TCP/IP Malicious Packet Detection
(SQL Injection Detection)

Submitted in Partial Fulfilment of the Requirements of Napier University for the degree of Master of Science in Advanced Networking

Submitted By:
Ashok Parchuri
06008115
MSc Advanced Networking
Authorship declaration

I, Ashok Parchuri, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained informed consent from all people I have involved in the work in this dissertation following the School's ethical guidelines

Signed: Ashok Parchuri

Date: 26th September 2008
Matriculation no: 06008115
Data Protection declaration

Under the 1998 Data Protection Act we cannot disclose your grade to an unauthorized person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name against one of the options below to state your preference.

<table>
<thead>
<tr>
<th>Option</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ashok Parchuri</td>
<td>The University may make this dissertation, with indicative grade, available to others.</td>
</tr>
<tr>
<td>Ashok Parchuri</td>
<td>The University may make this dissertation available to others, but the grade may not be disclosed.</td>
</tr>
<tr>
<td></td>
<td>The University may not make this dissertation available to others.</td>
</tr>
</tbody>
</table>
Abstract

With rapid improvements in the data transfer speeds in the last two decades the Internet has become a major opportunity to enterprises to advertise themselves and to maintain their data safely in servers to make it more accessible. With the improvement in Internet technologies, malicious activities are threat for enterprises to maintain integrity of their data.

The purpose of this thesis is to capture the data flowing in a network, and to analyze it to find malicious packets carrying SQL injection attacks. It aims to detect these attacks as the attacker can compromise a server simply using the web browser and for it can cause severe damage such as losing data from a database or even changing the values. The literature review section shows that previous methods of detection have used anomaly detection, where as this thesis uses a novel metric-based system, which measures the threat level of a URL.

In creating a prototype of the system, an application was created to detect malicious packets using the C# language with Microsoft Visual Studio 2005. WinPcap is used for capturing the network packets that are flowing through the network interface in promiscuous mode. This application was developed based on the idea of capturing the packets and comparing them for the malicious keywords that are used for SQL injection attacks. The keywords, such as SELECT, DELETE, OR, and FROM are assigned with a malicious metric value, along with possible threats in the URL from certain characters, such as ‘=’ and a single quote character. When the resulting summation value reaches more than a given threshold, the application alerts the user that it found an injection attack. The thesis presents different weights for these threat elements, and produces an overall threat level.

Several tests have been conducted for analyzing the threshold value. These have been are conducted using over 1,000 URL strings that have been captured from normal traffic, and some have been injected with malicious keywords. It has been found that the application successful captures all the malicious strings, but it also resulted in false positives for strings that are not malicious. For a run of 1,000 URLs, it detected 10 true-positives, and 30 false-positives.

It concludes with a critique of the application is made, along with suggesting the future improvements that can make the application to improve performance. For future work, the thesis presents methods that could improve the metric system.
# Contents

Authorship declaration.......................................................................................... 2  
Data Protection declaration ................................................................................. 3  
Abstract ................................................................................................................ 4  
Contents .................................................................................................................. 5  
List of Figures: ....................................................................................................... 8  
List of Tables: ....................................................................................................... 9  
Acknowledgements ............................................................................................... 10  
1 Introduction ......................................................................................................... 11  
   1.1 Context ........................................................................................................... 11  
   1.2 Aim ................................................................................................................ 11  
   1.3 Background ................................................................................................... 12  
   1.4 Thesis layout ................................................................................................. 12  
2 Background ......................................................................................................... 14  
   2.1 Introduction ................................................................................................... 14  
   2.2 TCP/IP ........................................................................................................... 14  
      2.2.1 TCP Segment Format ............................................................................ 14  
      2.2.2 IP Datagram .......................................................................................... 15  
   2.3 HTTP Protocol .............................................................................................. 16  
      2.3.1 HTTP Message Formats ....................................................................... 17  
      2.3.2 HTTP Methods ....................................................................................... 17  
      2.3.3 HTTP Response codes: ......................................................................... 18  
   2.4 Databases ...................................................................................................... 19  
   2.5 Structured Query Language (SQL) .............................................................. 20  
      2.5.1 Data Manipulation Language ................................................................ 20  
      2.5.2 Data Definition language ..................................................................... 21  
      2.5.3 Creating Databases ............................................................................... 21  
      2.5.4 Operators and characters in SQL .......................................................... 22
2.6 Database Administration ................................................................. 23
2.7 WinPcap ......................................................................................... 23
2.8 Conclusions .................................................................................. 24

3 Design ............................................................................................. 25
3.1 Introduction .................................................................................... 25
3.2 Outline of SQL injection ................................................................. 25
3.3 Recent Examples of SQL Injection attacks ...................................... 26
3.4 Web Application Processing ............................................................ 26
3.5 SQL Injection Types ....................................................................... 28
  3.5.1 SQL Manipulation ...................................................................... 28
  3.5.2 Code Injection ........................................................................... 29
  3.5.3 Function Call Injection ............................................................... 29
  3.5.4 Buffer overflow ......................................................................... 30
3.6 Existing Technologies to stop SQL Injection .................................... 31
  3.6.1 Defensive Programming ............................................................. 31
  3.6.2 Anomaly Detection .................................................................. 31
3.7 Simple SQL Injection Attack by Example ....................................... 31
3.8 Conclusion ...................................................................................... 36

4 Implementation .................................................................................. 37
4.1 Introduction .................................................................................... 37
4.2 Design Considerations ................................................................... 37
4.3 IDS Design ..................................................................................... 38
  4.3.1 Design Pre-requisites ............................................................... 38
  4.3.2 SQL Keywords ......................................................................... 39
4.4 Programming .................................................................................. 41
  4.4.1 Getting Network device List: .................................................... 41
  4.4.2 On Packet Arrival .................................................................... 41
  4.4.3 Capture the URL String ............................................................ 43
  4.4.4 Comparing The URL string words ......................................... 43
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4.5</td>
<td>The application location</td>
<td>43</td>
</tr>
<tr>
<td>4.5</td>
<td>Conclusions</td>
<td>44</td>
</tr>
<tr>
<td>5</td>
<td>Evaluation</td>
<td>45</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>45</td>
</tr>
<tr>
<td>5.2</td>
<td>Application Evaluation</td>
<td>45</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Example1</td>
<td>45</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Example 2</td>
<td>47</td>
</tr>
<tr>
<td>5.3</td>
<td>Alerts</td>
<td>48</td>
</tr>
<tr>
<td>5.4</td>
<td>Conclusions</td>
<td>48</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
<td>50</td>
</tr>
<tr>
<td>6.1</td>
<td>Critical Analysis</td>
<td>50</td>
</tr>
<tr>
<td>6.2</td>
<td>Future Work</td>
<td>50</td>
</tr>
<tr>
<td>6.3</td>
<td>Final Words</td>
<td>51</td>
</tr>
<tr>
<td>7</td>
<td>References</td>
<td>52</td>
</tr>
<tr>
<td>8</td>
<td>Appendix</td>
<td>54</td>
</tr>
<tr>
<td>8.1</td>
<td>Program for Capturing SQL Injection attacks</td>
<td>54</td>
</tr>
<tr>
<td>8.2</td>
<td>Output Screens</td>
<td>56</td>
</tr>
</tbody>
</table>
List of Figures:

Figure 2.1: TCP Segment Format ................................................................. 15
Figure 2.2: IP Packet Format .................................................................... 16
Figure 2.3: Request and Response Chain .................................................... 16
Figure 2.4: Sample Request and Response Communication ......................... 19
Figure 2.5: Sample HTTP Request Headers. ................................................... 19
Figure 2.6. Select statement output .............................................................. 22
Figure 3.1. Web application processing .......................................................... 28
Figure 3.2. Sql Database Table Sample ......................................................... 28
Figure 3.3. SQL database Table .................................................................. 32
Figure 3.4. Sample Login application .............................................................. 32
Figure 3.5. Sample login application result ..................................................... 33
Figure 3.6. SQL injection In Login application ................................................. 34
Figure 3.7. Login application result by injection attack .................................... 34
Figure 3.8. SQL injection attack Example ...................................................... 35
Figure 3.9. SQL injection Attack example 2. .................................................. 35
Figure 3.10. Error produced by login Application ............................................ 36
Figure 5.1. Evaluation1 Analysis values .......................................................... 46
Figure 5.2. Evaluation2 Analysis values .......................................................... 47
Figure 5.3. Alert screen .............................................................................. 48
Figure 8.1. Output when No Injection attack is detected ................................. 56
Figure 8.2 Output of application when attack detected ................................. 56
Figure 8.3. Sample Log file Output ............................................................... 57
List of Tables:

Table 2.1. HTTP Methods ........................................................................................................... 18
Table 2.2: HTTP Error Codes.................................................................................................... 18
Table 2.3. SQL Data Manipulation language Statements ...................................................... 20
Table 2.4. SQL Data Definition Language Statements ......................................................... 21
Table 2.5. SQL Logical Operators .......................................................................................... 23
Table 4.1. SQL injection attack Keywords .............................................................................. 39
Acknowledgements

Firstly, I would like to thank Professor William Buchanan, for his guidance and support throughout this project. In addition, I would like to thank Mr. Jim Jackson for being part of the marking process.

I would like to thank my father, mother, brother and everyone in my family for their extensive support all these time by all means needed.

I would like to extend my Gratitude to my late uncle Dr. Ravindra Parchuri for his support and encouragement.
1 Introduction

1.1 Context

Many organisations invest large amounts of money to secure the data of their consumers. If the organization loses its customers data like a credit card number or the identity details, it is likely the organization to lose its customers, such as with TkMaxx (IT ProA, 2008). Thus, it is important for organizations to take care about the data of the customers and to maintain the company profile properly. Normally organizations store their customers and employees profile in databases it is important for the organization to protect the database from all the possible attacks.

With an SQL injection, attack the intruder can construct a malicious URL, which contains SQL keywords to attack the database. This is typically caused when the middleware has not properly checked the incoming URL. SQL injection is thus one a type of attack that the administrator cannot identify easily, until the administrator receives mail from the consumers saying that their credit card details are in non-legitimate hands because of the poor database structure of the organization. This is a growing problem, as in the first six months of 2008 IT ProA highlighted a massive increase in SQL injection attacks. They highlight that Microsoft was responsible for a large-scale attack on 500,000 web servers which involved an SQL injection attack. Other attacks in 2008 have included the NHS and the UN. Tools such as RealPlayer have also been used as an agent of these attacks. A highlight figure is that one page is compromised every five seconds on the Internet (IT ProB, 2008).

