Requirements-Driven Mediation for Collaborative Security

Amel Bennaceur, Arosha Bandara, Michael Jackson, Wei Liu, Lionel Montrieux, Thein Than Tun, Yijun Yu, Bashar Nuseibeh

▶ **To cite this version:**

Amel Bennaceur, Arosha Bandara, Michael Jackson, Wei Liu, Lionel Montrieux, et al.. Requirements-Driven Mediation for Collaborative Security. SEAMS 2014 - 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, Jun 2014, Hyderabad, India. hal-00955562

HAL Id: hal-00955562
https://inria.hal.science/hal-00955562

Submitted on 18 Mar 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Requirements-Driven Mediation for Collaborative Security

Amel Bennaceur[1], Arosha Bandara[1], Michael Jackson[1], Wei Liu[2], Lionel Montrieux[1], Thein Than Tun[1], Yijun Yu[1], Bashar Nuseibeh[1,3]
[1] The Open University, Milton Keynes, UK
[2] Wuhan Institute of Technology, Wuhan, China
[3] Lero - The Irish Software Engineering Research Centre, Limerick, Ireland

ABSTRACT

Security is concerned with the protection of assets from intentional harm. Secure systems provide capabilities that enable such protection to satisfy some security requirements. In a world increasingly populated with mobile and ubiquitous computing technology, the scope and boundary of security systems can be uncertain and can change. A single functional component, or even multiple components individually, are often insufficient to satisfy complex security requirements on their own.

Adaptive security aims to enable systems to vary their protection in the face of changes in their operational environment. *Collaborative security*, which we propose in this paper, aims to exploit the selection and deployment of multiple, potentially heterogeneous, software-intensive components to collaborate in order to meet security requirements in the face of changes in the environment, changes in assets under protection and their values, and the discovery of new threats and vulnerabilities.

However, the components that need to collaborate may not have been designed and implemented to interact with one another collaboratively. To address this, we propose a novel framework for collaborative security that combines adaptive security, collaborative adaptation and an explicit representation of the capabilities of the software components that may be needed in order to achieve collaborative security. We elaborate on each of these framework elements, focusing in particular on the challenges and opportunities afforded by (1) the ability to capture, represent, and reason about the capabilities of different software components and their operational context, and (2) the ability of components to be selected and mediated at runtime in order to satisfy the security requirements. We illustrate our vision through a collaborative robotic implementation, and suggest some areas for future work.

Keywords

Security requirements, mediation, collaborative adaptation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SEAMS '14, June 2–3, 2014, Hyderabad, India
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

Categories and Subject Descriptors

D.2 [**Software Engineering**]: Requirements/Specifications; K.6 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security

1. INTRODUCTION

As our reliance on digital, connected devices increases, so does our need for security. Secure systems must provide the necessary capabilities to protect assets from intentional harm. These systems rely on an explicit definition of their security requirements to describe precisely which actions in a system are allowed and which ones are prohibited [21]. Once security requirements are specified, it becomes possible to concentrate on the security controls by which these security requirements can be satisfied.

The high-degree of dynamism and the inherent heterogeneity of mobile and ubiquitous computing environments make security requirements hard to satisfy. First, the components and the assets of the system might be mobile (e.g., smartphones and personal trackers), making the boundary of the system ill defined and uncertain. These components are also often developed independently, and may only be accessed through particular interfaces and specific protocols. For example, the protocol used by a lighting system is likely to be different from that used by a surveillance camera for video recoding. In addition, the physical environment must also be taken into account. For example, a surveillance camera may only be able to capture video when there is enough light. Therefore, the light may need to be turned on when the camera needs to be operated in the dark. It is difficult to design and develop the security controls necessary to protect assets when the operational context is continually changing, and no exact assumptions about the environment can be made.

The complexity of security controls necessary to satisfy security requirements in mobile and ubiquitous computing environments means that they may be unlikely to be implemented by a single component, or even by multiple components in isolation. For example, a smart home may be equipped with a whole range of electronic devices that can help keep the homeowner and their property safe from intruders. If the light is switched on without the door being opened, this can indicate an intrusion and should trigger the alarm. The lighting system, the door lock, and the alarm

each provides a specific *capability*, which put together allows the realisation of the security control necessary to protect the house. In this paper we propose a novel framework for *collaborative security* that leverages the capabilities of multiple components available in the environment in order to deploy the necessary security controls and satisfy security requirements. To realise collaborative security, we must be able to answer several questions:

- How do we capture, represent, and reason about the capabilities of components in ubiquitous computing environments?
- Which components should collaborate to satisfy security requirements?
- How do we make components collaborate?
- Can we ensure that the collaboration satisfies properties such as correctness, safety, and minimality?

