CHAPTER 23

Database Security

Abstract

This chapter looks at a wide range of database security issues. It begins by looking at external security threats, those threats that are shared with the larger networked environment in which databases exist. The discussion continues with internal threats, those that are unique to databases. The following major section of the chapter covers general solutions to external security threats. The discussion of internal solutions includes the SQL GRANT statement. The final section of the chapter covers backup and recovery solutions.

Keywords

database security
database security threats
database security solutions
database access rights
authorization matrix
SQL
SQL GRANT
database backup
database recovery

In our current computing environment, we usually think that the instant world-spanning access provided by the Internet is a good thing. However, that access has a dark side: those who would inadvertently or purposefully violate the security of our data. Security has always been a part of relational database management, but now it has become one of the most important issues facing database administrators.

One way to look at security is to consider the difference between security and privacy. Privacy is the need to restrict access to data, whether they are trade secrets, data that can be used as a basis for identity theft, or personal information that by law must be kept private. Security is what you do to ensure privacy.

Many people view network security as having three goals:

- Confidentiality: ensuring that data that must be kept private, stay private.
- Integrity: ensuring that data are accurate. For a security professional, this means that data must be protected from unauthorized modification and/or destruction.
- Availability: ensuring that data are accessible whenever needed by the organization. This implies protecting the network from anything that would make it unavailable, including such events as power outages.

One of the things that makes data theft such a problem is that it is possible to steal data without anyone knowing that a theft has occurred. A good thief will be able to enter a target system, copy data, and get out without leaving a trace. Because copying digital data does not affect the source, examining the data won't reveal that any copying has occurred. An accomplished thief will also modify system log files, erasing any trace of the illegal entry. Although major data breaches are reported widely in the media, it's a safe bet to say that there have been several times as many that go undetected.

The popular media (television, newspapers, and so on) would have you believe that the cause of almost all computer security problems is the "hacker." However, if you ask people actually working in the field, they will tell you that many of the security breaches they encounter come from sources internal to an organization and in particular, employees. (At one point, more than half of security problems were caused by employees; this number has diminished as Internet intrusions have increased.) Therefore, this does mean that it won't be sufficient to secure a network against external intrusion attempts; you must pay as much attention to what is occurring within your organization as you do to external threats. Databases, in particular, are especially vulnerable to internal security threats because direct access is typically provided only to employees.

Sources of External Security Threats

The Internet has been both a blessing and a curse to those who rely on computer networks to keep an organization in business. The global network has made it possible for potential customers, customers, and employees to reach an organization through its Web site. But with this new access have come the enormous problems caused by individuals and groups attempting illegal entry into computer networks and the computer systems they support.

Physical Threats

We are so focused on security issues that come over a network that we tend to ignore physical threats to our database installation. Certainly, we need to worry about a disgruntled employee with a hammer, but there is more to physical security risks than that. The major issue today is physical access to a server by a knowledgeable data thief.

All servers have at least one user account that has access rights to the entire computer. When the servers are housed in a locked location

All servers have at least one user account that has access rights to the entire computer. When the servers are housed in a locked location, operators tend to leave the privileged user account logged in. It makes administration just a bit easier. The operators rely on the security on the server room door to keep unauthorized people out. The problem with this strategy is that sometimes physical barriers aren't sufficient; a knowledgeable data thief will be able to circumvent whatever lock has been placed on the server room door and gain physical access to the servers. If the privileged accounts are left logged in, all the thief needs to do is sit down at a machine and start extracting data.

Hackers and Crackers

External threats are initiated by people the general population calls "hackers." Initially, however, the term "hacker" referred to someone who could write an ingenious bit of software. In fact, the phrase "a good hack" meant a particularly clever piece of programming. As with many technological terms, however, the meaning changed when the term entered the mainstream and therefore today anyone who attempts illegal access to a computer network is called a hacker.

There are many ways to classify those who break into computer systems, depending on which source you are reading. However, most lists of the types of hackers include the following (although they may be given different names):

- *White hat hackers*: This group considers itself to be the "good guys." (Whether they actually are good guys is open to question, however.) Although white hat hackers may crack a system, they do not do it for personal gain. When they find a vulnerability in a network, some hardware, or a piece of software, they report it to the network owner, hardware vendor, or software vendor, whichever is appropriate. They do not release information about the system vulnerability to the public until the vendor has had a chance to develop and release a fix for the problem. White hat hackers might also be hired by an organization to test a network's defenses.
  White hat hackers are extremely knowledgeable about networking, programming, and existing vulnerabilities that have been found and fixed. They often write their own system cracking tools.
- *Script kiddies*: The script kiddies are hacker "wannabes." They have little, if any, programming skill and therefore must rely on tools written by others. Psychological profiles of script kiddies indicate that they are generally male, young (under 30), and not socially well-adjusted. They are looked down upon by most other hackers.
  Script kiddies usually do not target specific networks, but instead scan for any system that is vulnerable to attack. They might try to deface a Web site, delete files from a target system, flood network bandwidth with unauthorized packets, or in some other way commit what amounts to cyber vandalism. Script kiddies typically don't want to keep their exploits secret. In fact, many of those that are caught are trapped because they have been bragging about what they have done.
- *Black hat hackers*: Black hat hackers are motivated by greed or a desire to cause harm. They target specific systems, write their own tools, and generally attempt to get in and out of a target system without being detected. Because they are very knowledgeable, and their activities often undetectable, black hat hackers are among the most dangerous.
- *Cyberterrorists*: Cyberterrorists are hackers who are motivated by a political, religious, or philosophical agenda. They may propagate their beliefs by defacing Web sites that support opposing positions. Given the current global political climate, there is also a reasonable fear that cyberterrorists may attempt to disable networks that handle utilities such as nuclear plants and water systems.