Autoweb.co.uk a U.K based advertising and marketing website is attacked by SQL injection in May 2008. The attack has done by injecting a 30 characters to overwrite the comments, by that the attacker able to gain access over the Microsoft SQL database. (Network World, 2008).

Databases stores important information related to the organization like consumer credit card numbers, usernames and passwords of the employees and consumers, and it is important for the company to protect the database. If the database is attacked all the information in the database can be lost and this results a big loss to the organization and its customers also. It is thus important to protect the data that is stored in the databases in the coming sections we evaluate how the databases work and how an intruder can attack these databases and the type of the attacks that a hacker can do on the databases, by evaluating these strategies this thesis thus produces a solution to reduce these attacks on the databases.

“Whatever can go wrong Will Go wrong and at the worst possible time in the Worst possible way.” (Murphy’s Law)

1.2 Aim

The aim of this project is to detect SQL injection attacks that an intruder is trying to do on the database when the code is running as a part of application executing. It includes a number of objectives:
Review the database structure.
Review the types of attacks that can be done on the databases.
Review how the SQL injection attacks can be done practically.
Develop an application that can detect the SQL injection attacks.
Perform an evaluation of the application.
Propose future developments that can be implemented in the application.

1.3 Background

The Internet became an integral part of human life and many enterprises dependent on it in different ways like storing employee profiles, accessing the files on remote servers and maintaining user information, and so on. The Internet is also an inexpensive solution for the enterprises to maintain a wide area network, and individual use the Internet for many other uses like shopping, meeting friends, reading news, and so on. Due to the rapid developments in Internet transfer speeds and the flexibility depending on the web applications is improved a lot. Because of the extensive use of internet in day-to-day life it became easier for hackers to attack on personal computers and to theft identity information like credit cards and personnel files.

The enterprises can protect their employees by taking high security measures like one-time passwords and unique identification numbers. However, for a normal consumer it is more likely to lose their personal information due to the attacks on internet.

Due to the pressure on the employees who are developing the application, they try to deliver the application quickly more than considering about all the security measures that need to consider when developing the application. This leads the program to be vulnerable to internet attacks.

One type of the attack need to be considered when developing is SQL injection by which the hacker can attack the background database application and get the credit card details of a customer to use it for unauthorized transactions. The SQL injection attacks are done on the internet applications more than the intranet applications.

Normally the administrator would not able to recognize that there is an attack happened on the database, because of the fact that the hacker can execute the SQL command as normal user. As the SQL injection, attack is executed as normal script that is executed by the application it is highly difficult for the administrator that there is an attack is running on the background.

1.4 Thesis layout

Chapter 1 Introduction. This chapter outlines the background technologies, defining the aim and objectives.

Chapter 2 Background. This chapter provides information about background technologies that are used in the thesis. This also explains about how the database structures and how the database tables are constructed.

Chapter 3 Design. Proposes the frame work that is required to design the application. Discusses about the technologies that are involved and
how the injection attacks can be performed on the database by examples.

Chapter 4 **Implementation.** Explains about how the application is implemented depending the framework that is described in chapter 3. Code snippets are included in this chapter to explain the application structure.

Chapter 5 **Evaluation.** The implemented application results are analyzed to calculate the values that are required for functioning of the application.

Chapter 6 **Conclusion.** The chapter summarizes the work that is performed in the thesis. Also provides a critical analysis on the application. Future work and developments that can be performed on this application are explained on this chapter.

Chapter 7 **References.** Documentation of resources that are used as a part of application development.

Chapter 8 **Appendix.** Includes the coding of all the application that is used, screenshots of the application is included.
2 Background

2.1 Introduction
This chapter discusses and outlines the theory that is required to understand the application. The application uses technologies like SQL Server, C# programming. To understand this application theory all the basic requirements are explained in this chapter. It begins with examining the core components of the TCP/IP protocol, HTTP message formats and the communication between a server and an end user. The theory discusses about the SQL language, which is used for the database programming. This chapter also provides a brief explanation about the database administration and the WinPcap application, which is used for capturing the network traffic.

2.2 TCP/IP
Transmission Control Protocol/Internet Protocol is a protocol, which is widely used in day-to-day internet communication. This Protocol is known as TCP Segment and IP datagram. TCP is a protocol, which is for reliable, connection-oriented, acknowledged data communication. TCP protocol is a byte stream protocol. The data flowing between the hosts is called as TCP segment. This protocol ensures reliable data communication between two nodes depending on proper sequence numbers.

The protocol ensures to give a reliable data communication by using Positive Acknowledgement with Retransmission (PAR) technology. PAR is a technology that resends the data when it does not hear an acknowledgement from the destination. When it says a connection-oriented Protocol it means the two peers using the TCP application should have a connection when transferring the data.

A process called “handshake” is performed in the process of connection orientation. Three way handshakes is a process when node a wants to communicate with node B, then it first sends a SYN (Synchronization) segment with its IP address. When the SYN segment arrives, at node B it replies with SYN+ACK segment, which sends the IP address of the Port B. When the SYN+ACK segment arrives at the node A, it establishes the connection and then sends the ACK segment with the data it wants to transfer to node B. This process is called as three-way handshake.

2.2.1 TCP Segment Format
Figure 2.1 specifies the segment format of the TCP, which makes for reliable communication between two nodes. It specifies the Source and Destination Port address, the sequence number, and the acknowledgement number. By using these fields, the protocol can provide reliable communication between the two nodes. The initial sequence number is the number that says the position of the data segment and it specifies the acknowledgement number that is expected by the sender of the segment.

Offset specifies the number of 32-bit word in TCP header. Reserved is a 6-Bit field that is reserved for future use. Control bits are the field that is used for controlling the connection between the two ports. When the control bit is set to URG, the urgent pointer field has to be supplied. Window is the field that specifies the number of octets. Checksum is the control field that covers the header and data fields. Urgent
pointer is the sequence number of the octet following urgent data. Options are available for different types of functions. Octets are appended to the header that ends on 32-bit word boundary and the next field is the data field once the TCP segment is passed to another node it validates the entire above field and then the application validates data.

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Number</td>
<td>Acknowledgement Number</td>
</tr>
<tr>
<td>Offset</td>
<td>Reserved</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent Pointer</td>
</tr>
<tr>
<td>Options</td>
<td>Padding</td>
</tr>
</tbody>
</table>

![Figure 2.1: TCP Segment Format](image)

### 2.2.2 IP Datagram

IP Datagram is the protocol, used to communicate between two peers over the internet. A TCP protocol can do a successful transmission between two peers when they are directly connected. The IP Datagram helps a TCP segment to flow between two computers in the network, by validating the source IP address and the destination IP address.

The IP datagram forwards the packet directly to the host by validating the destination address. When it cannot identify the destination address, it forwards the packet to a gateway, which then forwards the packet to destination. It can do the reliable delivery by importing the transmission protocols like Transmission Control Protocol and User Datagram Protocol. By importing any of these protocols, the IP datagram can make a reliable communication over the network.

Version is the field, which tells about the version of the IP frame (IPv4, IPv6). IHL is the IP header length that can be used to extend the IP datagram. Type of service is the field that can be used to indicate what type of service the IP datagram is expecting. Total length indicates the length of datagram in octets up to 65,535.

Identification field used to resemble the fragmented packets. Flag is the field, which tells about the fragmentation of the packet. If the flag field is set to one, it means it is having more fragments when it is set to zero it is the last fragment. Fragmented offset indicates the position in the datagram when the packet is fragments. Time to live is the field, which tells about the maximum time the IP packet can remain in the network. If the TTL field is zero the packet is destroyed. The TTL value is reduced when it is passing from one hop to another. The protocol specifies the type of protocol that is used in the data portion such as TCP, UDP. Header checksum is the field that is mainly for the header, when the header is modified the header checksum value must be calculated again. Source address and Destination address are the fields, which tell about the IP address of the source machine and the destination. Options are the field that contains more options. Padding is the field that is filled with zero’s to make the size of header as a 32-bit multiple.

The TCP segment depends on the IP datagram to make a successful communication over the network. While the IP datagram ensures the packets to reach the destination,
the TCP segment ensures reliable transmission. TCP is used between peer-to-peer communications where the data must be delivered. When the data is not delivered, the source retransmits the packets, as it was not able to receive the acknowledgement from the destination. When the delivery is not important factor to consider the data is sent to the receiver or not (In broadcasting) UDP protocol can be used. The IP packet format is represented in figure 2.2.

![Figure 2.2: IP Packet Format](image)

### 2.3 HTTP Protocol

Hypertext Transmission Protocol is the commonly used protocol in day-to-day internet communications. HTTP protocol works in the application layer and can be used for hypermedia information systems. HTTP protocol is in use from 1990 for data transfer between the server and the client. HTTP communication is done in a form of MIME like messages and the protocol communication is done by using the TCP/IP protocol. The default port for HTTP communication is port 80. However, the communication is not limited to only this port. According to the requirements, the HTTP protocol can use other ports for communication.

HTTP 1.1 is the current version of protocol, which uses same port for many request/response exchanges. The older version of HTTP protocol (HTTP 1.0) uses different ports for each request/response exchanges. The communication in HTTP protocol is done in form of request/response messages. Request is the message where a client requests for some information from the server and the server responds to the message in response format. Figure 2.3. Represents simple request and response chain between end-user and the server.

![Figure 2.3: Request and Response Chain](image)

The URI (Uniform Resource Identifier) strings do HTTP communication, they are known by many names like www addresses, Universal Document identifiers and Uniform Resource identifiers. The URI is a formatted string, which is used to find the resource through name or the location. HTTP is a form used to locate the network resources.
Http syntax looks like:
http://host:port[absolute path [? query]]

In the syntax http is the protocol, which is connecting to the host through the port specified (if the port not specified it uses port 80) and the absolute path is the address to reach the resource. The query can be any other parameter or process, which requests the server to perform an operation for requesting details for a particular entity. A query is called as an argument, crafted to run particular process in the server. An argument can be passed to server in many formats depend the process required.

