Security Testing and Compliance for Online Banking in Real-World

Hao Chen and Jean-Pierre Corriveau

***Abstract*—There is a continuously growing number of customers who use online banking because of its convenience. We consider two current online banking problems. First, we observe a lack of attention and research focusing on security issues relevant to the clients' side of online banking systems. Second, there are many security products used in online banking systems. However, security testing is still in its infancy and little is available to verify if those security products are working properly. We discuss the current security testing categories and standards, as well as common security testing approaches. We then propose an original scheme to design a compliance testing system for the security of online banking. Our proposal aims at suggesting to testers how to design security testing and identify potential vulnerabilities in current online banking systems.**

***Index Terms*—security testing, online banking, security standards, compliance**

I. INTRODUCTION

Internet-based electronic banking is also called online banking. There are a continuously growing number of customers using online banking because of its convenience. Banks also encourage their customers to use online banking since it can lower banks' costs. However, online systems providing banking services need to offer strong security because of the confidential information involved, as well as attacks against online banking authentication mechanisms [8].

There's no 'peace of mind' for online banking according to Mannan [12]: most banks mislead their customers about the security of online banking and make customers believe that banks will offer refunds if hackers steal money from accounts. For example, most banks' customer agreements require customers to install and maintain an up-to-date version of anti-virus, firewall and anti-spyware software. These security requirements must be satisfied if banks are to completely reimburse customers for losses resulting from unauthorized transactions through online banking. However, some customers are not aware or may not know how to satisfy such conditions. A survey of 123 advanced security-aware users [12] shows that most failed to satisfy common security requirements and would not be eligible to get a refund if hackers raided their accounts.

We find that there are two potential security problems in the current online banking systems. First, online banking systems constitute a kind of client/server application. Most research on online banking systems is focusing on security on the server's side and on network security (i.e., the creation of a secure channel between the clients' computers and the bank's servers). Solutions to ensure authenticity and confidentiality over Internet are widely available. However, little exists to address security on the clients' side. Second, there are many security technologies, implementations and products that can be selected and applied to online banking systems. However, it is difficult to find a testing system that can be used by the customers themselves to verify if those security products are running properly. Clearly, there is no real security protection without proper installation and configuration of such products.

Therefore it is very important to perform security testing on the client side in accordance with the security standards or policies of banks. To the best of our knowledge, there is currently no product for testing online banking systems on the client's side. In this paper, we sketch out how testers may perform online banking security testing.

Security technologies involve algorithms, protocols, standards, and mechanisms. Furthermore, it is commonly agreed that there are four levels of security: system, network, data and application [6]. Moreover, many security technologies can be used for each level. Consequently, once risks are identified and appropriate corresponding security technologies selected, the latter must be tested to demonstrate they meet the security demands of the relevant level(s). More precisely, testing results and metrics must provide this demonstration.

Our goal here is to summarize criteria and ideas that may be useful for the creation of a compliance testing approach for the security of online banking systems. Such an approach should be designed according to the security standards and policies of banks. And it should allow users to establish whether they satisfy the security demands of their bank before they start using online banking.

In section 2, we categorize security testing into black-box and white-box issues. Section 3 discusses security standards relevant to compliance testing. Section 4 summarizes common security testing approaches and technologies. Section 5 proposes how to build a compliance testing system using such security testing approaches. Related work is discussed in section 6.

Manuscript received December 30, 2008.

H. Chen is a Ph.D. candidate with the School of Computer Science, Carleton University, Ottawa, Ontario, CANADA K1S 5B6 (email: hchen9@scs.carleton.ca).

J.-P. Corriveau is an Associate Professor and Graduate Director with the School of Computer Science, Carleton University, Ottawa, Ontario, CANADA K1S 5B6 (phone: 1-613-520-2600 ext. 8751; fax: ext. 4344; email: jeanpier@scs.carleton.ca).

II. Security Testing Categories

Software security can be viewed as the absence of characteristics that pose a risk to the operator of the software or third parties if such characteristics are exploited with malicious intent [23]. Roughly put, security testing is about discovering risk in what is not specified. This definition focuses on 'exploits' rather than security functions. For example, buffer overflow errors may appear anywhere in a system and their impact usually does not depend on security functions.

