Global Journal of Computer Science and Technology: E
Network, Web & Security
Volume 14 Issue 2 Version 1.0 Year 2014
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Defensive Approaches on SQL Injection and Cross-Site Scripting Attacks

By Venkatramulu Sunkari & Dr. C. V. Guru Rao
Kits Warangal, India

Abstract- SQL Injection attacks are the most common attacks on the web applications. Statistical analysis says that so many web sites which interact with the database are prone to SQL Injection/XSS attacks. Different kinds of vulnerability detection system and attack detection systems exist, there is no efficient system for detecting these kinds of attacks. SQL Injection attacks are possible due to the design drawbacks of the websites which interact with back-end databases. Successful attacks may damage more. The state-of-art web application input validation echniques fails to identify the proper SQL/XSS Vulnerabilities accurately because of the systems correctness of sanity checking capability, proper placement of valuators on the applications. The systems fail while processing HTTP Parameter pollution attacks. An extensive survey on the SQL Injection attacks is conducted to present various detection and prevension mechanisms.

GJCST-E Classification : H.2.7

DEFENSIVEAPPROACHESONSQLINJECTIONANDCROSS-SITESCRIPTINGATTACKS

Strictly as per the compliance and regulations of:

Defensive Approaches on SQL Injection and Cross-Site Scripting Attacks

Venkatramulu Sunkari [α] & Dr. C. V. Guru Rao [σ]

Abstract - SQL Injection attacks are the most common attacks on the web applications. Statistical analysis says that so many web sites which interact with the database are prone to SQL Injection/XSS attacks. Different kinds of vulnerability detection system and attack detection systems exist, there is no efficient system for detecting these kinds of attacks. SQL Injection attacks are possible due to the design drawbacks of the websites which interact with back-end databases. Successful attacks may damage more. The state-of-art web application input validation echniques fails to identify the proper SQL/XSS Vulnerabilities accurately because of the systems correctness of sanity checking capability, proper placement of valuators on the applications. The systems fail while processing HTTP Parameter pollution attacks. An extensive survey on the SQL Injection attacks is conducted to present various detection and prevension mechanisms.

I. Introduction

SQL Injection attack is a web application vulnerability that occurs because of improper validations at the server side. National Vulnerability Database (NVD) is an International security organization and is organized by the U.S Government. In this, most of the security threats and the vulnerability (flaws) will be published. Each Vulnerability (Software Flaws) is identified with CVE-ID. When we see the vulnerabilities (CVE-IDs) published to till date there are total of 60598. Among all these vulnerabilities 5922 are sql injection flaws and 8074 are cross site scripting flaws. Exploit-db is a security community. The site publishes vulnerability details possibly with Proof Of Concept(POC). Vulnerability research or response teams and most of the hackers or crackers participate for their fame and name. This site provides a separate category called web apps. In this category we can see the website hacked details. Currently this site is publishing 100 to 200 POC for every month. Famous and Open Source Intrusion Detection System SNORT is providing detection logics not more than twenty. By these logics we can detect upto 20-40 sql injection attacks. So many commercial IDS/IPS Systems are also providing very few logics. By this analysis we can conclude that, SQL Injection attacks are more and there is no efficient detection system for detecting and for protecting web applications from SQL Injection attacks. In the most of the website home pages we see as the Fig. 1 text and password boxes to enter into the website. For example if we have login and password to use the web services, and login as admin and password as admin0123. We enter login, password and then we click on submit. Our browser sends the http GET request and these values(login, password) will be submitted to the appropriate program file, in the above example validate.jsp as an input parameters. In the middle of the transmission we can observe this request as
”GET http://www.example.com/validate.jsp?
username=admin&password=admin0123 HTTP/1.1 ”.

Here the validatation process on the server is validate.jsp and it accepts the parameters username and the password. If the above request is received by the www.example.com webserver, then that server sends the requested values to the validate.jsp with the argument values. Validate.jsp validates the username and password with its back-end database (Say ORACLE Server). Before interacting with the database validate. Jsp script creates a dynamic SQL Query for validating the user inputs. Let us assume that the code for the validate,.jsp is designed as Fig. reffunction. If this validate.jsp takes admin as username and admin0123 as password, then the dynamic query will be created at the runtime is var sql = ”select * from users where username = ’” + username + ”’ and password = ’” + password + ”’”. Dynamic query will be sql=select * from users where username=admin and password=admin0123. If the user or attacker enters the values for username, and password as ”Username : or 1=1 –” and ”Password : xyz” In the scenario, the dynamic query will be created below sql=select * from users where username= or 1=1– and password=xyz. In the sql statement username= will become one condition which returns false and the condition 1=1 which is tautology condition and returns always true. These two conditions here are joined with or. so that total result will be true for always. And the Statement (–) is used as comment statement in the most of the sql supported database management systems. If this comment statement statement appears in the middle of the SQL Query, then the rest of the query will be ignored. So that when we execute the above SQL Query, The result of execute query(sql) will be non-zero and returns all the records of the users table. And then attacker may gain the admin access,.(Because of the

