AENSI Journals

Australian Journal of Basic and Applied Sciences

ISSN:1991-8178

Journal home page: www.ajbasweb.com

Secure IDS: Web Content Attestation and Trusted Computing Against Web Vulnerabilities

[1]I. Shahanaz Begum and [2]Dr. G. Geetharamani

[1]Assistant Professor, Department of IT, Bharathidasan Institute of Technology Campus, Anna University, Tiruchirappalli-620024, Tamilnadu, India.
[2]Professor and Head, Department of Mathematics, Bharathidasan Institute of Technology Campus, Anna University, ,Tiruchirappalli-620 024, Tamilnadu, India.

ARTICLE INFO

Article history:
Received 25 October 2014
Received in revised form 26 November 2014
Accepted 29 December 2014
Available online 15 January 2015

Keywords:
Web Vulnerabilities, Intrusion Detection System (IDS), Cross Site Scripting (XSS) Attack, Cross Site Request Forgery (CSRF) Attack, JBoss Web Server, Attestation, Trusted Computing.

ABSTRACT

Now-a-days, all organizations have grown-up and outspread their boundaries with the help of web information. Usually, the front-endapplications are deployed on web servers to implement the web services. Several web attacks have been introduced against the processing of the web servers. Web servers can be vulnerable to numerous web attacks like Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF), and more. To address these problems, an effective and secure Intrusion Detection System (IDS) is built to overwhelm the issues of web vulnerabilities. At first, the session is generated and the IDS monitors malicious activities of XSS and CSRF attacks. JBoss web server supports only limited and host-based policies that deny/allow access to the protected resources. These policies cannot report the suspicious activity and enhances the system protection. The URL is analyzed for session generation and it proceeds with the attestation and trusted computing. Attestation is the trustworthy mechanism, wherein the security is a major challenge associated with lack of trust and vulnerable to unauthorized access. To lead a trust environment, the proposed approach comprises of the enhanced access control mechanism to detect and prevent the web vulnerabilities. Finally, a monitoring application is developed to detect and filter all the vulnerabilities of the web application process. The experimental analysis exhibit higher throughput and lower latency than the existing techniques.

To Cite This Article: I. Shahanaz Begum and Dr. G. Geetharamani, Secure IDS: Web Content Attestation and Trusted Computing Against Web Vulnerabilities. ***Aust. J. Basic & Appl. Sci.,*** *9(2): 193-203, 2015*

INTRODUCTION

WEB services and internet applications have been the indispensable part of today's organization process scenario. In general, the front-end applications are deployed on web servers to implement the web services. Numerous web attacks have been introduced against the processing of the web servers. A web server can be vulnerable to the various kinds of web attacks such as:

- Buffer Overflow Attack
- Cross Site Scripting Attack
- Cross Site Request Forgery Attack
- Distributed Denial of Service Attack
- Session Hijacking Attack

As stated by the Open Web Application Security Project (OWASP), one of the most common application level attacks is the Cross Site Scripting (XSS) in which the hackers employ to snitch into web applications. XSS attack is a class of web application vulnerabilities. JavaScript that made by the attacker can be executed for attaining access privileges to the web page contents, session cookies, and a distinct type of information. These kinds of attacks may steal the sensitive data confidentiality, undermine the schemes of authorization, victimize users, and malign web sites. Fig.1 describes an overview of the cross site request forgery, where the cross site scripting helps the CSRF bypass POST protection token. The link from the token protected area to the web server can be labeled as compromises the entire web application or compromises the end user data and the associated functions.

Corresponding Author: Shahanaz Begum, Assistant Professor, Department of IT, Bharathidasan Institute of Technology Campus, Anna University, Tiruchirappalli-620024, Tamilnadu, India.
Tel: 0431-2401386, E-mail: shasmile23@gmail.com

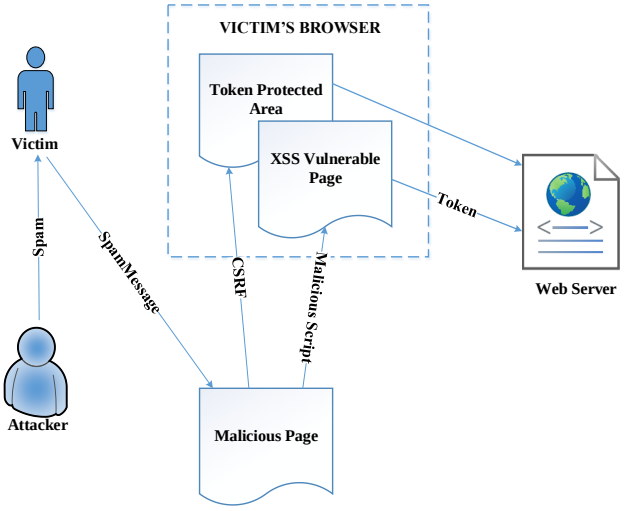

Fig.1: An illustration of the CSRF attack, where the XSS by passes CSRF protection token.

An effective and secure Intrusion Detection System (IDS) has to be built to address all the web vulnerabilities issues. IDS is a software application that monitors the activities of the network for analyzing the activities of the malicious websites. Many existing approaches detected the anomalies only by analyzing the source code of the web server. It requires the full application logic for the dynamic web-services. Web servers such as Apache, and JBoss support only the limited identity and host based policies. It can deny/allow access to the protected resources and these policies cannot report suspicious activity and does not improve system protection. Attestation and trusted computing is employed to offer trusted environment. Attestation is one of the components of trusted computing, which permits a program to authorize websites. Remote attestation for a single system make reliable statements about the software that run over the other system. Further, the authorization decisions are based on information that can be made by the remote party.

Therefore, the proposed scheme includes the enhanced access control mechanism in order to discover and prevent the web vulnerabilities. This scheme:

a) Builds secure access policy mechanism of tracking web user profiles, and parameters of an access request,
b) Monitors the execution of the requested operations,
c) Takes the countermeasures of the system's denied web request processes,
d) Updates the firewall policies, and
e) Tracks the web requests with abnormal parameters that violates the site policy.

The rest of this paper is organized as follows: Section 2 briefly overviews the related work based on the detection and prevention mechanisms of various attacks. In Section 3, the comprehensive details of the proposed methodology is presented. Section 4 describes the performance analysis of the proposed model with the existing technique. Section 5 summarizes with a brief conclusive remark and discussion on future works.

Related work:

This section describes the related work based on the detection and prevention of various attacks such as Cross Site Scripting (XSS), SQL injection attack, and proxy based Distributed Denial of Service (DDoS) attack. This review also describes the securing of the Java application from XSS and other attacks. *Stephan et al* introduced a passive detection system for identifying the successful Cross Site Scripting (XSS) attacks. The accuracy of this detection approach was examined and also verified its detection capability based on a prototypical implementation. From 20 popular web applications, a dataset of HTTP request/response was compiled with the XSS. This can be provided to the Internet Service Provider (ISP) in order to handle the XSS problem at the ground level itself (Stephen, *2011*). *Saxena et al* discussed the sanitization abstractions that were offered by the various frameworks of the web application development. This framework often failed to address the XSS sanitization subtleties. The XSS abstractions systematically were evaluated with commercially-used web frameworks. The parsing of the web browser and the web content transformation was complex in this model (Weinberger,*2011*).

Shar et al presented a code-auditing scheme to recover the defence model that was implemented in the program source code. This approach suggested the guidelines against XSS attacks to check the adequacy of the recovered model. The auditors may face the problems in upholding Tainted Information Flow Graphs (TIFGs), since the number of TIFGs grow as the number of HTML output grow in web applications. This method further validates the possibility of large and complex applications (Shar, 2012). *Subashini et al* surveyed the various security risks, which pose a threat to the cloud environment. Due to the nature of the service delivery models of

a cloud, this survey was more specific to the several security issues. It derives a framework in order to provide a practical solution than the meta-data information based security (Subashini, 2011).

Chen et al proposed nonce-spaces technique to prevent the XSS attacks. The randomized(X)HTML tags were employed for identifying and annotating the untrusted content. Nonce spaces-aware client could prevent all the attacks by prohibiting the policies. It also avoids node-splitting attacks for the noncespaces-unaware clients (Van Gundy, 2012). *Miller et al* developed a multi-layer framework for the web client security. Based on the mobile code instrumentation, the exploitable security vulnerabilities were isolated. Also, the runtime policies against malicious code constructs were enforced. This framework can be greatly enhanced further by IE specific JavaScript constructs like ActiveX controls (Ofuonye, 2013). *Laila et al* introduced a framework based on the detection of anomaly and misuse to discover SQL Infection Attack (IA). A profile for web application was created using the Web Anomaly Misuse Intrusion Detection (WAMID) framework. It represents the normal behavior of the application users with respect to the SQL queries. This framework can be extended to include the detection against cross site scripting attacks (Salama, 2012).

Rim et al proposed a novel methodology for discovering the web application vulnerabilities automatically. This method highlights the scenarios of different potential attacks. It comprises the exploitation of various successive vulnerabilities (Akrout, 2014). *Le et al* introduced the current and potential threats, which can directly affect the network operation. Cryptography and Intrusion Detection Systems (IDS) were reviewed for providing whole security from internal and external attackers. IDS was built to secure the operation of IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) (Le, 2012). *John et al* described a novel integrated and automated Vulnerability Analysis and Mitigation Solution as a Service (VAM-aaS). A static vulnerability analysis tool was developed for locating the possible matches in a target system. A set of seven open source web applications were employed to validate this approach (Almorsy, 2012).