Types of Attacks

When a hacker targets your network, what might you expect? There are a number of broad categories of attacks.

- *Denial of service*: A denial of service attack (DoS) attempts to prevent legitimate users from gaining access to network resources and, by extension, any database that uses the network. It can take the form of flooding a network or server with traffic so that legitimate messages can't get through or it can bring down a server. If you are monitoring traffic on your network, a DoS attack is fairly easy to detect. Unfortunately, it can be difficult to defend against and stop without disconnecting your network from the Internet.
- *Buffer overflow*: A buffer overflow attack takes advantage of a programming error in an application or system program. The hacker can insert his or her own code into a program and from there take control of a target system. Because they are the result of a programming error, buffer overflow conditions are almost impossible for a network engineer to detect. They are usually detected by hackers or the software vendor. The most common defense is a patch provided by that vendor.
  Closely related to the more general buffer overflow vulnerability, a *SQL injection attack* occurs when a hacker uses openings in SQL statements to insert and execute malicious code from a database application. Such attacks can be prevented by following specific syntax forms when embedding SQL in high-level programming languages. It is therefore important that application coding syntax rules be documented, updated, and readily accessible to application programmers.
- *Malware*: The term "malware" includes all types of malicious software, such as viruses, worms, and Trojan horses. The goal of a hacker in placing such software on a computer may be simple maliciousness or to provide access to the computer at a later date. Although there is a constantly escalating battle between those who write malware and those who write malware detection software, a good virus checker goes a long way to keeping network devices free from infection.
- *Social engineering*: A social engineering attack is an attempt to get system access information from employees using role-playing and misdirection. It is usually the prelude to an attempt to gain unauthorized access to the network. Because it is "no tech," social engineering is widely used, and can be very effective.
- *Brute force*: One way to gain access to a system is to run brute force login attempts. Assuming that a hacker knows one or more system login names, he can attempt to guess the passwords. By keeping and monitoring logs of who is attempting to log into a system, a network administrator can usually detect brute force break-in attacks.

Note: There is no gender discrimination intended with the use of the pronoun "he" when referring to hackers. The fact is that most hackers are male.

Sources of Internal Threats

Most internal threats come from two sources: employees and accidents. Employee threats may be intentional or accidental, as well.

Employee Threats

In most cases, employees know more about a network and the computers on it than any outsider. At the very least, they have legitimate access to

user accounts. IT personnel, of course, have various levels of increased access. Intentional employee security threats include the following:

- Personnel who employ hacking techniques to upgrade their legitimate access to root/administrator access, allowing them to divulge trade secrets, steal money, and so on, for personal or political gain.
- Personnel who take advantage of legitimate access to divulge trade secrets, steal money, and so on, for personal or political gain.
- Family members of employees who are visiting the office and have been given access to company computers to occupy them while waiting.
- As mentioned earlier, personnel who break into secure machine rooms to gain physical access to mainframe and other large system consoles.
- Former employees, especially those who did not leave the organization willingly, who are interested in revenge. Attacks may be physical, actually damaging equipment, or traditional hacking attacks, usually resulting in damaged data.

As dangerous as the intentional employee security threat may be, employees can also cause a great deal of damage unintentionally, such as

- Becoming the victim of a social engineering attack, unknowingly helping a hacker gain unauthorized network access.
- Unintentionally revealing confidential information.
- Physically damaging equipment, resulting in data loss.
- Misusing a system, introducing inaccurate and/or damaged data or accidentally deleting or modifying data.
- Installing personal equipment on a network (for example, a wireless access point) that isn't included in the organization's security measures.

Most unintentional employee threats, theoretically, can be handled through employee education. For example, it seems logical that instructing employees not to write passwords on sticky notes that are then fixed to monitors would help prevent compromised passwords. However, when you are dealing with human beings, even the best education can be forgotten in the stress of getting a job done on time.

Employees certainly can unintentionally damage a network. In addition, true accidents also occur. A security plan will need to guard against data damage and loss caused by

- Electrical power fluctuations
- Hardware failures
- Natural disasters, such as fire and flood

Guarding against accidental network damage includes power protection (for example, surge protectors, and UPSs) and comprehensive backup schemes. When done well, backup takes significant planning and disaster recovery rehearsals.

