American Journal of Applied Sciences 11 (7): 1164-1171, 2014
ISSN: 1546-9239

doi:10.3844/ajassp.2014.1164.1171 Published Online 11 (7) 2014 (http://www.thescipub.com/ajas.toc)

HUNTING PERNICIOUS ATTACKS IN WEB APPLICATIONS WITH XPROBER

Suguna, R., T. Kujani, N. Suganya and C. Krishnaveni

Department of CSE, SKR Engineering College, Chennai, India

Received 2014-02-27; Revised 2014-03-07; Accepted 2014-05-02

ABSTRACT

Nowadays internet is loaded with tons of innovative web applications. This instantaneous growth has paved way for a number of security exposures. Cross Site Scripting attacks (XSS), SQL Injection (SQLI) and Malicious File Execution (MFE) are the foremost web related vulnerabilities reported by Open Web Application Security Project (OWASP). The attackers take advantage of the vulnerabilities in the code of the web applications and engage in activities such as data breach, cookies stealing and password theft which results in severe consequences. The major cause for these glitches is that the scripts allow the user input without scanning for pernicious contents. Several security measures on server-side also available, but they are not applied in large scale, because of the deployment difficulty. On the Client-side, usage of security software worsens the client system's performance which in turn reduces the web surfing experience of the user. A new tool called XProber has been presented for verifying the string manipulating programs automatically. The pre and post conditions of common string functions using Push Down Automata (PDA) are computed and used to identify the presence of vulnerabilities. This approach is capable of finding hefty amount of pernicious attacks in web application and prevents the attacks.

Keywords: XSS, SQLI, MFE, PDA, XProber

1. INTRODUCTION

Web application has taken a new substantial resources of information communication among several types of service providers and end users. Computer Emergency Response Team (CERT) has issued an advisory on newly identified security vulnerabilities which affects all the web applications (OWASP, 2007). Cross site scripting, better known as XSS, is a subset of HTML injection. XSS is the most prevalent and pernicious web application security issue. XSS flaws occur whenever an application takes data that originated from a user and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser, which can hijack user sessions, deface web sites, insert hostile content, conduct phishing attacks and take over the user's browser using scripting malware. The malicious script is usually JavaScript, but any scripting language supported by the victim's browser is a potential target for this attack. Injection flaws, particularly SQL injection, are common in web applications. There are many types of injections: SQL, HTML, XML, OS command injection and many more. Injection occurs when user-supplied data is sent to an interpreter as part of a command or query. All web application frameworks that use interpreters or invoke other processes are vulnerable to injection attacks. Malicious File Execution (MFE) vulnerabilities exist in many web applications. Developers directly use or concatenate potentially aggressive input with some file or stream functions, or improperly trust the input files on the websites. This attack is particularly prevalent on PHP and extreme care must be taken with any stream or file function to ensure that user supplied input does not influence file names (OWASP, 2007).

The area of web usability has long intrigued researchers. It has been widely accepted that for a website to be successful, the level of usability has to be high. The reason is because of poorly designed website (Teoh *et al*., 2009).

Corresponding Author: SUGUNA, R., Department of CSE, SKR Engineering College, Chennai, India

Online advertising is a multibillion dollar business nowadays. Increasing web traffic to a site by directing or referring users provides a mechanism for organizations and individuals to make money through affiliate marketing (Blanc *et al*., 2011). The web provides the perfect framework for malware authors to blend together the techniques listed. This Malware redirects the traffic payload. Today's threats includes spam with exploit scripts to efficiently infect unsuspecting victims. It is necessary to propose suitable detection and prevention mechanisms to provide security for the information contents used by the web application (Kadirvelu and Arputharaj, 2011). **Figure 1** provides an overview of the key roles played by the web applications in malware attacks.

2. RELATED WORK

2.1. Cross Site Scripting

Cross-Site Scripting (XSS) is a type of computer security vulnerability typically found in Web applications that enables attackers to inject client-side script into Web pages viewed by other users. A XSS may be used by attackers to bypass access controls. XSS carried out on websites accounted for 80% of all security vulnerabilities documented by Symantec as of 2009.Their effect may range from a small inconvenience to a significant amount of security risk, based on the sensitivity of the data handled by the vulnerable site and the nature of any security mitigation implemented by the site's owner (https://www.owasp.org/index.php/Cross-site Scripting_(XSS)).

