How the SANS Critical Controls Prevent the Red Team from P0wning your Database

GIAC (GSLC) Gold Certification ISM 5100

Author: N. Dean Sapp, deansapp@hotmail.com
Advisor: Dr. Kees Leune

Accepted: August 31st 2014

Abstract

The SANS Critical Security Controls contain proven, battle tested security investments that reduce the risk to businesses from cyber breach. This paper describes a real-world scenario in which Red Team penetration testers compromised a highly confidential database without detection and then, after the business applied three carefully selected critical controls, the Red Team was prevented from compromising the same database again.

Many security professionals will argue that a layered defensive approach is the only way to prevent breach; while true in many cases, businesses need to prioritize and focus attention on the security controls that are most effective in their environment.

The goal of this paper is to identify the security controls that effectively stop many real world attacks against databases and can be used to help businesses balance security project funding with an acceptable level of residual risk, ongoing.

1. Introduction

Databases* are pervasive in the technologically savvy world we live in. If electronic information is currency, then the database is the equivalent of the Federal Reserve Bank for many companies (Litchfield, 2005). These databases often contain financial records, account numbers, social security numbers/national IDs, electronic personal health information (ePHI), and other valuable personally identifiable information (PII) or corporate confidential data.

Hackers often target databases and then use the data harvested from them to perform identity theft, obtain credit card accounts and commit financial fraud. In mature organizations, such as those adhering to the SANS 20 Critical Security Controls, databases are often protected with layers of security to prevent data loss or breach. In less mature organizations, database security is often an afterthought (SANS.org, 2014).

This paper will describe a real-world scenario in which penetration testers compromised a highly confidential financial database in an immature organization without detection. After the business applied three carefully selected SANS critical controls, the penetration testers were prevented from compromising the same database again.

*Note to the reader, throughout this paper the terms: Red Team, penetration test, and exercise may be used interchangeably. In addition, some common terms such as database or database engineer will be abbreviated and designated with parentheses such as database (DB). These abbreviations are defined in Appendix B for future reference.

N. Dean Sapp, deansapp@hotmail.com

2. The Penetration test background

As with all penetration testing, it is important to get written permission from the system owner and relevant service providers related to the scope of the testing before beginning.

For the scope of this penetration test, The ACME Corporation (hereafter ACME) allowed the penetration testers (Red Team) to act as if they were rogue employees who had been granted access to the company intranet, but did not have access to the highly confidential financial system (the target database). The Red team was not permitted to use social engineering techniques, but was given the IP address of the target system and the vendor's name (Oracle).

This was the Red Team's first assignment to attack an Oracle database. Previously, they had focused primarily on web application attacks and defenses. For the Red Team exercise, very little additional information was provided about the target.

Due to the sensitive nature of the financial records contained in the database, the Red Team was also requested to stop all testing if a sample of database records within the database could be retrieved.

The Red Team began the exercise by performing research on Oracle databases and common vulnerabilities known to exist in default/typical installations. Several websites, including work from security researchers such as Alexander Kornbrust, David Litchfield, Pete Finnigan and Chris Gates were used. These internet searches provided older but useful information on documented Oracle database vulnerabilities and methods to exploit them. One such search returned Chris Gates's Blackhat presentation from 2009 where he introduced his Oracle "Mixin" modules (Gates, 2009). Alexander Kornbrust also published a similar attack methodology on his blog and included references to Pete Finnigan's Oracle default password list as well (Kornbrust, 2009).

Armed with this research, but uncertain of the value this aged data would represent, the Red Team began the Reconnaissance process.

2.1. Reconnaissance

For Reconnaissance and other phases of the exercise, the Red Team used the Metasploit 3.3 framework and several add-on (auxiliary) modules to the standard msf/core. These were used during the Reconnaissance, Attack (Scanning), and Exploitation phases of the penetration test. Within Metasploit 3.3 (including the manually installed Oracle Instantclient version 10, Ruby-dbi and Ruby Oci8) were several different applications and tools that were used to carry out the exercise (Gates, 2009).

2.2. Reconnaissance Tools

These included but were not limited to the list below, in addition to the command(s) used for the task:

- Nmap was used to verify the DB ports, service and version used on the database server (assuming it was default Oracle…which it was.) By Scanning for a specific IP address and TCP port, the Red Team was able to minimize the traffic sent to the database to avoid detection.

  ```
  o Nmap –p T:1521 –sV (Target IP) –oG
    tcp_oracle_scan_results.txt
  ```

  (See Appendix C for further details)

- Metasploit 3.3 was used to run the, "Metasploit Mixin" modules to confirm the TNS listener was accessible and the expected version. The TCP Port 1521 and the IP address, RPORT and RHOST respectively, were the data elements essential for the reconnaissance.

  ```
  o Msf auxiliary (tnslsnr_version) > run
  ```

  (See Appendix C for further details)

Another tool that was used to test direct access to the database during the Reconnaissance phase of the test was SQL*Plus. SQL*Plus is a simple Oracle database command-line tool use by administrators, programmers, and Red Team members alike to test basic database functionality such as authentication and scripting (Oracle, 2006).

