SANS ANALYST PROGRAM

Sponsored by Oracle

Oracle Database Security: What to Look for and Where to Secure

A SANS Whitepaper – April 2010

Written by Tanya Baccam, SANS Oracle Database Security Course author/instructor

Authentication

Access Controls

Secure Configuration

Auditing

Defense in Depth

Introduction

Information assets are what differentiate many companies today. Information such as customer lists, financial data, personally identifiable information and proprietary information are common types of data stored in databases. In order to properly protect this information from external and internal attacks and misuse, organizations need to implement controls as close to this information as possible. This means protecting the data at the database layer.

By protecting the data at the database layer, we can control and manage critical information assets centrally. However, to properly protect the data at the database layer, there are a number of steps that must be taken within and around the database itself. In this paper, we discuss critical steps and controls for protecting the Oracle database, followed by steps for protecting data within the Oracle database.

When it comes to security audits, many of the vulnerabilities identified by auditors exist because of hold-over default configurations from earlier versions of Oracle that did not get changed during the upgrade process. Others exist because of the failure to take security into account when the system was designed.

Databases are extremely complex, so it is important for a database administrator (DBA) to understand the potential security impacts of the multiple configuration options that are available. When administrators focus on availability, they often overlook configuration issues that can introduce security vulnerabilities and expose confidential data. Applications and their handling of access and encryption also present a significant risk to backend databases. Just as the database needs to consider security, the application must also take security into account. Finally, there are publicized vulnerabilities in the database software itself.

So, what steps do you need to take to secure an Oracle database properly? In this paper, we discuss four risk management basics that you must address to protect databases and their sensitive data:

- **Authentication**
- **Access controls**
- **Secure configuration**
- **Auditing**

After we look at these key areas, we will also look at Oracle's Defense in Depth and its ability to protect the data in more detail and with more automation.

Authentication

Authentication is the first step in obtaining access to a database. Oracle has multiple ways to authenticate to the database. With those methods come multiple risks if authentication isn't handled properly for the environment. The most obvious vulnerabilities are default accounts and passwords. Since the release of Oracle9i, Oracle has been locking accounts and setting the passwords associated with most default accounts to be expired upon installation. In fact, only a handful of accounts are typically accessible after an installation.

Nonetheless, there are approximately 600 default username and password combinations that can potentially exist within the Oracle environment, depending on how it's configured. It's critical to the security of the system that you modify these default credentials. Fortunately, in Oracle Database 11g, Oracle includes the **dba_users_with_defpwd** view, which can make it easy to check for user accounts with default passwords by querying:

```
SQL> SELECT * FROM dba_users_with_defpwd
```

However, it is important to know that not all passwords can be verified via this view. Some are application specific, and we can't expect the Oracle database to know about all the application-specific accounts that may get added to the database. Therefore, you should still conduct a check to verify that no default accounts remain. To check for these accounts, you can leverage a default account listing,[1] and then compare it to the output you get from the following query:

```
SQL> SELECT username, password from dba_users;
```

If an application requires a default account, contact the vendor to request a change to the application, and then thoroughly audit access with the account.

Oracle also provides profiles to further protect the database accounts. By querying **dba_users**, we can identify what profile has been applied to which account:

```
SQL> SELECT username, profile FROM dba_users;
```

Once we know which profiles are being leveraged, we can query **dba_profiles** to determine the profile settings:

```
SQL> SELECT * FROM dba_profiles;
```

This is also where you will find the password-related settings.

[1] www.petefinnigan.com/default/default_password_list.htm

Here is a list of recommended profile settings for a typical administrator account:

```
SQL> create profile admin limit
  2   failed_login_attempts 3
  3   password_life_time 45
  4   password_reuse_max 20
  5   password_lock_time 1/24
  6   sessions_per_user 3
  7   connect_time 60
  8   idle_time 5;
```

Oracle also provides operating system authenticated accounts. Essentially, Oracle trusts that the operating system has authenticated the user. All Oracle looks at is the password.