Neha et al presented a mechanism for securing the Java web applications from XSS. This scheme applied a framework based on the approach of pattern matching. The malicious requests and responses were recorded using the attack recorder and response rejecter. This framework can be extended to enhance the XSS attack detection and prevention process (Mahapatra, 2014). *Priyanka et al* presented a new server-side defense scheme to resist the web proxy based distributed Denial of Service (DoS) attack. The benefits behind this approach were: (a) detection performance (b) independent of the traffic intensity, and (c) realizes the early detection. This can be enhanced further with dynamic activity with the mobile movement and the activities of the graphical interface for distance and query (Priyanka, 2014). *Shar et al* introduced a two-phase approach to discover and eliminate the potential XSS vulnerabilities in server programs. The discovery phase of XSSV identifies the program source code using static analysis. The elimination phase determined the context of each user input referenced in the well-known potential XSSVs (Shar, 2012).

Hydara et al reviewed 115 articles that relate to the research on cross site scripting. Several solutions or techniques were identified in this survey. Most of them were focused on the types of XSS and detection of the vulnerabilities. It can further ensure the protection from attackers and minimize the XSS incidents (Hydara, 2014). *Sharma et al* introduced a method for preventing SQL injection and reflected XSS attacks in PHP web applications. Along with a checking pattern in input fields, this method uses the logic of constructing a query model. This model can be further enhanced to prevent other types of XSS attacks, say Stored XSS. The solution can also be improved further for preventing other web attacks like file inclusion and several other vulnerabilities that classified by the Open Web Application Security Project (OWASP) (Sharma, 2012). *Shar et al* aimed to offer the solution to taint analyzers using a set of static code attributes. SQLI and XSS vulnerabilities were avoided from the observations of input sanitization code, which could be implemented in web applications. This accomplished an average result of the recall and false alarm rate in predicting the SQLI and XSS vulnerabilities (Shar, 2013).

Vijay et al presented a new trust enhanced security methodology for cloud services. Using trusted attestation techniques, the security attacks in cloud infra structures were discovered and prevented. The security properties have been employed for detecting and minimizing attacks among the cloud tenants. It maximizes the assurance of the tenant virtual machine transaction (Varadharajan, 2014). *Salas et al* described two security testing techniques, namely Penetration Testing and Fault Injection for emulating XSS attacks against web services. This technology was integrated with WS-Security (WSS) and security tokens. It could identify the sender and also assures the legitimate access control and enhances the vulnerability detection (Salas, 2014). *Tang et al* investigated the vulnerabilities of feedback control based internet services to the Low-Rate Denial-of-Service (LRDoS) attacks. The cost of this approach can be further discussed to mitigate the damage of the LRDoS attack (Tang, 2014). *Razzaq et al* proposed a new approach for semantic technologies in web application. A synergy was established between the web semantics and information security technologies. The performance was improved based upon a prototype attack detection scheme (Razzaq, 2014).

Australian Journal of Basic and Applied Sciences, 9(2) February 2015, Pages: 193-203

Web content attestation and trusted computing against web security vulnerabilities:

In a web application, security is well-defined by the roles that permit access to content by a URL pattern. This identifies the protected content and declares the set of information using the security constraint element. To secure the web content, one or more elements of the web resource collection are used. Each and every element consists of an optional series of URL pattern elements followed by the elements of the HTTP method. For configuring the authentication method, the optional login-config element is used. A custom Java Server Page (JSP) or HTML page is defined by enabling form-based authentication to log in. If an error occurs during log in, then it is directed to a separate page. The login or error pages are:

```
<login-config>
    <auth-method>FORM</ auth-method>
    <form-login-config>
     <form-login-page>/login.html</form-login-page>
     <form-error-page>/error.html</form-error-page>
    </form-login-config>
</login-config>
```

The JBoss web server maintains a session pool so that the information about authentication does not require to be present for each request. Fig.2 depicts the overall flow diagram of the proposed methodology. The three major components of an overall proposed methodology are described as follows:

1. Session Generation
2. Request Analysis
3. Trusted Web Hashing

3.1 Session Generation:

A website is a group of forms and links, where an HTTP request is generated for each time a session is requested. Depending on the website configuration, the request is created and the method is utilized by the user for requesting support by:

- Choosing a name from logged-in representative list
- Entering a unique session key
- Submitting an issue

