Volume 5, Issue 8, August 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Comparative Study of Detect IVAs over the Web Application

Chokhawala Kirit I.
Research Scholar
Mewar University, Chittorgarh, Rajasthan India

Dr. A. R. Patel
Dept. of Computer Sceince
H.N.G.University, Patan, Gujarat, India

Abstract- As daily use of Internet is expanding exponentially and access of Internet become more prevalent in our daily life but at the same time web application are becoming most attractive targets for hacker and cyber criminals. The paper identifies vulnerability attacks caused due to inputs performed by a user which are not properly validated across the web application. The existing IDS designed for validation vulnerability attacks are language reliable. Survey paper present a proposed IDS concept which is not language reliant i.e. it is designed for any web application developed with the support of PHP, Java, Dotnet etc. Such concept of IDS is helpful to detect input validation weaknesses like SQL injection attacks, Cross site scripting attacks, command injection attacks and directory traversal attacks; these were not detected in the extant IDS.

Keywords- SQL injection attacks, Cross site scripting attacks, command injection attacks and directory traversal attacks
General Terms- Web Security, Web Application, Input Validation Attacks

I. INTRODUCTION

Web Application are most widely used for providing service to the user like online shopping, online reservation, and many more application which is designed in perspective of user. So the web application is popular attacks target due to time and financial constraints, limited knowledge of the programming, limited knowledge security awareness, misconfiguration that is meant lack of awareness of the security configuration deployment on the part of the programmer. With the aid of the input validation attacks attacker can steal the confidential data which decrease the market values of the organization. Web applications generally use TCP port for the communication with server. This communication is not protected by the IVAs [1].The Open Web Security Project (OWASP)[2] Vulnerabilities: SQL Injection Attack, Cross Site Scripting Attack, Directory Attack, Command Injection Attack are input validation attacks. Existing invasion detection system is designed [4, 5, 6, 7 and 8] in a manner that detects SQL injection attacks; XSS attacks however they do not identify the directory traversal attacks and command injection attacks. Such IDS be projected are language definite for e.g. system designed for JAVA based web application; system designed for PHP based web application; system designed for .net based web application.

In this paper invasion detection system approach identify SQL Injection attacks, Cross- Site Scripting attacks, Directory Traversal attacks, Command Injection attacks, and is not language specific. This IDS approach require only window environment for detecting IVA over the internet. It requires the web reference for data to analyze the attacks if it detects several kind of IVA.The Proposed IDS concept can be further more efficient for getting several case of Input validation attacks and with the assistance of this concept server administrator can take efficient action against these attempts. Thus in this way this concept would reduce the analysis time and likewise increase the effectiveness of the organization. This research paper is divided into three sections. First section describes the description of input validation attacks. Second section describes literature survey. Third section describes comparison analysis with existing IDS and Proposed IDS. Rest of the paper describes conclusion and future work.

II. SECTION-I INPUT VALIDATION ATTACKS

The Input Validation Attacks (IVAs) attempt to submit data which the web application does not expect to receive, that causes very serious consequences like session hijack, SQL poisoning, source code disclosure, path traversal etc. Input validation is a security issue if an attacker discovers that the application makes unfounded assumptions about the type, length, format, or range of input data. The attacker can then supply carefully crafted input that compromises the application. When network and host level entry points are fully secured; the public interfaces exposed by the application become the only source of attack. The input to the application is a means to both test the system and a way to execute code on an attacker's behalf. If the applications blindly trust input. It may be susceptible to the following:

i. Buffer Overflows Attacks

Buffer overflows vulnerabilities are capable to direct denial of service attacks or code injection. A process crash is caused due to denial of service attack. Attacker injects code to modify the execution address of a program is code injection attack. The threat of buffer overflows attack is less likely to be on a managed code while a unmanaged code are more likely a subject of concern to avoid denial of service attacks.

ii. SQL Injection Attacks

SQL injection attack utilizes weaknesses in input validation runs uninformed commands in the database. It can happen when the request to a application utilizes input to create active SQL statements to contact the database. It can also happen if system code utilizes stored procedures with the purpose are passed as strings that contain unprocessed user input. Using the SQL injection attack, the assailant can perform uninformed commands in the database. The issue is enhancing as the executing command utilizes an over-privileged description to append to a database. Here in this occurrence it is possible to occupy database server to run operating system's command and potential cooperation with other servers, also it is able to destroy, retrieve and manipulate data.