Author α: Associate Proffesor In Cse Kits, Warangal.
Author σ: Proffessor and Head of Cse Department, SR Engineering College, Warangal. e-mails : venkatramulu10@gmail.com, guru_cv_rao@hotmail.com

entered user will be treated as the result of the first record and most of the SQL users Tables first record may be the admin). Because of no validations are done at the server-side for the user inputs, an attacker execute his own queries, instead of the developer expected query. And it is possible to insert another SQL Querries by

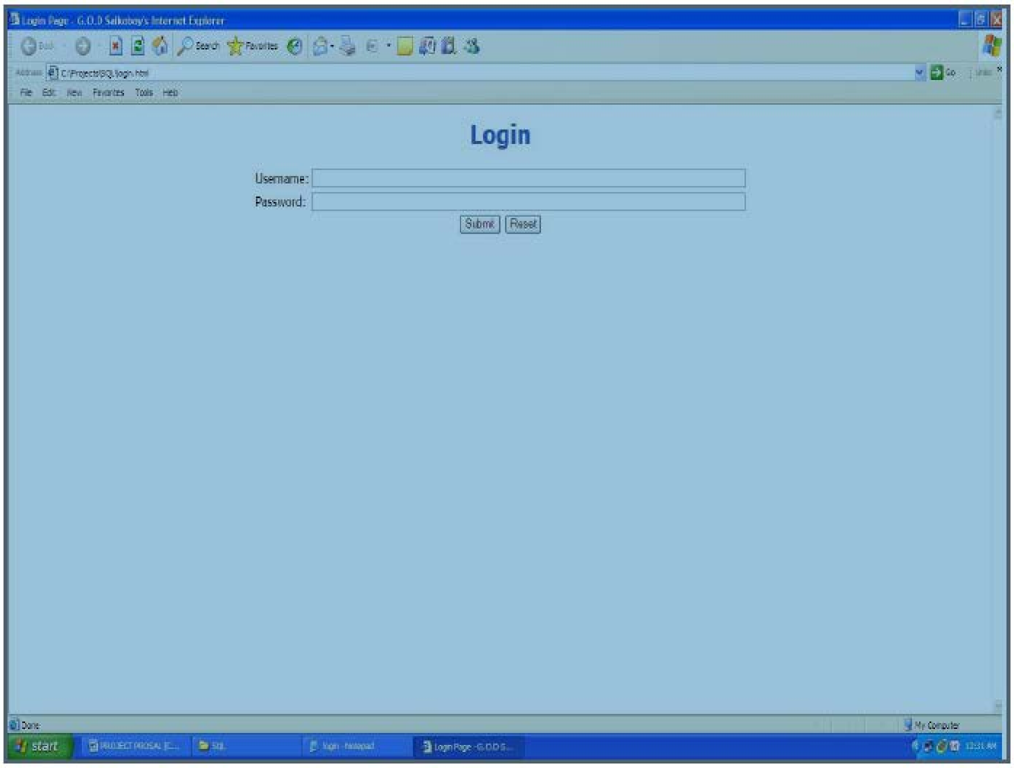
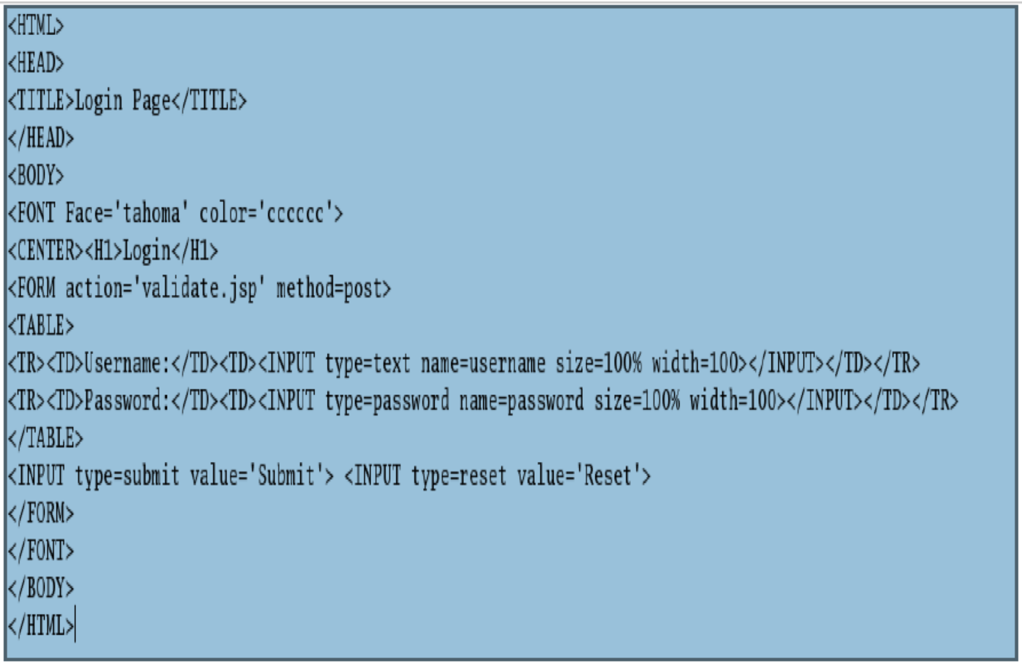
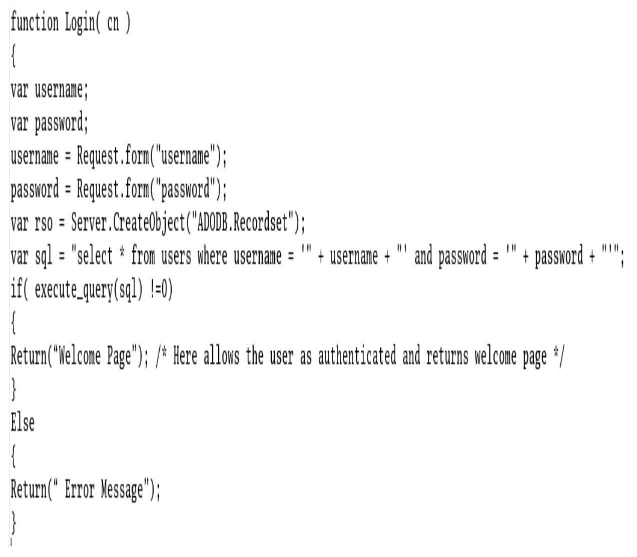