An argument is passed by using the special characters like |“(” | “)”| | “:” | | “\”| | “/”| | “?”| | “=”| these are some of the common special characters, which are sent to the server to process a request. HTTP communication uses product tokens to identify the user version and the server version of the software to make any conversions if necessary. They also use language tags to identify the language that the end user working with.

### 2.3.1 HTTP Message Formats

The HTTP communication is done in Request/Response format. HTTP protocol contains different headers called as Message-Header, Field-name, Field-Value and Field-Content. The headers are to ensure proper communication between the end-user and the server. It also contains the IP address of the server and client to ensure proper data communication.

**Request.** A request is generated from client to server, which includes the request line, general header, request header and entity header. A request line includes one of the methods that are defined in table 2.1, and the absolute URI to reach the destination. If the request header filed does not includes the absolute URI the host is determined by the host-header field. If the request message does not includes a proper URI string the server responds with a Bad Request error.

**Response.** Response is the message that is generated by the server as a reply to request message from the client. The response includes a status header, which informs about the status of the request and normal fields like General-Header, Response-Header and the Message Header. Http message includes different status codes depending on how the process is executed and whether it needs any more information from the client. The status codes of the response messages are defined in table 2.2.

### 2.3.2 HTTP Methods

HTTP Protocol defines eight methods to identify the process the user needs to execute. The methods used by the HTTP protocol are defined in table 2.1.
Table 2.1. HTTP Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Operation Performed by invoking the method</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Get requests a representation for a specified resource.</td>
</tr>
<tr>
<td>POST</td>
<td>Post submits data to an identified resource, which needs to be processed.</td>
</tr>
<tr>
<td>HEAD</td>
<td>Head is the similar method to get it requests data without the message. It requests the message like meta-information, which is included in the response headers.</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Connect is the method used to convert a connection from one format to another format. This method is used when converting normal connection to a SSL (Secure Socket Layer) connection.</td>
</tr>
<tr>
<td>OPTION</td>
<td>This method is used to validate the server functionality.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Delete is the method, which is used to delete a specified resource in the web server.</td>
</tr>
<tr>
<td>TRACE</td>
<td>Trace id is the method, which echoes the request that a client sends to the server so that the client can validate whether there is any interference happens to the message in communication.</td>
</tr>
<tr>
<td>PUT</td>
<td>Put uploads a specified resource in a particular location.</td>
</tr>
</tbody>
</table>

2.3.3 HTTP Response codes:

When a request is sent to the server, the server posts back a response code depending on how the process done in the server. To evaluate the HTTP messages, status codes helps us to identify what type of error the server is generated. This helps to identify the fake requests/ responses strings that are generated from the client machines. The status codes in HTTP are defined in three-integer number, as defined in Table 2.2.

Table 2.2: HTTP Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1XX</td>
<td>Indicates the server has received the request and it was processing the request.</td>
</tr>
<tr>
<td>2XX</td>
<td>Indicates success when the request is successfully received, processed and the request is accepted.</td>
</tr>
<tr>
<td>3XX</td>
<td>Indicates a redirection, It tells that further action may be necessary to reach the resource.</td>
</tr>
<tr>
<td>4XX</td>
<td>Indicates an error from the client machine, Like bad syntax or when the request is not properly created.</td>
</tr>
</tbody>
</table>
This indicates an error from server side, in the case when the server does not fulfill a valid request.

Sample request and response communication:

```
HTTP/1.1 302 Found
    Location: http://www.google.co.uk/ig?hl=en
    Content-Type: text/html
    Content-Length: 229
    Date: Wed, 10 Jan 2007 21:02:09 GMT

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>...
</head><body>
<h1>302 Moved</h1>
The document has moved
<a href="http://www.google.co.uk/ig?hl=en">here</a>.
</body></html>
```

Figure 2.4: Sample Request and Response Communication.

```
GET / HTTP/1.1
Host: www.google.co.uk
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; SLQ; .NET CLR 1.1.4322; InfoPath.2; .NET CLR 2.0.50727)
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Accept-Language: en-gb
Accept-Charset: ISO-8859-1, windows-1252
Connection: Keep-Alive
Cookie: PREF=406-9794869335161946; _GAL=1200159599; UFID=1200235897928321; _GAT=1; h=7-fqBgs4_BoKvyv0dnl

```

Figure 2.5: Sample HTTP Request Headers.

## 2.4 Databases

Database is a collection of data or records stored in computer in a structured format. Databases used nowadays are in relational databases format, where the data in a table is mutually related to the data in another table. This model is used to reduce the complexity to gather records from database. Databases contains tables with unique names with respect to the database, each table contains records of the data.

There are different types of databases are in use. In general all the databases basic functionality is same (to store the data). But users prefer to use different types of databases depending on the application compatibility and the speed that the database can execute the queries.

Most popular database systems are Oracle, MS SQL Server, and MySQL. Although
these three database systems use different formats to store the data and different protocols to communicate with the application, structured querying language (SQL) is used to view and modify the records.

The dynamic web applications like Shopping, mail, company intranet applications and so on uses database as a background application to store the data of the customer, employees. Databases create a record for user information that is supplied and stores the data in a structured relational format. This type of storage offers flexibility to the application to obtain particular record of an user without any complexity.

2.5 Structured Query Language (SQL)

SQL language is ANSI (American National Standards Institute) language that is used to perform operations against the database. It uses different query formats to Insert, delete, modify and viewing records from a database. SQL is the universal query language to perform actions over database systems. SQL language supports all the major database systems like MS Access, SQL Server, Oracle, DB2 and MYSQL. Though SQL language is standard there are many different versions in use. All the versions import the basic key words of SQL like Insert, Delete, and Modify etc.

There are many types of functions used in structured query language to perform operations against the databases. SQL language can be differentiated as two types, which are data manipulation language and data definition language. Data manipulation language contains the commands that are used to perform operations like selection, insertion, deletion of records in database tables. Data definition language commands are used to define the structure of the database and how the data has to be stored in particular database.

2.5.1 Data Manipulation Language

These are the statements used to perform operations against data in database records. There are four types of commands that are used for data manipulation. The statements are explained in table 2.3.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Operation performed by the keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select</td>
<td>This is the command used to Select particular information from the database. This statement is commonly used to pull the records from database.</td>
</tr>
<tr>
<td>Insert</td>
<td>This statement is mainly used to insert records into the table like new records.</td>
</tr>
<tr>
<td>Delete</td>
<td>This statement is used to delete records from the database tables.</td>
</tr>
<tr>
<td>Update</td>
<td>This statement is used to update any existing records in the database with new values.</td>
</tr>
</tbody>
</table>
2.5.2 Data Definition language

Data definition language statements are used to create new data structures for the records. These commands are used to perform operations against the structure of the database. The SQL statements are formed with one or more of the commands that are defined in Data Manipulation Language. The key commands that are used to do perform operations against the database structures are listed in table 2.4.

Table 2.4. SQL Data Definition Language Statements (MSDNb, 2008)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Operation performed by the keyword.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>Create statement can be used to create any new object in the database such as a new user, new database and new table.</td>
</tr>
<tr>
<td>Alter</td>
<td>Alter command is used to alter the properties of the existing user or the database or the table.</td>
</tr>
<tr>
<td>Drop</td>
<td>Drop statement is used to completely delete the records from the databases, this includes deleting the user or database or the tables.</td>
</tr>
</tbody>
</table>

2.5.3 Creating Databases

It is easier to understand the SQL language by observing some sample commands. This part explains some code snippets which explain how to create databases, database tables and storing user information in the tables.

# Code Snippet

```sql
#Creating database called Napier
Create database Napier.
#Creating table Students with some values.
Create table students (mat_number int, stu_name varchar (20)).
(This command is used to create students record in napier database with mat_number of type integer and stu_name of type character number 20 is specified that the stu_name record can be up to 20 characters).
```

# Code Snippet

```
#Inserting a record.
Insert into students (mat_number,stu_name) values(10001,'john');
(This statement inserts the data into students table with the values provided).
```

# Code Snippet

```
#Selecting records.
Select * from students;
(A select statement is used to view the record from the database. This can be passed with different arguments to view limited data also).
```

#Code Snippet

```
#Deleting data
Delete from students where mat_number=100001
(This command is executed to delete data from the table students where the mat_number is equal to 100001)
```
The code snippets are some of the basic functions that are performed against the database to maintain records. Structured query language uses special characters to indicate particular data types like characters and comments in query language. The attacker tries to use these special formatting characters in SQL language to attack against the database. A sample database table is represented in Figure 2.6 with some values.

![Figure 2.6. Select statement output.](image)

### 2.5.4 Operators and characters in SQL

SQL uses different types of operators and characters to get records from the database effectively. Logical operators are important functions in SQL that are used to obtain records from the database, when the logical condition evaluated as true. These logical operators test for a condition whether it is true or not. The operators return a Boolean value which is “true”, “false” or “unknown”. Some of the logical operators performs vital role in attackers script. The logical operators are used to pass execute the statement as true, where the real value of the command is false. The important SQL operators are listed in Table 2.4.

Special characters are the characters which are used in SQL language to specify that the user is passing some special data for execution. An SQL server script can take the values that has to be executed in the database server when it is properly inserted between single quotes (‘). The semicolon (;) statement indicates end of an SQL statement but these types of special characters can be used by the attackers to embed a new string to inject malicious code.

