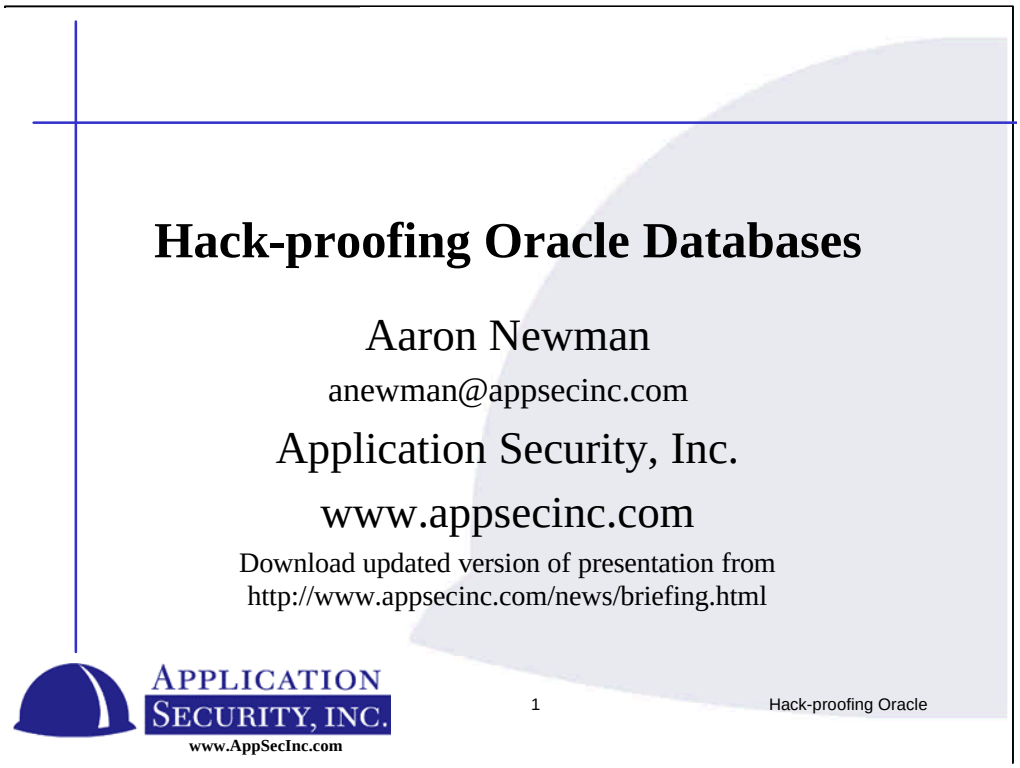
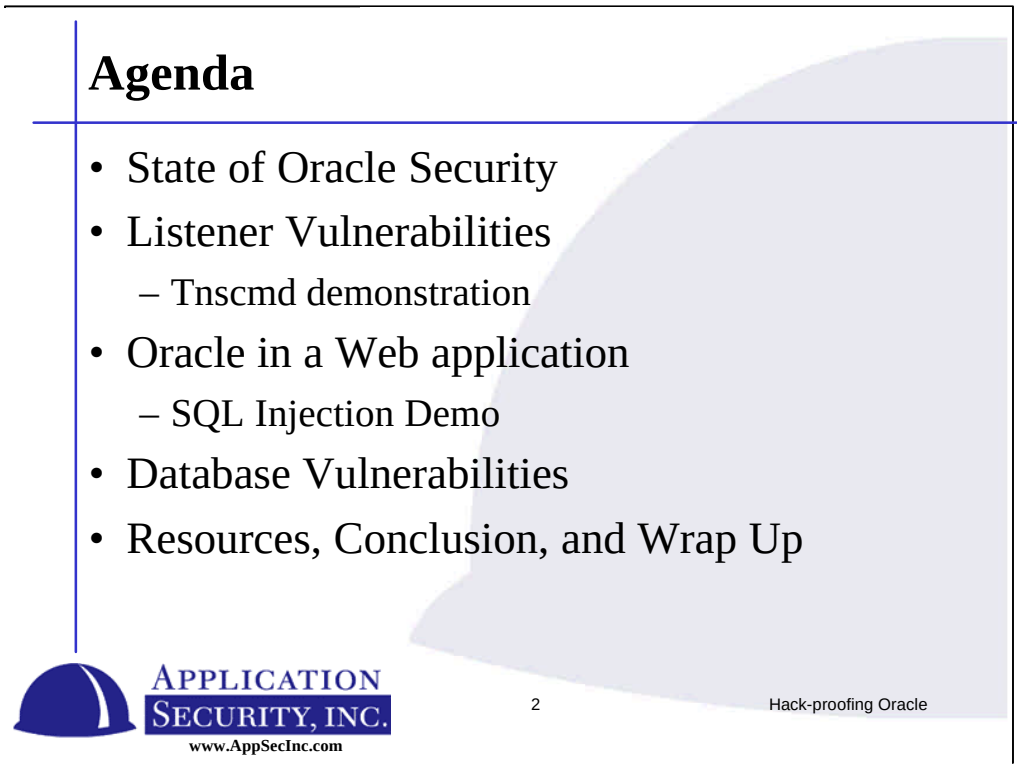


Good morning and welcome to this discussion on protecting your databases.

What we are going to do is look at security from a different perspective – to see how an attacker would approach your database and how you can stop an attacker. Without understanding the attacker, you likely will not be able to thwart an attack. That's why its important to be able to think like the attacker to be able to stop the attacker.



We will be covering various ways databases can be broken into and how to prevent yourself from being hacked.

Start with an introduction to talk about the state of Oracle security

We will be talking about listener security

Including a demonstration on using an attack tool that can be down loaded from the internet

We will talk about how a database could be hacked through a web server and how to prevent this from happening. We will be demonstrating an example SQL Injection demo.

We will be covering some basic database vulnerability and misconfigurations.

Then we will take some questions.

State of Oracle Security

APPLICATION SECURITY, INC.
www.AppSecInc.com

In the media

“Look what they've done to my database, Ma”

- **By John Leyden, The Register**

  Posted: 23/01/2002 at 17:40 GMT

- 1 out of 10 corporate databases connected to the Internet had a breach of security last year.
- Taken from a survey of 750 US database developers which also reveals growing concern about security issues.

http://www.theregister.co.uk/content/55/23800.html

APPLICATION SECURITY, INC.
www.AppSecInc.com

Underground Hacking World

- Increasing number of presentations on hacking databases at conferences
  - Blackhat, Defcon
- Exploits being written
- Worms found in the wild using databases
  - Alpha Voyager
  - Spida worm
- Whitepapers on attack Oracle

APPLICATION SECURITY, INC.
www.AppSecInc.com

Oracle Website – Alerts Web page

http://otn.oracle.com/deploy/security/index2.htm?Info&alerts.htm

- Prior to July 2000
  - One vulnerability acknowledged by Oracle
- From July 2000 to August 2002
  - 41 vulnerability reports on the Oracle website
- Vulnerabilities reported on SecurityFocus.com
  - About 75 vulnerabilities reported about Oracle

APPLICATION SECURITY, INC.
www.AppSecInc.com

Myth – Oracle is secure behind a firewall

- Is your database secure because it's behind a firewall?
- NO!!!
- Most security compromises are result of inside jobs
- Internal threats are the most dangerous
- Non-privileged users in the database

