A Tool for Automatic Enterprise Architecture Modeling

Markus Buschle, Hannes Holm, Teodor Sommestad, Mathias Ekstedt, and Khurram Shahzad

Industrial Information and Control Systems, KTH Royal Institute of Technology, Osquldas v. 12, SE-10044 Stockholm, Sweden
{markusb, hannesh , teodors, mathiase, khurrams }@ics.kth.se,

Abstract. Enterprise architecture is an approach which aim to provide decision support based on organization-wide models. The creation of these models is however cumbersome as multiple aspects of an organization need to be considered. The Enterprise Architecture approach would be significantly less demanding if data used to create the models could be collected automatically.
This paper illustrates how a vulnerability scanner can be utilized for data collection in order to automatically create enterprise architecture models. We show how this approach can be realized by extending an earlier presented Enterprise Architecture tool. An example is provided through a case study applying the tool on a real network.

Keywords: Enterprise Architecture, Automatic data collection, Automatic instantiation, Software tool, Security Analysis

1 Introduction

Enterprise Architecture (EA) is a comprehensive approach for management and decision-making based on models of the organization and its information systems. An enterprise is typically described through dimensions such as Business, Application, Technology and Information. [7]. These pictographic descriptions are used for system-quality analysis to provide valuable support for IT and business decision-making [3].

As these models are intended to provide reliable decision support it is imperative that they capture all the aspects of an organization which are of relevance. Thus, they often grow very large and contain several thousands of entities and an even larger number of relationships in between them. The creation of such large models is both time and cost consuming, as lots of stakeholders are involved and many different pieces of information have to be gathered. During the creation process the EA models are also likely to become (partly) outdated [1]. Thus, in order to provide the best possible decision support it needs to be ensured that EA models both are holistic and reflect the organizations current state.

Automatic data collection and model creation would be preferable as this would mean a reduced modeling effort and an increased quality of the collected

data. In current EA tools two approaches addressing automatic data collection can be found. The most common way is to import models that are made in 3rd party software. For example, BizzDesign Architect [2] can import from office applications. Thereby the automation aspect is the fact that data is reused and does not need to be manually entered if it is already available. The interpretation of data documented in the third-party software can however be resource- and time consuming, thus contradicting parts of the purpose with automatic data collection. Other tools such as for example Troux [11] allow the usage of SQL queries in order to load information from available data bases. This approach focuses on the extraction of the data-model and thereby the automatic creation of the information architecture as well as the business architecture based on process descriptions and similar documents.

In this paper we present how the Enterprise Architecture Analysis Tool [3] has been extended in order to automatically instantiate elements in EA models based on results from network scans. In comparison to the previously described approaches of other tools our implementation focuses on the Application and Technology layer of the organization. This information is gathered through an application of a vulnerability scanner that evaluates the network structure of an enterprise. Thereby attached network hosts and the functionality they provide can be discovered. Another difference is that the presented tool uses EA models for system-quality analysis, whereas commercial applications focus on modeling. As a running example we illustrate how a meta-model designed for cyber security analysis [9] can be (partly) automatically instantiated. The presented implementation is generic and can be used to support any kind of EA analysis.

The remainder of this paper is structured as follows. Section two describes the components used to realize the implementation and introduces into the meta-model that is used as running example. Section three describes how the information, which was automatically collected, is used to instantiate the meta-model for security evaluation. Section four exemplifies the tool application on real data collected by scanning a computer network used for security exercises. In section five the presented tool and the underlying approach are discusses as well as future work is described. Finally section six concludes the paper.

2 Preliminaries

This section describes the three components that we combined in order to automatically create EA models that can be used for security analysis. In subsection 2.1 the vulnerability scanner NeXpose [8], which is used for data collection, is explained. Subsection 2.2 describes the Enterprise Architecture Analysis Tool that is used to generate the models and evaluate them with regards to security aspects. Subsection 2.3 briefly introduces CySeMoL, the used meta-model which is partly instantiated using the automated data collection. The overall architecture can be seen in figure 1.

