REALIZING WEB APPLICATION VULNERABILITY ANALYSIS VIA AVDL

Ha-Thanh Le and Peter Kok Keong Loh
School of Computer Engineering, Nanyang Technological University, Block N4
Nanyang Avenue, 639798, Singapore

Keywords: Web application vulnerability, vulnerability description, AVDL, vulnerability analysis.

Abstract: Several vulnerability analysis techniques in web-based applications detect and report on different types of vulnerabilities. However, no single technique provides a generic technology-independent handling of web-based vulnerabilities. In this paper we present our experience with and experimental exemplification of using the Application Vulnerability Description Language (AVDL) to realize a unified data model for technology-independent vulnerability analysis of web applications. We also introduce an overview of a new web vulnerability analysis framework. This work is part of a project that is funded by the Centre for Strategic Infocomm Technologies, Ministry of Defence Singapore.

1 INTRODUCTION

The rapid rise of corporate web applications offers abundant opportunities for e-businesses to flourish. However, this also raises many security issues and exacerbates the demand for practical customer-friendly solutions (Raina 2004; Grossman 2007; IBM 2007). Most existing approaches are technology-specific and concentrate on solutions based on the application layer and program code (Static analyzers e.g., Pixy (Jovanovic, Kruegel et al. 2006), WebSSARI (Huang, Yu et al. 2004) and PHP's intrablock and intra/inter procedural 3-tier architecture (Xie and Aiken 2006) scan Web application source code for vulnerabilities, whereas dynamic tools such as WAVES (Huang, Huang et al. 2003) and PHP's taintedness tracking (Nguyen-Tuong, Guarnieri et al. 2005) try to detect attack while executing applications).

In this paper, we investigate the effectiveness of using the Application Vulnerability Description Language (AVDL) (OASIS 2007) to develop a unified data model used in a technology-independent, rule-base solution for vulnerability analysis of web-based applications.

We review other works that relate to web-based vulnerability analysis in section 2. In Section 3, we present an overview of our approach. In section 4, we present vulnerability analysis case studies with two web scanners and AVDL as a vulnerability description format. The paper concludes with Section 5.

2 RELATED WORKS

2.1 Web-based Vulnerability Taxonomies

For web applications, which are usually developed using high-level declarative languages, the vulnerabilities occur frequently in places where the script-based algorithms interact with other systems or components, such as databases, file systems, operating systems, or the network (Dowd, McDonald et al. 2006; Stamp 2006). In other words, systems can be compromised via web technologies, e.g. exploitation via a web script may start a security breach. Many web vulnerability classes have also been detected, classified and documented via technology-specific scanners (Nguyen-Tuong, Guarnieri et al. 2005; Dowd, McDonald et al. 2006; Siddharth and Doshi 2006; Cova, Felmetsger et al. 2007; Grossman 2007; IBM 2007). A model of vulnerability taxonomy such as (Bishop 1999), (Bazaz and Arthur 2007) and (Berghe, Riordan et al. 2005) has been proposed, on which new analytical methodologies may be designed and implemented. Others only concentrate research on potential classes

Table 1: Vulnerability Detection models and Analysis techniques.

Detection model	Negative approach	Match the models of known vulnerabilities against web-based applications to identify instances of the modeled vulnerabilities
	Positive approach	Use model of expected behavior of application to analyze the application behavior to identify abnormality caused by a security violation
Analysis technique	Static techniques	Provide set of pre-execution techniques for predicting dynamic properties of the analyzed program
	Dynamic techniques	Use a series of checks to detect vulnerabilities and prevent attacks at run-time

of vulnerability: cross-site scripting (XSS) (Jovanovic, Kruegel et al. 2006), SQL Injection (Halfond, Viegas et al. 2006; Kals, Kirda et al. 2006; Nguyen-Tuong, Guarnieri et al. 2005). Much research work also focused on PHP or the web scripting language (JSP, ASP).