APPLICATION SECURITY, INC.
www.AppSecInc.com

What to do about the situation

- The problem exists but it won't be fixed tomorrow
- But we must start plugging these holes
- Become aware of the risks and threat
- Find the right solutions

APPLICATION SECURITY, INC.
www.AppSecInc.com

We do need to start taking a proactive approach to securing databases

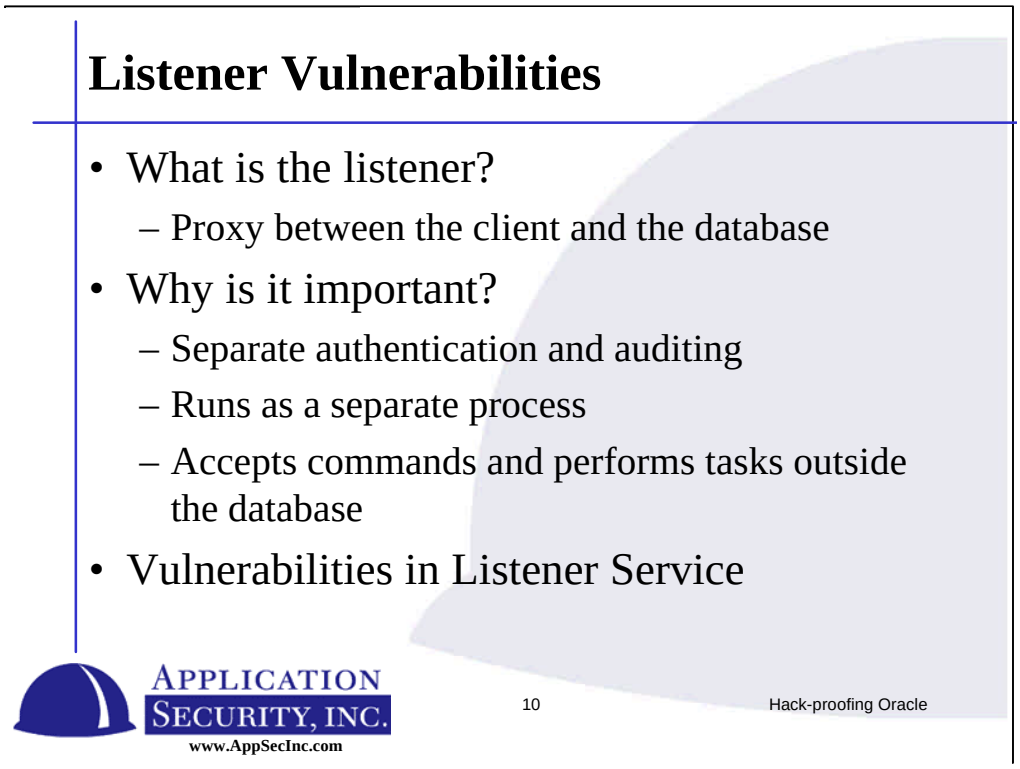
There is however a growing interest for database among black hats:

-In the past few year, the Black hat/Defcon conferences have had talks on Database security

-Exploits reported on Security Focus has increased dramatically over the last few months

Also, we will not be discussing operating system security, although this is a critical component of database security. You can't have one without the other.

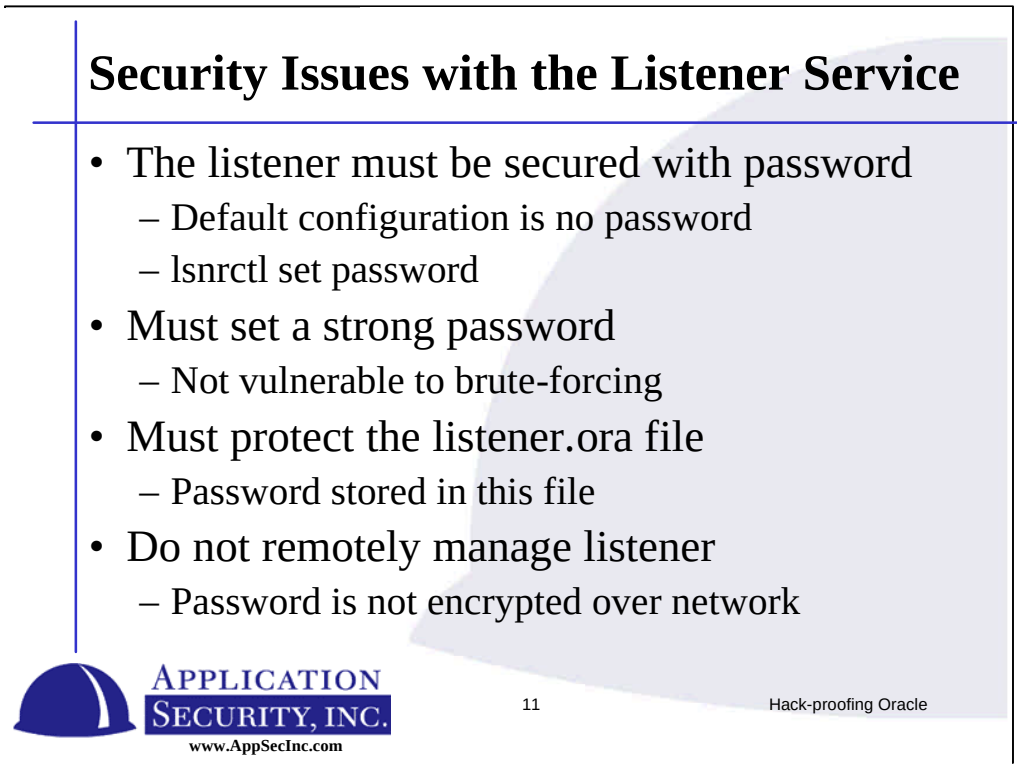
Securing the Listener service



Let's start by talking about a single component in the Oracle subsystem - the Listener service.

The listener service is a proxy during the connection process which sets up the connection between the client and the database. The client directs a connect to the listener which in turn hands the connection off to the database.

The issue that exists is that the listener has separate authentication and is controlled and administered outside of the database. The listener runs in a separate process, and in the past that process was run as setUID. The listener accepts commands and other tasks besides handing connections to the database.



How do we start securing the listener controlled.

First we must set a password on the listener service. Many DBAs don't even realize that a password must be set on the listener service. The listener accepts remote commands which means that if you do not set a password, any user on the network can send commands.