- SQL*Plus provided access to the database directly after an account was found with working credentials.

  -
    ```
    C:\Program Files>sqlplus
      username/password@10.10.10.47:1521/SID
    ```

Due to the older version of the database and the missing patches, this research and Reconnaissance work ultimately provided the technique used by the Red Team to breach the database.

2.3. Reconnaissance Techniques

From the research and Reconnaissance efforts, it became apparent to the Red Team that four important pieces of information were needed to attempt to access data in the database (Gates, 2009). These included the:

- Database Internet Protocol (IP) address 10.10.10.47
- The port number the TNS listener was using (default TCP 1521)
- The database Service Identifier/Name or SID – to be determined (TBD)
- The Username (TBD)
- The Username Password (TBD)

Prior to launching these attack tools, The Red Team was not aware of the version, patch levels, SID, default TNS Listener port, default accounts or any of the other information needed to access the financial database. With the IP address of the database provided during the scoping process of the penetration test and the TNS Listener port determined through a basic Nmap scan, the Red Team was only missing the last three data elements to launch an attack.

2.4. Attack Tools

During the Attack phase, Metasploit was used almost exclusively to pinpoint the database details and to launch the auxiliary Mixin modules. However, the Checkpwd 1.23 tool from red-database-security.com was later used to brute force/guess the DES password hashes for several Oracle accounts including SYSADM.

Oracle hashes, both DES and SHA-1, are stored in the data dictionary tables (read only tables used by the database to operate) and were retrieved with the monfog account. The tables include the PASSWORD column of SYS.USER$ accessible through the DBA_USER view of the data dictionary (Stuber, 2009).

- Metasploit 3.3 was used to run the, "Metasploit Mixin" modules to confirm the TNS listener was accessible and the expected version as well as other attacks. These included:
  - Msf auxiliary (tnslsnr_version)
  - Msf auxiliary (sid_enum) or (sid_brute)
  - Msf auxiliary (login_brute)
  - Msf auxiliary (lt_findricset)

  (See Appendix C for further details)
- Checkpwd 1.23 was used once the salted DES hashes were downloaded and guessed/cracked offline against a known password list in addition to a custom Red Team password list.

  -
    ```
    C:\>checkpwd
    system/monfog@//10.10.10.47:1521/ACME
    password_list.txt
    ```

  (A handful of accounts were guessed…one with DBA privileges)

Because of the default database settings and the deliberate and somewhat lucky actions of the Red Team, none of the Red Team activities were identified in the ACME monitoring or alerting systems. This would need to change for round two.

2.5. Attack Techniques/Methodology

Although some steps were conducted very quickly, due to ease of the attack, the Red Team progressed from Reconnaissance to Exploitation following these steps: (1) verifying the target IP and port, (2) determining the Oracle version, (3) determining the Oracle SID, (4) enumerating the user accounts on the database, (5) locating an account with privileges to the data dictionary, (6) downloading all of the hashes from the data dictionary, (7) cracking the hashes offline to obtain a DBA equivalent privileged account and ultimately, (8) data exfiltration of a few dozen highly sensitive financial records for evidence collection as screen shots (Gates, 2009).

In a typical penetration test, a real attacker would also likely (9) establish a backdoor with shell access for further attacks or (10) hide the evidence of the attack by altering logs or other common indicators of breach.

As discovered during the Attack phase, ACME maintained an Oracle 11g database. With some internal knowledge, the Red Team surmised it had been installed and configured by the ACME central database team and then handed over to the application specific DBE for ongoing management. It appeared the DBE used the standard database image which included the default logging and monitoring accounts for the corporation. (Fortunately for the Red Team, it also included a legacy monitoring account as well).

2.5.1. Determine the Oracle Version

Once the Red Team determined the IP address they were provided was, in fact, an Oracle database, they launched the Metasploit Mixin (tnslsnr_version) to verify the version and port. It was determined to be a highly vulnerable version of Oracle 11g (11.1.0.6.)

2.5.2. Determine the Oracle SID

The next step was to determine the Oracle Service Identifier (SID). The version of the TNS listener was vulnerable to attack so the Metasploit Mixin module (sid_enum) was used to find the SID for later use in the attack.

(For details on this attack see Appendix C)

2.5.3. Enumeration of User Accounts

During the enumeration process, a detailed list of common Oracle user names and default passwords were downloaded from Pete Finnigan's website and copied into Metasploit (Finnigan, 2014). However, none of the default account user name and password combinations were successfully able to access the database. Many of the default accounts were on the database, but the passwords had all been changed, or the accounts were administratively disabled.

However, during the enumeration process, a few other non-default Oracle account were identified on the database. One of these accounts (monfog) caught the Red team's attention.

