Model-driven Security: from Theory to Application

Zhendong Ma[1], Christian Wagner[1], Robert Woitsch[2], Florian Skopik[1], and Thomas Bleier[1]

[1] Safety & Security Department
Austrian Institute of Technology
2444 Seibersdorf, Austria

[2]BOC Asset Management GmbH
Bäckerstraße 5, 1110 Vienna, Austria

Abstract: As a specialization of model-driven architecture, model-driven security (MDS) is an approach that uses models to capture and represent a system's architecture and security requirements in software development. Through layers of abstraction, system designers and developers can leverage simple and abstract models to design large and complex systems and generate system artifacts such as security policies or application code from automated model transformations. Regarded as a promising approach to reduce complexity and increase efficiency in the design and development of security-critical software systems, MDS has attracted a great amount of interests in academia and industry. Nevertheless, most existing work concentrates on how to model system and security requirements and how to generate system artifacts. The practicability of MDS has not yet been fully assessed. In a recent pilot project, we have applied MDS to the design and development of Web services for an actual e-Government system in Austria. Our work shows that despite extensive research work, several aspects of MDS need to be adapted and further developed such that one can benefit from such an approach in practice. Our work to address these aspects provides a realistic assessment and valuable insights on the application of MDS to Web services in the real world.

Keywords: Model-driven security, Web service, e-Government, modeling method engineering

I. Introduction

Models are simplified and abstract representation of real-world systems. Models can be defined in mathematical [1], graphical [2], or textual forms. Model-driven development refers to the methodology that extensively uses models in the software development process. The advantage of model-driven development is that larger and complex systems can be built from much smaller and abstract models in a formal manner.

Model-driven Architecture (MDA), proposed by the Object Management Group (OMG), is an approach towards model-driven development [3]. The basic concept of MDA is to separate the specification of a system from the details of the platform that supports it [4]. Model-driven security (MDS) is a specialization of the MDA approach, where security requirements are added to system models and security infrastructures are directly generated from the models [5]. Generally, MDS deals with two basic questions in the system development process: how to model a design along with security policies and how to transform the design and security policies into an actual system [6].

The claimed advantage of MDS has stimulated a lot of research work in the last decade (cf. Section II). However, very few work (e.g., [7]) has been done to apply MDS in realistic scenarios and assess the practicability of such an approach. In a recent pilot project [8], we have applied MDS to the design and development of Web services for the Austrian e-Government system. During the project, we came to realize that although the concept of MDS is quite straight-forward, no existing approaches are readily applicable to solve actual problems in our targeted system. Therefore, we revisited the existing work on MDS and developed new ways that enable us to use MDS for the actual system. During the process, we gained more insights on what are the possible hindrance to the wider adaptation of MDS and how to apply it in practice. This paper describes our model-driven security engineering process and presents our findings during the process.

In our work, we design and develop a new modeling approach that is specific for the secure and efficient development of Web services for the Austrian e-Government system. We introduce a more rigid modeling method engineering process and implement our models on the Open Model Initiative platform [9], an open and community-based modeling platform. As one of the few attempts to apply MDS to an actual system, we provide a realistic assessment and valuable insights on the application of MDS in practice in the context of e-Government Web services.

The remainder of the paper is as follows: Section II gives a structured review of the theory and the historical development of model-driven security. Section III describes our pilot project in which we apply MDS to the secure development of e-Government Web services. We introduce the foundation of our modeling method in Section III-B, followed by model design and implementation in Section III-C and Section III-D, respectively. Section III-E explains in details the important facts concerning the pilot project. We discuss the practicability and open issues of MDS in Section IV. Section V concludes the paper with our agenda for further work.

II. Model-driven Security

In MDA, a system can be modeled on different abstract levels called "viewpoints", which include the computation independent model (CIM), the platform independent model (PIM), and the platform specific model (PSM). A CIM does not have details of a system and is used mainly by domain practitioner who do not necessarily have knowledge about artifacts used to realize the CIM. A PIM is a model that is independent from the platform, hence it can be used for different implementations. A PSM combines the specification in the PIM and the details of the platform. As a result, a system can be specified in models (often represented as text and drawings) that traverse the different levels of abstraction until the models are transformed into software artifacts or a running system. Applying MDA to security engineering, most MDS approaches focus on two aspects: (1) specify system models with security requirements, and (2) automatically generate artifacts of actual systems.