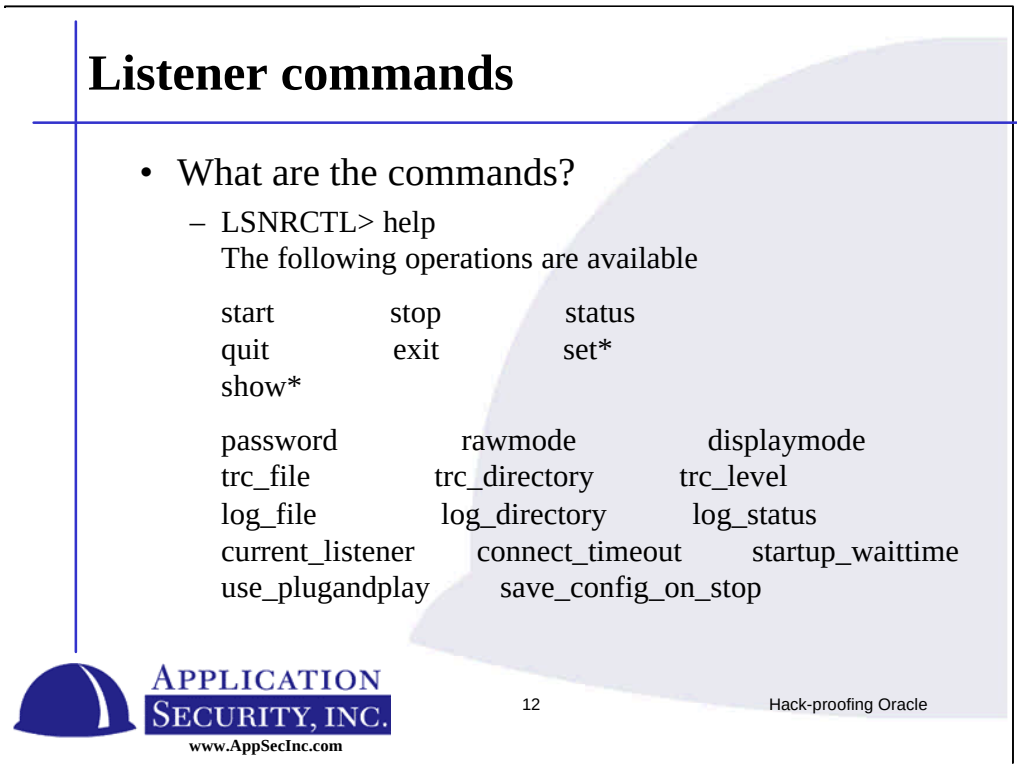
You can set the listener password one of two ways:

1) Using the listener controller utility – lsnrctl.
2) Setting the password in the listener.ora file.

You must choose a strong password because the listener controller is vulnerable to be brute-forced. A strong password is one which is not found in a dictionary, contains a combination of numbers letters and special characters, and is at least 8 characters long. If you do not set a strong password, a program can be written which attempts to connect to the listener using every possible combination of passwords. There is no password lockout feature of the listener.

The password itself is stored in the listener.ora file. If a user can read this file, they can use the password to log in remotely to the listener.

It is also recommended that you do not manage the listener remotely. The reason being is that the password is sent across the network in clear-text (or as a replayable hash). Anyone sniffing the network can discover the password if you are remotely managing the listener. Instead it is recommended that you connect



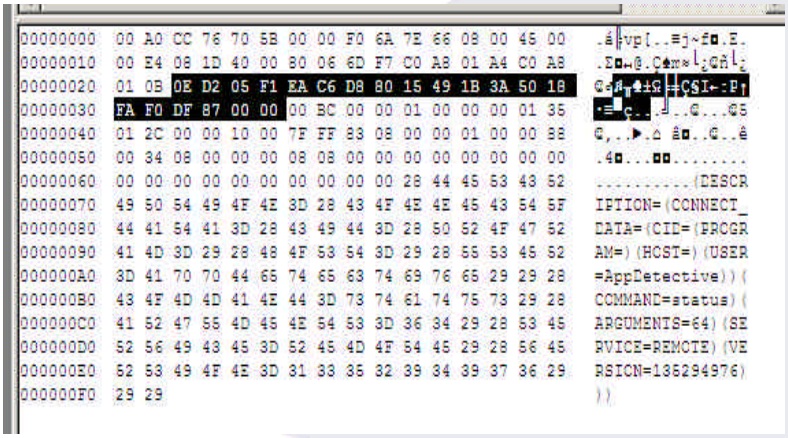
So what are the know problems with the listener services. To investigate this problems, we can pull up the listener controller and run the help command. This gives us a list of the commands we have at our access.

I know this is probably not readable for the audience. Take a look at the print-outs or the CDROM to look at the possible commands.

There is a command called set password - this is used to log us onto a listener. There is a couple of problems with this password - there's no lockout feature for this password, the auditing of these commands are separate from the standard oracle audit data, password does not expire - basically there is no password management features. This means writing a simple script to brute force this password, even if it is set strongly, is not very difficult.

Listener packet

- Below is an example of a command:



What happens when a command is entered into the listener controller?

A command is sent to the listener. If the listener is remote, the command is sent over the network. Here is an example of a packet. At the top of the packet we see a lot of strange characters – these represent fields and headers that form the structure of the packet.

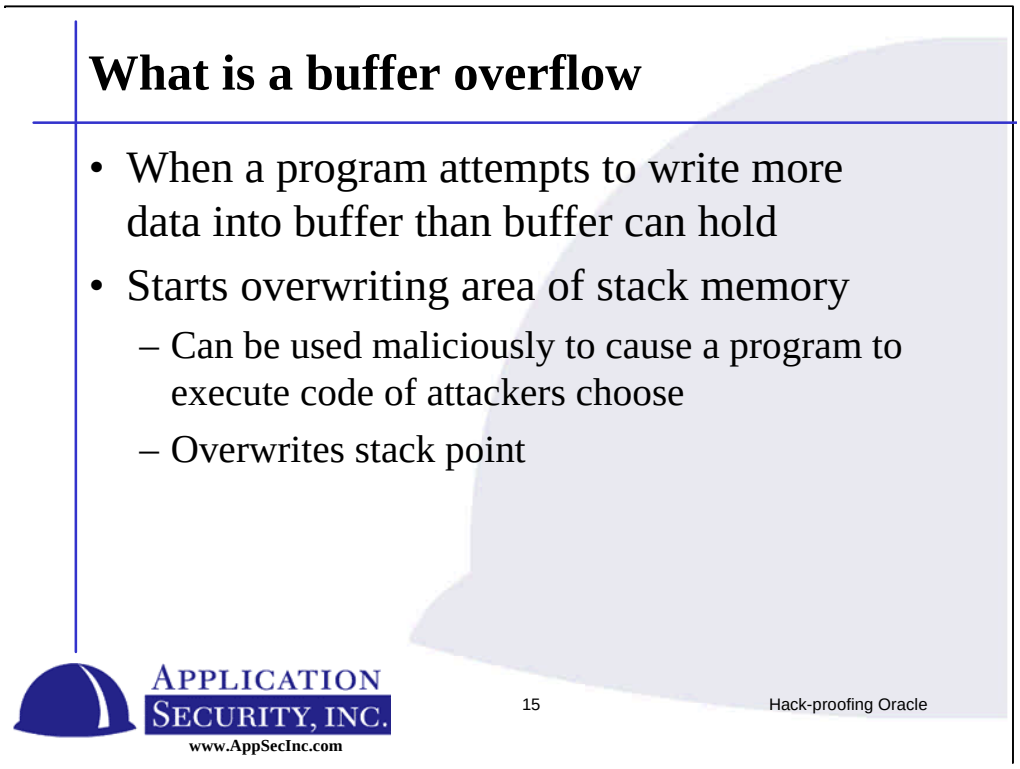
At the bottom on the packet is a connection string. Within this connection string is the command that the remote listener will execute. In this example we see:

COMMAND=status

Listener attack demo

http://www.jammed.com/~jwa/hacks/security/tnscmd/

APPLICATION SECURITY, INC.
www.AppSecInc.com



Before we start looking at buffer overflows in Oracle, we need to understand what a buffer overflow is.