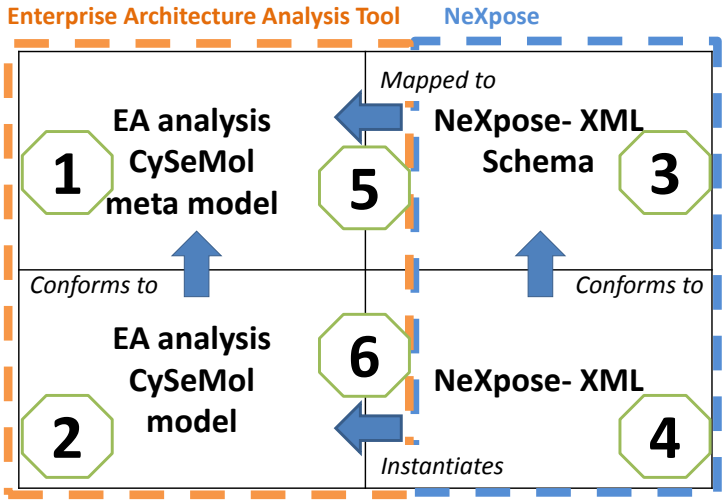

Fig. 1. The used architecture

2.1 NeXpose

The vulnerability scanner NeXpose was chosen in this project as it has demonstrated good results in previous tests [5].

NeXpose [8] is an active (i.e. it queries remote hosts for data) vulnerability scanner capable of both authenticated and unauthenticated scans. Authenticated scans involve providing the scanner with user accounts to hosts. They are typically less disturbing to normal operations and providing a higher degree of accuracy. However, it is not always the case that credentials are readily available for the individual(s) performing a scan.

NeXpose provides information regarding the network architecture in terms of all devices which are communicating over TCP or UDP, e.g. computers, firewalls and printers. The scanner identifies the operating systems or firmware that is running on the scanned devices and any services that are running. If the scanner is given credentials it is also able to assess all applications (and versions thereof) installed on a device and all user/administrator accounts on that device.

More security related functions of the scanner include that it can check for both software flaws and configuration errors. It is also capable of performing web application scans. NeXpose has approximately 53000 current signatures in its engine, with every signature corresponding to a certain vulnerability. NeXpose is also SCAP-compliant [6] and thus compliant with a suite of six commonly used protocols developed by the National Institute of Standards and Technology (NIST): i) Extensible Configuration Checklist Description Format (XCCDF), ii) Open Vulnerability and Assessment Language (OVAL), iii) Common Platform Enumeration (CPE), iv) Common Configuration Enumeration (CCE), v) Common Vulnerabilities and Exposures (CVE) and vi) Common Vulnerability Scoring System (CVSS).

2.2 Enterprise Architecture Analysis Tool

In [3] we presented a tool for EA analysis. This tool consists of two parts to be used in succession. The first component allows the definition of meta-models to describe a certain system quality of interest (**1** in Figure 1). This is done according to the PRM formalism [4] in terms of classes, attributes, and relations between them. Thereafter an execution of the second component is performed in order to describe an enterprise as an instantiated model (**2** in Figure 1), which is compliant to the previously defined meta-model. As the PRM formalism supports the expression of quantified theory the described enterprise can be evaluated with regards to the considered system quality described in the first component.

To use the results gained from NeXpose scans an extension of the tool was necessary. The result of NeXpose's scans can be exported to XML files (**4** in Figure 1), which are structured according to a schema definition file (XSD)[1] (**3** in Figure 1). We added the possibility to create mappings between XSD files and meta-models (**5** in Figure 1) in order to automatically instantiate the meta-model based on NeXpose's XML files (**6** in Figure 1). The used mapping is discussed in section 3.