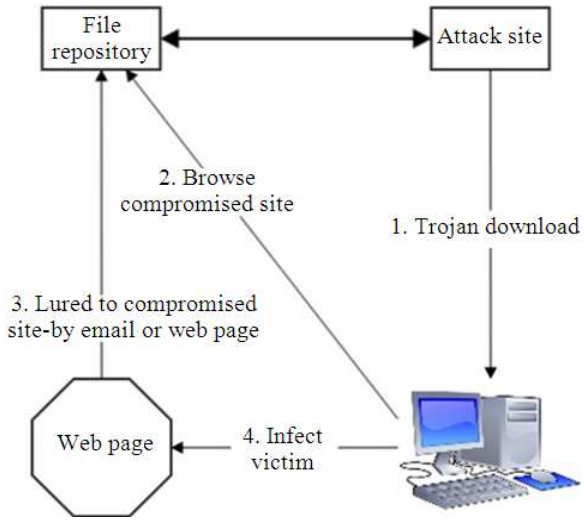

Fig. 1. An example Web site attack

There is no single, standardized classification of XSS flaws, but experts distinguish between two primary flavours: Non-persistent and persistent XSS. Some sources further divide these two groups into traditional (caused by server-side code flaws) and DOM-based (in client-side code). Cross-Site Scripting (XSS) is an attack technique that involves injecting attacker-supplied code into a user's browser. A browser instance can be a standard web browser client, or an object embedded in a software product such as the browser within an RSS reader, Win Amp, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to any other browser-supported technology.

When an attacker gets a user's browser to execute their code, the code will run within the security context (or zone) of the hosting web site. The code has the ability to modify and transmit any sensitive data which is used by the browser. XSS vulnerabilities have been reported and exploited since the 1990s. A prominent site affected in the past includes the sites like Twitter, Facebook, MySpace and Orkut etc. In recent years, cross-site scripting flaws surpassed buffer overflows to become the most common publicly reported security vulnerability. Many websites are open to XSS attacks.

A Cross-site scripted user could have their account hijacked for example stealing user cookies, redirecting the browser to another location, or possibly shows some fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks compromise the trust relationship between a web user and the web site.

2.2. SQL Injection

SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure or code that constructs SQL statements should be checked for injection vulnerabilities because SQL Server will execute all the queries that it receives which are syntactically valid. Even the parameterized data can be manipulated by the attacker who is skilled and determined (http://en.wikipedia.org/wiki/SQL_injection). The SQL injection consists of direct insertion of code into user-input variables that are combined with the SQL commands and executed. Some direct attack injects malicious code into strings that are destined for storage in a table. The malicious code is executed if the stored strings are subsequently concatenated into a dynamic SQL command. The injection process works by prematurely terminating a text string and appending a new command.

The inserted command may have additional strings appended to it before it is executed. The attacker terminates the injected string with a comment mark "--". Subsequent text is ignored during the execution time.

In SQL Injection (SQLI), the attacker executes malicious database statements by exploiting inadequate validation of data flowing from the user to the database. Using SQL injections, attackers can: Perform an INSERT in the injected SQL, ADD new data to the database, Could be embarrassing to find yourself selling politically incorrect items on an ecommerce site, Can MODIFY the data currently in the database, Can perform an UPDATE in the injected SQL, Can gain access of other user's system by obtaining their password. The SQL injection attack is shown in **Fig. 2**. All web application frameworks that use interpreters or invoke other processes are vulnerable to injection attacks. If user input is passed into an interpreter without validation or encoding, the application is vulnerable.

