Exercise 11 – Hardening

Stewart Johnston
CIS 150 – Intro to Database Administration
NCMC
johnstons1@student.ncmich.edu

1. Install only the required SQL server components
   (a) The effect of only installing required components is in part to reduce the possible attack surface of the Server. Adding other components later is in effect trivial, and offers no real benefit if installed early. Although the Server may not load those components until they are needed, I can't say this for certain. By waiting to install additional modules until they're needed, the runtime hardware demands are kept as light as possible.
   (b) I think this is best practice with any technology, from SQL to Operating System. Stripping out unnecessary modules from the Linux kernel and compiling it customized for the target hardware is not uncommon.
2. Don't install SQL Server Reporting Services (SSRS) on the same server as the database engine
   (a) Server Reporting Services run with internet facing pieces on top of Microsoft IIS, which has historically not been a bulwark of security. Running it on the same server as the DBMS adds enormous surface area potentially vulnerable.
   (b) For the same reasons as the first section, this seems to be a best practice which would be wise to apply in all circumstances. Even if the server it runs on is not on different hardware, merely a different virtual machine, that insulation of internetworked components is wise.
3. Disable services not expecting immediate use
   (a) This may come with some management overhead, but it is probable that much of it could be scripted away. This is another property which falls out of the first. Keeping both runtime hardware requirements and attack surface slim is not bad as long as it doesn't interfere with performance.

(b) Again, this seems to me like a best practice with few exceptions.

4. Change the default TCP/IP ports

   (a) This may be helpful insofar as preventing vulnerability to malware which is not particularly smart, likewise malicious people who are not particularly smart. That may well be enough to keep threats from organization machines from being effective.

   (b) Of the software means to protect a SQL server, this is likely one of the weakest. It isn't necessarily a bad thing to do, but it remains effective only if the server isn't vulnerable to port sniffing. For a low-risk server, changing the defaults may not be worth the hassle. The fewer users there are on a network accessing a server from organization machines, the less risk there is of malware threatening the server.

5. Disable the network protocols not required for operation

   (a) Another means of keeping the surface area slim, insulating the SQL server by running it without any other networking stack on the same host is wise.

   (b) This depends on the availability and difficulty in achieving this. If favoring named pipes over TCP/IP means changing significant chunks of application logic to connect properly, then it's certainly not worth the effort for such databases as we've been running.

6. Keeping anti-malware and firewall software updated and configured correctly

   (a) On its own, keeping firewall and anti-malware software running smoothly and correctly on any system will be tremendously effective in stopping a significant chunk of threats in their tracks. Firewall software which is too permissive, or blocks the incorrect ports, could prevent organization operations. Anti-malware software which is not updated can't catch any recently discovered threats.

   (b) Especially for a small business unlikely to face targeted or tailored threats, the benefit gained and the relative ease at which it is attained moves this into best practice territory, with very few exceptions.

7. Manage the surface area configurations

   (a) Microsoft in particular often does not have what I perceive as sane defaults. Using a tool to manage defaults before a server is even deployed, and to wrangle the exact policy and configuration after deployment is only smart. Being able to touch these configurations and following through on that ability is, to my mind, a solid chunk of an administrator's role.

   (b) Many defaults may not require wrangling, especially for smaller-scale organizations, those with low risk, or those without sensitive infor-

mation. Managing the surface area configurations will still inevitably come up. Even if only one policy sees change, doing so from one unified tool and tracking the changes is too easy an option to ignore.

8. Configuring authentication

   (a) The recommended course of action is to use Windows Authentication only, whenever possible. In a homogenous environment, this helps reduce moving parts and points of ingress. MS SQL Server makes it trivial to introduce new logins based on Windows Authentication, which is convenient. If the DBA is not wearing an excessive number of hats, such as also being the SysAdmin for the network, this can help offload the work of security. This comes with the trade of having to trust the SysAdmin to have strong security policy. The security is only as strong as the weakest link, after all.

   (b) For a small business, it is likely that the DBA is also the SysAdmin, and moving the security responsibility from the DBMS to the OS does not also move it outside of the DBA's influence. There isn't a compelling logistical reason not to use Windows Authentication for all Windows users on the network. The only qualm I have with it is that Windows Authentication is not particularly difficult to break, especially with access to the box. Still, it is a practical choice.

9. Configure administrative accounts

   (a) Removing the builtin administrator accounts and heavily restricting roles and permissions will go a long way to preventing privilege escalation and unauthorized tampering.

   (b) Again, the security is only as strong as the weakest link. Securing the server on which the DBMS runs is as important, if not more, than securing the DBMS itself. This should be fairly trivial to put in place, and seems to me an obvious best practice.

10. Provision service accounts using non-user accounts which are not built-in

    (a) User accounts have permissions on many machines, which makes them a more vulnerable vector for attack, so non-user accounts are best. Built-in accounts inherit some permissions from the defaults. Using custom non-user accounts is wise in combination with the principle of least privilege, and the separation of duties. Splitting up administration over multiple components of a hardware network is recommended for similar reasons.

    (b) This is mildly finicky, but for a small business, the small number of moving parts makes this kind of hardening practice trivial to implement. For increasingly sensitive data, this is increasingly necessary. Except for the forethought in configuring it, I can see no reason not to implement this.

11. Creating security groups

    (a) Applying roles and permissions to domain security groups makes administration of larger networks significantly easier. Rather than handling an increasing number of individual members at the DBMS level, handling the abstract security groups makes management significantly less granular at the DBMS.

    (b) For fewer than a dozen users, this is probably excessive extra work to implement this. Such small organizations as would be using the MyGuitarShop database would require so little overhead to just handle individual users, that adding the overhead of this abstraction doesn't make sense. Only if a small business were expecting significant growth would this abstraction be worth setting up at this scale.

12. Schema and object ownership

    (a) Granting ownership in wanton fashion is unwise. The dbo schema is so wide-reaching that no individual user should be mapped to own it, and when they do need to own something, allocating schemas is fairly trivial. Keeping those schemas on a per-database level, rather than allowing them to chain, will help prevent unauthorized (if unintentional) access to data.

    (b) Working with schemas is trivial to implement on the DBMS level, and there is no compelling reason not to adhere to this advice. This is a best practice with few exceptions, as far as I can tell.