Figure 1 : Sample Login Screen In the Web Applications



```
<HTML>
<HEAD>
<TITLE>Login Page</TITLE>
</HEAD>
<BODY>
<FONT Face='tahoma' color='cccccc'>
<CENTER><H1>Login</H1>
<FORM action='validate.jsp' method=post>
<TABLE>
<TR><TD>Username:</TD><TD><INPUT type=text name=username size=100% width=100></INPUT></TD></TR>
<TR><TD>Password:</TD><TD><INPUT type=password name=password size=100% width=100></INPUT></TD></TR>
</TABLE>
<INPUT type=submit value='Submit'> <INPUT type=reset value='Reset'>
</FORM>
</FONT>
</BODY>
</HTML>
```

Figure 2 : Sample HTML Code To Send Login Data



```
function Login( cn )
{
var username;
var password;
username = Request.form("username");
password = Request.form("password");
var rso = Server.CreateObject("ADODB.Recordset");
var sql = "select * from users where username = '" + username + "' and password = '" + password + "'";
if( execute_query(sql) !=0)
{
Return("Welcome Page"); /* Here allows the user as authenticated and returns welcome page */
}
Else
{
Return(" Error Message");
}
```

Figure 3 : Sample Validation Function

combining with UNION Statement. Example: If the attacker enters below values username: or 1=1 union insert into users values(sreedevi,sreedevi0123,admin) password: xyz Like this if any vulnerability found on the website parameter, an attacker can inject his own queries for insert,update,etc.

The result of the SQL Injection will be very severe. Like this we can find more number of attack or hacked details in the security websites.