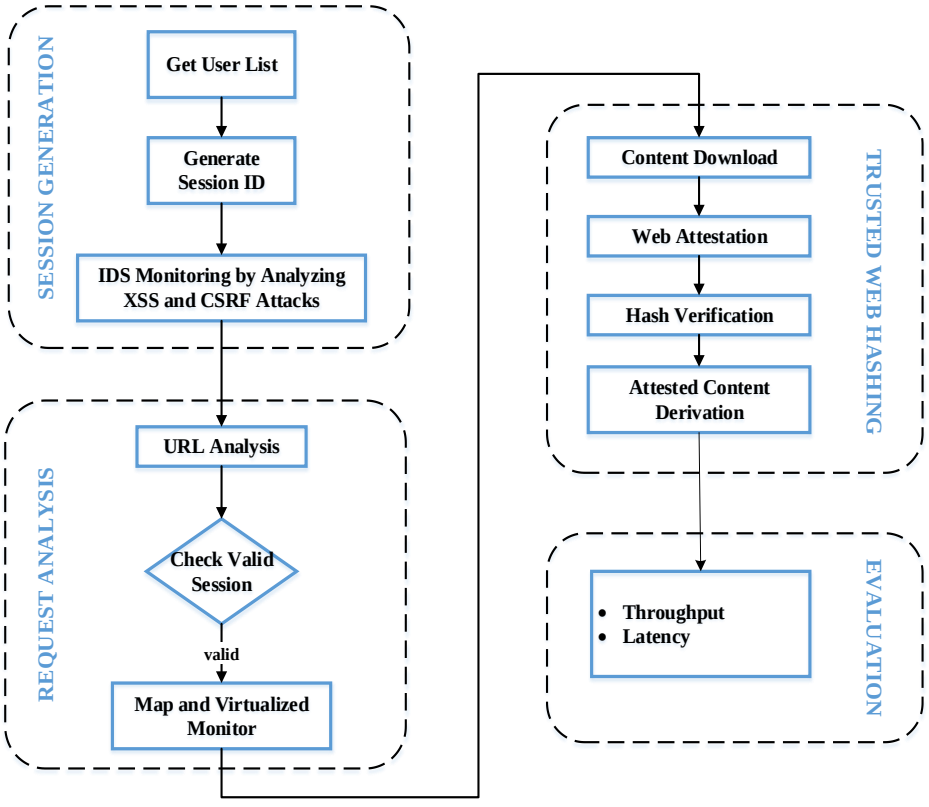

Fig. 2: An overall flow diagram of the proposed web content attestation and trusted computing against the web security vulnerability.

Australian Journal of Basic and Applied Sciences, 9(2) February 2015, Pages: 193-203

3.1.1 Session ID Generation:

Each session is discovered by a session ID, which is a 16 byte string established from the random values. By default, these values are retrieved and hashed with the Digital Signature Algorithm-Secure Hash Algorithm (DSA-SHA512), which is a cryptographic token interface standard. The DSA-SHA512 mechanism estimates the whole specification of the DSA, comprising the hashing with SHA. It does not have a parameter and its key and data length constraints are shown in table I. At session ID generation, the checks are performed for ensuring that the ID generated is unique. The session ID is allocated as part of a cookie once verified and further it is returned to the user. This cookie is used for identifying the user session and is expected in subsequent user requests.

Table I: Key and Data Length of DSA-SHA512.

Function	Type of Key	Length of Input	Length of Output
c_sign	DSA private key	Any	40
c_verify	DSA public key	Any, 40^2	N/A

A name value pair along with various optional attributes is the cookie passed to the user. Cookies are kept for several information tracking purposes and are stored as the text files on the user side. JSP supports HTTP cookies and the server script send a set of cookies to the browser. Cookies are set usually in the header of HTTP, where Set-Cookie header includes a name value pair, a GMT date, a path, and a domain. The name and a value will be URL encoded. The cookie is configured to be non- persistent, whereas on the user side, it will be neglected while the web browser exits. Nevertheless, on the server side, sessions terminate after a few seconds of inactivity in which time and session objects and their information are neglected.

3.1.2 Intrusion Detection System Monitoring:

For networks, the intrusion detection is the kind of security management system. An IDS collects and examines the information from several areas within a network. This identifies the possible security breaches that consist of both intrusions and misuse. The former denotes the attacks from outside the organization, whereas the latter is the attacks within the organization. The functions of IDS are:

- Monitoring and analyzing both the activities of user and system
- Analyzing the web vulnerabilities
- Assessing integrity of the file
- Recognize the attacks
- Abnormal activity pattern analysis

a) Analysis of XSS and CSRF attacks:

The Cross Site Scripting (XSS) attacks are the type of injection, where as the Cross Site Request Forgery (CSRF) is a type of malicious exploit of a website. The unauthorized commands are transmitted from a user in which the website trusts. Unlike XSS, which exploits the trust a customer has for a specific site, CSRF exploits the trust in which a site has in the customer's browser. The disclosure of the customer's session cookie and permitting an attacker to hijack the customer's session are the most severe attacks in XSS. The CSRF can be prevented with the token generation issue. A session and token can be created once the session is null and it is redirected to login page of JSP. Further, it stores the token in the hidden field. If the token matches, then the action class is executed. Otherwise, the new token is created for the similar session and redirect to the login page of JSP. The prevention rules for cross site scripting are mentioned as follows:

