CHAPTER 27

Database Security 2000

John R. Campbell
National Security Agency, Fort Meade, MD 20755-6730

Abstract: Database systems are being more and more used, with larger sized databases, and as components of very complex systems, that include numerous protocols. Database security problems in the past have only partially been solved. Is there any hope to provide adequate security for the new database systems? This paper describes user and system requirements and provides a list of hints and techniques to better secure this type of system.

1. INTRODUCTION

Have we solved the DBMS Security Problem? No, the DBMS security problem was never completely solved. Parts of the problem were solved, but, in many cases, were not implemented by the popular vendors. Inference, aggregation and individual I&A in complex systems were never completely solved. In addition, DBMS's are constantly evolving to meet the current expectations of users. Therefore the means of securing the data must change with the changes in architectures, methodologies and applications. Existing solutions are now even more insufficient. However, the overall goals of database information security, that of information privacy, integrity, availability and nonrepudiation remains the same. The data must be protected.

2. Environment:

The environment continues to become more interesting and more difficult to work in. Databases are more widely used than ever. More formats and protocols are used to access and link to data sources. Web

browser access is everywhere and interconnectivity is a keyword. But with such configurations, can we assure that only authorized persons are accessing the data? Do we have usable audit for such transactions? The configuration today could be cell phone request to data source. The cell phone or laptop could be located anywhere, possibly in an analog-only cell area, as could the data source. At a recent conference in Maryland, an attendee was accessing a California database during a break in the conference via a cell phone/laptop client. No one thought this act was unusual.

To make matters worse, there are now fewer trusted operating systems to mount database systems on. This trend is likely to continue. Trusted operating systems are not selling very well and few, if any potential users, have made their use mandatory. Components, in general, have low levels of assurance. What happens to security when you use low-assurance components, that are now more complex, in both hardware and software, and which are strung together in ever-larger strings?

Some of the old solutions may are not too useful anymore. Stand-alone systems are infrequently used and simple client-server systems are used less and less. New solutions and mechanisms may require new security solutions. For example, a new switching and transfer mode, such as the asynchronous transfer mode (ATM), may require new encryption equipment to match the ATM's format and speed. Fortunately some solutions exist for ATM. How good are these solutions? Does the security equipment exist for other new technical solutions? Is there an overall security policy? Is it written? Does the system comply with the security policy?

The widespread use of ActiveX, Java and other mobile codes causes more security concerns. Database management systems are using these codes for connectivity, flexibility and applications. The future promises even more use of these codes. While having many positive qualities, these codes also may have capabilities that the user, or database administrator may not be aware of, such as rebooting a target computer, or capturing passwords and files. Here the selection and proper use of an appropriate product may ease some of the security problems.

Database architectures have become more varied. In addition to 2-tier architectures, such as client/server architecture, there exist 3-tier architectures, that could include a transaction server, and n-tier architectures that may have multiple presentation, application and data tiers as well as a transaction manager. Is your security solution appropriate for the architecture that you are working with? Further considerations include whether the Distributed Component Object Model (DCOM) or Common Object Request Broker Architecture (CORBA) is being used.

Bigger databases, terabytes in length, exist. Data warehouses and marts are common. Data mining is in. Inference and aggregation, using a variety of techniques, is a prime goal. How do you prevent some and allow others to use these techniques?

What do users desire? They desire much more functionality, with widely distributed systems over varying communications, while preserving ease-of-use. These systems are complex, requiring the use and integration of many protocols. E-commerce has increased the perception of the user that data has real value, and with the additional perception that their systems may be vulnerable, users are requesting strong Identification and Authentication. Non-repudiation is now very important. In part, to satisfy these goals, smart cards, PCM cards, certificates or biometrics devices may be used. Users also want strong access controls to determine who accesses their database or file and what portion of a database an individual can see. Very important to the user is ease-of-use with many hoping for a single signon. The easiest way to have security bypassed is to make it too hard or complicated. Multiple, hard to remember, frequently changed passwords, which although in theory is good security, is a good example of a methodology that is frequently circumvented.

3. Tools/Solutions

Considering all these problems and complexities, what do we have to work with to build a secure system? First, we can look to mechanisms in existing commercial database management systems. The traditional roles, passwords, privilege settings, audit logs and backups still exist and are usable. New features have been added in some systems to make use of, for example, smart cards, Kerberos, CORBA compliance, encryption, certificates, including X.509 certificates, public key directories, and biometric authenticators.

Second, network and component operating systems still have security mechanisms and may also use some of the newer ones mentioned in the previous paragraph. In addition, OS and DBMS may be coupled in Trust relationships between the two.

Third, there has been a tremendous growth in firewalls of various types, intrusion detection systems, hashing integrity schemes to insure file integrity, public keys including infrastructure, virtual private networks, stronger encryption, and better use of encryption, including public key encryption. A potential problem is how well the Database System can use these improvements. Are the security features transparent to the user? Has the database vendor provided appropriate interfaces to “seamlessly” use these features? How easy are they to use? Do I have to write my own

interfaces, or other code? How much assurance can I place in each feature and in the system security as a composable whole?

Fourth, one practical method to improve security is to limit the functionality of the system. A system that just needs to answer HTTP queries can be built more securely than a system that has to handle many protocols as protocol filtering is easier. A "read only" system is easier to protect than one where users can write into the database. If necessary, the users could enter data into another system, where it can be checked and verified before being copied over to the database. Or, I could use a one way mechanism, such as one way fiber, to fill a database, and then periodically refresh the database, a means of "throwing the data over the fence"?

Fifth, an Extranet or Intranet is easier to protect than the Internet, because I only have to worry, about "insiders", who can be screened and more easily disciplined. One level systems are easier to protect than multi-level systems. With one-level systems, I worry less about write-downs or spillage as I do with multilevel systems.

Sixth, If I need a multilevel system, because, say, ease of use, I could form a composite system based on risk. If the group that is putting information into the database is known and trusted to me, I may be willing to let them work at multiple levels, in effect, using a MLS system to input into a database. I may force each submitter to sign his data entry. If the groups that is obtaining data from the database only require it at one level, then I may restrict them to that level and to read only. If the groups that read the data read it in only one format, then I can protect even further.

Seventh, system configuration, as usual, is a weak point. Does system configuration for the system exist? Is the system baselined? Is there system configuration documentation? What documentation exists? What are the procedures for changing hardware or software? Have all the relevant patches been applied? How do you know? How does the next person, who replaces you, know? Are you using the latest version of virus detecting software? How do you know? And so on. It is very easy to get into problems at this point.

Eighth, defense in depth, using intrusion detection, operating systems with differing architectures, filters, firewalls, guards and encryption may increase security, if the system is designed properly.

Ninth, the entire system must be looked at. Remember we are protecting the data. In one system, I was asked to look at the security of a database server, and I did. But you can not stop with just looking at a component. An input to the system was downgraded data, and there was, in my opinion, much more risk associated with the downgrading equipment and procedures than there was with the single-level database server.

Tenth, as systems become more complex, standards, at all levels, become even more important. Levels of software, including security software have to interface, as do the various components. It's great if all hardware and software can be purchased from one vendor, both for compatibility and accountability reasons, and, also, hopefully, for security policy consistency. However, it has been my experience that we usually have to use multiple vendors, and, often, provide custom code to complete the system solution.

Eleventh, try to avoid projects where security is thought of as an after-thought. The ideal system is built from a pre-established list of functional and security requirements. I've had the good fortune of doing this once. However, the later in the system development security is regarded as needed, the poorer the security result is likely to be.

Finally, the physical, personnel, and procedural techniques that we have used in the past to secure systems are all the more important now, in this new era of complex, buggy yet critical systems.