2.2 Web-based Vulnerability Analysis

Several tools and techniques have been developed to analyze vulnerabilities in web-based applications. Cova et. al. (Cova, Felmetsger et al. 2007) classify vulnerability analysis of web-based applications according to detection models and analysis techniques (Table 1).

More recent research proposed vulnerability detection methods using static analysis schemes (Huang, Yu et al. 2004; Livshits and Lam 2005; Minamide 2005; Jovanovic, Kruegel et al. 2006; Xie and Aiken 2006), Ghosh et. al. based on fault injection (Ghosh, O'Connor et al. 1998) while Halfond et. al. (Halfond, Orso et al. 2006) and Nguyen-Tuong et. al. (Nguyen-Tuong, Guarnieri et al. 2005) use tainting technique. Another positive and dynamic methodology such as penetration testing (Hurst 2007) forces anomalous program states during application execution to observe application behavior and infer the forensics of potential vulnerabilities.

Several techniques are implemented in web application scanners (Kals, Kirda et al. 2006; Fong and Okun 2007). Scanner output can help to assess the security of the web application. However, no single scanner provides a technology-independent coverage of possible vulnerabilities. An experiment conducted by Suto (Suto 2007) on three commercial scanners revealed that NTOSpider scans and covers most of the vulnerabilities while other tools AppScan (Watchfire) missed 88% while WebInspect (SPIDynamics) missed 95% of the legitimate vulnerabilities found by NTOSpider.

Several automatic tools are developed: Nguyen-Tuong et. al. (Nguyen-Tuong, Guarnieri et al. 2005) using precise tainting over information flow. On the other hand, Woo et. al. (Woo, Alhazmi et al. 2006) proposed a methodology that must work with web browsers. In [29], the authors extended the VDM model to AML vulnerability model for web applications. Sets of vulnerability discovery data were examined and fitted to a vulnerability discovery model that will be used for projection of both current and future vulnerabilities.

2.3 Web Vulnerability Description

Although there are many vulnerability analysis and detection tools for web-based applications, none of them provides a complete methodology to find more vulnerabilities (Cova, Felmetsger et al. 2007). Testers and QA team must rely on a combination of tools and must understand different vulnerability description formats consequently. However, tools usually possess overlapping functionalities, raising costs, lowering performance, incurring data surplus and overheads in vulnerability analysis and detection. While there are existing vulnerability databases such as (SecurityFocus 2007), Bugtraq (SecurityFocus 2007) and CVE (CVE 2007) focus on the description of known vulnerabilities, J. Steffan and M. Schumacher (Steffan and Schumacher 2002) suggest a graph-based collaborative attack modeling, which provides meaningful knowledge sharing method with more detail vulnerability description.

OASIS recommended using web Application Vulnerability Description Language - AVDL (OASIS 2004) "a standard XML format that allows entities (such as applications, organizations, or institutes) to communicate information regarding web application vulnerabilities". The other two are Web Application Security standard – WAS and VulnXML (OASIS 2007). Among these, AVDL is expected to become the most common descriptive

