2006

SQL Injection - Threats to Medical Systems: The Issues and Countermeasures

Craig Valli
Edith Cowan University

SQL Injection - Threats to Medical Systems: The Issues and Countermeasures

Craig Valli
School of Computer and Information Science
Edith Cowan University
Mount Lawley WA, Australia

Abstract - *A vast majority of medical information systems use Standard Query Language databases (SQL) as the underlying technology to deliver medical records in a timely and efficient manner. SQL is a standardised and well entrenched database technology, which allows for the development of robust, customised applications for information management. In recent years, SQL has been used as the back-end to many successful web client accessible applications. The use of SQL in this manner has been greatly enhanced through the development of server side scripting languages such as Microsoft ASP and open source systems such as PHP. These allow for the representation and extraction of data from a database and have a range of manipulation and display possibilities allowing a developer a rich tapestry of options. However, these scripting languages have enabled the ability for malicious users to directly modify, manipulate or destroy SQL databases. In addition to those server side scripting language problems there is also malicious software in the form of worms specifically targeting SQL databases.*

Keywords: SQL, injection, medical, countermeasure

1 Introduction

Increasingly medical practitioners are either being encouraged or forced to use information technology to make the businesses more efficient or accountable. This impetus for adoption of information technology into modern medical practices comes from several sectors. In the case of Australia, the Australian Federal government has either mandated or provided significant financial incentive for medical practitioners to modernise their practices by installing Information Systems. Similar schemes are also to be found other nation states around the world. As well as government forces requiring a move into information technology many of the medical supply companies are likewise following suit. Whether it be physical goods such as pharmaceutical or the provision of knowledge such as test results for pathology or x-ray of these suppliers increasingly are requiring ordering and delivery of goods and services via electronic systems that utilise the Internet as a principal conduit.

Consequently, as a result of this trend, medical practices are now availing themselves of broadband technologies and SQL enabled medical practice/client management databases to meet this demand and also provide them with strategic advantage through the usage of IT. There is little argument that a modern database (typically SQL) centric information system will garner significant cost savings and strategic advantage for any information dependent organization. This cost saving is principally in timely delivery of information for legitimate purposes or use by the organisation. In addition to increased speed of retrieval and the ability to perform meaningful analysis and searches on data, cost savings can be found in significantly reduced storage space required to house records. A standard 2m tall rack with a foot print of $0.25m^2$ or $0.5m^3$ can hold a server and several terabytes of storage space. This could replace several large rooms of paper based storage, realising significant savings in reduced rental costs for floor space and supporting infrastructure. Reduced labour expense in the physical retrieval and re-storage of the record is also found in itself a further significant cost saving to a business.

What makes this current push more volatile is that the systems that are being delivered by vendors are becoming more multi-functional and at the same time monolithic. A SQL enabled DBMS allows application developers to produce a single data entry and aggregation point for all medical practice business billing, notes, pharmacy, pathology, tests etc a sound business concept in theory. This philosophy is very problematic for securing all information about a patient i.e one breach on one level and an attacker potentially has the complete picture.

These market trends and forces are now bringing a significant threat profile in the form of SQL injection and manipulation within reach of a medical practices client records. No longer does an attacker need to be physically present to see the medical record they simply purloin or modify the information across the ether connecting the medical practice to a more purportedly, productive world.

2 What is SQL injection?

SQL injection is normally made possible as a result of programming flaws in an application that allow for changing the query parameters. This is a result of poor validation or checking of the input. It is a common misconception that the, well known, SQL slammer worm

was an SQL injection attack, but it was actually a result of a serious flaw/exploit in the underlying server infrastructure.

SQL injection attacks work because the application layer is not properly checked for its input[1]. In fact it is possible to perform a SQL injection attack on a properly secure and patched SQL server as it is simply responding to the SQL query it receives. The queries, which in these cases has been intercepted or modified by attacker interactions, remain legitimate or accurate SQL syntax after modification by the attacker.

To illustrate this, take a SQL procedure that takes a username and password as input on a web based login form box that could look like Figure 1.

Figure 1: Login prompt

"select * from database.user

where username=’ " . $userName . " ’ and

password=password(’ ". $passWord . " ’);"

Instead of a valid user-name, the malicious user sets the $userName variable to the string: ’ or 1=1; - -’, causing the CGI script to issue the following SQL query to the database:

"select * from database.user

where username=’’ or 1=1;-- ’ and

password=password(’ ". $passWord . " ’);"

The problem here is that - - is the SQL comment command i.e all text after this is ignored so now the query really reads like

select * from database.user where username=’’ or 1=1;

This means that the database will now return a list of usernames to the screen of the attacker and they can go about attempt a break in by password guessing. This is a simple select statement in SQL which is principal used to query or extract information from a database. A more sinister form of attack is when the attacker actually uses the INSERT or UPDATE statements in SQL to create or modify an existing record using similar mechanics of modification. There are various other methods of attack using SQL injection techniques that are already well documented. It is not necessary to include each variant in this paper as its progress as an exploit is well beyond proof of concept [1-5]. As one software developer put it, "From the outside, all SQL injections look the same. But there are five or six ways to set up a call that has SQL injections."[2] The point of the previous explanatory text was to illustrate the ease of attacks.