External Remedies

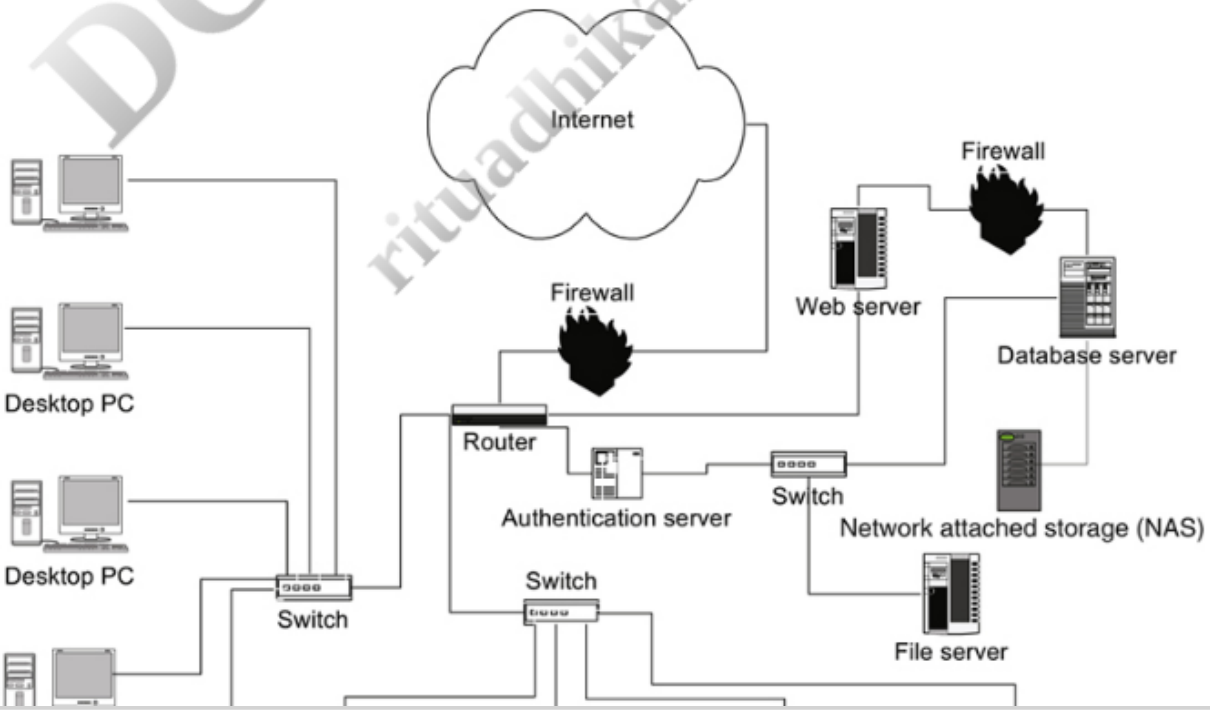
External security measures are typically outside the responsibility of database administrators. However, you should be aware of what security protections external to the database are in place and, if they appear inadequate, know how to make your concerns and recommendations heard. In this section, we'll therefore look at security strategies that should be in place, external to the database, to provide a first line of defense.

Securing the Perimeter: Firewalls

A *firewall* is a piece of software that filters incoming and outgoing network traffic, and stops messages that violate the rules that define allowable traffic. It is typically placed between the Internet and an internal network. Its primary job is to eliminate as much undesirable network traffic as possible.

Note: You may hear a firewall spoken of as a piece of hardware. However, a firewall device is really a special-purpose computer that runs firewall software. Because the device is dedicated to the firewall application, it may be more powerful than firewall software that is added to a router or other network interconnection device.

If you look at Figure 23.1, you'll notice that the network looks very much like something that might be used by SmartMart from Chapter 15. Specifically, it includes a Web server that is exposed to the Internet through a router and a database server that is isolated from the Web server by a second firewall.



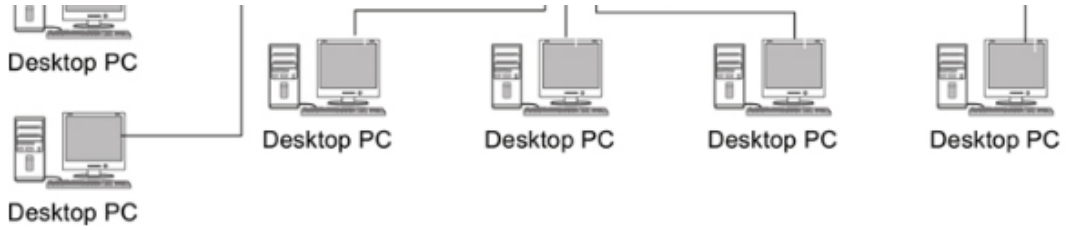


FIGURE 23.1 **Using firewalls to secure a network and create a DMZ to protect the database server.**

The first firewall—the one connected to the edge router—allows specific messages to pass, including those intended for the Web server and for destinations that represent legitimate network traffic (for example, e-mail and remote employees).

The edge router will send all Web traffic to the Web server, preventing it from getting onto the internal network. However, because Web users need access to data stored on the database server, simply routing that traffic to the Web server doesn't provide protection for the database.

To protect the database, only messages from the Web server are permitted to interact with the database. A second firewall has therefore been placed between the two servers. The Web server is said to reside in a DMZ, a part of the network that is walled off from the internal network.

A Web transaction that involves access to the database server proceeds as follows:

1. User's browser generates a request for data stored in the database (for example, a page from a retail catalog) and transmits it to the company network.
2. The edge router passes the request to the Web server.
3. The Web server requests data from the database.
4. The firewall between the Web server and the database server passes the message because it comes from the Web server.
5. The database server retrieves the requested data and sends it back through the firewall to the Web server.
6. The Web server formats the data and sends a response to the user, whose browser displays the new Web page.

Notice that internal users have direct access to the Web server, without having to pass through the DMZ. The assumption is that internal users will be authorized for direct database access.

Handling Malware

Malware infecting a database server can be a serious problem. The result may be loss of data, loss of access to the database, or loss of control of the database server's hardware. Protection against malware is typically provided by "virus protection" software running on firewalls and the servers themselves.

Most current virus protection software handles worms, Trojan horses, and bots, as well as viruses. The most important thing to keep in mind, however, is that there is an ever-escalating battle between those who write malware and those who produce the virus protection software. As soon as a new threat is identified, the software developers rush to add the new malware to their protection database; the malware producers then write new malware that is typically more powerful and sophisticated than previous releases. You can never be completely safe from malware because there is always a lag, however short, between the detection of a new piece of malware and the updating of virus protection software to handle that malware. The best you can do is to update the database that accompanies your virus protection software regularly.

Buffer Overflows

Because a buffer overflow problem is a flaw in the way software is written, there is really nothing an organization without access to the source code and a staff of programmers can do to fix it. An organization must rely on the software developer to release updates (*patches*) to the code.

Patching is a cooperative operation, in that once the vendor has released a patch, it is up to organizations using the software to install the patch. Nonetheless, the best defense against buffer overflow vulnerabilities (and any other vulnerabilities caused by bugs in software) is to apply all available patches.