2.3. Malicious File Execution

MFE vulnerabilities exist in many web applications. Developers directly use or concatenate potentially aggressive input with some file or stream functions, or improperly trust the input files on the websites. On many platforms, frameworks allow the use of the external references like URLs or file systems. When the data is not checked properly, this can lead to arbitrary remote and aggressive content being invoked or processed by the web server. This allows attackers to perform:

- Remote code execution
- Remote root kit installation and complete system compromise
- On Windows, internal system compromise may be possible through the use of PHP's SMB file wrappers

This attack is particularly prevalent on PHP and extreme care must be taken with any stream or file function to ensure that user supplied input does not influence file names (OWASP, 2007). **Figure 3** shows a scenario of a Malicious File execution attack. Some of the tools used by the Existing system to prevent the Pernicious Attacks are enumerated below.

2.4. Cross site Scripting Attack (XSS)

- Term Rewriting System (Huang *et al*., 2003)
- Encryption Techniques (Mono Alphabetic substitution scheme)
- Cookie Rewriting Technique

2.5. Malicious File Execution (MFE)

- Content Sniffing Blocker

2.6. SQL Injection Attacks (SQLI)

- Data flow Analysis
- Constraint Analysis

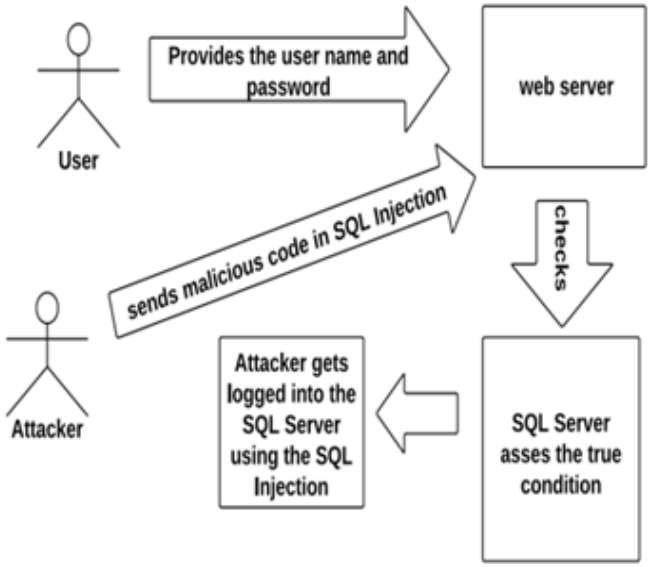

Fig. 2. SQL injections

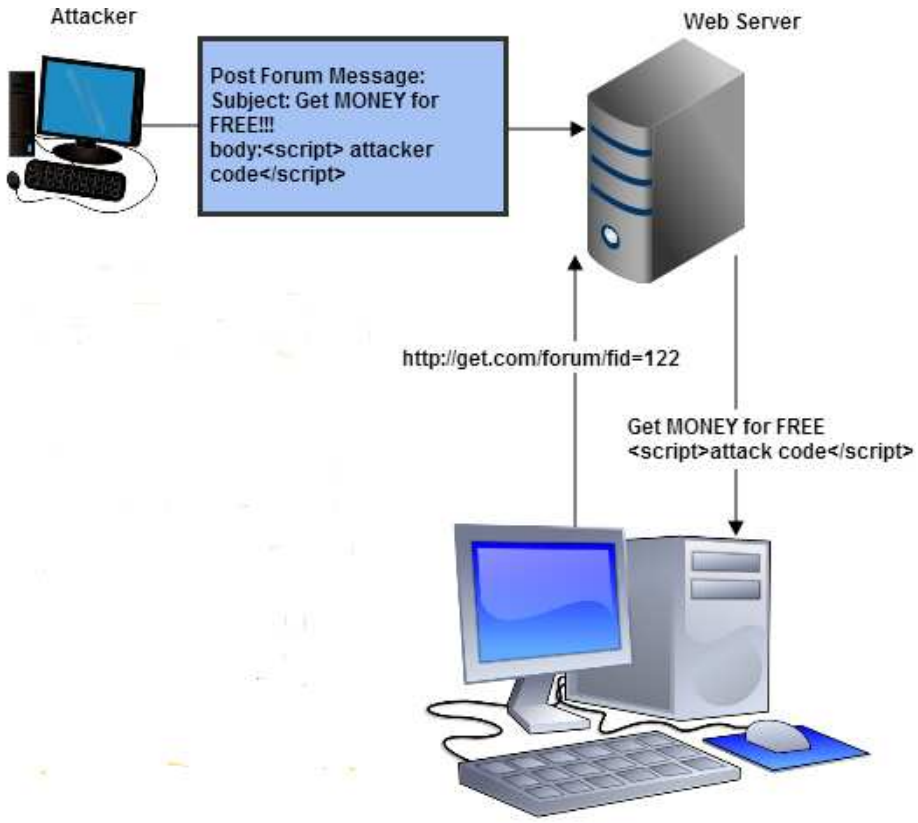

Fig. 3. Scenario of a Malicious File execution attack

Web Security via Static Analysis and Runtime Inspection (Web SARI) code analysis tool pinpoints the code requiring runtime checks and inserts the checks (Sanctum Inc., 2004). For automated Web application security assessment, this tool can be effectively used. Web Application Vulnerability and Error Scanner (WAVES)-black-box security testing tool for Web apps (Rattipong and Bunyatnoparat, 2011) used to identify poor scripting practices that leads the web apps vulnerable to XSS, SQLI, MFE etc. Similar methodologies are implemented by profitable projects such as Kavado's Scan, SPI Dynamic's Web Inspect and AppScan (Kiezun and Jayaraman, 2009). This methodology do not deliver instant Web application security. It also consumes resource excessively on the server which may severely degrade its performance.

The most effective solution is to disable the support for all scripting languages on the client side. If this is not possible, it is recommended to provide caution while browsing dubious web pages and clicking on links in anonymous e-mails. Also, updating the browser to the latest version and patches is important (Tiwari and Bansal, 2008). But typically, users do not disable all scripting language support or to update their browsers.

3. SYSTEM MODEL

Motivated by the existing issues, a innovative Tool named XProber is presented to prevent web browsers from attacks (Arulsuju, 2011). Experimental result indicates that the XProber detection method is an innovative method and it can be used to detect the above mentioned three attacks in the web application program (Yu *et al*., 2010). Compared to the existing systems the performance of the proposed system is higher. An automata-based symbolic string analyses for automatic verification of string manipulating programs is used (Hopcroft *et al*., 2000). Push Down Automata