Prevention Rules for XSS
Rule #0 : Never insert untrusted data except in allowed locations
Rule #1 : HTML escape before inserting untrusted data into HTML element content
Rule #2 : Attribute escape before inserting untrusted data into HTML common attributes
Rule #3 : JavaScript escape before inserting untrusted data into JavaScript data values
Rule #4 : CSS escape and strictly validate before inserting untrusted data into HTML style property values
Rule #5 : URL escape before inserting untrusted data into HTML URL parameter values
Rule #6 : Sanitize HTML markup with a library designed for the job
Rule #7 : Prevent DOM-based XSS

XSS flaws can be difficult to discover and eliminate from a web application. The best method for identifying the flaws is to implement a security review of the code. It searches for all input from an HTTP request and possibly provides the HTML output. The attacks can be protected by the Open Web Application Security Project-Enterprise Security API (OWASP-ESAPI). For high performance encoding, the OWASP Java Encoder Project is used. The main purpose of the ESAPI is to offer a simple interface, which provides all the functions of security. Its architecture is very simple with group of classes and the basic design includes:

- A set of security control interfaces

Australian Journal of Basic and Applied Sciences, 9(2) February 2015, Pages: 193-203

- A reference implementation for each security control
- Own implementation for each security control.

3.1.3 URL Analysis:

The URL is verified after monitoring the attack analysis and are explained in the pseudo code given below. The URL request U_i and the status of the request are the input and output respectively. Initially, the URL from the i^{th} user is parsed and then predicts the session ID. Consequently, the predicted session ID PS_i checks for validation. If the PS_i is valid, then the request is mapped to the server and the request status is said to be accepted. Otherwise, the URL request is rejected and then return the status of the request.

Pseudocode for URL Verification
Input : URL Request U_i
Output : Request Status S_i
1. Get URL from i^{th} user
2. Parsing URL predict session ID
3. $PS_i \leftarrow U_i$
4. if(IsValid(PS_i) = true)
a. Mapping Request to server
b. $S_i \leftarrow$ Accepted
5. end if
6. else
a. Reject URL request $S_i \leftarrow$ Rejected
7. return S_i

3.2 Web Content Attestation and Trusted Computing:

For web attested content, an overview of the system architecture is shown in the fig.3. An attested timestamp is provided by the time server to the JBoss web server. This offers the integrity measured content to the clients. From the time server, the current time can be directly verified by the web browser. The core elements that are defined in this system are JBoss web server, a time server, and a web browser. A JBoss web server creates static or dynamic web content and delivers the content integrity proof to the clients. A time server provides an attestation of the current time to the web server. Whereas, a web browser added an extension to verify the proofs accomplished from the web server. This can query the time server securely to verify its attestation independently. The operations of the system are described as:

- From the JBoss web server, a client requests a page and a URL to the content attestation.
- From the time server, the server hashes the web contents in the Trusted Platform Module (TPM).
- A database is concatenated along with a cryptographic proof system.
- A system attestation is generated by the resulting hash as a challenge to the TPM.
- The client obtains and authenticates attestation from the web server and the time server.
- The root of the cryptographic proof system is computed.

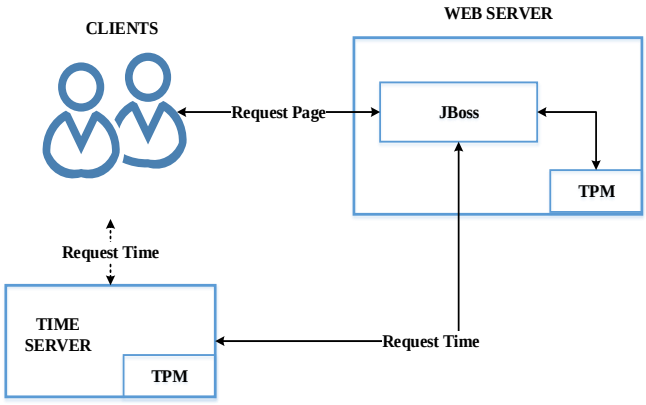

Fig. 3: An overview of the system architecture for web attested content.

3.2.1 Hash Verification:

Each user of a web application should install the browser extension in their web browser. Before running, it checks whether the applications are properly signed or not. The following pseudo code explains the web content verification. The site public key is embedded in the server's verification certificate hosting the web application. Finally, the users will visit the correct website via https.