Programs set up areas of memory to read and write data to and from. These pieces of memory are contiguous. When a program sets up 1000 bytes and then tries to write 1001 bytes into it, it runs over and writes the memory that is allocated for other data. By finding a place in which the program can be caused to overwrite memory, malicious things can be done.

Overwriting the stack pointer is a particularly dangerous area to overwrite because it allows an attacker to redirect the execution of the program. It is very common for a buffer overflow in stack memory to allow the attacker to overwrite the stack pointer and write operation codes to memory. Then when the current function is ended, execution returns to the opcode sent by the attacker.

This is extremely dangerous when the data that is being written is from the network.

Buffer overflows in the listener service

- Example of a connection string
  - (DESCRIPTION=(CONNECT_DATA=(CID=(PROGRAM=)(HOST=)(USER=))(COMMAND=status) (SERVICE=LIST80) (VERSION=135294976)))
- Finding buffer overflows:
  - Try changing this values to see what happens
  - Try USER= with 4,000 Xs after it
  - Try SERVICE= with 4000 Xs after it
  - Etc…

APPLICATION SECURITY, INC.
www.AppSecInc.com

How does a hacker find buffer overflows?

To demonstrate how this is typically done, we look at a common command sent to the listener service. The connection string above is an example.

A hacker would start by trying to send a long string as part of the connection string. For instance, try sending USER= with a large amount of garbage data. If the developer of the listener declared a 1024 bytes buffer on the stack that would receive the username, that buffer will end up overwritten. Programs need to be intelligent enough to see that it only has 1024 bytes allocated and not try to copy more than that.

Buffer overflows in the listener

- Oracle 8.1.7
  - Sending 1 kilobyte of data for COMMAND= caused crash
  - Sending more than 4 kilobytes in the COMMAND= caused core dump
    - Problem in structured-exception handler allows hacker to execute code
- Oracle 9.0.1
  - Sending 1 kilobyte of data for SERVICE=

APPLICATION SECURITY, INC.
www.AppSecInc.com

There have been a number of vulnerabilities found in the listener that are caused by exactly what we just described.

In Oracle 8.1.7, several problems where found with sending large chunks of data for COMMAND=

In Oracle 9 release 1, several problems were found sending SERVICE=

Manipulating header field values

- Typical command
  - .T.....6.,.....:.....4.....(CONNECT_DATA=.)
- Garbage characters represent header information
  - Offset to data
  - Size of connection string
  - Size of packet
  - Type of packet

There have also been vulnerabilities found based on manipulating the header fields of the packets sent to Oracle.

A typical command sent to the listener has several dozen bytes of information about the packet. Included in these fields are data such as:

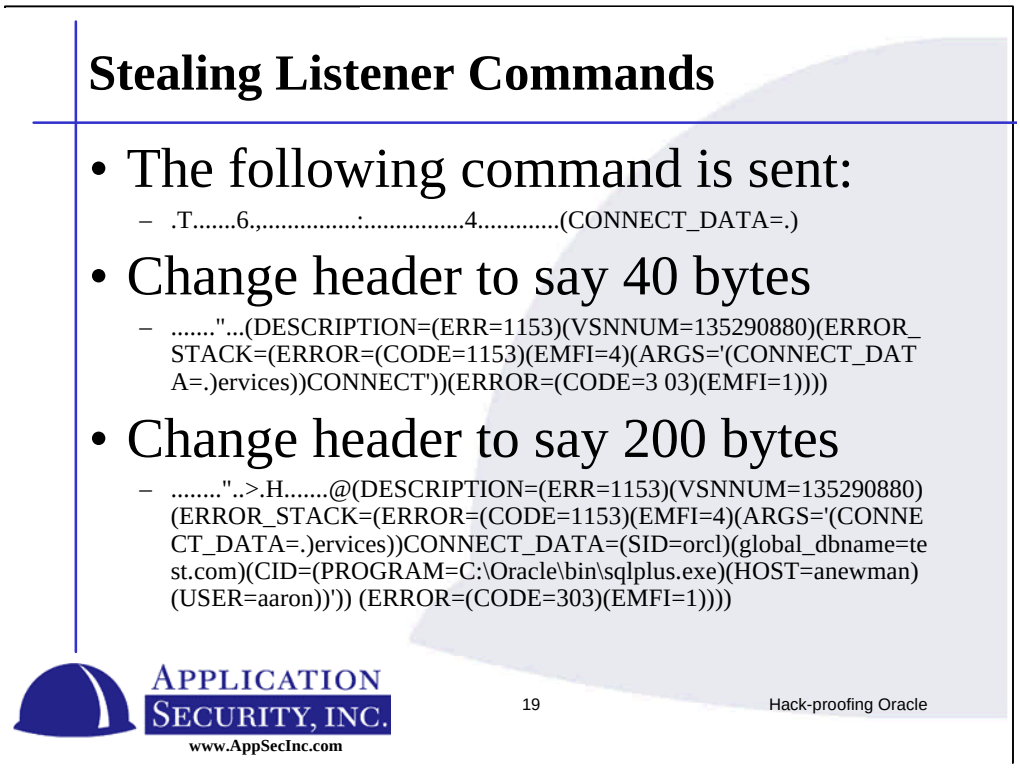
Offset to the connection string

Size of connection string

Size of packet

Type of packet

By manipulating these values, Oracle does weird things. Oracle does not expect these values to be maliciously changed, which makes it vulnerable.



If you fake the size of the data packet you send, the listener will return to you any data in its command buffer up to the size of the buffer you sent. In other words, if the previous command submitted by another user was 100 characters long, and the return value for you command was 10 characters long, the first 10 character will be copied over by the listener, it will not correctly null terminate the return value, and apparently does some kind of string copy, returning to you your 10 character and the last 90 characters of the previous command.

This is useful to an attacker in several ways - it can be used to look for database usernames. If someone else logged into the server using the password, you will be able to retrieve this value from the buffer - of course, that is not easy to do since most people should not be running commands against the listener all that often - but the risk still exists.

External Procedures

- Functions in DLL and shared libraries
- Can be called from PL/SQL
- Setup by creating libraries and packages:
  - CREATE LIBRARY test AS ‘msvcrt,dll’;
    CREATE PACKAGE test_function IS PROCEDURE exec(command IN CHAR);
    CREATE PACKAGE BODY test_function IS PROCEDURE exec(command IN CHAR)
    IS EXTERNAL NAME “system”
    LIBRARY test;

There are also security concerns with external procedure servers. Xprocs allow functions to be created in Oracle that reference DLL or shared libraries on the operating system. This is a powerful feature that make the database able to do anything the OS can do. Of course with great power comes great responsibility.

An Xproc can be created to point to any DLL on the system. This can be used to point to operating system DLLs that allow you to execute operating system commands as if at an operating system prompt. This allows the database to access any resource on the operating system that the database server has access to.

The first concern you should have is that the privilege CREATE LIBRARY and CREATE PROCEDURE gives a user the ability to gain full control of the database and likely the operating system. However this is not the biggest concern.