In taking initial steps in answering these questions, we build upon *adaptive security* [20] and *collaborative adaptation* [23]. The former focuses on identifying the security controls necessary to keep security requirements satisfied despite changes in the environment. The latter concentrates on the mechanisms necessary to make multiple components collaborate. However, this collaboration is often hampered by differences in the implementations of independently-developed components. To address these differences without changing the components, intermediary software components, called *mediators* [22], systematically compensate for the differences in the implementations of these components.

We propose an initial realisation of collaborative security whereby the security controls are identified using adaptive security techniques and their deployment is performed by first selecting the appropriate components, according to their capabilities, and then synthesising the mediators that enable them to interact successfully. The selection and mediation of multiple components as a means to enact security controls, rather than achieving integration or interoperability only, raises a number of challenges for collaborative security research.

The paper is structured as follows. Section 2 presents our framework for collaborative security that combines adaptive security, collaborative adaptation and an explicit representation of the capabilities of the software components. Section 3 describes the key techniques used in adaptive security to select, analyse, and deploy security controls. Section 4 examines the principal techniques used in collaborative adaptation to select and mediate components in order to satisfy some functional requirements. Section 5 introduces an implementation of collaborative security by combining and unifying techniques from adaptive security and collaborative adaptation. It also illustrates its application through a robotics example. Section 6 discusses the open research questions.

2. A FRAMEWORK FOR COLLABORATIVE SECURITY

Collaborative security aims to exploit the collaboration of multiple, heterogeneous, software-intensive components in order to meet security requirements in the face of changes in the environment, changes in assets under protection and their values, and the discovery of new threats and vulnerabilities. A framework for collaborative security bridges the gap between the security requirements and the components available in the environment. Our framework revolves around three concepts: *security controls*, *capabilities*, and *mediators*.

Security controls specify the mechanisms that need to be deployed in order to enforce security requirements. According to the value of the assets to protect, the potential threats, and knowledge of the system over time—the sort of attacks that have worked, their consequences, and how they were stopped— we must *determine* the appropriate security controls. The deployment of these security controls is achieved through the collaboration of multiple components with different capabilities.

Central in this framework is the notion of *capabilities*. The capability of a component models what a component can do, how it can do it, and under which conditions. In order to enable the analysis and reasoning necessary for meeting security requirements, we must be able to *capture*, *represent*, and *reason about* the capabilities of the components. Capturing capabilities allows us to express what are the components available in the environment. Representing capabilities allows us to describe the features of the components explicitly. Reasoning about capabilities allows us to select the components that need to collaborate in order to realise the appropriate security controls. As the selected components can be developed independently, mediators are used to enable them to interoperate.

Mediators enable heterogeneous components to interoperate by reconciling the differences in their implementations. As the components that need to interact are not known a priori, we must dynamically *synthesise* and *deploy* the appropriate mediators to enable the selected components to collaborate in order to realise the security controls.

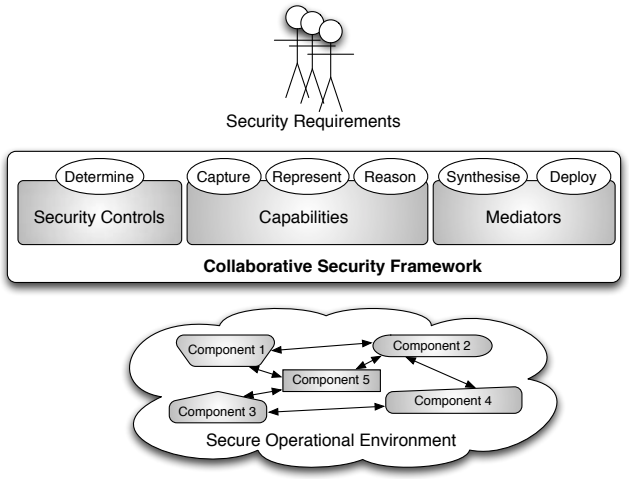

Figure 1: Collaborative Security Framework

Figure 1 depicts the main elements of the collaborative security framework and the functions associated with each of them. In subsequent sections, we elaborate on the opportunities and challenges provided by adaptive security and collaborative adaptation to support each of these elements. Specifically, adaptive security is concerned with the analysis of security requirements and the operational context in

order to determine the security controls that must be deployed to keep a system secure. Collaborative adaptation is concerned with the selection of multiple components, according to their capabilities, and their mediation, in order to realise adaptations that cannot be handled using a single component, or multiple components in isolation.