(PDA) is used for computing the pre- and post-conditions of the common string functions (Sipser, 1997). The concept of PDA is explained below.

A Push Down Automaton (PDA) is one of the types of automation with a memory. The concept of Stack automata in PDA can recognize a more number of languages. PDA can handle all context-free languages. The PDA reads a symbol from the top of the Stack only. The Push and Pop operations takes place only on the top of the PDA as shown in **Fig. 4**. The stack of the PDA contains the unprocessed data and a traversal takes place in pre-order. Pushdown automata choose a transition by indexing a table by input, the symbol at the top of the stack and the current state. This means that those three parameters completely determine the transition path that is chosen.

Thus, the tool developed works with the concept of PDA for detecting vulnerabilities in web applications and with proper sanitization results in the removal of vulnerabilities. The proposed XProber system model is shown in **Fig. 5**.

3.1. Parser and Taint Analyzer

The initial step in this analysis is that the given input, PHP Script is parsed and the Control Flow Graph (CFG) is constructed by the Parser. PHP programs do not have a single entry point so each script is processed by itself along with all files included by that script. The CFG is then sent to the taint analyzer where the alias and dependency analyses are done to generate dependency graphs. The number of its nodes is linear to the number of the string operations in the program under a static environment. Loop structures contribute cyclic dependency relations. If there is no tainted data flow to the sink, taint analysis reports that the dependency graph is secure; otherwise, the dependency graph is tainted and passed to the string analyzer for more inspection.

3.2. String Analyzer

The string analyzer implements the vulnerability which is identified by the taint analysis based on the tainted dependency. The dependency graphs are pre-processed to provide the optimized results. A new acyclic dependency graph is constructed and the vulnerability analysis is done on the acyclic graph so that the nodes not in a cycle are processed only once. In the forward analysis, the post images to nodes are propagated in the topological order, initializing input nodes to PDAs accepting arbitrary strings. Upon termination, an intersection of the language of the PDA of the sink node with the attack pattern is performed. The sink is not vulnerable with respect to the attack pattern only when the intersection is not empty. Otherwise, we perform the backward analysis and propagate the pre images to nodes in the reverse topological order, initializing the sink node to a PDA that accepts the intersection of the result of the forward analysis and the attack pattern. Therefore the vulnerability signatures are the results of the backward analysis for each input node.

