Investigating Database Security in a Networked Environment

Matthew Giuliani
Computer Science Department
University of Wisconsin – Eau Claire
Eau Claire, WI 54701
giuliaml@uwec.edu

Eric Lobner
Computer Science Department
University of Wisconsin – Eau Claire
Eau Claire, WI 54701
lobnerec@uwec.edu

Paul J. Wagner, Ph.D.
Computer Science Department
University of Wisconsin – Eau Claire
Eau Claire, WI 54701
wagnerpj@uwec.edu

Abstract

Database and network security have traditionally been separate fields within the realm of computer security. Currently, there is very little work done to understand the security of data flowing between client and database systems. Although vender specific information is available for many database systems, there are very few comparative studies that analyze multiple database systems. In this study, five database systems were analyzed and subjected to tests from three separate platforms. The testing platforms included two software applications and one vendor supplied administration/query tool. The security of each system was analyzed for five criteria that represent the relative security of data being transmitted to and from the database system. While passwords were always encrypted, other information such as password length, usernames, and query content were sometimes exposed. This research suggests the need for increased usage of encrypted protocols such as SSL in database applications.

Background and Current Problems

As the world around us becomes more dependent on computer information systems, the need has arisen for careful analysis of current security systems protecting sensitive data. Damage, misuse or capture of sensitive data may not only affect a single user, but may have widespread consequences affecting untold numbers of people.

Database systems emerged in the mid 1960's and have been continually developed and modified into the complex systems that exist in today's market. A database is a collection of facts (data) stored on a physical medium in such a way that a question, or query, can be asked of this data and an answer returned. A database management system (DBMS) is a set of programs used to connect to the database and run queries about the information stored in the database. Most often the actual database is stored on a separate computer called a database server which accepts queries and commands from remote clients.

Only recently, due to the widespread explosion of the internet and the need for storing more sensitive data, has database security become a primary concern for many database administrators. In December of 1985 the Federal Government issued a standard against which all computer security systems could be evaluated[1]. The Department of Defense Trusted Computer System Evaluation Criteria (TCSEC), also known as the Orange Book, was primarily developed "to provide users with a yardstick with which to assess the degree of trust that can be placed in computer systems for the secure processing of classified or other sensitive information." The U.S. Computer Security Center released the Trusted Database Interpretation in 1991 that used the TCSEC criteria to specifically evaluate database management systems. Although originally developed for government use, these criteria are often applied to commercial database systems resulting in a grade evaluating the confidentiality and security of information.[2]

Despite a stringent set of security criteria, attacks and vulnerabilities still occur even in the most advanced and up to date systems. In February of 2003 a breach in the security of a database owned by Data Processors International, a credit card processing company, released upwards of 8 million credit card numbers to the individuals responsible for the attack[3]. The release of the credit card numbers was estimated to cost the major credit card companies approximately $200 million dollars in cancellation and renewal expenses. Many other types of sensitive information are now being stored, including: medical records, insurance claims, social security numbers, and financial information, are all stored in complex database systems which could have disastrous effects if the security of these systems was compromised.

Investigation

Identifying the security level of a particular database management system can be a difficult task. First, there are multiple components to overall DBMS security, including how secure the information is sitting on a storage device, and how secure the information is when moved to and from the database over computer networks. Second, most past research and information has focused on the security of data on disk, leaving the security of data in a networked environment difficult to ascertain. Third, this information tends to be given for each DBMS separately, and has not been collected in a comparative form across multiple DBMSs that is easily available to those interested in determining database security. It is important to note that this study does not hope to explore the security risks of sending information over the internet, as most sensitive information is encrypted via HTTPS, but rather the network security of information once it has reached an organization's internal intranet.

The major objectives of this research are threefold. First, research the network security status of five major DBMS, both commercial (Oracle[4], Microsoft SQL Server 2005[5], and IBM DB2[6]) and open source (MySQL[7] and PostgreSQL[8]), by determining the security of networked query/administration clients and applications accessing those DBMSs. Second, develop testing applications that use existing network packet sniffing software to confirm prior claims or establish current status where that information is not given. Third, generate a comparative chart showing the status of secure communications for the above five DBMSs.

Hardware and Software Systems

In an effort to produce as accurate as results as possible, the network conditions, hardware systems and software used during tests were kept as similar and consistent as possible. All tests were completed within the University of Wisconsin – Eau Claire intranet. Figure 1 shows a diagram of the basic network and system setup utilized during testing.