Security testing involves two types of testing: functional and non-functional. The former consists in testing security features or components, such as an access control system, in order to ensure the system works. It can be performed using a traditional testing approach. For example, testing authentication with user name and password constitutes functional security testing. We are able to ensure the system meets security functional requirements after testing because we know the expected results and behavior of the system. Non-functional security testing is also called risk-based security testing. Its purpose is to test for a malicious attack, which may require probing undocumented vulnerabilities such as SQL insertion or a poor password. It is difficult for traditional software testers to perform this kind of testing because they may not think as an attacker aiming to exploit the system.

Orthogonal to the functional/non-functional dichotomy, security testing can be performed using both white box and black box testing to reveal possible software risks and potential exploits. White box testing takes into account the internal mechanism of a system or component [26]. Conversely, black box testing ignores the internal mechanism of a system or component and focuses on the outputs generated in response to selected inputs and conditions [26]. That is, black box testing runs software without considering the source code. Consequently, software testers need to try various malicious inputs in order to break a system. Some security testing tools do treat software applications as black boxes. For example, there are some black-box vulnerability scanners such as Nikto [17] and Nessus [21], which use large repositories of known software flaws to discover security problems by launching attacks against an application.

Compliance testing, also called conformance testing, aims to verify whether a product or system meets the supplied specification. Regardless of the domain or specification, conformance testing is a form of black-box testing [27]. That is, conformance tests are derived solely from the product or system requirements or specifications.

III. Security Standards

Security compliance may include analysis and testing of the system for conformance to a set of security standards. It is important that a security evaluation of IT products be performed using official standards that contribute to the objectivity of the results. Some of the most popular security standards include CC/CEM, ISO 17799, COBIT, NIST, Basel II, ISO 21188 and FISCAM [22]. Let us elaborate:

The Common Criteria for IT Security Evaluation (CC) [4] and the Common Methodology for IT Security Evaluation (CEM) [5] are used as the standard evaluation criteria and methodology for all security evaluations of IT products Evaluations are done based on the seven evaluation assurance levels (EALs) [22]. The Common Criteria is an international standard (ISO/IEC 15408) and has received worldwide acceptance. It is a very important framework for the evaluation of IT products and systems with respect to their security mechanisms.

The ISO 17799 standard can be used to develop a security policy within an organization. It addresses information security policy, access control, information systems maintenance and compliance. The Control Objectives for Information and Related Technology (COBIT) [28] is a framework and supporting toolset for IT management, created by the Information Systems Audit and Control Association (ISACA), and the IT Governance Institute (ITGI). It contains high-level controls for system security such as managed security, logical access control, security of online data, user account security and controls, data classification and firewall architectures [28]. COBIT allows managers to bridge the gap between control requirements, technical issues and business risks. The National Institute of Standards and Technology (NIST) provides a guideline on Network Security Testing for operating systems. Underlying Technical Models for Information Technology Security [18] provide security features and known security attacks on IT systems. For secure systems or products, the National Computer Security Centre (NCSC) has published the US Department of Defense "Trusted Computer System Evaluation Criteria" (referred to as the "Orange Book").

Basel II is a security standard of Internet banking. It describes a number of security controls including authentication of customers, non-repudiation of online transactions, authorization controls, data integrity of transactions, audit trails and confidentiality of bank information. The International Organization for Standardization's ISO 21188:2006 [10] is a framework of Public Key Infrastructure for Financial Services and describes requirements to enable certificate-based solutions for secure Internet banking applications. It also defines security targets and procedures used for the risk management process. The Federal Information System Controls Audit Manual (FISCAM) was designed to evaluate the general and application controls over financial systems. It is very useful for developing a security program or application.

The open question is to decide how these standards can be used as part of a systematic approach to compliance security testing for online banking.

IV. Security Testing Approaches and Technologies

There are some common testing approaches for security, which are based on testing targets such as: overall system or one component, server or client side, network or operating system. For example, monitoring tools focus more on network and file systems, while vulnerability scanners are to test web applications rather than underlying operating systems. Knowing what a method targets for testing must guide the selection of such methods.