Patch management can become a nightmare for an IT staff. There are so many patches released for some types of software (for example, Microsoft Windows, in all its myriad versions) that it is difficult to know which patches are important, stable and, in a large organization, which patches have been applied to which machine. Nonetheless, to protect your database server (both the operating system and the DBMS), you will need to work with IT to ensure that all necessary patches have been applied.

Physical Server Security

Physical security requires a two-pronged approach: preventing physical access to the server and, should that fail, securing the administrative accounts. Actual physical methods include any or all of the following:

- Security cameras outside machine/server room doors to record who enters, exits, and loiters in the area.
- Smart locks on machine/server room doors that store the code of each individual who enters and exits, along with the date and time of an entry or exit. Smart locks can be equipped with biometric identification devices if desired. (These will be discussed shortly.)
- Removal of signs from machine/server room doors and hallways so that no one can locate hardware rooms by simply walking the hallways of the building.

The output produced by security cameras and smart locks must be examined regularly to determine if any unusual access patterns appear.

Should an unauthorized person manage to defeat the physical security, he will probably want to gain software access to a computer. (We are assuming that physical damage to the equipment is a rare goal of an intruder.) This means that the administrative accounts for each server must be secured. First, the accounts should never be left logged in. It may be more convenient for operators, but it makes the servers vulnerable to anyone who happens to walk by. Second, login attempts to the administrative accounts should be limited. For example, after three failed attempts to log in, the server should disallow further login attempts for some predetermined length of time.

User Authentication

Any user who is going to have direct access to a database first needs to be authenticated for access to the local area network. The forms that such authentication takes depend on the size and security risks of the network

authentication takes depend on the size and security risks of the network.

Positive user identification requires three things:

- Something the user knows
- Something the user has
- Something the user is

The first can be achieved with passwords, the second with physical login devices, and the third with biometrics.

User IDs and Passwords (What the User Knows)

The first line of defense for any network authentication scheme is the user ID and password. User IDs in and of themselves are not generally considered to be private; in fact, many are based on user e-mail addresses. The security, therefore, resides in the password. General security practice tells us the following about passwords:

- Longer passwords are better than shorter passwords.
- Passwords with a combination of letters, numbers, and special characters (for example, punctuation) are more secure then passwords that are all letters or numbers.
- User education is needed to ensure that users don't share their passwords with anyone, or write them down where others may find them.
- Passwords should be changed at regular intervals.

Although "general wisdom" dictates that passwords should be changed regularly, there are some problems with that policy. When users are forced to change their passwords, they often forget which password they have used. The solution is to write the password down, sometimes placing it in an insecure location such as the center drawer of a desk or even on a sticky note affixed to a monitor.

Login Devices (What User Has)

The second layer of user authentication is requiring that someone attempting to log in present a physical device that only an authorized user will have. The device has some way of making it unique to the user ID.

Login devices include access cards that a user must scan in some way before entering a password and devices that issue one-time passwords that the user keys in as well as a normal password. For example, Union Bank and Trust offers business customers a physical token that generates a single-user six-digit code every 60 seconds. The token itself is small enough to put on a keychain. Use of the code from the token is required for a variety of online financial transactions.[1]

Some authentication tokens require that the token be physically inserted into a computer connected directly to the network. For example, the eToken series from SafeNet consists primarily of devices that are to be plugged into a USB port to authenticate both the user and the user's location.[2]

The advantage of a login device is that it is small—usually small enough to attach to a keychain—so that there is little difficulty for the user to keep the device available. However, if the user doesn't have the device and needs access, there must be either an alternative form of authentication available or the user will not gain access.

Biometrics (Who the User Is)

Biometric identification—identification based on characteristics of a person's body—has long been a part of science fiction. The idea of retina prints, thumb prints, palm prints, and facial scans isn't particularly far-fetched, however. Today you can purchase a mouse with a thumb print reader that sends the print to a computer for authentication. You may need a fingerprint to unlock your smartphone or laptop. Extremely high security installations today can use retina, face, and palm prints to permit physical access; the readers are too large and expensive for mobile use.

VPNs

Remote access to a database is typical today. Users travel; users work from home. The days are long gone where remote users dialed into their work network using a modem and a standard telephone line. Most remote access reaches an internal network over the Internet. The problem facing a database administrator is ensuring that the data remain safe while traveling on the external network.

One commonly applied solution is a *virtual private network* (VPN). VPNs provide encryption for data transmissions over the Internet. Although there are several types of VPNs, many use a security protocol known as *IPSec*, including the VPNs that are built into desktop operating systems such as Windows and Mac OS X.

IPSec encrypts the data, turning it into something that is meaningless to anyone who does not have the secret encryption keys. Even if data are intercepted traveling over the Internet, they will be useless. We say that IPSec provides a secure *tunnel* for data (Figure 23.2.) One type of tunneling encrypts data only when they are actually on the Internet; the other provides *end-to-end* encryption where data are encrypted from the sending computer to the destination network.

