Building Secure Application Software: Methods, Tools, and Practical Experiences

● Yuko Nakayama

(Manuscript received December 6, 2006)

To build secure software, including Web applications, we must reconsider conventional software development lifecycles and establish new methods. However, because there are so many issues to be resolved, most software developers will probably be reluctant to perform such tasks. To improve software security, we must establish more efficient methods and tools to encourage these people. To share our best practices, this paper describes our efforts toward reforming the development processes in Fujitsu and introduces some methods and tools we have developed. As the first step, our activities have especially focused on well-known, basic Web application vulnerabilities. We have established an interview method to transfer experts' knowledge and skills to system engineers (SEs) as well as to check basic Web application security. We have also been building a management tool called Security Inspection Assistance Tool (SIAT). SIAT facilitates communication among people, reduces their workload, and facilitates skill transfer from security experts to SEs.

1. Introduction

In this age of the Internet, many Web applications have been developed, and they are of growing importance in everyday life. Among them are confidential applications such as Internet banking, e-commerce, and e-governance applications. However, a large number of vulnerable Web applications still exist on the Internet,[1] and some of them have caused serious economic losses and some have even led to crimes.[2]

To reduce these risks, it is not enough just to add security functions (security software and appliances) to vulnerable software because many of the vulnerabilities are due to inappropriate specifications or poor design from a security standpoint. Instead, we must reconsider conventional software development life cycles to enable security quality control from the early development phase.

However, there are many challenges to accomplishing such changes and many things to learn about software security. For example, the Web site "Build Security In"[3] provides various information about software security. Also, Microsoft has released books and established Web sites such as "Writing Secure Code, second edition"[4] and "Security Developer Center"[5] detailing their experiences and best practices. Because there are so many issues to be resolved, most software developers will probably be reluctant to make the first step toward improving software security. To encourage these people, we need to establish more efficient methods and tools for building secure software.

To share our best practices, this paper describes our efforts toward reforming development processes in Fujitsu and introduces our methods and tools. We have especially focused on well-known, basic Web application vulnerabilities as the first step because, compared with other generic software vulnerabilities, it is easier to show concrete countermeasures and formulate a

development process and methods based on them. We aim at building secure Web applications and also familiarizing as many project leaders and developers as possible with the basic knowledge of Web application security.

2. Lessons learned

Fujitsu Laboratories' application security team (of which the author is a member) has been tackling security vulnerabilities of software and systems since 2000. We have found vulnerabilities and helped developers fix them, supported activities to reform development processes, and developed some methods and tools for building secure software. This section describes some of the lessons learned from our experience.

Firstly, processes and methods must be tailored to each organization. For example, in Fujitsu, the Software Unit and Solution Business Group (which includes multiple Solution Business Units) adopted different processes. The Software Unit develops various kinds of middleware and has to address very diverse threats. This has led to a focus on threat analysis and an organization in which the security architects play a key role. The security architects, who are deployed at each section, accumulate knowledge about product-specific problems and help other people develop secure software. On the other hand, the Solution Business Group, which provides system integration services, takes different approaches. This paper mainly describes the efforts in the Government & Public Solutions Unit of the Solution Business Group and their best practices. This unit has improved its application development process targeted at Web application security.

Secondly, Web application security is an aspect of software quality and business risk. The Government & Public Solutions Unit therefore positions Web application security as a software quality issue and controls it as part of the unit's comprehensive project management system. This unit has been steadily improving its software development process and quality assurance methods.

Thirdly, to provide effective and efficient aid for a large number of system integration projects, it is important to define a clear division of roles among people. As a best practice, the Government & Public Solutions Unit divides its roles into the Software Engineering Process Group (SEPG), system engineers (SEs), and security experts (**Figure 1**). These roles are described below together with some issues associated with them.

1) Software Engineering Process Group (SEPG)

As well as improving the overall software development process, SEPG manages all activities related to building secure Web applications. SEPG defines the process and guidelines based on opinions of security experts, modifies them to suit the Government & Public Solutions Unit's organization, and then promotes them. In spite of SEPG's important role, its efforts as secretariats tend to be inconspicuous and unappreciated by others. Getting its work to the attention of people, especially managers, is one of the key factors to success. Also, to sustain a reformation over the years, it is essential to lighten people's workload and keep them motivated. Section 4.1 introduces our management tool for making the activities of this group more efficient and noticeable.

2) System engineers (SEs)

SEs are responsible for building systems, including Web applications. Ideally, every SE should be highly skilled in the area of security,

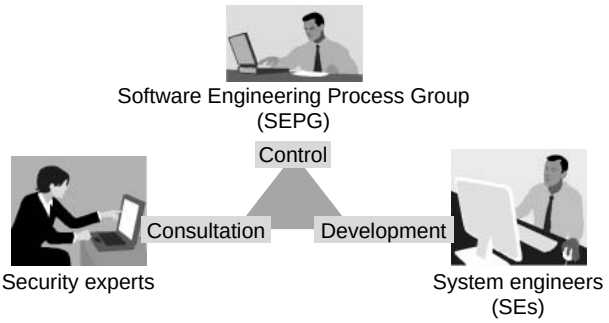

Figure 1
Three roles in secure application development.

but it is unrealistic to expect them to have specialized knowledge about an entire system. At the same time, because they are project leaders who manage developers, it is crucial for SEs to understand the essence of security vulnerabilities and countermeasures. Therefore, the knowledge of security experts should be transferred to SEs in an effective way. It is especially important for security experts to avoid using security jargon when they communicate with SEs. Section 3.1 introduces our interview method for facilitating communication between security experts and SEs.

3) Security experts

Security experts are expected to help in each project with their specialized skills and knowledge. However, only a few security experts can devote much of their time to in-house assistance, and it is quite hard to produce experts in a short time. It is therefore necessary to reduce their workload so they can consult and audit for as many projects as possible. First of all, methods and tools are needed to promote the transfer of their knowledge to SEs. This issue is discussed in Sections 3 and 4. In the second place, security experts should only present SEs with countermeasures. SEs or their managers should choose an appropriate countermeasure for their project because, in many cases, the choice is a business decision related to costs and the delivery period. In addition, to prevent security experts from having too much authority, they should not double as SEPGs. Full-time SEPG members should be appointed independently so they can bridge differences of opinion between security experts and SEs to avoid the risk of emotional confrontations.

Lastly, methods and tools are needed to provide effective and efficient aid for a large number of projects. As the first step, we formulated the methods described in Section 3 for the Government & Public Solutions Unit. As the second step, based on these methods, we have been building the tools described in Section 4.

3. Methods

In light of the lessons learned from past projects, the Government & Public Solutions Unit established a new development process for building secure Web applications in the spring of 2005. This process includes some security checking activities (**Figure 2**). The following is a summary of these activities.

1) Interviews for SEs

Face-to-face consultations between SEs and security experts. The details are given in Section 3.1.

2) White box test

Source code review with analysis tools (white box testing tools). However, the existing methods, which rely only on these testing tools, have their limits. We are now trying to establish a more effective method, which is briefly mentioned in Section 4.2.

3) Penetration test

Security test that focuses exclusively on Web application level vulnerabilities and is mainly done using a Web application scanner (black-box testing tool).

4) Wording check

Check on the wording of descriptions and procedures in Web pages and manuals. The details are given in Section 3.2.

5) Infrastructure test

Security test for middleware and networks.

Among these activities, the interviews for SEs and the wording check are characteristic of this process. This section describes these two methods in detail.

3.1 Interviews for SEs

Because SEs are project leaders who manage a large number of developers, it is crucial for them to understand security vulnerabilities related to their applications and disseminate accurate countermeasures to their developers. However, they are too busy to learn specialized knowledge about security. On the other hand, there are too few security experts to expect them to give every

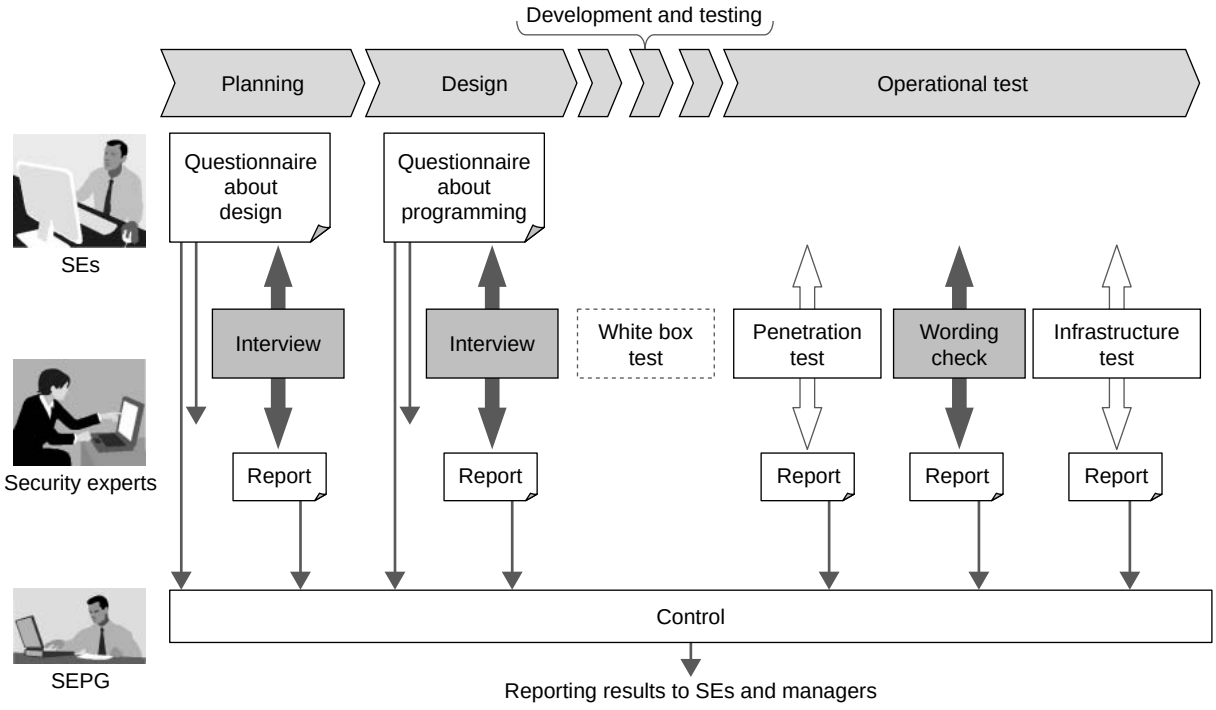

Figure 2
Outline of security checking process for Web applications.

possible help to every project. To break this deadlock, we have built an interview method with the following characteristics:

1) Security baseline for Web applications

To cover a large number of projects, we decided to focus on Web application security as the initial step. In order to deal with a large number of projects uniformly, we first made a list of common vulnerabilities and their countermeasure as a baseline of Web application security. Next, we translated them into plain words in the form of questionnaires. The questionnaires also ask whether a project has any components beyond the baseline such as digitally signed mobile code (signed Java applets or ActiveX controls) or components installed on user PCs. Because these components need advanced support by security experts, we are trying to minimize their number and find them in an early phase of development.

2) Questionnaires without security jargon

Security experts tend to explain things using security jargon. However, this jargon makes it difficult for SEs to grasp the root of a problem and the essence of its countermeasure. A question such as, "Have you adapted proper countermeasures to cross-site scripting (XSS)?" is a good example of jargon that SEs must make an extra effort to understand. SEs are likely to misunderstand this type of jargon and take improper countermeasures as a consequence. Questionnaires must be written in simple, plain, and concrete words. Such questions should therefore be worded, for example, as, "Have you replaced < with <, > with >,...?" or, "Have you avoided using the following JavaScript functions or methods: eval(), document.write(),...?" Our questionnaires consist of about 60 to 80 of these kinds of questions for each development phase.

3) Repeated interviews in early phase of development

If a vulnerability is found in the later phases of development, for example, at the penetration test in the operational test phase, it will be necessary to return to the programming, design, or even

planning phase to fix it, which often will be very costly to do. To avoid this situation, we conduct two interviews in the early phase of development: one just before the design phase and another just before the development (programming) phase. We repeatedly ask similar questions in these two interviews. The main purpose of the first interview is to stimulate the SEs' minds to think about security. In the second interview, we try to ensure that the SEs develop a deeper understanding for concrete countermeasures related to specific programming languages, frameworks, and middleware. We hope that, compared to self-education and study in a classroom, these interviews provide a better way for SEs to learn the skills of security experts.

3.2 Wording check

As is often the case, the wording of descriptions and procedures is an oft-forgotten aspect of security. Proper wording is as important as secure programming. For example, if a manual instructed users to change the security settings for Web browsers to a lower level without careful consideration, users would take these settings too lightly and consequently jeopardize the security of their PCs. To prevent this sort of problem, in the operational test phase, security experts inspect descriptions and procedures in the Web pages and manuals by using the system. Especially, they carefully check the details of descriptions and procedures that mention settings, dialogs, digital certificates, and anything else related to security.

Currently, security experts have a key role in this wording check. However, we will facilitate a gradual skill transfer from security experts to SEs by adapting the management tool described in Section 4.1.

4. Tools

To make our process and methods more effective and efficient, we have developed some tools for building secure Web applications. This section describes our management tool and then briefly mentions other tools.

4.1 Management tool

To apply the methods described above to many more projects, we particularly need to improve the efficiency of the SEPG and security experts' tasks related to interviews. It is also necessary to transfer security experts' knowledge and skills to SEs. We have been developing a management tool called Security Inspection Assistance Tool (SIAT) to accomplish these objectives. SIAT facilitates communication among SEPG, security experts, SEs, and managers. It also helps security experts accumulate the basic knowledge and skills of Web application security and helps SEs acquire them (**Figure 3**). The major functions of SIAT and a typical workflow using SIAT are described below.

1) Arrangements (SEPG)

SEPG makes arrangements for interviews and other security checks using SIAT. Because SEPG deals with many projects, SIAT helps it to consolidate the management and lighten its workload. SEPG also tracks and monitors projects using SIAT to confirm whether all the projects undergo security checks and take countermeasures against the problems that security experts have pointed out.

2) Answering the questionnaires (SEs)

SEs fill out questionnaires on SIAT prior to interviews. Because most of the SEs work in the customers' office or in satellite offices nearby the customers, it is difficult for them to have frequent face-to-face communication with security experts. To save precious time in interviews, SEs can find common questions and answers in the knowledge-base (Wiki) in SIAT. They can also use SIAT to submit inquiries to security experts about unclear points in questionnaires.

3) Consulting (security experts)

As mentioned above, there are only a few security experts compared with the number of projects. It is therefore also important for security experts to do their work online and do it more

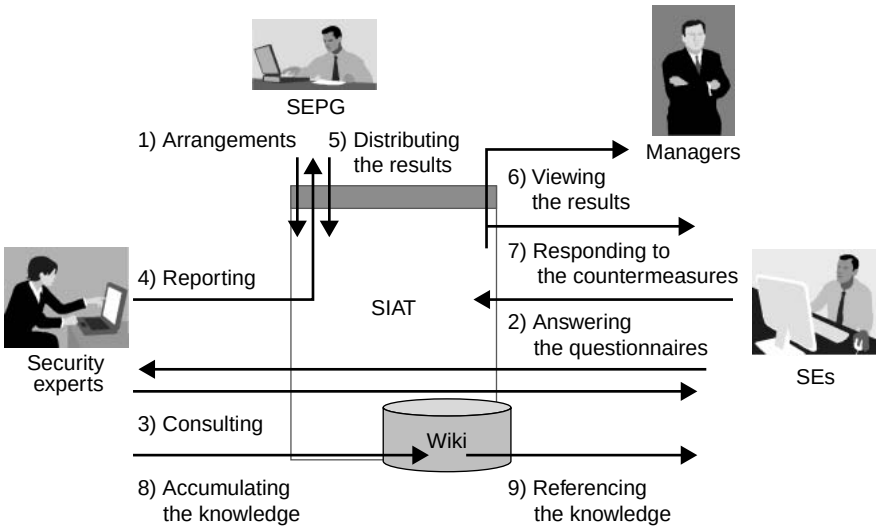

Figure 3
Overview of Security Inspection Assistance Tool (SIAT).

efficiently. Security experts can answer SEs' basic questions through SIAT, freeing up precious time in face-to-face interviews.

4) Reporting (security experts)

Security experts report their results not to SEs directly but to SEPG through SIAT. Then, SEPG checks whether their reports are easy for SEs to understand before releasing them.

5) Distributing the results (SEPG)

SEPG makes a thorough review of the reports written by security experts and then distributes them only to the necessary people (SEs and managers) through SIAT. Because the reports include confidential information about vulnerabilities in customers' systems, SEPG strictly controls access to them using SIAT.

6) Viewing the results (SEs and managers)

Only the SEs and managers who are authorized by SEPG can obtain the results of security checks on SIAT. They can also acquire background knowledge stored in the wiki knowledgebase of SIAT.

7) Responding to the countermeasures (SEs)

SEs must decide how to address the problems pointed out by security experts and report their countermeasures to SEPG through SIAT. SEPG monitors the SEs' responses on SIAT and reminds them to report their countermeasures if necessary.

8) Accumulating the knowledge (security experts)

To reduce the workload of security experts, it is necessary to transfer their skills to SEs in incremental steps. Security experts can accumulate the basic knowledge and typical questions and answers about Web application security in wiki. They can also enhance them to follow newly discovered vulnerabilities.

9) Referencing the knowledge (SEs)

SEs can refer to the wiki whenever they need background knowledge about questionnaires. Each item of questionnaires is linked to a Web page on the wiki, so SEs can easily browse the information.

4.2 Other tools

We have developed some Web application security testing tools for both white box and black box testing.[6),7)] We have also evaluated some commercial and free testing tools. These activities have shown us that in most cases, existing testing tools produce many false-positive results

and require considerable efforts from security experts to sort them from actual vulnerabilities. The existing methods, which rely only on these testing tools, have their limits. To improve the efficiency of testing, we have proposed a testing method that uses coding conventions in conjunction with white box testing tools.[8),9)] We will apply this method to real projects in the future.

5. Conclusion

This paper described our efforts to improve development processes in Fujitsu and introduced methods and tools for building secure Web applications. We are now applying these technologies to system integration projects and refining them. We have proposed a testing method that uses coding conventions in conjunction with white box testing tools and will enhance it in the future. We will also develop a threat analysis method to efficiently address the needs for IT security evaluations and certifications such as the ISO/IEC 15408 Information Technology Security Evaluation and Standard.[10)]

References

1) Symantec: Symantec Internet Security Threat Report Trends for January 06 — June 06. Volume X, September 2006. *http://www.symantec.com/specprog/threatreport/ent-whitepaper_symantec_internet_security_threat_report_x_09_2006.en-us.pdf*
2) Web Application Security Consortium: The Web Hacking Incidents Database. *http://www.webappsec.org/projects/whid/*
3) Department of Homeland Security (DHS), National Cyber Security Division (NCSD): Build Security In. *https://buildsecurityin.us-cert.gov/daisy/bsi/home.html*
4) M. Howard and D. LeBlanc: Writing Secure Code. second edition, Redmond, Washington, Microsoft Press, 2003.
5) Microsoft: Security Developer Center. *http://msdn.microsoft.com/security/*
6) H. Kojima, Y. Nakayama, S. Kawashima, and R. Fujikawa: An audit-assisting tool for writing secure Java code. (in Japanese), IPSJ SIG Technical Reports, 2002-CSEC-020, 2003, 18, p.83-88.
7) Y. Yamaoka, I. Morikawa, H. Kojima, and Y. Nakayama: Proposal of a method for Web application black box test considering reproducibility of Web pages. (in Japanese), Computer Security Symposium 2004 (CSS2004), Volume II of II, IPSJ Symposium Series, 2004, 11, p.703-708.
8) T. Okubo, Y. Nakayama, Y. Wataguchi, and H. Tanaka: A Study on Software Development Method which Fulfills Specified Security Requirements. (in Japanese), Computer Security Symposium 2006 (CSS2006), IPSJ Symposium Series, 2006, 11, p.387-392.
9) T. Okubo and H. Tanaka: Secure Software Development Method with Coding Conventions and Frameworks. The First International Workshop on Secure Software Engineering (SecSE 2007), IEEE Computer Society. (2007) (To be published).
10) INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN (IPA): IT Security Evaluation and Certification. *http://www.ipa.go.jp/english/security/second.html*

Yuko Nakayama, *Fujitsu Laboratories Ltd.*

Ms. Nakayama received the B.S. degree in Electrical and Computer Engineering from Yokohama National University, Yokohama, Japan in 1989. She joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1989, where she has been engaged in research and development of software engineering and security. She is a member of the Information Processing Society of Japan (IPSJ), the Association for Computing Machinery (ACM), and the Computer Society of the Institute of Electrical and Electronics Engineers (IEEE).

E-mail: nakayama.yuko@jp.fujitsu.com