A. Modeling

As an extensive general-purpose modeling language, the Unified Modeling Language (UML) is widely adopted in the area of software development. Thus, early approaches to MDS attempted to extend UML to cover security aspects. UMLSec [10, 11] is a UML extension for secure system development. In UMLSec, security requirements such as confidentiality or secure information flows are added to UML diagrams though extension mechanisms of stereotypes, tagged values, and constraints, collectively defined in a UML profile. SecureUML [6, 12] extends UML for modeling system designs and access control requirements. The main difference is that SecureUML strictly follows the MDA paradigm and emphasizes on modeling method. SecureUML is a security modeling language that has well-defined syntax and semantics. The syntax of SecureUML is divided into an abstract syntax and a concrete syntax. The abstract syntax is defined using Meta-Object Facility (MOF), an OMG standard for formalizing meta models of UML, which includes domain specific knowledge related to security. The concrete syntax is defined using UML profile including stereotypes and tagged values. SecureUML semantics provides a formal basis for model transformation. Semantics in SecureUML are defined by a mapping from models into set-theoretic relational structures with constraints.

B. Generating Artifacts

The main appeal of MDS is that system artifacts can be automatically generated from models. Many approaches have been proposed targeting different hardware and software systems. The authors of [13–16] have developed a framework that applies MDS to Web services. Their framework relies on the existence of a security infrastructure based on the eXtensible Access Control Markup Language (XACML) reference architecture specified in [17]. Aspects related to the system design together with security requirements are modeled and then transformed into XACML (i.e., an XML-based access control policy language). A Policy Enforcement Point (PEP) in the security infrastructure controls the access to a Web service and its resource according to the access policy specified

in the XACML. In [13], Alam *et al.* proposed to use UML user interface models to represent a Web service and its operations and uses the Object Constraint Language (OCL) to specify access policies to these operations. In [14], Breu *et al.* added a workflow model to the framework, which models workflows on a global and local level. Submodels such as document models and role models are employed for describing the data flow and a user's role in a workflow, respectively. In [15], Alam *et al.* extended the framework for trust management between Web services in service-oriented architectures (SOA), in which additional components are added to integrate trust management requirements into system modeling and model transformation. In a similar approach, Nakamura *et al.* [18] showed how to apply MDS to configure security-related artifacts in Web services for commercial products. In their approach, security requirements are added to UML-based application models in the design phase. The models are then transformed into security configuration files targeting the IBM WebSphere Application Server (WAS) for deployment.

Business processes play a key role in SOA for defining how Web services are invoked and how the services interact with each other. Jensen and Feja [19] applied MDS to business processes, in which they extended the ARIS platform [20], a business process management software, by adding security requirements to process models for the generation of security-enhanced business processes for Web services. Souza *et al.* [21] incorporated security requirements into Business Process Modeling Notation (BPMN) models and transformed the models into Web Services Business Process Execution Language (WS-BPEL) files for securing service compositions in SOA. Menzel and Meinel [22] proposed SecureSOA, an extension to [12] that is tailored to model security requirements in SOA system designs.

C. Tool Support

Beside theory and methodology, several MDS tools have been developed in the past. They are either prove-of-concept implementations as a part of the research work, or implementations to apply MDS to address security issues in a specific domain. UMLsec is supported by an open source analysis tool based on XMI output of the diagrams from UML CASE tools [10]. CARiSMA [23] is a re-implementation of the UMLsec tool based on Eclipse Modeling Framework (EMF). SSG [24] is an Eclipse-based development environment to design and generate secure GUI implementations. It allows users to model access control policies proposed in [12]. SECTET [15] includes a proof-of-concept implementation using EMF and OpenArchitectureWare. The Service Security Lab [25] is a MDS-based virtual testing environment for creating and testing Web service-based composed applications. Open Policy Management Framework (OpenPMF) [26] is a commercial software that uses MDS to generate security policies for access control and auditing.

III. Apply MDS to e-Government Systems

Despite extensive research work in the direction of MDS, a very few work has be done to assess the applicability of such an approach to solve real world problems. Our work on the

application of MDS to e-Government systems has the goal to gain more insight into the issue.

A. Background

Web services are loosely coupled, self-contained software modules that can be accessed programmatically using existing Internet technology, found and assembled dynamically to serve a particular function, solve a specific problem, or deliver a particular solution to a customer [27]. Web services use well-defined interfaces and message-based interactions to expose legacy or new business applications and services over the existing network infrastructure, and enable agile, robust and cost-efficient development of information systems. With these advantages, Web services are becoming the primary choice for implementing applications in e-Government systems to support and simplify internal and external governmental processes and provide government services over the Internet.

Due to their distributed nature, Web services as well as e-Government systems must be secured against various cyber attacks. Furthermore, the system must ensure that valuable resources are only accessible to authorized parties. However, due to their scale and complexity, correct and efficient security engineering for e-Government systems is not a trivial task. The situation is even aggravated due to the fact that many service developers might lack special knowledge and experience in designing and developing secure systems.