3. ADAPTIVE SECURITY

Adaptive security aims to enable systems to vary their protection in the face of changes in their operational environment. An adaptive security solution specifies how to: (i) *monitor* the environment and evaluate the properties of the operational context, (ii) *determine* the adequate security controls enabling the satisfaction of security requirements according to the properties of the environment, and (iii) *deploy* these security controls. Figure 2 illustrates the principal elements of an adaptive security solution, which are described in the following.

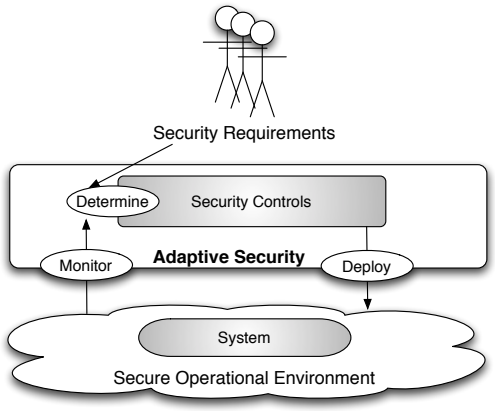

Figure 2: Adaptive Security

Monitoring the environment. The system must be aware of the properties of its operational environment in order to adapt its protection accordingly. Monitoring provides the mechanisms that collect, aggregate, filter and report details collected from the environment [7]. In a security context, the focus of monitoring is on the assets and their values as well as on potential threats and vulnerabilities. Besides techniques that detect violations of the security requirements [15], it is also possible to detect threats by calculating the likelihood of a potential violation of security requirements based on the correlation between past events [17].

Determination of security controls. Deciding what security controls are adequate is difficult because security threats are diverse and often unpredictable, and because the security controls selected often impact usability, performance, and other quality attributes of a system. Hence, the determination of security controls is a multi-objective and cost-sensitive decision-making problem [24].

A requirement-driven approach for adaptive security enables analysing and reasoning about the costs and benefits of the security controls. For example, Salehie *et al.* [20] propose an approach in which a runtime model that combines goals, threats, and assets models is used to evaluate the cost and benefit of applying each security control and choosing the most appropriate one.

Deployment of security controls. As most modern software systems are distributed and increasingly connected, the deployment of a security control may necessitate intervention at different parts of the system. DISCOA [8] defines a model-driven approach to support the deployment of security controls using point cuts over the architectural model of a specific system. Design-time security patterns are increasingly used to provide flexible and effective solutions for incorporating security mechanisms into software systems [9]. However, security controls need not be enacted using a single component but rather through the collaboration of multiple components available in the environment. In the following section, we therefore describe techniques to make components collaborate, albeit to satisfy functional requirements.

4. COLLABORATIVE ADAPTATION

Collaborative adaptation aims to address complex adaptations that cannot be handled by a single component, or by multiple components in isolation. A collaborative adaptation solution specifies how to: (i) *represent* capabilities in order to provide an explicit description of the components and enable *reasoning* about their collaboration, (ii) *capture* the capabilities of the components available in the environment, and (iii) *synthesise* and *deploy* the appropriate mediators that enable the selected components to interoperate even though they have heterogenous implementations. Figure 3 illustrates the principal elements of a collaborative adaptation solution.