The major problem with this form of attack is that if there is no integrity checking run on the database inputs, modification can be undertaken at will by an attacker without fear of detection. Many intrusion detection systems are still not fully checking for this type of malicious input into a system hence the data streams will be allowed through. One of the other issues is that supporting server side scripting architectures such as PHP and Microsoft ASP have some limited in-built methods for checking for such inputs with most requiring action from the application developer. PHP has for instance magic_quotes which checks for and can help against such incursions but it must be enabled it is not set on by default.

3 How feasible are attacks on medical records?

The use of SQL injection attack, and the subsequent consequences to a medical information system, can only be described as potentially catastrophic and life threatening. The ability to potentially change any data contained within a database of this sort is simply not an acceptable option. However, potentially many custom built SQL applications are susceptible to these attacks and that includes medical systems.

The enabler of this type of attack is simply a broadband connection that allows an attacker to probe your server. Now many people reading this would postulate that it is impossible to target medical systems specifically, think again. Using Google or similar search engines an attacker can target systems with a high probability that they are medical systems. Some simple targeting examples follow in Table 1. The examples given here allow for broad reconnaissance of the Internet for host to potentially compromise. A more determined attacker can actually use Google or other search engines to interrogate a domain name or website of a medical practice. This is standard practice now for many penetration testing schemes and there are several published books to aid you in using search engines for this purpose[6]. Further specificity can be arrived at by using domain name interrogation and other IP probing tools such Nmap[7].

Table 1 Sample Google Queries

Google Query	Hits
intitle:medical inurl:login	20000 hits
inurl:surgery inurl:login	899 hits
inurl:doctor inurl:login	695 hits
intitle:hospital inurl:login	696 hits
intitle:patient inurl:login	682 hits

The reason that this is a risk is that for many small businesses is the actual web server that serves web content itself may be contained on the practice SQL server. In fact many vendor products designed for small businesses actually encourage this sort of deployment e.g Microsoft Small Business Server suite by providing a monolithic solution to the provision of business applications and services.

4 Mitigating against SQL injection

The following are some simple steps that medical practises can take to secure there client or practise management systems from SQL injection. These are by no means an exhaustive list of options but are the relatively easy to effect and can great reduce or even nullify the threat of SQL injection from external attackers.

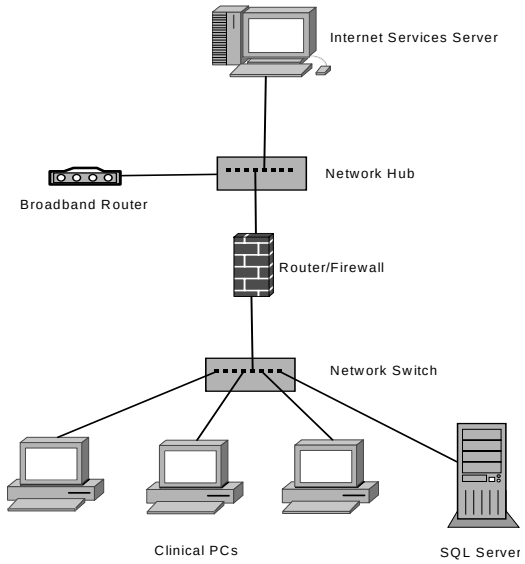
4.1 Simple Infrastructure Changes

As mentioned before, vendors will often encourage using a monolithic architecture i.e. all components on one server by having a one size fits all small business server "solution". This server normally serves the SQL database, web services, web proxy and is used to connect directly to the Internet. The connection to the Internet in most of these cases is via an ADSL router or similar broadband based router directly connected to the server. At best a simple dual bastion design is utilised in these situations to protect the network. This network design is not the strongest and is becoming increasingly ineffective against newer attacks.

Cost is a factor in any business and more so often in smaller business were economies of scale and profitability may not be as high. But regardless of cost a risk that is potential business terminator should not be too heavily discounted in considering risk for a medical practise. Significant security advantage can be gained by separating the practice/client SQL server from the Internet and web access server and using a DMZ mentality see Figure 2.

The purchase of a new Internet and web server to physically separate servers will easily be defrayed against the significant risk this purchase reduces. It will help also mitigate risk of other Internet borne attacks such as viruses and spyware which is significant and warrants factoring into this decision.

Figure 2 – DMZ Deployment SQL Server



The use of a de-militarized zone (DMZ) with firewalling and routing will afford a higher level of protection for the SQL server. The SQL server would no longer be accessible directly from the Internet sitting behind the DMZ in the private network with the practice computers. This type of defensive arrangement would hamper significantly the possible reconnaissance from search engines and other simple attack methods that could be employed by hackers on a monolithic system.

