WEB APPLICATION SECURITY INSTRUCTIONAL PARADIGMS AND THE IS CURRICULUM

J. Packy Laverty, Robert Morris University, laverty@rmu.edu
John J. Scarpino, Robert Morris University, scarpino@rmu.edu

ABSTRACT

This document provides an overview of the growing importance of web application security threats and its role in the IS security curriculum. Two alternative instructional paradigms designed to present web application security were reviewed. Secure Programming curricula have been used to present detailed coverage from a software coding perspective. However, the Secure Programming Paradigm may present challenges in the choice of programming language or the required level of programming prerequisites that may not be appropriate for an Information Systems curriculum.

As an alternative, the Automated Web Application Testing Paradigm using IBM's AppScan web security testing tool presents web application security from a quality assurance and testing perspective that may be integrated within the Software Development Life Cycle (SDLC). Recommendations for the integration of web application security in context of an Information Systems curriculum will be discussed.

Keywords: Application Security, Web Application Security Testing, Automated Application Testing Tools, IBM AppScan, Secure Programming, IS Model Curriculum.

INTRODUCTION

Information assurance and security relies on three fundamental IT infrastructure components: network services, host operating system, and targeted applications [1]. The security threats, vulnerabilities and risks of these three fundamental IT infrastructure components has been recognized at different level of coverage in IS, IT and CS Model Curricula. The IS 2002 Model Curriculum and Guidelines for Undergraduate Degree programs in Information Systems recommended the coverage of information assurance and security in three core courses: IS Fundamentals (IS 2002.1), Electronic Business Strategy (IS 2002.2) and Network and Data Communication (I2 2002.6) [2]. Application Security was not specifically mentioned in the IS 2002 Body of Knowledge. The ACM/AIS Undergraduate Curriculum Revision Task Force's IS 2009 Draft Document proposed an IT Security and Risk Management and an IT Audit and Control elective courses [3]. The SIGITE Curriculum Committee's IT2005 model curriculum recommended 23 core hours in Information Assurance and Security Theory and twenty (20) core hours in Network Security, but only four (5) core hours in application and web security [4].

Manson el al. (2003) [5] and Crowley (2003) [6] compared academic (ISECON 2002 and IEEE/ACM Task Force Model Curriculum) to government and industry application security common bodies of knowledge, e.g., NIST 800-16, NSTISSI 4011. While both academic and government/industry standards shared common domains in network security, cryptography, security policy and secure computing systems, industry standards did recognize the importance of application security more than academic standards. The importance of application and web security was partially confirmed by interviews of recent graduates and IT professional conducted by Bogolea and Wijekumar (2004) [7].

Dornseif et al. (2005) proposed that while there appears to be a general consensus by universities on information assurance and security, this consensus tends to either be too theoretical (like cryptographic protocols or formal modeling) or may emphasize defensive security topics (like access control, techniques, firewalls, and VPN. An Applied IT Security course was proposed [8]. While Attack/Defend [9] and "Capture the Flag" [10] security curriculum design models have been used in Information Assurance and Security Curricula, new developments like security threats in Web-based systems, e.g., cross-site scripting, and botnets are often ignored [8].

Several selected textbooks about Java and C++ were reviewed by Wood and Skovira (2007). Their conclusion was that application security concepts were not adequately discussed in the reviewed

textbooks. Their recommendation was to add more application security concepts to the C++ and Java program language textbooks and to emphasize testing of application code [45]. This lack of secure programming coverage may be due to the complexity of programs [46].

While network and operating system security have been addressed in IS and IT model curriculums, previous research may indicate that the inclusion of application security curricula may have been too limited. This limitation may be more obvious when compared to web application security threats. Web application security presents both different and unique challenges to identity management and access to user information [11].

The importance of web application security and the instructional vehicle to deliver that body of knowledge needs to be studied. Supplementing existing network or operating system security curricula with web application security content may not be adequate considering the growing frequency of attacks and the increasing responsibilities of regulatory compliance. The lessons learned from the design, development and delivery of two graduate courses: Secure Programming and Automated Web Application Security Testing, will be compared as alternative instructional paradigms to present web application security in the Information Systems curriculum.