The (--) double dash indicates a comment statement in SQL language. An attacker can use this operator to enter into the database maliciously by making the original command as comment statement. The attackers also uses logical statements like 1=1 and a=a that always evaluates as true. When the code is evaluated against the database it always results with true value, the database assumes that the attacker has passed valid credentials and returns the records that the malicious user requested for.
Table 2.5. SQL Logical Operators (MSDNc, 2008)

<table>
<thead>
<tr>
<th>Logical Operator</th>
<th>Action performed by the operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>This return true if a set of comparisons are true.</td>
</tr>
<tr>
<td>And</td>
<td>This returns true if both of the expressions are true.</td>
</tr>
<tr>
<td>Between</td>
<td>This returns true if the value is within the range of condition specified.</td>
</tr>
<tr>
<td>Exists</td>
<td>Returns true if the query contains any rows.</td>
</tr>
<tr>
<td>Not</td>
<td>This operator can reverse of the Boolean operator.</td>
</tr>
<tr>
<td>Or</td>
<td>Returns true if any of the condition is satisfied.</td>
</tr>
</tbody>
</table>

2.6 Database Administration

Database administration takes very important role to avoid attacks against database servers. Database administration deals with maintaining integrity and security to the records. If proper planning is not done when creating the applications it can do severe damage to the enterprise. For example, when installing a database the database prompts for administrator login. Most of the users use the default passwords that are provided by the database system. Like user “SA” for the MS SQL server. The “SA” user is created automatically when creating a database and the default login has the great privileges and this is unalterable, by gaining access to the user “SA” the attacker is able to perform any kind of operation against the database. So it is advisable that the administrator doesn’t use the “SA” login for routine checked in the databases. When the database is configured with these types of default accounts it makes attacker work easy to penetrate into database system.

To reduce the risk of SQL injection attacks the administrator can create many numbers of accounts for each users and the database system allows the administrator to configure privileges for user depending his role in the organization. It is advisable to use less privilege user login for normal data applications like database lookup’s without updating or need to change the data. These are some of the things that have to be considered when creating a database application. Database administrators configure all the settings to the database so by taking preventive measures the application can be protected from attacks.

2.7 WinPcap

WinPcap is a tool that is used for capturing data packets in windows environments. This tool is equivalent to the tool “TCPDUMP” that is used in Linux operating systems. The application allows the packets to transmit and bypass the protocol stack. This also has additional futures like Kernel-level packet capturing and network statics engine to support remote packet capturing.

WinPcap is used as the network interface by many tools, which are used for network packet scanning, network monitoring, traffic generators and network intrusion
detection systems and so on. Snort and Wireshark are some of the popular tools that use WinPcap as the core tool to monitor the network traffic. WinPcap can be integrated with many applications like Java, C# to analyze the packets. This is a popular tool which is used in many open source and commercial network scanning applications.

The tool is used for capturing raw data that is flowing in the network. It can scan the data that is coming into the system and the data that is going to other hosts also. It can gather the statistical information about the network traffic there by making easier to analyze the data. WinPcap can also filters the data according to the rules that are created by the user.

WinPcap can scan any type of network devices that is presented in the system. Though it can get details of all the network adaptors, it is having some problems for scanning the packets that are flowing through the wireless adaptors. The application just captures the packets and shows the details but it cannot stop the packets to reach the application. It can filter the packets that are sniffed. For sniffing the network packets through winpcap it is essential to set the network device in promiscuous mode. If the network adaptors is not set to promiscuous mode WinPcap can’t see the packets that are designated to particular machine though it can show the packets that are broadcast and multicast.

This thesis uses SharpPcap which is a tool developed based on WinPcap. SharpPcap tool provides API for capturing, injecting the packets by .net framework and the associated programming languages.

### 2.8 Conclusions

This chapter outlines and describes about the latest technologies that are required to understand the thesis. It is explained how the actual data communication happens between the server and an end-user then it discuss about the HTTP protocol. The theory also presented with several applications and commands that are used with the SQL language, it also explains about WinPcap tool, which is used for many network sniffers in the industry.

The application Winpcap is selected to capture the network traffic because this is the tool that is widely used in the industrial applications to capture the network traffic. The request and response chain communications are important to understand the data that has to be captures. It also explains different keywords that are used in HTTP protocol and SQL server, this chapter provides explanation about the traffic that has to be captured and the keywords that are to be observed to detect SQL injection attacks.
3 Design

3.1 Introduction

Software programs often need to use different languages in a single program. Such as a web application that is developed by using C-Sharp program use SQL queries to perform operations against database. When constructing these queries programmers often do not consider about the software flaws. This type of programming leads an attacker to attack on the database server by injecting specially crafted strings. These types of attacks are called as injection attacks.

Nowadays people depending on web applications are improved to do shopping to make reservations, to check mails and so on. When user’s utilizing the web applications, they believe that the applications are reliable to provide with credit card information and some sensitive personal information. All the information that is given by the end user is stored in database in the server.

Due to the low reliability of the web applications they became targets to the attackers to obtain personal information and the credit card numbers. The attackers try to attack the database which acts as the storage space for all the user information. One of the serious threats for the database attacks are SQL injection.

3.2 Outline of SQL injection

SQL injection is a type of attack where the attacker can execute the SQL script in application layer of the program. This makes the application to think that the code is executed in normal operation of the application. And it doesn’t show any errors to the administrator, as the code is executed as a part of the application. When the attacker is injecting carefully crafted malicious code at application layer it is impossible for the administrator to identify the attack as the application performs the operations without any errors.

In SQL injection attack an attacker executes the SQL script as normal script that is generated by application to perform an action against the database. An attacker can do this by doing string manipulation in the application, or by constructing a new structured string to execute within the application. When the string is executed as a part of the application the database recognizes a query that is generated by the application requesting for some information or performing manipulations against the database. When a attacker gains access to the database by the application it makes the attackers work easy to execute more number of malicious queries against the database.

By using this attack an attacker can penetrate through the database and he can also break the administrator login identities against database. This provides more privileges to user to do any type of action against the database. At this stage the attacker can leak any information, the attacker can also drop important information from the database, the hacker can also corrupt the database. This attack cannot be recognized by the administrator when it is running as it is identified as normal application execution, so this attack poses serious threat to web applications.

The SQL injection is very easy to perform by validating the structure of the database.
The attacker just needs a web browser to perform these kind of attacks. Once the attacker is able to find the way to gain entry into the database he can perform any action against the database unless the user account created for the application have less privileges to perform actions over the database.

### 3.3 Recent Examples of SQL Injection attacks

Mass SQL Injection attacks become a severe threat to many companies where as the attacker can perform SQL Injection attacks on many websites at a time. According to SC Magazine (Carr. J, 08 January 2008), an automated SQL Injection attack is performed on 70,000 websites in 1st week of January 2008.

According to Allen Wilson (Washkuch, 2006), research vice president in secure works “SQL injection attacks are becoming more popular because they are successful and easy to orchestrate”. He also claims that there is an increasing demand for identity information stored in databases. Hackers perform SQL injection attacks because they need just a web browser no assemblies required to perform these attacks.

E-Crime (e-Crime, 2006) survey claims that the SQL injection attacks often leads to financial loss to 40% of organizations. The survey also reveals that the companies lost higher than $700,000 on average. These figures tell us how important is to stop attacks on the web services.

The SQL injection attacks are not only limited to low profile organization. In June 2007 the software giant Micro soft website is defaced by using SQL injection attacks the Hacker manipulated the Cascading Style Sheets which are used to styling the websites, (Zone-H). By capturing an error from debugging the application the attacker changed the website front page.

According to SECREN0 (Scerno, the People VS e-commerce consumer Attitudes to data Security) an organization which works for providing security for databases 25 million child benefits records are stolen by a junior officer from HM Revenue and Customs as he is able to download the details from the database. It also says that the SQL injection attacks are increasing at around 250% per year.

The examples show the importance to effectively stop the SQL injection attacks. The attacks not only leads to financial loss to organization they also leads the customers to lose faith on the organization. According to SECREN0 more than 50% of the people don’t come back when the website is attacked once. Only 45% of the people think that the data is protected by the organizations.

### 3.4 Web Application Processing

Figure 3.1 shows how a web application is processed, when the user requested some information from the server. A user can reach the server by the URL address. A URL contains parts for addressing to the server or a database or other application.

Consider a sample string (http://www.abcd.com/show.asp?user=xyz). Consider this is a string the user xyz is used to request the server to see the page show.asp. The string is passed through the web browser from the end user computer. Internet works as a communication channel between the End-user machine and the server.

The communication string is formed with different elements each required to process...
at different stages in the server. Considering the same sample string each part can be explained as, the HTTP is the protocol specification which validates at the firewall. The firewall detects if there is any other improper protocol is used for communication. If firewall detects usage of an improper protocol then the firewall stops the communication. If the request is a known and authentic protocol the firewall simply allows the string to execute in the server.

A web server is the application execution machine that executes the user’s request. Normally every web server works by accepting the HTTP strings and responding in the form of HTTP strings. A web server can be simply called as host. The host validates the request and passes the request to application server. Application server is the component which is used to Process the requested application from the server. In an HTTP application all the requested web pages are processed by application server.

The application server processes the requested information by the server. The application server is the program that a programmer creates relationship with the Databases and all other external applications that are to be processed by the application. The application servers are written in different languages like asp, java, .NET, jsp and html. The language used is depends on the user flexibility to programming and the flexibility for relating with other applications.

The application server then passes the request to the database and the database responds to the application server with the requested information. The application server then passes the information to the user over the internet.

During the application execution process the firewall is the only point which is used to allow or deny the network traffic passing through it. The firewall can normally stop any traffic depending on the set of rules created by the operating system or by the firewall application creator. The web server and application server doesn’t verify any security threat except the user credentials like the username and passwords.

While the application server is process the attack can only be stopped at the stage of firewall. But firewall application is not an intelligent application to stop the attacks that are coming as a request to the server. The request from the end-user is passed to the server is in the form of normal URL string. As the request is passed to the server in a form of string the firewall will not able to stop the attack before it reaches the database for execution.

As the SQL injection attack passes through all the stages as like a normal request from genuine user the core components of the server may not be able to detect the attack on the database. So the web server requires a system which can efficiently observe all the traffic that is coming from the user and to analyzing them for malicious activity. Figure 3.1 shows the simple communication between an end-user and the server.
3.5 SQL Injection Types

SQL injection can be classified into four different types according to the type of the string the attacker is using to attack the database application.

- SQL manipulation.
- Code Injection.
- Function Call Injection.
- Buffer overflows.

3.5.1 SQL Manipulation

SQL Manipulation is the basic and common type of attack that an attacker tries to use against the database. In this type of attack an attacker tries to manipulate the original code that is passed by the application by adding some extra information to it. In this type of attack the attacker also uses the key words like “UNION”, “INTERSECT”, or “MINUS”.