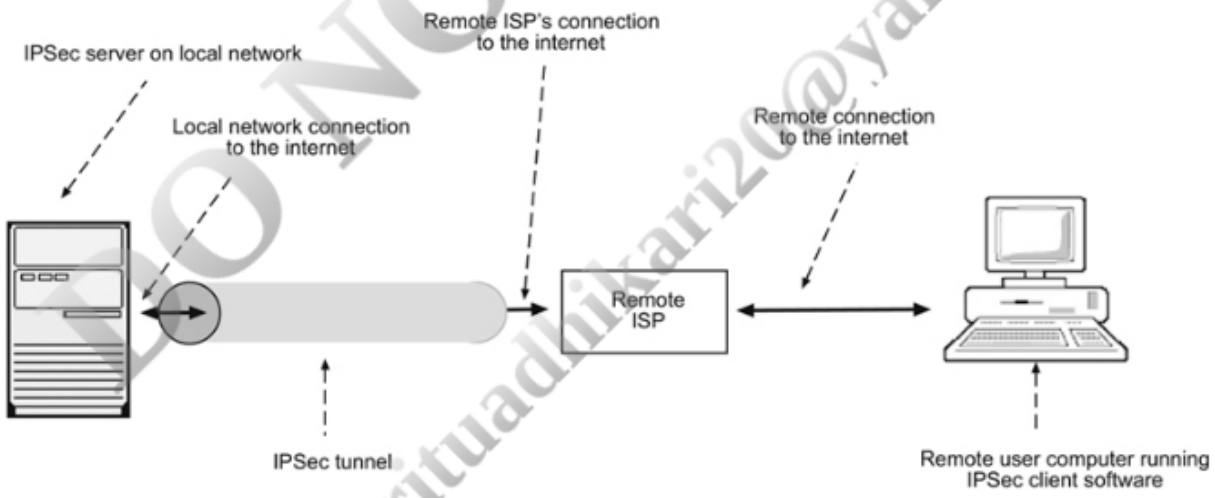


FIGURE 23.2 **The architecture of an IPSec VPN.**

To operate a VPN, the destination network must have a VPN server, a machine that receives all incoming VPN traffic and handles authenticating the VPN user. The VPN server acts as a gatekeeper to the internal network. Once authenticated, VPN users can interact with the destination network as if they were physically present at the network's location.

Combating Social Engineering

Social engineering isn't a technical attack at all—it's a psychological/ behavioral attack—and, therefore, can't be stopped by technical means. It requires employee education to teach employees to recognize this type of attack, and how to guard against it.

As an example, consider the following scenario: Jane Jones is the secretary for the R&D department of a high tech firm. Her boss, John Smith, often works at home. Ms. Jones has been with the company for many years, and she is a trusted employee. She knows the user names and passwords for her computer, Mr. Smith's desktop and laptop, and Mr. Smith's mainframe account.

One morning, Ms. Jones receives a telephone call. "Ms. Jones, this is James Doe from IT. I have some upgrades that I need to install on your computer and Mr. Smith's computer. I don't need to come to your office. I can do it over the network, if I have the user IDs and passwords."

"Oh, that sounds fine," says Ms. Jones. "I hate it when IT has to come by and interrupt my work to fix something. My user ID is …." And she gives Mr. Doe the user names and passwords, just as he requested.

gives Mr. Doe the user names and passwords, just as he requested.

Unfortunately, the man who claims to be James Doe isn't who he says he is. He's a hacker, and, with some simple research and a phone call, has received access to two corporate desktops. First, he checked the corporate directory online. It was simple to find the name of an IT employee, as well as the names of the head of R&D and his secretary. Then, all he had to do was to place his phone call to Ms. Jones. She cooperated, and he was in business.

The phony James Doe does install a file or two on each of the compromised computers. However, the files are actually a Trojan horse that he can activate later when he needs control of the compromised machines.

Ms. Jones made one critical error: she revealed user names and passwords. She felt comfortable doing so because she knew that someone named James Doe worked for IT. She had never thought that there would be a problem trusting someone from that department; the idea of an impersonator never crossed her mind.

There is no technological solution to a social engineering attack of this type. The best prevention is employee awareness. Training sessions with role playing to demonstrate how such attacks are perpetrated can be very helpful. A few simple policies will also go a long way:

- Never reveal your user ID or password to anyone, no matter who that person claims to be.
- If someone claims to be a corporate employee and asks for your user name and password, take their name, supervisor's name, and extension. Then hang up and call the supervisor to report the attempt to obtain a user ID and password.
- If someone claims to be a vendor representative (or anyone else who is not an employee) and asks for a user ID and password, hang up and notify IT security.

An organization should also take steps to restrict the information that it makes public, so that it becomes more difficult for a hacker to develop a convincing social engineering attack. For example, employee directories should not be available publicly. Instead, use titles (such as "IT Manager") in contact lists accessible to non-employees. Organizations with registered Internet domain names should also restrict the information available to those who perform a "whois" search on the domain name.

Handling Other Employee Threats

There are many things an organization can do to guard against other employee threats. They include the following:

- Develop and enforce policies and procedures for users who want to install their own hardware and software on corporate machines. Use network discovery and mapping software to monitor network hardware and detect any unauthorized equipment.
- Develop and enforce policies for users who wish to use personal computing devices on the company network.
- Conduct employee training sessions to familiarize employees with the organization's policies on the release of information.
- Document all organizational security policies, distribute to employees, and require employees to sign, indicating that they have read and accepted the policies.
- Require employees to take two consecutive weeks of vacation at least once every two years. If an employee is hacking the organization's information systems and covering up the unauthorized access, an absence of two weeks is likely long enough to expose what is occurring.
- When an employee is going to be fired, disable all of the employee's computer accounts prior to telling the employee about the termination.

Internal Solutions

To this point, the security measures we've discussed have all been applied outside the DBMS. They are put in place by network administrators, rather than database personnel. There are, however, at least two layers of security that a relational DBMS adds to whatever is supplied by the network.

Internal Database User IDs and Passwords

The user IDs and passwords we discussed earlier in this chapter are used to give a user access to a network. They do not necessarily (and probably shouldn't) give access to a database. Most of today's relational DBMSs provide their own user ID and password mechanism. Once a user has gained access to the network, he or she must authenticate again to gain direct access to the database (either at the command line or with an application program).

It is important to distinguish between direct access to the database and account access by Web customers. Someone making a purchase on a Web site does not interact directly with the database; only the Web server has that type of access. A Web user supplies a user name and password, both of which are probably stored in the database. The Web server sends a query to the DBMS to retrieve data that match the user ID/password pair. Assuming that a matching account is found, the Web server can then send a query to retrieve the Web user's account data. Web customers cannot issue ad hoc queries using a query language; they only can use the browser-based application provided for them. Therefore, there is little that the typical Web user can do to compromise the security of the database.

However, internal database users have direct access to database elements for which their account has been configured. They can formulate ad hoc queries at a command line to manipulate the tables or views to which they have access. The trick, then, is to tailor access to database elements based on what each user ID "needs to know." A relational database accomplishes this using an authorization matrix.

Authorization Matrices

Most DBMSs that support SQL use their data dictionaries to provide a level of security. Known as an *authorization matrix*, this type of security provides control of access rights to tables, views, and their components. Like other structural elements in the database, the authorization matrix is stored in tables along with the data dictionary.

Types of Access Rights

By default, the user who created a database element is the only user that has access to that element. Nonetheless, access rights can be *granted* to other users. There are six types of access right that you can grant:

- SELECT: allows a user to retrieve data from a table or view.
- INSERT: allows a user to insert new rows in a table or updatable view. Permission may be granted to specific columns, rather than the entire database element.
- UPDATE: allows a user to modify rows in a table or updatable view. Permission may be granted to specific columns, rather than the entire

database element.
- DELETE: allows a user to delete rows from a table or updatable view.
- REFERENCES: allows a user to reference a table column as a foreign key in a table that that he or she creates. Permission may be granted to specific columns, rather than the entire database element.
- ALL PRIVILEGES: give a user all of the preceding rights to a table or view.

By default, granting access rights to another user does not give the user the right to pass those rights on to others. If, however, you add a WITH GRANT OPTION clause, you give the user the ability to grant the rights that he or she has to another user.

Using an Authorization Matrix

Whenever a user makes a request to the DBMS to manipulate data, the DBMS first consults the authorization matrix to determine whether the user has the rights to perform the requested action. If the DBMS cannot find a row with a matching user ID and table or view identifier, then the user has no right at all to the database element. If a row with a matching user ID and table identifier exists, then the DBMS checks for the specific rights that the user has to the table or view and either permits or disallows the requested database access.

Database Implementations

Access rights to tables and views are stored in the data dictionary. Although the details of the data dictionary tables vary from one DBMS to another, you will usually find access rights split between two system tables named something like *systableperm* and *syscolperm*. The first table is used when access rights are granted to entire tables or views; the second is used when rights are granted to specific columns within a table or view.

A *systableperm* table has a structure similar to the following:

```
systableperm (table_id, grantee, grantor,
     slsectauth, insertauth, deleteauth, updateauth,
     updatecols, referenceauth)
```

The columns represent:
- table_id: an identifier for the table or view
- grantee: the user ID to which rights have been granted
- grantor: the user ID granting the rights
- selectauth: the grantee's SELECT rights
- insertauth: the grantee's INSERT rights
- deleteauth: the grantee's INSERT rights
- updateauth: the grantee's UPDATE rights
- updatecols: indicates whether rights have been granted to specific columns, rather than the entire table or view. When this value is Y (yes), the DBMS must also look in *syscolperm* to determine whether a user has the rights to perform a specific action against the database.

The columns that hold the access rights take one of three values: Y (yes), N (no), or G (Yes with grant option).

Granting and Revoking Access Rights

When you create an element of database structure, the user name under which you are working becomes that element's owner. The owner has the right to do anything to that element; all other users have no rights at all. This means that if tables and views are going to be accessible to other users, you must grant them access rights.

Granting Rights

To grant rights to another user, a user that either created the database element (and, therefore, has all rights to it) or has GRANT rights issues a GRANT statement:

```
GRANT type_of_rights
ON table_or_view_name TO user_ID
```

For example, if the DBA of *Antique Opticals* wants to allow the accounting manager (who has a user ID of acctg_mgr) to access an order summary view, the DBA would type

```
GRANT SELECT
ON order_summary TO acctg_mgr;
```

To allow the accounting manager to pass those rights on to others, the DBA would need to add one line to the SQL:

```
GRANT SELECT
ON order_summary TO acctg_mgr;
WITH GRANT OPTION
```

If *Antique Opticals* wants to give some student interns limited rights to some of the base tables, the GRANT might be written:

```
GRANT SELECT, UPDATE (retail_price, distributor_name)
ON item TO intern1, intern2, intern3;
```

The preceding example grants SELECT rights to the entire table, but gives UPDATE rights on only two specific columns. Notice also that you can grant multiple rights in the same command, as well as the same group of rights, to more than one user. However, a single GRANT applies to only one table or view.

In most cases, rights are granted to specific user IDs. You can, however, make database elements accessible to anyone, giving rights to the special user ID PUBLIC. For example, the following statement gives every authorized user the rights to see the order summary view:

```
GRANT SELECT
ON order_summary TO PUBLIC;
```

Revoking Rights

To remove previously granted rights, use the REVOKE statement, whose syntax is almost the opposite of GRANT:

```
REVOKE access_rights
ON table_or_view_name FROM user_ID
```

For example, if *Antique Opticals'* summer interns have finished their work for the year, the DBA might want to remove their access from the database:

```
REVOKE SELECT, UPDATE (retail_price, distributor_numb)
ON item FROM intern1, intern2, intern3;
```

If the user from which you are revoking rights has the GRANT option for those rights, then you also need to make a decision about what to do if the user has passed on those rights. In the following case, the REVOKE option will be disallowed if the acctg_mgr user has passed on his or her rights:

```
REVOKE SELECT
ON order_summary FROM acctg_mgr
```

```
RESTRICT;
```

In contrast, the syntax

```
REVOKE SELECT
ON order_summary FROM acctg_mgr
CASCADE;
```

will remove the rights from the acctg_mgr ID, along with any user IDs to which acctg_mgr granted rights.