The Austrian e-Government system is composed of several different technical concepts which complement each other. One important concept in the Austrian e-Government system is the portal group protocol "Portalverbund Protokoll" (PVP) [28]. In general, PVP is a set of protocols and specifications that define a secure, decentralized, and extensible information system based on current Web technologies.

As illustrated in Figure 1, participants in the PVP (e.g., users and service providers) can reside in different organizational domains. A domain may for instance be an internal network of a public or private organization. Different domains are loosely interconnected by HTTP or SOAP messages exchanged over existing network technologies. The PVP in fact facilitates a federated identity management system through the use of security gateways called "user portal" and "application portal". Such a federated identity management system requires that the participating organizations are mutually able to authenticate and authorize their users. This fact eventually requires the mutual establishment of trust between the participating domains, which is achieved by joining a multilateral contractual agreement. A portal operator is responsible for user rights administration and enforcement of access control. For example, if an organization in Domain A needs to access an external service in Domain B, it signs an agreement with the service provider in Domain B. Access control mechanisms are provided by the "application portal" of the service provider in Domain B. In turn, the application portal delegates the access control to the user portal of the organization in Domain A. Consequently, the organization can grant local users access rights for a service in another domain. A user with the access right needs only to authenticate to the user portal for accessing the external service. The advantage of such a federated identity and trust management is that it eliminates the need for a single point of administration to ensure the scalability of the e-Government system, simplifies and expedites service application and consumption, and enables single sign-on (SSO).
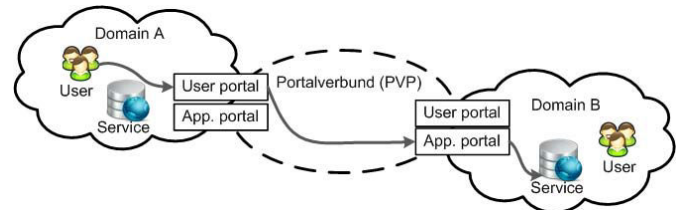
Figure. 1: Architecture of portal group protocol

Also specified in PVP is a standard data structure for message exchanges, which includes information of the requesting organization, the user whose access right is granted by the organization, and a PVP role. This set of information is called a PVP token and is sent together with a certain service request.

It should be noticed that Figure 1 only provides a basic view. The actual system is much more flexible and variant to satisfy requirements of different stakeholders. For example, multiple services might be orchestrated to provide a new service, or an internal service might be deployed to interact with a user and other internal/external services.

Since security is one of the most important issues for e-Government systems, PVP recommends the use of HTTP together with Transport Layer Security (TLS) or Secure Sockets layer (SSL) for the confidentiality, authentication, and integrity of message exchange between two Web services. Furthermore, the contractual agreements make sure that trust relationship is established before any service can be linked to the system. This means, instead of building security from scratch as what is often assumed in related work (cf. II), the focus of security engineering is on how to locally manage access control and how to configure security-related parameters of Web services on a heterogeneous hardware/software platform. Therefore, existing work on MDS are not readily applicable to the PVP system in Austria.

B. Modeling Method

Model-driven security is a combination of modeling and security engineering. Existing work on MDS mainly focuses on security engineering. Since modeling is an equally important but often overlooked part of MDS, we take a more rigid approach to modeling that takes existing work on modeling methods into consideration.

In general, models are a reflection of reality and describe relevant parts of the so-called "system under observation". Most commonly used models in information science are diagrammatic or graphical models, in which semantic primitives in form of graphical objects are used to create human readable but (semi-)formalized models.

One of the challenges in MDS is to bridge the gap of human software engineers that are possibly inexperienced in secure Web service implementation on the one side and a formal correct extract of the model for computer interpretation on the other side. This is called "semantic loss" that appears when models are transformed between different levels of for-

malizations. In order to manage semantic losses, each model needs therefore a built-in semantic that is usually expressed in specification documents. The description and implementation of this semantic, as well as the level of freedom for the use are the success criteria if a model is applicable and useful. In order to ensure that each created model is independent from the modeler, it is necessary to introduce a specification of the model.

The concept of *meta models* deals with specifications of models and are implemented as modeling languages. Traditionally, the Meta-Object Facility (MOF) layers [29, 30] are used to describe IT-oriented diagrammatic modeling languages that define available semantic primitives. Each modeling language covers (a) syntax, (b) semantic and (c) notation [31]. In addition, each modeling language requires a modeling procedure, i.e., a sequence of phases that are followed in order to create a model. Modeling language and modeling procedure are called "modeling technique". A modeling method consists of modeling techniques and additional functionality that is provided either for the modeling language or for the modeling procedure.

