International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-4, Issue-12)

Research Article | **December 2015**

Detection and Prevention of Input Validation Attacks: Survey and Strategies

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

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

Abstract-

Web applications play a very important role in today's world and in individual life as well as in any country's development. Securing web applications has become extremely important as the information processed by web applications has become critical to corporations, customers, organizations, and countries. Web applications manage a wide array of information including financial data, medical records, social security numbers, intellectual property and national security data. Web based attacks are considered by security experts to be the greatest and least understood of all risks related to confidentiality, availability, and integrity. This paper will introduce and address input validation attacks from attack to detection and solutions for the same. Attacks covered are SQL Injection (SQLI), Cross Site Scripting (CSS) and Directory Traversal.

Keywords- SQL injection attacks, Cross site scripting attacks and directory traversal attacks

General Terms- Web Security, Web Application, Input Validation Attacks

I. INTRODUCTION

A Web application is an application program that is stored on a remote server and delivered over the Internet through a browser interface.

A web application or web service is a software application that is accessible using a web browser or HTTP(s) user agent. A web application security deals with the security of websites, web applications and web services . A poor application security measures can lead to breaches in data [1]:

- ***Authentication:***
  The process of uniquely identifying the clients of applications and services. These might be end users, other services, processes, or computers.
- ***Authorization:***
  The authenticated client is permitted to access the resources(include files, databases, tables, rows, and so on) and operations(performing transactions such as purchasing a product, transferring money from one account to another, or increasing a customer's credit rating.).
- ***Integrity:*** Protection of information from tampering, forgery, or accidental changes.
- ***Confidentiality:***
  Ensures that applications and data is accessible to only the users intended and authorized to have access. It is the process of making sure that data remains private and confidential, and that it cannot be viewed by unauthorized users or eavesdroppers who monitor the flow of traffic across a network.
- ***Availability:*** Ensures that authorized users have access to the application and the data when required. From a security perspective, availability means that systems remain available for legitimate users.
- ***Accountability:*** Ensure accuracy of data and guide against unauthorized modifications

Commonly web servers, application servers, and web application environments are susceptible to following types of vulnerabilities.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 detection and prevention survey and schemes of input validation attacks. Rest of the paper describes conclusion.

II. 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:

© 2015, IJERMT All Rights Reserved

1. 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.

2. Cross-Site Scripting Attacks (CSS or XSS)

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.

3. Directory or 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. LITERATURE SURVEY

SQL Injection Attacks: While Typing SQL keywords and control signs an intruder is able to modify the structure of SQL query developed by a Web designer. SQL Injection is a kind of web application security exposure in which an attacker is able to expose a database SQL command, which is executed by a web application. SQL Injection attacks can take place when a web application utilizes user-supplied data without proper validation or encoding as part of a command or query. SQL injection attacks are nothing but injecting malicious queries by the hackers into the application projected queries to get the desired outputs from the database.SQL Injection allows an attacker to create, read, update, modify, or delete data stored in the back-end database. Thus, SQL injection exploits security vulnerabilities at the database layer.

In SQL-Injection we exploited the vulnerability by injecting SQL Queries as user inputs. A web application isvulnerable to SQL injection attacks when malicious content can flow into SQL queries without being fully sanitized, which allows the attacker to trigger malicious SQL operations by injecting SQL keywords or operators[2].

Cross-Site Scripting Attacks: In XSS, we inject code (basically client side scripting) to the remote server. A web application is vulnerable to XSS attacks when malicious contents can flow into web responses without being fully sanitized, which allows the attacker to execute malicious scripts in victims' browsers, since the web browser trusts the contents returned by the web application under the same-origin policy.

Common consequences of XSS attacks include disclosure of users' sensitive information, such as cookie details and credit card information.

There exist different forms of XSS attacks, depending on how malicious code is submitted to the vulnerable application and later echoed from the application to its users[3].

Reflected (Non Persistent):

Attacker needs to monitor the web application then design URL so user provide the data to server for processing of request and in the mean time clicking on the URL. Attacker hijacks the web site and prompts the error message.