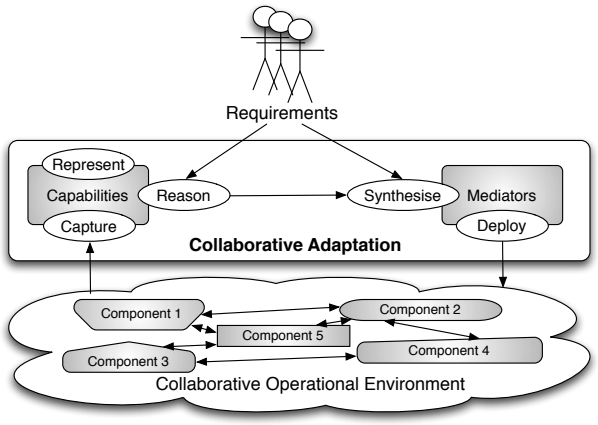

Figure 3: Collaborative Adaptation

Representing and reasoning about capabilities. Since a capability models a component, its representation depends on the kind of analysis and reasoning that need to be performed on the associated component.

Tropos [4] defines a capability as "the ability of an actor of defining, choosing and executing a plan for the fulfilment of a goal, given certain world conditions and in presence of a specific event". In this definition, capabilities are used in the context of an agent-oriented software development methodology rather than in a runtime environment. We regard the capability of a component as independent of the plan it has to execute, and which may change at runtime.

In the Semantic Web Services domain [18], a capability describes what the service does, i.e. the functionality it provides to its clients. It is described using the inputs, outputs, pre-, and post-conditions of the service, all of which are associated with concepts in some domain ontology. In addition, a process model specifies how the service achieves its capability and a service grounding describes the information necessary to invoke the service.

An ontology-based description of services has many advantages: (i) it promotes the semantic matching between clients requests and available services, (ii) it eases the construction of composition of services by making explicit the input, output, pre- and post-conditions of the services as well as their behaviours, and (iii) it facilitates interoperability by formalising both the meaning of the input/output and the behaviour of services.

Nevertheless, solutions based on process algebra and automata have proven more suitable for modelling and analysing the behaviour of components. Hence, they are often used to specify, formally, the behaviour of components (and connectors) in a software architecture.

Capturing capabilities. In ubiquitous computing environments, components often advertise their presence using standard discovery protocols (e.g., UPnP-SSDP, Bonjour, and Jini) [21]. However, these discovery protocols often only provide the syntactic interfaces rather then a rich capability representation. Consequently, learning techniques are often used to infer additional information about the components and complete their capabilities. The additional information can include the semantics of the interface of a component [3], its behaviour [16], or its non-functional properties [10].

Synthesis of mediators. Mediators enable heterogeneous components to interoperate in a non-intrusive way, i.e. without changing the internal implementation of these components. Mediation research has thus far focused primarily on design time activities [13]. There is however a shift towards runtime synthesis of mediators. Furthermore, the complexity of software systems is such that it is difficult to develop 'correct' mediators manually; i.e. mediators that guarantee that the components interact without errors (e.g., deadlocks) and terminate successfully. Inverardi and Tivoli [12] propose an approach to compute a mediator that composes a set of pre-defined patterns in order to guarantee that the interaction of components is deadlock-free. Cavallaro *et al.* [6] combine assembly methods with pairwise mediators to enable the satisfaction of functional requirements. The former consider the structural constraints and specify a coarse-grained composition of components based on the functionality provided or required by each component, while the latter enforce this composition despite the behavioural differences that may exist between each pair of components given the correspondence between the interfaces of the components.

However, in environments where there is little or no knowledge about the components that are going to meet and interact, the complete generation of suitable mediators must happen at runtime while existing approaches assume to be given the correspondence between the interfaces of components or that some mediation patterns are known a priori and composed at runtime. In addition, the synthesised mediators solve the differences between components only at the application-level while in a world increasingly populated with mobile and ubiquitous computing technology, the differences between components span both the application and middleware layers.

Deployment of mediators. As a conceptual paradigm that facilitates the communication and coordination of distributed components despite the differences in hardware and operating systems, middleware has often been used as an enabler for collaborative adaptation. For example, M^2 [23] introduces a message-based collaboration protocol to implement collaborative adaptation and defines an interface to plug legacy components. Nevertheless, when collaboration takes place at runtime, it is necessary to execute mediators that are automatically synthesised for that purpose. Starlink [5] is a runtime framework that executes mediators specified using a domain-specific language that describes the behaviour of this mediator and the data translations that it must perform. Starlink also hides the heterogeneity of middleware protocols by generating, at runtime, the appropriate parsers and composers that translate network messages into and from the actions of the domain-specific language.