Pseudocode for Web Content Verification
Procedure : Process-Response (URL, cert, response)
Input : URL, cert
Output : Hash Verification
// URL --- Requested URL
// CERT--- Server's Verification Certificate
i. IfCERT contains attribute pub_key then
a. pk ← cert.pub_key
b. sig ← response.header["Content_Signature"]
c. if not VERIFYSIG(pk, response,sig)then
i. return ABORT
ii. ifURL contains parameter "Content_Hash=h"then
a. if hash(response) ≠ h then
i. return ABORT
iii. return PASS //Attested Web Content

In attested content derivation, the client can view the vulnerability free web content from the JBoss web server through the trusted computing. This process is mainly focused for avoiding the XSS attacks.

Performance analysis:

In this section, we empirically analysis the performance of the proposed system that presented in the preceding section. The number of vulnerabilities, throughput, latency, processing time in terms of the fault types, number of users, and number of URLs are compared with the existing system in the trusted environment. Table II shows the information of parameters used in the proposed application, where all the tests were performed.

Table II: Parameters used in the Proposed Web Application

PARAMETERS	VALUE
Number of URLs	50
Web Server	JBoss
API	OWASP-ESAPI
Processor	Dual Core
RAM	2 GB

Number of URLs vs. Processing Time:

The processing time in the analysis of a hashing-form web content is the time to complete a prescribed procedure. The output is increased by decreasing the processing time. Fig.4 shows better results of the processing time for the proposed hashing-form web content than the existing unaltered web service (Moyer, 2012). The processing time of hashing-form web content increases gradually for the number of URLs.

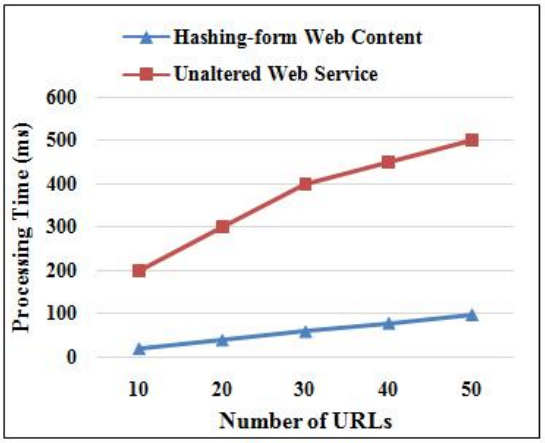

Fig. 4: The results of processing time for the proposed hashing-form web content and the existing unaltered web service with respect to the number of URLs.

Fault Types vs. Number of Vulnerabilities:

The major deviation comes from the number of vulnerabilities detected. Fig.5 illustrates the result of a number of vulnerabilities for the fault types such as WPFV, MIFS, MFC, MIEB, and WLEC. The comparison is shown for the vulnerabilities in the field of strong type JSP and the strong type VB (Fonseca, 2014). For the fault type WPFC and MIFS, the strong type JSP has the value of 0, which denotes vulnerability free content.

The fault type MFC, MIEB, and WLEC of strong type JSP has the value of 1, where the vulnerabilities of existing strong type VB is higher than the proposed strong type JSP.

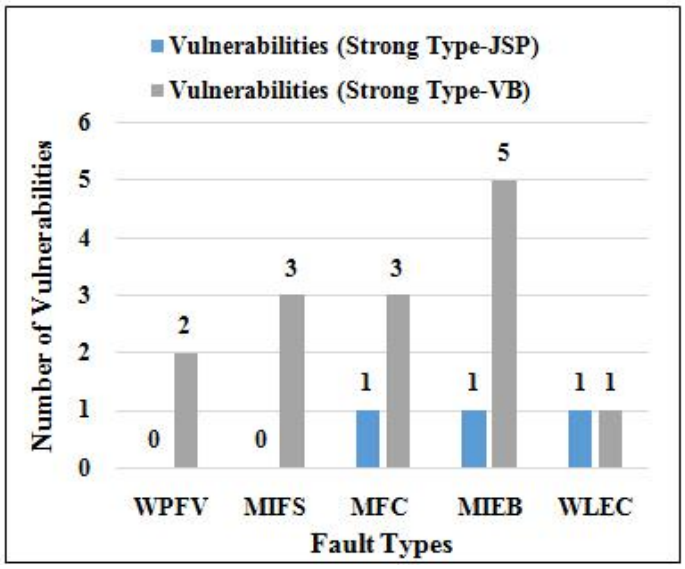

Fig. 5: The results of strong typed, web application vulnerabilities for the proposed JBoss and the existing VB with respect to the fault type.

Processing Time of DSA-SHA512:

Fig.6 depicts the result of the processing time for the proposed DSA-SHA512 and the existing SHA algorithm. The processing time of proposed DSA-SHA512 gradually increases for the number of URLs. When compared to the SHA algorithm, DSA-SHA512 has the lower processing time.