Example:

Consider a login application which uses the database table shown in figure 3.2 (name auth). When the application is processed for the id_num, the application validates u_name and pass of the user. Upon passing valid login credentials the database returns Id_num for the user.

<table>
<thead>
<tr>
<th>u_name</th>
<th>Pass</th>
<th>Id_num</th>
</tr>
</thead>
<tbody>
<tr>
<td>john</td>
<td>john</td>
<td>1001</td>
</tr>
<tr>
<td>peter</td>
<td>peter</td>
<td>1002</td>
</tr>
<tr>
<td>kevin</td>
<td>kevin</td>
<td>1003</td>
</tr>
<tr>
<td>rachel</td>
<td>rachel</td>
<td>1004</td>
</tr>
<tr>
<td>jenny</td>
<td>jenny</td>
<td>1005</td>
</tr>
</tbody>
</table>

Consider the following SQL code used for validating:  
```sql
Select Id_num from auth where u_name= 'peter' and Pass= 'peter';
```
(This command returns the id number of peter).
The code which is used for validating the data had a “where” clause and it is constructed with the “and” operator. So to satisfy the above condition both of the credentials should be same. If any of the credential is wrong this command results an error.

The attacker can pass a valid user name and construct another statement to prove both conditions are true. The code an attacker can be constructed is. By gaining the Id_num from the database the application thinks that it was used by peter and the attacker is provided with all the resources that peter can use.

```
#Injection Code.
select id_num from auth where u_name= 'peter' and Pass= 'peter' or 'a'='a';
(This code validates against the database as the u_name is peter and to satisfy the AND operator the attacker used OR operator and it validates as true. Upon validating this code the query returns the id_num to the application.)
```

### 3.5.2 Code Injection

Code injection is same as the attack type of SQL Manipulation. But in this type of attack the attacker adds crafted SQL script to change the properties of user to delete user account. Considering the database server can execute consequent strings. By using code injection the attacker who is gained access to “peter” login can delete another user account.

```
#Code Snippet
SELECT Id_num FROM auth WHERE u_name= 'peter' AND Pass= 'peter' OR 'a'='a';
DELETE FROM Auth WHERE u_name= 'Kevin';
This code provides the attacker with access to peter credentials and it also deletes the user Kevin from database.
```

### 3.5.3 Function Call Injection

A database comes with built in functions to perform certain operations over the database. Or the database server allows a user to create custom functions to perform many operations against the database tables. The functions are very critical to the database as they perform actions like connection to networks. The functions which are used in the databases for proper functioning can be used by an attacker to attack the database. Custom functions are easy way to gain access by the attacker to inject malicious code in the database.

The attacker can use the functions to send the database to another location in the network. The functions are called by the attacker by using normal SQL statements like SELECT, INSERT, DELETE and MODIFY. The functions that are called by using select statement doesn’t do much harm to the database but the functions that are executed by insert, modify and delete statements can change the data in the database tables.

A simple example is by using the SQL translate statement that the user input can be converted into another format, by passing different code in the user input section the user can change the database structure or the attacker can also add new user to the database and change the passwords of existing users for gaining access over the database.
Example:

```sql
# Consider “Translate” function in oracle. This function is used to translate user input into another format.
SELECT Translate (string1, string_to_replace, Replace String) from table_name;

# The translate statement replaces the characters matching with the string_to_replace characters in string 1 with the respective characters in replace string.
TRANSLATE ('123ier', '123', 'nap'); this function returns "napier"

# The attacker can manipulate this statement to pass some different input to the database.
Translate (''|| myappadmin.adduser ('admin', 'password') ||' ', '123', 'nap')
```

The attacker uses a custom function “myappadmin.adduser” to add new user to the database. When the new user is added to the database it gives the attacker full control over the database. An attacker can also use web address to redirect the user to another server.

### 3.5.4 Buffer overflow

In the buffer overflow attack the user passes more characters to the database, where the number of input characters is limited by the databases. This action can overflow allocated buffer and overwrites adjacent locations in the memory. By crafting the input carefully the attacker can gain access over the database or this type of attack can also confuse the database, thus the database can shutdown unexpectedly.

Buffer overflow attacks are normally crafted by making use of security loop holes in the programming. So by installing updated security patches to the databases the administrator can stop these types of attacks.

Some of the databases are updated with new technologies can stop any connections coming from the application when a buffer overflow attack is triggered. By making use of this the attacker can trigger the buffer overflow attacks many times. This results the database to stop acting against any commands that are to be executed by the application. This attack can effect normal operation of the application.

Bind Variables are another concept that a developer can use to save system resources and to reduce the application execution time. When a command is used against a database the command is saved in shared pool. When a SQL command is passed to database by the application, the database checks in the shared pool to verify whether the command is executed previously or not. If the command is not executed before the database goes through all the process to execute the command, if the database is able to find the result in the shared pool it directly uses the result that is stored in shared pool to response to the database query.

So to save the application resources developer’s uses bind variables with the SQL statements. The attackers can try to manipulate the bind variables to execute applications maliciously. Generally oracle is immune to this type of attacks as oracle will use the value of bind variables exclusively. And the oracle database works such as not to reveal any value from the database, when there are no matching values.
3.6 Existing Technologies to stop SQL Injection

3.6.1 Defensive Programming

Defensive Programming is a Programming practice that was done on the integrated application code when the software is in development stage. The programmer tries to minimize all the bugs in the programming, and the programmer tries to find out the way to use the code for hacking purpose. So by this type of coding practice the programmer will be able to find out the security weakness in the code. By securing the code the programmer can possibly stop potential attacks on the website.

The code can be analyzed in many ways like reducing the complexity of the program. Doing reviews on the code again and again to find out the possible vulnerabilities of the code and to perform software testing on the code. By forming the programming with the above measures the programmer can develop a code which may be immune.

Ultimately the attacker tries to find out new ways to penetrate into the code. When the hacker finds a new method that is not tested while application programming the attacker may be successful at some stage. The user also limited to test the application with the attacking techniques he knows about.

3.6.2 Anomaly Detection

Anomaly detection technique is a method where the administrator observes the network traffic. By observing the network traffic the administrator can find when there is a possible attack performed against the server. The anomaly system verifies the traffic which is going through the network by analyzing the recorded behavior with the network traffic there is possibility to find out the attacks.

The anomaly detection is classified in many types like rule-based, model-based and statistical analysis. The programmer creates a set of rules to define possible types of attacks that can be performed on the program when the rule is not satisfied there might be a possible attack on the database. In the model based approach the application imports the anomaly techniques that are characterized to define attacks on the server. If the incoming traffic doesn’t meets the model, the application indicates there is a possible attack on the database. Statistical analysis is a different approach where the program calculates the system behavior by measuring certain variables overtime and it takes average point of the calculated variables. If the new traffic exceeds the thresholds, indicates there might be a possible attack going on the server.

The anomaly detection techniques are very good in detecting the attacks like Buffer overflows and different kind of attacks but as the user can be able to pass the data in a method which represents like normal traffic these techniques are unsuccessful for detection of SQL Injection attacks.

3.7 Simple SQL Injection Attack by Example

A simple SQL injection attack can be explained as running the SQL code which is not intended to run by the application. If the application is creating the SQL strings by using the user input it makes easier for the hacker to attack on the database. The user just needs a web browser to attack the SQL server.
To demonstrate simple SQL Injection a database table is created with the field names as Username (u_name), Password (pass), E-mail ID (email), Identity Number (id). The fields are filled with some random values. The database table is displayed in Figure 3.3.

The simple application created takes two variables as Username and Password. If the user passes valid credentials for the two variables then the application is designed to transfer the user to another page and display the user’s E-mail ID and Identity Number. The sample application uses the table that represented in figure 3.3.

Figure 3.3. SQL database Table.

Figure 3.4. Sample Login application

Figure 3.4. Represents a simple web application which running in the local server. Unlike the traditional password hiding this application is designed to display the password to demonstrate the SQL Injection attacks.

The database contains Paul as a username and the password for the user is Paul. When
the user Paul tries to login the webpage the application simply redirects user Paul to another page with the e-mail id as a variable in the html string. The page captures the email string from the html code then it shows the user e-mail id and identity number. The output is shown in figure 3.5. this output is generated when user ‘Paul’ passed valid credentials to the application.

Figure 3.5. Sample login application result.

While the application functionality is to validate the user input the hacker can easily craft new SQL string by taking the user variables as an advantage.

# Code Snippet

The SQL query designed to get the user details from the database is designed as:
```
SELECT email,id FROM users1 WHERE u_name='" + TextBox1.Text + "' and pass='" +TextBox2.Text+"';
```

on validating the user input from the textboxes the application gets the email id and identity number (id) from the database. When the valid user details are passed the application sends the string to the database with valid credentials like.

# Code Snippet

```
SELECT email,id FROM users1 where u_name='paul' and pass='paul';
```

As the username and password are valid credentials the database executes the command and returns the requested values. A hacker can easily craft the string which can always executes as true in the database by using the keywords. If the hacker injects the code as (' OR '1'='1) this allways validates true in the database and the database sends the responses to the application with the valid information.

# Code Snippet

The database string is crafted as:
```
SELECT email,id FROM users1 where u_name='paul' and pass='' or '1' = '1';
```

The code snippet shows a string that is injected with the malicious code and it allways executes as true in the database so the database assumes it is the credentials passed by valid user and responds with the requested information.
The hacker pass the credentials that are malicious, represented in Figure 3.6, to perform the injection operation. The hacker is trying to pass the command that always validates true in the database. The information database responded with is shown in Figure 3.7. Where the hacker gains access to the database with invalid credentials.

Let us consider that the application is designed such as that the application validates the user password and doesn’t allows the hackers to penetrate with wrong credentials. Then the hacker can take advantage of the email variable which is passed to second page when the user passes valid credentials. It is common that the website host will have an email id for admin like admin@xyz.com. Or a hacker can browse through the website and can find a contact email for a representative in the company. This can also help the hacker to penetrate into the website. The injection attack is represented in figure 3.8.
Figure 3.8. SQL injection attack Example.
The hacker entered an email id of an employee in the organization and the email id executes the application. The application considers it as a valid input and displays the requested information.