IMPORTANCE OF WEB APPLICATION SECURITY

The passage of the Sarbanes-Oxley Act (SOX) of 2002 [12], the Health Insurance and Portability Act (HIPPA) of 1996 [13], and the Financial Services Modernization Act of 1999 [14, 15, 16] require proactive compliance for a wide-range of security objectives designed to protect user identity and access to user information. Best practices and security strategies have changed not only in scope, but also have changed in the type of security vulnerabilities and attacks. To be more specific, the impact of web application security attacks has grown. In the 2002 CSI/FBI Computer Crime and Security Survey, viruses and laptop theft incidents lead the top of the list. There were no FBI Security Incident Categories that directly reflected web application security in the 2002 Survey [17]. Since 2004 the CSI/FBI Computer Crime and Security Survey started measuring web application security attacks and reported an increasing trend. The results of the 2008 CSI/FBI study reported that 11% of 517 respondents experienced a successful web application security attack and "concerns around the security of Web applications (and the secure design of applications in general) also appeared with more-than-average frequency" [18]. A FTC Survey reported that in 2005 there were 8.3 million victims of Identity Theft [19].

A headline news article in USA Today on March, 17 2009 indicated that SQL injection attacks on Web sites have reached an estimated 450,000 per day. What's more, IBM Internet Security Systems found almost 50% more infected Web pages in the last three months of 2008 than it did in all of 2007 [20].

Organizations may not have adapted to these changes in security vulnerabilities. Danny Allan, Director of Watchfire an industry leading tool for security testing, reported that 90% of the security budget was being spent on network security, while only 10% was spent on web application security. Yet 75% of all attacks were directed towards web applications [21]. Cross-Site Scripting (XSS) and SQL Injection are #1 and #2 reported vulnerabilities [22]. Firewalls or intrusion detection systems do not protect web applications from attacks like Cross-Site Scripting, SQL Injections and Buffer Overflow [23].

The increased consumer demand for new web application technologies has been matched with increased web application technology content in the Information Systems curriculum, e.g., .NET, PHP, Java Servlets and JSP. While web technologies, computer network and host operating security topics have been integrated into IS curriculum web application security content may be lacking.

Security topics, such as Access Control Lists, (embedded in firewalls, routers and switches), cryptographic tools (SSL, private and public keys, hashes and digital certificates, etc.), Authentication, Intrusion Detection Systems, Virus Protection, etc., are topics frequently included in the Information Systems Curriculum. However, web applications require more than network and operating system secure "best practices" [21].

Authenticating to a web application is different than authenticating to an operating system. An HTML interface is used to enter authentication information rather than using an operating system prompt. User accounts and passwords are stored in an application database. Information transmitted between the client and server uses the HTTP protocol. Since HTTP is a stateless protocol, session tokens or cookies are used to store authentication state after logon. HTTP cookies and sessions present unique vulnerabilities to web applications not normally encountered in other security contexts.

Basic network security practice attempts to either prevent unauthorized access to a network resource or attempts to intercept or change the content of a network message. However, web applications are opened for Internet access. Closing access to a HTTP port is not a viable option. Encrypting the contents of a network message, e.g., SSL, provides no protection if an attacker steals the user account, password or session using cross-site scripting or session hijacking attacks, or forces the web application to disclose sensitive information using injection attacks [24].

There are many web application threats. Cross-site Scripting (XSS) injects malicious code into a target web server or redirects web users to a malicious web server by exploiting insecure JavaScript or VBScript Code [24, 25, 26]. Buffer overflow attacks are not only a threat to web applications, but also may affect the web server itself 29, 30, 31, 32, 48]. The HTML form is a vehicle to exploit application injection attacks [33, 48]. Injection of SQL query statements as input data into an HTML form exploits a variety of server-side programming languages, e.g., Java, ASP, PHP, etc [26]. SQL Injection attacks attempt to retrieve or alter information stored in the application's database. Session Hijacking attempts to bypass authentication by forging, guessing or stealing session_ids or session cookies, e.g., session fixation or insufficient session expiration [1, 22]. Path Transversal attacks attempt to insert code into the response URL, e.g., "..\", in an attempt to access directories outside the web server's root or to alter the application to behave in a different way [34].