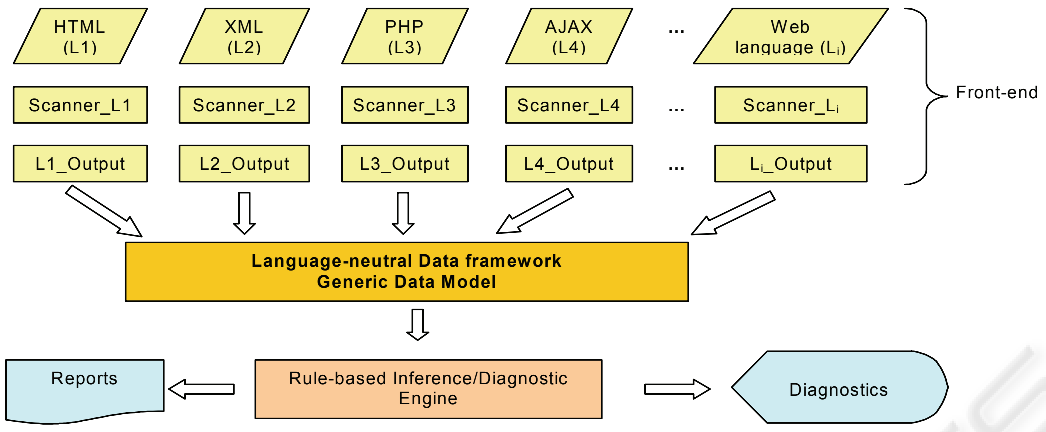


Figure 1: The framework for Web application vulnerability analysis.

notation which creates a secure web environment that automates mundane security operations.

3 NEW VULNERABILITY ANALYSIS MODEL

3.1 Analyzing Scanner Output

Many vulnerabilities are common among different web technologies, the rest are particular for each of them due to language characteristics, compile process, executing methodology. The question is how to detect vulnerabilities with least dependence on development technologies and programming languages. Our approach (Le and Loh 2007) is to develop an analysis technique that would cover vulnerabilities over multiple web-programming technologies. We propose an approach comprising a combination of vulnerability scanners to generate output and categorization with a language-neutral data model using appropriate mining techniques for vulnerability data analysis and rule extraction and classification. A functional layer with a rule-based inference engine and diagnostics capability is needed to generate reports for the users (Figure 1).

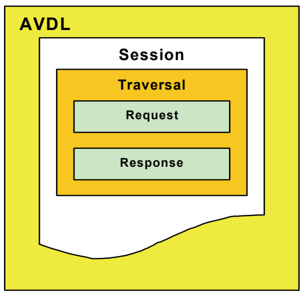
3.2 AVDL – Unified Vulnerability Description

AVDL, VulnXML and WAS are XML-based standards for describing web application security properties and vulnerabilities in uniform way. In our research, we use AVDL (OASIS 2004; OASIS 2004) as a vulnerability description format in the data model. The report generated by the inference engine will also be AVDL-compliant.

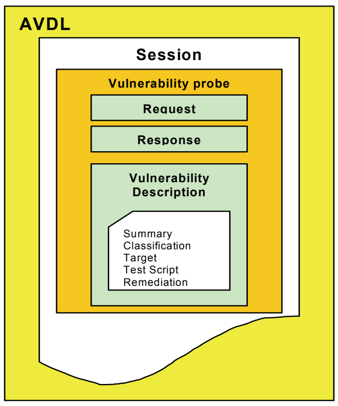
Many analysis techniques rely on the input/output schema and process flow to detect the abnormalities which is considered as potential signatures of vulnerability. AVDL Traversal Structure (TS) output provides information of user-level transaction activity (Figure 2a). This structure describes request/response pair for the round-trip HTTP traversal to the server and contains sufficient descriptive data (type of request, connection methodology, targeted host, URI, protocol version of request data, header structure) needed for vulnerability analysis. The detected vulnerabilities within the web application are described using AVDL Vulnerability Probe Structure (VPS) (Figure 2b).

3.3 Front End

Commercially available static analysis tools and utilities, shareware and freeware (Insecure.org 2007) will be used here for extendability. These tools enable detection coverage of a wider web technology spectrum or even applications that are cooperative. Outputs of these utilities/tools will be mined to form a language-neutral data model that serves as the data interface to a rule-based inference engine. The design of the data model with this approach is sufficiently generic to support future rule-learning in an adaptive rule-based inference engine. Code vulnerability and data integrity rules based on standard vulnerability descriptions would form the core of the prototype rule-base.



(a)



(b)

Figure 2: AVDL: (a) Traversal structure and (b) Vulnerability Probe structure.