After a few minutes of research, monfog was determined to likely be an older monitoring account for the database team's use. A short password list was created based on common passwords information from Reconnaissance on the ACME intranet site. The Red Team started running the Metasploit Mixin brute force password module (login_brute) and was successful logging in on the first try. The password was the same as the account name.

2.5.4. Download the Hashes from the Data Dictionary

Once the Red Team was able to access the monfog account, they started checking to determine the privileges the account had. Fortunately for the Red Team, the account had access to the data dictionary. Once the data dictionary was accessed, it was clear that data dictionary was storing both DES and SHA1 forms of the password hashes.

2.5.5. Crack the Hashes Offline

The Red Team then downloaded both sets of the password hashes locally to a secure partition of one of the penetration testing computers and configured Checkpwd 1.23 to brute force/guess the passwords hashes. For the first round of brute force attempts, the Red Team chose the DES hashes (Korbrust, 2009). They included the common password lists from Pete Finnigan's site as well as a previously successful list of common passwords used at ACME. The Red team had planned on running the password hashes cracking modules on the more secure, SHA1 hashes if the first batch was unsuccessful, but that ended up not being necessary. They secured the penetration testing computer and launched the brute force cracking module on the DES hashes and let it run overnight. Upon returning in the morning, the Red team discovered several username and password accounts were guessed correctly.

2.5.6. Login as SYSADM

As the Red Team reviewed the privileges for the accounts on the database that were guessed, one of the accounts was found to be the SYSADM user. This account had also been granted the SYSDBA role, apparently for convenience when managing the database. With this SYSADM account, the Red Team started to review the table space and structure of the database. It was not long before they found an unencrypted column of employee IDs, first and last names, and annual salary information. This finding was documented and brought the Red Team exercise to a halt.

2.6. Red Team Wins

In total, from the time the tools were set up and the troubleshooting completed, it took approximately six hours of hands on labor to execute the different phases of the penetration test and to log into the database with a DBA user (excluding the time overnight when the offline password guessing occurred).

As a result of the Red Team exercise, a report was prepared including a brief slide deck as well as a presentation of the findings for ACME management.

The Risk Manager for ACME began researching mitigating strategies that would protect the company from a similar insider attack in the future. The Information Security & Risk Manager began researching the SANS 20 Critical Security Controls and identified three controls that appeared to provide the best value to the organization for the investment needed to protect this highly confidential financial information (SANS.org, 2014).

These recommendations were budgeted, prioritized and implemented by the DB team in the subsequent quarter to prepare for round two of the Red Team exercises.

3. SANS Critical Controls for Round Two

3.1. Playing Defense

The Red Team exercise demonstrated that additional defenses were necessary for ACME to prevent breach from an internal employee, vendor or third-party determined to cause harm. ACME consulted with their Risk Manager to review the SANS 20 Critical

Security Controls and to identify the controls that would provide the most value for the desired levels of protection (SANS.org, 2014).

The Risk Manager researched the current security controls, monitoring efforts, and defensive posture to recommend three controls for ACME to focus on. These controls included:

1. Control 11 – Limitation and Control of Network Ports, Protocols and Services
2. Control 3 – Secure Configuration for Hardware & Software on Laptops, Workstations and Servers
3. Control 16 – Account Monitoring and Control

3.2. Limitation and Control of Network Ports, Protocols and Services. (Network Segmentation)

SANS Critical control 11 pertains to the, "Limitation and Control of Network Ports, Protocols and Services." During the pen test, the Red Team was able to access the confidential database listener easily since the database resided on the same logical network as the end-user computers. The Red Team was able to TNSPing the Oracle listener without having to find and crack VPN credentials or other more sophisticated pivoting techniques to attempt authentication requests directly on the database. In addition, the default Oracle database port, 1521 was used, so the Nmap scan quickly identified and fingerprinted the database accurately.

3.2.1. Control of Network Ports and Protocols

ACME management decided an inexpensive configuration change they could make on the database was to alter the default port from 1521 to a non-standard port to reduce the ease with which the Red Team or unauthorized users could find the Oracle listener during future Reconnaissance attempts.

ACME agreed a more expensive but important control needed was network segmentation. The Risk Manager argued that segmentation would limit the exposure to the database listener from unauthorized attack while also helping to reduce the scope of the security monitoring needed from the end-user network. A new network segment was created with VPN access required to connect end-users to the segment. This access was provisioned for the DB team and a limited number of other users with a legitimate business need.

They also authorized the security team to use an existing Network Intrusion Detection System (NIDS) sensor and re-configure it to review the network traffic traversing the secure zone and to alert whenever connections were attempted on the default Oracle TNS Listener (1521), MS SQL (1433/1434), MySQL (3306) PostgreSQL (5432) or other common DB ports (Litchfield, 2005).

While the secure zone was being built, the DB environments (production and non-production lanes) were monitored to identify (learn) the expected and actual ports and applications connecting to the database for a period of several months. During the monitoring window, the ACME business units using the financial system ran their standard monthly, quarterly and yearly reports to identify all of the standard DB system connections. This list was reviewed, approved and documented as the baseline.