SECURE PROGRAMMING PARADIGM

Secured Programming (Secured Coding) has been used to present web application security concepts within the context of Secured Programming Principles [35]. At a minimum all program code must be able to withstand attacks [36, 45, 46]. But, is it necessary for students to actually code secured programs to understand and appreciate web application security? Presenting web application security from secured programming approach does require a minimal level of program language knowledge. Given the diversity of application programming languages and prerequisite competencies, it may be difficult to implement a Secure Programming Paradigm in an Information Systems Curriculum.

Security patterns used to prevent web application threats vary by programming language. For example, the program language syntax and input validation patterns used to prevent cross site scripting [37, 38] varies between Java Servlets and .NET code. The classes coded to implement Java Cryptographic Services (JCA) [1] are different than .Net Cryptography classes [39]. Likewise, the classes used to implement Java Authentication and Authorization Services (JAAS) are significantly different that .NET when protecting applications using user-based or code-based security [1, 39]. There is no requirement that Secure Programming be limited to web applications.

In the Accreditation Board for Engineering and Technology (ABET) 2008-2009 Criteria for Accrediting Computing Programs, the Information Systems Accreditation Standards recommend a one-year, or two programming language courses [40]. The content of these programming language courses are not prescribed. There may be a question whether two programming language courses could provide the prerequisite competencies to either include or prepare for Secured Programming concepts. Furthermore, the Secure Programming Paradigm may over emphasis syntax and pattern, rather than the role of web application security within a broader context - the system development life cycle (SDLC).

AUTOMATED WEB APPLICATION SECURITY TESTING PARADIGM

The Automated Web Application Testing Paradigm presents web application security from a quality assurance and testing perspective. The Automated Web Application Testing Paradigm uses a penetration testing approach. Less emphasis is placed on programming syntax and pattern. More emphasis is placed on understanding web application security within the context of the system development life cycle (SDLC). Web Automated tools will test against a web application running on a web server and runs through various permutations to find its vulnerabilities using a client-based testing tool. Potter [41] states that today's *security testing tools* do not simply scan the surface an application and regurgitate a list of defects, rather, they delve deep into each individual application on the premise that the usability of these applications affect the health of the system as a whole. In addition to application testing, the security settings of the web server and operating file system which stores web content are also tested for vulnerabilities.

Web Application Security Testing tools may be used in a variety of roles, e.g., Content Experts, Business Process Analysts, Developers, System Acceptance, Compliance Testing or Auditing. A Tool-Based

approach may be used by individuals with limited or no program development experience. The integration of business, project management, software testing and quality assurance roles throughout the System Development Life Cycle may be important in a IS curriculum [42].

Auronen (2002) reviewed the features of several commercial and open source Web Application Testing tools. The review provided a framework to compare and evaluate the Web Application Testing tools to meet the needs of an engagement [47]. The current version of several Web Application Testing tools were reviewed and IBM AppScan was selected. The reasons for selecting IBM AppScan were: a) comprehensive testing of an entire web site, b) academic support, c) use in industry, d) ease of use, and e) professional reporting features, which supported different audience backgrounds.

IBM provides instructional material and software downloads for a variety of applications, including AppScan [43]. AppScan is found under Rational in the IBM Course Repository and Software Downloads. In addition to instructional support materials, IBM's Student Portal provides students with access to several Unified Process games and exercises, resume postings, and student project opportunities [44].

The IBM Academic Initiative requires that an educational institution or academic department be certified as a sponsoring institution. This institution registration process is integrated with a faculty membership application for an IBM universal ID and password. Additional information concerning the courses that a faculty member teaches is required. A program agreement must be completed for each institution. The process is simple, straight-forward and is conducted online and via email.