II. Sql Injection Attacks

In the most of the website home pages we see the text and password boxes as shown in Fig. 1 to enter into the website. In general this page is used to allow the authorized persons from the remote to use the web application services. For this kind of pages, most of the developers develop the code as Fig. 2. For example if we have login and password to use the web services, and login as admin and password as admin0123. We enter login, password and then we click on submit. Our browser sends the http GET request and these values(login, password) will be submitted to the appropriate program file, in the above example validate.jsp as an input parameters. In the middle of the transmission we can observe this request as GET/validate.jsp?username = admin&password = admin0123 HTTP/1.1 Here the validatation process on the server is validate.jsp and it accepts the parameters username and the password. If the above request is received by the www.example.com webserver, then that server sends the requested values to the validate.jsp with the argument values. Validate.jsp validates the username and password with its back-end database (Say ORACLE Server). Before interacting with the database validate. Jsp script creates a dynamic SQL Query for validating the user inputs. Let us assume that the code for the validate,.jsp is designed as Fig. reffunction.

a) Validate.jsp

If this validate.jsp takes admin as username and admin0123 as password, then the dynamic query will be created at the runtime is

$varsql = "select\_ fromuserswhereusername = 0" + username + "0andpassword = 0" + password + "0";$ Dynamic query will be $sql = select_ fromuserswhereusername = adminandpassword = admin$0123; If the user or attacker enters the values for username, and password as below
Username : or 1=1 - -
Password : xyz

In the above scenario, the dynamic query will be created below

$sql = select\_ fromuserswhereusername = or1 = 1 - -andpassword = xyz;$ in the above sql statement username= will become one condition which returns false and the condition 1=1 which is tautology condition and returns always true. These two conditions here are joined with or. so that total result will be true for always. And the Statement (–) is used as comment statement in the most of the sql supported database management systems. If this comment statement statement appears in the middle of the SQL Query, then the rest of the query will be ignored. So that when we execute the above SQL Query, The result of execute query(sql)will be non-zero and returns all the records of the users table. And then attacker may gain the admin access,.(Because of the entered user will be treated as the result of the first record and most of the SQL users Tables first record may be the admin). Because of no validations are done at the server-side for the user inputs, an attacker execute his own queries, instead of the developer expected query. And it is possible to insert another SQL Querries by combining with UNION Statement. Example: If the attacker enters below values username: ' or 1=1 union insert into users values(venkat,venkat0123,admin) - -
password: xyz

Like this if any vulnerability found on the website parameter, an attacker can inject his own queries for insert,update,etc,.

The result of the SQL Injection will be very severe. Like this we can find more number of attack or hacked details in the security websites.

III. Sql Injection Defence Schemes

a) PaulE et al Scheme

In [1] the authors developed a white-box tool to verify software security. In general software requirement specifications, source code, designs and executable code to be analysed by tools. In this work, the authors developed a security scanner tool. It can analyse the functional bevaior. Due to the widespread use of the World Wide Web and proliferation of web application

World Wide Web and proliferation of web application vulnerabilities, application level web security and assurance requires major attention. This specification defines a minimum capability to help software professionals understand how a tool will meet their software assurance needs. The tool can be used as software assurance tool and it can scan the software for security vulnerability for some extent.

b) Muthuprasanna et.at Method

In [2] the authors developed a model of hybrid approach, which combines static code verification and runtime analysis. Webservice protection became necessary because of the use of webapplications is increasing in the internet. Deployments of Webapplication firewalls, next generation firewalls, application detection systems and intrusion detection and prevention systems are increasing to protect web servers.

c) Hossain et.at Method

In [3] the authors developed a mutation based testing tool to verify the web application resistance against SQL Injection vulnerabilities. The authors stated that the current scenario of testing web application cannot eliminate web application vulnerabilities. The proposed that injecting attack pattern into the source code of the web application, by that mutations based test cases can be generated. The generated test cases can potentially find the SQL Injection vulnerabilities. The authors named the tool as MUSIC. The tool is evaluated on open source web applications written in JSP. The tool is further impleted for PHP and other known languages.

d) Russell et.at Method

In [4] the authors developed a low-level approach to find the runtime applicability of sql statement which are prepared at runtime. The authors achieved this using call level interface (CLI) by interacting ODBC or JDBC. Using this approach, the authors evaluated the runtime SQL statement with SQL DOM approach. CLI can be used for to verify the correctness, but SQL DOM is can be used to identify the SQL statement applicability such as user permissions. SQL DOM can be prepared automatically by interacting with the database schema. The authors evaoluated the system for performance. The approach is a offline approach. The posed SQL statements should be given as an input the tool. So, it cannot be directly applied for dynamic query evoluation. The authors are extending the work with XPATH query language for dynamic queries verification.