One of the common misconceptions in network security is that the use of network switch will prevent/stop a hacker from penetrating a network further, this is simply not true. The use of ARP poisoning software such as ettercap [8] which is readily available from the Internet will allow the attacker to penetrate a network switch.

Effective firewalling technology is often seen as expensive and in the domicile of scientists in white coats this is no longer the case. Many commodity based ADSL routers for instance have been certified by the same testing schemas that are applied to large enterprise level firewalls and have proven equally fit for purpose[9]. Furthermore, modern operating systems now have as there base operating system a competent firewalling system that can further strengthen defences.

4.2 Input or Data Sanitisation

Removal of any unwanted characters say ' , " , ; , or - - from the received from the input stream. Allowing these

types of characters that are correct SQL syntax may allow SQL injection attacks to be performed on your SQL based application.

However, one particular problem from an SQL injection viewpoint is that you will need to allow certain special characters such as ' or - so a patients name such as O'Bree or Della-Agostini could be used in the application. Now while the use of such characters is good for correct name syntax it is bad in mitigating SQL injection via syntactic insertion. The alternative is to replace the characters with a close substitute such as ' with ‘ or a character as that would not represent or be legitimate SQL syntax. By doing this substitution it would be useful for the mitigating against propagation of SQL injection attempts. The converse however would have to happen for display of database records in reports such as queries or output to other systems otherwise the names would be grammatically incorrect.

It is interesting to note that such routines for cleansing or substitution can have significant reuse in all aspects of the application and often only need to be created once and reused via calls to the validation of substitution procedure.

4.3 Strict SQL database permissions

Allowing unrestricted access to the SQL application although it might be expedient for development of systems it is not ideal for their eventual deployment and will cause security issues for the owner of the software. A proper mapping of appropriate DBMS read or write permissions for each role within the system can mitigate against SQL injection. This is typically done at the developer level which is achieved by giving only necessary permissions to the user or procedure to perform that particular job function or process. In databases this is accomplished typically via the use of views. A view is used typically for displaying queries so unless a user needs to directly update a record they should see only a read-only version of the record (view).

4.4 Reducing Customisation

By using a stored and tested procedure instead of allowing dynamic queries in your SQL based applications. This will reduce the chance of such attacks and if properly tested and constructed should also increase server efficiency and performance gains are also garnered by this approach by restricting inappropriate resource usage. As an example a user can not accidentally query or modify the entire database schema by the input of an inappropriate query parameter. The same logic is then applied to malicious codes if no query interface is given then the presentation of malformed SQL code becomes a difficult outcome to achieve.

4.5 Simple System Auditing

The use of some of the in-built auditing tools in most SQL server implementations could provide at least alerting that some malicious activity is being undertaken. Furthermore, conventional performance measuring tools provide with server operating systems might indicate an unnecessary large server load as a result of intense SQL queries being run in non-business hours. Incidents of these types are relatively easy to detect as a system over time will tend towards a normalised pattern of loads and behaviours from its users.

5 Conclusion

SQL injection may seem a reasonably esoteric subject for medical practices to consider in a threat profile. However, as medical practices become more dependent on the use of information and network technology they will need to avail themselves of security posture and countermeasure beyond simple virus protection and simple firewalling. Information contained on medical systems is systems of interest to more than the stereotypical hacker who lays waste to data with impunity or wants to “ownz” the system. The target group for such systems data includes a wide range of attack profiles and motivations. A dialogue between stakeholders in this arena should be on-going and is vital if we are to protect data from deletion or even worse modification.

Further practical research is about to be started from an Australian context where the leading products in this area of practice/client management which utilize SQL will be fully penetration tested for SQL injection and range of other ancillary security threats. It is hoped that all of the software tested goes through unscathed and is robust. Finally, as people can literally live or die as a result of the data held in these systems it does bring new meaning to the words data integrity or would it be better couched as patient integrity.

6 References

[1] S. King, "How to plug online gaps," in *Computer Weekly*, 2004, pp. 30.

[2] J. deJong, "Top Ten, Other Lists Catalog Security Threats," in *Software Development Times*, 2005, pp. 5.

[3] L. Gomes, "PORTALS: More Scary Tales Involving Big Holes In Web-Site Security," in *Wall Street Journal*, 2004, pp. B.1.

[4] R. Kwon, "Is Your Web Site at Risk of Injection?," *Baseline*, vol. 2, pp. 108, 2002.
[5] M. Thurman, "Security problems put survey app on sidelines," in *Computerworld*, vol. 37, 2003, pp. 30.
[6] J. Long and E. Skoudis, *Google Hacking for Penetration Testers*: Syngress, 2001.
[7] Fyodor, "Remote OS detection via TCP/IP stack fingerprinting," vol. 2002, 1998.
[8] A. Ornaghi and Marco, "ettercap," 2005.
[9] Netgear, "ProSafe™ VPN Firewall w/4 Port 10/100 Switch and Print Server Model FR114P," NetGear, 2005.