SECURING DATABASE SERVERS

With the ever-expanding data requirements for Web applications, database administrators often configure security parameters at the OS and database layer. Unfortunately, administrators seldom consider implementing security at a network layer to protect the data from prying eyes. Recent cases of hacking into the IT infrastructure of finance firms show us that an increasing level of security awareness is required in this space, and this article aims to address this issue. It is targeted at systems and database administrators, as well as senior IT management staff, featuring case studies from a few popular open source databases.

For starters, there are seven layers that form the OSI networking model. Beginning at the physical layer, it goes up to the data-link, network, transport, session, presentation, and the application layers. While the physical-layer security is taken care of by surveillance systems, the application-layer security is handled by code instrumentation, which is the job of the developers. In the wake of overall security awareness in the IT world, responsible product managers incorporate code security as a mandated practice in their code-deployment release cycles.

All the intermediate five layers of the OSI model do require security measures too, by some means or the other, irrespective of the software

component to which they are catering. Let’s take a look at these layers, which are usually not taken too seriously by security administrators. It is important to note that these layers actually need to be more secure than the layers at the top and bottom, because there are such a variety of attacks on the interim layers that there is no single means, tool or process to control them.

The initial levels of compromising vulnerabilities usually start at the data-link and network layers (alternatively called Layer-2 and Layer-3 attacks). Layer-2 security is susceptible to MAC address spoofing, Layer-2 to flooding attacks, whereas Layer-3 security is subject to IP address-spoofing attacks. Similarly, Layers 4, 5 and 6 are prone to packet-crafting, session-stealing and cryptography-based attacks, respectively.

To understand this better, please refer to Figure 1, which shows various security methodologies and how each of these maps into the OSI layers of networking.

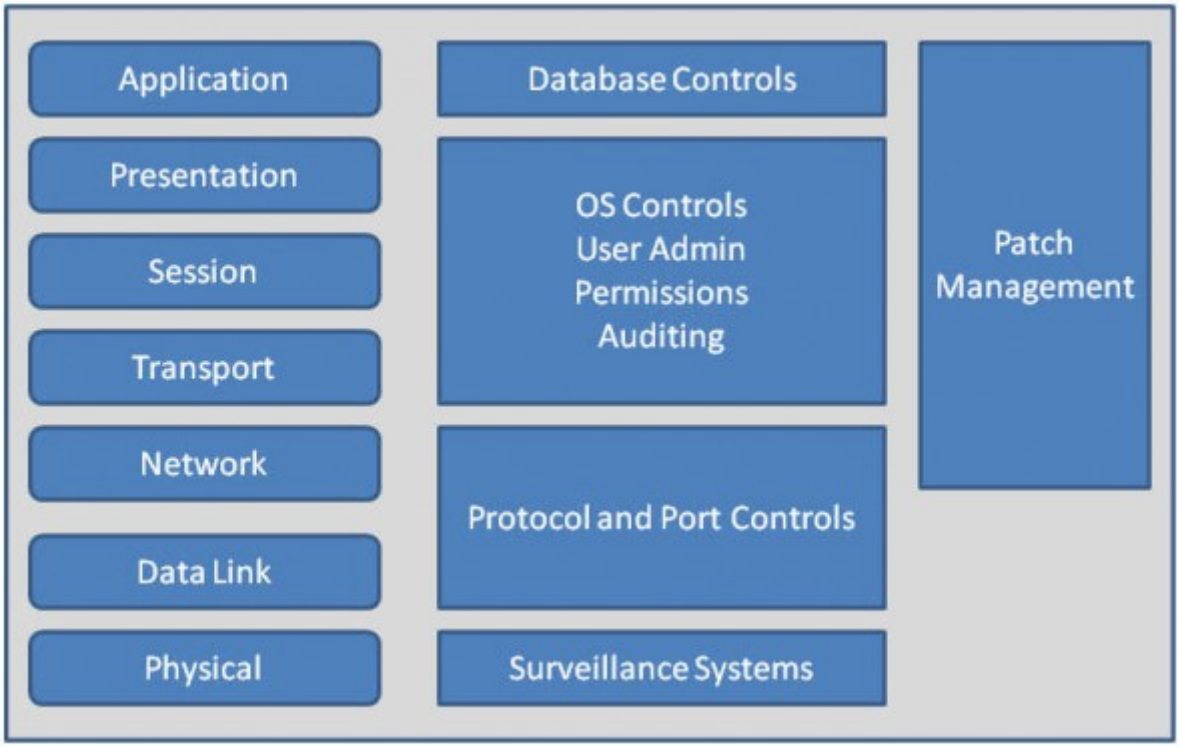

Figure 1: Security methodologies for various layers

While there is no such thing as 100 per cent security, the security design can incorporate carefully thought-out, customised security mechanisms, which

can lead to a total security solution in an IT infrastructure. There is a wide range of devices from the security view-point, such as firewalls, intrusion-detection devices, content firewalls, etc., covering each layer, and also overlapping multiple network layers.

Why is typical security not enough for database servers?

Securing database servers needs administrators to go the extra mile in terms of technical design efforts. It is a job to be carried out by sysadmins as well as database admins, to cover all the network layers. Often, the practice is to assume that using OS-level user administration and leveraging the product's built-in security is enough to protect databases.

Unfortunately, this is not true. While the user- and role-based models with strong passwords for SA accounts, etc., are essential, they exist only at the application layer; relying on them alone defeats the purpose of securing databases.

The technical reason behind why legacy methods fail is because a database is a widely and continuously accessible component, which makes it more vulnerable and susceptible to attacks. Database security requirements touch all networking layers, and hence, need careful design.

As an example, consider a database server that accepts connections from within the corporate network as well as from outside. It is accessed programmatically by the frontend software, as well as directly by admin and development staff to run queries, etc. It is accessed by backup services, anti-virus servers, and other maintenance-based utilities, and thus is widely accessible.

Any of these human or automated means of accessing the database server make it vulnerable to possible attacks originating at various networking touch-points. The latest spyware, and vulnerabilities that revolve around the infamous SQL-injection method, prove that something needs to be done beyond these legacy methods.

This is exactly where the security mechanisms and solutions that span the lower layers in the networking model come into the picture. Let us look at a few moderately advanced methods to secure database servers, followed by some serious solutions.

Securing database servers

NAT and PAT

Using network address translation and port address translation is the first recommended step. Since database products use predefined default ports, it is the first thing hackers look for, and hence should be changed. Even though the database IP address and port is not exposed to the outside world, it is a best practice to change it, to keep spyware and viruses away. Any standard firewall nowadays provides NAT and PAT features.

A demilitarised zone (DMZ)

Please refer to Figure 2, which shows a typical database server farm in a corporate IT infrastructure, hosting mission-critical databases and data-warehousing services.

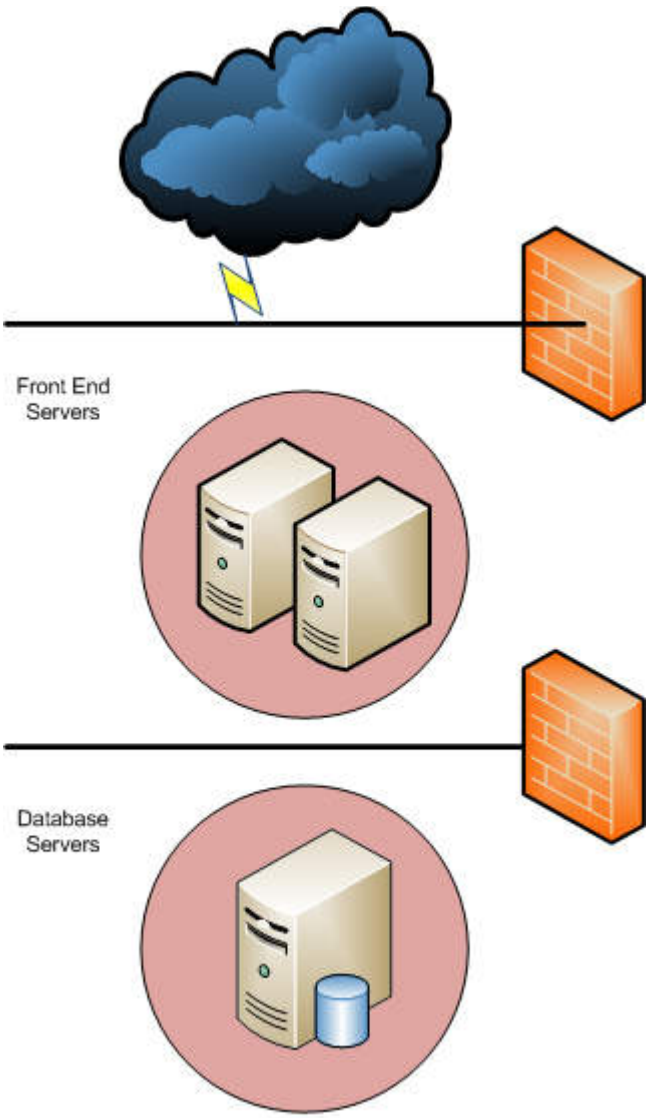

Figure 2: Database server farm

As we can see, the database server is not exposed to the Internet, but is separated from it by an additional firewall. This design method is well-known, and is called a demilitarised zone (DMZ). In earlier days, it was a practice to keep database servers in the same network as the frontend/Web application servers, making them prone to attacks. An additional layer of security is now achieved by the second firewall, which opens up only the database port for the frontend servers, and not the IP addresses of the entire Internet. If the database carries mission-critical datasets of sensitive information, this method can be clubbed together with the NAT-PAT method, making it even harder for malware to breach the database server.

Content-based firewalls

The latest firewall products are equipped with the capability to look into the data packets flowing in either direction, and provide systems administrators a means to configure actions based on the intercepted content. These firewalls, sometimes called Layer-7 firewalls, are capable of reading SQL queries, query data, validation data, XML data, etc. Deploying such a firewall can certainly reduce virus attacks drastically, because data packets pertaining to ill-behaved operations are dropped and reported.

SSL connections

While SSL technology is commonly used to secure Web-server connections to browsers, it can also be used between frontend servers and the backend database server. SSL uses public-key and private-key cryptography to encrypt all connections between caller and listener. While this solution is a bit costly and takes a toll on data-transfer speeds, it is worth the effort to alleviate man-in-the-middle attacks.

IPSec security

The real challenge while securing a database server is to figure out who should access it. In terms of people, it is as simple as deploying user-ID and password-based access. However, if the network is compromised by a hacker, this won’t help much. A further step, in such a case, would be to deploy IPSec security, whereby each server will connect to the database server using an IPSec token, making it a unique and non-crackable connection.

Securing open source databases

A database administrator always carries out the essential chores to secure a database; however, for Linux-based open source solutions, a little extra effort is required. While there is no single solution to cover all open source database distributions, there are a few good tools for each.

A popular open source database is MySQL, which is used by many commercial websites as their backend. MySQL comes with the necessary scripts to harden the server from the security perspective, such as scripts to remove test databases, lower system and database privileges, enable the built-in Linux firewall to configure default database ports and block others, etc.

MongoDB and CouchDB are two famous database servers running on Apache distributions. Both come equipped with the great feature of IP binding, whereby the database kernel services are bound to one single IP. Adding an open source content firewall or a NAT-PAT solution can help create a very cost-effective and, yet, secure database farm.

The Hypertable open source database, which is modelled after Google's BigTable DB structure, provides an extensive library of API calls that are used exclusively for adding security features. It is easy to write scripts using those API calls, to securely deploy and configure Hypertable distributions in a big server farm.

Unlike Microsoft SQL Server, Oracle and Sybase products, while designing security for open source products, it is important to incorporate a combination of network security methods, as well as the database product's application-layer security.

Summary

While much effort goes into securing a database, due diligence is required to secure the server hosting the database too. The security measures taken for the application layer as well as the network layer must go hand-in-hand to keep mission-critical data safe from hackers. A rock-solid change-control process needs to be in place to tighten the IP stack, the patch-management solutions, as well as the physical security.

It is often found that Layer-2 security is not designed adequately, or in some cases is simply absent, which makes all the application-layer security measures futile. Senior IT management and database administrators need to work with systems administrators to amicably formulate an end-to-end

security solution. This applies to all database-centric solutions, irrespective of whether they are off-the-shelf or FOSS-based.

Source : http://www.opensourceforu.com/2011/05/securing-database-servers/