Remotely calling External Procedures

- Not “officially” support
  - But it works
- ExtProcs are another connection point for listener
  - SID_LIST_LISTENER =
  - (SID_LIST =
  - (SID_DESC =
  - (SID_NAME = PLSExtProc)
  - (ORACLE_HOME = E:\oracle\ora81)
  - (PROGRAM = extproc)
- How does ExtProc authenticate the user
  - IT DOESN’T!!!!!!!!!

APPLICATION SECURITY, INC.
www.AppSecInc.com

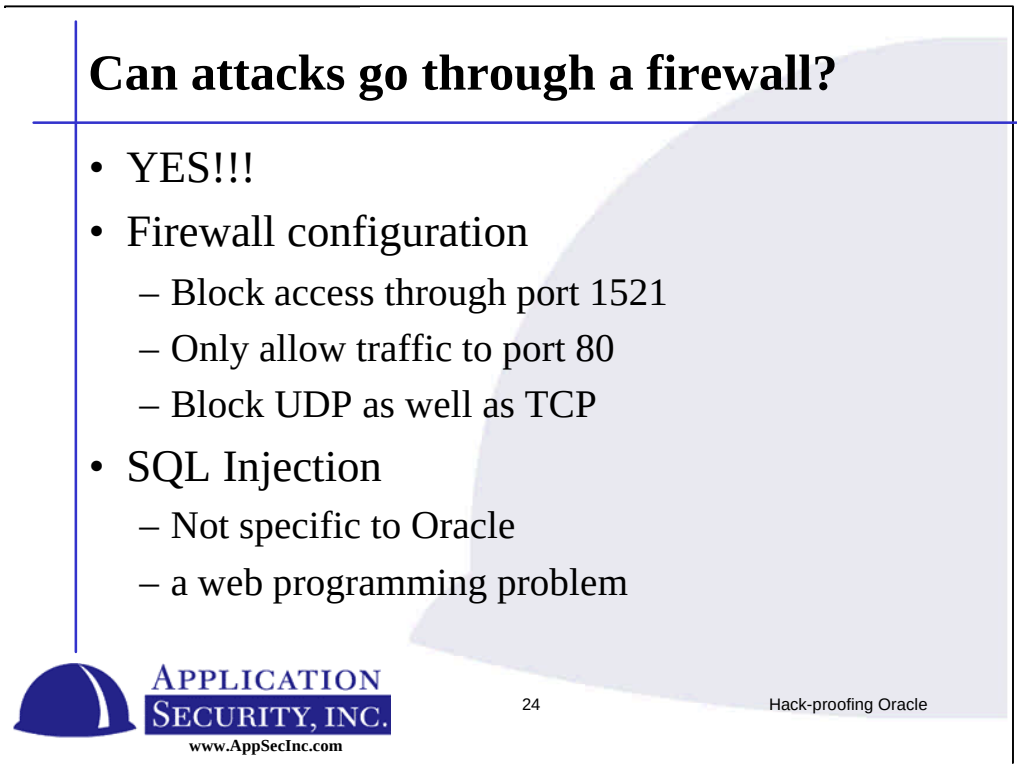
Default setup - External Procedures

- Automatically configured?
  - Oracle 8i – YES
  - Oracle 9i - NO
- How do we fix this?
- Callout listener
  - Do not create ExtProc as another listener endpoint
  - Create its own entry in the listener.ora file
- Can only be called local then

APPLICATION SECURITY, INC.
www.AppSecInc.com

Oracle in a Web application

APPLICATION SECURITY, INC.
www.AppSecInc.com



Most database administrators believe that there database is safe if it is behind a firewall

Even if your firewall is properly configured, attacks can be made through web applications. These attacks are not vendor problems, they are problems caused by bad programming practices on behalf of Application developers.

We've discovered that more sites than not are vulnerable to this problem in one way or another. How this attacks works varies slightly from database to database, but the fundamental problem is the same for all databases.

The simplest way to verify whether you are vulnerable or not is to embed a single quote into each field on each form and verify the results. Some sites will return the error results claiming a syntax error. Some sites will catch the error and not report anything. Of course, these sites are still vulnerable, but they is much harder to exploit if you do not get the feedback from the error messages.

How does it work?

- Modify the query
- Change:
  - Select * from my_table where column_x = '1'
- To:
  - Select * from my_table where column_x = '1' UNION select password from DBA_USERS where 'q'='q'

APPLICATION SECURITY, INC.
www.AppSecInc.com

25

Hack-proofing Oracle

So how does the exploit work? What the exploit does is changes a SQL Statement to another SQL statement. In this example we see a single query being converted into 2 queries. There are also ways to modify the where criteria to update or delete rows not meant to be updated or deleted.

With other databases you can embed a second command into the query. Oracle does not allow you to do this. Instead an attacker would need to figure out how to supplement the end of the query.

Note the 'q' = 'q' at the end. This is used because we must handle the second single quote that the ASP page is adding onto the end of the page. This clause simply evaluates to TRUE.

Example JSP page

```
Package myseverlets;
<….>
String sql = new String(“SELECT * FROM
  WebUsers WHERE Username=’” +
  request.getParameter(“username”) + “’
  AND Password=’” +
  request.getParameter(“password”) + “’”
 stmt = Conn.prepareStatement(sql)
Rs = stmt.executeQuery()
```

Exploiting the problem is much simpler if you can access the source of the web page. You should not be able to see this data, however there are many bugs that allow you to view the source, and I’m sure there are still lots that have not yet been discovered.

The problem with our ASP code is that we are concatenating our SQL statement together without parsing out any single quotes. Parsing out single quotes is a good first step, but its recommended that you actually use parameterized SQL statements instead.

Valid Input

- If I set the username and password to:
  - Username: Bob
  - Password: Hardtoguesspassword
- The sql statement is:
  - SELECT * FROM WebUsers WHERE Username='Bob' AND Password='Hardtoguess'

APPLICATION SECURITY, INC.
www.AppSecInc.com

Here we have the case of a typical authentication mechanism used to login to a web site. You must enter your password and your username. Using these two fields we get a SQL statement that selects from the tables where the username and password match the input. If a match is found, the user is authenticated. If the recordset in our code is empty, then an invalid username or password must have been provided and the login is denied.

Hacker Input

- Instead enter the password:
  - Aa' OR 'A'='A
- The sql statement now becomes:
  - SELECT * FROM WebUsers WHERE Username='Bob' AND Password='Aa' OR 'A'='A'
- The attacker is now in the database!

APPLICATION SECURITY, INC.
www.AppSecInc.com

An attacker instead of using a regular password, enters some letter, uses a single quote to end the string literal, then inserts another boolean expression in the where clause. Obviously this boolean expression is TRUE which returns all the rows in the table.

The kicker is that when the recordset contains the entire set of users, the first one will typically be the Administrator of the system, so there is a good chance you will have full access to the application.

Selecting from other Tables

- To select data other than the rows from the table being selected from.
- UNION the SQL Statement with the DBA_USERS view.

Here's another example of how to pull data back from other tables that are not directly involved in the current query. The best method is to find a screen that contains a dynamic list of items. If the SQL only looks at a single value, then the attacker won't be able to get back all the data requested.

The trick here is to make your single query into 2 queries and union them. This is somewhat difficult because you must match up the number of columns and column types. However, if the server provides you the error messages, the task is very doable. Error will be something like:

Number of columns does not match

Or

2nd column in UNION statement does not match the type of the first statement.