However, collaborative adaptation targets integration and interoperability and is agnostic to security concerns. In other words, collaborative adaptation solutions cannot readily be used to satisfy security requirements as they do not explicitly reason about assets and their values as well as potential threats and vulnerabilities of the system.

5. TOWARDS COLLABORATIVE SECURITY

We propose an initial solution for collaborative security, called *collaborative adaptive security*, based on our work on requirement-driven adaptive security, which is supported by the SecuriTAS tool [19], and dynamic synthesis of mediators in ubiquitous environments, which is supported by the MICS tool [1]. Collaborative adaptive security enables the selection and mediation of components to deploy the adequate security controls in order to keep the system secure in a changing operational context. Figure 4 illustrates this solution.

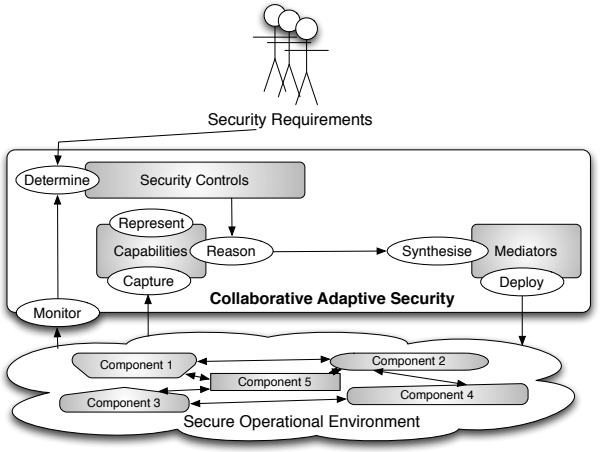

Figure 4: Collaborative Adaptive Security

To determine the adequate security controls, SecuriTAS maintains a runtime model that combines goal, asset, and threat models, and uses it to compute the utility of each security control. The security goals are associated with pre-

defined security controls. Assets are linked to the security goals and associated threats. Once the security controls are identified, we consider the capabilities of the available components and make them collaborate to realise these security controls.

The capabilities of components are represented using a combination of ontologies and transition systems. Ontologies are used to describe the high-level functionality the component requires from or provides to its environment, and to define the semantics of the actions of its interface. Transition systems are used to specify the behaviour of the component formally.

To determine which components need to collaborate to enact the selected security control, we must match the goal model against the high-level functionalities of the available components. For the moment, let us consider that the leaves of the goal model correspond to the high-level functionalities of the available components. However, the collaboration between independently-developed components is often hampered by differences in their interfaces and behaviours. Therefore, mediators are synthesised which systematically compensate for these differences by mapping the interfaces of the components and coordinating their behaviours.

The automated synthesis of mediators, implemented by MICS, is performed in several steps. The first step is interface matching, which identifies the semantic correspondence between the actions required by one component and those provided by the others. We incorporate the use of ontology reasoning within constraint solvers, by defining an encoding of the ontology relations using arithmetic operators supported by widespread solvers, and use it to perform interface matching efficiently. For each identified correspondence, we generate an associated matching process that performs the necessary translations between the actions of the components' interfaces. The second step is the synthesis of correct-by-construction mediators. To do so, we analyse the behaviours of components so as to generate the mediator that combines the matching processes in a way that guarantees that the components progress and reach their final states without errors. The synthesised mediator is the most general component that ensures freedom of both communication mismatches and deadlock in the composition of the components [2]. The last step consists in making the synthesised mediator concrete by incorporating all the details about the interaction of components. To do so, we compute the translation functions necessary to reconcile the differences in the syntax of the input/output data used by the components and coordinate the different interaction patterns that can be used by middleware solutions. Hence, mediation is tackled from the application to the middleware layer in an integrated way. The mediators we synthesise act as: (i) translators by ensuring the meaningful exchange of information between components, (ii) controllers by coordinating the behaviours of the components to ensure the absence of errors in their interaction, and (iii) middleware by enabling the interaction of components across the network so that each component receives the data it expects at the right moment and in the right format.

