Security Characterisation and Integrity Assurance for Software Components and Component-Based Systems

Jun Han and Yuliang Zheng
Peninsula School of Computing and Information Technology
Monash University, McMahons Road, Frankston, Vic 3199, Australia
e-mails: {jhan,yzheng}@monash.edu.au

Abstract

Software systems are increasingly assembled from components that are developed by and purchased from third parties, for technical and economic gains. In such component based software development, the functionality and quality-of-service attributes of the software components should be clearly and adequately specified (or packaged) through their interfaces, so that the characteristics of the systems assembled from the components can be analysed relative to the system requirements. In this paper, we consider one particular quality-of-service attribute, i.e., *security*, and outline an approach to (1) specifying the security characteristics of software components and (2) analysing the security properties of component-based systems in terms of their component characteristics and system architectures. The approach is partially based on the *Common Criteria for Information Technology Security Evaluation*. In addition, we also introduce our work on ensuring the integrity of software components as part of the infrastructural support for component based software engineering.

1 Introduction

Component based software engineering (CBSE) has recently attracted tremendous attention from both the software industry and the research community. It has been widely recognised that more and more software systems are being built by assembling existing and new components. In Web-based systems, for example, the software components may even be distributed over the Internet and dynamically assembled into target systems. It has been shown that CBSE not only delivers technical benefits for the development of large scale systems, but also has positive impacts on the management and structuring of projects and organisations [1]. A key to the success of CBSE is its ability to use software components that are often developed by and purchased from third parties. In such a scenario, it is the norm that the components are delivered in binary form and their source code and design information are not available to the *system* developers. As such, the software components should be adequately specified or packaged through their interfaces, to facilitate proper usage.

In general, the interface specification or packaging of a software component should involve the syntactic and semantic specification of its functional interface and the specification of its quality-of-service attributes (such as security, reliability and performance). Support for syntactic interface specification has been well studied in the form of interface definition languages (IDLs), e.g., those from the three major industry leaders: Sun's JavaBeans, Microsoft's COM components, and the expected CORBA components. While having made CBSE practical, these industrial standards generally do not support semantic interface specification. To achieve

component (re)use with confidence, precise semantic specification of component interfaces is necessary. Semantic specifications of individual interface operations have been advocated in object oriented programming languages like Eiffel [5] and CBSE approaches like Catalysis [2], in the form of pre-/post-conditions. Additional semantic constraints about how the interface elements of a component depend on each other and how the component is to interact with other components should also be specified [3].

In [3], we have proposed a framework for defining interfaces of software components. This framework not only deals with the syntactic and semantic specification of component interface, but also allows the specification of non-functional quality-of-service (QoS) attributes (code named *illities* [7]) of components. In the context of building systems from existing components, the characterisation of the components' illities and their impact on the assembled systems are particularly important because the components are usually provided as blackboxes.

For a particular non-functional property or QoS attribute, we need to address two issues: (1) how to characterise that specific property for a given component, and (2) how to analyse the component's impact on the enclosing system in a given context of use (i.e., in the context of a system architecture). A related issue is whether the characterisation of the non-functional property will change in different contexts of use. The interface definition of component illity characterisation is dependent on the specific characterisation models developed. In this paper, we investigate the security aspect of software components and its impact on system composition, and outline an approach to the development of a security characterisation model for software components and component-based systems. Besides, we also identify the need for ensuring the integrity of software components.

In the next section, we give a brief overview of our general framework for the characterisation of software components. In section 3, we present our approach to security characterisation of software components and component-based systems, which is partially based on the *Common Criteria for Information Technology Security Evaluation* [6]. In section 4, we introduce our work on ensuring the integrity of software components as part of the infrastructural support for component based software engineering. Finally we conclude in section 5.

2 A framework for software component characterisation

As argued in the previous section, proper characterisation of software components is essential to their effective management and use in the context of component based software engineering. While there have been industrial and experimental projects that build systems from (existing) components, the approaches taken are ad hoc and heavily rely on the specifics of the systems and components concerned. That is, component-based system development is still very much in its infancy, and there are no proven systematic approaches to follow. Characterisation of components through comprehensive interface definition is a step towards such systematic approaches and their enabling technologies. In this section, we introduce a model for comprehensive component interface characterisation, in the hope to provide a basis for the development, management and use of components.

Figure 1 shows the overall structure for component interface characterisation in our framework. At the bottom level is the *interface signature* of the component, which forms the basis for the component's interaction with the outside world and includes all the necessary mechanisms for such interaction (i.e., properties, operations and events). The next level up is the *interface constraints* about the component signature to ensure their proper use. That is, the use of the component will be subject to these constraints. The interface signature and constraints of a

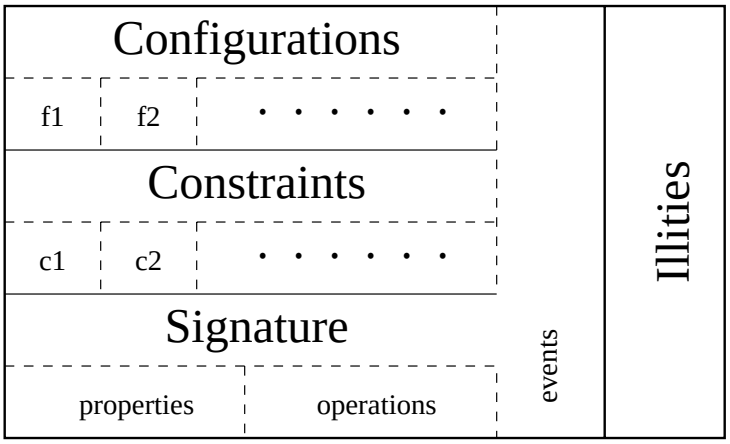

Figure 1: Structure of Component Interface

component define the overall capability of the component. The third level concerns the *packaging* of the interface signature according to the component's roles in given contexts of use, so that the component interface has different *configurations* depending on the use scenarios. The fourth aspect of component interface is about the characterisation of the component in terms of their *non-functional QoS attributes* or *illities*. The non-functional properties occupy a special place in this component interface structure, and may interact with the interface signature and configurations.

Further details about the framework can be found in [3], especially about interface signature, interface constraints and interface configurations. In the next section, we consider security as one of the illities in the context of this framework, and outline an approach to security characterisation of software components and component-based systems.

3 An approach to security characterisation

Security is an important aspect of software systems, especially for distributed security-sensitive systems. When we assemble systems from existing components, it is vital that we must be able to trust these components. Therefore, we need to be aware of the security characteristics of these components and their impact on the target systems. In order to provide such security-related information of a component, a model for security characterisation of components and component-based systems is required to augment our framework for component interface definition.

Security of information technology products and systems. Over the years, there has been much effort in evaluating Information Technology (IT) security, i.e., the security properties of IT products and systems, including hardware, software and firmware. There have been the *Trusted Computer System Evaluation Criteria (TCSEC)* developed in the United States, the *Information Technology Security Evaluation Criteria (ITSEC)* developed by the European Commission, the *Canadian Trusted Computer Product Evaluation Criteria (CTCPEC)*, and the *Federal Criteria for Information Technology Security (FC)* from the United States. Since the early 1990s, the sponsoring organisations of the above standards under the coordination of ISO have been working together to align their criteria and create a single international standard of IT security evaluation criteria for general use in the global IT market. The result

is the current draft international standard, *Common Criteria for Information Technology Security Evaluation, version 2.0* – commonly referred to as the *Common Criteria* or simply *CC*, released in May 1998 [6].

Given the ever-increasing wide-spread use and trade of IT products, IT security concerns are not only for the highly security-sensitive IT products. In fact, any IT products acquired from the market present certain security risks, although with different levels of sensitivity. To use the acquired IT products with confidence, their security properties must be measured and made explicit. The Common Criteria represent a coordinated effort addressing this issue.

In component based software engineering, the security issue becomes more prevalent. Many components of a target software system to be assembled may be acquired from or delegated to third parties. The security properties of each component will be part of and impact on the target system's security. In such a scenario, we must know the security characteristics of the components to be able to evaluate the assembled system. Another equally important aspect that impacts on the target system's security is the system architecture that connects the components in a specific manner. In addressing the issue of security characterisation of software components and component-based systems, we propose to

1. identify and measure the security characteristics of a software component through the use and adaptation of the Common Criteria, and

2. analyse and evaluate the security properties of a composed system in terms of characteristics of its components and its system architecture.

The Common Criteria identify the various security requirements for IT products and systems, and provide a good starting point for characterising software components, i.e., with the components being regarded as IT products/systems. However, the Common Criteria do not directly address system composition, and therefore much investigation is required to evaluate a composed system based on the component characteristics and the system architecture.

Security characteristics of software components. As mentioned above, we propose to use the Common Criteria as the basis of a security characterisation model for software components. The Common Criteria provide a framework for evaluating IT systems, and enumerate the specific security requirements for such systems. The security requirements are divided into two categories: security functional requirements and security assurance requirements. The *security functional requirements* describe the desired security behaviour or functions expected of an IT system to counter threats in the system's operating environment. These requirements are classified according to the security issues they address, and with varied levels of security strength. They include requirements in the following classes: security audit, communication, cryptographic support, user data protection, identification and authentication, security management, privacy, protection of system security functions (security meta-data), resource utilisation, system access, and trusted path/channels.

The *security assurance requirements* mainly concern the development and operating process of the IT system, with the view that a more defined and rigorous process delivers higher confidence in the system's security behaviour and operation. These requirements are classified according to the process issues they address, and with varied levels of security strength. The process issues include: life cycle support, configuration management, development, tests, vulnerability assessment, guidance documents, delivery and operation, and assurance maintenance. The Common Criteria have also identified seven evaluation assurance levels by including assurance requirements of appropriate strength into each of these levels.

Besides, the functional and assurance security requirements for a particular IT system are usually specified in a *security target* document by selecting and instantiating the relevant requirements with particular security strengths from the above general classes. This document is developed according to a security policy, and is the basis of the security assurance process.

In characterising the security properties of a software component, we regard the component as an IT system to be evaluated according to the Common Criteria. Such characterisation will include both the security functional properties and security assurance properties of the component. These properties will identify which requirements of the Common Criteria are met at which levels of security strength. For example, a set of properties based on the Common Criteria will characterise how user data is protected with which levels of strength.

For a given software component, only certain security requirements may apply depending on the nature of the component. In general, therefore, a CC requirement may or may not be applicable; for an applicable CC requirement, the component will have a specific level of protection strength. This applies to both the security functional requirements and the security assurance requirements. For the security assurance requirements, the characterisation of a component may directly use the detailed individual requirements, use one of the more coarse-grained evaluation assurance levels, or use a combination of both. The use of the detailed requirements will provide more information for system analysis.

Such a CC-based security characterisation of software component will be similar to a security target document, except that the characterisation shows the security properties that the component possesses while the security target document sets out the security requirements for the component that may or may not be actually realised. Besides, the characterisation should take a more formulated and succinct form rather than a lengthy document, but may have additional justification documentation.

Given the large number of security requirements to be considered under the Common Criteria, tool support is very much desirable. The tools will manage the security evaluation framework as well as the security properties of given components. In general, the characterised security properties of a component are delivered together with the component as meta-data, just like other interface definition information of the component. The Common Criteria as an international standard provide an ideal starting point for the understanding and exchange of the security characterisation information.

