Int. J. Communications, Network and System Sciences, 2012, 5, 337-342
http://dx.doi.org/10.4236/ijcns.2012.56044 Published Online June 2012 (http://www.SciRP.org/journal/ijcns)

Review and Measuring the Efficiency of SQL Injection Method in Preventing E-Mail Hacking

Ahmed A. M. Sharadqeh[1], As'ad Mahmoud Alnaser[2], Omar Al Heyasat[2], Ashraf Abdel-Karim Abu-Ein[1], Hazem (Moh'd Said) Hatamleh[3]
[1]Computer Engineering Department, Faculty of Engineering Technology, Al-Balqa' Applied University, Amman, Jordan
[2]Computer Engineering Department, Al-Balqa' Applied University, Salt, Jordan
[3]Computer Science Department, Ajloun University College, Al-Balqa' Applied University, Ajloun, Jordan
Email: ashraf.abuain@fet.edu.jo, hazim-hh@bau.edu.jo, {sharadqh_78, asad1_99}@yahoo.com

Received April 12, 2012; revised May 6, 2012; accepted May 16, 2012

ABSTRACT

E-mail hackers use many methods in their work, in this article, most of such efficient methods are demonstrated and compared. Different methods and stages of such methods are listed here, in order to reveal such methods and to take care of them but the most common discussed method in this paper is SQL method. SQL injection is a type of security exploit in which the attacker adds SQL statements through a web application's input fields or hidden parameters to gain access to resources or make changes to data. It is found that the SQL is an efficient way in preventing E-mail hacking and its efficiency reaches about 80%. The method of SQL injection can be considered as an efficient way comparing with other methods.

Keywords: E-Mail; Hackers; SQL; Security

1. Introduction

Since 1984 and the commercialization of the Internet, the number of computers linked to the Internet as hosts has grown from approximately 1000 in 1984 to over 150 million in 2005. This growth means that there are more than 150 million people providing, using and sharing resources and information via the public network. These users and hosts are located all over the globe and are in every country that has access to the Internet. Different types of information and resources are available in many different forms, from informational web sites (static content), to interactive sites that display different results depending on the interaction (dynamic content) of the user. As with every population, you will have good persons, and bad persons, and every web site can fall victim to criminal activities. This paper will explain some of the different ways that criminals can do harm or get unauthorized information and how programmers can stop this by implementing security into their programming code [1-3].

When web pages were first implemented they only displayed unchanging information, which is called static content. They may have included hyperlinks that would point to other web sites or web pages, which made it easy for users to get information from multiple or related web sites. These web sites didn't offer any interactivity with users and didn't require user input. Techniques and programming languages were developed that allows newer web pages to include text boxes, check boxes, radio buttons, drop down lists and command buttons so that the user can provide data, information and commands to the web site. These web sites that change consists of dynamic web content and are also called interactive. Dynamic content is material on a Web page that is added or altered, usually after the page has been loaded by the Web browser and usually in response to actions or requests by the user. Michael Roche (2007), focused on making a survey of wireless attack tools focusing on 802.11 and Bluetooth. It includes attack tools for three major categories: confidentiality, integrity, and availability. Confidentiality attack tools focus on the content of the data and are best known for encryption cracking. Integrity attacks tools focus on the data in transmission and include frame insertion, man in the middle, and replay attacks. Finally, availability attack tools focus on Denial of Service (DoS) attacks. FERNANDO M *et al.* (2012), the authors are joined by a distinguished cyber security specialist, Wayne Lee, who provides a discussion of measures that businesses can implement immediately to secure their data and improve defenses against a data breach. Pinguelo and Muller also analyzed the various public-private partnerships on cyber-defense currently at work in the United States, and explain how those partnerships are helping create a safer cyber-future.