To apply such a modeling method to MDS, our main task is to design and develop appropriate meta models according to the modeling method engineering, and provide developers with support on the secure and efficient development of e-Government services through user friendly modeling tools and partial automation of software artifacts.

C. Meta Model Design

The first task in our modeling procedure is to construct a meta model, which contains information, constraints, and semantics that are relevant for describing the Austrian e-Government system. Based on the analysis of the system and interviews with the developers, we identify five most important parts for the development of Web services in the Austrian e-Government environment. Therefore, our core meta model consists of six submodels:

- a *Business Process Model* which allows the modeling of interactions between a user and a Web service, as well as the workflows of orchestrated Web services either inside or outside the PVP system;

- a *Web Service Model* for modeling Web service endpoints and operations;

- a *Security Model* that allows to define typical security requirements for e-Government services such as confidentiality and integrity;

- an *Access Control Model* for controlling the access to a certain Web service operation;

- a *Work Profile Model* for assisting the specification of access controls;

- and an eGovernment Model that defines the required parameters to connect to the PVP system, for example, a PVP token.

Figure 2 shows the architecture of our meta model, represented as a UML class diagram. In the *Business Process*

Model, we choose to use the notations defined in the WS-BPEL specification [32] because it is the *de facto* standard for composing business processes with Web services. The *Web Service Model* has a *Webservice* class that is represented by an address of a Web service endpoint and is composed of one to many *Operation* objects that include input and output parameters. An *Operation* is called by an *Invoke* activity. The *Security Model* consists of a *Security Requirements* class which allows the specification of security requirements such as confidentiality, integrity, and non-repudiation that impose on certain Web service operations. The *Access Control Model* consists of classes for modeling Role-based Access Control (RBAC). In the RBAC model, a *User* is a natural person and may have one to many roles in the system. A *Role* represents a job function of a group of users in the system. A *Permission* controls the access to a resource, for example, a certain Web service operation. The *eGovernment Model* has a *PvpToken* class, which specifies necessary information a Web service needed in its requests to communicate with other services in the PVP system. Inspired by the work on scenario-driven role engineering processes in [33], we define a *Work Profile Model*. A *Work Profile* of a certain role in the *Access Control Model* consists of all associated *Tasks* a role may perform. A *Task* in our case is represented as a WS-BPEL business process. In this way, a *Permission* can also be implicitly defined via a *Role*'s work profile.

D. Model Implementation

A modeling platform provides the necessary tool support for modeling method engineering that deals with specification, implementation and deployment of modeling methods. In our project, we choose the Open Model Initiative [9] because it applies the open source concept in the domain of modeling method engineering and hence supports all three phases with (a) foundational material, (b) tools and platforms and (c) enables communities for each of the modeling methods. Following the vision "models for everyone", this initiative has the aim to support any modeling method engineer in accomplishing the specification, implementation and deployment of modeling methods [34]. Because of its uniqueness for creating new modeling methods, it has been used in our MDS approach for Web services in the e-Government system.

The Open Model Initiative platform offers an Administration Toolkit for user and meta model administration and a Modeling Toolkit for developing concrete instance models of the defined meta models. A meta model is created by defining its classes and relation classes, each with certain attributes. A class represents an object that can be drawn into a canvas of the Modeling Toolkit. The graphical representation of such an object is specified in the GraphRep attribute. Semantic of the models is achieved by defining relation classes which control the objects' connectivities.

E. Use Case Study

As a proof-of-concept implementation, we apply the MDS approach to the development of a secure Web service as a part of a disaster management system.