Who Has Access to What

The internal database security measures we have discussed to this point assume that we know which users should have access to which data. For example, does a bookkeeper need access to data from the last audit or should access to those data be restricted to the comptroller? Such decisions may not be as easy as they first appear. Consider the following scenario (loosely based on a real incident).

The Human Relations department for a small private college occupies very cramped quarters. Originally, personnel files were not converted to machine-readable form, but instead were stored in a half dozen locked four-drawer file cabinets stored in the reception area, behind the receptionist's desk. The receptionist kept the keys to the file cabinets and gave them to HR employees as needed.

The problem with this arrangement was that the receptionist, who was a long-time college employee, had a habit of peeking at the personnel files. She was known to have the juiciest gossip in the building at coffee break time.

One day, the receptionist came in to work and saw the file cabinets being moved out of the reception area. In their place, a PC had been placed on her desk. The new HR system was online and an IT staff member had come to train the receptionist. She was supplied with a word processor, access to the college faculty/staff directory, and an appointment scheduling application.

"How do I get to the personnel files?" she asked at the end of the training session.

"'You don't have that application," she was told.

"Why not?"

The IT staff member shrugged. "You'll have to ask the database administrator."

The receptionist was out the door, headed for IT.

She found the database administrator in his office.

"Why didn't I get the personnel application?" she asked.

"Because you don't need access to that information to do your job and legally we have to ensure the privacy of those data."

At this point, the receptionist was furious. "But I've always had access to the personnel files!" she shouted.

The scene degenerated from there.

The problem would appear to be that the HR receptionist should never have had access to the personnel files in the first place. However, it's a bit more complicated than that. Access to information makes people feel powerful. It's the "I know something you don't know" syndrome. Remove or restrict access and any many people feel that they have lost power and status in the organization.

One solution to this problem that some organizations have adopted is to appoint a committee to handle decisions of who has access to what. Users who feel that they need additional access appeal to the committee, rather than to a single individual. This protects the staff members making the decisions and provides broader input to the decision making process.

The other side of this issue is data sharing. Occasionally, you may run into employees who have control of data that need to be shared, but the employee is reluctant to release the data. A researcher who controls survey data, a district manager who handles data about the activities of salespeople in the field… Psychologically, the issue is the same as determining data access: Access to data and controlling data can make people feel powerful and important.

Data sharing can be mandated by a supervisor (if necessary, as a condition to continued employment). However, it is often better to try to persuade the data owner that there is benefit to everyone if the data are shared. By the same token, the committee that makes decisions about data access must also be willing to listen to the data owner's arguments in favor of keeping the data restricted and be willing to agree if the arguments are compelling.

Backup and Recovery

Every discussion of database security needs to spend at least some time looking at preparation for catastrophic failures. Disk drives develop bad sectors, making it impossible to read or write data; natural disasters can fill the server room with water. Earthquakes and fires happen.

When such failures occur, there is only one thing you can do: Revert to a backup copy. In this section, we'll look at making backups and how they fit into a disaster recovery scheme.

Backup

We don't usually think of backup as part of a security strategy, but, in some circumstances, it can be the most effective way to recover from security problems. If a server becomes so compromised by malware that it

is impossible to remove (perhaps because virus protection software hasn't been developed for the specific malware), then a reasonable solution is to reformat any affected hard disks and restore the database from the most recent backup. Backups are also your only defense against physical damage to data storage, such as a failed hard disk.

Note: Some small databases are kept on solid state devices (SSDs). SSDs don't have the same failure points as magnetic disks, but they are subject to wear: A bit on an SSD can be written only so many times before it wears out. Therefore, a very volatile and/or old database stored on an SSD is vulnerable to media failure.

A backup copy is a usable fallback strategy only if you have a backup that isn't too old and you are certain that the backup copy is clean (in other words, not infected by malware).

How often should you make backup copies? The answer depends on the volatility of your data. In other words, how much do your data change? If the database is primarily for retrieval, with very little data modification, then a complete backup once a week and incremental backups every day may be sufficient. However, an active transaction database in which data are constantly being entered may need complete daily backups. It comes down to a decision of how much you can afford to lose versus the time and effort needed to make backups often.

Assuming that you have decided on a backup interval, how many backups should you keep? If you back up daily, is it enough to keep a week's worth? Or will less do or do you need more? In this case, it depends a great deal on the risk of malware. You want to keep enough backups that you can go far enough back in time to obtain a clean copy of the database (one without the malware). However, it is also true that malware may affect the server operating system without harming the database, in which case the most recent backup will be clean. Then you can fall back on the "three generations of backups" strategy, where you keep a rotation of "child," "father," and "grandfather" backup copies.

It used to be easy to choose backup media. Hard disk storage was expensive; tape storage was slow and provided only sequential retrieval, but it was cheap. We backed up to tape almost exclusively until the mid-2000s, when hard disk storage became large enough and cheap enough to be seen as a viable backup device. Some mainframe installations continue to use tape cartridges, but even large databases are quickly being migrated to disk backup media. Small systems, which once could back up to optical drives, now use disks as backup media almost exclusively.

The issue of backup has a psychological component, as well as a technical component: How can you be certain that backups are being made as scheduled? At first, this may not seem to be something to worry about, but consider the following scenario (which is a true story).

In the mid-1980s, a database application was installed for an outpatient psychiatry clinic that was affiliated with a major hospital in a major northeastern city. The application, which primarily handled patient scheduling, needed to manage more than 25,000 patient visits a year, split between about 85 clinicians. The database itself was placed on a server in a secured room.

