Secure Your Database

Your database is the foundation of your e-business. Follow this advice to avoid compromising your strongest corporate asset—your data.

By Bill Jaeger, METASeS, Director of Applied Research

Can you mitigate database security risks?

Yes!

- Separate Web and database servers
- Implement strict access controls and permissions
- Use database views, stored procedures, and encryption

But...

- It's complicated
- Some approaches to using individual keys for encryption may raise significant key management issues
- Encryption impacts system performance

E-business applications are complex, consisting of many interdependent pieces, most importantly the database. Whether flat file, object, or relational, the database forms the foundation for all but the simplest e-business systems. The roles of databases are many, including storing system configuration parameters and credentials, providing raw content for use in dynamic page generation, providing data for order fulfillment, and storing customer data such as shipping addresses and credit card numbers.

Regardless of the specific product or application, the security and integrity of the data contained within the database is of great concern. After all, this information keeps your e-business running. Unauthorized modification of even a single piece of information within a database can lead to bad publicity, lawsuits, or the shut-down of a business.

Despite the sensitive system, business, and customer information typically contained within databases, first-hand experience and recent media coverage of compromised Web sites show that databases are at risk.

Reality check: Companies are failing to implement adequate security measures.

The risk is so great, in fact, that the world's largest payment network, Visa, is establishing mandatory Internet security guidelines that merchants must follow. American Express is going one better with its Private Payments program, offering cardholders a unique, one-time use credit card number for each online transaction.

From the outside: Protect your server

A critical aspect of maintaining the security and integrity of the data within a database involves the overall system and network security that supports the database server. Generally, an attacker who gains access to the operating system or physical server supporting a database can easily gain full access to all of the data contained within that database, regardless of the access controls enforced by applications or the database itself.

Key defenses: Eliminate unnecessary network services, separate Web and database servers, patch known security vulnerabilities, and ensure correct file and device permissions.

Secure and disable unnecessary network services

Database software, like most operating systems and complex applications, provides a number of services that allow remote system management, distributed processing, and other network-related functions. In many cases, those services are enabled by default and are often "protected" by using either no password or a vendor-supplied default password. Those services can be attractive targets for an attacker, potentially allowing easy access to the system and database.

To mitigate risks associated with passwords, change all default passwords. Additionally, disable all unnecessary services on both the server and the database itself, such as Finger and remote administration, to block an attacker from using them as an entry point into the system. For required services, restrict access through filtering mechanisms, such as firewalls and routers, as described later in the article.

Separate Web and database servers

Many e-businesses place all their servers, including Web, application, and database servers, behind a single, dual-interface firewall or choke router. While this prevents

Bill Jaeger is director of applied research at METASeS, a full service provider of end-to-end e-business security solutions and services. http://www.metases.com.

For a determined attacker, even the most stringent network safeguards are insufficient.

(with appropriate configuration) Internet traffic from reaching your database servers, it doesn't stop a malicious user from compromising your Web server and using it to attack your database server. Should any single server be compromised, it can be used to subject the other servers to a full barrage of attacks.

Isolate your database servers, particularly those containing sensitive information, from a Web site's demilitarized zone (DMZ) and locate them on a physically separate network segment from the Web and other Internet-accessible servers that support your e-business. Ideally, partition your database server off from the Web servers by a dedicated firewall. This firewall, illustrated in figure 1, should only allow database traffic between the Web server and database server. You should deny and log all traffic from any other location, or other types of traffic from the Web server, such as Telnet.

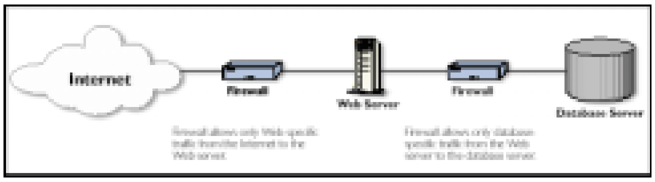
Internet
Firewall
Web Server
Firewall
Database Server
Firewall allows only Web-specific traffic from the Internet to the Web server.
Firewall allows only database-specific traffic from the Web server to the database server.

Figure 1: Logical depiction of physically separate Web and database servers—Partition your database and Web servers by a dedicated firewall.

Caution: Through such partitioning, your database can, at worst, be attacked using only native database protocols. Exploiting weaknesses in the database communication protocols can still lead to a system compromise.