Flaw hypothesis testing (also called 'penetration' testing when applied to security testing) is based on propagation of errors due to vulnerabilities in order to find possible ways to attack a system. The method was first introduced by Linde in [11], and the idea is to hypothesize possible flaws, and then test whether these hypotheses are true. Penetration testing is a black box testing and has been used in several tools built for network security testing, e.g. SAINT [1]. The quality of data collected from network traffic or vulnerability databases is of vital importance in order to do penetration testing effectively. Fuzzing [16] sends random inputs to an interface in order to find a bug by chance. Fuzzing is about how to look, rather than what to look for. Fault injection [24] is more targeted than fuzzing. By making use of fuzzing, we can identify flaws in software that we do not have full access to and perform a black-box testing. Fault injection can be used for testing ActiveX objects, file formats, command-line executables, and shared memory segments [29]. A number of useful tools can automate this process and identify various vulnerabilities. For example, OLEView is a tool included with Microsoft Visual Studio that lists installed ActiveX, COM, and OLE objects. It allows the user to view the properties and implemented interfaces, and can be used to check whether a given ActiveX object is scriptable and marked safe [29]. If there is a serious vulnerability in ActiveX components that might be activated by a remote Web page, an attacker may access and control the user's computer by taking advantage of the vulnerability.

Interactive proxies used for Web application security testing are better to understand test requirements and produce results than automation tools. WebScarab [3] is a tool which operates as an interactive proxy, allowing the user to review and modify requests created by the browser before sending to the server, and to review and modify responses returned from the server [3]. WebScarab is able to test secure web applications because it can intercept HTTPS besides HTTP communication. Both software testers and security specialists can use it to debug problems and identify vulnerabilities.

Vulnerability scanners are systematically used to test Web applications, but they have poor performance [15] because of their automation and because they generate too many false positives [25]. OWASP top ten [2] is useful to find common vulnerabilities and security problems required for testing. For example, cross site scripting (XSS), injection flaws, and malicious file execution are all listed in OWASP top ten and considered as most critical security flaws for web applications. The main purpose of using vulnerability scanners is to identify those specified flaws on the list by conducting security checks. In addition, OWASP top ten is often used as a minimum standard for web application vulnerability assessment and security tesing to find security defects.

An attack tree [20] focuses on possible attacks against a system. It provides a threat model and is very useful to perform security analysis and evaluation. However, a systematic method is needed to develop the tree. It is open to formalization [14].

STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege) [9] may be viewed as an aspect-oriented programming (AOP) approach. It considers a set of generic attack techniques and analyzes a system's ability to defend itself against such attacks. It may guide testers in focusing on important components.

There are other common approaches and techniques to security testing such as a) checklists, monitoring tools that are systematically used for network traffic and file access; b) honeypot, which is a systematic approach to collecting and interpreting evidence obtained after putting a system on the Internet and waiting for hackers to access; and c) some methods for observing and modeling the behavior of a system in order to determine if it is secure.

Two important aspects of compliance testing are standards and testing methods. We have described most common security standards in section 3. In this section, we referred to a number of testing approaches suitable for security compliance testing.

V. Compliance Testing for Online Banking Security

We propose to use security-testing technologies to design a compliance testing system in order to ensure online banking systems comply with security criteria and standards.

Compliance testing is used to determine whether a system meets some specific standard(s). There are three reasons why we propose compliance testing for online banking security. First, compliance testing will establish a desirable exchange of information between customers and service providers (e.g., banks). Compliance testing will likely increase customers' trust in products and services. When customers are more confident about the security of online banking, they use it more, which is desirable for both parties (with respect to convenience and cost). Also, service providers can substantiate their claims about the security of their systems without imposing (in fine print customer agreements) hard-to-test conditions on the systems of the users. For example, a security product or system may obtain corresponding security certification if it complies with specified standard after testing. Second, compliance testing is an objective method to evaluate products against a standard. International standards are recognized all over the world, so testing the online banking system based on these standards is objective. On the contrary, testing is subjective when it is with respect to internal security requirements created by a bank. Third, requirements in security are different from others for software systems. Most approaches to security testing treat the implementation under test (IUT) as a black box. The internal structure of IUT may not be accessible. Compliance testing also follows black box testing. Therefore, we may use security testing approaches to design and perform compliance testing because they rely on black box testing techniques. Also, generally a compliance test suite is a collection of combinations of legal and illegal inputs to the IUT, together with a corresponding collection of expected results or the corresponding outputs that can be compared with a reference implementation respecting the relevant standard.

We propose a general process for building a compliance testing system for security.

First, select the security standard or policies that the online banking system under test (SUT) must comply with. For example, Common Criteria or Basel II could be chosen. Most banks announce that their online banking applications meet or surpass the minimum industry standards because security has been the main concern of customers in the use of online banking. Therefore, we may select a bank with respect to the standards that banks use or mention in their security policy or agreements.

Second, create a set of detailed requirements or specifications to show how specific tasks meet the specific standard(s). For example, consider secure operating system requirements such as using up-to-date anti-virus, anti-spyware software and a firewall may be defined in detailed specifications. No one can guarantee that one computer is 100 percent secure, but installing an anti-virus and firewall can definitely lower the risk of disclosing confidential information or being hacked. In [19], the author shows how online banking is not as secure as it should be and how it is possible and not difficult to create a virus to break security and access bank accounts. Thus, the point to be grasped is that detailed requirements are the key facet of the compliance testing system.

Third, select corresponding security testing approaches or tools to develop tests according to these detailed security specifications. Different testing approaches focus on different areas (as previously mentioned). The area of the selected testing approaches should match the related area of detailed security requirements or specifications of the step two. Testing approaches may include vulnerability scanning, penetration testing, log review, integrity and configuration checkers. Some of these testing techniques are used together to gain a more comprehensive assessment of the overall security. For example, using a log review to detect the existence of required security applications in the operating systems of a customer's computer is not enough because such detection does not guarantee these applications are working properly. Therefore we need to check configurations and monitor the behavior of the applications in order to demonstrate compliance to the online banking requirements at the client side.

Fourth, verify and demonstrate that the compliance of overall system has been achieved. Vulnerability assessments, penetration testing and audit reviews of security controls ensure that policies and standards are correct. Implemented security of the overall system on the specific platform can be tested and compared with the requirements and configurations to ensure that the security compliances are verified as per the criteria and requirements.

It is hoped that a compliance testing method will be helpful in ensuring that security requirements are operational and satisfy the relevant standards or policies. In addition, such a method must be able to establish whether the selected security technologies are appropriate for the security level required. In essence, a compliance testing method may be viewed as a proactive rather than reactive approach to security, one that lowers risk and threats.

VI. Related Work

Security of Internet applications has become increasingly important. Research exists on security compliance testing for network applications, but focuses on testing application protocols rather than application systems such as online banking systems. Syntax-based testing is a black box, data driven testing technique for applications for which inputs can be described formally. SCL [13] is a language designed to express the syntax and context sensitive constraints of protocols and is a component of Protocol Tester [13] that uses syntax-based testing to evaluate the security of network applications. In addition, software transformation and program comprehension techniques [7] are used to assist in the security testing of network applications focusing on state sensitive protocols.

VII. Conclusions and Future Work

This paper describes current online banking problems and discusses the need for security testing for online banking systems on clients' side. We investigate the current categories, criteria, and approaches for security testing. We then sketch out how to design a compliance testing system for the security of online banking.

This paper does not attempt to develop a specific testing tool for online banking. We propose an approach for security testing based on the international security standards and policies, which may help developers and testers to acquire a good understanding of security testing and develop a general compliance testing system focusing on client side security. We believe the testing system will help customers increase their confidence when performing transactions through online banking and ensure that the online banking systems meet the relevant security requirements and polices in order to lower the risks of being attacked by hacker.

The compliance testing approach we proposed is still a very sketchy conceptual framework. Future work includes: 1) Addressing compliance testing of a web-based system executed on a remote server, using downloadable files for local execution or a combination of remote and local access and execution. 2) Self-test capabilities and/or formal certification testing. 3) Making a decision on the presence or absence of any risk on a customer's computer and on a bank's website. This entails providing customers with an understanding of the security policy of their bank. 4) In addition, usability issues need to be considered. This depends on what constitutes an acceptable security level and on the trade-off between usability and security.

References

[1] "SAINT", 2007. [Online]. Available: http://www.saintcorporation.com/

[2] "OWASP Top Ten Project", 2008. [Online]. Available: http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[3] "OWASP WebScarab Project", 2008. [Online]. Available: http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project

[4] CC, "Common Criteria for Information Technology Security Evaluation" Version 3.1, ISO 15408, 2006. [Online]. Available: http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R1.pdf

[5] CEM, "Common Criteria for Information Technology Security Evaluation", version3.1, 2006. [Online]. Available: http://www.commoncriteriaportal.org/files/ccfiles/CEMV3.1R2.pdf

[6] H. Chen and J.P. Corriveau, "Security Features and Technologies for E-business Architecture Design", *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS '07),* March 21 - 27, 2007, Hong Kong, pp. 1150-1156, Newswood Limited, 2007.

