

# A Software Engineering Perspective for Services Security

Jun Han

School of Information Technology  
Swinburne University of Technology  
John Street, Hawthorn, Vic. 3122, Australia  
jhan@it.swin.edu.au

**Abstract.** Services are usually developed and deployed independently; and systems can be formed by composing relevant services to achieve set goals. In such an open and dynamic environment, security is of paramount importance. We have seen much work in the traditional area of information and network security, focusing on developing various security techniques. More recently, there have been efforts in integrating the security techniques into languages and infrastructural support that are used for developing services and systems. In fact, the development of services and the composition of service-based systems are software engineering activities. As such, they need to be viewed from a software engineering perspective. In this paper, we introduce an approach to services security engineering, to answer the questions like what the security properties of services and service-based systems are and how they meet the user's security requirements. It deals with the issues of (1) security property characterisation for services, (2) compositional security analysis for service-based systems, and (3) certification of services.

## 1 Introduction

Service oriented computing has promised a new way to deliver information technology support for individuals and businesses. Services in this context, including Web and Grid services, are software applications that are deployed over standard computing platforms, and are aimed at being integrated or composed with each other to form Internet-based systems and perform cross-application transactions. As the Internet is a hostile environment, security for services and their compositions is of great concern.

In the broad context of dealing with software and system security, the current practices adopt a *defensive*, *retrospective* and *reactive* line of thinking. That is, they tend to follow the path of patching up security holes found in systems that are often built without systematic security considerations [3], or security being treated as a “after-thought” or “add-on” in system development such as firewalls, sandboxes and security wrappers [12, 11]. While these may be the most practical ways available to deal with system security, it definitely does not represent a satisfactory situation. Instead, a more appropriate engineering approach should be taken. In general, this requires

- the understanding of the security risks and requirements for a system,
- the availability of security techniques that can be used, and
- the way of how the system can be developed using the security techniques to meet its security requirements.

While the development of various security techniques such as encryption algorithms and key exchange protocols has been the main topic of the information security community, there has been limited study into the software engineering aspects of system security, i.e., how to use the security techniques in software and system development to satisfy system security requirements in a *proactive* and *predictive* manner.

In this paper, we focus on the software engineering issues concerning the security of services and service compositions. In particular, we consider the following two aspects:

1. For an individual service, its specific (ensured and required) security properties need to be characterised and published so that the potential users can assess its suitability in given contexts of use.
2. For a service composition, we need to analyse the compatibility of the interacting services' (ensured and required) security properties and deduce the system-level security properties based on those of the individual services.

Only by achieving these two objectives can we answer the questions concerning the security of services and service-based systems. Otherwise, there will always be security concerns about composing independent third-party services, which will undermine the future of service oriented computing as a whole. In the remainder of this paper, we introduce an approach to services security engineering, aiming to address the above issues.

## 2 An Approach to Services Security Engineering

The security properties of a service will be part of and impact on the security of a system that uses it. As such, they must be characterised and made explicit so that the users can be aware of its security characteristics and use it with confidence. Another equally important aspect that impacts on the target system's security is the system's composition architecture that connects the services in a specific manner. In addressing the issue of security characterisation for services and service-based systems, our approach has three major components:

1. characterise and publish the security properties of individual services through the use, adaptation and formalisation of the Common Criteria [2],
2. certify the security properties of services against their implementations, and
3. analyse and deduce the security properties of a composed system in terms of the characteristics of its services and its composition architecture.

While our research has been mainly on the first and third aspects (characterisation and compositional analysis), we also give an account of the specific requirements for the second aspect (certification). Note that our approach is set in the context of a general framework for component-based software [6].

## 2.1 Characterisation and Publication of Service Security

The ISO/IEC International Standard 15408, *Common Criteria for Information Technology Security Evaluation, version 2.1* – commonly referenced as the *Common Criteria* or simply *CC* [2], identifies the various security requirements for IT products and systems, and provides a good starting point for characterising the security properties of services, i.e., with the services being regarded as IT products/systems. Using the Common Criteria, we are able to analyse and identify the types of security properties and the levels of security strength that a service has implemented. For example, a set of properties based on the Common Criteria can be used to characterise how user data is protected with which levels of strength.