There are also a couple of configuration parameters that come into play when using operating system authenticated accounts, specifically, the **remote_os_authent** and **remote_os_roles** parameters. These parameter settings determine whether Oracle will trust remote operating systems or only the local operating system for these accounts (or roles). For example, if **remote_os_authent** is set to TRUE, Oracle will trust any remote operating system. Therefore, you should limit the number of operating system authenticated accounts you allow and what operating systems can do the authentication. The following query identifies accounts that are operating system authenticated:

```
SQL> SELECT username FROM dba_users WHERE password='EXTERNAL';
```

When additional controls are required, use strong authentication, such as the controls provided by PKI, Kerberos, or RADIUS. Oracle gives you the capability of leveraging multiple authentication methods through Oracle Advanced Security.[2]

In addition, Oracle provides a default password complexity check with the database. The script is called **utlpwdmg.sql** and can be found in **$ORACLE_HOME/rdbms/admin**. The default password complexity check verifies the following password attributes:

- The password contains no fewer than eight characters and does not exceed 30 characters.
- The password is not the same as the user name, nor is it the user name spelled backward or with numeric characters appended.
- The password is not the same as the server name or the server name with the numbers one through 100 appended.
- The password is not too simple, for example, **welcome1**, **database1**, **account1**, **user1234**, **password1**, **oracle**, **oracle123**, **computer1**, **abcdefg1**, or **change_on_install**.
- The password includes at least one numeric and one alphabetic character.
- The password differs from the previous password by at least three letters.

You can create your own customized password complexity check simply by copying and editing the default script Oracle provides. Once the script is ready, have your database administrator run it. Please refer to the Oracle documentation for instructions on how to turn on the password complexity check.

[2] www.oracle.com/technology/deploy/security/database-security/advanced-security/index.html

Access Controls

Verifying the access controls is critical for properly securing the database. After a user has been authenticated, access controls dictate what that user is allowed to do. Historically, the primary focus for application developers was to make it easy to collaborate and get the application built quickly. As a result, there was little, if any, focus on the common security principal of least privilege. Usability concerns also contributed to users being granted a number of privileges and/or permissions that they did not need. For example, when functionality is not operating as expected, an unknowing administrator might issue a command like "grant all privileges to user" to allow access, sometimes just temporarily, but then forget to revert the privileges back to the original settings. With this level of access, users receive over 100 privileges, many of which they may not need, and some of which may be in violation of regulatory compliance statutes.

There is also a PUBLIC access setting in which any user that has an account for the Oracle database is automatically given privileges granted to PUBLIC. By default, PUBLIC has a larger number of privileges, many of which should be restricted in most environments. A specific privilege can literally get granted to PUBLIC. When locking down access controls, then, we need to look at each individual user's privileges as well as the privileges granted to PUBLIC. The following queries identify the privileges granted within the environment:

```
SQL> SELECT * FROM dba_sys_privs;
SQL> SELECT * FROM dba_role_privs;
SQL> SELECT * FROM dba_col_privs;
SQL> SELECT * FROM dba_tab_privs;
```

Separation of duties is also important. Separation of duties applies in a lot of different ways in environments. For example, the roles of DBA and auditor should have differing responsibilities with separate access requirements. It's also important to separate environments—production, test, and development—another separation of duties issue that organizations sometimes miss.

Links make great pathways to escalating privileges. Review the database links to ensure that only authorized links are leveraged and that the links remain within the same environment, for example prod to prod, test to test, or dev to dev. Conduct a review of each link and its business purpose to minimize escalation. The following query identifies the links that exist within a database:

```
SQL> SELECT * FROM sys.link$;
```

Finally, for access control purposes, you must also take the application into consideration. Sometimes it is impossible to implement controls at the database level, and we must rely on the application. Each application is different and can introduce unique risks to the database, such as vulnerabilities to SQL injection attacks and other application-to-database attack methods. For a look at some of the common web application vulnerabilities, visit www.owasp.org.

Secure Configuration

When it comes to the topic of configuration and where to secure, we cannot cover every possible step. Here, we highlight some of the key areas of configuration to address. These include:

Listener: The initial contact point for an Oracle database is the listener. When a remote user wants to connect to the Oracle database, he or she contacts the listener to make that connection. Without the listener, we could not connect to the database. It is important to make sure that you are logging the listener. Also be sure to leverage the local operating system authentication architecture that has been incorporated in all releases of Oracle since the release of Oracle Database 10g. The listener.ora file will contain information such as the log settings.

Parameters: Following is a list of key parameters that can control many of the security risks to the database. Certainly these are not all of the important parameters, but they are the highest-risk parameters. Some of these parameters did not exist prior to Oracle Database 11g, although the majority did. A suggested setting or an applicable note is included in parenthesis in some places. Note that there are always real-world considerations that may or may not allow the suggested setting to be implemented.