[7] T.R. Dean, G.S.N. Knight, "Applying Software Transformation Techniques to Security Testing", *International Workshop on Software Evolution and Transformation,* Delft, Netherlands*,* November 2004, pp. 49-52. [Online]. Available: http://post.queensu.ca/~trd/research/papers/step2005.pdf

[8] A. Hiltgen.; T. Kramp; T. Weigold; "Secure Internet-banking Authentication," *IEEE Security and Privacy*, vol. 4, no. 2, 2006, p. 21-29.

[9] M. Howard and D. C. Le Blanc, *Writing Secure Code*, Microsoft Press, 2002.

[10] International Organization for Standardization, "Public Key Infrastructure for Financial Services -Practices and Policy Framework," ISO 21188, 2006.

[11] R. Linde. "Operating System Penetration". *Proceedings of the National Computer Conference*, AFIPS Press, Montvale, NJ, 1975.

[12] M. Mannan, P.C. van Oorschot. "Security and Usability: The Gap in Real-World Online Banking". *New Security Paradigms Workshop 2007 (NSPW'07),* New Hampshire, USA, Sept.18-21 2007.

[13] S. Marquis, T. Dean, G.S.N. Knight, "SCL: A Language for Security Testing of Network Applications", *Proc. CASCON 2005*, Toronto, Oct. 2005. [Online]. Available: http://tarpit.rmc.ca/knight/papers/SCL.pdf

[14] S. Mauw.; M. Oostdijk, "Foundations of Attack Trees" in: Dongho Won and Seungjoo Kim, editors, *International Conference on Information Security and Cryptology (ICISC 2005),* LNCS 3935, pages 186-198, Seoul, Korea, December 2005. Springer-Verlag, Berlin.

[15] G. McGraw, *Software Security: Building Security In*, Addison-Wesley, 2006.

[16] B. Miller, et al., "Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services", Technical Report CS-TR-95-1268, University of Wisconsin, April 1995.

[17] Nikto. "Web Server Scanner", 2005. [Online]. Available: http://www.cirt.net/code/nikto.shtml

[18] National Institute of Standards and Technology, 2001. "Underlying Technical Models for Information Technology Security", 2001. *Special Publication 800-33*. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf

[19] F. Puente; J.D. Sandoval; P. Hernandez; C.J. Molina; "Improving online banking security with hardware devices", *39th Annual 2005 International Carnahan Conference on Security Technology (CCST '05),* 2005. 11-14 Oct. 2005 pp.174 - 177

[20] B. Schneier, "Attack Trees: Modeling security threats", *Dr. Dobb's Journal*, Dec. 1999.

[21] Tenable Network Security. "Nessus Open Source Vulnerability Scanner Project", 2005. [Online]. Available: http://www.nessus.org/

[22] H. F. Tipton, M Krause *Information Security Management Handbook*, *sixth edition*, volume 2, Auerbach Publiations, Taylor & Francis Group, 2008.

[23] S. Turpe, **"**Security Testing: Turning Practice into Theory"**.** *Software Testing Verification and Validation Workshop, 2008. ICSTW apos;08.* IEEE International Conference, 9-11 April 2008 pp. 294 - 302

[24] J.A. Whittacker.; H. H.Thompson, *How to Break Software Security*, Addison-Wesley Longman, Amsterdam, 2003.

[25] A. Wiegenstein,; F. Weidemann; M. Schumacher; S. Schinzel, "Web Application Vulnerability Scanners - a Benchmark", Version 1.0, Virtual Forge GmbH, 2006-10-04. [Online]. Available: http://www.virtualforge.de/whitepapers/web_scanner_benchmark.pdf

[26] IEEE, "IEEE Standard 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology", 1990.

[27] C. Hagwood, "Statistics of Software Conformance Testing". Statistical Engineering Division, National Institute of Standards and Technology (NIST), 2001. [Online]. Available: http://www.itl.nist.gov/div898/pubs/ar/ar1998/node10.html

[28] COBIT, "Control Objectives for Information and Related Technology", 2008 [Online]. Available: http://www.isaca.org/cobit

[29] C. Wysopal, L. Nelson, D. D. Zovi, and E. Dustin, "Testing Fault Injection in Local Applications", in *The Art of Software Security Testing: Identifying Software Security Flaws,* Addison-Wesley Professional, part of the Symantec Press series, 2007.