Once monitoring was completed, the non-production DB was logically moved into the secure network zone and related, non-production, application configurations changes were made. Troubleshooting ensued. This process was repeated for the production DB and after another round of troubleshooting related problems, host based firewall access control list (ACL) rules were applied to the TNS listener (valid node checking) and host OS to drop all network traffic attempting to connect to the listener that was not identified in the monitoring baseline. Then, real-time alerting was configured to report any unusual attempts to access the DB to the DBE, DB team and the Information Security & Network Operations Center (ISNOC).

3.2.2. Secure Zones

After the second penetration test, ACME further expanded control 11 into a, "Secure Zone" philosophy where basic security services were established to protect any of the databases, web or fileservers residing in the same logical network. This control included other important layers such as ongoing vulnerability Scanning, rogue network device detection, system inventory discovery, boundary defenses and other controls described in SANS 20 critical controls (SANS.org, 2014). Of course, in a layered approach, these were added as the organization matured over time.

3.3. Secure Configuration for Hardware & Software on Laptops, Workstations and Servers (DB Hardening)

SANS Critical control 3 pertains to the, "Secure Configuration for Hardware & Software on Laptops, Workstations and Servers." During the pen test, the Red team was able to leverage exploits targeting the TNS listener since it had not been patched to the most recent major or minor versions available. As a result, once the IP address, (provided to the Red Team) Oracle TNS listener version, and SID were discovered, it was only a matter of time before the database was compromised.

The ACME Risk Manager communicated to the business the critical importance of patching and maintaining sensitive systems such as the financial database as part of routine maintenance and not as one-time project work (as well as recounting the results of the risk assessment conducted several months prior to the penetration test as evidence).

As a result, ACME management prioritized the patching of the database from 11.1.0.6 to 11.2.0.3. This effort was significant since it required a minor version release, but was managed well by the DB team as they tested the changes in the non-production environment and worked with the application teams to upgrade the financial software to a current version as well. Once the software version was patched with the latest Oracle Critical Patch Updates (CPU), future attacks bypassing other layers of defense, would also need to have zero-day exploits or other social engineering attacks ready to make any additional progress in an attack; thus significantly reducing the risk from breach.

3.3.1. Secure DB System

ACME further expanded control 3 into a, "Secure System" philosophy where important database scripts were created to automatically harden the database. These scripts were developed utilizing the Oracle, "Project Lockdown" recommendations, and best practices from industry experts, SANS.org and other regulatory bodies. These controls included, but were not limited to the following:

- Changing all default passwords on the database after build-out
- Revoking unnecessary privileges and public grants
- Expiring and locking important but unused accounts

- Enabling valid node checking (as described earlier)
- Configuring post authentication validation (an additional layer of defense for valid accounts that should not have privileges granted to them)
- Disabling the remote password file
- Verifying password strength of privileged accounts met the password policy security standard guidelines
- Ensuring logging was enabled for drop, adds, inserts, deletes and other significant (DDL and DML events)

The value of creating lockdown scripts became evident after subsequent Oracle databases major and minor versions were installed at ACME and the DBEs became more aware of the Oracle's practice of rolling back default permissions and account settings during the version upgrade process.

To gain visibility on the DB security effort, the DB team also produced a DB security dashboard (SharePoint site) to report violations from the database scripts to ensure ACME management understood the progress on securing the financial database and for ongoing database security efforts.

As an aside, like many penetration tests, the results of the Red Team exercise were used as the catalyst by the Risk Manager and others to drive a significant culture change and to help prioritize the security investments for ACME. Prior to this exercise, the ACME management team did not have sufficient data to make sound risk management and business decisions pertaining to the financial database.

The last of the three SANS critical security controls implemented to prevent a repeat of the financial database breach was, "Account Monitoring and Control."

3.4. Account Monitoring and Control

During the pen test, the Red Team was able to access different user accounts on the database without detection. The TNS listener logs were not granular enough to detect the user enumeration on the DB. Once an account was found (monfog) the first login attempt made by the Red Team was successful, so no failed logins were reported. This finding resulted in significant changes to the daily operations of the ISNOC and the account management of the DB, with the understanding that successful account logins could also be malicious.

During the penetration testing post mortem the DB team made three significant observations that were documented. (1) Few, if any of the privileged user accounts, or important roles or privileges granted to traditionally non-privileges accounts (making them higher risk) were being logged. This included the unnecessary (legacy) monfog account. In addition, (2) listener logs were being sent to the ISNOC, but tasks related to the TNS listener were not refined enough for the ISNOC analysts to clearly understand the actions they needed to take. (3) Further review indicated that failed logins were not sent via syslog or other logging methods nor were they reviewed as standard tasks by the ISNOC; although, it wouldn't have made a difference in this case. As a result, all three of these changes were made, improved upon, and included as standard tasks for every ISNOC shift.

3.4.1. Account Control

To reduce the risks from unauthorized accounts being used on the database, the database team reviewed and systematically removed legacy accounts that were not needed on the database. This process effectively reduced the accounts on the database from well over 250 to a few dozen. Shared accounts were disabled and replaced with unique accounts that could be attributed to a single, named individual.

For user or service accounts that were needed, the team began enforcing (via scripts) a password policy requiring all passwords to be at least 16+ characters with requisite alpha-numeric and supported symbol complexity including letters, numbers and the symbols "#", "_" and "$" (Emmons, 2005). (Prior to 11g, case sensitivity was not enforceable). After the DB team upgraded the database to version 11.2.0.3 and removed the legacy 10G password flag, they were able to require case sensitivity on all passwords. Password aging was also enforced through automated scripting and policy configuration.

The DB team also set many of the accounts that were necessary for DB functionality but not used by administrative users to be administratively expired and locked. Some accounts were also set to have an, "Unbreakable" password hash. This is a process where the DBE replaced the actual DES and SHA1 hashes in the data dictionary with the word, "Unbreakable" so they could never be used in an unauthorized manner.

(An example of changing the hash is below.)

```
o SQL> alter user CTXSYS identified by
   Unbreakable;
o User altered.
```

(See Appendix C for further details)

These user and service level account controls significantly improved the security of the DB for the second round of Red Team exercises.

4. Red Team Tries again

Upon completion of the three critical controls, the ACME management team requested another penetration test of the financial system database. For the second round of testing, the same information was given to the Red Team as part of the scoping and approval exercise.

Unfortunately for the Red Team, the additional security controls provided significant obstacles requiring substantially more time and effort to make it through the steps of the penetration test.

4.1. Control 11— Limitation and Control of Network Ports, Protocols and services

The Red Team began Reconnaissance of the database much the same way as the first exercise. One notable difference was the inability of the Red Team to quickly and easily access the database server directly. After substantial effort and several failed attempts, the Red Team acknowledged they could not successfully access the database without the need to socially engineer a privileged user with VPN access to access the secure segment.

With this concession, the ACME management team provided VPN credentials to the Red Team to continue testing as if they were a privileged user, attackers or insiders who had already compromised VPN account access. It was inevitable that the Red Team would have found a way to access the database through some out of scope channel, so providing the VPN access was considered a time savings agreement to determine if the other security controls were also implemented correctly. Arguably, this is where the Red Team exercise could have concluded, but ACME really wanted to exercise the other critical controls once inside the secure network.

4.2. Control 3 – Secure Configuration for Hardware & Software on Laptops, Workstations and Servers

After accessing the secure network zone, the Red Team began the process of trying to identify the database version and other details necessary to repeat the initial attack. They were disappointed again. The quiet Nmap string failed to identify the DB port running on TCP 1521. With a wider search, their Nmap fingerprinting software was able to identify the non-default TNS listener port, but the ISNOC was able to detect the Red Team's scan attempts and reported the unauthorized NIDS traffic to management.

ACME management was delighted to know that the defenses were working and the real-time alerting controls and operational tasks were effective. ACME management allowed the Red Team to continue attacking even though they had been identified, to determine if the other countermeasures were equally strong.

Moving ahead, the Red Team attempted to perform vulnerability scans on the TNS Listener but continued to get connection resets. They concluded that a host based firewall was involved and requested permission to spoof the IP Address of the Backup server for the remainder of the attack. The request was permitted as well as the request to disable any other valid node checking features. Again, this would have likely been the end of the exercise, but management wanted to see if the Red Team could still get in (simulating the DBE forgetting to turn the defenses back on after maintenance work or other human errors).

The Red Team then attempted to connect to the TNS listener but the Metasploit TNS Mixin' modules were unable to identify the SID or enumerate any user accounts on the database since the vulnerability used in the exploit had been patched. The Red Team was then able to identify the SID by researching on the database team's intranet site and

Wiki; a fact that was later corrected with stronger authentication requirements added the sites.

4.3. Control 16 – Account Monitoring and Control

With this last piece of critical information, the Red Team used the previously successful Pete Finnigan, Oracle password list as well as an internal word list to attempt to brute force the database. After several days of guessing/brute forcing passwords, none of the username/password lists provided successful access to the database. All the while, the DBE and the ISNOC received large numbers of failed login account alerts that clearly indicated unauthorized access attempts. The Red Team was officially out of time and the defenses held.

Had this been a real attack, likely the hackers would have used out of scope targets, social engineering or other methods to bypass many of the layers of defenses. The defense in depth approach of the SANS 20 Critical controls are designed to make real attacks difficult, time consuming and noisy enough that they alert the ISNOC and the DBEs that they are under attack so they can take appropriate actions (SANS.org, 2014). In this scenario, that was precisely the case.

5. Conclusion

The Red Team exercises crystalized the importance for ACME to get back to security basics described in several of the SANS Critical Controls. These basics included but were not limited to the following: enforcing and using strong passwords (control 16), proactively managing privileged DB accounts (control 16), and following a regular patch cycle for sensitive systems (control 3).