Sample ASP Page

```
Dim sql
Sql = “SELECT * FROM PRODUCT WHERE
  ProductName=’” & product_name & “’”
Set rs = Conn.OpenRecordset(sql)
‘ return the rows to the browser
```

Once again we have the ASP page. An attacker does not really need this, but it does make our lives easier for demonstration purposes. Once again we are not using parameterized queries, but instead are concatenating a string to build our SQL statement.

Valid Input

- Set the product_name to :
  - DVD Player
- The SQL Statement is now:
  - SELECT * FROM PRODUCT WHERE ProductName='DVD Player'

APPLICATION SECURITY, INC.
www.AppSecInc.com

In a typical request we would use the valid name of a product to search for.

The SQL Statement would select from the PRODUCTS table.

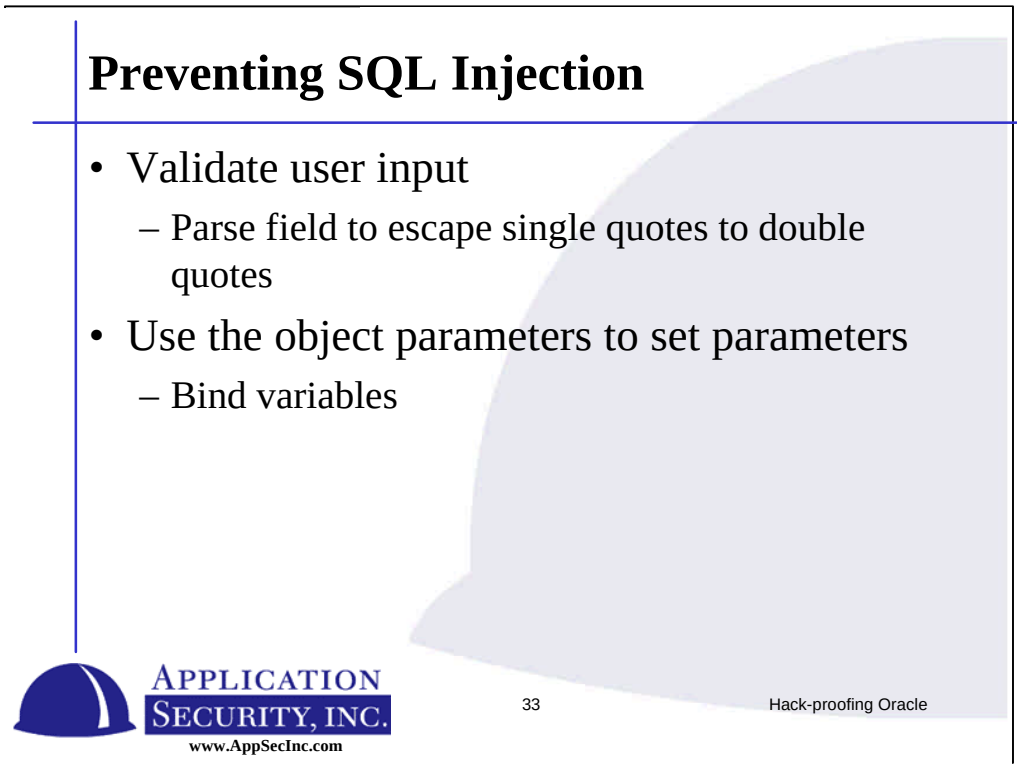
Hacker Input

- Set the product_name to :
  - test' UNION select username, password from dba_users where 'a' = 'a
- The SQL Statement is now:
  - SELECT * FROM PRODUCT WHERE ProductName='test' UNION select username, password from dba_users where 'a'='a'

APPLICATION SECURITY, INC.
www.AppSecInc.com

An attacker would instead want to get a copy of the password hashes from your databases. Once he has these hashes, he can start brute-forcing them.

Instead of entering a single word, the attacker uses a single quote to end the string literal, the adds a UNION command and a second statement. Notice at the end that he must still handle the fact that the code will place another single quote at the end, so we end our second SQL query with 'a'='a. This last clause evaluates to TRUE causing all rows to be returned from the dba_users table.



How do you prevent this.

You are going to need to review and update all you CGI scripts, ASP pages, etc…

Suggest that you setup a programming guideline for web programmers that includes emphasis on using parameterized queries and not concatenating strings for SQL.

Also you can escape single quotes into 2 single quotes although this method is riskier since it is much easier to miss parsing input somewhere.

SQL Injection demo

ASP page, IIS web server

Oracle database

APPLICATION SECURITY, INC.

www.AppSecInc.com

Database Vulnerabilities

APPLICATION SECURITY, INC.
www.AppSecInc.com

Database Security Issues

- sqlnet.log
- Popular Oracle Security Issues
- PL/SQL Vulnerabilities
  - Examples
- Host Operating System
  - Known Issues Installing Oracle
  - Lockdown Protection Procedures

APPLICATION SECURITY, INC.
www.AppSecInc.com

Sqlnet.log

- File is created in a directory when a connection attempt fails from a machine
- Gives too much information – username, IP address, date, etc…
- Have seen many times on public web sites

APPLICATION SECURITY, INC.
www.AppSecInc.com

There is a file that is created when a connection to a database is made. This file is created on the client and records information about failed connections. If you have an Oracle client on your workstation, go search for sqlnet.log.

This file contains the connection string, including username, host connecting to, host connecting from, when the attempt was made, etc… This can be very, very bad. Has anyone every typed there password in the user name field or in the server field. This information ends up recorded in the file.

Make sure you have not accidentally created this file on your web server. Search for this file and delete it. Don't make connections from the web server. I have seen this file on client's web sites.

Popular Oracle Security Issues

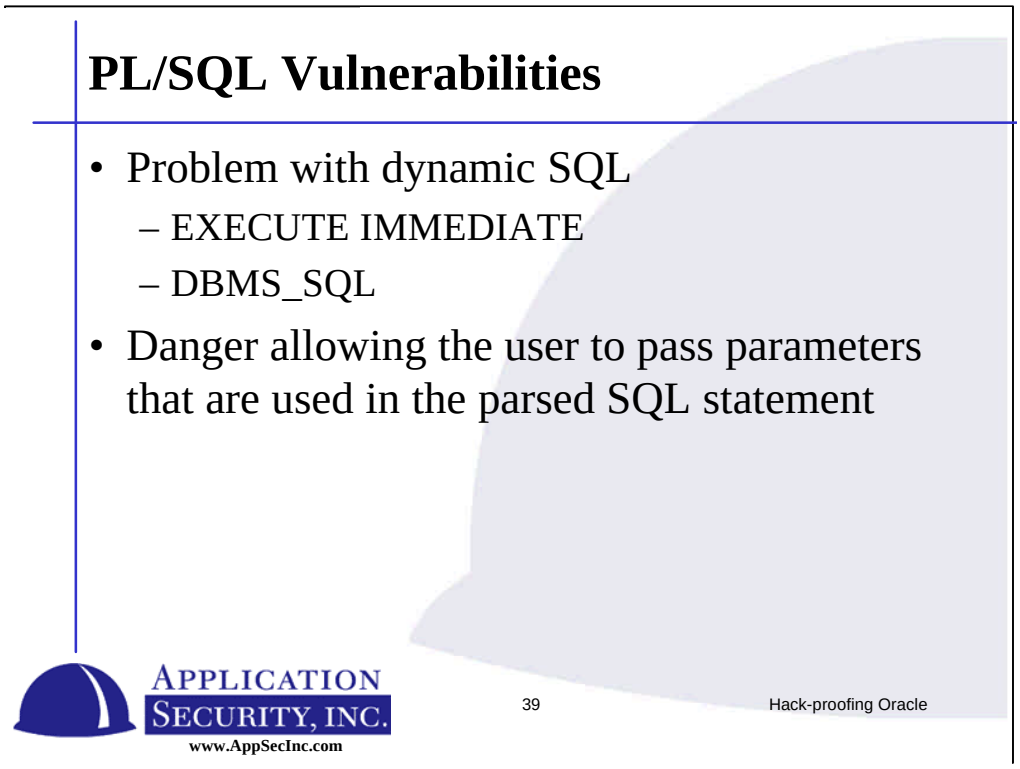
- Default passwords!
  - SYS, SYSTEM, DBSNMP, OUTLN,MDSYS, SCOTT
- Password management features not enabled
  - No password lockout by default
  - No password expiration by default
- Public permissions on ALL_USERS view