3.3. Automata Based Library (ABL)

Automata operations such as concatenation, intersection, replacement, widen, union and all core string operations are handled by ABL. All string and automata manipulation operations that are required are sent to ABL along with the string and/or automata parameters during the vulnerability analysis. ABL executes the operations mentioned and returns automaton.

4. IMPLEMENTATION

This solution has been implemented using open source Mozilla Firefox 1.5 web browser. The Mozilla Firefox web browser executes JavaScript programs included in web pages with the help of the Prevention tool called XProber. The tool plays a significant role in the implemented web browser. It is used to execute JavaScript programs that appear in web pages. Mozilla combined with the tool XProber does not allow any malicious code to execute on it. So the clients using Mozilla with XProber is free from malicious attacks.

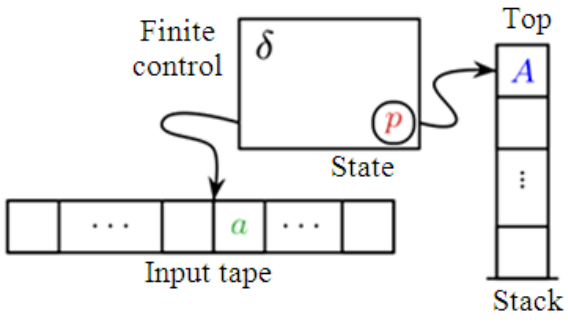

Fig. 4. A diagram of the pushdown automaton

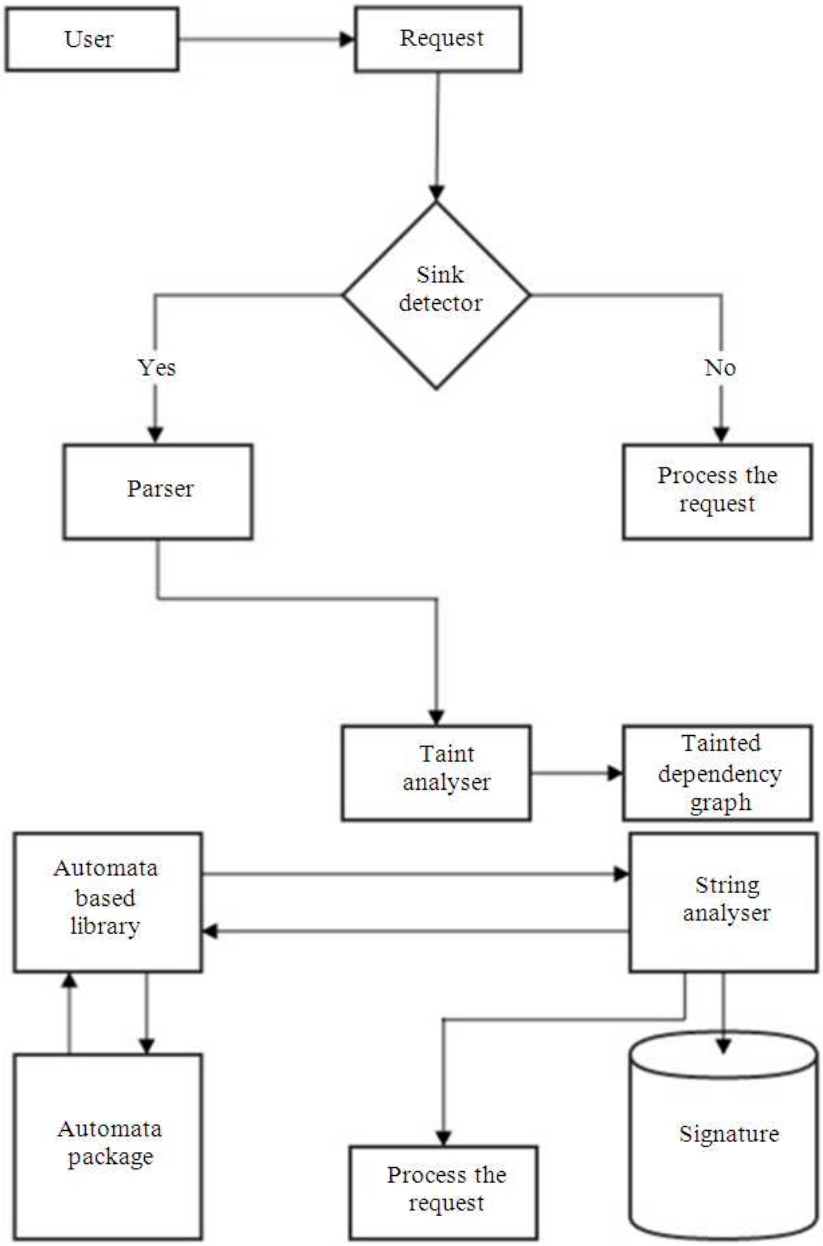

Fig. 5. Proposed system

4.1. Security Evaluation

The proposed solution has been tested with the malicious inputs on vulnerable websites.