3.4 Inference Engine

The design will incorporate a suitable combination of forward and/or backward chaining techniques. As a consideration for extension to support learning is the incorporation of a fuzzy-neuro network to update the rules after conductance of appropriate training.

This rule-based approach has the following advantages:

a) It will have inherently lower execution overheads compared to a run-time monitor that is loaded together with the suspect/compromised program.

b) It can also flexibly handle programs based on an arbitrary programming language and suspected malware/compromised application(s).

c) It avoids the need to compress large execution control traces and the use of pattern matching techniques.

d) More tractable approach than defining a high-level specification language.

4 CASE STUDIES

We conducted case study with Acunetix Cross Site Scripting scanner (Free edition) (Acunetix 2007) and IBM Rational AppScan evaluation version (Watchfire 2007) to test whether web application scanners can cover the same amount of vulnerabilities and provide equivalent outputs. Then, we used AVDL to provide a descriptive format to specifying the architectural views of real vulnerabilities. Because both scanners do not support AVDL we make use of an AVDL schema to describe the scanner outputs and evaluate if the AVDL formatted outputs deliver the same effective description as the original outputs.

The case study is performed on http://demo.testfire.net/ website (Microsoft IIS 6.0 server, APS.NET) provided by IBM Rational AppScan evaluation version. Acunetix Free edition can only detect Cross Site Scripting vulnerabilities on demo website.

Results from test cases show that IBM Rational AppScan detects 79 Security issues in which 7 are XSS vulnerabilities (Figure 3 in Appendix) while Acunetix discovered 73 XSS vulnerabilities (Figure 4 in Appendix). However, Acunetix counts the total instance of vulnerabilities according to variants of each exploit was tested while AppScan counts on the vulnerable positions in file which was scanned. The number of variants of same vulnerability is also different in two scanners. AppScan found vulnerabilities in more files than Acunetix did. This result (Table 2) may indicate that different scanners do not cover same vulnerabilities and they do not provide complete scanning solution even within the same application. Moreover, the vulnerability outputs of each of the two scanners are specific to itself and do not provide equivalent and relevant information.

Users often have a vague idea on how to approach a vulnerability description when referencing the output of more than one scanner. In this experiment, we use AVDL schema (OASIS 2003) to describe the XSS vulnerability extracted from scanners output. First, we list all files which contain XSS vulnerabilities. Then, we compare and select exploit variants together with detail description of vulnerability. The result is a generic XSS vulnerability description of tested website.

Table 2: XSS vulnerabilities detected.

	Vulnerable files	Variants	# vulnerabilities
Acunetix	4	73	73
AppScan	6	76	7

During the case study, we discovered that the use of AVDL is highly effective in making the concept of vulnerability concrete and tangible. With the aid of AVDL, web application vulnerability is no longer an abstract, overlapping and error-prone idea but a tangible object of modeling and analytical specification process.

5 CONCLUSIONS

Web applications having become popular, wide spread and rapidly proliferated raises many security issues and exacerbates the demand for practical solutions. Manual security solutions targeted at these vulnerabilities are language-dependent, type-specific, labor-intensive, expensive and error-prone. In this paper, we have evaluated the use of a language-neutral data model as part of a new framework for web application vulnerability analysis. Our framework is extendible being based on existing web application scanners and AVDL as a uniform vulnerability description format.

At the current stage, we conduct case studies with different web application scanners and evaluating their outputs using AVDL. We continue with the unified data model as a data interface for the rule-based inference engine which incorporates vulnerability analysis and prediction capability. In due course, we hope to provide a commercializable tool to web site administrators and web developers to actively secure their applications.

ACKNOWLEDGEMENTS

This work is funded by Centre for Strategic Infocomm Technologies, Ministry of Defence Singapore.

REFERENCES

Acunetix. (2007). "Acunetix Cross Site Scripting Scanner." from http://www.acunetix.com/cross-site-scripting/scanner.htm.