Students downloaded and installed AppScan on their personal computers. The installation process was straight forward and required one hour to install. The current 7.7 version of AppScan is based on the Eclipse IDE framework. AppScan can be used as a standalone application or may be integrated with other Rational Unified Process applications, e.g., Requirement Analysis (IBM RequisitePro), Quality Manager, Software Development (WebSphere), etc. Instructional materials and software downloads are available for all of these RUP products.

Anyone can download IBM's AppScan for a 30-day free trial evaluation. Registered faculty or students may download an Academic Edition version of AppScan which has a longer license period. However, both versions are limited to scanning a test site sponsored by IBM. Altoro Mutual (www.testfire.net), a hypothetical online bank is used to demonstrate the effectiveness of AppScan. (See Exhibit 1.) Penetration testing of other web sites is not permitted. However, the results of the security scan will vary as new features are added to the test site.

After conducting a test, AppScan provides a visual summary of the results organized in four panes: URL-based, Dashboard, Severity Summary and Details. The Results Window is well-organized and easy-to-use. (See Exhibit 2.)

The URL-based pane displays application objects in a Window-based folder and file format. If you right-click on an application a web browser will display the HTML code that was tested. The Details pane provides: a) a security issue description, b) an advisory, c) fix recommendations, and d) HTTP request/response code. The actual server application code is not provided. Students are not required to be programmers.

AppScan provides a detailed printed Web Application Report and it is stored in PDF format. The report is organized into six major sections: Executive Summary, Detailed Security Issues, Remediation Tasks, Application Data, Application URLs, Advisories & Fix Recommendations. To understand the content of these reports, students must have some perquisite knowledge in HTTP and web application security threats.

CONCLUSIONS AND RECOMMENDATIONS

Web application security threats will continue to grow as the eCommerce expands and organizations find new ways to interact with customers. Application security threats such as viruses, Trojan horses, malware, spyware, privacy attacks and identity theft have become more pervasive and sophisticated. Government and Industry standards are increasingly recognizing the importance of application security. Since IS model curriculums do not directly address application security threats beyond a conceptual level, coverage of application security may not be adequate.

Secured Programming and Automated Web Security Testing are alternative instructional paradigms that may be used to include application security in the IS curriculum as an elective course. However, each paradigm emphasizes different aspects of application security. While the Secured Programming Paradigm may emphasize the "Best Practices" to develop web-safe applications, the Web Security Testing Paradigm uses a white-box testing approach to test the web application.

As compared to a Secured Programming Paradigm, the Automated Web Security Testing Paradigm may offer the following instructional advantages for a IS Model Curriculum: 1) minimal student or instructor web application development expertise, 2) emphasis of business and systems analysis needs rather than program syntax and pattern, and 3) re-enforcement on software testing practices within the system development life cycle. By its nature, Automated Web Security tools are based on business requirements and those requirements may be shared by other project management roles beyond the application programmer. As organizations expand their commitment to software quality assurance, using a testing paradigm to teach web application threats will enable students to understand that software quality is not just for programmers. While the results of Automated Web Application test tools may provide content valuable to web application programmers, there are no requirements to be a programmer to use these tools. The Automated Web Application Security Paradigm is a better fit for the Information Systems Curriculum than the Secured Programming paradigm alternative.

Automated Web Application Security tools may be integrated into an existing course or offered as a new course. Some knowledge of web application security threats, vulnerabilities and risks is required for students to understand and interpret the information provided by the user interface or printed security reports. The level of instructional coverage of web application threats may be varied for the needs of the course. Students may be encouraged to develop their own test plan before actually using the tool. After a test scan has been conducted, instructors can assign different directories or applications to students or groups and require them to prepare a management security report and analysis. The management security report may also include recommendations for remediation and development of a retest plan to emphasize the iterative nature of application testing.

The Automated Web Application Security Paradigm may provide an innovative way to integrate application security, business requirements, application and systems testing within the software development life cycle. To gain further insight in web application security within the IS Curriculum, the following aspects should be studied further:

- What should be the role of web application security as compared to other security infrastructure components, i.e., network and operating system, in the IS curriculum?
- What would be best instructional method to present web application security concepts in the IS curriculum, i.e., Secure Programming or Web Application Testing?
- Which Web Application Testing Tool would be most appropriate in the IS curriculum?