We are currently analysing the security functional requirements of the Common Criteria to formulate a practical model for characterising the security properties of software components. Among the issues addressed are the formalisation of individual requirements and their dependencies. At the same time, relevant tool support is also being investigated.

Security properties of component-based software systems. The security characterisation of a software system assembled from components should take a form similar to that of a component. After all, the composed system is an IT system and may be used as a component of another larger system. As such, the security characterisation of the target system could be done in a way similar to that of an *atomic* component. Given that the security properties of the components used are already available, however, it is natural and advisable to use these component properties together with the system's composition architecture and process to arrive at the system's security characterisation. That is, the security properties of a component-based system depend on those of the components used and the system architecture. Assuming the component properties are characterised and defined as outline above, we have to consider the ways of interaction between these components according to the system architecture and how

these interactions impact on the components and the composed system. Therefore, we need a component- and architecture-based composition model for software security. Unfortunately, the Common Criteria do not directly address system composition issues. According to the security concerns covered by the Common Criteria, this software security composition model should be based on the following aspects:

1. the security properties of individual components,
2. the system architecture of the target system, and
3. the process of architecture design and system composition.

While the first two items contribute to both the security functional properties and the security assurance properties of the target system, the last item is mainly concerned with the system's security assurance properties.

In developing the security composition model, we need to consider the security compatibility of the components as dictated by the architectural interactions, the trade-offs and compromises between individual components' security strength in the system context, the derivation of system-wide properties from component properties and component interactions, the security impact of the overall architecture topology, and the relationships or dependencies between the system and its underlying enabling technologies (as part of the system's security environment). We are currently investigating these issues.

4 Integrity assurance for software components

In the previous sections, emphasis has been placed on characterising inherent properties of software components, especially those pertinent to security. Another important issue in the development, distribution and application of software components is related to the *integrity* of the components, namely, how to ensure that any unauthorised modification of a software component, be it accidental or malicious, can be easily detected by a customer or another software component that depends on it. It applies to not only the implementation of the component functionality but also the characterisation or interface definition of the component. This issue is especially important in dynamically configurable distributed software systems, where system components may be acquired or purchased on the Internet on a per-use basis. It has the same importance for software systems involving mobile agents.

We further pursue this direction of research in [4], where we propose a comprehensive authentication infrastructure that supports the integrity of software components, including commercial-of-the-shelf software. This authentication infrastructure is an integral part of the infrastructural support for component based software engineering.

5 Conclusions

In this paper, we have proposed an approach to the security characterisation of software components and component-based systems, and introduced our work on ensuring the integrity of software components as part of the infrastructural support for component based software engineering. Our approach to security characterisation is partially based on the draft international security evaluation standard, the Common Criteria, and aims to develop a composition model

for software security. Our work on integrity assurance focuses on the development of a comprehensive authentication infrastructure for the development, distribution and use of software components.

References

[1] CBSE98. *Proceedings of International Workshop on Component-Based Software Engineering*. Kyoto, Japan, April 1998.

[2] D. D'Souza and A.C. Wills. *Objects, Components and Frameworks with UML: The Catalysis Approach*. Addison-Wesley, 1998.

[3] J. Han. A comprehensive interface definition framework for software components. In *Proceedings of 1998 Asia-Pacific Software Engineering Conference*, pages (in press – 8 pages), Taipei, Taiwan, December 1998. IEEE Computer Society.

[4] J. Han and Y. Zheng. Infrastructural support for the integrity of software components. Work in progress, Peninsula School of Computing and Information Technology, Monash University, Melbourne, Australia, 1998.

[5] B. Meyer. *Object-Oriented Software Construction*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 1997.

[6] Common Criteria Project. *Common Criteria for Information Technology Security Evaluation*. NIST, USA, http://csrc.nist.gov/cc/, May 1998.

[7] C. Thompson. *Workshop on Compositional Software Architectures: Workshop Report*. http://www.objs.com/workshops/ws9801/report.html, Monterey, USA, January 1998.