A hacker can inject the code in a way that the application couldn’t understand there by generating an exception the program this can help the hacker to get some details of the database table name and the fields in the table which helps the hacker to identify the design of the table. Hacker can craft the strings carefully to generate an error from the database. The simple command is represented in figure 3.9.

Figure 3.9. SQL injection Attack example 2.
The figure3.9, input generates an error in the application program and the programs responds with the error showing the background code that is used as a part of the application development. This helps the attacker to understand what the language the application is coded in is and it also helps the attacker to capture the table names and fields in the table, which can help the attacker to craft the strings accordingly.
Figure 3.10. Error produced by login Application
As the exception handling is not handled properly in the application the application
displays the commands that are used in background of the application. The simple
exception error can give the information like the table name different field names of
the database table which makes it easier for the hacker to penetrate into the database
easily.

3.8 Conclusion
This chapter explains about the SQL Injection attack in great detail. It is very
important to understand how the attack’s can take place on a web server and the
methodology’s the hacker can use to attack the web servers. It also explains the SQL
Injection process on a sample application to give a clear example for how the
Injection attacks takes place. It is also explained about usage of SQL strings in
different ways to attack the databases and it also discusses about the previous
technologies that are used for detecting the Injection attacks.

This part has reviewed how exactly the SQL Injection attacks can happen, and the
sample application gives great benefit for the reader to understand how exactly the
injection attacks done.
4 Implementation

4.1 Introduction

SQL injection can be prevented by taking good care in the application design and protecting every function and procedures that are used by web applications. The application designer must take good care about protecting every single SQL statement that is dynamically generated by web applications. If the statements are not well protected by the web application designer, a single statement can cause great damage to the database.

Strings for passing the usernames and passwords to the databases must be created carefully with bind variables. If the SQL is not crafted carefully using the bind variables, the value from the user is passed as a form of strings to the database this makes the attacker’s work easy to execute malicious SQL over the database. And the SQL statements should be created such as there is no concatenation in between SQL strings and parameters.

The Input against the database must be validated properly to stop attacks against the database. If the input is validated properly it is possible for the administrator to find malicious code that the attacker is trying to use against the databases. By using the special characters like single quotes the attacker can successfully enter into the database. It is recommended that all the special types of characters must be validated properly before the application passes them into the database.

4.2 Design Considerations

As discussed in previous chapters function is a very important program that is used to do critical actions over the database. By default the functions in the database are created as public to allow all the users to use them. As they are public the functions can also be used by the attacker. The functions can do critical operations such as changing administration rights to the users and changing user names and passwords for the databases. So it is recommended that the functions which are not used by the application are restricted with access limitations.

An end-user can send some commands to the database by inserting malicious code in the URL strings. So by validating the code that is coming to the server from the end user the SQL injection attacks can be determined easily.

Most applications prepare SQL statements with the strings which are already constructed by the programmer. The SQL statement takes the user inputs and it forms a string with valid data variables from the user. Then the string is executed against the database by using the application programming techniques. As the application gets the normal data to it in the form of URL strings the user data passed through it would be in a form of a string.

The ultimate aim for the attacker is to craft a SQL string which can be an executed against the database successfully. The attacker tries to use the keywords like ‘OR’, ‘AND’ which normally evaluates the executed conditions are true all the times. The attacker also uses the character Single Quote (’) which are used to specify a character string in SQL data type. By using the Single Quote character successfully the attacker
can close actual data that has to be passed to validate against the database, thereby creating a query that would execute against database.

SQL Server can execute a batch of queries at a time so if the attacker insert another query with the query that is used for the database execution. By using this facility the attacker can modify update or delete records in the database.

4.3  IDS Design

Intrusion Detection Systems (IDS) are the programs which are written to find possible threats against the application. Intrusion detection systems verify the traffic that is flowing through the server and it generates an alert when it finds there is possible type of attack against the server.

The intrusion detection systems designed to protect the application by monitoring the protocols that are used by the application traffic. An IDS which is developed to protect certain type of application is called as application protocol intrusion detection system. This normally sits with in the server and monitors the traffic and communication against some communication protocols.

The project proposes an application protocol intrusion detection system which is designed to scan the network traffic for particular keywords that are use to inject malicious code against the database. the intrusion detection system can be designed in such a way that it scans the network traffic coming into the server for particular ports and by scanning all the traffic that is coming from end-users the IDS can be designed to scan for particular keywords.

Once the IDS detects there is an attack going on the server it alerts the administrator to take preventive measures against the attack. By observing the request where it is coming from the administrator can able to block the attacker for some time period. There by taking care about the attack that is going on the database the administrator can modify the security settings to the server to stop potential attacks.

The proposed application is designed in such a way that the application scans all the network traffic that flows through particular ports. Then the application takes all the URL strings that are transmitted to the application. The URL strings are separated for each word with some delimiter characters. The application then scans for the keywords which are normally used to inject malicious code in the URL strings.

After validating for the malicious code the application assigns score of maliciousness for the URL strings when the URL string score crosses the threshold level of maliciousness score then the application considers that it is a malicious code that is injected by the user and it logs the malicious string into a log file. The administrator can verify the log file of the application of the application to take preventive measures to protect the database attacks.

4.3.1 Design Pre-requisites

The proposed IDS design is to scan all the network traffic and to evaluate all the URL strings that are flowing through the network. To scan the packets that are flowing through the network this project uses WinPCap Tool which is an industry standard tool to monitor all the network traffic. WinPcap is a standard network tool which is
used in many applications like Wire shark and snort which are designed to monitor the network traffic.

The proposed project coding is done in C# in Visual studio 2005. C# is an Object Oriented Language which is developed by Microsoft Corporation in the part of .net Development. The C# programming uses Microsoft .net Framework library to import many classes which are already designed to perform particular functions. This project uses .net Framework version 2.0 to use the classes that are already defined in the framework.

In some parts the proposed programming also uses SQL Server 2005 for testing purposes. SQL server is a database server which is used to manage the user database in the form of tables. The application is designed in such a way that the result is displayed in console format so that the application can be easily redesigned to integrate with many other browser applications like Internet Explorer and Mozilla Firefox.

The application is designed in such a way that the user needs to select the Network interface to monitor the traffic. The application can also be extended with more sets of SQL Injection keywords to effectively detect the Injection attack with regards to the customer own application.

### 4.3.2 SQL Keywords

The application requires validating the URL strings with different set of keywords that are used to perform Injection attack. The keywords are part of the SQL language which is normally used to perform operation on the tables.

The application scans the network traffic and it uses combination of keywords to effectively detect the SQL Injection attack. When the keyword combination is found the application assigns a weighted score to the URL string. If the weighted Score is greater than the threshold level the application automatically logs the packet to the Log file and alerts the user that there is an SQL Injection attack is going on the server.

The proposed IDS uses set of keywords from the table 4.1. On finding both keywords from the URL string the ids returns with a weighted score that is assigned to the keywords.

To assign weighted scores can be valued by considering some scenarios which set of keywords are most dangerous and which are not. The keywords that are to be observed by IDS are specified in table 4.1.

#### Table 4.1. SQL injection attack Keywords.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Operation Performed on the database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select</td>
<td>Select is the keyword that is used to capture data from the database tables.</td>
</tr>
<tr>
<td>Delete</td>
<td>This keyword is used to remove the data from the database tables.</td>
</tr>
<tr>
<td><strong>Insert</strong></td>
<td>Insert command is used to Add data to the database tables when the command is passed to the database with required values it adds the data to the database. Simply it is the command used to add new rows to the database.</td>
</tr>
<tr>
<td><strong>update</strong></td>
<td>Updates the existing data with the new data specified. It is normally used with the set keyword.</td>
</tr>
<tr>
<td><strong>Create</strong></td>
<td>Create is the keyword that is used to create new data elements in the database such as creating the database, Creating tables in the database.</td>
</tr>
<tr>
<td><strong>Drop</strong></td>
<td>Drop is the command that used to remove tables from the database. If the hacker can able to perform administrator actions on the database he can use this keyword to delete all the tables form the database.</td>
</tr>
<tr>
<td><strong>Alter</strong></td>
<td>Alter is the command that used for modifying the database table by adding more columns. By adding more columns to the database the hacker can inject malicious data permanently to the database.</td>
</tr>
<tr>
<td><strong>Where</strong></td>
<td>This is most commonly used keyword in SQL queries. The keyword is used to perform validation on particular fields. Upon validation if the command returns true it performs the requested operation against the database.</td>
</tr>
<tr>
<td><strong>Like</strong></td>
<td>Like is the keyword that can be used to perform the operations when the hacker doesn’t know the exact data to be queried with. Like can take arguments like ‘%’ to validate the data approximately.</td>
</tr>
<tr>
<td><strong>And</strong></td>
<td>And is the keyword that performs Boolean and operation on the database. If the first condition and the second condition validate as true then it return requested values from the database.</td>
</tr>
<tr>
<td><strong>Or</strong></td>
<td>Or is the keyword that performs Boolean Or operation on the database. If any of the condition validate as true then it responds with the requested information from the database.</td>
</tr>
<tr>
<td><strong>‘=’</strong></td>
<td>‘=’ is a keyword that normally used by the hackers when they are performing the operations with or keyword. Hackers use this keyword to perform the operation is always valid as true.</td>
</tr>
<tr>
<td><strong>From</strong></td>
<td>From is a common keyword that is used to specify on which table the operation has to be performed.</td>
</tr>
</tbody>
</table>
4.4 Programming

4.4.1 Getting Network device List:
The following code explains you how to get the network device list. This uses SharpPcap tool to capture all the network devices that are present in the system. On capturing the network devices it is required to change the network device number. On changing the network device number the program can scan all the packets that are flowing through that network device.

```csharp
using System;
using System.Collections.Generic;
using System.Text;
using Tamir.IPLib;
using Tamir.IPLib.Packets;
namespace sqlinject1
{
    class Program
    {
        static void Main(string[] args)
        {
            //To get all Networkable devices
            PcapDeviceList getNetConnections = SharpPcap.GetAllDevices();
            // network connection 1 (change as required)
            NetworkDevice netConn = (NetworkDevice)getNetConnections[0];
            PcapDevice device = netConn;
            // Define packet handler
            device.PcapOnPacketArrival +=
                new SharpPcap.PacketArrivalEvent(device_PcapOnPacketArrival);
            //Open the device for capturing
            //true -- means promiscuous mode
            //1000 -- means a read wait of 1000ms
            device.PcapOpen(true, 1000);
            Console.WriteLine("Network connection: {0}",
                device.PcapDescription);
            //Start the capturing process
            device.PcapStartCapture();
            Console.Write("Press any <RETURN> to exit");
            Console.Read();
            device.PcapStopCapture();
            device.PcapClose();
        }
    }
}
```

4.4.2 On Packet Arrival
When a packet is arrived on the network interface the following code invokes and it scans for the TCP Packets. The packet’s source address and the destination address are captured. If the packets are flowing through Port Number 80 which is used for HTTP traffic the program invokes method called GetURLString which is used for capturing the URL string from the packet.