iii. Cross-Site Scripting Attacks

XSS attack originates by an uninformed code to execute in user's web browser though the web browser is attached to a faithful Web site. These XSS attacks aim the application's client whereas application itself is used as the medium used for the assault. User's browser downloads the script code from a trusted Web site; the browser is uninformed of the fact that the code is not legitimate. Internet Explorer safety measures zones supply no protection regarding the script code attack. A user's validation cookies are usually intended by attack as the attacker code has right to use the cookies related with the object Web site.

Example of Cross-Site Scripting:

Initially the attacker must persuade the user to connect to a cautiously prepared hyperlink, for e.g., by grouping a link in an email forwarded by the user or else by joining a malicious link to a post by a newsgroup. The linkage points to open to attack page in the application that becomes invalidated to the web browser within output stream of HTML. For example, consider following two links.

Below is a link depicting legitimate user sign on to a web app:

http://www.browsewebapplication.com/signon.aspx?username=abc

Here is a malicious link:

www.browsewebapplication.com/signon.aspx?username=<scrip t>alert ('hacker code') </script>

As the Web application takes the query sequence, fails to properly validate it, and after that leads to the web browser, causing the script code to run in the browser. The preceding example displays an undamaging pop-up message. As the script is appropriate, the assailant can slowly create the user's validation cookie, moreover post this on his site, and consequently generate a call to the objective Web site as the legitimated user.

iv. Canonicalization

Unusual structure of input that determine to the similar regular name (the fundamental name), be referred to canonicalization. Rule code is mainly vulnerable to canonicalization concern if it creates protection assessment support on the name of a resource that is agreed by a program as input. Files, directory paths, and URLs are reserve type that are vulnerable to canonicalization for the reason that in each case many different ways are present to correspond to the same name. Sometimes names of file can also be challenging.

v. Command Injection Attack

Executing System/Operating System Commands through the application are the sources of Command injection Attacks. Java has the provision to execute system commands with the Runtime. Exec () method. Improper validation of the exec method arguments leads to vulnerability in the source code and attacker can pass malicious commands through the exec argument and gain control of the system. The following sample code illustrates the Command injection vulnerability [3].

```
Class Exec class
{public static void main (String args [])
{ Runtime rt = Runtime.getRuntime ();
Process proc = rt.exec ("cmd.exe /C") ;}}
```

vi. Directory Traversal Attack

This attack is used to obtain passwords. Most of the systems do not store passwords in plain text form or in encrypted form. Usually encrypted form passwords are avoided because a conceded key which leads to concede of all passwords in data base store. A key lost means that password is invalidated. With the aid of this directory traversal attacks attacker uses a program to iterate through all the words in a dictionary and computes the hash for each word. Thus the weak passwords such as "bob" can be cracked easily. Once attacker gets a list of password hashes, the directory traversal attack can be performed offline even without the interaction with the application. With the aid of directory traversal attack the assault get directory path of the web application.

III. SECTION-II LITERATURE SURVEY

Existing system in practice used in development and test time to prevent or detect input validation vulnerability attacks so as to improve programs for input validation vulnerability attacks can be reduced. Subsequent section depicts a study of those techniques and also compares with our approach.

i. Protect Web Application using Positive Tainting and Syntax-Aware Evaluation[4]

Halfond, William GJ, Alessandro Orso, and Pete Manolios proposed an automatic approach for prevention and

detection of SQLIAs. This approach uses four terms to detect SQLIA which are Syntax-aware evaluation of queries string, Exact and proficient taint propagation, Positive tainting and nominal deployment condition. This approach is a SQL Injection attacks detection system only and provides language support for Java.

ii. VIPER used for SQL Injection Attacks Detection [5]