We have been experimenting with collaborative adaptive security an early prototype demonstrator using two robots: a programmable autonomous vacuum cleaner (iRobot Create[1]) and a humanoid robot (NAO[2]). The two robots need to collaborate in order to secure a particular area in our laboratory. The iRobot Create has a simple capability that consists of executing the moving commands it receives. NAO has the object-recognition capability and can indicate to iRobot Create the area in which it can move. Both iRobot Create and NAO rely on discovery protocols to advertise their presence in the environment, the former uses Bluetooth discovery while the latter uses Bonjour. Nevertheless, putting iRobot Create and NAO in the same environment is not enough to satisfy the security requirements, as they cannot interact with one another spontaneously. What is needed is a mediator that makes them collaborate in order to realise the security requirement. Therefore, a mediator is synthesised which exchanges messages with both robots through their specific interfaces, using the iRobot Create Open Interface to communicate with iRobot Create, and NAOqi to communicate with NAO; and coordinates the behaviours of the components by first receiving the messages from NAO then sending the commands to iRobot Create.

Analysis. To present collaborative security more precisely, we describe it using Jackson and Zave's framework for requirements engineering [14].

Let E denote environment properties,
R_s denote security requirements, and
S denote the specification of a software system that satisfies the security requirement R_s, i.e. $S, E \vdash R_s$.

When the environment properties evolve into E', the specification of the software system S, and the software system itself, may no longer satisfy the security requirement R_s, i.e. $S, E' \nvdash R_s$. As a result, the system must be adapted so as to keep the security requirements satisfied.

To ensure that the security requirement R_s remains satisfied in the environment E', adaptive security transforms the specification of the software system S into a specification S' such that $S', E' \vdash R_s$. S' is obtained from S through the deployment of the adequate security controls. Nevertheless, the specification S' need not be achieved using a single software component; rather, we may take advantage of the capabilities of the different components available in the environment and make them collaborate so as to realise S'.

Let $\mathcal{C} = \{C_1, \ldots, C_n\}$ denote the set of the capabilities of the components available in the environment E'. Note that $\mathcal{C}$ may include the component(s) of S as well. None of the components available in the environment is able to satisfy the security requirement R_s, i.e. $\forall C_i \in \mathcal{C} : C_i, E' \nvdash R_s$. It might be the case that even the conjunction of multiple components cannot satisfy the security requirement R_s, i.e. $\forall C^i \subseteq \mathcal{C} : C^i, E' \nvdash R_s$. Collaborative security seeks a subset of components C^j, together with the specification M of the associated mediator(s) such that $C^j, M, E' \vdash R$.

6. A RESEARCH AGENDA

The increasing ubiquity of connected devices both challenges and supports security. The challenges arise from the frequent and unpredictable changes in the environment, in assets under protection and their values, and the discovery of new threats and vulnerabilities. The support comes from the plethora of devices that can be composed in a multitude

[1]http://www.irobot.com/us/learn/Educators/Create.aspx
[2]http://www.aldebaran-robotics.com/en/

of ways in order to protect valuable assets. Collaborative security aims to address new or changing threats by enabling the runtime deployment of adequate security controls through the collaboration of the components available in the operational environment. In this paper, we suggested that collaborative security may be realised by combining techniques from adaptive security and collaborative adaptation. Nevertheless, many challenges remain.

The choice of security controls need not be specified but may be elicited according to the capabilities available in the environment. We envision inferring, at runtime, new security controls that a developer or a user did not consider at design time. We are also considering the use of security arguments to drive the satisfaction of security requirements [11]. We are thus investigating if argumentation can be used to determine the necessary security controls. Attempting to construct a security satisfaction argument exposes trust assumptions and oversights within the system that can affect security. At runtime, we can evaluate these assumptions to decide if they might be relaxed or invalidated. In addition, as ubiquitous computing and cyber-physical systems can influence their environment, they might change it in order to validate the necessary assumptions that would satisfy the security requirements. In addition, the trustworthiness of the individual components may help determine the components that need to collaborate in order to achieve security.

Finally, while collaboration can help maintain security requirements, how can we prevent it from introducing vulnerabilities? In other words, how do we synthesise mediators that constrain the collaborative behaviour so as to disable anti-goals and prevent attacks from succeeding?

We believe that collaborative security is a fertile research area, with both potential and challenges, and we invite other researchers to collaborate with us in addressing some of these challenges.

7. ACKNOWLEDGMENTS

We acknowledge SFI grant 10/CE/I1855 and ERC Advanced Grant no. 291652 (ASAP).

8. REFERENCES

[1] A. Bennaceur. *Dynamic Synthesis of Mediators in Ubiquitous Environments*. PhD thesis, Université Paris VI, 2013. `http://hal.inria.fr/tel-00849402/en`.

