WHITE PAPER: ENTERPRISE SECURITY

Strengthening Database Security

White Paper: Enterprise Security

Strengthening Database Security

Contents

Introduction

Virtually every corporation maintains a database that contains critical business information—from customer contact or account data to order tracking information and human resource records. Today, enterprises must comply with industry and government regulations that charge businesses with ensuring the security of this sensitive information.

At the same time, databases are increasingly subject to attack by internal and external attackers who no longer simply seek notoriety but now want financial rewards. By compromising the security of databases and obtaining customers' personal data, committing fraud, or blackmailing the targeted company, both internal and external attackers can jeopardize the reputation, financial standing, and customer trust of a business.

Consequently, database security has become a serious concern for an increasing number of corporations. Security vendors are responding by designing technologies that complement and extend traditional database solutions. One such technology is database attack detection.

This white paper examines the techniques currently in use to protect database information and discusses some new technologies and approaches that can improve data security and integrity.

Traditional database security

A variety of techniques exist to combat database attacks. This section examines the more common ones and discusses their drawbacks. Table 1 shows the strengths and weaknesses of each approach.

Table 1. Effectiveness of solutions against database attacks

Solution/Threat	Anomalous (malicious) queries	Illegal/ failed logins	Access to sensitive data	Sniffing DB traffic	SQL injection attacks	Abuse of granted privileges	Network hacking of DB server	Theft of DB server
Database auditing	○	◒	○	○	○	◒	○	○
Role-based access control	◒	◒	◒	○	○	○	○	○
Database field encryption	○	○	○	○	○	○	○	●
Host IDS on database server	○	◒	○	○	○	○	◒	●
Network IDS	○	○	○	○	○	○	●	○
Network encryption	○	○	○	●	○	○	○	○

Database auditing

Most databases can generate an audit log of operations performed on the database. Auditing records malicious activities on the database, such as failed logins or attempts to abuse access privileges, but it does not highlight internal attacks, such as someone with valid permissions taking data for purposes other than the intended use. Also, it takes time to review audit logs, and malicious activity may not be obvious to the untrained eye. The main reason many companies do not keep extensive audit trails, however, is because maintaining an audit log requires significant processing power, which has a negative impact on the database server. In fact, auditing often causes a 30 percent or greater degradation in overall database performance—a hit that many organizations cannot afford to take.

Role-based access control

The traditional approach to database security is access control. With this approach, database tables are configured with access controls to ensure that only authorized users with specific roles are issuing database queries. Access controls provide a necessary level of database protection; however, they are typically looser than desirable to accommodate the needs of the client application (for example, CRM).

This discrepancy is the outcome of a disparity between the way applications and databases are engineered. An enterprise application must serve thousands or potentially millions of different users simultaneously, accessing the database rapidly to perform various actions requiring varying permissions. The application front end accepts requests (in the form of HTTP or SOAP) from users, and its business logic processes those requests, sending appropriate queries to the back-end database to retrieve or update information. Today, most enterprise applications are designed to access the database via a single master account with high-level (for example, root) access to the database. This carte blanche access to the database enables the application to process virtually any request on behalf of any user without a concern about security.

Leaving access controls looser than necessary on the database results in weaker security. If an attacker compromises the application, he or she can cause it to issue other commands to the database with the near root-level access granted to the application. SQL injection is one type of attack that exploits this weakness in the typical application-database system. The attacker enters specially crafted characters into an application field (for instance, the username field of a login Web page), which are not detected by the application and are sent directly to the database. This

allows the attacker to issue commands directly to the database through the application. Since the application has near root-level access to the database, the command likely will not violate any access control rules enforced by the database, enabling the attacker to leverage the full capabilities of the application.

In-database field encryption

In-database field encryption is the process of encrypting high-valued fields, such as credit card or national identity numbers, directly in the database. Such encryption safeguards the database in the event that an attacker steals the hard drive of the database server or the entire database server machine. However, given that the database client application must have access to these fields to serve its users, either the database or the application must decrypt them before providing this information to the user. An attacker who compromises the front-end application can rely on the application or the database to automatically decrypt these fields. Thus, traditional field-level encryption provides little protection against application-launched attacks or attacks by insiders who have sufficiently privileged logins to the database.

Host-level solutions

Host-based intrusion detection systems (HIDS) have their own limitations with regard to database protection. Most either inspect audit trails or observe system modifications; consequently, they can inspect only behaviors that are logged by applications or activities that exhibit behavior on the host for which detection signatures can be created (such as modifications to key system files or settings). Unfortunately, malicious queries to a database (for example, to obtain all password/credit card data) generate no log entries or detectable behaviors. So, as far as the database is concerned, a malicious query is just as valid as a legitimate query—it only references different data.