Ke *et al.* 2005, discussed the SQL as an injection attack targets interactive web applications that employ database services. These applications accept user inputs and use them to form SQL statements at runtime. During an SQL injection attack, an attacker might provide malicious SQL query segments as user input which could result in a different database request. By using SQL injection attacks, an attacker could thus obtain and/or modify confidential/sensitive information. An attacker could even use a SQL injection vulnerability as a rudimentary IP/Port scanner of the internal corporate network. Several papers in literature have proposed ways to prevent SQL injection attacks in the application layer by examining dynamic SQL query semantics at runtime. However, very little emphasis is laid on securing stored procedures in the database layer which could also suffer from SQL injection attacks. Some papers in literature even refer to stored procedures as a remedy against SQL injection attacks. As stored procedures reside on the database front, the methods proposed by them cannot be applied to secure stored procedures themselves. Debasish *et al.* 2010, A large number of web applications, especially those deployed by companies for e-business operations involve high reliability, efficiency and confidentiality. Such applications are often written in script languages like PHP embedded in HTML, allowing establishing connection to databases, retrieving data, and putting them in the Web. One of the most common in web application attacks is SQL Injection. In this an attacker attempts to use malicious crafted input strings so that the dynamic SQL queries generated by the web application is different from the structure designed by the developer. In this paper, an attempt has been made to classify the SQL Injection attacks based on the vulnerabilities in web applications. A brief review of the existing approaches for the detection of SQL injection attack also has been presented. Further paper presents an effective detection method (DUD) for the SQL injection based on dynamic query matching. The DUD approach is independent of the developer's initialization of syntactical rules, valid trusted string database, static or pre-generated program code checking, etc. Also, DUD is significant in view of its simple detection mechanism as well as its high detection rate.

1.1. Basic Architecture

The basic architecture of a web application can be thought of in terms of structure and language. The structure for interactive applications is usually multi-leveled and can be written in multiple languages.

1.2. Multi-Tier

Web application architecture can be thought of as a multi or n-Tier structure. This means there are several layers needed to build a web application and usually made up of at least three layers; the presentation tier (front end), application (business logic) and data tiers (back end). The presentation tier, which is very commonly referred to the .front end, is portion of the application that the user interacts with directly and is normally displayed by a software known as the browser. This user interaction portion is referred to as the user interface. The application or business logic tier of the web application is the main part of the program where the programming code is written and stored. This is the brain of the application where all the commands and instructions are carried out. The data tier is the portion of the web application that stores all of the data and is commonly referred to as the back end. Most applications store the data in a type of file management system called a database, which has made the interaction between user and data possible. **Figure 1** below, illustrates the flow as the user loads their browser into memory, then requests a web page from the application server, the application server then will request information from the back end or database and the data is sent back through the application and rendered in the browser by the HTML code [4,5].

1.3. Languages

Web applications can be written in many languages both high-level and scripting. Hypertext Markup Language (HTML) is a markup language that is typed into a text document and is used with an interpreter (*i.e.* Netscape Navigator, Microsoft Internet Explorer, or Apple Safari, etc.) to create static content and to display the results of a dynamic content web page. Languages such as ASP.NET and J2EE are high-level languages used to write large-scale web applications that are more complex and harder to code. These languages are used for heavy duty computing. Procedural languages such as ASP, Cold Fusion, PHP and scripting languages such as JavaScript and VBScript used for lighter, less complex web page content and are the most common languages used.

1.4. Security

Since the Internet has more criminals lurking around trying to steal important information, there have to be laws, standards and practices put in place to help protect the confidentiality, integrity and availability of data that is accessed through the Internet. When consumers open an account, register to receive information or purchase a product from your business, it's very likely that they entrust their personal information to you as part of the process. If their information is compromised, the consequences can be far reaching: consumers can be at risk of identity theft, or they can become less willing or even unwilling to continue to do business with you.

1.5. Vulnerable

A security hole is defined as the program [that] has a flaw that allows an attacker to "exploit it". Thus comes the "exploit" that denotes a program, or technique to take advantage of the flaw, or "vulnerability". In the past, security was considered a networking topic and threat countermeasures were implemented on the network and operating system. Those measures included firewalls and intrusion detection systems. However .traditional network-based countermeasures, such as firewalls and IDS, do very little in the way of offering protection to web-based applications [7,8].

The problem is made worse by unsecured application development as most experienced developers are not knowledgeable about the new web application environments, but are immerging from the legacy hard coded procedural applications and have not yet caught on to the new developing script programs. Another challenge met by application development is the time to market. mind frame of marketing, this is the idea that the product must be on the market as soon as possible, meaning that testing is not done to alleviate all bugs in the system prior to market. Hackers are sometimes able to anticipate inadequacies and the coding practices adopted by programmers. Often, the "speed to market" attitude pushes application developers to overlook standard and secure coding practices. Lack of knowledge by new or inexperienced programmers, coupled with quick and shoddy programming, makes web applications an easy target for criminals [6-8].