- **AUDIT_FILE_DEST**
- **AUDIT_SYS_OPERATIONS (Should be set to TRUE.)**
- **AUDIT_TRAIL (Avoid FALSE or NONE settings.)**
- **DIAGNOSTIC_DEST**
- **DISPATCHERS**
- **GLOBAL_NAMES (Should be set to TRUE.)**
- **LOG_ARCHIVE_% (Note: There are multiple parameters that exist that begin with LOG_ARCHIVE.)**
- **MAX_ENABLED_ROLES**
- **O7_DICTIONARY_ACCESSIBILITY (Should be set to FALSE.)**
- **OS_AUTHENT_PREFIX (Should be set to NULL, if possible. Should not be set to ops$.)**
- **OS_ROLES (Should be set to FALSE.)**

- **REMOTE_LISTENER (Should be set to NULL, unless a remote listener is needed.)**
- **REMOTE_LOGIN_PASSWORDFILE (Should be set to NONE, if possible.)**
- **REMOTE_OS_AUTHENT (Should be set to FALSE.)**
- **REMOTE_OS_ROLES (Should be set to FALSE.)**
- **RESOURCE_LIMIT (Should be set to TRUE.)**
- **SEC_CASE_SENSITIVE_LOGON (Should be set to TRUE.)**
- **SEC_MAX_FAILED_LOGIN_ATTEMPTS (Should be set to 10.)**
- **SEC_PROTOCOL_ERROR_FURTHER_ACTION (Avoid the setting NONE.)**
- **SEC_PROTOCOL_ERROR_TRACE_ACTION (Avoid the setting NONE.)**
- **SEC_RETURN_SERVER_RELEASE_BANNER (Should be set to FALSE.)**
- **SMTP_OUT_SERVER (Should list only authorized SMTP servers, if utilized.)**
- **SPFILE**
- **SQL92_SECURITY (Should be set to TRUE.)**
- **UTL_FILE_DIR (Should be set to a specific directory used only for necessary purposes. Values such as /tmp or * should not exist.)**
- **_TRACE_FILES_PUBLIC (Should be set to FALSE.)**

You can learn more about each of these parameters from Oracle's site.[3]

Patching: You can address the security vulnerabilities we mentioned earlier by applying patches. Without the appropriate patches in place, an attacker may be able use open vulnerabilities to escalate their privileges or even, potentially, to gain access. To view the patches that have been installed, use the command opatch lsinventory at the operating system.

Surrounding Environment: Ultimately, if you greatly simplify an Oracle database, it is essentially a bunch of files on an operating system; therefore, operating system security is critical. Additionally, network security controls, such as firewalls, IDS/IPS, segmentation, and so on, are extremely useful and provide additional layers of protection against attackers and insider abuse of the database.

[3] http://download.oracle.com/docs/cd/E11882_01/server.112/e10820/index.htm

Auditing

Security means not only preventing attacks, but also detecting potential attacks. After all, the best security in the world is not going to stop every attacker. That is where auditing can be beneficial. Auditing allows us to monitor the environment and identify potential attacks. First, the **audit_sys_operations** parameter should be set to TRUE. The **audit_trail** parameter tells us if basic auditing has been enabled within the database and where the database audit data is being sent. For example, audit data can be written to the database, syslog, or to XML files based on how the parameter is specified. Assuming auditing has been enabled, specific views tell us what auditing is occurring. Specifically, the following queries tell us if statement or privilege actions are being audited.

```
SQL> SELECT * FROM dba_stmt_audit_opts
```

```
SQL> SELECT * FROM dba_priv_audit_opts;
```

Use the following query to determine the object auditing:

```
SQL> SELECT * FROM dba_obj_audit_opts;
```

If auditing is not turned on, you should enable auditing of activities such as creating sessions and other data definition commands such as creating or dropping tables. The command **audit create session** would audit such activity. Starting with Oracle Database 11g, Oracle provides a recommended list of default audit settings. You can turn these settings on during install with Oracle Database 11g Release 1. Oracle Database 11g Release 2 automatically turns on the recommended default settings. Look for the list of the default audit settings in the Oracle documentation.