Bazaz, A. and J. D. Arthur (2007). Towards A Taxonomy of Vulnerabilities. *Proceedings of the 40th Annual Hawaii International Conference on System Sciences, 2007. HICSS 2007.* Waikoloa, HI**:** 163a - 163a.

Berghe, C. V., J. Riordan, et al. (2005). A Vulnerability Taxonomy Methodology applied to Web Services.

Bishop, M. (1999). Vulnerabilities Analysis. *Web proceedings of the 2nd International Workshop on Recent Advances in Intrusion Detection (RAID'99)*, West Lafayette, Indiana, USA.

Cova, M., V. Felmetsger, et al. (2007). Vulnerability Analysis of Web-based Applications. *Test and Analysis of Web Services,* Springer Berlin Heidelberg**:** 363-394.

CVE. (2007). "CVE - Common Vulnerabilities and Exposures (CVE)." from http://cve.mitre.org/.

Dowd, M., J. McDonald, et al. (2006). Chapter 1,2,3,4,8,13,17,18. *The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities*, Addison Wesley Professional.

Fong, E. and V. Okun (2007). Web Application Scanners: Definitions and Functions. *Proceedings of the 40th Annual Hawaii International Conference on System Sciences, 2007. HICSS'07*, Waikoloa, HI, IEEE.

Ghosh, A. K., T. O'Connor, et al. (1998). An Automated Approach for Identifying Potential Vulnerabilities in Software. *Proceeding of the 1998 IEEE Symposium on Security and Privacy***:** 0104.

Grossman, J. (2007). WhiteHat Website Security Statistics Report, WhiteHat Security.

Halfond, W. G. J., A. Orso, et al. (2006). Using positive tainting and syntax-aware evaluation to counter SQL injection attacks. *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering SIGSOFT '06/FSE-14* Portland, Oregon, USA, ACM Press**:** 175-185.

Halfond, W. G. J., J. Viegas, et al. (2006). A Classification of SQL Injection Attacks and Countermeasures. *Proceedings of the IEEE International Symposium on Secure Software Engineering (ISSSE 2006)* Arlington, VA, USA.

Huang, Y.-W., S.-K. Huang, et al. (2003). Web application security assessment by fault injection and behavior monitoring. *Proceedings of the 12th international conference on World Wide Web*. Budapest, Hungary, ACM Press**:** 148-159.

Huang, Y.-W., F. Yu, et al. (2004). Securing web application code by static analysis and runtime protection. *Proceedings of the 13th international conference on World Wide Web*. New York, NY, USA, ACM Press**:** 40-52.

Hurst, D. (2007, 09 Feb 2007). "Asking the Right Question: Penetration Testing vs. Vulnerability Analysis Tools, Which Is Best?" from http://www.infosecwriters.com/texts.php?op=display&id=537.

IBM (2007). Cyber Attacks On The Rise: IBM 2007 Midyear Report, IBM Corporation. IBM Internet Security Systems™ X-Force® Research and Development.

Insecure.org. (2007). "Top 10 Web Vulnerability Scanners." Retrieved September, 2007, from http://sectools.org/web-scanners.html.

Jovanovic, N., C. Kruegel, et al. (2006). Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities (Short paper). *Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06)*: 258-263.

Kals, S., E. Kirda, et al. (2006). SecuBat: A Web Vulnerability Scanner. *Proceedings of the 15th international conference on World Wide Web (WWW 2006)*. Edinburgh, Scotland: 247 - 256.

Le, H. T. and P. K. K. Loh (2007). Unified Approach to Vulnerability Analysis of Web Applications. *The International e-Conference on Computer Science 2007 (IeCCS 2007)*. T. E. Simos.

Livshits, B. and M. S. Lam (2005). Finding Security Vulnerabilities in Java Applications with Static Analysis. *USENIX Security Symposium*: 16.