The use case is described as follows: a medium-sized industrial city in Austria is situated next to the river Danube. Due to its location, it is subject to flooding as a result of days
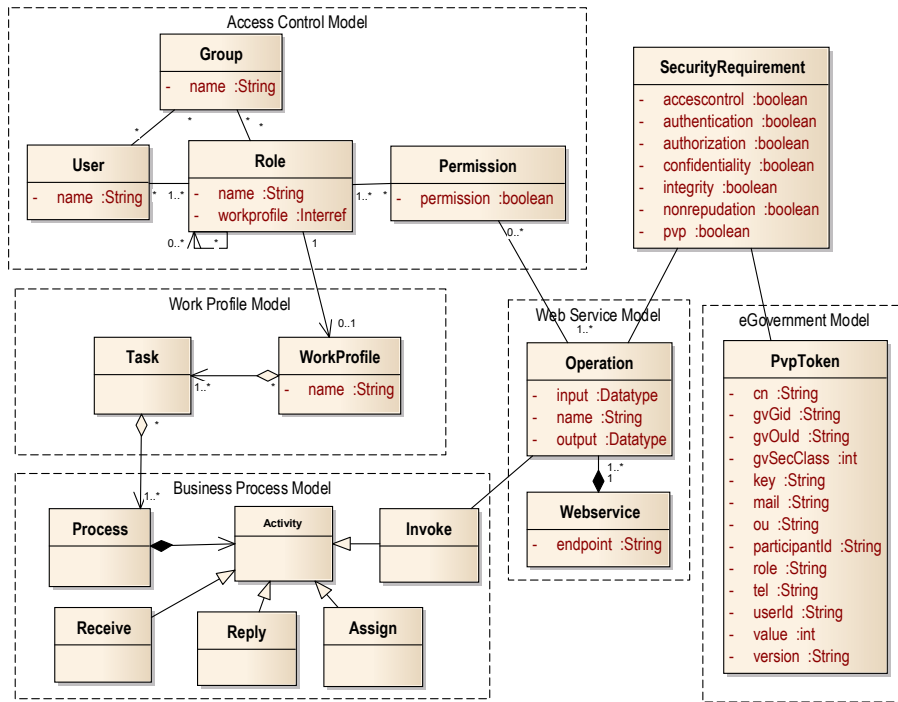
Figure. 2: Meta model class diagram

of heavy rain fall in Danube and its tributaries. A flooding in the area would have catastrophic consequences and affect chemical factories and critical infrastructures such as power and water supply, and gas distribution systems, as well as civil facilities like the city hospital. The Web service will provide crucial information services to the emergency relief unit, by aggregating information from different sources in the e-Government system, including resident registration information and cartographic material, and presenting them to the leader and the other members of the squad to make correct and prompt decisions and plans.

An overview of the use case development process is illustrated in Figure 3. The meta model mentioned in Section III-C is implemented using the Administration tool provided by the Open Model Initiative platform. In this stage, decisions regarding the graphical representation have to be made. In the next step, a concrete instance model is created that contains important information regarding the endpoints of the orchestrated Web services. The instance model is then made available to the software artifact generator as an XML file. The generator generates the Web service that will aggregate data from various sources to be used by the emergency relief unit.

Some of the data are provided by Web services situated within the e-Government PVP system. Therefore, the service must meet a set of security requirements that are imposed on the services connected to the PVP system. Furthermore, the service must utilize a fine-grained access control mechanism, allowing members of the relief team access only to information that is required to fulfill the task. The focus of the use case is on modeling of security features of the Web service and the automatic transformation of such a model to Web service artifacts.

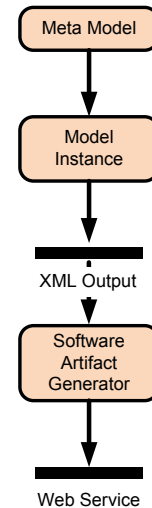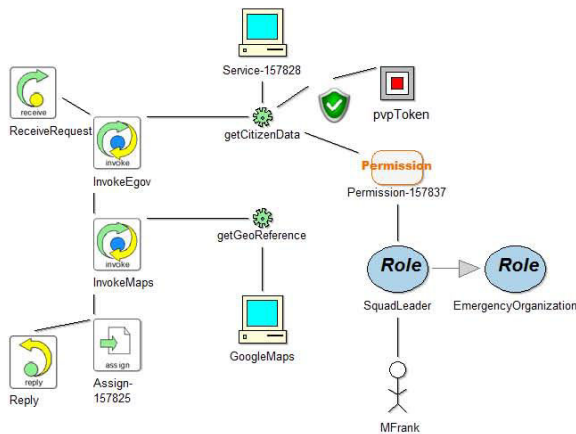As an example, Figure 4 shows a screenshot of the model implementation on the Open Model Initiative plat-

Figure. 3: Use case development overview

form. The model is an instantiation of the meta model defined in Section III-C. The screenshot shows two operations `getCitizenData` and `getGeoReference`. The operation `getCitizenData` calls a "resident registration service" in the PVP system, and the operation `getGeoReference` retrieves the corresponding map data from the openly available GoogleMaps service. The business process defined in our model orchestrates the operations to fetch resident registration data and map data, aggregates the data and presents the data in a Web browser. On the other side of the figure, we specify a RBAC model by mapping each user to the roles and defining the permissions associated with the roles. Notice that the lock symbol in the figure represents the rest of the security requirements (i.e., security requirement except access control) on

`getCitizenData`. The communications among the Web services (e.g., `eGov Service`) within the PVP system require PVP tokens. In `pvpToken`, we can specify all parameters related to a valid PVP token.