Oracle also offers Fine-Grained Auditing (FGA), which allows you to audit under certain circumstances. For example, you could create a policy that says, "When a user reads an annual salary greater than $1,500, create an audit record." The following is an example of such an FGA policy:

```
SQL> begin
  2    dbms_fga.add_policy(object_schema=>'SCOTT',
  3         object_name=>'EMP',
  4         policy_name=>'EMP_POLICY',
  5         audit_condition=>'sal>1500',
  6         audit_column=>'ename,deptno');
  7  end;
  8 /
PL/SQL procedure successfully completed.
```

In this case, we have created a policy named **EMP_POLICY** on the **scott.emp** object. If a **sal** that is greater than 1500 is read and the query includes the **ename** or **deptno** column, an audit record will be created. As this example illustrates, you can get very specific with FGA auditing. Oracle Database 9i originally offered FGA for SELECT statement. With Oracle Database 10g, Oracle added FGA for DML (data manipulation language) such as INSERTS, UPDATES and DELETES.

FGA auditing creates a separate audit log, and the policies are stored in **sys.fga$**. Review each policy to understand what auditing has been enabled for FGA purposes.

Once auditing it enabled, it is important to centralize the audit data and create reports so you can review the audit records. Create a business process that includes reviewing the audit trails on a regular basis. You can use Oracle's Audit Vault to centralize and analyze the audit data. Oracle Audit Vault will automatically clean up legacy data after it has transferred the data to its secure warehouse.

[3] http://download.oracle.com/docs/cd/E11882_01/server.112/e10820/index.htm

Defense in Depth

Oracle continues to update and provide additional controls that allow us to implement an ever more effective defense-in-depth strategy around our database systems, applications and data. Here is a brief description of some of Oracle's latest in-depth security options available as extra software components.

Oracle Advanced Security: Encryption and Authentication

One of the critical areas that organizations have become more and more concerned about is encryption. Most compliance and regulatory requirements have dictated that encryption must be addressed. However, what we're really talking about is encryption for data at rest and data in transit working with strong authentication controls. The Oracle Advanced Security option provides coverage in all these areas with Transparent Data Encryption (TDE), network encryption, and strong authentication.

Data at rest: For data at rest, TDE is a feature of Oracle's Advanced Security beginning with the Enterprise Edition, Oracle Database 10g Release 2. You can apply TDE at the column level or to an entire tablespace in Oracle Database 11g Release 1 and later versions. This allows flexibility when it comes to what data we want to encrypt and where, and it is completely transparent to the user and applications. TDE protects data when someone attempts to read the raw data files using UNIX-level commands such as grep, bypassing the Oracle authentication and access control mechanisms.

There are a few key steps you must take to set up TDE.

1. First, create the master encryption key. There is one master encryption key per database, and that key is used to encrypt other keys inside the database. The master encryption key is stored in an Oracle Wallet (a PKCS #12 file stored on the operating system). Once the master encryption key is initialized you should immediately back up the Oracle Wallet to multiple secure locations.
2. Identify the specific columns of data you want to encrypt (e.g., credit card number, Social Security number, address), as well as the encryption algorithm you wish to use. TDE automatically uses the Advanced Encryption Standard (AES) algorithm.
3. Issue an **alter table** statement to Oracle once you have identified the columns and algorithm to use, and all the column data will be encrypted. This is basically an update statement internally. At this point, an additional encryption key will be created for that table and stored in the Oracle data dictionary, encrypted using the master key. Once you have done this, all future data added to the table and updates to existing data will be encrypted automatically.

In Oracle Database 11g, it is also possible to store the TDE master key externally on a Hardware Security Module (HSM). A single HSM can be co-located on the network with multiple Oracle databases, providing a single storage location.

Data at rest can "rest" in many locations, such as in the database itself, on end-user systems, in backups, and within other environments. Be sure to consider each of the key business locations in your encryption policies. For example, data encrypted using the Oracle Advanced Security TDE will remain encrypted on backups and will also remain encrypted in Oracle REDO logs. However, if you wish to encrypt the entire Oracle database for backup purposes only, Oracle RMAN can use the Oracle Advanced Security TDE capabilities during the backup process. In addition, Oracle Secure Backup can backup up directly to tape while maintaining full encryption capabilities on the data sent to tape.

Data in transit: Data moving to and from the database must also be protected. At a minimum, administrative traffic should be encrypted using tools such as SSH. By default, the network traffic, excluding the authentication credentials, is sent to an Oracle database in cleartext. Therefore, we need to protect this communication. Oracle's Advanced Security Option can provide standards-based encryption (AES, 3DES, RC4) for data in transit, along with encryption key support. By using Oracle's Advanced Security you can enforce a policy that no unencrypted data can even be sent from the database. Using a VPN or SSH to encrypt the data in transit is also an option; however, this is another layer of security that can't be supported by the Oracle database and must be managed separately.