ACME also realized that to defend against skilled attackers, additional detective controls were needed. These controls, though not explicitly identified in the SANS Critical Controls but were inferred in the related controls including: changing default TNS Listener ports (control 11), establishing NIDS rules to fire when connections to default DB ports were attempted (control 11), and creating effective ISNOC tasks and procedures to alert when unauthorized attempts were discovered (control 16).

Over time, and in an attempt to provide additional layers of protection, ACME also discovered that their implementation of a secure zone (control 11) was very costly and difficult to implement in a large enterprise environment. Significant planning and an accurate inventory of network diagrams, ports and dependencies were needed prior to making a network segmentation move. The process to implement this control was the longest and most time consuming of them all and is recommended only after completing the more basic controls.

In summary, these findings suggest the SANS Critical Controls do provide strong protections again both Red Team and real world adversaries wishing to access confidential databases or other highly sensitive systems. It is understood that given enough time and opportunity, cyber-hackers can often circumnavigate or bypass many of the security controls in use. Businesses can protect themselves and take reasonable due care to protect their information using a defense in depth approach described in these three controls and expanding to include other security controls as needed.

References

Emmons, Jon (2005). *What are the default restrictions on Oracle Passwords?* Retrieved August 25, 2014 from http://www.lifeaftercoffee.com/2005/11/07/what-are-the-default-restrictions-on-oracle-passwords/

Finnigan, Pete (2014). *Default Password List*. Retrieved August 5, 2014 from http://www.petefinnigan.com/default/default_password_list.htm

Gates, Chris (2009). *Attacking Oracle with the Metasploit Framework*. Retrieved August 2, 2014 from http://www.blackhat.com/presentations/bh-usa-09/GATES/BHUSA09-Gates-OracleMetasploit-SLIDES.pdf

Gates, Chris (2009). *Attacking Oracle with the Metasploit Framework*. Retrieved August 2, 2014 from https://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-chris_gates-breaking_metasploit.pdf

Kennedy, David (2011). *Metasploit: The penetration tester's guide*. San Francisco: No Starch Press.

Kornbrust, Alexander (2009). *Oracle Metasploit Presentation*. Retrieved August 5, 2014 from http://blog.red-database-security.com/2009/07/30/oracle-metasploit-presentation/

Kornbrust, Alexander (2009). *Oracle Security blog*. Retrieved August 2, 2014 from http://blog.red-database-security.com/2009/11/29/ighashgpu-cracking-oracle-passwords-with-790-million-passwordssecond/

Litchfield, David (2007). *The Oracle hacker's handbook: hacking and defending Oracle.* Indianapolis, IN: Wiley Technology Pub..

Litchfield, David (2005). *The database hacker's handbook: defending database servers.* Indianapolis, IN: Wiley Pub..

Mastin, R. (2001). *Telecom & Networking Glossary: A plain English guide to cutting-edge telecommunications technology, terms and acronyms (2nd edition).* Newport: Aegis Pub..

N. Dean Sapp, deansapp@hotmail.com

McClure, S., & Scambray, J. (2009). Scanning. *Hacking exposed 6: network security secrets & solutions* (10th anniversary ed.,). New York: McGraw-Hill.

Oppleman, V., & Friedrichs, O. (2005). Redefining the DMZ: Securing Critical Systems. *Extreme exploits: advanced defenses against hardcore hacks*. New York: McGraw-Hill/Osborne.

Oracle.com (2006). *Oracle database express edition 2 day developer guide.*

Retrieved August 29, 2014 from http://docs.oracle.com/cd/B25329_01/doc/appdev.102/b25108.pdf

SANS.org (2014). *Critical Security Controls.* Retrieved August 2, 2014 from http://www.SANS.org/critical-security-controls/

Stuber, Sean (2009). *How Oracle Stores Passwords*. Retrieved August 24, 2014 from http://www.experts-exchange.com/Database/Oracle/A_855-How-Oracle-Stores-Passwords.html

N. Dean Sapp, deansapp@hotmail.com

5. Appendix A

5.1.1. Critical Security Controls - Version 5

The SANS Critical Security Controls are described on the SANS.org website as an effective cyber defense against growing and increasingly sophisticated cyber warfare. The history and practicality of these defenses are described below (SANS.org, 2014).

"Over the years, many security standards and requirements frameworks have been developed in attempts to address risks to enterprise systems and the critical data in them. However, most of these efforts have essentially become exercises in reporting on compliance and have actually diverted security program resources from the constantly evolving attacks that must be addressed. In 2008, this was recognized as a serious problem by the U.S. National Security Agency (NSA), and they began an effort that took an "offense must inform defense" approach to prioritizing a list of the controls that would have the greatest impact in improving risk posture against real-world threats. A consortium of U.S. and international agencies quickly grew, and was joined by experts from private industry and around the globe. Ultimately, recommendations for what became the Critical Security Controls… were coordinated through the SANS Institute" (SANS.org, 2014).

