database server, so external commands do not arrive directly on the internal network. Your internal database will also be vulnerable if the perimeter network is compromised. If any machine is compromised, it may be vulnerable to sniffing attacks; if the web server is compromised, an intruder will have the full power of the remote database access mechanism available.

- Put the web server on the perimeter network and the database server on the internal network, and use a custom-written protocol to connect them. You are relying on your ability to construct a secure protocol that enables the transactions you need. It just changes the protocols used to connect the web server to the database. Creating a secure client/server application is not easy, so implementing the communications yourself may simply introduce different security problem.
- There are two web servers. Put one web server on the perimeter network and the other web server and the database server on the internal network. No database traffic will have to pass through the firewall; on the other hand, custom-written protocol is necessary to connect the web server on the perimeter network and the other web server on the internal network. You are relying on your ability to construct a secure protocol. The custom-written protocol connect two web servers and internal server talks to the database server. Creating a secure client/server application is not easy.
- There is a web server and an application server. Put the web server on the perimeter network and the application server and database on the internal network. No database traffic will have to pass through the firewall; on the other hand, the web server will forward the request to the application server through firewall. The way of communication between the web server and the application server is vender specific.

All of these approaches can be successful; putting the database on the perimeter network provides the most protection for your internal network, and putting the web server on the internal network provides the least.

General concepts, functionalities

Server security is a key aspect of server management for web hosting providers and server administrators. Here, we look at ten techniques for hardening servers and monitoring them for security vulnerabilities.

1. Use Public Key Authentication For SSH

Remove unencrypted access. No one should use telnet, ftp or http to manage servers anymore. SSH, SFTP and https are the accepted standards. For even better security, get rid of password authentication on SSH altogether. Instead, use SSH keys. Each user has a public key and a private key. The private key is kept by the user. The public key is kept on the server. When the user tries to login, SSH makes sure the public key matches the private key. Once password logins are disabled, there’s no risk of a successful brute force attack against a weak password.

2. Strong Passwords

A security hardened server is a challenge for criminals, but you would be surprised how many server administrators leave the front door wide open. People—including those who should know better—tend to choose easily guessed passwords. Last year, brute force attacks against servers with weak SSH passwords resulted in a spate of ransomware attacks. Use long and random passwords—long passphrases are better and finally restrict users with login type access.

3. Install and Configure the CSF Firewall

The Config Server Firewall is a feature-rich, free firewall that can protect a server against a wide variety of attacks. Its features include stateful packet inspection, authentication failure rate limiting, flood protection, directory watching, and the use of external block lists. CSF is a fantastic tool, and is a lot easier to manage than iptables.

4. Install and Configure Fail2Ban

Every server on the web is plagued by bots looking for weaknesses. Fail2Ban trawls through your server's logs in search of patterns that indicate malicious connections, such as too many failed authentication attempts or too many connections from the same IP. It can then block connections from those IPs and notify an administrator account.

5. Install Malware Scanning Software

Ideally, you want to keep malicious individuals out of your server, but if they do manage to breach the server's security, you want to know about it as soon as possible. ClamAV is an excellent malware scanning tool for Linux, and rkhunter is useful for finding rootkits. In combination, there's a good chance they will find any malware a hacker might install on a server. AIDE can be used to generate a hashed table of files on the system and then compare the hash count of the files daily to confirm no changes have been made to system-critical files.

6. Keep Software Up-To-Date

Out-of-date software is likely to contain security vulnerabilities that are known to hackers, as Equifax recently discovered to everyone's cost. If you ignore all the other advice in this article — which you should not—you should at the very least update using your Linux distribution's package manager.

7. Backup Regularly

You may not think of backups as a security measure, but the main reason we secure a server is to keep the data stored on it safe. It's impossible to guarantee that a server will never be compromised, so data should be encrypted and backed-up to an offsite location. Regular testing of recovery from comprehensive backups will neuter ransomware attacks.

8. Monitor Logs

Logs are a vitally important security tool. A server collects enormous amounts of information about what it does and who connects to it. Patterns in that data often reveal malicious behavior or security compromises. Logwatch is an excellent daily summary tool that can analyze, summarize, and generate reports about what's happening on your server. Logsentry can be used for hourly reports for more active monitoring of ingress.

9. Turn Off Unnecessary Services

Any internet-facing software that isn't essential to the server's function should be disabled. The fewer points of contact between the server's internal environment and the outside world, the better. Most Linux distributions—including CentOS and Ubuntu—include a tool for managing services.

This also applies to the web server engine itself, turn of modules you don't need, remove language modules not in use, disable web server status, and debugging pages. The less information you provide about your underlying infrastructure the smaller the footprint becomes to attack you with.

10. Install ModSecurity

ModSecurity is a Web Application Firewall—it operates at a higher level than the CSF firewall and is designed to deal with threats against the application layer. In a nutshell, it stops many types of attack against web applications, including content management systems like WordPress and eCommerce stores like Magento. ModSecurity used to be an Apache module, but it is now available for NGINX too.

Consider these options also:

Take steps to mitigate XSS attacks (Cross Site Scripting) by adding the settings to the servers that force the server and client to confirm who they are talking to. OWASP has a wealth of information and tutorials.

Use HTTP2 (or http1.2) Implementations of HTTP/2 MUST use TLS version 1.2 or higher for HTTP/2 over TLS. This upgraded engine for http benefits server load by talking binary to the client rather than text and hat improve interoperability, it reduces exposure to known security vulnerabilities and reduce the potential for site display issues from different client browser access.