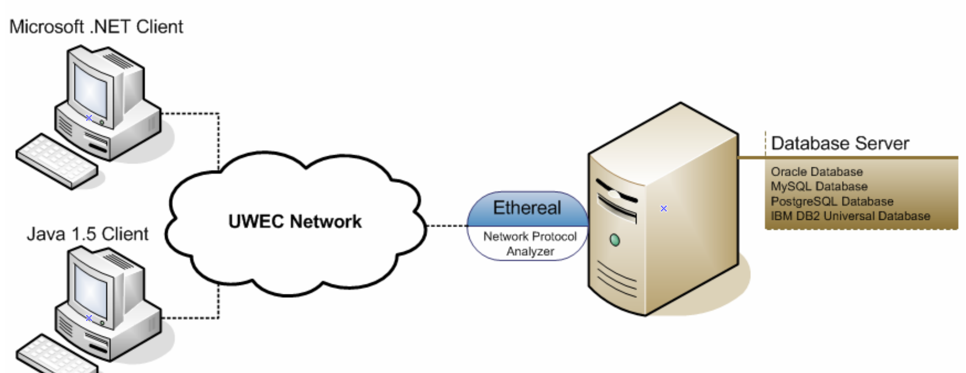

Figure 1. Typical Network and System Diagram

A total of three systems were utilized in the testing process of the five database products. Two systems were designated as client machines and loaded with the appropriate development environment for each of the programming languages tested. Both client machines were Windows XP systems having the most current updates at the time of testing. The Microsoft .NET client contained Microsoft Visual Studio[9] 2005 development environment while the Java[10] 1.5 client contained the latest version of

Eclipse development environment with the Java 1.5 SDK installed. The database server was a Linux system running the Red Hat[11] distribution under kernel 2.4.21-37. Ethereal[12] Network Protocol Analyzer version 0.10.13 was installed on the database server to capture network packets coming into and out of the database server. Remote desktop software was used on the client machines for purposes of installing and configuring the individual database systems.

Testing Methodology

The testing of a typical database system followed the same general steps. First, the most recent version of DBMS was downloaded from the vendor website onto the database server. The DBMS was then unpacked and installed with the most typical options and set to accept incoming connections (see sections on client authentication for each DBMS). The Oracle, MySQL, PostgreSQL and DB2 databases were installed on the Linux database server. Microsoft SQL Server required a Windows environment to operate and was installed and configured on the .NET client machine. The Java client machine was then configured with the Microsoft .NET development environment to provide testing for both programming language platforms.

Once each database was installed and configured, a test database was created for each DBMS using prepackaged tools included within the DBMS download. A database was created with a sample customers table which included the first name, last name and credit card numbers for five imaginary records. Each table was created using the same script, defining all columns the same width and data-type. After the database and associated table were created, two database connection applications were created on each of the two client machines. In addition, the standard query/administration tool for each DBMS, when available, was installed and configured on one of the client machines. The java application for each DBMS was created using the most recent JDBC (type 4) driver provided by the database vendor. The .NET application required similar steps in order to connect to the various DBMSs. Each database application performed three steps; connect to the database, run query, close database connection. All database connection applications ran a simple query to return all columns for all rows in the table. At the time of testing a known issue was present regarding the inability of Visual Studio 2005 to connect to versions of IBM DB2 Express-C. As such, the only testing platform utilized against the IBM DB2 – Express C database was the java 1.5 application platform.

Once each application was ensured to be working properly, security tests for each DBMS were conducted. Each test was started by connecting to the database server, via remote desktop software, and beginning a packet capture of all network packets flowing into or out of the database system. Each application was then executed and allowed to complete, after which the packet capture was stopped on the database server. In addition to the two applications developed, an additional test was performed by remotely connecting to each DBMS via the default administration/query browser provided by each vendor. A packet capture was started on the database server before the administration tool logged onto the DBMS and was stopped after results of the query were returned back to the client machine. Additional tests were performed on each DBMS if options relating to authentication methods could be changed reasonably (see PostgreSQL authentication section).

Once all tests were completed, the packet capture files were loaded back into Ethereal and analyzed for five conditions of security. The first three conditions related to whether the SQL query sent to the database was visible, whether the specific database instance was visible and whether the results of the SQL query being sent back to the client were visible. The MySQL database did not have what we would consider a traditional ‘database instance’ and was not evaluated as a condition of security for this database. The final two conditions prove to be most interesting and related to whether usernames and passwords required to access the database were visible across a network connection. Every packet capture file was analyzed for each of the five conditions and results of the analysis were organized into a table (see Table 1: Database Security Results) relating the security of each database.