Figure 6 shows the proportion of potential vulnerabilities in the modern web browser like Firefox, Microsoft's internet explorer and opera on the same architecture and environment without security implementation. It has been observed that there are many variants of XSS attacks exist and the approach is tested with the data collected from various research sites. It has been observed that these potential vulnerabilities have been decreased drastically after the implementation of XProber.

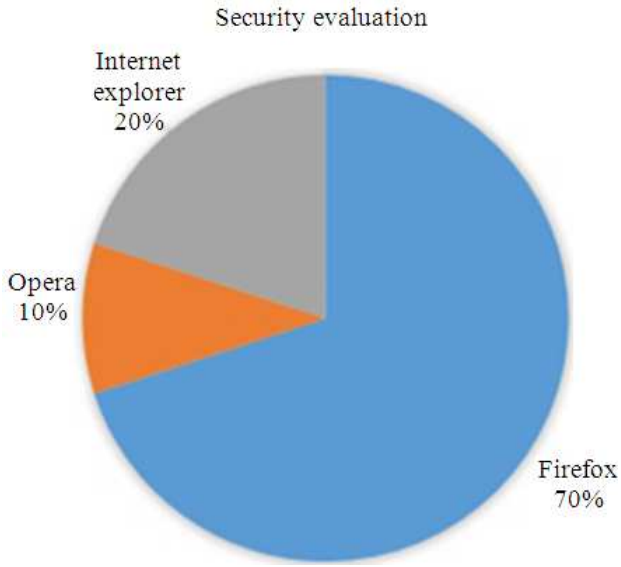

Fig. 6. Security evaluations on different browsers

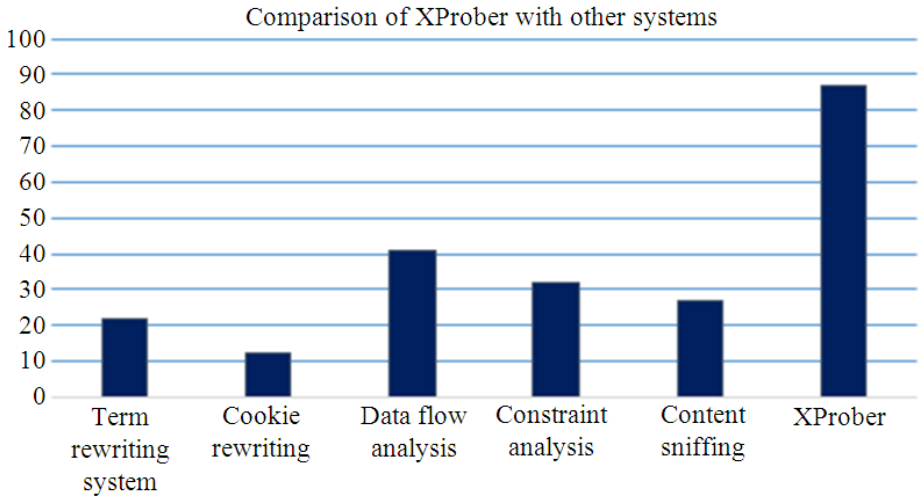

Fig. 7. Comparison of existing system with proposed system

4.2. Performance Evaluation

The performance of the end user's system has not been affected by the implementation of XProber. The performance test was carried between a Microsoft Windows 7 system on Intel chipset with 2GB RAM with XProber and another system with the same specification but without XProber. The web page load time is compared between the two systems, no web page time lags noticed in XProber. The percentage of pernicious threats that our XProber discovered are compared in **Fig. 7**.