Minamide, Y. (2005). Static approximation of dynamically generated Web pages. *Proceedings of the 14th International World Wide Web Conference*. Chiba, Japan ACM Press: 432 - 441.

Nguyen-Tuong, A., S. Guarnieri, et al. (2005). Automatically Hardening Web Applications Using Precise Tainting. *Proceedings of the 20th IFIP International Information Security Conference*. Makuhari-Messe, Chiba, Japan.

NT Objectives, I. (2007). "NTOSpider." Retrieved October, 2007, from http://www.ntobjectives.com/products/ntospider.php.

OASIS. (2003). "AVDL XML Schema." Retrieved December, 2007, from http://www.oasis-open.org/committees/download.php/5065/avdl.xsd.

OASIS (2004). Application Vulnerabilty Decription Language v1.0.

OASIS (2004). Technical Overview of the Application Vulnerability Description Language (AVDL) V1.0. Version 1.0, 22 March 2004, OASIS Open.

OASIS. (2007). "Application Security Standards." Retrieved November, 2007, from http://xml.coverpages.org/appSecurity.html.

OASIS. (2007). "OASIS homepage." Retrieved 18 November 2007, from http://www.oasis-open.org/home/index.php.

Raina, K. (2004). "Trends in Web Application Security." Retrieved September, from http://www.securityfocus.com/print/infocus/1809.

SecurityFocus. (2007). "Bugtraq Mailing list." Retrieved 31/10/2007, from http://www.securityfocus.com/archive/1.

SecurityFocus. (2007). "Vulnerabilities list." Retrieved 31/10/2007, from http://www.securityfocus.com/vulnerabilities.

Siddharth, S. and P. Doshi. (2006, 1/11/2007). "Five common Web application vulnerabilities." Retrieved 1/11/2007, from http://www.securityfocus.com/infocus/1864.

SPIDynamics. (2007). "WebInspect." Retrieved September, from http://www.spidynamics.com/products/webinspect/.

Stamp, M. (2006). *Information Security: Principles and Practice*, John Wiley & Sons.

Steffan, J. and M. Schumacher (2002). Collaborative attack modeling. *Proceedings of the 2002 ACM symposium on Applied computing SAC 2002*. Madrid, Spain ACM: 253-259.

Suto, L. (2007, October, 2007). "Analyzing the Effectiveness and Coverage of Web Application Security Scanners." from http://ha.ckers.org/blog/20071014/web-application-scanning-depth-statistics/.

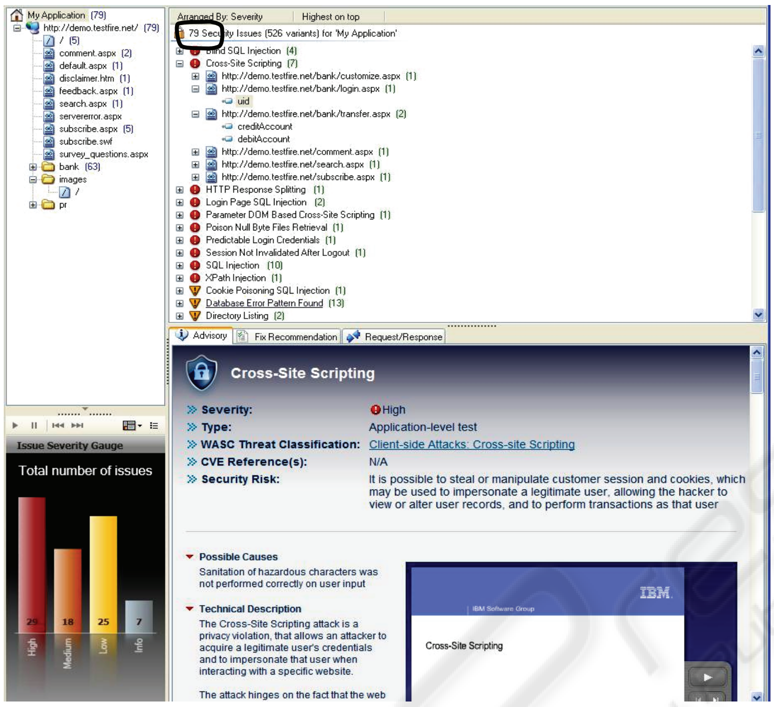
Watchfire. (2007). "AppScan." Retrieved September 2007, from http://www.watchfire.com/.