APPLICATION SECURITY, INC.
www.AppSecInc.com

Oracle has a variety of default passwords – number is in the dozens. Which ones exist depend on which options you've installed, which version of the database you install, etc… It's very easy to accidentally leave one on. Some are very difficult to change and features will break if changed improperly.

Password management features, such as password lockout, expiration, reuse parameters are implemented through profiles. By default profiles are turned off.

There are several sensitive views that give you lots of information that people don't need to see. The ALL_USERS view is a list of usernames in the database. Gives a non-privileged account a large set of targets to attack.



This issue is almost identical to the SQL Injection problem. You can insert additional statements into the SQL to manipulate or show records you shouldn't be accessing. This is really only an issue when a function runs using OWNER RIGHTS, not when a procedure is executed using INVOKER RIGHTS.

There are two ways to create SQL Statements on the fly in PL/SQL code – Execute immediate and through the package DBMS_SQL. Why would you need to do this – if you did not know the name of the table or columns at compile time or if you wanted to execute DDL or DCL code in a procedure.

Several of the package shipped with Oracle use these statements.

Dynamic SQL Example

```
CREATE PROCEDURE BAD_CODING_EXAMPLE ( NEW_PASSWORD
    VARCHAR2 ) AS
TEST VARCHAR2;
BEGIN
-- DO SOME WORK HERE

EXECUTE IMMEDIATE 'UPDATE ' || TABLE_NAME || ' SET ' ||
    COLUMN_NAME || ' = ''' || NEW_PASSWORD || '''‘ WHERE USERNAME=
    = ''' || CURRENT_USER_NAME || ''';

END BAD_CODING_EXAMPLE;
```

Here is an example of a procedure that allows you to update your username. You wouldn’t want to write your procedures like this.

Just like SQL Injection code, we are concatenating the strings together to create a SQL statement.

Valid input

- Input
  - EXEC BAD_CODING_EXAMPLE(‘testabc’);
- SQL Created
  - UPDATE APPLICATION_USERS SET PASSWORD = ‘testabc’ WHERE USERNAME = ‘aaron’

APPLICATION SECURITY, INC.
www.AppSecInc.com

Here we have an example of what valid input should look like. Here you are updating your password to testabc. This creates a SQL statement that updates the APPLICATION_USERS table for the user aaron.

The input is run from any OCI connection, ODBC connection, SQL*Plus, etc…

Note this does require having a valid account in the database and having execute permissions on the procedure.

Hacker input

- Input
  - EXEC BAD_CODING_EXAMPLE(‘testabc’’, ADMIN=1, FULL_NAME=‘’TEST’);
- SQL Created
  - UPDATE APPLICATION_USERS SET PASSWORD = ‘testabc‘, ADMIN=1, FULL_NAME=‘TEST’ WHERE USERNAME = ‘aaron’

APPLICATION SECURITY, INC.
www.AppSecInc.com

Here we see that an additional clause was added to the input. First the attacker ends the string literal by using 2 single quotes, which PL/SQL translates to a single quote. Then the attacker inserts a comma and a second column to update. Updating the ADMIN column makes this user an administrator for the application.

Notice that we need to tack another column on at the end to get handle the final single quote.

Getting to the operating system

- Oracle on NT typically runs as LocalSystem
  - Act as part of the OS privilege
- Oracle on Unix runs as the oracle user
  - Privilege to all oracle files
- Procedures such as:
  - UTL_FILE, UTL_HTTP
- System privileges such as Create Library

APPLICATION SECURITY, INC.
www.AppSecInc.com

Once an attacker has gained access to the database, they can also get access to the operating system. Not really anyway to prevent them. There are several procedures that can be used – UTL_FILE is the most lethal. Gives you full access to the file I/O.

There are limitations to this feature. UTL_FILE_DIR parameter can be set to limit the directories you can write to. The administrator can of course modify this parameter.

Oracle also allows you to load libraries into a separate process space using the EXTPROC executable file. Can load pretty much any shared library or DLL the oracle user has access to.

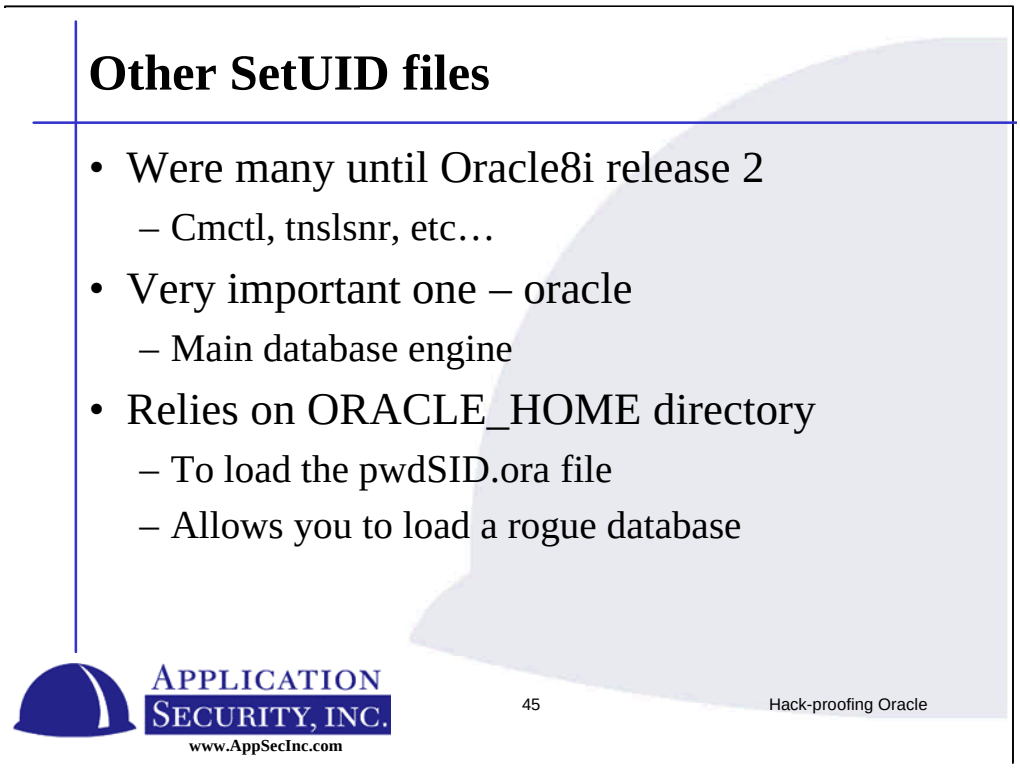
On the operating system

- Oracle has many setUID files
- Oratclsh was setUID root
  - TCL debugger
  - Allowed you to run a script as root
  - Change setuid immediately, even if you are not using

APPLICATION SECURITY, INC.
www.AppSecInc.com

We all know there are problems with files that are Setuid. In the past oracle shipped with 15 files setuid.