Network-level solutions

Traditional network-level security solutions like network intrusion prevention systems (NIPS) and firewalls are designed to detect hacking attacks and exploits. Security vendors author and distribute signatures to detect common attacks such as malformed packets, buffer overflows, attempts to download UNIX password files, and zombie control communications. These signatures are produced based on an analysis of known exploit tools as well as operating system and application vulnerabilities, enabling them to detect a wide variety of attacks across a diverse set of customer networks.

Many valid database attack scenarios, however, involve a legitimate user issuing commands that appear valid to the database. Typical database attacks do not require exploit tools or the transmission of malformed packets. All the attacker has to do is log in with a sufficiently privileged account and issue syntactically correct queries to the database. There may be nothing overtly wrong with such a malicious query other than it may attempt to access more than the usual amount of information. Consequently, such a query would not be caught by a network-based intrusion detection or prevention solution. For example, while a normal query to the database might ask for a single user record, a malicious query might ask for all user records, including additional password and credit card information. A traditional NIPS or Web application firewall has no way of distinguishing that the single-record query is valid but that the other query accessing every account in the database is malicious.

In addition, network-based encryption employed by databases poses a challenge to NIPS. In many environments, connections between a client application and the database are encrypted, blinding the transactions to any NIPS. While a traditional NIPS can easily protect against malformed TCP/IP packets, it is unable to detect malicious query packets when they are sent to the database in an encrypted form. All malicious traffic looks like valid traffic to such an intrusion prevention system.

Network encryption

When applications and clients send SQL data to and from the database, it may be encrypted (for example, via SSL) to protect its confidentiality and integrity. This technique can prevent prying eyes from sniffing the data as it passes to and from the SQL server; however, encrypted data must sooner or later be converted to plain text in the client or the application. At this point, the data is only as good as the security and integrity of the user and/or application. Weaknesses at this point, such as those that enable SQL injection attacks, could potentially go undetected.

Database attack detection

For database security to advance, loopholes must be addressed. That's where a new technology called database attack detection comes in. Database attack detection augments and complements traditional security measures such as access controls, encryption, scanners, and network and host security. It works by examining each SQL command or query that is sent to a database (typically across a network) and then verifying whether that query is malicious or not. It works in real time or near real-time, detecting malicious attacks from internal and external sources.

SQL inspection

The database attack detection system is installed so that it can inspect traffic going to and coming from the database. Each request to the database is intercepted and inspected. Inspection can occur passively by making a duplicate copy of traffic observed, enabling transparent operation and maintaining existing performance levels.

To enable it to discern the difference between malicious and nonmalicious SQL queries, the database attack detection system undergoes real-world training, learning typical enterprise database usage before being deployed in a detection mode. Similar to the techniques used by some antispam systems, the database attack detection system spends some time looking at typical, legitimate SQL queries before being placed in detection mode. From this study, patterns are extracted that define the boundaries of the SQL queries observed. This set of patterns serves as the baseline set of SQL queries that the database administrator considers valid. The set is generalized to cover legitimate variations and is configurable by the administrator.

Because the number and type of SQL commands sent to a database by an application remain largely unchanged over long periods of time, this method of detection is very effective. For example, corporate databases typically receive queries from a very limited set of applications: ERP, CRM, e-commerce systems, and so on. These applications normally have a baseline set of user-facing functionality that translates into a baseline set of SQL commands to the database. The database will not likely see many SQL commands outside of this baseline set, making the database a natural fit for anomaly detection. If the database security system sees a query that would not normally be generated by the particular front-end application (ERP, CRM, and so on), it is flagged as a suspicious event.

For example, consider what would happen if an attacker were to compromise a Web server and then issue the following malicious query to the database:

```
SELECT credit, expiration, name FROM users WHERE name LIKE '%';
```

The query is valid and would be executed by the database; however, it could compromise thousands of credit card numbers from the database. Such a query would not likely be issued by a legitimate client application. With proper training, a database attack detection system could easily identify this query as an attack, block the request, and alert the administrator (see the following section).

Detection

The pattern set determined during the learning phase enables the database attack detection system to know the difference between a regular and a malicious query—for example, between a common request for a set of records and an unusual request for all records. The system intercepts all queries and compares each to the known set of approved queries. Queries not found in the known set generate an alert.

