ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

SQL INJECTION VULNERABILITIES IN INTEGRATION METHODS OF CAPTCHA IN WEB APPLICATIONS

Arshad Kamal*, Mohd Shamsh Alam

* M Tech Scholar, Department of Computer Science & Engineering, Al Falah School of Engineering & Technology, Faridabad, Haryana, India.

M Tech Scholar, Department of Computer Science & Engineering, Al Falah School of Engineering & Technology, Faridabad, Haryana, India.

ABSTRACT

SQL Injection is defined as a kind of security attacks through which intruders, hackers or malicious users easily get access to the web application's underlying database. SQL Injection is only of the top 10 techniques used by hackers to exploit the websites to take the confidential data for different purpose.

The solution to SQL Injection attack has been proposed by Oracle and Other experts which is based on certain guidelines and checks while designing an application to keep the minimum target surface area and control procedures.

KEYWORDS: SQL Injection, CAPTCHA, Web Security, Public Private Captcha Integration.

INTRODUCTION

We see a high similarity in the human need model defined by Maslow to the current requirement on security in the internet age. Maslow defines Basic needs as the primary need and is the first step in the model hierarchy. Security and Safety comes second. Similarly, we see the trend in the IT and Internet evolution. Initial push was to make the connectivity available, bring the services at the door steps. As the internet users has increased and it has found place in all the activities we do. The basic fundamentals of information security i.e. Availability, Confidentiality and Integrity becomes very important.

SQL Injection is only of the top 10 techniques used by hackers to exploit the websites to take the confidential data for different purpose.

The solution to SQL Injection attack has been proposed by Oracle and Other experts which is based on certain guidelines and checks while designing an application to keep the minimum target surface area and control procedures. Denial of Service (DoS) is also one of another technique used by hackers to block the services on a website, or to do malicious activities.

CAPTCHA is one of the techniques used by the application developers to ensure that the website services are not put down by automated scripts by malicious users. CAPTCHA is based on Turing Test and here the system differentiates between a human interface and machine interface.

CAPTCHA has been very successful and it has found its application and uses in almost all the websites we use. There have been many research activities happened to increase the complexity and Strength of CAPTCHA.
In the current time, highly complex and strong CAPTCHA is available as a third party service which can be integrated in application.

The purpose of the paper is to highlight the scenario where making only CAPTCHA strong doesn't make application secure, if proper techniques suggested controlling SQL Injection attacks are not followed.

SQL INJECTION VULNERABILITIES

SQL Injection is defined as a kind of security attacks through which intruders, hackers or malicious users easily get access to the web application's underlying database. Attacker adds SQL code to a web form input box to gain access or to make changes to data. For example a hacker can authenticate a User ID and Password by simply typing ' OR '1'='1 in place of User ID and Password. if the caution would had not been taken into consideration. It Will look like, SELECT * FROM USERS WHERE NAME =' ' OR '1'='1' - - '

Hence we can say that the web application vulnerabilities are the main cause of any kind of attack [1].

The vulnerabilities that might exists naturally in web applications and can be exploited by SQL Injection attacks could be one of following,

A. INVALIDATED INPUT
B. GENEROUS PRIVILIGES
C. UNCONTROLLED VARIABLE SIZE
D. ERROR MESSAGE
E. VARIABLE ORPHISM
F. DYNAMIC SQL WITH CONCATENATED INPUT
G. CLIENT SIDE ONLY CONTROL
H. STORED PROCEDURES
I. INTO OUTFILE SUPPORT
J. MULTIPLE STATEMENTS
K. SUB SELECTS

CAPTCHA IN WEB APPLICATIONS

CAPTCHA has become de facto standard for security measures on the World Wide Web that prevent automated scripts from abusing online services, invented in 2000 at CMU by Luis Von Ahn, Manuel Blum, Nicholas J. Hooper and John Langford. CAPTCHA stands for Completely Automated Public Turing Test to tell computers and human apart, it acts like a judge and participants like a user [2].

CAPTCHA is a defensive system that prevents web bots from abusing online services on the internet including free e-mail providers, wikis, blogs etc. It is a HIP system that prevents users from bots in an online environment. Various application of CAPTCHA is as :

- Protecting online polls
- Preventing E-mail Spam
- Web Registration Protection
- E-Ticketing
- Preventing Dictionary Attacks
- Search Engine bots
- Automating Document Identification

Types of CAPTCHA

a. Text Based
b. Image based
c. Puzzle based
d. Audio based

RELATED WORK

SQL Injection has been addressed by many proposed and developed methods. These detection and prevention schemes follow different approaches to protect at various levels of the architecture like network, server, application etc. There are some methodologies applicable to other types of vulnerabilities like Cross-Site-Scripting (XSS) [3] or XPath Injection. Some are applicable to web application while others are specific to another implementation-independent like reverse-proxies [4]. Some of are as follows.