Ciampa, A., Visaggio, C. A., & Di Penta, proposed a heuristic- based approach for detecting SQL-injection vulnerabilities in Web applications. In this technique, SQL Injection attack is detected by using heuristic rules which are intended to increase the possibility of work out some problem based approach. It basically performs panetration testing of the web application. This approach analyzes the web request for detrmining hyperlinks configuration along with input supplied by a client and generates a error message, if some type of SQL insertion occurres. This tool uses automatic technique to recognize SQL Injection vulnerability proving support for any language.

iii. AMNESIA: Analysis and Monitering for Neutralizing SQLI Attacks[6]

AMNESIA technique is a runtime SQL Iinjection attack detection tool over web application. These technique operates on static approach likewise for runtime monitoring. Before executeion on a database server, it detects structural query injected by a form support approach. This technique has two partition one is a static part which is used to put together a legel queries using program analysis whereas another is in dynamic part which dynamically generates the queries automatically next to statically build queries using monitoring during runtime. If queries resist the approach then the technique prevents the operation of the queries on a database server. Steps involved for preventing injection by using this tool (a)Identify the hotspot, (b)Build SQL-query models, (c)Instrument application, (d) Runtime monitoring.

iv. ARDILLA Tool[7]

Kieyzun, A., Guo, P. J., Jayaraman, K., & Ernst developed a method to recognize the SQL Injection attack and XSS attack vulnerabilities. This technique works on unmodified existing code, produce concrete input that depict vulnerabilities and work before software be deployed. This is a programmed tool designed for creating attacks. It is white box testing tool in a conduct that it requires source code of the application. Basis of the tool is the input generation and alteration, also taint spread to expose variants of an execution that exploit vulnerability.

v. D-WAV: A Web Application Vulnerabilities Detection Tool Using Characteristics of Web Form[8]

This approach proposed a novel programmed dynamic testing technique of the web form characteristics based on its analysis result. As the web forms are analysed, they generate test suites to detect vulnerability on web application to handle security issues caused by malicious data input. This technique detects SQL Injection and XSS vulnerabilities over web application.

vi. On Predictive Errors of SQL Injection Attack Detected by the Feature of the Single Character[9]

Takeshi Matsuda,Daiki Koizumi,Michio Sonoda,Shigeichi Hirasawa proposed a technique that uses a sigmoid function for detecting SQL injection attacks. An algorithm for detection of SQL Injection Attack supported by using a single character is proposed. When the SQL character string is the SQL Injection, it describe an attack character string. This approach used to decrease the analytical error in SQL Injection attack detection.

vii. Obfuscation-based Analysis of SQL Injection Attacks[10]

Halder, Raju, and Agostino Cortesi proposed Obfuscation-based approach of SQLIA. They implemented collective composition of static testing and dynamic testing which is based on the obfuscation and de-obfuscation of SQL commands. SQL Injection attacks can be simply detected for the reason that dynamic verification is conceded out on obfuscated queries, at atomic formula level only those atomic formulas which are tagged as vulnerable. And this come close to find the source foundation of SQL Injection attacks in dynamic query creation.

IV. SECTION-III COMPARISON ANALYSIS AND PROPOSED IDS

This section shows that why our proposed concept is advanced to previous IDS to detect input validation attacks in web application. Table 1 gives the comparative view of the existing IDS with our proposed IDS concept.

Table 1: Comparison with existing IDS

Detection IDS	Detect IVAs	Languages
WASP[4]	SQL Injection	JAVA supported web application
VIPER[5]	SQL Injection	ANY
AMNESIA[6]	SQL Injection	JAVA supported web application
ARDILLA[7]	SQL Injection, XSS attacks	PHP supported web application
D-WAV[8]	SQL Injection, XSS attacks	ANY
Our Proposed IDS	SQL Injection, XSS, Command Injection, Directory Traversal attacks	ANY

In this analysis table our propose IDS will take hold for all web applications that are developed using several languages like PHP, java, Dot Net etc. The proposed System will also hold the XSS attacks, SQL Injection attacks, Directory Traversal attacks, Command Injection attacks. Previous IDS system WASP [4] does not identify XSS attacks, directory traversal attacks, command injection attacks whereas they perform detection only on Java supported web application. A different tool is VIPER [5] which is language independent but does not support XSS attacks and directory traversal attacks, command injection attacks. AMNESIA [6] performs detection simply on Java supported web application and detects only SQL Injection attacks. ARDILLA [7] does not detect directory traversal attacks, command injection attack and they execute detection just on PHP supported web application. And another tool D-WAV [8]is an automatic technique that provides any language support but detects XSS and SQL Injection attacks only.