Database Security in a Networked Environment

	Able to view SQL Query	*Able to view Database Instance*	*Able to view Query Results*	*Able to view Username*	*Able to view Password*
Oracle Database 10g					
Java 1.5 with JDBC	Yes	Yes	Yes	Yes	No
VS 2005 with .NET	Yes	Yes	Yes	Yes	No
SQL Plus Worksheet	Yes	No	Yes	No	No
MySQL Database Server					
Java 1.5 with JDBC	Yes	-	Yes	Yes	No
VS 2005 with .NET	Yes	-	Yes	Yes	No
MySQL Query Browser	Yes	-	No	Yes	No
PostgreSQL Core Database					
JDBC with 'trust' client authentication	Yes	Yes	Yes	Yes	No
JDBC with 'password' client authentication	Yes	Yes	Yes	Yes	**Yes**
VS 2005 .NET with 'trust' client authentication	Yes	Yes	Yes	Yes	No
VS 2005 .NET with 'password' client authentication	Yes	Yes	Yes	Yes	**Yes**
pgAdmin 1.4.1 with 'trust' auth.	Yes	Yes	Yes	Yes	No
pgAdmin 1.4.1 with 'password' auth.	Yes	Yes	Yes	Yes	**Yes**
IBM DB2 Universal Database Express-C					
Java 1.5 with JDBC [Type 4]	Yes	Yes	Yes	Yes	No
VS 2005 with .NET	IBM Known Issue				
Microsoft SQL Server 2005 Express					
Java 1.5 with JDBC [Type 4]	Yes	DB Name -Yes DB Inst. - Yes	Yes	Yes	**Length Given**
VS 2005 with .NET	Yes	DB Name - Yes DB Inst. - Yes	Yes	No	No
Microsoft SQL Server Management Studio Express	Yes	DB Name - Yes DB Inst. - Yes	Yes	No	No

Table 1: Database Security Results

Oracle Database 10g Authentication

Oracle supports several authentication methods; the method used in our testing was authentication by Oracle Database. With this method, the client provides a username and password which is compared against that stored in the Oracle database. Before any transmission of user credentials, the password is encrypted using a modified AES (Advanced Encryption Standard) algorithm. Other Oracle authentication methods include: Network, Multitier Authentication and Authorization, and Secure Socket Layer Protocol. These methods were beyond the scope of this project and were not taken into account in our testing.

MySQL Database Server Authentication

MySQL provides only one method of authentication. User passwords are not actually stored on the MySQL server, rather, the hash values computed from the password is stored. When a client attempts a connection, a hash value is computed from the provided password and is compared against the hash value stored in the user table on the MySQL server. MySQL 4+ uses a 41 byte hash, while previous versions use only a 16 byte hash.

PostgreSQL Core Database Authentication

PostgreSQL provides numerous methods for client authentication, which can be set to a client's preference on the PostgreSQL server. The first method is Trust authentication, where any user who can connect to the server using an existing username is assumed to be authorized for the database, including superusers. This method should only be used when every system allowed to connect to the database server can be trusted. Passwords sent using trust authentication are encrypted for network transmission. The second method for client authentication is Password authentication, which offers three types of password encryption: MD5-hashed, crypt-encrypted, and clear-text. For connections over the internet, MD5-hashed is the preferred method to subdue the capture and deciphering of the database password. Likewise, the crypt-encrypted option will encrypt the password, but will not work on passwords that have been encrypted in pg_authid, and should only be used for backward compatibility. The clear-text option will not encrypt the password at all, and will be sent via clear-text. The third method is Kerberos authentication, a leading standard in secure authentication. PostgreSQL supports Kerberos version 5, and this functionality must be enabled when installing the database server. This method was beyond the scope of this project and not taken into account in our testing.

IBM DB2 Universal Database Express-C Authentication

IBM DB2 uses Server authentication as the default method for client connections. The Server method is invoked even if no method is explicitly specified. With Server authentication, a username and password must be supplied by the client for connection with the DB2 server. For clients accessing a local DB2 server, operating system authentication is sufficient; however, for remote connections a username and password must be explicitly stated even when the client system is located on the same domain as the DB2 server. Using the Server_Encrypt option, the password will be encrypted before transmission over the open network. DB2 employs several more authentication methods, such as: Client, DCS, DCE and Kerberos. These methods have not been taken into account in our testing.

Microsoft SQL Server 2005 Express Authentication

Microsoft SQL Server provides two different authentication methods: Windows Authentication, and SQL Server authentication. Windows authentication calls back to a client's Windows NT or Windows 2000 user account to obtain a validated username and password to authenticate. If a client attempts a connection while providing a blank username, Windows authentication will be used. Likewise, if a user sends an explicit username and password, they will be ignored and Windows authentication will still be used. SQL Server authentication is used with clients who are connecting over a non-trusted connection. The user provides a username and password, which are compared against existing user accounts on the SQL Server.

Overall Comparison

Overall, a majority of the database products tested in this study exhibited similar characteristics of security. The following is a breakdown of similarities and differences exhibited by each test platform on the five database systems tested.

The specific SQL query text executed on all databases tested was visible using techniques discussed in this study. The database instance name for each database was also readily visible using techniques discussed in this study. Although the query itself and database instance do not reveal any specific sensitive information, it does disclose some evidence to the structure of the tables within a given database.

All database products and test platforms except one, MySQL database via MySQL Query browser, were able to view the information returned from a given query on the database. Information that can be contained within the result of a query can be as simple as a first or last name, all the way to as sensitive as credit card numbers and other personal information. While encryption of data is another issue that we did not examine in this research, the possibility of returning unencrypted data from a DBMS is quite alarming and could lead to the compromise of extensive amounts of sensitive data.

The database username and password are sensitive pieces of information that if gotten into the wrong hands could lead to a complete loss of sensitive information stored in databases. Database usernames were visible for all typical application connections from both Java and .NET environments. Usernames for the open source database query browsers demonstrated similar characteristics as the application platforms. The commercial products of Oracle and Microsoft took more care with their query/administration clients to encrypt the usernames being used to connect to a particular database. It was also noted that from the Microsoft .NET application environment to a Microsoft SQL Server database, encryption of the username occurred and was the only instance of a username being encrypted from an application platform.

The area of database password visibility demonstrated the most interesting results. Oracle database passwords for all three platforms were encrypted. MySQL database passwords were also encrypted for each of the three testing platforms. PostgreSQL demonstrated significant variance in database password visibility. The tests conducted using 'password' authentication demonstrated results in line with those discussed in the PostgreSQL authentication section as being passed to the database server as clear-text. When the PostgreSQL server was switched to 'trust' authentication, the passwords immediately became encrypted and undetectable. IBM DB2 showed results consistent with the majority of other tests demonstrating an encrypted password on the single platform that was ran against the database. The Microsoft database showed another significant security find of this project. Using a Java application platform with the standard JDBC driver, the length of the database password being used was readily available. While the actual password is not visible, any data that can be recovered about a specific password can help an attacker to discover the actual password.

Conclusion

This study set out to determine the network security of five of the leading database products on the market. Through the testing of these databases, two significant finds have been discovered. First, using database settings described in this study, the information being returned from a query is oftentimes sent unencrypted over a network transmission. Sensitive data such as usernames, passwords, credit card numbers and a host of other information are stored and retrieved at some point from a database. An attacker with access to network resources with the same approach and techniques performed in this study could easily compromise vast amounts of sensitive information. The second find this study discovered was information regarding password length being given in communication between and Java environment and Microsoft SQL Server 2005. Although the actual password was not shown to be visible, the length of the password is enough information to knock down exponentially the time needed to crack the password. It is often a small security leak that can provide the entrance to a whole slew of additional information desired by an attacker.

The results of this study are a first step in the analysis of information available regarding network and database security. Certainly more research needs to be performed testing for some of the other common conditions in database networks. Research exploring the authentication methods not tested in this study could prove to reveal other trends pertaining to database security in a networked environment.

References

[1] Department of Defense Trusted Computer System Evaluation Criteria.
http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html

[2] Computer Security Criteria: Security Evaluations and Assessment. Oracle Corp.
http://www.oracle.com/technology/deploy/security/seceval/pdf/seceval_wp.pdf

[3] Hacker Hits up to 8M Credit Cards. CNN Money Online.
http://money.cnn.com/2003/02/18/technology/creditcards/

[4] Oracle Corporation. http://www.oracle.com

[5] Microsoft Corporation. http://www.microsoft.com/sql/default.mspx

[6] IBM Corporation. http://www-306.ibm.com/software/data/db2/

[7] MySQL Database. http://www.mysql.com/

[8] PostgreSQL Database. http://www.postgresql.org/

[9] Microsoft Visual Studio .NET Framework. http://msdn.microsoft.com/vstudio/

[10] Java Technology. http://www.sun.com/java/

[11] Red Hat Linux Distribution. http://www.redhat.com/

[12] Ethereal Network Protocol Analyzer. http://www.ethereal.com/