The following code gets the URL string by GetURLString method then it uses some delimiter characters which are normally used in the URL strings to separate each word of the string. Upon seperating each word of the URL string the following code invokes the method called getscore. The getscore method returns the weighted value which is then added to the level.
Upon adding the weighted value to the level the value of the level increases. If the level value reaches more than the threshold level then the following code alerts the administrator that there is a possible SQL Injection attack going on the server, and it logs the packet into a text file which can be a future reference to the administrator to prevent attacks.

If the URL string level score doesn’t reach the threshold level then the packet is simply discarded.

```csharp
public static void device_PcapOnPacketArrival(Object sender, Packet packet)
{
    //Start Scanning if it is a TCP Packet
    if (packet is TCPPacket)
    {
        //capture packet details
        DateTime time = packet.PcapHeader.Date;
        int len = packet.PcapHeader.PacketLength;
        TCPPacket tcp = (TCPPacket)packet;
        string srcIp = tcp.SourceAddress;
        string dstIp = tcp.DestinationAddress;
        int srcPort = tcp.SourcePort;
        int dstPort = tcp.DestinationPort;
        string getstring;
        if (dstPort == 80)
        {
            //If the packet is flowing through port 80 then get the URL
            String getstring = GetURLString(tcp.Data);
            //Separate each word in the string wherever it find the following characters
            char[] delimiterChars = { ' ', ',', '/', ':', '	' };
            string text = getstring;
            string[] words = text.Split(delimiterChars);
            string[] found = { "select", "delete", "=" };
            //Initialising weighted score
            //If the set of keywords is found increment the score
            int level = 10;
            level += getscore(text, "delete", "from", 30);
            level += getscore(text, "or", "=", 40);
            level += getscore(text, "select", "from", 20);
            level += getscore(text, "and", "=", 30);
            level += getscore(text, "drop", "table", 40);
            level += getscore(text, "insert", "table", 30);
            level += getscore(text, "select", "from", 30);
            level += getscore(text, "update", "set", 40);
            level += getscore(text, "create", "table", 30);
            level += getscore(text, "admin", "drop", 30);
            System.Console.WriteLine(level);
            //threshold level for the weighted score
            //if the weighted score is more than the threshold level display the message in console window and log the packet
            if (level > 30)
            {
                Console.WriteLine("Sql Injection Attack Detected");
                System.IO.StreamWriter sw1 = System.IO.File.AppendText("logfile.txt");
                sw1.WriteLine(text + " Detected at: " + System.DateTime.Now);
                sw1.WriteLine(text);
                sw1.Close();
            }
            else
            {
                //If the weighted score is less than the threshold level.
                Console.WriteLine("No Sql Injection detected");
            }
        }
    }
}
```
4.4.3 Capture the URL String

The following programming code is used to capture the URL string from the network device. The URL strings normally have GET method and POST method. GET is used when the computer is requesting some information from another server. POST is used when the computer is responding to a remote machine with requested information.

The user is required to change the Method to GET or POST according to the Program Usage. If the program is Used for scanning for the outgoing packets than the GET keyword can be used. If the program is used in the server side then it is required to change the value to post.

```
#Code Snippet
//Method to capture URL String from the TCP Packet
public static string GetURLString(byte[] data)
{
    string s = System.Text.ASCIIEncoding.ASCII.GetString(data);
    if (s == "") return ("");
    string[] strings = s.Split('r');
    for (int i = 0; i < strings.Length; i++)
    {
        //Get is for all the strings that are requested by the network device. //Post is to scan all the HTTP traffic that is incoming to network device.
        if (strings[i].StartsWith("GET"))
        {
            string[] final = strings[i].Split(' ');
            return (final[1]);
        }
    }
    return ("");
}
```

4.4.4 Comparing The URL string words

The following code is used for comparing the words that are separated from the URL string. The Indexof method in C# is used to find the occurrence of the keyword in the string. If the keyword that is specified is found in the URL string then it returns the score to level which is then added to weighted score.

```
#Code Snippet
//Methods to compare the separated HTTP String Words
public static int getscore(string word, string tofind, int score)
{
    if (word.IndexOf(tofind) > 0) return (score);
    return (0);
}

public static int getscore(string word, string tofind1, string tofind2, int score)
{
    if ((word.IndexOf(tofind1) > 0) && word.IndexOf(tofind2) > 0)
    return (score);
    return (0);
}
```

4.4.5 The application location

It is important to configure the location of application as it determines which part of
the URL string it needs to capture to detect Injection attacks. As discussed in chapter 2 it is known that the packet which is travelling from the internet comes through the system after scanned by the firewall. The firewall is unable to stop the SQL Injection attack as the string is passed to the server as a normal request that is generated by the web browser.

The web server is designed to evaluate whether the request is intended to its own server or a different one. If the request is designated to the same server then the server address is validated in the web server process and it then passes the remaining string to the application server for processing the request.

The application server is designed to connect the database server to perform the requested operations on the database server. Upon getting the result form the database server it then responds to the application server with requested information.

Figure 4.1 represents the location of the application in server for successful monitoring of Injection attacks.

**4.5 Conclusions**

This chapter proposes an intrusion detection system to identify the SQL injection attacks. The programming is done by using Microsoft Visual Studio and using C# programming language. The chapter also explains about the methodology that has to be used on each part of the TCP/IP packet processing. It starts from evaluating the packet from the stage of entering into the network till the scanning of the application for any maliciousness. The programming part is included according to the process it is written for and the program is explained wherever it is necessary.
5 Evaluation

5.1 Introduction

This chapter discusses about the evaluation of the above discussed program to effectively scan the network packets for SQL Injection attacks. To evaluate the application different SQL Injection strings are added to the URL strings file and the application is processed as a standalone application on the text file to evaluate for accuracy.

Several tests are conducted on different types of strings and the results are saved into an excel file to observe the level where the injection attack is detected properly. In the application evaluation phase the CPU and RAM usage tests also performed to evaluate the load that is generation on the CPU. The application is designed to run continuously to monitor the network traffic, so it is very important to consider the load that is generating on the system performance.

The application is evaluated with a text file which contains captured URL strings from the network traffic. The network traffic is captured into a text file and it is injected with some effected SQL Injection strings. Once the evaluation is carried out by the application on these strings the results are stores into a excel file with their values. For successful evaluation the application is actually set the threshold level to 10 so all the strings are reported as defective.

5.2 Application Evaluation:

5.2.1 Example1

For the evaluation test 1 the text file is captured with 789 URL strings and out of that 10 strings are actually infected with Injection attack. These strings are captured in normal condition where there is no SQL Injection attack is process. The result file sorted depending on the level of the threat.

When the application is performed on the strings it resulted with 789 strings where as the application consists 734 strings at level 10, which are non-infected strings. The remaining 55 strings are reported as defected and their reported level is above 40. The application also resulted 15 strings with level above 50. The application reported 13 strings above level 70. And it also reported 2 strings with level as 100.

So by considering there are actually 10 strings are affected, the application is resulted with 55 strings at level 40. By verifying the result file it is found that all the injected strings are reported with at least a level of 40. In level 50 it detected two strings which are affected with Injection attack. The level 70 contains 13 URL strings out of which only 3 are affected. The level 70 showed 10 strings with are not actually infected. The level 100 shows 2 URL strings which are actually infected with Injection attack. For ease of analysis a graphical representation of the results are shown below.
False Positives- The false positive condition can be defined as when the application detects maliciousness when there is no maliciousness found in the actual result.

The False positive rate for the application can be calculated by using the formula, 
\[
\text{(reported non malicious strings/total resulted strings)} \times 100
\]

Reported non malicious strings=44  
Total reported strings=54  
According the formula = \((44/54) \times 100= 81.48\%\)

True Positives- True positive can be defined as the when the application is resulted with the actual strings that are malicious. The true positives can be calculated by \((\text{Actual malicious strings/Total reported strings}) \times 100\)

True positives is \(= (10/54) \times 100= 18.51\%\)

By the experiment above it is found that all the affected strings are shown as infected on the level above 40. But the level 40 is having more number of strings that are not actually infected. The level 50 is having 2 infected strings and the level is not having any strings that are not infected. Level 70 again consists 11 strings out of that 8 are
not infected and 3 strings are actually infected. Level 100 is can be a definite injection attack because this level can be formed only when matches more number of injection attack keywords.

5.2.2 Example 2

For evaluation in this example 968 URL strings are captured into a text file and out of the 968 string 10 strings are injected with affected SQL code. The application is performed against the URL strings and the results are stored into the excel file.

The application was able to find all the affected strings with a value greater than 40 and the results are varied between value 40 and value 100. The application resulted with 40 strings which are having value 40 or more than that. One URL string with value 50, 6 strings with a value of 70. The application also resulted with 3 strings with a value of 100.

Graphical Representation of Evaluation2:

![Graphical Representation of Evaluation2](image)

Graph 5.2. Graphical representation of evaluation 2 Values.

Analysis Values:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Total Results</td>
<td>30</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Non Malicious</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Malicious</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 5.2. Evaluation2 Analysis values.

False Positives are \(= (30/40) \times 100 = 75\%\).
True Positives are \( \frac{10}{40} \times 100 = 25\% \).

As there is no injection attacks found on any of the above examples the level up to 30 is ignored. The injection attacks with a value 40 are normally having 2 keywords. The value 70 normally represents there are two set of keywords. The injection attack with a value of 100 means it is having 3 set of keywords.