5. CONCLUSION

Many websites are susceptible to XSS, SQLI, MFE and other attacks. Expreimental results prove that the proposed security solution is much effective. XSS, SQLI, MFE vulnerabilities exist in almost all platforms and the proposed solution works on any platform. It can been implemented on a platform independent browser and with a few modifications it can be used with other operating systems. An automata-based string analysis technique is presented for vulnerability signature

generation and vulnerability analysis. The analysis represents the attack pattern as a regular expression. Given a pre-scripted JSP program as an input: (1) It checks for the presence of vulnerability based on the given attack pattern, (2) It generates a PDA characterizing the set of all user inputs that may exploit the vulnerability. This solution can be further extended to cover other pernicious attacks and vulnerabilities. It can be applied as a common resolution which could be used in all the web browsers.

6. REFERENCES

Arulsuju, D., 2011. Hunting malicious attacks in social networks. Proceedings of the 3rd International Conference on Advanced Computing, Dec. 14-16, IEEE Xplore Press, Chennai, pp: 13-17. DOI: 10.1109/ICoAC.2011.6165172

Blanc, G., R. Ando and Y. Kadobayashi, 2011. Term-Rewriting Deobfuscation for Static Client-Side Scripting Malware Detection. Proceedings of the 4th IFIP International Conference on New Technologies, Mobility and Security, Feb. 7-10, IEEE Xplore Press, Paris, pp: 1-6. DOI: 10.1109/NTMS.2011.5720649

Hopcroft, J.E., R.M. Rotwani and J.D. Ullman, 2000. Introduction to automata theory, language and computability.

Huang, Y.W., S.K. Huang, T.P. Lin and C.H. Tsai, 2003. Web application security assessment by fault injection and behavior monitoring Proceedings of the 12th international conference on World Wide Web, (WW' 03), ACM, New York, pp: 148-159. DOI: 10.1145/775152.775174

Kadirvelu, S. and K. Arputharaj, 2011. Handling web and database requests using fuzzy rules for anomaly intrusion detection. J. Comput. Sci., 7: 255-261. DOI**:** 10.3844/jcssp.2011.255.261

Kiezun, A. and K. Jayaraman, 2009. Automatic creation of sql injection and xss attacks.

OWASP, 2007. Open web application security project the ten most critical web application security vulnerabilities.

Rattipong, P. and P. Bunyatnoparat, 2011. Protecting cookies from cross site scripting attacks using dynamic cookies rewriting technique. Proceedings of the 13th International Conference on Advanced Communication Technology, Feb. 13-16, IEEE Xplore Press, Seoul, pp: 1090-1094.

Sanctum Inc., 2004. Web application security testing-appscan 3.5.

Sipser, M., 1997. Introduction to the Theory of Computation. PWS Publishing, ISBN-10: 0-534-94728-X. Section 2.2: Pushdown Automata, pp: 101-114.

Teoh, K.K., T.S. Ong, P.W. Lim, R.P.Y. Liong and C.Y. Yap, 2009. Explorations on web usability. Am. J. Applied Sci., 6: 424-429. DOI: 10.3844/ajassp.2009.424.429

Tiwari, S. and R. Bansal, 2008. Optimized client side solution for cross site scripting. Proceedings of the 16th International Conference on Networks, Dec. 12-14, IEEE Xplore Press, New Delhi, pp: 1-4. DOI: 10.1109/ICON.2008.4772647

Yu, F., M. Alkhalaf and T. Bultan, 2010. Stranger: An automata-based string analysis tool for php. Lecture Notes Comput. Sci., 60185: 154-157. DOI: 10.1007/978-3-642-12002-2_13