Proposed IDS

The paper proposes an improved detection of input validation attacks on web application. Our proposed detection concept will detect to recognize Cross Side Scripting attacks, SQL injection attacks, Command injection attacks and also detect Directory Traversal attacks. In addition with this, proposed IDS model support every web applications developed using several languages like PHP, java, Dot net etc. Thus our proposed method has a concept to be used for detecting and securing web application from input validation vulnerability. And also expect that the concept will reduce the analysis time because entire process operates without developer interaction.

V. CONCLUSION AND FUTURE WORK

This paper present a survey on web application attacks i.e. types of security threats within web application. The paper proposes an improved detection of input validation attacks on web application. Our proposed detection concept will detect to recognize Cross Side Scripting attacks, SQL injection attacks, Command injection attacks and also detect Directory Traversal attacks. In addition with this, proposed IDS model support every web applications developed using several languages like PHP, java, Dot net etc. Thus our proposed method has a concept to be used for detecting and securing web application from input validation vulnerability. And also expect that the concept will reduce the analysis time because entire process operates without developer interaction. Future work of our study will be the implementation of a technique that uses a method for the detection of input validation attacks on web application. Additionally this paper proposes an improved and efficient tool that would provide web security.

ACKNOWLEDGEMENTS

This work was supported by the department of computer science, Hemchandracharya North Gujarat University of Patan-India. I specially wants to thank my guide Dr. A. R. Patel, who give me constant source of inspiration in me. Finally I want to thanks my family and friends for their support.

REFERENCES

[1] Dainotti, A.; Gargiulo, F.; Kuncheva, L.I.; Pescape, A.; Sansone, C., "Identification of Traffic Flows Hiding behind TCP Port 80," Communications (ICC), 2010 IEEE International Conference on , vol., no., pp.1,6, 23-27 May 2010 ISSN 1550-3607.

[2] OWASPD-Open Web Application Security Project. “Top ten most critical Web Application Security Risks”, https://www.owasp.org/index.php/Top_10_2013-Top_10.

[3] “Security Code Review- Identifying Web Vulnerabilities”by Kiran Maraju.

[4] Halfond, William GJ, Alessandro Orso, and Pete Manolios. "WASP: Protecting Web applications using positive tainting and syntax-aware evaluation." Software Engineering, IEEE Transactions on 34.1 (2008): 65-81.

[5] Ciampa, A., Visaggio, C. A., & Di Penta, M. (2010, May). “A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications”. In Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems (pp. 43-49). ACM.

[6] William G.J. Halfond and Alessandro Orso, ”AMNESIA: Analysis and Monitoring for Neutralizing SQLI Attacks” 20th IEEE/ACM International Conference on Automated Software Engineering, Long Beach, USA 2005, pp. 174-183.

[7] Kieyzun, A., Guo, P. J., Jayaraman, K., & Ernst, M. D. (2009, May). Automatic creation of SQL injection and cross-site scripting attacks. In Software Engineering, 2009.

[8] Lijiu Zhang, Quing Gu, Shushen Peng, Xiang Chen, Haigang Zhao, Daoxu,” D-WAV Aweb Application Vulnerabilities Detection Tool Using Characteristics of Web Forms” ICSEA’10, IEEE.

[9] Takeshi Matsuda,Daiki Koizumi,Michio Sonoda,Shigeichi Hirasawa, ”On predictive errors of SQL injection attack detection by the feature of the single character” Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on 9-12 Oct 2011, On Page 1722-1727.

[10] Halder, Raju, and Agostino Cortesi. "Obfuscation-based analysis of SQL injection attacks." In Computers and Communications (ISCC), 2010 IEEE Symposium on, pp. 931-938. IEEE, 2010.

[11] Halfond, W. G., & Orso, A. (2006, May). Preventing SQL injection attacks using AMNESIA. In Proceedings of the 28th international conference on Software engineering (pp. 795-798). ACM.