Woo, S.-W., O. H. Alhazmi, et al. (2006). An Analysis Of The Vulnerability Disovery Process In Web Browsers. *10th IASTED International Conference SOFTWARE ENGINEERING AND APPLICATIONS*, Dallas, TX, USA.

Xie, Y. and A. Aiken (2006). Static Detection of Security Vulnerabilities in Scripting Languages. *Proceedings of the 15th USENIX Security Symposium (USENIX'06)*. Vancouver, B.C., Canada: 179–192.

APPENDIX

Figure 3 and figure 4 are listed in this section due to their over size.



```
My Application (79)
  http://demo.testfire.net/ (79)
    / (5)
    comment.aspx (2)
    default.aspx (1)
    disclaimer.htm (1)
    feedback.aspx (1)
    search.aspx (1)
    servererror.aspx
    subscribe.aspx (5)
    subscribe.swf
    survey_questions.aspx
    bank (63)
    images
      /
    pr
```

```
Arranged By: Severity    Highest on top
79 Security Issues (526 variants) for 'My Application'
  Blind SQL Injection (4)
  Cross-Site Scripting (7)
    http://demo.testfire.net/bank/customize.aspx (1)
    http://demo.testfire.net/bank/login.aspx (1)
      uid
    http://demo.testfire.net/bank/transfer.aspx (2)
      creditAccount
      debitAccount
    http://demo.testfire.net/comment.aspx (1)
    http://demo.testfire.net/search.aspx (1)
    http://demo.testfire.net/subscribe.aspx (1)
  HTTP Response Splitting (1)
  Login Page SQL Injection (2)
  Parameter DOM Based Cross-Site Scripting (1)
  Poison Null Byte Files Retrieval (1)
  Predictable Login Credentials (1)
  Session Not Invalidated After Logout (1)
  SQL Injection (10)
  XPath Injection (1)
  Cookie Poisoning SQL Injection (1)
  Database Error Pattern Found (13)
  Directory Listing (2)
```

Advisory | Fix Recommendation | Request/Response

Cross-Site Scripting

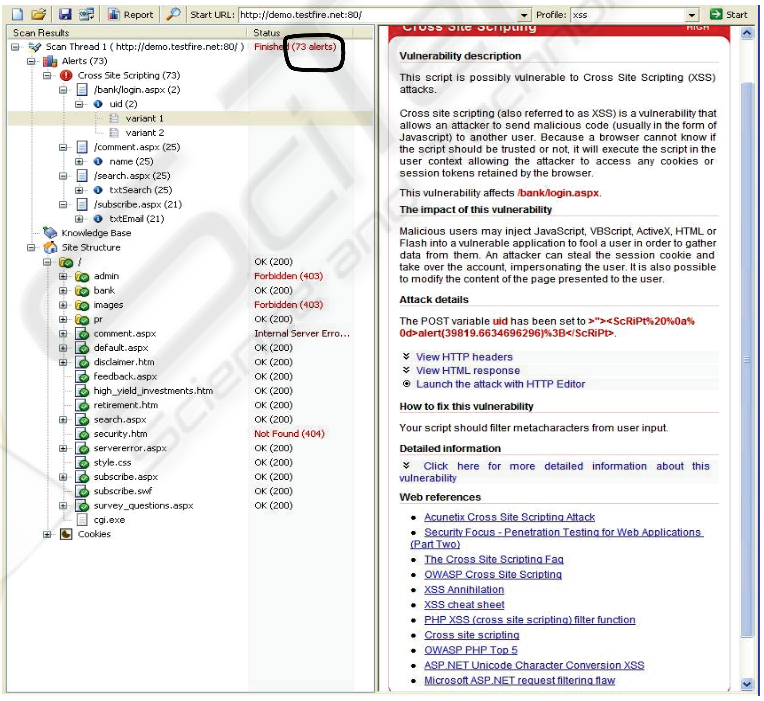
Severity:	High
Type:	Application-level test
WASC Threat Classification:	Client-side Attacks: Cross-site Scripting
CVE Reference(s):	N/A
Security Risk:	It is possible to steal or manipulate customer session and cookies, which may be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user