Tip: In cases where it isn't possible to place a dedicated firewall between the Web and database servers—because of cost reasons, for example—it's still possible to receive some of the same benefit by implementing firewall-like filters directly on the database server. For UNIX servers, you may be able to use IP Chains, IP Filters, or TCP Wrappers. For Windows NT or Windows 2000 servers, you can use the filtering capability within the Network Connections "TCP/IP Security" configuration screen.

Eliminate known security vulnerabilities

As with applications and operating systems, database servers can also have vulnerabilities that lead to unauthorized data access, loss of integrity, or total system compromise.

Several versions of Oracle 8 for UNIX, for example, include an incorrect permission setting on the oratclsh TCL command interpreter used in conjunction with the Intelligent Agent option. This incorrect setting lets any attacker open a command shell—even one operating with unprivileged guest access—to gain root or administrative access to the entire system. Once an attacker gains administrative access to a database server, he can gain full access to all of the data stored therein.

New reality: Other vulnerabilities can be exploited directly over the network, eliminating the need for an attacker to first gain access to an account local to the database server. One such vulnerability that's particularly devastating is the Remote Data Services (RDS) bug contained within certain versions of Microsoft's Active Data Objects (ADO) architecture. This bug has led to the compromise of many Web servers and their supporting databases, reportedly including those of Microsoft and U.S. federal agencies.

To minimize the impact of vulnerabilities, keep your systems up-to-date with security patches released by vendors. Security patches on production systems are often not kept current. Microsoft's RDS bug, for example, is still being actively exploited today, more than a year-and-a-half after it was discovered and patches were released by Microsoft to fix the problem.

Guard permissions and least privilege

Beware: Incorrect or gratuitous file permissions can pose threats similar to those of product vulnerabilities. The damage that incorrect file and device permissions can cause is limitless. Overly liberal permission settings on a device allow unauthorized access that can undermine the security of an entire file system or database; inappropriate permissions on Windows NT systems allow backup copies of the Security Account Manager (SAM) password database to be read by everyone.

What to do: To protect against attacks that leverage inappropriate file permissions, the database or system administrator should investigate the nature of all critical files supplied with, or relied upon by, the database. In addition, applications should be run under the principle of least privilege, meaning that each application is run with the minimum set of permissions necessary to perform the required tasks.

While it may be easier to run every application with system-level privileges (for example, root or administrator privileges), doing so virtually guarantees that attackers can compromise an entire system should they locate a single vulnerability.

On the other hand, if each application runs with the absolute minimum set of privileges required, an attacker that compromises one application would, at worst, be able to compromise other applications sharing the same set of privileges.

Tip: To limit the damage that one compromised program could potentially do to other applications, it often makes sense to have each application running under a distinct set of limited privileges.

On the inside: Protect your data

Databases, particularly relational database management systems (RDBMSs), can be very complicated to install, configure, and maintain. In part because of this complexity, often combined with developers' or security practitioners' unfamiliarity with database configuration and internals, database security precautions are often inadequate.

Key defenses: Use database access controls, restrict access through views and stored procedures, and use encryption to protect the integrity and confidentiality of information.

Use database access controls

Without the ability to selectively grant and restrict access to a database and its data, arbitrary users can add and delete information at will. Even if access controls are enforced by

Web applications, data contained within the database is still at risk if a malicious user circumvents the Web application and accesses the database directly.

Caution: Unfortunately, it's quite common for developers to place all access controls within the application itself, leaving the database exposed. This practice is typically reflected in the use of a generic application user ID that has full reign over the database tables and, in some cases, the entire database. In such a case, an attacker who circumvents application-enforced access controls could erase an entire database with a single `DROP DATABASE` command. Without current backups, all data would be lost and your e-business at risk.

Most databases, however, support some form of access control that can restrict what users, groups of users, or applications can access or change the database. In their simplest form, database access controls may leverage the underlying operating system's file permission capabilities. More sophisticated databases, such as an RDBMS, provide multiple layers of fine-grained, object-level access controls that can minimize or eliminate the need to rely upon the Web application for database access control enforcement.

Limit access through views and stored procedures

Two powerful mechanisms for restricting access to data stored within an RDBMS are views and stored procedures. Used appropriately, these mechanisms can provide an effective means of protecting data by limiting the amount of information that an application, and malicious users attempting to exploit that application, can access.