The twenty domains of security controls in version 5 include:

1: Inventory of Authorized and Unauthorized Devices

2: Inventory of Authorized and Unauthorized Software

3: Secure Configurations for Hardware and Software on Mobile Devices, Laptops, Workstations, and Servers

4: Continuous Vulnerability Assessment and Remediation

5: Malware Defenses

6: Application Software Security

7: Wireless Access Control

8: Data Recovery Capability

9: Security Skills Assessment and Appropriate Training to Fill Gaps

10: Secure Configurations for Network Devices such as Firewalls, Routers, and Switches

11: Limitation and Control of Network Ports, Protocols, and Services

12: Controlled Use of Administrative Privileges

13: Boundary Defense

14: Maintenance, Monitoring, and Analysis of Audit Logs

15: Controlled Access Based on the Need to Know

16: Account Monitoring and Control

17: Data Protection

18: Incident Response and Management

19: Secure Network Engineering

20: Penetration Tests and Red Team Exercises (SANS.org, 2014).

N. Dean Sapp, deansapp@hotmail.com

6. Appendix B – Glossary of Terms

Database (DB) – "1. A set of data that is required for a specific purpose or is fundamental to a system, project, enterprise or business. A database may consist of one or more data banks and be geographically distributed among several repositories. 2. A formally structured collection of data. In automated information systems, the database is manipulated using a database management system" (Mastin, 2001).

Database Engineer (DBE) – A technologist performing tasks related to a database.

Database Management System (DBMS) – The DBMS typically includes special software and/or hardware, a data storage structure (schema) and a method for users to interact with the data stored in a relational way.

Data Definition Language (DDL) – Database statements related to database schema or structure operations. Common examples include Drop, Alter, Create, Rename, etc.

Data Encryption Standard (DES) – The cryptographic hash algorithm used by Oracle to convert plain text into cypher text for storage of passwords in version 10g and earlier.

Data Manipulation Language (DML) – Database statements used to manage data within schema objects. Common examples include Insert, Update, Delete, Select, etc.

Hash – The cryptographic process used by Oracle and other database vendors to convert varying length plain text passwords into a fixed length numerical cypher text value. DES and SHA-1 are common hash algorithms used by Oracle.

Information Security & Network Operations Center (ISNOC) – The team responsible for 24 X 7 X 365 monitoring, assessment and defense of the computer systems at ACME.

Internet Protocol (IP) address – "The method (or protocol) used to route information sent from one computer to another on the Internet or other data networks, such as corporate intranets or industry extranets" (Mastin, 2001). A unique network address allowing computers on the ACME network to talk to one another.

Oracle Service Identifier (SID) – The equivalent to the name of the database used by Oracle as a unique identifier.

Red Team – The team of internal ACME penetration testers or, “ethical hackers” whose purpose is to identify security risks before they are discovered and exploited by malicious individuals.

Salted Hash – The concept of adding an additional piece of information (called the salt) to the cryptographic process of hashing passwords (converting plaintext into cypher text.) In Oracle version 10g, the salt value appears to be the same as the username.

Secure–Hash 1 (SHA–1) – The secure-hash cryptographic hash algorithm used by Oracle to convert plain text into cypher text for storage of passwords in version 11g.

TNS Listener – Part of the Oracle architecture used to make connections to the database. Often the contents of the tnsnames.ora file defines the TNS listener settings needs to connect to the database.

N. Dean Sapp, deansapp@hotmail.com

7. Appendix C – Tool Examples and Explanations

Nmap

Command: `Nmap –p T:1521 –sV (Target IP) –oG tcp_oracle_scan_results.txt`

The Red Team used this specific Nmap command (–p T:1521) to scan for only the default Oracle TCP port of 1521 to avoid a broad sweep of ports and detection by the monitoring team.

The –sV switch allows Nmap to detect what service and version are running on the target IP (which was obfuscated per ACME's request), basically performing database fingerprinting.

The –oG switch provides an organized way to output the results to a text file for evidence collection for the penetration test report.

Metasploit "Mixin" modules – The Red Team followed Chris Gate's four step approach (Gates, 2009).

As a note, the older Metasploit 3.3 framework was used instead of newer releases such as 3.3.3 or the more current 4.7.2 to ensure the fewest number of troubleshooting steps during the testing, of which there were many particularly with the oracle instantclient_10_2 install as well as the older version of Ruby 1.8.6.

Step 1–Determine the Oracle Version (Assuming you already have the IP Address)

```
Msf auxiliary(tnslsnr_version)> set RHOSTS 10.10.10.47
RHOSTS=>10.10.10.47
Msf auxiliary(tnslsnr_version)>  run
```

(Trimmed for Brevity)

```
[*] Host 10.10.10.47 is running: Linux: Version
11.1.0.6 - Production
```

N. Dean Sapp, deansapp@hotmail.com

```
[*] Auxiliary module execution completed
Msf auxiliary(tnslsnr_version)>  pt_notes
[*] Time: Fri May 23 15:40:32 -0600 2014 Note:
host=10.10.10.47 type=VERSION data=Linux: Version
11.1.0.6 - Production
```

Step 2–Determine the Oracle SID with sid_enum.