Possible Causes
Sanitation of hazardous characters was not performed correctly on user input

Technical Description
The Cross-Site Scripting attack is a privacy violation, that allows an attacker to acquire a legitimate user's credentials and to impersonate that user when interacting with a specific website.

The attack hinges on the fact that the web

Figure 3: AppScan result screen shot.



Report | Start URL: http://demo.testfire.net:80/ | Profile: xss | Start

```
Scan Results                                              Status
Scan Thread 1 ( http://demo.testfire.net:80/ )            Finished (73 alerts)
  Alerts (73)
    Cross Site Scripting (73)
      /bank/login.aspx (2)
        uid (2)
          variant 1
          variant 2
      /comment.aspx (25)
        name (25)
      /search.aspx (25)
        txtSearch (25)
      /subscribe.aspx (21)
        txtEmail (21)
  Knowledge Base
  Site Structure
    /                                                     OK (200)
      admin                                               Forbidden (403)
      bank                                                OK (200)
      images                                              Forbidden (403)
      pr                                                  OK (200)
      comment.aspx                                        Internal Server Erro...
      default.aspx                                        OK (200)
      disclaimer.htm                                      OK (200)
      feedback.aspx                                       OK (200)
      high_yield_investments.htm                          OK (200)
      retirement.htm                                      OK (200)
      search.aspx                                         OK (200)
      security.htm                                        Not Found (404)
      servererror.aspx                                    OK (200)
      style.css                                           OK (200)
      subscribe.aspx                                      OK (200)
      subscribe.swf                                       OK (200)
      survey_questions.aspx                               OK (200)
      cgi.exe
    Cookies
```

Cross Site Scripting HIGH

Vulnerability description

This script is possibly vulnerable to Cross Site Scripting (XSS) attacks.

Cross site scripting (also referred to as XSS) is a vulnerability that allows an attacker to send malicious code (usually in the form of Javascript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context allowing the attacker to access any cookies or session tokens retained by the browser.

This vulnerability affects **/bank/login.aspx**.

The impact of this vulnerability

Malicious users may inject JavaScript, VBScript, ActiveX, HTML or Flash into a vulnerable application to fool a user in order to gather data from them. An attacker can steal the session cookie and take over the account, impersonating the user. It is also possible to modify the content of the page presented to the user.

Attack details

The POST variable **uid** has been set to **>"><ScRiPt%20%0a%0d>alert(39819.6634696296)%3B</ScRiPt>**.

View HTTP headers
View HTML response
Launch the attack with HTTP Editor

How to fix this vulnerability

Your script should filter metacharacters from user input.

Detailed information

Click here for more detailed information about this vulnerability

Web references

- Acunetix Cross Site Scripting Attack
- Security Focus - Penetration Testing for Web Applications (Part Two)
- The Cross Site Scripting Faq
- OWASP Cross Site Scripting
- XSS Annihilation
- XSS cheat sheet
- PHP XSS (cross site scripting) filter function
- Cross site scripting
- OWASP PHP Top 5
- ASP.NET Unicode Character Conversion XSS
- Microsoft ASP.NET request filtering flaw

Figure 4: Acunetix Cross Site Scripting results screen shot.