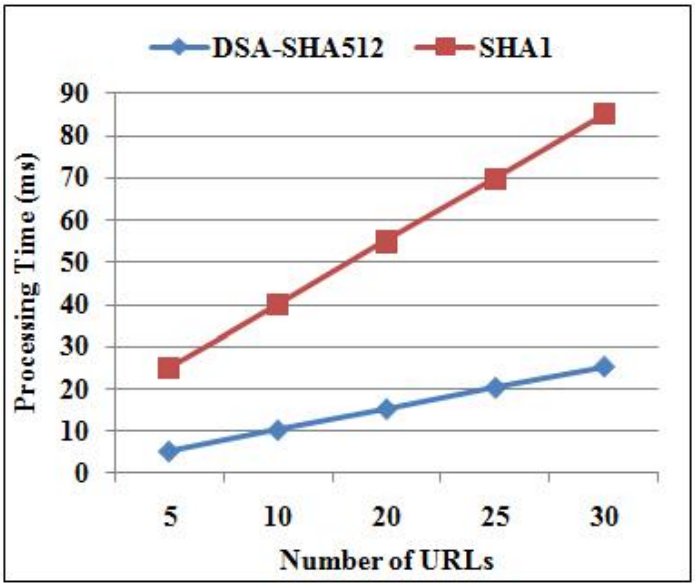

Fig. 6: The result of processing time for the proposed DSA-SHA512 and the existing SHA with respect to the number of URLs.

Latency:

Latency measures how much time it takes data packets to travel between one point to another point with no delay. It is a synonym for delay and the network latency includes propagation, transmission, router, and other storage delays. In this analysis, the performance of a web application developed using our JBoss framework for similar applications implemented using the other existing frameworks.

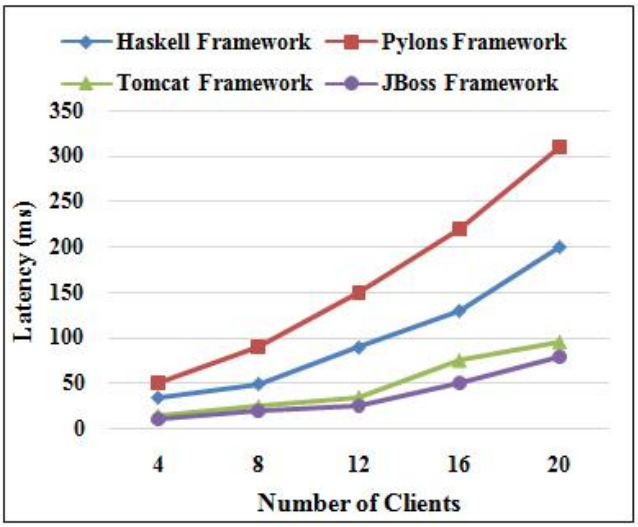

Fig. 7: The performance of latency for the existing Haskell, Pylons, Tomcat, and the proposed JBoss-based web applications.

Fig. 7 presents the averaged latency for each framework tested. The number of simultaneous clients issuing demands is varied. Therefore, the average response latency in milliseconds is recorded. JBoss framework performed competitively compared to Pylons, Tomcat, and Haskell frameworks (Robertson, 2009). With the number of clients, the latency plot shows that JBoss framework scales significantly better than the other frameworks.

Throughput:

Throughput is the number of data transfer from one location to another location in a specified quantity of time. The data transfer rate is used for measuring the performance of disk drives and networks in relation to the throughput.

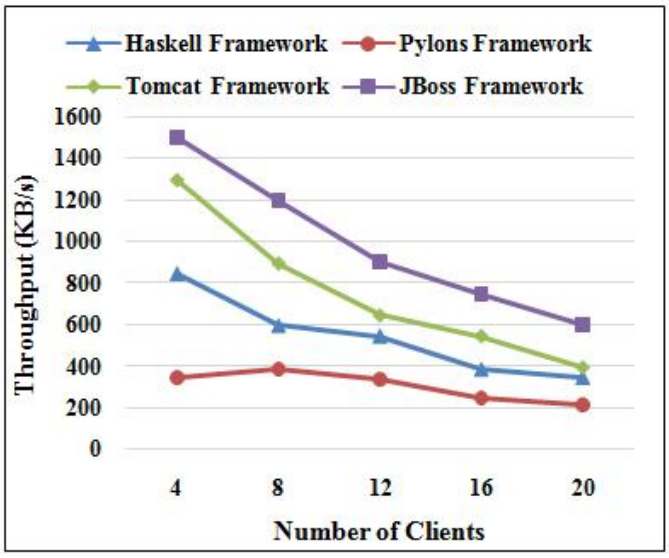

Fig. 8: The performance of throughput for the existing Haskell, Pylons, Tomcat, and the proposed JBoss-based web applications

Fig. 8 demonstrates the throughput for each framework tested. The aggregate throughput in kilobytes is recorded as the number of simultaneous clients issuing different demands. JBoss framework exhibit higher throughput compared to Pylons, Tomcat, and Haskell frameworks (Robertson, 2009). With the number of clients, the throughput plot shows that JBoss framework scales decreases gradually.

Conclusion and future work:

Many web attacks have been presented for the web server processing that can be vulnerable to numerous web attacks like Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF), and more. To address the problems of web vulnerabilities, an effective and secure Intrusion Detection System (IDS) is built. Initially, it generates the session and monitors the analysis of attacks. A JBoss framework is developed for web applications to construct the vulnerability free content. Subsequently, the URL is analyzed to verify its request status and proceeds with the attestation and trusted computing. To lead a trusted environment, the proposed approach includes the improved access control mechanism for detecting and preventing the web vulnerabilities. At last, a monitoring application is developed to filter all the vulnerabilities of the web application process. The experimental analysis exhibits higher throughput and lower latency than the existing techniques in terms of number of clients.

In future, the data mining techniques can be integrated to prevent the web vulnerabilities. It will identify the false positives using the machine learning classifiers and also discovers the vulnerabilities with less false positives. Moreover, the semi-supervised learning methods will also extend with the proposed system for effective web vulnerability prediction.

REFERENCES

Stephen, M.J., 2011. "Prevention of cross site scripting with E-Guard algorithm," *International Journal of Computer Applications*, 22(5).

Weinberger, J., 2011. "A systematic analysis of xss sanitization in web application frameworks," in *Computer Security–ESORICS 2011*, ed: Springer, pp: 150-171.

Shar, L.K. and H.B.K. Tan, 2012. "Auditing the XSS defence features implemented in web application programs," *IET Software*, 6(4): 377-390.

Subashini, S. and V. Kavitha, 2011. "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, 34(1): 1-11.

Van Gundy, M. and H. Chen, 2012. "Noncespaces: Using randomization to defeat cross-site scripting attacks," *computers & security*, 31(4): 612-628.

Ofuonye, E. and J. Miller, 2013. "Securing web-clients with instrumented code and dynamic runtime monitoring," *Journal of Systems and Software*, 86(6): 1689-1711.

Salama, S.E., 2012. "Web anomaly misuse intrusion detection framework for SQL injection detection," *International Journal of Advanced Computer Science & Applications*,3(3).

Akrout, R., 2014. "An automated black box approach for web vulnerability identification and attack scenario generation," *Journal of the Brazilian Computer Society*, 20(1): 1-16.

Le, A., 2012. "6LoWPAN: a study on QoS security threats and countermeasures using intrusion detection system approach," *International Journal of Communication Systems*, 25(9): 1189-1212.

Almorsy, M., 2012. 2012. "VAM-aaS: online cloud services security vulnerability analysis and mitigation-as-a-service," in *Web Information Systems Engineering-WISE 2012*, ed: Springer, pp: 411-425.

Mahapatra, R., 2014. "A Pattern Based Approach to Secure Web Applications from XSS Attacks."

Priyanka, S., 2014. "Resisting Web Proxy-Based HTTP Attacks Through Cross-Site Scripting," *International Journal of Computer Science*

Shar, L.K. and H.B.K. Tan, 2012. "Automated removal of cross site scripting vulnerabilities in web applications," *Information and Software Technology*, 54(5): 467-478.

Hydara, I., 2014. "Current State of research on cross-site scripting (XSS)–A systematic literature review," *Information and Software Technology*.

Sharma, P., 2012. "Integrated approach to prevent SQL injection attack and reflected cross site scripting attack," *International Journal of System Assurance Engineering and Management*, 3(4): 343-351.

Shar, L.K. and H.B.K. Tan, 2013. "Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns," *Information and Software Technology*, 55(10): 1767-1780.

Varadharajan, V. and U. Tupakula, 2014. "Counteracting security attacks in virtual machines in the cloud using property based attestation," *Journal of Network and Computer Applications*, 40: 31-45.

Salas, M. and E. Martins, 2014. "Security Testing Methodology for Vulnerabilities Detection of XSS in Web Services and WS-Security," *Electronic Notes in Theoretical Computer Science*, 302: 133-154.

Tang, Y., 2014. "Modeling the vulnerability of feedback-control based Internet services to low-rate DoS attacks," *IEEE Transactions on Information Forensics and Security*.

Razzaq, A., 2014. "Ontology for attack detection: An intelligent approach to web application security," *Computers & Security*, 45: 124-146.

Moyer, T., 2012. "Scalable web content attestation," *Computers, IEEE Transactions on*, 61(5): 686-699.

Fonseca, J., 2014. "Analysis of field data on web security vulnerabilities," *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*.

Robertson, W.K. and G. Vigna, 2009. "Static Enforcement of Web Application Integrity Through Strong Typing," in *USENIX Security Symposium*, pp: 283-298.