The Red Team used this to successfully identify the SID against a dictionary of possible four letter SIDs in the sids.txt file.

```
Msf auxiliary (sid_enum) > run
[*] Identified SID for 10.10.10.47: PLSExtProc
[*] Identified SID for 10.10.10.47: ACME
[*] Identified SERVICE_NAME for 10.10.10.47:
PLSExtProc
[*] Identified SERVICE_NAME for 10.10.10.47: ACME
```

The (sid_enum) calls:

```
Msf auxiliary (sid_enum) > use
auxiliary/scanner/oracle/spy_sid
Msf auxiliary (spy_sid)> run
```

(Trimmed for Brevity)

Or, determine the Oracle SID with a brute force attack with a txt file of possible SIDs.

```
Msf auxiliary (sid_brute) > run
[*] Starting brute force on 10.10.10.47, using sids
from /home/ds/ms/msf3/dev/data/exploits/sid.txt
[*] Found SID ‘ACME’ for host 10.10.10.47
[*] Auxiliary module execution completed
```

Step 3–Determine the Oracle Username/Password combinations

Using the Pete Finnigan default Oracle username and Password list, create or download the list of default usernames and password combinations. For the attack, the Red Team also used a list of well-known internal accounts.

```
Msf auxiliary(login_brute)> set SID ACME
SID => ACME
Msf auxiliary(login_brute)> run
```

(This is where you would expect to see failed logins)

```
[-] ORA-01017: invalid username/password; login denied
But they did not since they had tested the account
access prior using SQL Plus.
[*] Auxiliary module execution completed
Msf auxiliary(login_brute) > pt_notes
[*] Time: Fri May 23 16:40:32 -0600 2014 Note:
host=10.10.10.47
Type=BRUTEFORCED_ACCOUNT data=MONFOG/MONFOG
```

Step 4–Putting it all together for the attack.

With the data found so far, the Red Team can launch Privilege Escalation SQL Injection attacks with the lt_findricset module.

```
Msf auxiliary (lt_findricset) > set RHOST 10.10.10.47
RHOST=>10.10.10.47
Msf auxiliary (lt_findricset) > set RPORT 1521
RPORT=>1521
Msf auxiliary (lt_findricset) > set DBUSER MONFOG
DBUSER=> MONFOG
Msf auxiliary (lt_findricset) > set DBPASS MONFOG
DBPASS=>MONFOG
Msf auxiliary (lt_findricset) > set SID ACME
SID=>ACME
Msf auxiliary (lt_findricset) > set SQL GRANT DBA TO
MONFOG
SQL=> GRANT DBA TO MONFOG
Msf auxialry )lt_findricset)>run
```

(Trimmed for Brevity)

This attack worked but in this case, it wasn't necessary. The monfog account had access to the data dictionary so hashes were available. Once cracked, the SYSADM user with the DBA privileges was accessible and the database contents were known.

The code for the tnslsnr_version.rb and other Metasploit Mixin modules are available from: https://github.com/pwnieexpress/metasploit-framework/blob/master/modules/auxiliary/scanner/oracle/tnslsnr_version.rb

Setting an, "Unbreakable" password hash

The process for changing the password hash value to a value that will not match the expected hash value for Oracle 10g and prior versions is simple; but it is recommended to securely backup a copy of the existing hash if you believe there is a chance you will need to ever use it again (Stuber, 2009).

```
SQL> connect dbadminusername/dbadminpassword@acme
```

(Connect to the database with a user with permissions to query DBA_USERS and to ALTER USER.)

```
Connected.
SQL> select username, password from dba_users where
username = 'CTXSYS';
```

(Backup the hash value if you want to restore it later.)

```
USERNAME   PASSWORD
---------- ------------------
CTXSYS     24ABAB8B06281B4C

SQL> alter user CTXSYS identified by Unbreakable;
User altered.
```

(Change the hash to "Unbreakable.")

For Oracle version 11g, new security controls were introduced to support passwords with case sensitivity. Prior to version 11g, all passwords were converted to uppercase before hashing begins. For backward compatibility, Oracle includes legacy support for DES hashes with a flag in the database labeled "10G". If the 11g database PASSWORD_VERSION column of the DBA_USERS contains this flag, these instructions will work. If this flag is not present in the PASSWORD_VERSION column, then the database is using the more secure SHA-1 hash and the DBE will need to find the hash in the SYS.USER$.SPARE4 column (Stuber, 2009).

The Red Team discovered the 10G flag in the PASSWORD_VERSION column, so they knew the database contained both the DES and SHA-1 versions of the hashes.

N. Dean Sapp, deansapp@hotmail.com