A view is a virtual database object derived from a query against one or more database tables. A view contains no data, but can be thought of as a window into a subset of rows and / or columns from the underlying tables it represents. Figure 2 illustrates a logical representation of a view.

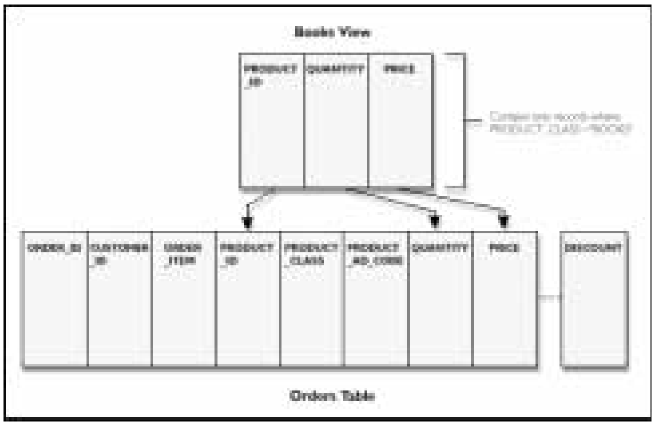

Figure 2: An example of a view—A view contains no data, but provides a window into a subset of rows and/or columns from the underlying tables that it represents.

Most databases provide the ability to define a different set of access privileges to a view than the underlying tables. By creating views and assigning user or group permissions to those views, but not to the underlying tables, you can restrict access to selected columns and rows. Users then only have access to the data allowed by a view; the rest of the data is not visible or accessible.

At worst, circumventing application-enforced access controls that operate against a view reveal the subset of data represented by the view. Compare that to circumventing application-enforced access controls that operate against an entire table. Doing so would potentially reveal the contents of the entire table or tables. Even if column restrictions were enforced through the use of the access controls described previously, a malicious user would still be able to see all the records in the table(s), as table-level access controls can't be applied on a row-by-row basis. Depending upon the application, this could provide a malicious user with a considerable amount of sensitive information.

One more idea: While views provide a robust mechanism for enforcing read-only access, you can use stored procedures to enforce access controls on both read and write operations. Stored procedures provide a mechanism for encapsulating business logic within the database, thereby abstracting an application's interactions with the database. As with views, a user typically requires only permission to execute a stored procedure, and not permission to access the underlying database tables.

By encapsulating all database access in stored procedures, and letting only stored procedures be executed against a database, a malicious user is prevented from originating or modifying queries and submitting them directly to the database for execution. At worst, this constrains a malicious user's access to the database using the defined stored procedures. As such, you can protect data within the underlying tables from direct manipulation.

Stored procedures can encapsulate complex logic such as validating user input, transforming it, and then executing dozens of individual SQL calls. This could, for example, let an application populate the dozen tables required to create a new account at an e-commerce Web site with a single parameterized stored procedure call. For example:

```
ADD_CUSTOMER ('First', 'Last');.
```

Encrypt communication between servers

In much the same way that intercepting and potentially altering sensitive communications, such as credit card numbers, between Web browsers and Web servers is a concern, intercepting and altering the communication between Web application servers and database servers may also be an issue.

Caution: You may need to heighten this concern in cases where server-to-server communication occurs across a non-trusted network such as an organization's DMZ or the Internet.

Similar to the use of Secure Sockets Layer (SSL) encryption to protect data transmitted between Web browsers and Web servers, you can also use encryption for communication between Web application servers and database servers. Depending upon your network architecture and the products deployed at your site, server-to-server encryption can encrypt all data transmitted using a mechanism such as IPSec, or only encrypt database-specific data using encryption-enabled database protocols. Oracle SQL*Net V2.0 and Net8, for example, support data encryption between systems. (For more on IPSec, see the October 2000 issue of INTERNET SECURITY ADVISOR for an article by Ken Masica.)

Tip: You can also use freely available tools like stunnel to provide SSL encryption and server authentication to otherwise insecure protocols.

Encrypt data

Reality check: Despite employing all these mitigation strategies, it's still possible for an attacker to access the contents of your database by exploiting inappropriately configured security perimeters, an as-of-yet undisclosed product vulnerability, or salvaging an inappropriately disposed of backup tape. For a determined attacker, even the most stringent network safeguards, of which firewalls and SSL are a component, are insufficient.

Given the sensitive information that the typical e-business stores in its databases, such as credit card numbers, customer identity information, and passwords, the risk of data theft is very real.

In November 1999, CD Universe's database compromise reportedly resulted in the release of more than 300,000 credit card numbers. Once compromised, those credit card numbers were freely distributed on the Internet. More recently, Western Union's September 2000 compromise revealed 15,700 credit and debit card numbers.

Key defense: The last resort against such compromises is to encrypt sensitive—if not all—data that's stored in databases. While encryption won't prevent the attacker from retrieving the encrypted data stored in the database, it will prevent the attacker from being able to recover and use the plain text contents of the data. Unless the attackers can decrypt the data, they'll be left with meaningless gibberish.

Encrypting the data using a proven, strong encryption algorithm such as 3DES, Blowfish, IDEA, or RC4, combined with strong keys and proper implementation, makes it impossible for an attacker to access the encrypted data without discovering the encryption keys used.

Recommendation: For symmetric encryption algorithms, use a key of at least 112-bit strength. Don't store encryption keys in their entirety in the application, on disk, or within the database. Instead, distribute raw components of key material throughout the application, and then assemble and transform it as needed. When no longer needed, destroy the key and overwrite the memory used to store the key.

Best case scenario: Ideally, a unique key protects each customer's sensitive information or a single set of related information, thereby minimizing the damage that can be done should an attacker guess or use brute force to access a single encryption key. At best, guessing one key should reveal only one customer's information or related set of information. Some approaches to using individual keys may raise significant key management issues. Such issues must be thoroughly evaluated and understood prior to implementation. If possible, a cryptographic expert should help design and verify your cryptographic and key management plans.

The performance impact of encryption

Be prepared: Despite its benefits, encryption does exact a performance penalty against e-business applications and servers. This performance penalty increases with the amount, type, and strength of encryption performed. Generally speaking, asymmetric, public key encryption has a greater performance impact than symmetric encryption. Further, stronger keys—that is, more bits—for a given type of encryption have a greater performance impact than smaller keys.

Reality check: Striking the proper balance between acceptable performance and acceptable security can be tricky. Unfortunately, offering blanket guidance on the appropriate balance isn't possible because that balance must be evaluated on a case-by-case basis.

Note, however, that offloading cryptographic computations to specialized hardware devices can lessen the performance impact of encryption. Several vendors make hardware-based cryptographic accelerators, including Intel, nCipher, and Rainbow Technologies.

Pour a concrete foundation

As e-business continues to flourish, so too do the security risks and exposures that you and your customers must face. Day-in and day-out, countless Web sites ask customers to entrust sensitive information to their databases without any real assurance that the data contained therein is safe. High-profile exposures call for greater security measures from credit card companies, but target-the-effect-not-the-cause moves only underscore that far too many e-businesses are built upon a foundation of sand.

Understanding the exposures and corrective actions outlined in this article are a strong first step towards replacing—or hopefully altogether avoiding—a shaky foundation.

TOOLS, ALGORITHMS, & VENDORS MENTIONED IN THIS ARTICLE

For information on the tools mentioned in this article, see:

IP Chains, by Rusty Russell, et al
http://netfilter.samba.org/ipchains

IP Filter, by Darren Reed
http://coombs.anu.edu.au/~avalon/ip-filter.html

Stunnel, by Michal Trojnara
http://www.stunnel.org

TCP Wrappers, by Wietse Venema
ttp://ftp.porcupine.org/pub/security/index.html

For information on cryptographic algorithms mentioned in this article, see:

RSA Laboratories' Frequently Asked Questions
http://www.rsasecurity.com/rsalabs/faq

Applied Cryptography, by Bruce Schneier (John Wiley & Sons, 1995)

For information on the cryptographic accelerators mentioned in this article, see:

Intel NetStructure e-Commerce Accelerators,
http://www.intel.com/netstructure/ecommerce_equipment.htm

nCipher SSL nFast Acceleration Products,
http://www.ncipher.com

Rainbow Technologies CryptoSwift SSL Acceleration Products
http://www.rainbow.com

Coming up

Look ahead to more articles during 2001 by security expert Bill Jaeger on securing critical components of your e-business. ADVISOR