



Backup design approach

Production server file backup

1. FTP into production server.
2. Compute the hash for each files against the known hash from the database on the remote server
3. If the hash is different, download the file to the vm server and perform virus scanning. If all is good create a today date directory and store the encrypted file along with its parent folder inside that date and create a new entry for the new hash, file name and timestamp in the remote server database.

If virus... might even be false positive, send an email notification to the designated email for investigating manually.

compression? this will be optional

Production Database server backup

1. Connect via ssh or phpmyadmin (will implement ssh for now)
2. Download the database(s) create hashes compare the hash of that on the remote server, if hash different download the .sql to the remote backup server store in directory of the date it was back up. store the hash, the encrypted db sql, file name and timestamp

Important design constraints:

Script(s) for back up will resident on the remote backup server(s)..... place the script(s) on multiple different remote backup servers give you multiple backup options

Production Database / Production servers should never contain credentials for the remote backup server.

Production Database / Production Servers should never be able to initialize with the remote server. All communication should only be possible from the remote backup server.

This is done to ensure that either production database or production server were compromised or hacked, the attacker dont have any access to the remote backup server

Failure to adherence to those constraints will result in a compromised remote back up server if the production database and production servers were to be compromised.

Files must be encrypted prior to backup on the remoted server