© 2015, IJERMT All Rights Reserved

Stored (Persistent):
In Stored attack the malware code is embedded in a web page or the vulnerable data stored in web server. So the malicious malware injects everyone into this attack. So this attack is much more powerful and harmful to affect web pages. The malicious code runs at client side of the participant to compromise its information security blindly. The participated browsers are poor in capability detecting such scripts with assumes that the service providers protected them. Some of these capabilities (all the special characters (e.g., "<", ">", "&", etc.)) need to be identified and encoded if they are included into the output, or they need to be filtered by the web application included into the input. As consequence, the problem should be considered at the client side in default. The accuracy and performance of previous works which used to detect malicious JavaScript attacks that doesn't satisfies the users need, moreover the generality of the tools is a problem to detect malicious JavaScript code from different websites.

Document Object Model (DOM) based XSS:
This attack launched when injected code manipulates sites JavaScript code or variables, rather than HTML objects. JavaScript often use user inputs to modify the DOM. Input can be URL parameters XHR Responses, HTTP headers etc. Server side input validation logic fails at data sanitization.
A data structure which permits to access and modifies the content, structure, and style of HTML documents dynamically in client-side scripting code i.e. DOM. Browser populates some of its properties relies on request parameters, rather than document characteristics.
In this case, the vulnerable application presents to the users an HTML page that uses data from parts of its DOM in insecure ways. In other words, DOM-based XSS occurs when the malicious scripts are injected into the client-side JavaScript code for execution, even without sending to the server side. It is worth noting that DOM-based XSS is extremely difficult to handle using only server-side defenses.
For example, the document.URL and document. location properties are set to the URL of the document by the browser. If an HTML page contains code that dynamically changes the appearance of the page using the content of document.URL (e.g., to show to the user the URL associated with the page), it is possible to use a maliciously crafted URL to execute malicious scripting code.
Directory Traversal Attacks: Web server consist of directories where files, information, application functions are stored to provide the services which does not have access to users. But attackers obtain these unauthorized directories, by traversing the directory in the address area of web browser and may misuse it[1].

Directory traversal is a type of HTTP exploit that is used by attackers to gain unauthorized access to restricted directories and files.

The goal of a Directory Traversal attack is to execute commands that will access files that are intended to be restricted. This type of attack uses HTTP to bypass Web server and Web application security. It is enabled by insufficient and missing security measures in servers and websites.

These attacks can be viewed in two basic groups: attacks that target directory traversal vulnerabilities in the web server and attacks that target vulnerabilities in application code.

IV. DETECTION AND PREVENTION OF INPUT VALIDATION ATTACKS

SQL INJECTION ATTACKS:

DETECTION:

- IDS Approach, Content filtering, penetration testing, and defensive coding, Generic Signatures , Accurate and Efficient Taint Propagation, Syntax- Aware Evaluation , Minimal Deployment Requirements [3].

PREVENTION:

- An Authentication Scheme using Hybrid Encryption.
- Effective SQL Injection Attack Reconstruction Using Network Recording.
- Insecure Query Processing in the Delay/Fault Tolerant Mobile Sensor Network (DFT-MSN) and Mobile Peer to Peer Network.
- Secure Query Processing In Delay Tolerant Network Using Java Cryptography Architecture.
- SQL Injection Attack Detection using the Removal of SQL Query Attribute Values [3].
- Dynamic Candidate Evaluations Approach to prevent SQL injection.
- Obfuscation-based Analysis of SQL Injection Attacks.
- SQL injection Detection via Automatic Test Case Generation of Programs.
- Combinatorial Method for Preventing SQL Injection Attacks.
- An Approach for SQL Injection Vulnerability Detection- AMNeSIA [3].
- Use language specific libraries to perform the same functions as shell commands and system calls
- Check for existing reusable libraries to validate input, and safely perform system functions, or develop your own.
- Perform design and code reviews on the reusable libraries to ensure security.
- Use stored Procedures.
- Data validation (to ensure input isn't malicious code).
- Run commands with very minimal privileges.
- A clustering approach for web vulnerabilities detection.

© 2015, IJERMT All Rights Reserved

- Error pattern approach.
- SAFELI framework aims at identifying the SQL Injection attacks during the compile-time.
- Thomas et al., in suggest an automated prepared statement generation algorithm.
- Ruse et al. propose a technique that uses automatic test case generation.
- Haixia and Zhihong propose a secure database testing design for Web applications.
- Roichman and Gudes, in order to secure Web application databases, suggest using a fine-grained access control to Web databases.
- Shin et al. suggest SQLUnitGen, a Static-analysis-based tool that automate testing for identifying input manipulation vulnerabilities
- Kemalis and Tzouramanis in suggest using a novel specification-based methodology
- SQLrand approach using randomized SQL query language, targeting a particular *CGI* application is proposed by Boyd and Keromytis.
- Parse Tree Validation Approach compared the parse tree of a particular statement at runtime and its original statement.
- Algorithm with SQLCHECK on a real time environment by Su and Wassermann.
- In Manual approaches, defensive programming and code review are applied.
- Automated approach: Static analysis FindBugs and Web vulnerability scanning.

CROSS-SITE SCRIPTING ATTACKS:

DETECTION:

- Acunetix Web Vulnerability Scanner and XSSer penetration testing tool that automates the process of detecting and exploiting XSS injections in any website.

PREVENTION:

- Automatic Creation of SQLI and XSS Attacks.
- Protecting Cookies from Cross Site Script attacks Using Dynamic Cookies Rewriting Technique.
- An Execution-flow Based Method for Detecting Cross-Site Scripting Attacks.
- Perform data integrity checks on data prior to their submission to ensure the data are reasonable.
- When possible, restrict all end-user input to alphanumeric content.
- Filter output by converting text/data which might have dangerous HTML characters to its encoded format:

'<' and '>' to '<' and '>'
'(' and ')' to '(' and ')'
'#' and '&' to '#' and '&'

- Recommend filtering on input as much as possible [4].

DIRECTORY TRAVERSAL ATTACKS:

DETECTION:

- A Web vulnerability scanning and manual penetration testing techniques to detect directory traversal vulnerabilities and It can be located in web server software/files or in application code that is executed on the server.

PREVENTION:

- Recent web servers, including IIS and Apache, provide protection against this type of directory traversal attack.
- Web servers use two types of mechanisms to prevent access unauthorized access to restricted directories and files:

i. Access Controls Lists (ACLs) – Limits access to specific files within the root directory. Only users who are listed in the ACL for a file can access that file. Administrators use Access Control Lists to define user access rights and privileges for viewing, modifying, and executing files.
ii. The Root Directory – Limits user access to a specific directory (known as the root directory) and its subdirectories. The root directory is the top-most directory on a server file system. User access is confined to the root directory, meaning users are unable to access directories or files outside of the root.

V. CONCLUSION

Web applications have been growing extremely fast with innovative programming languages and technologies. This results in challenges for web application security, which requires extensive and continuous efforts from security researchers. This paper provides survey on Input Validation Attacks such as SQL Injection (SQLI), Cross Site Scripting(CSS) and Directory Traversal with detection and prevention schemes. But with various complexities such as an increasing amount of application code and logic, multiple web applications are integrated and embedding third-party programs with security it is not possible to provide clean solution to all these attacks. As new types of attacks are always up-and-coming which requires security professionals to quickly act in response and put a massive number of web applications at risk.

© 2015, IJERMT All Rights Reserved

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] Katkar Anjali S., Kulkarni Raj B., "Web Vulnerability Detection and Security Mechanism", International Journal of Soft Computing and Engineering (IJSCE), ISSN: 2231-2307, Volume-2, Issue-4, and September 2012.

[2] Xiaowei Li, Yuan Xue, "A Survey on Server-side Approaches to Securing Web Applications", ACM 1529-3785/2013/0700-0001, 2013.

[3] Rahul Johari, Pankaj Sharma, "A Survey On Web Application Vulnerabilities (SQLIA, XSS) Exploitation and Security Engine for SQL Injection" 2012 International Conference on Communication Systems and Network Technologies 978-0-7695-4692-6/12, IEEE DOI 10.1109/CSNT.2012.104, 2012.

[4] Open Web Application Security Project[Online], https://www.owasp.org/index.php/Main_Page

[5] Diallo Abdoulaye Kindy and Al-Sakib Khan Pathan "A Detailed Survey on Various Aspects of SQL Injection: Vulnerabilities, Innovative Attacks, and Remedies."

[6] Giovanni Vigna "Vulnerability Analysis of Web-based Applications", 2007.

[7] "Security Code Review- Identifying Web Vulnerabilities"by Kiran Maraju.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

© 2015, IJERMT All Rights Reserved