- Active input data encoding
- Tainting [5]

- Instruction set Randomization [6]

PROPOSED INTEGRATION METHOD OF CAPTCHA IN WEB APPLICATIONS TO PREVENT SQL INJECTION ATTACK

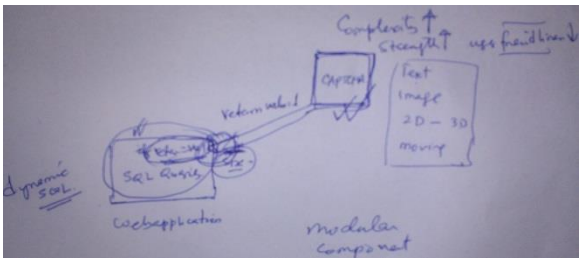

Figure 1: Proposed work layout

During the study of CAPTCHA, we have done following observations which we have initiated in our current project. Our proposal is to introduce one private attribute in CAPTCHA, to make it Public-Private key based validation. The CA[P]TCHA changes to CA[PP]TCHA and here P for Public is replaced with PP for Public Private.

Figure 2: Public Private Captcha login

With the introduction of additional Private variable in session, and validation to happen based on the combination of Public-Private Key, we will be able to reduce the complexity of the CAPTCHA without compromising on the existing strength of the CAPTCHA.

During the study on Cyber Security Risk and Ethical hacking practices, we observe that SQL Injection is a known vulnerability for many years, and still is ranked as number one risk. In the current work we are trying to integrate it with CAPTCHA. The vulnerabilities can be of many types; however we have focused on SQL Injection understanding its criticality based on OWASP 2013 Top 10 reports.

Examples of SQL Injection
Scenario #1: The application uses untrusted data in the construction of the following vulnerable SQL call:
String Query="select * from accounts where custID='"+request.getParameter("id")+"'";
Scenario #2: Similarly, an application's blind trust in frameworks may result in queries that are still vulnerable (eg. Hibernate Query Language):
Query HQLQuery= session.createQuery("FROM accounts WHERE custID='" + request.getParameter("id") + "'");

In both cases, the attacker modifies the 'id' parameter value in her browser to send:
'or '1'='1. For Example:
http://example.com/app/accountView?id='or' 1'='1
This changes the meaning of both queries to return all the records from the accounts table. More dangerous attacks could modify data or even invoke stored procedures.
SQL Injection Based on 1=1 is Always True.
SQL Injection Based on "=" is Always True.

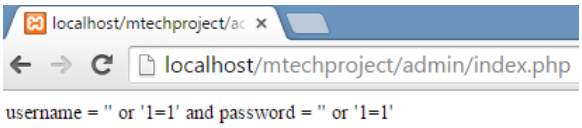

Figure 3: SQL Injection

In the both scenarios above, the SQL string formed is a valid SQL, and login happens successfully.

SQL Injection Based on Batched SQL Statements One of the proven way to protect a web site from SQL Injection attacks is to use SQL parameters. SQL Parameters are values that are added to an SQL query at execution time, in a controlled manner.

RESULT AND CONCLUSION

To minimize the vulnerabilities within web applications, it's important to improve Internet hygiene. This involves the following best practices for application design, implementation, configuration, testing and maintenance.

In the current work we have studied the application security with key focus on SQL Injection and CAPTCHA. We agree that there has been lots of research work and innovations happened towards making the applications secure, however there still remains area to improve looking at the risks involved.

We would like to conclude highlighting that the saying Prevention is Better than Cure holds true also in the field of information technology i.e. we need to make the information technology user technology more and more aware to be security driven. Methods and Resources used by hackers should be proactively deployed and practiced during the software development life cycle before production release to reduce vulnerability and make application strong and secure.

REFERENCES

[1] T. Pietraszek and C. V. Berghe. Defending against Injection Attacks through Context-Sensitive String evaluation. Recent Advances in Intrusion Detection, Volume: 3858, Pp: 124-145, 2006.
[2] McAfee White Paper on Bypassing CAPTCHAs by Impersonating CAPTCHA Providers by Gursev Singh Karla.
[3] Rami M.F. Jnena, Modern Approach for WEB Applications Vulnerability Analysis, The Islamic University of Gaza, Deanery of Graduate Studies, Faculty of Engg.
[4] Janot, E.: SQLDOM4J: Preventing SQL Injections in Object-Oriented Applications. Master thesis, Concordia University College of Alberta (2008), http://waziboo.com/thesis
[5] Huang, Y., Yu, F., Hang, C., Tsai, C., Lee, D., Kuo, S.: Securing Web Application Code by Static Analysis and Runtime Protection. In: Di Nitto, E., Murphy, A.L. (eds.) 13th international conference on World Wide Web, pp. 40--52. ACM, New York (2004)
[6] Boyd, S., Keromytis, A.: SQLrand: Preventing SQL Injection Attacks. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 292--304. Springer, Heidelberg (2004)