1.6. Hacking Techniques Stages

There are multiple ways a criminal can target the web application tiers. According to Hacking Techniques in Wired Networks: an attack will go through several steps and these actions will use more than one hacking technique. Seven of these are outlined below.

1.6.1. Reconnaissance

In this step the criminal will gather information about the target system. In web applications this can include the usernames, passwords, input parameters, programming or script languages, type of server and operating system.

1.6.2. Probe

In this phase the criminal detects weaknesses in the system. SQL injection is the most used here for web applications in order to find out what permissions are placed on the system and where the security holes exist.

1.6.3. Toehold

Once the vulnerabilities are determined in the probe stage, then the criminal will proceed to the toehold phase where the intruder begins to enter the system, build a connection, usually referred to a session, search the system for information and begin exploiting the vulnerability.

1.6.4. Advancement

This is the phase in which the criminal will move look for configuration errors and move from an unprivileged account into a privileged, or move from a normal user with little access to one of an administrator or super user, in which case the criminal would have full access to create, delete, modify, retrieve or move files and information.

1.6.5. Stealth

The stealth stage is a step that is taken to mask the intruder's presence in the system. This is done by accessing and modifying local log files to remove any evidence that would alert or suggest an attack has taken place.

1.6.6. Listening Post

In the listening post phase, the intruder will set up a backdoor, which is a malicious program that will ensure that future activities will not be logged. These programs are called stealth or backdoor tools, or sniffer. These will report false information on the file access and processes on the network and system in order to disguise the intruders' subsequent entries.

1.6.7. Takeover

The takeover stage is used to expand the control from one system to others on the same or connecting networks. The intruder can use a sniffer to detect other information on other hosts such as usernames and passwords. This can be used to find host machines that trust the computer the criminal is currently attacking so that the intruder can get access to the other hosts. These steps can be done in a number of ways and use a number of tools for each. Some of these are enabled simply by poor programming and should be controlled by the programmer at the design phase.

2. Top Ten Vulnerabilities

The Ten Most Critical Web Application Security Vulnerabilities, which states that most network security ignores the content of HTTP traffic and therefore creates no policing of traffic through port 80. This means that there is much vulnerability that exists. The top ten are outlined in the document are listed below:

1) Un-validated Parameters;
2) Broken Access Control;
3) Broken Account and Session Management;
4) Cross-Site Scripting Flaws;

5) Buffer Overflows;
6) Command Injection Flaws;
7) Error Handling Problems;
8) Insecure Use of Cryptography;
9) Remote Administration Flaws;
10) Web and Application Server Miss-configuration.

Now we know where vulnerabilities exist and the steps criminals take to access, steal or deface information, we can look at some of the measures web site developers can take in implementing security to protect an organization's assets, such as data validation [8,9].

Data Validation

The most common weakness in web applications is the failure to properly validate input from the user (client or environment) in URL, query strings, form fields, hidden fields, cookies and headers. User input should be checked for items such as strongly typed data, correct syntax, data within length boundaries, data that contains only permitted characters, or if numeric data is within range boundaries. Web applications use input from users in order to determine how to react to an HTTP request, if the data does not meet these criteria, then the application may be subject to SQL injection or buffer overflow [2, 7,8].