Figure. 4: Screenshot of model implementation on Open Model Initiative platform

In the next step the created model on the Open Model Initiative platform is exported in a machine-readable XML format. Listing 1 shows an example snippet of such an XML file. As shown, lots of information in this file is only relevant for the graphical representation of the models in the Open Model Initiative platform. Therefore, in the next step the exported XML file is parsed for relevant information which is then handed over to the software artifact generator.

```
1  ...
2  <INSTANCE id="obj.69611" class="User" name="User-69611">
3  <ATTRIBUTE name="Position" type="STRING">NODE x:2cm y:3.5cm
4       w:1.44cm h:3.86cm index:1</ATTRIBUTE>
5  <ATTRIBUTE name="External_tool_coupling" type="STRING">
6  </ATTRIBUTE>
7  <ATTRIBUTE name="fontcolor" type="EXPRESSION">EXPR expr:
8       (set(targets, ctobjs("Is_inside")),
9  cond
10   (tokcnt(targets,"_")=0,"black",
11    (cond
12     (tokcnt(targets,"_")=1,aval (VAL ctobjs("Is_inside"),
13      "Fontcolor"),
14    (
15    set(tk,token(targets,0,"_")),
16               set(t,VAL (copy(aval(VAL tk, "Position"),
17               search(aval(VAL tk, "Position"),"index",0)
18               +6,-1))),
19    fortok(x,targets,"_",
20       (
21       set(c,VAL (copy(aval(VAL x, "Position"),
22       search(aval(VAL x, "Position"),"index",0)+6,-1))),
23       cond(c&gt;=t,(set(res,x),set(t,c)),"")
24       )),
25               aval (VAL res, "Fontcolor")
26    ))
27    )
28   )) val:"black"</ATTRIBUTE>
29  <ATTRIBUTE name="ID" type="LONGSTRING"></ATTRIBUTE>
30  <ATTRIBUTE name="GUID"
31  type="STRING">A54CE472-7B6C-46F1-914B-2C01F6CE5B7B
32  </ATTRIBUTE>
33  </INSTANCE>
34  <INSTANCE id="obj.69614" class="User" name="User-69614">
35  <ATTRIBUTE
36  name="Position" type="STRING">NODE x:2cm y:11.5cm w:1.44cm
37  h:3.86cm
38  index:2</ATTRIBUTE>
39  <ATTRIBUTE name="External_tool_coupling" type="STRING">
40  </ATTRIBUTE>
41  ...
```

Listing 1: Snapshot of an exported XML file.

The generator performs the actual transformation from the model to source code, which is either a JAX-WS or a Axis2 Web service running on an Apache Tomcat server. We test the generated e-Government Web service against the format requirements of the SOAP header structure defined by the PVP protocol. We also test the exchanged PVP token for validity. The tests are conducted by soapUI [35] test cases. The test set provides the developers of e-Government applications with a useful tool for verifying and even certifying their products against Austrian e-Government system specifications. As a result, the generated Web service is fully compliant with current Austrian e-Government requirements.

IV. Lessons Learned

In theory, model-driven security can achieve more secure and efficient design and development of IT applications as well as systems. Nevertheless, despite its much proclaimed advantages, MDS has not been widely adopted in practice. Our work to apply MDS to Web services for an actual e-Government system has led us to the following observations, which provide assessments in realistic settings and possible directions for further developments.

First, as a supporting method for security engineering, MDS can be applied in certain circumstances, for example, for Web service-based e-Government systems, where many aspects of the system are relatively standardized. However, there is no one-size-fits-all MDS solution. MDS needs to be adapted on a system-by-system basis. In this respect, one needs to balance the cost-benefit factor, i.e., the effort on the adaptation of the MDS and the benefit in terms of security level and development time saved. In this sense, the development of system-independent meta models can reduce the effort on adaptations for new systems.

Second, since in most cases MDS is targeted at developers, flexible and user-friendly modeling tool support is a crucial factor for the wide adoption of MDS in practice. Given that current tool support of MDS is very limited, a holistic and community-based effort is necessary to push forward the adoption of MDS.

Third, modeling is only a half of the whole process of MDS. The correct and automatic transformation of models to artifacts for target systems is equally important. Again, existing work on transformation is limited, in which transformation libraries can not be re-used across different approaches. Thus, a holistic and community-based approach to interpret and transform the models will benefit the users of MDS and reduce the time on repeated work.