References

1. Steel, C., Nagappan, R., Lai. R., (2006). Core Security Parameters: Best Practices and Strategies for J2EE, Web Services and Identity Management. Prentice Hall.

2. Gorgone, T., Davis, G., Valacich, J., Topi, H., Feinstein, D., Longenecker, H. (2002). IS 2002: Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems. Association for Computing Machinery (ACM), Association for Information Systems (AIS), Association of Information Technology Professionals (AITP). Retrieved 3/20/2009 from web site: http://192.245.222.212:8009/IS2002Doc/IS%202002%2012-31-2002.pdf

3. Valacich, J., Topi, H. (2008). Joint ACM/AIS Undergraduate Curriculum Revision Task Force IS 2009. Retrieved 3/20/2009 from web site: http://blogsandwikis.bentley.edu/iscurriculum/index.php/Main_Page

4. SIGITE Curriculum Committee, Computing Curriculum 2005, IT Volume (IT2005). (2005). Retrieved 3/24/2009 from web site: http://www.acm.org/education/curric_vols/IT_October_2005.pdf

5. Manson, D., Curl, S. (2003). A Comparison of Academic and Government and Information Standards. *Information Systems Education Journal*, 1(39).

6. Crowley, E. (2003). Information System Security Curricula Development. Proceedings of the 4th Conference on Information Technology Curriculum.

7. Bogolea, B., Wijekumar, K. (2004). Information Security Curriculum Creation: A Case Study. Proceedings of the 1st

Annual Conference on Information Security Curriculum Development.

8. Dornseif, M., Freiling, C., Mink, M., Pimenidis, L.(2005). Teaching data security at university degree level. Proceedings of the Fourth World Conference on Information Security Education.

9. Yurcik, W., Doss, D. (2001). Different Approaches in the Teaching of Information Systems Security. Retrieved 3/24/2009 from web site: http://www.sosresearch.org/publications/ISECON.2001.Yurik.pdf

10. Vigna. G. (2003). Teaching Network Security through Live Exercises. World Conference on Information Security Proceedings.

11. Achieving HIPAA Compliance With Identity Management from Sun. Retrieved 3/2/2009 from web site: http://www.sun.com/software/products/identity/wp_hipaa_identity_mgmt.pdf

12. Sarbanes-Oxley Act. H.R. 3763, July 30, 2002. Retrieved 2/15/2009 from web site: http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=107_cong_bills&docid=f:h3763enr.tst.pdf

13. Executive Summary of HIPAA Provisions. Retrieved 2/23/2009 from web site: http://www.thedocuteam.com/docs/hipaaprovisions.doc

14. Federal Trade Commission: Gramm-Leach-Bliley Act - Disclosure of Nonpublic Personal Information. Retrieved 2/23/2009 from web site: http://www.ftc.gov/privacy/glbact/glbsub1.htm

15. Federal Trade Commission: Gramm-Leach-Bliley Act - Fraudulent Access to Financial Information. Retrieved 2/23/2009 from web site: http://www.ftc.gov/privacy/glbact/glbsub2.htm

16. U.S. Senate Committee on Banking, Housing, and Urban Affairs. Gramm-Leach-Bliley Act of 1999 - Summary of Provisions. Retrieved 2/24/2009 from web site: http://banking.senate.gov/conf/grmleach.htm

17. FTC Releases Survey of Identity Theft in the U.S. Study Shows 8.3 Million Victims in 2005. Retrieved 2/13/2009 from web site: http://www.ftc.gov/opa/2007/11/idtheft.shtm

18. 2002 CSI/FBI Computer Crime and Security Survey. Computer Security Institute. Retrieved 2/13/2009 from web site: wwww.gocsi.com

19. FTC Releases Survey of Identity Theft in the U.S. Study Shows 8.3 Million Victims in 2005. Retrieved 2/13/2009 from web site: http://www.ftc.gov/opa/2007/11/idtheft.shtm