3. Results and Discussion

3.1. SQL Injection

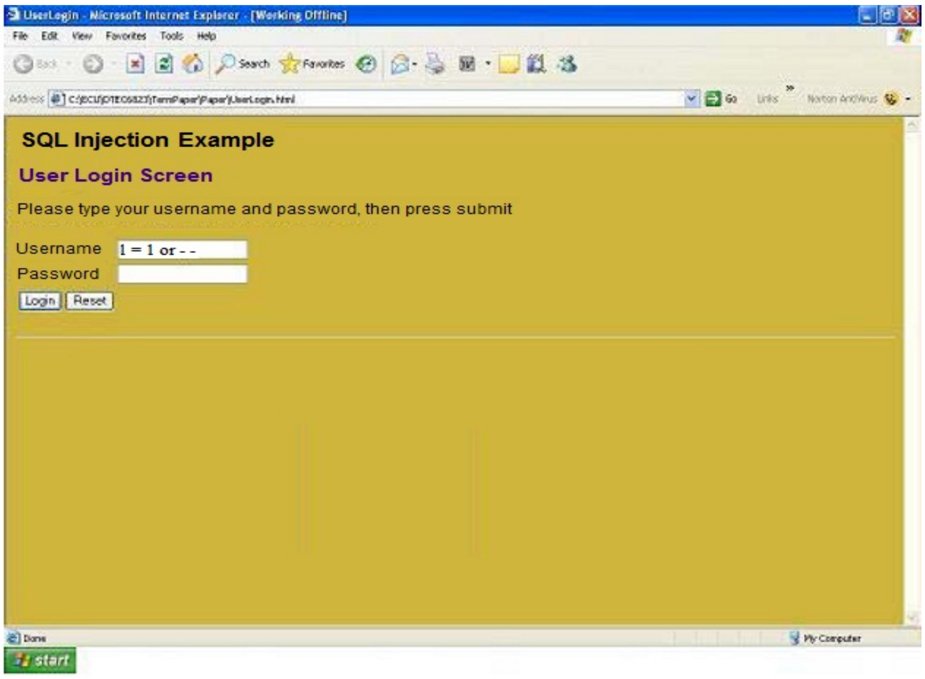
SQL injection is a type of security exploit in which the attacker adds SQL statements through a web application's input fields or hidden parameters to gain access to resources or make changes to data 12. A resource field is usually considered a column in a database that holds important data about an individual, customer or entity. Validation is key to making sure these SQL statements don't contain criminal code necessary to illegally obtain unauthorized information.

This can be done using any database, as no any one vendor is exempt. SQL injection is not a defect of Microsoft SQL Server. It is also a problem for every other database vendor as well. Let's take a look at how SQL injection works by looking at a simple database with user names and passwords. In a normal database, the table named Login sees **Table 1**.

The primary ID is a unique identifier for the row and is only used by the database and programmers to identify that record. The Username field contains each user's identity and the Password field contains the user's chosen password. Now let's look at a user sign on screen shown in **Figure 1**.

Table 1. User names and passwords.

Primary ID	Username	Password
1	mickey123	mic123
2	minnie456	min456
3	donald789	don789
4	goofy012	goo012


UserLogin - Microsoft Internet Explorer - [Working Offline]
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\ECU\OTEC6823\TermPaper\Paper\UserLogin.html
Go Links Norton AntiVirus
Done
My Computer
start

SQL Injection Example

User Login Screen

Please type your username and password, then press submit

Username 1 = 1 or - -
Password
Login Reset

Figure 1. SQL injection screen.

If the form field of the web page for the username didn't validate and allowed us to type. 1 = 1 or – –. Then the resulting query would be read as: SELECT * FROM Login WHERE Username = **' ' or 1 = 1**– – AND password = **' '**.

In order to understand this, you need a little knowledge of queries. Since understanding of queries is beyond the Scope of this paper, we will only explain what this sample query means as shown in **Table 2**.

At this point a query has been sent to the database engine, which will yield all records in the database. Using other code in addition with this, will yield a list of all usernames and passwords in the database. Once this data is collected, you can gain connectivity to the database and have completed the toehold phase of hacking.

Once the toehold phase is complete, the criminal can move into the advancement phase by writing all of the username and passwords into a file and the file can be transferred to the criminal's computer. Other phases of hacking can be completed using SQL injection and database query commands meaning that multiple servers can be accessed and compromised. **Table 3** shows an example of query commands and their functions, giving an indication of the power of this type of exploit.

Table 2. Codes and functions.

Statement or code	Description
SELECT	Instructs the database engine to retrieve recorders.
***(wildcard character)**	Instructs the database engine to Select all fields.
FROM login	Instructs the database engine to retrieve recorders from the table named Login.
WHERE	Provides criteria to match when selecting records.
" "	Evaluated by the database engine will cause all fields that are null ' ' or empty to be returned (which are really useless here, except that it negates the ' in the SQL statement that would be created in a normal; query).
or	Instructs the code to look for other criteria in addition to the first criteria.
1 = 1	Evaluated by the database engine to be true which will then yield all records in the table.
– –	Acts as a comment in code and is not executed.

Table 3. Commands and functions.