V. Conclusion and Future Work

Model-driven security (MDS) applies model-driven architecture to security engineering, in which models are used to simplify system design (through abstraction) and generate artifacts (through model transformation). It is regarded as a promising approach to reduce complexity and increase efficiency in the design and development of security-critical software systems. In order to have a realistic assessment of such an approach, we have applied MDS to the design and development of Web services in the Austrian e-Government system. Despite the straight-forward process described in

most of the literatures (i.e., specify system model and security policy followed by automatic generation of the actual system), our work in practice has revealed that existing work on MDS is not readily applicable to the specific system in our case. As a result, we designed and developed new models and strengthened the MDS approach with a rigid modeling method engineering process. Based on the outcome from the modeling process, we generated artifacts that can be used to construct actual Web services for the Austrian e-Government system.

Our work in this paper focuses on tailoring MDS to the secure and efficient development of Web services instead of developing theories that abstract away from the nuts and bolts of actual systems. Our findings during the process led us to the observation that several gaps between the theory and practice (i.e., very system-dependent, lack of tool support and transformation libraries) need to be addressed before MDS can be really applied in the real world. Closing these gaps are on the agenda of our future work.

Furthermore, most existing work focuses only on software design and development. In our future work, we will also investigate the possibility to leverage on modeling methods and the advances in MDS to simplify the process and increase the efficiency of integrating security activities into the whole development life cycle.

Acknowledgments

References

[1] L. Morogan, "A model of synaptic formation between neurons in a network," *IJCISIM*, vol. 1, pp. 80–87, 2009.

[2] J. Wakatsuki, T. Saito, A. Abrishamian, M. Sakamoto, and T. Uozumi, "Construction of graphical models for temporary application of facial packs," *IJCISIM*, vol. 1, pp. 399–406, 2009.

[3] M. Clavel, V. Silva, C. Braga, and M. Egea, *Model Driven Architecture: Foundations and Applications*, I. Schieferdecker and A. Hartman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, Jun. 2008.

[4] Object Management Group (OMG), "MDA guide version 1.0.1," http://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf, June 2003.

[5] D. Basin, M. Clavel, and M. Egea, "A decade of model-driven security," in *ACM SACMAT*, 2011, pp. 1–10.

[6] D. Basin, J. Doser, and T. Lodderstedt, "Model driven security for process-oriented systems," in *ACM SACMAT*, 2003, pp. 100–109.

[7] M. Clavel, V. Silva, C. Braga, and M. Egea, "Model-driven security in practice: An industrial experience," in *ECMDA-FA*, 2008, pp. 326–337.

[8] Z. Ma, C. Wagner, and T. Bleier, "Model-driven security for web services in e-government system: Ideal and real," in *NWeSP*, 2011, pp. 221 –226.

[9] "Open Model Initiative," http://www.openmodels.at.

[10] J. Jürjens, "UMLsec: Extending UML for Secure Systems Development," in *UML*, 2002, pp. 412–425.

[11] J. Juerjens, *Secure Systems Development with UML*. SpringerVerlag, 2003.

[12] D. Basin, J. Doser, and T. Lodderstedt, "Model driven security: From UML models to access control infrastructures," *ACM Trans. Softw. Eng. Methodol.*, vol. 15, pp. 39–91, January 2006.

[13] M. Alam, R. Breu, and M. Breu, "Model driven security for web services (mds4ws)," in *INMIC*, dec. 2004, pp. 498–505.

[14] R. Breu, M. Hafner, and B. Weber, "Modeling and realizing security-critical inter-organizational workflows," in *IASSE*, 2004, pp. 139–144.

[15] M. Alam, R. Breu, and M. Hafner, "Model-driven security engineering for trust management in SECTET," *JSW*, vol. 2, no. 1, pp. 47–59, 2007.

[16] M. Hafner and R. Breu, *Security engineering for service-oriented architectures*. Springer, 2009.

[17] OASIS, "eXtensible Access Control Markup Language (XACML) v2.0," 2005.

[18] Y. Nakamura, M. Tatsubori, T. Imamura, and K. Ono, "Model-driven security based on a web services security architecture," in *SCC*, 2005, pp. 7–15.

[19] M. Jensen and S. Feja, "A security modeling approach for web-service-based business processes," in *ECBS 2009*, 2009, pp. 340–347.

[20] Software AG, "ARIS Platform," http://www.softwareag.com/corporate/products/aris_platform/default.asp.

[21] A. R. Souza, B. L. Silva, F. A. Lins *et al.*, "Incorporating security requirements into service composition: From modelling to execution," in *ICSOC*, 2009, pp. 373–388.

[22] M. Menzel and C. Meinel, "Securesoa modelling security requirements for service-oriented architectures," in *SCC*, 2010, pp. 146 –153.

[23] "CARiSMA," http://vm4a003.itmc.tu-dortmund.de/carisma.

[24] M. A. G. de Dios, C. Dania, M. Schläpfer, D. A. Basin, M. Clavel, and M. Egea, "SSG: a model-based development environment for smart, security-aware GUIs," in *ICSE*, 2010, pp. 311–312.