The training capability can also be augmented by a configurable mechanism that enables security administrators to create their own detection rules. A rich-rule language enables administrators to filter not only by the SQL command but also by the name of the user, machine, and application program issuing the query and the time the query was issued. With this capability, for example, the administrator can author rules to send an alert even when an administrative user with a specific IP address range attempts to access credit card data from the database during off-hours or weekends.

This anomaly detection–based technology enables the database attack detection system to see through valid but malicious queries. Administrators can choose to block further communication or investigate the matter in more detail.

High performance/low risk

Performance is the bottom line with databases. Even effective security solutions are avoided if they degrade performance. Database attack detection uses a detection method of pattern matching that is fast, enabling the system to operate efficiently.

Moreover, the database detection logic can run on an entirely separate passive network appliance, resulting in zero performance impact on the database server. By running passively using a network sniffer to extract SQL data from the network, database attack detection eliminates any risk of performance degradation or database downtime. The connection to the database from the client is unaffected, and the data stream passes straight to the database. The sniffer makes a copy of the data, which is fed to the database attack detection system. So, the system can inspect the stream copy in near real-time, while the original stream proceeds to the database, ensuring database availability.

Network encryption

If sniffed traffic traveling between the client and the database server is encrypted, then providing a set of decryption keys to the database attack detection engine will still allow it to scan and monitor all database transactions. There are two common techniques for encrypting data: Secure Sockets Layer (SSL) and IPsec. Network encryption works especially well with SSL; the database attack detection system can run in a passive network sniff mode, simply copying SQL traffic as it passes through the network. IPsec, on the other hand, requires the database attack detection engine to sit in between the client and the database and act as a transparent relay.

Auditing

A convenient side effect of capturing SQL data as it passes from client to server is that the data is also available for purposes other than attack detection. One such purpose is auditing. In today's increasingly regulated business climate, organizations are required to keep audit logs documenting who has accessed sensitive data. The Sarbanes-Oxley Act and HIPAA, for example, involve corporate financial data and patient medical records.

Established IT environments face many challenges in retrofitting an audit process into their existing infrastructure without suffering a significant performance hit. Auditing capabilities residing on a database server can easily introduce a performance overhead degradation as high as 30 percent.

The SQL data capture capabilities of a separate database attack detection facility provide an excellent opportunity to build an audit log of SQL activity without any additional overhead on the database server. If necessary, the captured data can be made anonymous by removing sensitive values such as credit card numbers or Social Security numbers and replacing them with generic values, for example, xxxx-xxxx-xxxx-xxxx.

Conclusion

Database attack detection addresses the limitations of existing database security technologies. By profiling the application layer and looking for deviations from known behavior, it can detect threats that would evade traditional network and host intrusion detection systems. It operates with a knowledge of specific applications, enabling it to notice malicious activity even if that activity does not result in system changes or suspicious log entries. It also does not need to run on the host (although it could), resulting in a highly scalable system.

Table 2. Database attack detection coverage

Solution/Threat	Anomalous (malicious) queries	Illegal/ failed logins	Access to sensitive data	Snifing DB traffic	SQL injection attacks	Abuse of granted privileges	Network hacking of DB server	Theft of DB server
Database attack detection	●	●	●	○	●	●	●	○

Database attack detection represents a valuable supplement to corporate information security by defending databases against inappropriate and malicious requests for sensitive information. By deploying database intrusion detection in the enterprise, businesses have a powerful tool for safeguarding information assets, protecting their reputation, and maintaining customer trust.

To round out the solution, complementary measures can be employed for complete database security. Table 2 illustrates a couple of threats not directly covered by database attack detection. Encrypting data as it passes between the client and the database server addresses the issue of database traffic being sniffed. And database field encryption combined with a host-based intrusion detection system can help prevent the theft of data directly from the database server.

Together, these solutions can help to ensure that sensitive data remains in the hands of those who have authorized access.

About Symantec

Symantec is a global leader in infrastructure software, enabling businesses and consumers to have confidence in a connected world. The company helps customers protect their infrastructure, information, and interactions by delivering software and services that address risks to security, availability, compliance, and performance. Headquartered in Cupertino, Calif., Symantec has operations in 40 countries. More information is available at www.symantec.com.

For specific country offices and contact numbers, please visit our Web site. For product information in the U.S., call toll-free 1 (800) 745 6054.

Symantec Corporation
World Headquarters
20330 Stevens Creek Boulevard
Cupertino, CA 95014 USA
+1 (408) 517 8000
1 (800) 721 3934
www.symantec.com