Command	Function
SELECT	Selects rows (records)
INSERT INTO	Insert rows (records)
DELETE	Deletes rows (records)
UPDATE	Changes rows (records)
CREATE TABLE	Creates a new table

The database is not the only entity that can be affected here; the criminal can also access the network as well as the operating system at this point. An example of starting an FTP service using SQL injection is as follows: ';exec master..xp_servicecontrol 'start','FTP.

3.2. Efficiency of SQL Injection Method

Efficiency is defined as the output of any system divided by the input, *i.e.* if ten of hackers tried to make hacking to ten E-mails then if the SQL injection method prevents 8 hacking of such E-mails then the efficiency of the SQL injection method is equal to 0.8 or 80%.

The efficiency of the method can be defined:

Efficiency % = (no. of revealed hackers/total no. of hackers) × 100%.

In this work SQL injection method is carried out on about 100 virtual hackers it passes in catching about 79 hackers and prevent them so,

$$\text{Eff.} = (79/100) \times 100\% = 79\%.$$

So if the hacker using the data-base system of the hacked e-mail to drag out the password of the user or any information leads to make the e-mail hacking SQL injection method will pass in about 80% to prevent this. In future it is suggested to build a method not to prevent hacking but to hack hackers. By giving them some true information and then determine their IP or specify their location.

Comparing with other methods the efficiencies of such method are compared in **Figure 2**.

4. Conclusion

There are many different ways for a criminal to break into a system and wreak havoc on a network or computer system. It is up to the web application coder to do their part in making sure the applications they design are not vulnerable to any known threats. The SQL method depends on a data-base system which directly depends on quires of the hackers about any information belongs to the hacked E-mail. It is found that 80% of such hackers

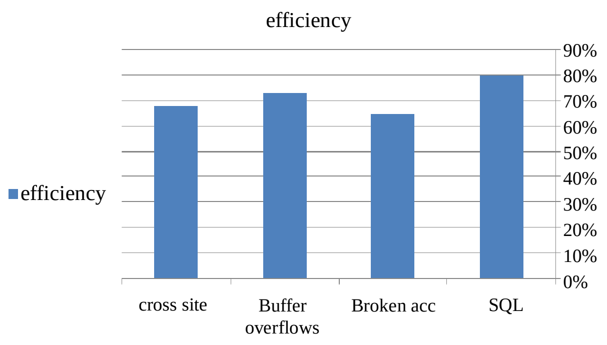


Figure 2. Some hacking methods comparison.

are caught by using SQL injection method.

REFERENCES

[1] C. Cerrudo, "Manipulating Microsoft SQL Server Using SQL Injection," Application Security, Inc., 2005. http://research.mwjournal.com/detail/RES/1124462486_292.html

[2] D. Das, U. Sharma and D. K. Bhattacharyya, "An Approach to Detection of SQL Injection Attack Based on Dynamic Query Matching," *International Journal of Computer Applications*, Vol. 1, No. 25, 2010, pp. 28-34.

[3] G. B. Shelly, T. J. Cashman and M. E. Vermaat, "Discovering Computers 2005: A Gateway to Information," Course Technology, Boston, 2004.

[4] K. Stasiak, "Web Application Security," *Information Systems Control Journal*, Vol. 6, 2002. http://www.isaca.org/Content/ContentGroups/Journal1/20023/Web_Application_Security.htm

[5] P. Carey, "Creating Web Pages with HTML and Dynamic HTML," Course Technology, Boston, 2001.

[6] M. Roche, "Wireless Hacking Tools," 2007. http://www.cse.wustl.edu/~jain/cse57107/ftp/wireless_hacking/2007

[7] S. Garfenkel and G. Spafford, "Secure AGI/CGI Programming," *World Wide Web Journal*, Vol. 2, No. 3, 1997. http://www.w3j.com/7/s3.garfinkel.wrap.html.

[8] W. Ke, M. Muthuprasanna and S. Kothari, "Preventing SQL Injection Attacks in Stored Procedures," *Proceedings of the Australian Software Engineering Conference*, Brisbane, 31 March-1 April 2005, pp. 191-1978.

[9] F. M. Pinguelo and B. W. Muller, "Virtual Crimes, Real Damages Part II," *Virginia Journal of Law & Technology*, Vol. 17, No. 1, 2010.