2.3 CySeMoL

This paper exemplifies the mapping functionality by instantiating a subset of the meta-model of the CySeMoL (Cyber Security Modeling Language)[10]. This modeling language follows the abstract model presented in [9] and uses the PRM formalism to estimate the value of security attributes from an architecture model. Its meta-model covers both technical and organizational aspects of security and does in total contain 20 entities, 30 entity-relationships and a number of inter-dependent attributes. Four of these entities and three of its relationships can be mapped to elements produced by NeXpose. This subset of CySeMoL is depicted in the left part of Figure 2. While only a subset of the total number of entities and relations could be instantiated, this subset includes entities and relations which are of high multiplicity in enterprises, and thus require lots of effort to model.

3 The mapping

In this section we describe how we matched the structure of NeXpose's results to entities of the CySeMol language in order to instantiate the language based on scans. As described earlier, this was done based on the XSD file that describes the structure of the reports.

For our implementation we used four elements that a NeXpose result contains. At first we mapped *fingerprintsType* and *osType* to the *OperatingSystem*

[1] The XSD file (Report_XML_Export_Schema.xsd) is part of the NeXpose Community Edition that can be downloaded from http://www.rapid7.com

class of CySeMol, visualized as **Mapping 1** in figure 2. This allows us to determine the used operating system of a computer identified by NeXpose. The second mapping (**Mapping 2** in figure 2) relates *softwareType* and *fingerprintType* to *SoftwareProduct* in order to identify the software that is executed on the considered system. Thirdly (**Mapping 3**) we mapped *endpointsType* and *endpointType* to *Service* in order to identify at which ports services are provided by a machine. Finally a mapping between *service_fingerprints_Type* and *service_fingerprint_Type* to *SoftwareProduct* was made (**Mapping 4**) in order to describe the software that provide services on the machine of interest.

Additionally we considered the hierarchical structure of the XSD file in order to derive relationships. This made it possible to add the relationships *Operates*, *ControlledBy*, and *ProductOf* as they are shown in Figure 2.

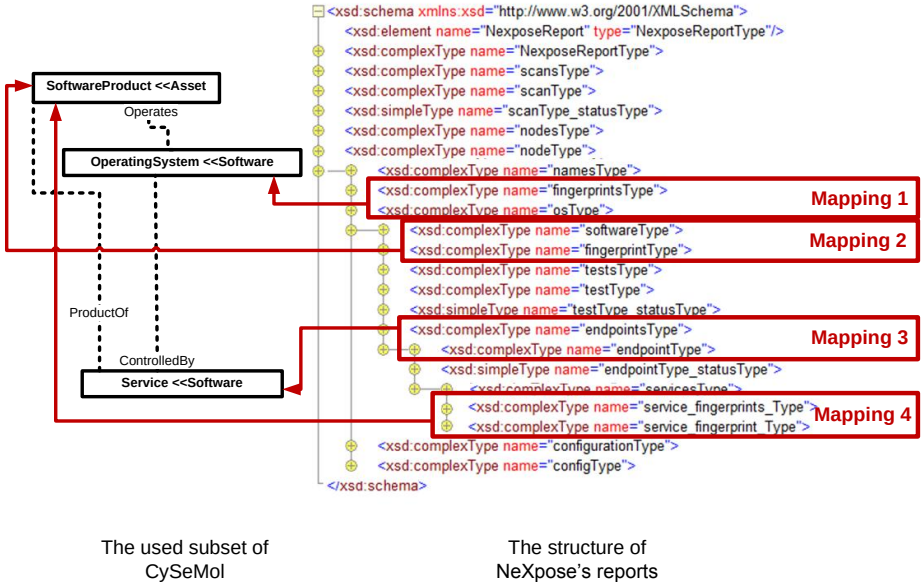

Fig. 2. The implemented mapping

4 Example

In this section we describe how we tested the implementation on a real network. We give a brief introduction to the background of the collected data. Afterwards we depict how the resulting auto-generated model looks like.

