Oracle Forensics
(David Litchfield)
Chapter 4
Preventing Break-Ins

As the old adage goes prevention is much better than the cure. It would be nice to live in a world where forensics weren't necessary but we have to face reality: companies and organizations suffer break-ins all too often and forensic investigations are necessary. Looking at recent data security breaches it quickly becomes apparent that far too many could have easily been prevented with just a little bit of forethought and if a little more effort had been put in. Changing default passwords would stop a great number of break-ins; regularly checking for SQL injection vulnerabilities in web applications would solve just as many. These things are not hard to do, not by any stretch of the imagination. Not only is it easy to do but it's the right thing to do – protecting your customers' interests is the same as protecting your own. With this in mind we'll look at some guidelines that will go a long way to preventing break-ins. This chapter, being only a chapter, is not intended to cover the full ins and outs of securing an Oracle environment. It will cover, however, the "9 steps for the 99 percent solution" – in other words following these 9 steps will solve 99 percent of the problem. For an excellent guide to securing Oracle may I suggest "Oracle Security Step-by-Step" by Pete Finnigan?

The 9 steps for the 99% solution are as follows

Protect Database Servers with a firewall
Adopt Good Account Keeping Practices
Reduce Attack Surface
Use the principle of least privilege
Assess vulnerability and keep up to date with patches
Develop and enforce coding standards
Use Encryption
Enable Auditing
Deploy Network based Intrusion Detection/Prevention Systems

Protect Database Servers with a firewall

Whilst, this sounds like a no-brainer (as indeed do many of these steps), according to the "Database Exposure Survey 2005" there are an estimated 140,000 Oracle database servers on the internet, not protected by a firewall. This is an incredibly large number – too large. All database servers, production or otherwise, but especially production should be behind a firewall that has been configured to prevent anyone from accessing it from the "outside". Indeed, the network should be designed so that it's possible to use the firewall to limit access to the database server to only those systems that need access as a strict business requirement. Applying defence in depth strategies a DBA can also use TCP valid node checking on the database server itself. TCP valid node checking prevents access to the database server by IP address and or hostname and can be used in white list or black list mode. Employing TCP valid node checking also helps reduce risk of exposure in the case of a mis-configured firewall. For example, let's say the firewall block access to ports 0 to 1023 and 1521 on the database server. If the database server is running the XML Database (XDB) then it will be listening on TCP ports 2100 and 8080 as well and the attacker will still be able to access these. They won't, however, if TCP valid node checking is used.

Adopt Good Account Keeping Practices
Oracle is well known for installing a large number of default user accounts with default passwords. There are lists posted all over the internet detailing what these accounts are and what their passwords are and any cracker will be able to rattle a list of at least twenty off the top of their head. Change Default Passwords! I find it incredible that this advice is rarely listened to – probably 6 to 7 out of 10 Oracle servers I come across during my assessment work have one or more default accounts with default passwords – usually it's the DBSNMP account followed closely by CTXSYS, MDSYS and WKSYS in terms of numbers. The situation has gotten better with 10g – 10g now installs with most of the accounts locked and expired and, for those that aren't (i.e. the SYS, SYSTEM, SYSMAN and DBSNMP accounts), the installer needs to set a password for them. DBAs should require strong passwords and require that the passwords be changed on a regular basis. I'd recommend once a month for normal user accounts but if that sounds too draconian I'd set a maximum of three months before mandating a password change. To stop users from re-using the same password it's possible to a keep a history. Password policies can be enforced using profiles.

Reduce Attack Surface
The more functionality there is that is exposed to an attacker means there is a greater chance of there being a bug that they can take advantage of to gain unauthorized access to the system. Accordingly it follows that, if you don't use some component, take it out. For example if you don't use the XML Database turn it off; if you don't actively use add-ons like InterMedia, Context or Spatial then drop the schema and all the objects from the database if they've been installed. If you don't use external procedures, then disable them. Not only does this help improve security robustness by reducing attack surface but it also considerably improves performance: less CPU cycles are spent processing instructions from unnecessary code. For those components that are used then only those accounts that need access to them to allow business applications to function correctly should be given access. This leads us to step three – use the Principle of Least Privilege.

Use the principle of least privilege
Simply, this means only give a user account those privileges required to do their job, and no more. It may take a little bit of effort to work out what those privileges might be to get an application to work correctly but it is definitely "worth it". Far better this than simply granting a user DBA privileges "just to get the damn application working" and opening up a gaping security hole. Using the principle of least privilege applies to role assignment and the granting of system and object privileges. The more privileges a user has the more likely it is that they will be able to abuse this to gain DBA privileges. The Oracle Hacker's Handbook has a chapter called "Indirect Privilege" escalation that details how granting users certain system privileges are tantamount to granting DBA privileges. For example, the CREATE ANY TRIGGER system privilege can be trivially abused by an attacker to gain DBA privileges – indeed this is pretty much true of all of the "ANY" privileges but sticking with the

CREATE ANT TRIGGER privilege, the attacker uses this privilege to create a trigger in the SYSTEM schema on the OL$ table after an INSERT. As PUBLIC can insert into this table the attacker can get the trigger to fire. The trigger will execute with SYSTEM privileges and so, if they attacker places SQL in the trigger that executes a "GRANT DBA TO PUBLIC", for instance, then it will execute successfully – and the attacker can gain DBA privileges. For certain users, having just the "CREATE PROCEDURE" privilege is enough for them to elevate to DBA.