e) Tania et.at Method

Software systems are complex for verification and validation. Software faults causes security vulnerabilities and causes for security breaches. Several methods such as SQL attack tree models and fault injection models are best comparable to this work [5]. The tool injects the critical attack patterns onto the system and verifies the result for vulnerability existence. The validation methods provided with this tool avoids false positives. The tool reports accurate report, each vulnerability reported by the tool will be based on the behavior of the application at the time of attack injection. As a future work, the authors targeted to generate injection methods based on attack tree models.

f) Huajun et.al Method

Phishing attack is an identity theft attack, mostly on banking, online-transactions etc,. The attackers uses socail sites to steal the user's sensitive information such as credit-card details, account details etc,. Phishing also includes social engineering schemes. Social engineering schemes can be using emails, phone calls claiming that the callers are from valid authorities. Phishing attacks are typically cross-site scripting attacks. The authors [6] proposed few strategies to avoid phishing attacks.

Anti-phishing are classified into three categories by the authors. Server-side anti-phishing strategies, browser-side antiphishing strategies, and online training anti-phishing strategies.

1. Server-side anti-phishing strategies: This approach will be applied the server side. It works similar to anti-spam systems. It verifies the content delivered to the server. If anything which is very closely related to phishing, the detection system prevents it at the server and not to reached to the victim.
2. Browser-side anti-phishing strategies: This approach is brower based approach with plug-in. The plug-in monitors the application behavior at the user-side. If it behaves as cheating or phishing it avoids the attack. The browser based approaches can be categorised as Blacklist approach, visual-clue-based approaches or capta based approach, webpage-feature-based approaches and information flow approaches.
3. Online training anti-phishing strategies: The last strategy suggests that the internet users should have proper training on phishing attacks, how to avoid them. This approaches clears the anti-phishing philosophies. The strategy is to create awareness on phishing attacks. The authors suggest to have a technology called webpage watermarking to fight against phishing attacks.

g) Anderson Morais et al Method

Attack Injection model [7] for security protocol testing suggests to have attack injection to find the web application vulnerabilities. The approaches includes attack tree model to generate testcases. The attack tree model prepares all possible cases. fault injector injects attack patterns onto the server system. The fault injects prepares executes scripts that are collected from the internet. The approach can be used as blackbox model.

The authors created a framework which executes the given scripts. The authors are focusing on UML based representation to generate attack scenarios in future.

h) Sushila Madan et al Method

Web applications are most vulnerable to popular attacks and risks. SQL Injections and cross-site scripting attacks are more popular attacks on web applications. Threat modeling provides a complete assessment on the web application. With techniques such as attack possible entry point, attack trees, privilise escalation chances the tester or security assessment team can indentify the threats on the system. In [8] the authors aimed to create attack risk model called ADMIRE. The system is consise, structured. The approach is step wise approach. The steps includes : (i)Analyze the security objectives (ii) Divide the application (iii) Mark the vulnerabilities (iv) Identify the threats (v) Rank the threat (vi) Eliminate the threat. The model is white-box model. It verifies the application code.

It is specific to a programming language.

i) Parvaiz et al Method

In [9], the authors suggests that the attack tree model is not possible in all cases and is difficult to build the security model. Applications operates in different modes, capturing every aspect becomes difficult to design the security model with attack tree models. Hence the authors proposed a new approach which provides syntax and graphical security models. The new model includes nodes such as PAND node, k/n node, SEQ node, CSUB node, and Housing node. The system provides syntax and graphical represntation for every node. The model allows the developer to understand the system affectively. The system is fault resistant and avoid vulnerabilities during development phase. In most of the cases the tree includes AND/OR models to represent the system structure. As a future plan, the authors are working on to define calculation rules for the new nodes to distingush the node values for different security attributes up to the root node of the tree.

j) Nenad et al Method

[10] Along with web applications even vulnerabilities have grown. Since reviews of manual code are costly, timeconsuming and even error-prone, the need for solutions has become evident. This addresses the problem of web applications which is vulnerable by means of static source code analysis. Many analysis like flow-sensitive, interprocedural, context-sensitive data flow and even literal analysis are used to discover vulnerable points in a program and also to improve the correctness and precision of the results. Pixy, the open source prototype implementation of our concepts, is targeted at detecting cross-site scripting vulnerabilities in PHP scripts.