4.1 The setup

The main experimental setup was designed by the Swedish Defence Research Agency (FOI) in Linköping, Sweden with the support of the Swedish National

Defence College (SNDC). Also, a group of computer security specialists and computer security researchers originating from various northern-European governments, military, private sectors and academic institutions were part of designing the network architecture.

The environment was set to describe a simplified critical information infrastructure at a small electrical power utility. The environment was composed of 20 physical PC servers running a total of 28 virtual machines, divided into four VLAN segments. Various operating systems and versions thereof were used in the network, e.g. Windows XP SP2, Debian 5.0 and Windows Server 2003 SP1. Each host had several different network services operating, e.g. web-, mail-, media-, remote connection- and file sharing services. Furthermore, every host was more or less vulnerable through software flaws and/or poor configurations.

4.2 The result

We performed a NeXpose scan on the setup environment and thereafter applied the mapping as presented in chapter 3. The resulting auto generated model consists of 28 instances of CySeMol's *OperatingSystem* class. Furthermore 225 instances of the *Service* class and 141 instantiations of the *SoftwareProduct* class were automatically generated. The generated components were related based on the relations that are specified in CySeMol. Figure 3 shows the resulting model exemplary for one computer of the environment as the full model is to big to be shown here.

5 Discussion and future work

This paper demonstrates that vulnerability scanners can provide useful support for the creation of EA models. As mentioned earlier, the results of a scan do not deliver a complete EA model, but require some completion work. The application of an automated scan however significantly reduces modeling effort and provides an EA analyst with a model stub which he or she can complement with other types of data.

The validity and reliability of the proposed approach can be discussed from two different viewpoints: i) how much of the meta-model that can be captured, both in scope (i.e. how much of the meta-model that can be instantiated) and context (i.e. if the scanner provides all the information needed to accurately capture the context of a variable), and ii) how accurate a vulnerability scanner is at assessing the instantiated variables. Regarding i), most of the more modeling intensive concepts of CySeMoL are captured and all context are accurate. That is, the scanner provides e.g. all the information regarding vulnerabilities that CySeMoL requires. Regarding ii), the scanning accuracy in terms of assessing vulnerabilities is studied in [5]. The accuracy in terms of assessing software, operating systems and such is something that will be examined in future works.

It would also be interesting to look at other variables provided by automated vulnerability scanning, e.g. user accounts of systems. Furthermore, automated