You can also use Oracle Advanced Security to enhance authentication controls. For greater identity assurance, implement strong/multiple factor authentication of database users and application servers. This can reduce the risk of stolen credentials within a given environment.

Oracle Data Masking: Obfuscating Development Data

Proper separation of environments, including production, test and development environments, is another critical control with which many organizations struggle. In most cases, developers should not have access to production data unless it is impossible to duplicate the test without it. If developers must have access to the production environment for troubleshooting or similar purposes, they should be given only temporary, read-only access. The data within the development environment should also be properly sanitized to ensure that production data is not leaked.

To ensure that developers don't have access to production data at regular intervals in the development and/or test environments, you can leverage Oracle's Data Masking. Essentially it makes substitutions for the original production data with realistic, but not actual, data for use in testing systems. The data can meet certain formats and requirements, so it's as similar to real production data as possible without giving away sensitive information in the nonproduction environments. Referential integrity can be preserved, so applications continue to work.

In order to use Data Masking, there are two key steps involved: creation of data masking formats and creation of definitions. Oracle's format library has common masking formats that you can adopt and use for sensitive data. For example, credit card numbers, Social Security numbers and phone numbers, among other data types, can be masked automatically. Once you have created the masking formats, you can create a data masking definition, which is the association of tables and columns with a masking format.

To setup data masking, select the column you wish to mask and add any associated columns, if the relationships are not already being enforced by the database. Once you have the column selected, you can associate the appropriate masking format with the column being masked. Once these items have been identified, a masking script is created. Read this script to make sure everything has, in fact, been set up correctly. DBAs have the capability to schedule the masking process at an appropriate time, such as right after a production database has been cloned to a staging environment.

Oracle Database Vault: Operational Controls and Separation of Duty

Oracle Database Vault enables transparent enforcement of operational controls within the database around when, where and how the database is accessed and by whom, as well as controls over what happens within the database from an administrative perspective. Oracle Database Vault enables organizations to prevent application data access by privileged users, enforce change control policies on database structures, and set a variety of user access controls per policy and regulatory requirements without changing a single line of code or performing expensive and time-consuming least privilege exercises. These types of automated controls are essential to addressing concerns over application bypass as well as outsourcing and database consolidation. Oracle Database Vault realms are essentially containers that provide protection zones from privileged users. The Database Vault administrator can specify what items should be placed in a realm. For example, to create a realm, the Database Vault administrator would simply type in a descriptive name, set the status to enabled, set the audit options, and then select the realm of objects, such as application objects, or specific tables.

Second, you can use command rules to create control over how SQL statements can be used in the database. Command rules can be used in conjunction with rule sets to do more specific filtering to control conditions where, when and how data can or cannot be viewed. For example, you could filter based on the time of day, IP address, host name or other user attributes. Third, you can set multifactor authorization with rule sets that leverage multiple factors such as IP addresses and the user information to control access. Database Vault comes with out-of-the box policies for Oracle applications, including Oracle E-Business Suite, Oracle JD Edwards Enterprise One, Oracle PeopleSoft Application, and Oracle Siebel. In addition, Oracle has worked with SAP to develop and certify realms that can be used in SAP environments.

The Oracle Database Vault dashboard allows for monitoring of policies and configuration setup, and it provides dozens of default reports. These capabilities allow you to monitor users and violations of controls that have been put in place. By simply selecting a report and clicking **Run Report**, you can retrieve user activity details.

Oracle Label Security: View Restrictions and Controls

Oracle offers a couple of other access control solutions such as Virtual Private Databases and Oracle Label Security. The database can enforce row-level access control. Virtual Private Databases provides row-level security such that a user is either allowed or disallowed from seeing certain rows of information. Essentially a WHERE clause known as a predicate gets added to a user's query and controls what records are returned to the querying user automatically.