But when the results are evaluated the strings are normally having only one set of keywords on each of the detected values. But the application is returning some strings, which are bigger in length as an injection attack affected strings. Though this is not a result the application designed to perform the result can be helpful to find the buffer overflow attacks which are happened through the URL strings.

An administrator can effectively evaluate all the strings that are captured by the application and he can then separate which are actual injection attack strings. By taking preventive measures in the programming the administrator can be able to stop the future SQL Injection attacks.

As the application is resulting strings which are greater in length and actual injection attacks at the level more than 40. The threshold level of the application can be considered as 40. If the level is exactly 40 then the string can be considered as a threat. If the level value is greater than 40 then the string can be considered as actual SQL Injection threat and the administrator can perform preventive actions based on the evaluation.

### 5.3 Alerts

The application is designed to alert the administrator when there is an actual SQL injection attack is going on the server. The threshold value for the application is considered as 40. When the URL string crosses the threshold value 40 the application displays a message box indication that “SQL Injection detected”. This helps the administrator to take preventive measures when the attack is in progress. The sample alert screen is shown in figure 5.1.

![Alert Screen](image)

Figure 5.3. Alert screen

### 5.4 Conclusions

This chapter discusses about the results that are generated by the application. The results are evaluated considering the maliciousness in the URL strings. The chapter aims to find the threshold level value to detect the maliciousness. The analysis is presented in graphical format.

The results are evaluated at each level and the levels between 10 and 40 the values are
not given any result of maliciousness strings in that level. From the level 40 the output of the application contains many strings that are not maliciousness. But the application resulted with the strings that are higher in level. So these strings normally may contain Buffer Overflow attacks. The application considers the maliciousness level as 40, and analyzing the actual results is suggested.

The program is designed to alert the administrator when there is an actual SQL Injection attack is in process with the server. The administrator can verify the log file when he identifies the attack, then he can take further action to stop the attacker to penetrate into the database.
6 Conclusion

This thesis aimed at capturing the network TCP/IP packets that are flowing in the network and scan them to find the maliciousness. These packets are scanned for an SQL Injection type of maliciousness. This is carried out by implementing a network packet sniffer and analyzing the packets that are flowing through the network ports. Scanning for the maliciousness is achieved by splitting-up the URL string into keywords and comparing the keywords against the set of keywords and characters that are used in SQL injection attacks.

The SharpPcap application is used to scan the network traffic, and the project is implemented in Visual C# using the Microsoft .NET Framework. Different tests were carried out to find the threshold value of maliciousness. This chapter provides a critical analysis of the whole project and suggestions for future work.

6.1 Critical Analysis

This thesis conducted tests on TCP/IP packets to find maliciousness in the network traffic. Though this is an effective method to detect SQL injection attacks, the method takes more response time to perform an operation and it uses more system resources, which are valuable for server operations.

There are number of aspects that could be implemented in more efficient manner, and it aims to report the maliciousness over a given threshold level. It is found that the actual numbers of non-malicious results are higher than the number of malicious attacks. The threshold level can be adjusted to a particular level to effectively report the Injection attacks without reporting non-malicious URL strings. The false alerts were typically caused by long URL strings, and with some strings with an ‘=’ character. These caused a relatively high score, which identified maliciousness, where there was none.

The program can thus find the maliciousness that is injected in HTTP packets and store them in another log file to analyze the results by an administrator (non real-time). Though it is clear that the higher the value; the more serious the threat can be. It is not a definite solution, as a lower value result may also miss possible threats. An example of this is:

```
/it/xslt/webtrends/sdctag1.js/ where field = &anything& or &x=&x&
```

Which gives a low score (40), but has a serious threat in it. The threshold of 40 obviously needs to be further investigated, but a lower value might not pick up all the threats, whereas too high a value will have too many false alerts.

The thesis thus aims to help the administrator to verify the type of attack that could have happened on the database by alert message. But if the administrator does not take any preventive measures, this could cause a threat on the database. This could be as severe that an attacker could change the whole database structure before the administrator finds out that an attack is going on.

6.2 Future Work

This thesis is developed presents ideas for future work related to Injection attacks. Considering the hackers are trying to find new types of attacks day-by-day it is...
important for this application to have future improvements to effectively scan the new type of attacks. It is well known that the SQL keywords can be used in different combinations to attack the databases (Steve Friedl., 2007). The developed program leaves the future scope to add more keywords to the application to effectively scan the new type of attacks.

The URL strings that are logged into the log file can be evaluated by an administrator to take preventive measures to stop the Injection attacks. But by blocking the hacker IP address as soon as the injection attack found in the network this application can offer more security to stop the SQL injection attacks, this could be an effective to stop the Injection attacks, but by implementing this in the application the application can also block legitimate user’s IP address. It is important to consider that a genuine user’s IP address could not be blocked accidently. This could be done on host-based firewall on the server, or could be by reconfiguration of a network firewall, using Cisco ACLs.

The application can be implemented in an internet server to evaluate the actual effectiveness of the application. This evaluation could produce a good analysis on real-time SQL injection attacks. The threshold value level used to detect an injection attack can be changed according to the actual traffic that is observed in the internet server. The application can also be designed as an add-on for browsers, or a windows application, to continually scan the network usage and applications. This application can be developed in such a way that it uses fewer system resources to protect the system from crashes due to low system resources.

6.3 Final Words

This thesis was aimed to scan the network traffic that is infected and to report the administrator that there is an attack operating against the database. The experiments are developed in console based application, with an option to increase or decrease the threshold value, and the actual keywords that has to be scanned.

Although the entire application is made to provide security against the databases, if the programmer can make an effective application in the development stage, without security threats and software flaws, the solution could be much easier. The core problem that can give a chance to hacker to penetrate into the database is badly designed and implemented software code.
7 References


Zone-H Microsoft Defaced, Again Retrieved on 03 August 2008 from, http://www.zone-h.org/content/view/14780/31/
8 Appendix

8.1 Program for Capturing SQL Injection attacks

This shows code snippets of the overall code used.

```csharp
#Code Snippet
using System;
using System.Collections.Generic;
using System.Text;
using System.Globalization;
using System.Threading;
using Tamir.IPLib;
using Tamir.IPLib.Packets;
namespace sqlinject1
{
    class Program
    {
        static void Main(string[] args)
        {
            //To get all Networkable devices
            PcapDeviceList getNetConnections = SharpPcap.GetAllDevices();
            // network connection 1 (change as required)
            NetworkDevice netConn = (NetworkDevice)getNetConnections[0];
            PcapDevice device = netConn;
            // Define packet handler
            device.PcapOnPacketArrival +=
            new SharpPcap.PacketArrivalEvent(device_PcapOnPacketArrival);
            //Open the device for capturing
            //true -- means promiscuous mode
            //1000 -- means a read wait of 1000ms
            device.PcapOpen(true, 1000);
            Console.WriteLine("Network connection: {0}", device.PcapDescription);
            //Start the capturing process
            device.PcapStartCapture();
            Console.Write("Press any <RETURN> to exit");
            Console.Read();
            device.PcapStopCapture();
        }
        public static void device_PcapOnPacketArrival(object sender, Packet packet)
        {
            //Start Scanning if it is a TCP Packet
            if (packet is TCPPacket)
            {
                //capture packet details
                DateTime time = packet.PcapHeader.Date;
                int len = packet.PcapHeader.PacketLength;
                TCPPacket tcp = (TCPPacket)packet;
                string srcIp = tcp.SourceAddress;
                string dstIp = tcp.DestinationAddress;
                string srcPort = tcp.SourcePort;
                string dstPort = tcp.DestinationPort;
                string getstring;
                if (dstPort == 80)
                {
                    //If the packet is flowing through port 80 then get the URL String
                    getstring = GetURLString(tcp.Data);
                    string getstring1 = getstring.ToLower();
                    string getstring2 = getstring1.Replace("'", "&");
                    char[] delimiterChars = {',', ',', '&', '/', ':', '\t'};
                    string text = getstring2;
                    string[] words = text.Split(delimiterChars);
                    //Seperate each word in the string Whereever it find the following characters
                    int level = 10;
                }
            }
        }
    }
}
```
level += getscore(text, "delete", "from", 30);
level += getscore(text, "or", ",", 40);
level += getscore(text, "select", "from", 20);
level += getscore(text, "and", ",", 30);
level += getscore(text, "insert", "table", 40);
level += getscore(text, "union", "select", 30);
level += getscore(text, "update", ",set", 40);
level += getscore(text, "create", "table", 30);
level += getscore(text, "admin", "drop", 30);
System.Console.WriteLine(level);

//threshold level for the weighted score
//if the weighted score is more than the threshold level display the message in
console window and log the packet
if (level > 40)
{
    Console.WriteLine("Sql Injection Attack Detected");
    System.Console.WriteLine("Original text: '{0}'", text);
    System.IO.StreamWriter sw1 = System.IO.File.AppendText("logfile1.txt");
    sw1.WriteLine(text + " Detected at: " + System.DateTime.Now);
    sw1.WriteLine(text);
    sw1.Close();
} else
{
    //If the weighted score is less than the threshold level.
    Console.WriteLine("No Sql Injection detected");
}

//Method to capture URL String from the TCP Packet
public static string GetURLString(byte[] data)
{
    string s = System.Text.ASCIIEncoding.ASCII.GetString(data);
    if (s == ".") return ("");
    string[] strings = s.Split(\r');
    for (int i = 0; i < strings.Length; i++)
    {
        //Get is for all the strings that are requested by the network device.
        //Post is to scan all the HTTP traffic that is incoming to network device.
        if (strings[i].StartsWith("GET"))
        {
            string[] final = strings[i].Split( ' ');
            return (final[1]);
        }
    }
    return ("");  
}

//Methods to compare the Seperated HTTP String Words
public static int getscore(string word, string tofind, int score)
{
    if (word.IndexOf(tofind) > 0) return (score);
    return (0);
}
public static int getscore(string word, string tofind1, string tofind2, int score)
{
    if ((word.IndexOf(tofind1) > 0) && word.IndexOf(tofind2) > 0) return (score);
    return (0);
8.2 Output Screens

Figure 8.1. Output when No Injection attack is detected.

Figure 8.2 Output of application when attack detected.
Figure 8.3. Sample Log file Output.