[2] A. Bennaceur, C. Chilton, M. Isberner, , and B. Jonsson. Automated mediator synthesis: Combining behavioural and ontological reasoning. In *Proc. of SEFM*, 2013.

[3] A. Bennaceur, V. Issarny, D. Sykes, F. Howar, M. Isberner, B. Steffen, R. Johansson, and A. Moschitti. Machine learning for emergent middleware. In *Proc. of the Joint workshop on Intel. Methods for Soft. System Eng., JIMSE*, 2012.

[4] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3), 2004.

[5] Y.-D. Bromberg, P. Grace, L. Réveillère, and G. S. Blair. Bridging the interoperability gap: Overcoming combined application and middleware heterogeneity. In *Proc. of Middleware*, 2011.

[6] L. Cavallaro, P. Sawyer, D. Sykes, N. Bencomo, and V. Issarny. Satisfying requirements for pervasive service compositions. In *Proc. of the 7th Workshop on Models@run.time*, 2012.

[7] A. Computing et al. An architectural blueprint for autonomic computing. *IBM White Paper*, 2006.

[8] Ö. E. Demir, P. T. Devanbu, N. Medvidovic, and E. Wohlstadter. Discoa: architectural adaptations for security and qos. *ACM SIGSOFT Software Engineering Notes*, 30(4), 2005.

[9] E. Fernandez-Buglioni. *Security patterns in practice: designing secure architectures using software patterns*. John Wiley & Sons, 2013.

[10] C. Ghezzi, L. S. Pinto, P. Spoletini, and G. Tamburrelli. Managing non-functional uncertainty via model-driven adaptivity. In *Proc. of ICSE*, 2013.

[11] C. B. Haley, R. C. Laney, J. D. Moffett, and B. Nuseibeh. Security requirements engineering: A framework for representation and analysis. *IEEE Trans. Software Eng.*, 34(1), 2008.

[12] P. Inverardi and M. Tivoli. Automatic synthesis of modular connectors via composition of protocol mediation patterns. In *Proc. of ICSE*, 2013.

[13] V. Issarny and A. Bennaceur. Composing distributed systems: Overcoming the interoperability challenge. In *HATS International School on Formal Models for Components and Objects*. Springer Verlag, 2012.

[14] M. Jackson and P. Zave. Deriving specifications from requirements: An example. In *Proc. of ICSE*, 1995.

[15] A. Lazarevic, V. Kumar, and J. Srivastava. Intrusion detection: A survey. In *Managing Cyber Threats*. Springer, 2005.

[16] D. Lorenzoli, L. Mariani, and M. Pezzè. Automatic generation of software behavioral models. In *Proc. of ICSE*, 2008.

[17] D. Lorenzoli and G. Spanoudakis. Detection of security and dependability threats: A belief based reasoning approach. In *Proc. of the 3rd International Conference on Emerging Security Information, Systems and Technologie, SECURWARE*, 2009.

[18] D. Martin, M. Burstein, D. McDermott, S. McIlraith, M. Paolucci, K. Sycara, D. McGuinness, E. Sirin, and N. Srinivasan. Bringing semantics to web services with OWL-S. In *Proc. of WWW*, 2007.

[19] L. Pasquale, C. Menghi, M. Salehie, L. Cavallaro, I. Omoronyia, and B. Nuseibeh. SecuriTAS: a tool for engineering adaptive security. In *Proc. of FSE*, 2012.

[20] M. Salehie, L. Pasquale, I. Omoronyia, R. Ali, and B. Nuseibeh. Requirements-driven adaptive security: Protecting variable assets at runtime. In *Proc. of RE*, 2012.

[21] A. Tanenbaum and M. Van Steen. *Distributed systems: principles and paradigms*. Prentice Hall, 2006.

[22] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 1992.

[23] Z. Yang, Z. Zhou, B. H. C. Cheng, and P. K. McKinley. Enabling collaborative adaptation across legacy components. In *Proc. of the 3rd Workshop on Adaptive and Reflective Middleware, ARM*, 2004.

[24] E. Yuan, N. Esfahani, and S. Malek. A systematic survey of self-protecting software systems. *ACM Trans. on Autonomous and Adaptive Syst.*, to appear.