[25] M. Menzel, R. Warschofsky, I. Thomas, C. Willems, and C. Meinel, "The service security lab: A model-driven platform to compose and explore service security in the cloud," in *SERVICES-1*, 2010, pp. 115 –122.

[26] "OpenPMF," `http://www.objectsecurity.com/en-products-openpmf.html`.

[27] M. P. Papazoglou, *Web Services: Principles and Technology*. Pearson, Prentice Hall, 2008.

[28] R. Hörbe and M. Werzowa, "Portalverbund whitepaper," `http://reference.e-government.gv.at`.

[29] Object Management Group (OMG) , "MOF core specification," `http://www.omg.org/spec/MOF/2.0/`, 2006.

[30] H. Kühn, "Methodenintegration im business engineering," Ph.D. dissertation, University of Vienna, 2004.

[31] D. Karagiannis and H. Kühn, "Metamodelling platforms," in *EC-WEB*, 2002, pp. 182–.

[32] OASIS, "Web Services Business Process Execution Language Version 2.0," 2007.

[33] M. Strembeck, "Scenario-driven role engineering," *Security & Privacy*, vol. 8, no. 1, pp. 28–35, 2010.

[34] D. Karagiannis, W. Grossmann, and P. Höfferer, "Open model initiative - a feasibility study," `http://cms.dke.univie.ac.at/uploads/media/Open_Models_Feasibility_Study_SEPT_2008.pdf`, September 2008.

[35] "soapUI," `http://www.soapui.org/`.

Author Biographies

Zhendong Ma works as a research scientist at Austrian Institute of Technology focusing on security of communication and information systems. He holds a master degree in Telecommunications from Technical University of Denmark and a PhD degree in Computer Science from Ulm University, Germany. His current research interests include privacy enhanced technologies, model-driven security, SOA and Cloud security, critical infrastructure protection, and national cyber defense.

Christian Wagner graduated in computer science from University of Applied Sciences Technikum Wien in 2008 and holds a degree in business administration from the business college BHAK in Wiener Neustadt. From August 2005 to May 2009 he developed speech recognition systems for medical systems at Philips Speech Recognition Systems. He joined AIT in 2009 as research engineer in the area of secure software architectures.

Robert Woitsch holds a PhD in business informatics and is currently responsible for European and National research projects within the consulting company BOC (www.boc-group.com) in Vienna, in the domain of knowledge management and technology enhanced learning. He deals with over 20 EU research projects since 2000 starting with the EU-funded projects ADVISOR, PROMOTE and EKMF and has recently been working on KM-aspects within the EU-projects Akogrimo, FIT, LD-Cast, Brein, AsIsKnown, MATURE and now coordinates plugIT. Mr. Woitsch is involved in commercial projects in the design of documentation processes, skill management and knowledge balances and is a member of the Austrian Standardization Institute contributing to the ON-Workshop 1144 "Knowledge Management". Beside his engagement at BOC he teaches at the Department of Knowledge and Business Engineering at the Faculty of Computer Science at the University of Vienna. The tight coupling between BOC and the University of Vienna is expressed in about 40 joined papers and the involvement as reviewer and member of programme committees in KM-conferences.

Florian Skopik is currently with the AIT Austrian Institute of Technology focusing on applied research of security aspects in distributed systems and service-oriented architectures. Current research interests include secure smart grids and the security of critical infrastructures. Before joining AIT, he received a bachelor degree in Technical Computer Science, and master degrees in Computer Science Management and Software Engineering and Internet Computing from the Vienna University of Technology. He was with the Distributed Systems Group at the Vienna University of Technology as a research assistant and post-doctoral research scientist, where he also received a PhD, dealing with enterprise collaboration systems and service-oriented computing. In parallel to his studies, he worked as a software developer in the industry for more than 10 years. Florian Skopik further spent a sabbatical at IBM Research India in Bangalore for several months.

Thomas Bleier joined AIT in 2005 and is currently the program manager for the research program "IT Security", focusing on applied research of security aspects in distributed systems and service-oriented architectures. Current research interests include secure system design and the security of critical infrastructures. Before joining AIT Thomas Bleier was working in the industry for more than 10 years as a Systems Architect, Project Manager, Software Developer and Technical Consultant. He holds a master's degree in Information Security Management from the Danube-University Krems, a master's degree in Technical Informatics from the Technical University of Vienna, and a degree in Computer Science and Business Administration from the Technical College HTL Wr. Neustadt, is a certified project manager (IPMA Level C) and Certified SCRUM Master, and also holds several technical certifications.