20. Website-infecting SQL injection attacks hit 450,000 a Day. Retrieved 3/24/2009 from web site: http://www.usatoday.com/tech/news/2009-03-16-sql-attacks-cyber-security_N.htm

21. Yeo, V. Hackers ride on Web app vulnerabilities. Retrieved 3/24/2009 from web site: http://www.zdnetasia.com/news/security/0,39044215,61969925,00.htm

22. Web Threat Classification Taxonomy. Retrieved 3/14/2009 from web site: http://cwe.mitre.org/documents/sources/WASCThreatClassificationTaxonomyGraphic.pdf

23. Web Application Security Attacks. Retrieved 3/10/2009 from web site: http://www.cenzic.com/resources/reg-required/videos/Gartner-on-Web-Application-Security/

24. Cross-Site Scripting Security Exposure Executive Summary. Retrieved 3/14/2009 from web site: http://technet.microsoft.com/en-us/library/cc750326.aspx

25. Rafail, J. Cross-Site Scripting Vulnerabilities. Retrieved 3/10/2009 from web site: www.cert.org/archive/pdf/cross_site_scripting.pdf

26. Gregg, M. (2006). Certified Ethical Hacker. Que Publishing.

27. Introduction: Buffer Overflow Vulnerabilities. Retrieved 3/10/2009 from web site: http://www.linuxsecurity.com/content/view/118881/49/

28. NIST National Vulnerability Database. Retrieved 3/10/2009 from web site: http://web.nvd.nist.gov

29. Apache 1.3.37 htpasswd buffer overflow vulnerability. Retrieved 3/10/2009 from web site: http://seclists.org/fulldisclosure/2007/Jan/0044.html

30. Apache httpd 1.3 vulnerabilities. Retrieved 3/10/2009 from web site: http://httpd.apache.org/security/vulnerabilities_13.html

31. Microsoft Internet Information Services Remote Buffer Overflow. Retrieved 3/10/2009 from web site: http://research.eeye.com/html/advisories/published/AD20010618.html

32. Buffer Overflow in Core Microsoft Windows DLL. Retrieved 3/10/2009 from web site: http://www.cert.org/advisories/CA-2003-09.html\

33. McAlearney, A. Automated SQL Injection: What Your Enterprise Needs to Know. Retrieved 3/14/2009 from web site: http://searchsecurity.techtarget.com/news/article/0,289142,sid14_gci996075,00.html#

34. Patch Transversal Attack. Retrieved 3/14/2009 from web site: http://www.spamfighter.com/News-6260-Path-Transversal-Attack.htm

35. OWASP Secure Coding Principles. Retrieved 3/14/2009 from web site: "http://www.owasp.org/index.php/Secure_Coding_Principles

36. Definition of Secure Code. Retrieved 3/15/2009 from web site: http://www.pcmag.com/encyclopedia_term/0,2542,t=secure+code&i=58654,00.asp

37. Mitigating Cross-site Scripting with HTTP-only Cookies. Retrieved 3/15/2009 from web site: http://msdn.microsoft.com/en-us/library/ms533046.aspx

38. Prevent Cross-Site (XSS) Malicious Content. Retrieved 3/15/2009 from web site: http://tldp.org/HOWTO/Secure-Programs-HOWTO/cross-site-malicious-content.html

39. Thorssteinson, P., Ganesh, G. (2004). Net Security and Cryptography. Pearson Education.

40. Criteria for Accrediting Computer Programs. Retrieved 3/15/2009 from web site: http://www.abet.org/Linked%20Documents-UPDATE/Criteria%20and%20PP/C001%2008-09%20CAC%20Criteria%2011-8-07.pdf

41. Software Security Testing. Retrieved 3/15/2009 from web site: http://www.cigital.com/papers/download/bsi4-testing.pdf

42. Kohun, F., Wood, D., Laverty, J. (2007). Systems Oriented Architecture, Unified Process life Cycle, and IS Model Curriculum Compatibility: Meeting Industry Needs. *Information Systems Education Journal*, 5(1).

43. IBM Academic Initiative. Retrieved 3/10/2009 from web site: http://www-304.ibm.com/jct01005c/university/scholars/academicinitiative/

44. IBM student Portal. Retrieved 3/10/2009 from web site: http://www-304.ibm.com/jct01005c/university/students/index1.html

45. Wood, D., Skovria, R. (*2007*). Secure Programming Concepts in Selected C++ and Java Textbooks. *Issues in Information Systems, 8(1).*

46. Linder, F. (2006). Software Security is Software Reliability. *Communication of the ACM 49(7), 128*

47. Auronen, L. (2002). Tool-Based Approach to Assessing Web Application Security. Retrieved 6/15/2009 from web site: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.893&rep=rep1&type=pdf

48. Howard, M., LeBlanc, D., Viega, J. (2005). 19 Deadly Sins of Software Security: Programming Flaws and How to Fix Them. McGraw-Hill/Osborne.

Exhibit 1 - IBM AppScan (AltoroMutual Test Site)

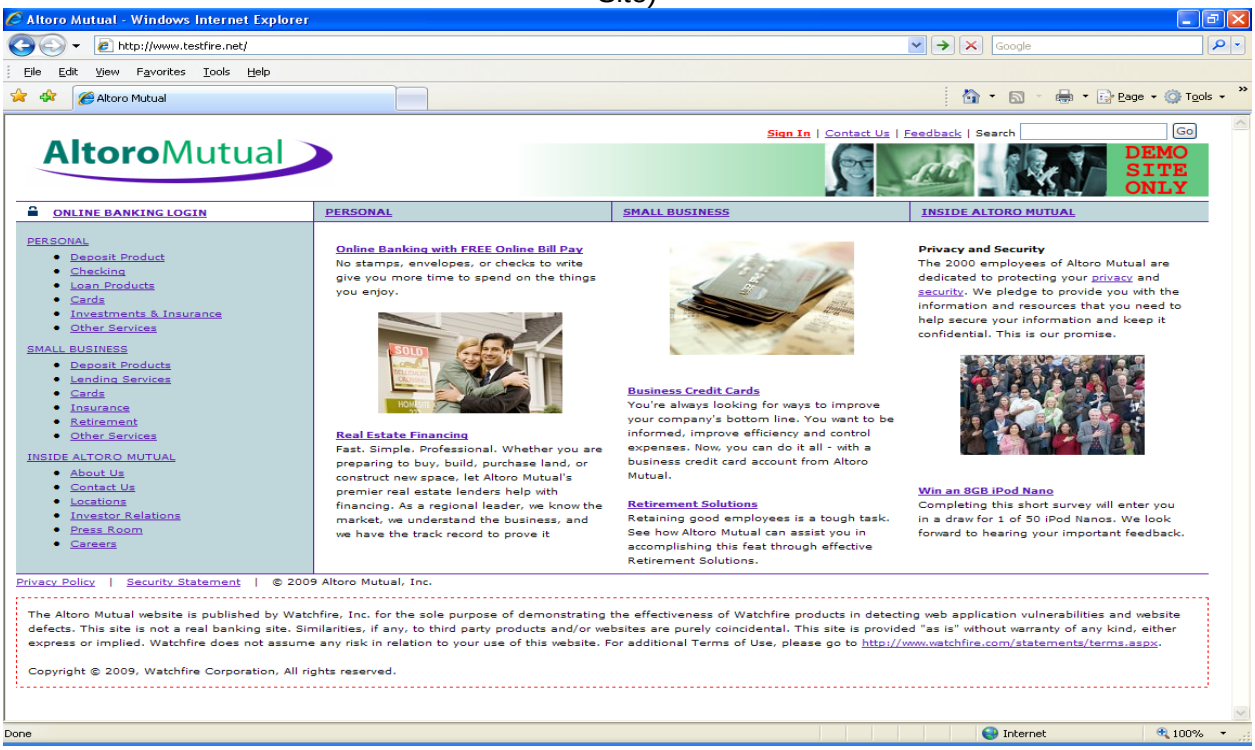

Exhibit 2 IBM AppScan Test Summary Results