The system is capable to cover huge number of vulnerabilities. The system can scan the application code dynamically.

k) Kaarina Karppinen et al Method

These days the big problem is Hidden functionality whereas we cannot be sure that the software does not contain malicious code. Due to architecture violations many security vulnerabilities arises and architecture analysis tools will assist in detecting these vulnerabilities.Such visual images can be used to detect vulnerabilities and ultimately help to design software architectures that meet their security requirements.SAVE [11] is one approach used to detect the violation and what effects the violation had on the system. This kind of analysis with SAVE is new and proving to be advantageous as it adds more details to the evaluation. The SAVE downside is that it is more complex compared to static analysis. The future plans will include developing the SAVE tool further by adding more features, such as automatic comparison of dynamic views and encoding of correct visual images that visual images that together could be used to identify malicious behaviour.

IV. Cross-Site Scripting Attacks

This attack can be done on the vulnerable web application to inject the attacker code. Using this attack, an attacker can inject his own code such as javascript into the web application. Some of the results of the Cross Site Scripting attacks are website hacking or web site defacement,. Whenever user requests the hacked website, then the attacker page will be returned. For example NOKIA website is hacked using cross site scripting, In the Hacked time if any user access the NOKIA website, users will get the hackers page,. By this attack an attacker can gain the sensitive information of the website,. And he can disrupt the webservices.

a) Example 1

In most of the websites, we can see the login and password information. If there are no validations for the user inputs, then the attacker can inject his HTML or SCRIPT code as inputs to the vulnerable pages. By this attacker executes his own script on the server side or the client side.

b) Example 2

If the web page is like FIGURE 1.0, and the user has username and the password like

username:venkat

Password :venkat0123

And The Result of the submission of the user inputs is like Here Web Server is returning a dynamic web page with the user inputs. Attacker can send an attack below

Username : venkat < script > alert(SiteisHacked);< /script >

Password : venkat0123

The above script tag is executed in the web server and the result will be submitted to the validate.jsp. if the above script is written for attackers purpose then that will be very dangerous.

V. Conclusion

The state-of-art web application input validation techniques fails to identify the proper SQL/XSS Vulnerabilities accurately because of the systems correctness of sanity checking capability, proper placement of valuators on the applications. The systems fail while processing HTTP Parameter pollution attacks. Hence the paper proposes a novel technique called Input PArameter Analysis System (IPAAS). The proposed system works in three phases as Input Parameter Extraction, Parameter Type Learning, and Runtime detection with the learned Parameter Types. Because the system operates on self learning approach, and applies on the HTTP traffic, it reduces the developers or security analysts efforts and increases the chances of attack detection accuracy.

References Références Referencias

1. P. E. et al. (2008, Feb) Software assurance tools: Web application security scanner. Functional Specification Version 1.0.
2. M. Muthuprasanna, K. Wei, and S. Kothari, “Eliminating SQL Injection Attacks - A Transparent Defense Mechanism,” in International Workshop on Web Site Evolution, 2006, pp. 22–32.
3. H. Shahriar and M. Zulkernine, “MUSIC: Mutation-based SQL Injection Vulnerability Checking,” in International Conference on Quality Software, 2008, pp. 77–86.
4. R. A. McClure and I. H. Krger, “SQL DOM: compile time checking of dynamic SQL statements,” in International Conference on Software Engineering, 2005, pp. 88–96.
5. T. Basso, P. C. S. Fernandes, M. Jino, and R. Moraes, “Analysis of the effect of Java software faults on security vulnerabilities and their detection by commercial web vulnerability scanner tool,” in International Conference on Dependable Systems and Networks Workshops, 2010.
6. H. Huang, J. Tan, and L. Liu, “Countermeasure Techniques for Deceptive Phishing Attack,” in International Conference on New Trends in Information and Service Science, 2009.
7. A. N. P. Morais, E. Martins, A. R. Cavalli, and W. Jimenez, “Security Protocol Testing Using Attack Trees,” in IEEE International Conference on Computational Science and Engineering, 2009, pp. 690–697.
8. S. Madan and S. Madan, “Shielding against SQL Injection Attacks Using ADMIRE Model,” in International Conference on Computational Intelligence, Communication Systems and Networks, 2009.
9. P. A. Khand, “System level security modeling using attack trees,” in International Conference on Computer, Control and Communication, 2009.
10. N. Jovanovic, C. Kruegel, and E. Kirda, “Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities (Short Paper),” in IEEE Symposium on Security and Privacy, 2006, pp. 258–263.
11. K. Karppinen, M. Lindvall, and L. Yonkwa, “Detecting Security Vulnerabilities with Software Architecture Analysis Tools,” in International Conference on Software Testing, Verification, and Validation, 2008.