As the Common Criteria are written in natural language and are not amenable to formal analysis. To provide precise and automated support for security analysis, the specification of security properties should take a more formal and succinct form than a lengthy informal document. We have analysed the security functional requirements of the Common Criteria and formulated a formal model for service security characterisation and specification. For a given service, we distinguish its ensured and required properties. A *required* property is one that has to be satisfied by a user (i.e., another service) when the user wants to use the service or a particular functionality of the service. An *ensured* property is one that the service guarantees to its users when performing certain functionality, which may be subject to certain required properties being met. A required or ensured security property is a formalised statement that states a fact about or dependency between certain security properties. It adopts a logic programming style. The following example illustrates our approach to security characterisation and specification for services.

Let us consider an online tax return processing system. The tax office provides an online tax-processing service that processes people’s tax returns. People use a submission client (i.e., another service) to submit their tax forms containing all the required information. The tax processing service requires that the tax form be encrypted with the tax-processing service’s public key before submission for the purposes of confidentiality and integrity. Some of the security properties associated with the tax-processing service *t*’s tax return lodgment functionality are as follows:

$$\begin{aligned} & \textit{owned}(k, \textit{this}). \\ & \textit{owned}(k^{-1}, \textit{this}). \\ & \textit{signed}(\textit{tax\_statement}, k^{-1}) \leftarrow \textit{encrypted}(\textit{tax\_form}, k), \textit{owned}(k, \textit{this}). \end{aligned}$$

The first two formulas state the properties that the service owns a public key  $k$  and private key  $k^{-1}$ . The third formula states that if the submitted tax form is encrypted with the tax processing service’s public key, then the tax processing statement will be returned signed with the service’s private key for verifying authenticity. Note that in this formula, there are two required property statements (in the tail of the formula) and one ensured property (i.e., the head of the formula). A further property could state that the tax statement is also encrypted using the submitter’s public key.

On the other hand, the submission client  $c$  may have the following properties:

$$\begin{aligned} & \textit{owned}(k, t). \\ & \textit{encrypted}(\textit{tax\_form}, k). \\ & \textit{sees\_signed}(\textit{this}, \textit{tax\_statement}) \leftarrow \\ & \quad \textit{signed}(\textit{tax\_statement}, k^{-1}), \textit{owned}(k^{-1}, t). \end{aligned}$$

The first two formulas state that the submission client ensures that the tax form is encrypted with the tax-processing service's public key. The third formula states that the submission client requires that the tax processing statement be signed with the tax-processing service's private key so that it can verify the statement's authenticity by applying the tax-processing service's public key.

In general, the required and ensured security properties of a service together with their dependencies need to be included in a service's published description. In this way, the potential users can assess the service's suitability. In the above example, both the tax-processing service and the submission client can access each other's security properties and assess if the other service satisfies their own security requirements (see section 2.3 for further discussion).

## 2.2 Certification of Service Security

A service has an implementation and a description. In particular, the security property description of a service should reflect the security measures adopted in the service's implementation. In deploying such a service, however, how can a potential user be assured that the implementation actually conforms to the description and any unauthorised modification of the service (either the implementation or the description), be it accidental or malicious, can be easily detected? This is about the *integrity* of the service, and concerns both the service implementation and description. This issue is especially important in the context of dynamically configurable service-based systems, such as those using Web services, Grid services or mobile agents.

To satisfy the above integrity requirement for a service, the service description should be packaged together with the service implementation through techniques like introspection. Then the packaged service needs to be verified, certified, digitally stamped [4], and sealed by a certification authority. The certified service's description also contains certification-related information, including details about the certificate, certification stamp, validity period and so on, which can be revealed when queried. This information is read-only to other services, and can only be altered by the issuing certification authority.

In general, services can only be tested and certified individually, not within the context of the complete composed system [7]. The certification process involves the following tasks. First, the conformance between the service implementation and the service description needs to be checked, including the security properties. Second, the service description is approved and certified based on the result of the conformance check. Third, the service implementation and description is sealed for integrity. Finally, the issued certificate is registered so that

its authenticity can be verified by interested parties. The certificate contains a unique service identifier that is accessible from the service description by others.

The certified assurances must be verifiable statically and dynamically by other services or system integrators. Once certified, a re-compilation of the service would automatically erase all the relevant certification and identity related information. In fact, a tampered description or implementation would result in a void certificate, and this could be established by the contacting services from the information packaged with the service. If the service needs to alter its security properties, it requires a new certificate after the re-compilation. In general, such a certification scheme for services (including their implementations and descriptions) will significantly increase the user confidence in these services.

While the evaluation and certification of services is an important part of our approach, we primarily rely on existing technology and infrastructure as well as others' research in this regard. For example, the certification authority could be performed by government security evaluation agencies such as the Digital Signal Directorate in Australia. More on service or software component certification can be found in [4, 13].

### **2.3 Security Analysis for Service-Based Systems**

A service-based system is a composition of individual services. These individual services are usually provided by third parties and consequently their development information is not available. To analyse the system's security characteristics, we have to rely on the published security properties of the individual services and its composition architecture. As such, we need a service-based, architecture-directed composition model for security analysis.

While highlighting the types of security properties for services and systems, the Common Criteria do not directly address system composition issues. In developing the security composition model, we need to consider the security compatibility of the services as dictated by the architectural interactions, the trade-offs and compromises between individual services' security strength in the system context, the derivation of system-wide properties from service properties and service interactions, the security impact of the overall architecture, and the relationships or dependencies between the system and its underlying enabling technologies (as part of the system's security environment). To date, we have focused on two of these issues, namely, the security compatibility between interacting services and the derivation/checking of system-wide security properties.

In the tax processing example given earlier, for instance, a question concerns whether or not the given tax-processing service and the tax submission client actually satisfy each other's security requirements for carrying out the tax lodgment activity. In fact, they do satisfy each other's requirements as follows: the submission client's ensured property of using the processing service's public key to encrypt its tax form satisfies the corresponding required property of the processing service; consequently, the processing service ensures that the tax statement will be digitally signed using its private key before sending to the submission client; in turn, this satisfies the submission client's corresponding

security requirement for receiving its tax statement. As such, the tax lodgment transaction can be carried out between the two services. On the other hand, let us assume that the submission client service also requires the tax statement being encrypted (for confidentiality) as well as signed. This additional requirement can not be met by the processing service. Consequently, the submission client will not (be able to) use that processing service.

Regarding the checking or derivation of system-wide security properties, let us extend the above example with a banking service. The banking service  $b$  provides a functionality for settling tax returns, i.e., receiving instructions from the tax processing service, transferring money between relevant accounts, and notifying both the tax processing service and the submission client about the tax settlement. For the interaction with the tax processing service, the banking service has the following security properties:

$$\begin{aligned} & \textit{owned}(k2, \textit{this}). \\ & \textit{owned}(k2^{-1}, \textit{this}). \\ & \textit{signed}(\textit{tax\_settlement}, k2^{-1}) \leftarrow \\ & \quad \textit{encrypted}(\textit{tax\_instruction}(c), k2), \textit{owed}(k2, \textit{this}). \end{aligned}$$

The relevant security properties of the tax processing service in relation to the banking service functionality are as follows:

$$\begin{aligned} & \textit{owned}(k2, b). \\ & \textit{encrypted}(\textit{tax\_instruction}(c), k2). \\ & \textit{sees\_signed}(\textit{this}, \textit{tax\_settlement}) \leftarrow \\ & \quad \textit{signed}(\textit{tax\_settlement}, k2^{-1}), \textit{owned}(k2^{-1}, b). \end{aligned}$$

Note that the above two groups of formulas are very much similar to those concerning the relationship between the tax processing service and the submission client. We can see that the tax processing service  $t$  and the banking service  $b$  satisfy each other's security requirements, i.e., compatible at the peer level.

For the interaction with the submission client, the banking service has the following security properties:

$$\begin{aligned} & \textit{signed}(\textit{tax\_settlement}, k2^{-1}) \leftarrow \\ & \quad \textit{encrypted}(\textit{tax\_instruction}(c), k2), \textit{owed}(k2, \textit{this}). \end{aligned}$$

The above formula states that the tax settlement notice is also sent to the tax submission client. Correspondingly, the submission client has the following security properties:

$$\begin{aligned} & \textit{sees\_signed}(\textit{this}, \textit{tax\_settlement}) \leftarrow \\ & \quad \textit{signed}(\textit{tax\_settlement}, k2^{-1}), \textit{owned}(k2^{-1}, b). \end{aligned}$$

The formula states that the client sees its tax settlement from the banking service signed with the banking service's private key. In fact, this represents a system-wide property, which can only be deduced from combining the two compatible binary security compositions between the submission client  $c$  and the processing service  $t$  and between the processing service  $t$  and the banking service  $b$ .

We are currently developing a prototype tool kit that supports the publication of the security properties for services as part of their descriptions (just like their functional description information), and the compositional security reasoning for service-based systems, including analysis of peer-level security compatibility between services and derivation/checking of system-wide security properties.

### 3 Related Work

As mentioned earlier, there has been a long history and much work in developing techniques for information and network security. Encryption algorithms, digital signature schemes, security key exchange protocols and firewalls are just some of these techniques. They are the basic techniques for implementing system security, just like other programming techniques for implementing system functionality. We are all aware that basic programming techniques is not adequate for building large-scale complex systems. Similarly, basic security implementation techniques are not sufficient for dealing with the security of such complex systems.

Modularity has been an essential tool for dealing with software complexity, and has allowed us to introduce the concepts of software components and component-based systems [6]. Functionally, we need to describe *what* a component does, and to analyse how the functional requirements of the system are met based on those of the components. For security, we need the corresponding techniques for security description and compositional analysis [9]. As services are essentially independent software components, the same argument applies to services and service-based systems. Therefore, there is not only the need to have techniques for implementing security, but also the need for characterising and specifying the security properties of services and the need for analysing the security properties of service-based systems in meeting the systems' security requirements.

Following the development of component software, Web services and Grid services in recent years, there has been much effort in making security techniques standard constructs/libraries in various programming and service composition languages. They include Java security [5], Web Services Security [1] and other related standards such as WS-Trust, WS-SecureConversation and Security Assertion Markup Language (SAML). These efforts essentially provide implementation support for security. The issues of security property description and compositional analysis remain unaddressed.

Only in the past few years have we seen calls for moving security issues in services and systems to the next level, i.e., investigating the issues from a software engineering perspective [8]. It includes the characterisation of service/component security properties and the compositional security analysis for service/component-based systems [9, 10] as well as security certification [4]. Much more needs to be done to realise the goals of security-aware service-oriented computing with open dynamic service composition.

## 4 Conclusions

In this paper, we have introduced an approach to services security engineering. It includes security characterisation and description for services, and compositional security analysis for service-based systems. The approach and related techniques are set within a framework of service security certification. Our approach to security characterisation is partially based on the international security evaluation standard, the Common Criteria. The compositional analysis techniques allow us to check the security compatibility between interacting services and to verify whether or not the security requirements for a system are met.

*Acknowledgment.* I would like to thank Khaled Khan for his collaboration on security engineering for component software.

## References

1. B. Atkinson et al. Web services security (WS-Security). Working Group Report <http://www-106.ibm.com/developerworks/webservices/library/ws-secure/>, IBM, April 2002.
2. Common Criteria Project/ISO. *Common Criteria for Information Technology Security Evaluation, version 2.1 (ISO/IEC International Standard 15408)*. NIST, USA and ISO, Switzerland, <http://csrc.nist.gov/cc/>, December 1999.
3. A. Ghosh, C. Howell, and J.A. Whittaker. Building software securely from the ground up. *IEEE Software*, 19(1):14–16, 2002.
4. A. Ghosh and G. McGraw. An approach for certifying security in software components. In *Proc. 21st National Information Systems Security Conference*, 1998.
5. L. Gong, G. Ellison, and M. Dageforde. *Inside Java 2 Platform Security*. Addison-Wesley, Reading, MA, USA, 2003.
6. J. Han. A comprehensive interface definition framework for software components. In *Proc. 1998 Asia-Pacific Software Engineering Conference*, pages 110–117, 1998.
7. J. Hopkins. Component primer. *Communications of the ACM*, 43(10):27–30, 2000.
8. IEEE. Special issue on building software securely. *IEEE Software*, 19(1), 2002.
9. K. Khan and J. Han. Security aware software composition. *IEEE Software*, 19(1):34–41, 2002.
10. K. Khan and J. Han. A security characterisation framework for trustworthy component based software systems. In *Proc. 27th Annual International Computer Software and Applications Conference*, pages 164–169, 2003.
11. P. Sewell and J. Vitek. Secure composition of insecure components. In *Proc. 12th IEEE Computer Security Foundations Workshop*, pages 136–150, 1999.
12. J. Voas. The challenges of using COTS software in component-based development. *IEEE Computer*, pages 44–45, 1998.
13. J. Voas. Certifying software for high-assurance environments. *IEEE Software*, (4):48–54, 1999.