The last patient appointment was scheduled for 5 pm. Most staff left at that time. However, the receptionist stayed until 6 pm to close up after the last patient left. Her job during that last hour included making a daily backup of the database.

About a month after the application went into day-to-day use, the database developer who installed the system received a frantic call from the office manager. There were 22 unexplained files on the receptionist's computer. The office manager was afraid that something was terribly wrong.

Something was indeed wrong, but not what anyone would have imagined. The database developer discovered that the unidentified files were temporary files left by the database application. Each time the application was launched from the server it downloaded the structure of the database and its application from the server and kept them locally until the client software was shut down. The presence of the temporary files meant that the receptionist wasn't quitting the application, but merely turning off her computer. If she wasn't quitting the database application properly, was she making backup copies?

As you can guess, she wasn't. The only backup that existed was the one that the database developer made the day the application was installed and data were migrated into the database. When asked why the backups weren't made, the receptionist admitted that it was just too much trouble.

The solution was a warning from the office manager and additional training in backup procedures. The office manager also monitored the backups more closely.

The moral to the story is that just having a backup strategy in place isn't enough. You need to make certain that the backups are actually being made.

Disaster Recovery

The term *disaster recovery* refers to the activities that must take place to bring the database back into use after it has been damaged in some way. In large organizations, database disaster recovery will likely be part of a broader organizational disaster recovery plan. However, in a small organization, it may be up to the database administrator to coordinate recovery.

A disaster recovery plan usually includes specifications for the following:

- Where backup copies will be kept so that they will remain undamaged even if the server room is damaged.
- How new hardware will be obtained.
- How the database and its applications are to be restored from the backups.
- Procedures for handling data until the restored database is available for use.
- Lists of those affected by the database failure and procedures for notifying them when the failure occurs and when the database is available again.

The location of backup copies is vitally important. If they are kept in the server room, they are likely to be destroyed during a flood, fire, or earthquake, along with the server itself. At least one backup copy should therefore be stored off-site.

Large organizations often contract with commercial data storage facilities to handle their backups. The storage facility maintains temperature-controlled rooms to extend the life of backup media. They may also operate trucks that deliver and pick up the backup media so that the off-site backup remains relatively current.

For small organizations, it's not unheard of for an IT staff member to take backups home for safe keeping. This probably isn't the best strategy, given that the backup may not be accessible when needed. Some use bank safe deposit boxes for offsite storage, although those generally are only accessible during normal business hours. If a small organization needs 24/7 access to off-site backup copies, the expense of using a commercial facility may be justified.

When a disaster occurs, an organization needs to be up and running as quickly as possible. If the server room and its machines are damaged, there must be some other way to obtain hardware. Small organizations can simply purchase new equipment, but they must have a plan for where the new hardware will be located and how network connections will be obtained.

Large organizations often contract with *hot sites*, businesses that provide hardware that can run the organization's software. Hot sites store backup copies and guarantee that they can have the organization's applications up and running in a specified amount of time.

Once a disaster recovery plan has been written, it must be tested. Organizations need to run a variety of disaster recovery drills, simulating a range of system failures. It is not unusual to discover that what appears to be a good plan on paper doesn't work in practice. The plan needs to be

range of system failures. It is not unusual to discover that what appears to be a good plan on paper doesn't work in practice. The plan needs to be modified as needed to ensure that the organization can recover its information systems should a disaster occur.

The Bottom Line: How Much Security Do You Need?

Many of the security measures described in this chapter are costly, such as adding user authentication measures beyond passwords and user IDs and contracting with a hot site. How important are high-end security measures? Should your organization invest in them?

The answer depends on a number of factors, but primarily on the type of data being protected:

- Are there laws governing the privacy of the data?
- Could the data be used as the basis for identity theft?
- Do the data represent trade secrets that could seriously compromise the organization's market position should they be disclosed?

The final question you must answer is how much risk you are willing to tolerate. Assume that it is affordable to install enough security to protect against about 80 percent of security threats. However, it may well cost as much as that entire 80 percent to achieve an additional one or two percent of protection. If you are willing (and able) to tolerate a 20 percent chance of a security breach, then the 80 percent security is enough. However, if that is too much risk, then you will need to take more security measures, regardless of cost.

For Further Reading

Bond R, See KY-K, Wong CKM, Chan Y-KH. *Understanding DB2 9 Security*. IBM Press; 2006.
Clarke J. *SQL Injection attacks and Defense*. Syngress; 2009.
Davis M. *Hacking Exposed Malware and Rootkits*. McGraw-Hill Osborne Media; 2009.
Faragallah OS, et al. *Multilevel Security for Relational Databases*. Auerbach; 2014.
Gertz M, Jajodia S. *Handbook of Database Security: Applications and Trends*. Springer; 2007.
Kenan K. *Cryptography in the Database: The Last Line of Defense*. Addison-Wesley Professional; 2005.
Litchfield D. *The Oracle Hacker's Handbook: Hacking and Defending Oracle*. Wiley; 2007.
Litchfield D, Anley C, Heasman B, Grindlay, et al. *The Database Hacker's Handbook: Defending Database Servers*. Wiley; 2005.
Natan RB. *Implementing Database Security and Auditing*. Digital Press; 2005.
Thuraisingham B. *Database and Applications Security: Integrating Information Security and Data Management*. Auerbach Publications; 2005.
Welsh TR. *A Manager's Guide to Handling Information Security Incidents*. Auerbach; 2009.
Wright J, Cache J. *Hacking Exposed Wireless, Third Edition: Wireless Security Secrets and Solutions*. McGraw-Hill Education; 2015.

[1] http://www.bankatunion.com/home/business/online/goidtokens/goidtokensfaqs

[2] http://www.safenet-inc.com/data-protection/password-protection-applications/?aldn-true