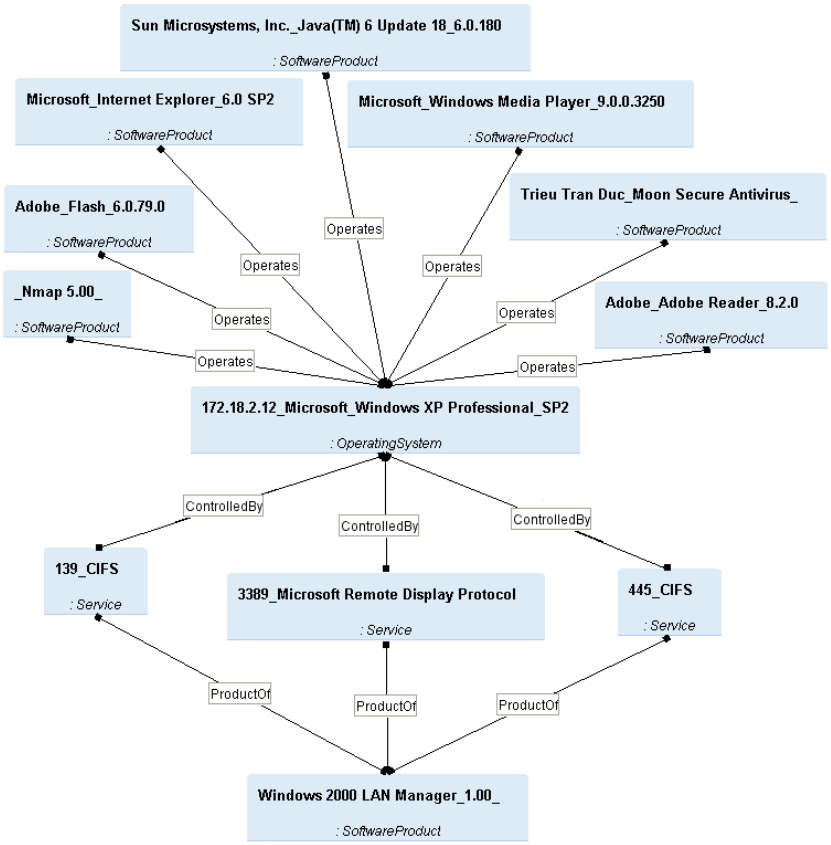

Fig. 3. The implemented mapping

scanning could be mapped to more commonly used EA frameworks such as ArchiMate [7] to increase the usage of the method.

Additionally in future work it might be investigated how other data sources can be used in order to provide input to automatic model creation and further reduce the manual tasks necessary. Examples of such sources are access control lists, ERP systems, accounting systems and UDDI registries. Especially how automatic data collection for the domains that so far not have been considered (the Business Layer and the information architecture) can be carried out, needs to be investigated. The long-term goal is to minimize the manual effort required to generate EA models.

The fact that enterprises are changing in the course of time is an important aspect too. The support for periodic scans leading to an automatic model update might therefore be implemented in the present tool as well.

It is also possible to collect information on vulnerabilities of services and software. This is something that we aim to incorporate in a future project in order to improve the analysis functionality.

6 Conclusion

In this paper we presented an extension of our previously developed tool that allows the automatic generation of elements for Enterprise Architecture models. The input for these models is provided by a vulnerability scanner, which was used to identify infrastructure elements and applications that were part of a computer network. Our implementation is generic even though CySeMol, a meta-model for security analysis, was used as a running example. The data gained from the vulnerability scanner can be used to instantiate any meta-model, as soon as a mapping has been defined. The scan with NeXpose took less than an hour and the creation of the EA model using that data was next to instantaneous. Thus, it should be a viable option for EA architects. We have also illustrated the architecture of our implementation and described used components in detail. Finally, we have presented a practical application based on real data of our implementation. Thereby we have shown the feasibility of our approach.

References

1. Aier, S., Buckl, S., Franke, U., Gleichauf, B., Johnson, P., Närman, P., Schweda, C., Ullberg, J.: A survival analysis of application life spans based on enterprise architecture models. In: 3rd International Workshop on Enterprise Modelling and Information Systems Architectures, Ulm, Germany. pp. 141–154 (2009)
2. BiZZdesign: BiZZdesign Architect. `http://www.bizzdesign.com` (Mar 2011)
3. Buschle, M., Ullberg, J., Franke, U., Lagerström, R., Sommestad, T.: A tool for enterprise architecture analysis using the prm formalism. In: CAiSE2010 Forum PostProceedings (2010)
4. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: Proc. of the 16th International Joint Conference on Artificial Intelligence. pp. 1300–1309. Morgan Kaufman (1999)
5. Holm, H., Sommestad, T., Almroth, J., Persson, M.: A quantitative evaluation of vulnerability scanning. Information Management & Computer Security (to be published)
6. Johnson, C., Quinn, S., Scarfone, K., Waltermire, D.: The technical specification for the security content automation protocol (SCAP). NIST Special Publication 800, 126 (2009)
7. Lankhorst, M.M.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer, Berlin, Heidelberg, Germany, 2nd edn. (2009)
8. Rapid7: NeXpose. `http://www.rapid7.com/` (Mar 2011)
9. Sommestad, T., Ekstedt, M., Johnson, P.: A probabilistic relational model for security risk analysis. Computers & Security 29(6), 659–679 (2010)
10. Sommestad, T., Ekstedt, M., Nordström, L.: A case study applying the Cyber Security Modeling Language (2010)
11. Troux Technologies: Metis. `http://www.troux.com/products/` (Mar 2011)