A DBA should also attempt limit privileges by time. For example, if a user has a strict business requirement to create a function in the database server (and no-one else can do it for them for whatever reason) then they should only be granted the "CREATE PROCEDURE" privilege temporarily. The DBA, after management approval, can grant the privilege and inform the user. Once the user has created the function they should then tell the DBA and the privilege can be revoked.

Assess vulnerability and keep up to date with patches
New bugs are found an announced by the security community every day and every so often some of these affect Oracle products. Indeed, every three months Oracle releases a Critical Patch Update to help solve most of the Oracle related issues. It is imperative for every organization to perform regular vulnerability assessments of their production and development environments to using an up to date database scanner. There are plenty of good commercial database vulnerability assessment scanners on the market such as NGSSQuirreL for Oracle or Appdetective and the results of an assessment using tools such as these will demonstrate the servers' current exposure to risk and steps can then be taken to address the risk or assume it. As far as Oracle supplied patches are concerned or any security-driven changes to the configuration, these should be thoroughly tested on development servers or a mirror of the production environment before being rolled out to the actual product systems. Not keeping up to date with patches will affect PCI status.

With regards to performing vulnerability assessments, do them regularly. As well as assessing the database server, be sure to assess any applications that feed into the database server, especially web based applications. Tools such as NGSSoftware's Typhon III or Spidynamic's WebInspect are good choices. Even in the most secure database a simple SQL injection flaw in the web application can lead to a total compromise of data. This leads to the next principle.

Use Encryption
For sensitive data, such as customer information and PII, you should always require the use of strong encryption. The PCI Data Security Standard mandates this and the there's not a lot more than can be said about this. It's pretty much common sense.

Enable Auditing
Enabling auditing strengths an organization's security posture by giving a security administrator the ability to search for policy violations. Further in the event of a breach the audit trail should help track down the perpetrator. Auditing everything, whilst extremely valuable when it comes to analyzing the aftermath of a server compromise can quickly fill up disk space and this may not be possible for some organizations. At a minimum, both successful and unsuccessful logons should be audited as well as access to key assets – i.e. those assets that the organization has

deemed critical enough or sensitive enough that it would “cost too much” if they were stolen or deleted. With regard to financial data, Sarbanes-Oxley mandates that access to such data be audited.

Deploy Network based Intrusion Detection/Prevention Systems
I’m not convinced that database-specific intrusion detection/prevention systems are currently fully capable of detecting a professional attacker, let alone prevent them but they do at least provide some protection. The problem is that SQL is programmable and as such there is an infinite number of ways an attacker can encode and launch an attack. They could, for example, load half the exploit into a table as data on Monday and load the rest on Tuesday; on Wednesday they update the data by xoring it with 0x55 and then select the data, base64 decode it and pass it to a function which is vulnerable to a 0-day buffer overflow. The real value of IDS/IPS is their ability to record what has happened (i.e. what packets have been sent) on the network. In the event of an incident this will prove enormously useful in determining exactly what happened. Beyond this, despite the gloom about not catching the professional attacker, IDS/IPS will catch the vast majority of attempted attacks so this is makes them worth deploying; just bear in mind that they aren’t the be all and end all. Lastly, remember that your network or host based IDS/IPS can also be a target, too, and sort of goes against the principle of reducing attack surface. When looking for a solution ensure that it’s been put through its paces by serious security researchers. Don’t listen to sales speak here – go out and google for it.

Develop and enforce coding standards
This is the last of the nine principles and one of the most important. These days it is not uncommon to find a network that is fully patched but can still be broken into by weak custom code – for example SQL injection flaws in web applications. If your organization develops any custom code for your applications you need to ensure your developers follow secure programming guidelines. If you don’t have any write some or download a set from the Internet and edit it to suit your needs. If custom code development is outsourced to a third party then, as part of the contract, you should require that they have and use secure programming guidelines; you should ask to see them and don’t be frightened to have your technical staff quiz their technical staff. Whether code is developed internally or externally it should be thoroughly reviewed for weaknesses before it is placed on production systems.

Wrapping Up
Some of the measures introduced here are easy to implement such as changing default user IDs and passwords. Other require careful consideration. One area not discussed has been the development of a security plan and procedures. Failing to plan is planning to fail. In the event of an incident what would you do? Knowing what to do will prevent time being wasted, streamline the recovery and “knowing what to do” comes from having a clear, documented plan. This plan should detail who is responsible for what systems, who makes up the incident response team, what exactly to do in the event of a breach. Each company and organization is different and each will have different needs. Where and how a company operates will affect this security plan – data security breach notification laws may determine a large part of what is done and legal advice may be needed.