The biggest problem was the TCL debugger, designed to test applications before they were used in the DBSNMP engine. This debugger is setuid and is owned by root. Was very easy to exploit this to get full root access. There is a patch, but I'm sure there are other problems with the file that could be exploited. This is a problem even if the dbsnmp agent is not being used.



In release 2 of Oracle 8i (version 8.1.6) most of the SetUID bits have been disabled.

Two remain – on dbsnmp and oracle files. Both of these require the SetUID to work properly. Instead we recommend using different strategies. Remove the execute privilege on both these files from everybody except the owner. Use the owner account to startup the database.

There are other strategies to start the database using groups and SetGUID, but they are fairly complex.

Oracle is the main executable program. If you leave execute permissions on this file, anybody with an account on the server can startup a rogue instance and use the UTL_FILE package to write files. This can occur because Oracle relies on the ORACLE_HOME environment variable.

Dbsnmp is setuid and owned by root. Don't trust it.

Installing Oracle

- Oracle trusts the /tmp directory
- If a file is created before the Oracle file is written, it is overwritten but retains the permissions
- Allows backdoors to be injected into installation

Oracle creates files in the /tmp directory during installation. Although the umask is 022, if someone before hand creates the directories and files that oracle creates during the installation, the old files permissions are retained allowing the attacker to inject code into the installation.

There are many scripts in this installation that makes it much easy to inject code, so an attacker would not have to be sophisticated enough to create a root tool kit.

Lockdown the operating system

- Lock all users out of the OS during installation
- Set the TMP_DIR directory to a secured directory
- Lockdown ORACLE_HOME permissions
- Remove setUID from all files
- Rename the UNIX oracle account

To prevent this injection during an installation, we recommend you take several steps:

-don't allow other accounts to be logged in while the installation is going on (this isn't 100% effective since the attacker can plant something to wait for an installation to start that attacks during the install).

-Better strategy is to set the environment variable TMP_DIR, sometime TEMP_DIR on older versions of Oracle. This causes the temporary files to be written somewhere else. The secure secure this directory before starting the installation.

Rename the unix acccount that owns oracle. It is always oracle, and makes is a target.

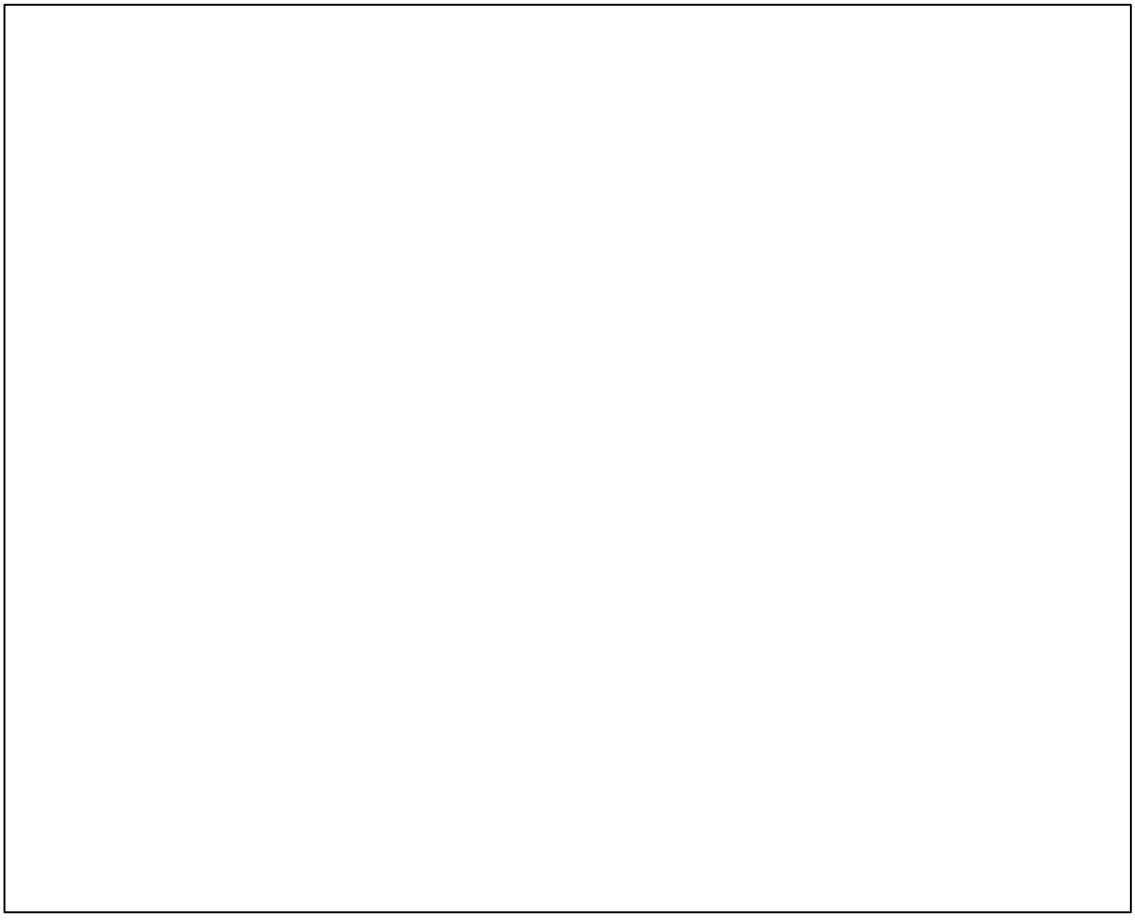
Resources, Conclusion, and Wrap Up

www.AppSecInc.com

How to Combat Hackers

- Stay patched –
  - http://metalink.oracle.com
- Security alerts:
  - www.oraclesecurity.net/resources/mailinglist.html
- Security Discussion Board
  - www.oraclesecurity.net/cgi-bin/ubb/ultimatebb.cgi
- Check out security solutions at:
  - www.appsecinc.com

APPLICATION SECURITY, INC.
www.AppSecInc.com



How to Combat Hackers

- Defense in depth
- Multiple levels of security
  - Perform audits and pen tests on your database on a regular basis
  - Encryption of data-in-motion
  - Encryption of data-at-rest
  - Monitor your log files
  - Implement intrusion detection

APPLICATION SECURITY, INC.
www.AppSecInc.com

Questions?

- About
  - Oracle security features
  - Vulnerabilities
  - Protecting your database
- Email me at:

anewman@appsecinc.com

www.appsecinc.com

APPLICATION SECURITY, INC.
www.AppSecInc.com