AOSecBuilder: An Aspect-Oriented Security Component Development Toolkit

CHUNLEI WANG, MINHUAN HUANG
Beijing Institute of System Engineering
P.O.B 9702-19, 100101 Beijing
CHINA

Abstract: - Tools support plays an important role in developing aspect-oriented software security components by providing reusable security aspects library and making the creation and extension of security aspect components convenient and efficient. In this paper, we investigate the general aspect-oriented software security development process, and propose an Eclipse-based software security aspect development tool, AOSecBuilder, to support security aspect component development. This tool provides reusability of software security aspects, and improves security developer’s productivity. We also discuss the future efforts of AOSecBuilder.

Key-Words: - AOSD, AspectJ, AJDT, Security Development Process, Security Tool

1 Introduction
Software security usually has a “cross-cutting” nature that a solution to single security problem might involve modifications across the whole application. Aspect-Oriented Programming [1] has been proposed as a promising approach for modularizing cross-cutting concerns in software. As mentioned in [2], we have proposed a prototype implementation of a Java security aspects library called JSAL, which provides general and reusable security relevant aspects for constructing secure applications.

In our research project, we have proposed Aspect-Oriented Software Security Development Process (AOSSDP), and an associated development tool, AOSecBuilder, which allows software security developers to develop software security components based on AspectJ and JSAL. This tool is built upon Eclipse [3], the most popular open source software development platform, allowing developers to take full advantage of the platform functions. AOSecBuilder also includes the ability to provide reusable software security components (e.g. encryption, authentication, etc) based on JSAL.

The remainder of this paper is organized as follows: Section 2 provides the background on previous related works in aspect-oriented software security development. Section 3 describes AOSSDP, Section 4 describes AOSecBuilder and its core functions, proposes an example application in which security functions have been developed in its environment. In section 5, we summarize our work, and discuss future efforts relevant to our research.

2 Related Works
AspectJ [4] is a popular aspect-oriented programming language, and the Eclipse AspectJ Development Tools (AJDT) project [5] provides tools for editing, building, and debugging AspectJ programs on the Eclipse platform. As a popular aspect-oriented programming tool, AJDT plays an important role not only in developing aspect-oriented programs, but also in assisting newcomers in learning the concepts of aspect-orientation. JSAL is a simple, flexible and extensible security aspect library implemented in Java and AspectJ. In JSAL, abstract security aspect components are provided with abstract pointcuts to realize the basic mechanism for reusing aspects. Furthermore, popular security packages in Java, upon which JSAL has been built (e.g. JCE [6] and JAAS [7]), assure the generality to a certain extent. Through JSAL, software security aspects are provided as the reusable security components in the development process.

3 Aspect-Oriented Software Security Development Process
With the rapidly increased complexities of software, software system’s security suffers more threats. In general, system’s security is affected by many kinds of factors, such as structure designs, system configurations and software developments, etc. In this paper, we mainly concern the security challenges in software development domain.

As we all know, software’s vulnerabilities are the main source of security holes, so we must
consider the design and implementation issues of software’s security components in the development lifecycle when constructing secure system. And, we often need to enhance software development methods in accordance with security goals. However, there is short of software security development methods and associated tools in current software engineering practices, and widely used development tools cannot effectively support analysis, design and implementation of software system’s security today.

In fact, one important reason of the constant appearance of security vulnerabilities is security may occur in any possible places of the program. People cannot simply isolate and concentrate on these issues in limited code segments (or modules) to solve security problems. Therefore, security functions can be viewed as system’s crosscutting concerns. The scattering and intermixing of software security functions is one of the causes of poor implementation of software security. Because of the extremely complexities of security, implementing consistent security policies in entire system is very bothering, error-prone, and expensive. Also because of limited views and constrained resources, the security design scheme usually only solves current concerns, and that causes the system’s security scheme difficult to evolve, understand and trace, and finally results in low quality of the whole system.

Separation of aspects or concerns, as the basic principle of software engineering, is introduced to control the continuously increased complexities of application programs. In brief, this principle suggests isolating different interests of problems, dealing with each of them separately, and finally synthesizing all these interests into one result. At present, the technologies applied to deal with crosscutting concerns are mainly divided into two types:

- **Interception based methods**: Certain events running in the program are intercepted and crosscutting concerns are extraordinarily processed before or after these events. In application containers such as Enterprise Java Beans (EJB) [8] and .NET framework [9], some limited, well separated security characteristics are provided.
- **Weave based methods**: When compiling, loading or even running codes that implement crosscutting concerns, the weavers are responsible for weaving these separated concerns with other code segments, such as AOP technology.

During our research, we investigated AOP technology and it’s applications in processing software security concerns, and analyzed aspect component model of AspectJ. As some of our research results, we proposed AOSSDP, designed and implemented Eclipse-based AOSecBuilder toolkit in accordance with the process to support security aspect development and improve software security reusability.

Through AOSSDP, the development of application’s security functions can be largely simplified in contrast with traditional software development methods. In requirement definition and analysis phase, system analysts need to identify the security requirements that will be applied with aspect-oriented technology. In general, non-functional security requirements are described with security policy methods.

After defining security policies and finishing software designs, non-functional security requirements can be mapped to abstract security mechanisms. Then, based on this system’s implementation technologies and runtime environment, system security developers need to determine concrete security methods, such as security communication protocols for a certain data channel. Since conveniently replacing concrete security methods is allowed by AOSSDP, the security decisions in the analysis phase is not final, and you can modify concrete security methods in accordance with feedbacks during implementation or even deployment phase.

During development phase, application developers apply aspect-oriented technology to write software security aspect codes based on defined security policies and concrete security methods. With the support of JSAL and AOSecBuilder, developers can determine security-related pointcuts through instantiating security aspect components’ abstract pointcuts. By reusing high quality security codes provided by JSAL, security developers can finish their programming tasks quickly and efficiently.

Although application’s normal business logic can be isolated from the security functions, its structure affects the implementation of security functions remarkably. In general, the designs and implementations of business logic should follow common rules, such as modularization, naming consistency, etc. By following these rules, application’s security aspects can be defined more easily.

Aspect-oriented software security development process is shown in figure 1. The dashed frames represent tasks that can be finished through reusable security aspect components. With the support of AOSSDP, business logic and security functions can be isolated and developed in parallel, which usually are processed in serial or mixed manners in traditional frameworks.
During requirement definition phase, security issues are proposed as non-functional requirements. These requirements are embodied as security policies. In AOSSDP, security policies are implemented through one or several aspects combinations. These aspects invoke concrete security methods to implement security functions, and define pointcuts for describing the relationships with business logic. By this means, the defects of “security function scattering” in traditional software development processes are largely avoided.

4 AOSecBuilder Descriptions and Sample Application

In our project, we proposed aspect-oriented software security development process, and implemented Java software security aspect component library prototype based on AspectJ. To facilitate software security components development, we also designed and implemented AOSecBuilder, an aspect-oriented software security component integrated development environment based on the open source Eclipse development tool platform. Eclipse’s underline runtime environment implementation is based on OSGi specification [10], and its all functions are provided as plugin (OSGi Bundle) forms. Therefore, we provided two plugin types in Eclipse platform to provide software security aspect component development functions:

- Core plugin – encapsulates JSAL, and integrates AspectJ and AJDT, to provide JSAL’s all security aspect component functions in the integrated development environment, and supports security aspect components’ build/weave functions.
- User interface plugin – provides software security aspect library view, software security aspect editor and software security aspect creation wizard based on software security aspect components provided by core plugin.

In the guidance of AOSSDP, software security designers can design needed security function components based on determined security policies. In AOSecBuilder, software security developers can implement application’s security function components based on JSAL. AOSecBuilder provides JSAL view for navigating and invoking security aspect components provided in JSAL, as shown in figure 2.

For the convenience of creating security aspect components, AOSecBuilder provides security aspect creation wizard, as shown in figure 3. These components can be inherited from JSAL or implemented alone.

AOSecBuilder toolkit also integrates AJDT, providing security aspect components’ compiling and weaving functions. After designing/developing application’s business logic and security functions respectively, AOSecBuilder’s project can be compiled and weaved to generate Java bytecodes and production applications eventually.

To demonstrate how to use AOSecBuilder for security components development, we created an Eclipse Java project called SECFTPD, which re-implemented security enhanced version of jFTPd that had done in [2]. In this project, we implemented four aspects based upon AOSecBuilder’s JSAL to deal with crosscutting security behavior. Among these aspects, two (FTPSession, FTPConnectionSecurity) contain the security logic of...
the FTPConnection class, which includes keeping session state and enforcing the access control policy, and the other two (FTPHandlerSecurity, FTPHandlerSecurityState) have similar responsibilities with regard to the FTPHandler class. Through AOSecBuilder security component development tool, the goal of “separation of security concern” is achieved, while the modularity of security implementations is enhanced and security components’ reusability is improved.

![Fig 3: software security aspect creation wizard](image)

5 Conclusion
AOSSDP provides a general aspect-oriented software security process model that simplifies application’s security function developments in contrast with traditional software development methods. With the support of AOSSDP, business logic and security functions can be isolated and developed in parallel, and the defect of “security function scattering” in traditional software development processes is largely avoided and software security aspect development can be progressed in a normative manner.

AOSecBuilder provides a rich set of features which aid in the development of aspect-oriented security components built on AspectJ and JSAL. These features include JSAL View, Security Aspect Creation Wizard, and Security Aspect Editor, etc. AOSecBuilder also integrates with AJDT to provide aspect debugging, compiling, and weaving functions.

Planned extensions to AOSecBuilder will further extend security aspect components in JSAL, enhance the tool’s support for security aspect design patterns [11], and improve the tool’s high usability both to novice and to experienced aspect-oriented security programmers.

References: