEE International Symposium on Integrated Network Management*.

Fikes, R.; Jenkins, J.; and Frank, G. 2003. JTP: A System Architecture and Component Library for Hybrid Reasoning. In *Proceedings of the Seventh World Multiconference on Systemics, Cybernetics, and Informatics*.

Foster, I., and Kesselman, C., eds. 1998. *The Grid: A Blueprint for a New Computing Infrastructure*. Morgan Kaufmann.

Haarslev, V., and Möller, R. 2001. Description of the RACER System and its Application. In Goble, C. A.; McGuinness, D. L.; Möller, R.; and Patel-Schneider, P. F., eds., *Working Notes of the 2001 International Description Logics Workshop (DL-2001)*, volume 49 of *CEUR Workshop Proceedings*.

Kagal, L.; Finin, T.; and Joshi, A. 2003. A Policy Based Approach to Security for the Semantic Web. In Fensel, D.; Sycara, K.; and Mylopoulos, J., eds., *Proceedings of the 2nd International Semantic Web Conference*, volume 2870 of *LNCS*, 402–418. Springer.

Mcguinness, D. L., and van Harmelen, F. OWL Web Ontology Language: Overview. http://www.w3.org/TR/2003/PR-owl-features-20031215/.

Paolucci, M.; Kawamura, T.; Payne, T. R.; and Sycara, K. P. 2002. Semantic Matchmaking of Web Services Capabilities. In Horrocks, I., and Hendler, J. A., eds., *International Semantic Web Conference*, volume 2342 of *LNCS*, 333–347. Springer.

Smith, I.; Cohen, P.; Bradshaw, J.; Greaves, M.; and Holmack, H. 1998. Designing Conversation Policies using Joint Intention Theory. In *Third Internation Conference on Multi Agent Systems*, 269–276. IEEE Press.

Suri, N.; Carvalho, M.; Bradshaw, J.; Breedy, M. R.; Cowin, T. B.; Groth, P. T.; Saavedra, R.; and Uszok, A. 2003. Enforcement of Communications Policies in Software Agent Systems through Mobile Code. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks*, 247–250. IEEE Computer Society.

Surridge, M. 2002. A rough guide to grid security. Technical Report UKes-2002-05, National e-Science Centre. http://www.nesc.ac.uk/technical_papers/RoughGuidetoGridSecurityV1_1a.pdf.

Taylor, S.; Surridge, M.; and Marvin, D. 2004. Grid resources for industrial applications. In *submitted to WWW2004*.

Uszok, A.; Bradshaw, J.; Hayes, P.; Jeffers, R.; Johnson, M.; Kulkarni, S.; Breedy, M.; Lott, J.; and Bunch, L. 2003a. DAML reality Check: A Case Study of KAoS Domain and Policy Services. In Fensel, D.; Sycara, K.; and Mylopoulos, J., eds., *Proceedings of the 2nd International Semantic Web Conference*, volume 2870 of *LNCS*. Springer.

Uszok, A.; Bradshaw, J.; Jeffers, R.; Suri, N.; Hayes, P. J.; Breedy, M. R.; L. Bunch, M. J.; Kulkarni, S.; and Lott, J. 2003b. KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks*, 93–98. IEEE Computer Society.

W3C. Web services activity. http://www.w3.org/2002/ws/.