Oracle Label Security takes this a step further by assigning a data classification label to data to control what a user can or cannot view. Oracle Label Security mediates access based on the user's label authorization (or security clearance) and the data classification label assigned to the data. Oracle Label Security provides a policy-based architecture that enables multiple sets of data classification labels to be defined within the database and managed using a single umbrella policy. Oracle Label Security is also integrated with Oracle Identity Management so that user label authorizations can be defined centrally within the enterprise. Oracle Label Security can also be used in a decision support role such that the user label authorizations become factors for Oracle Database Vault. This enables you to further separate duties within an environment. For example, an Oracle Database Vault command rule could verify a user's label authorization before allowing a command to execute.

Oracle Audit Vault: Monitoring for Compliance and Reporting

Oracle Audit Vault consolidates and secures the audit data generated by Oracle databases, as well as Microsoft SQL Server, Sybase and DB2, addressing many of the compliance reporting requirements. Centralization of audit information is important to identify when an attack may be occurring. Plus, this allows you to place monitoring controls and alerts in a centralized location. Oracle Audit Vault provides dozens of out-of-the-box compliance-related reports that help internal and external auditors quickly assess security and risk conditions.

In addition, Oracle Audit Vault provides entitlement reports for Oracle databases across the enterprise, providing a single view of users, roles and associated privileges. Oracle Audit Vault provides attestation for reports such that specific individuals must sign-off on a report. Once created, reports can be emailed to specific individuals. In addition, Oracle Audit Vault alerts can be used to highlight specific activity. Alerts can be categorized based on severity.

Oracle Audit Vault includes multiple roles to protect the audit data once it is consolidated into the Oracle Audit Vault warehouse. For example, the auditor role can view the reports and set the audit policies, while the administrator role can enable the collection of audit data and monitor the health of the database. The auditor can log into the Oracle Audit Vault dashboard for an overview of the environment, a listing of alerts and predefined reports, whereas the administrator can log into Audit Vault, and via the dashboard, set filters to focus in a particular area and access audit policy settings and list alerts.

Oracle Total Recall: Tracking Data Changes

Many times organizations also want to know how data has changed over a period of time. Oracle Total Recall can transparently track changes to data over a given timeframe. For example, if you wanted to see how an employee's salary data has been changed, Oracle Total Recall allows you to view this information. You can query data for forensic purposes, potentially as a result of an Oracle Audit Vault alert, or to identify errors that have occurred.

Oracle Configuration Management: Maintaining Best Practices

Security is a cyclical process. Risks change and our environments change, so we need to continually assess the security state of the environment. Oracle Configuration Management allows us to view, search, compare, analyze, look at historical data, manage violations and view reports on the current environment. Oracle's Configuration Management Pack provides the ability to scan multiple Oracle systems, enabling continuous scanning of the environments to identify and correct configuration drift over time.

As an illustration, under the Policies tab, by clicking on **Library**, you can view the policy rule library, which lists the policy rules that Enterprise Manager evaluates. You can customize the policies as required for the environment. Once the policies are finalized, you can use Configuration Management to identify any violations. When you click on the Policies tab, you can view the current violations that exist in the environment for an immediate list of vulnerabilities. Even better, you can also lock down a given configuration and prevent or receive alerts for any unauthorized changes to the configuration.

Summary

Databases store information that is critical to organizations and their customers. In order to protect that data, we need to take the time to secure our critical databases and the data within them.

The steps highlighted in this paper offer a good start to securing the Oracle database. Oracle also recommends its brief security primer as part of its Project Lockdown, a four-phase process for implementing in-depth security controls in Oracle databases.[4]

Oracle continues to develop defense-in-depth controls at the database layer, bringing security to the location where the organization's most critical data is housed and processed. We need to leverage these tools and secure the environment by turning off unnecessary services and making use of access controls, audit capabilities and other controls. Organizations should also look into Oracle Advanced Security options for strong encryption, audit and other in-depth features being provided by Oracle and its partners.

Databases, with all their complexities, will not be secured overnight. However, by continually implementing additional controls, database security can be achieved, maintained and improved upon over time.

[4] www.oracle.com/technology/pub/articles/project_lockdown/index.html

About the Author

Tanya Baccam is a SANS senior instructor as well as a SANS courseware author. She is the current author for the SANS Security 509: Securing Oracle Databases course. She works for Baccam Consulting, where she provides many security consulting services for clients, including system audits, vulnerability and risk assessments, database audits, and web application audits. Today much of her time is spent on the security of databases and applications within organizations. Tanya has also played an integral role in developing multiple business applications. She currently holds the CPA, GCFW, GCIH, CISSP, CISM, CISA, and OCP DBA certifications.

SANS would like to thank its sponsors: