PeteFinnigan.com Limited

Oracle Security Expertise

Newsletter Issue 001 – July 2003

Welcome!

Welcome to the very first PeteFinnigan.com Newsletter! Now that the first issue is out I intend to write a newsletter about every couple of months or so. Time permitting of course! So what is it all about? What will these newsletters contain?

Well I want each newsletter to consist of two or three short technical articles (or ramblings about something in particular, personally I like to try things out for myself to see what happens, this is how I learn, I think that sums up the essence of what I want these technical writings to be like) and probably one or two pertinent news items if anything is happening at the time. The news letters will predominantly be Oracle security based as that is what we are interested in here but they could be about other interesting technical / programming issues, Oracle related of course. Although security is my prime interest knowledge of any other interesting area leads to a better understanding of the product and often reveals some new ideas in security!

As this is the first newsletter I thought it might be worthwhile to say a little about myself and my company PeteFinnigan.com limited. Then I want to talk about some interesting snippets on SQL*Plus I have been discussing recently with a colleague and some items I have posted and seen on one of the newsgroups.

So let us get on with it!

Some background

I have been working in the IT industry since early 1995 and prior to that in engineering doing electrical design for a large manufacturer. I hold a first class honours degree in electronic and electrical systems engineering from Leeds metropolitan university in the north of England worked for part time, one day per week and graduating in 1995. I was also awarded the IEE prize for being the highest graded student. I have had many roles in IT including developer, consultant, tester, implementer, all involving Oracle in some way throughout this time.

I have had an interest in security and Oracle internals for many years and over the past two years I have worked to develop security standards for Oracle installations resulting in the publishing of the *Oracle Security Step-by-step – a survival guide for Oracle security* for the SANS Institute. I have written and published many articles in this field including for *security focus, iDefense.com, SANS* Institute, my own website and *TISC*.

I formed PeteFinnigan.com Limited in 2003 to provide my extensive Oracle security knowledge to clients worldwide as part of security audits, forensics work audit configuration and anything Oracle security related. We are based in the North of England near the city of York. I started the company niaeve with the aim of concentrating on oracle and security only. I believe that specialising in this way can benefit those clients who do not have the time to train their staff to be specialists in this area. I also strongly believe in knowledge sharing and this is why my web site although in its infancy hosts a lot of free information on Oracle security and this will continue to be added to as time goes by. I am also regularly writing papers on the subject of Oracle security for various websites.

Enough background!!!

News!

A recent posting on the news site eweek (http://www.eweek.com/article2/0,3959,1120074,00.asp?kc=EWAV10209KTX1K0100440) discussed two interesting new tools to be coming soon from Oracle:

- The first is a tool to manage which patches have been applied and which need to be applied. The tool is a general patch tool not just security patches but Oracle said that security patches should be flagged differently. This is great news as it has always been very difficult to reliably determine exactly which patches have been applied. The info in v$version is not always updated and the installerActions log also is not always reliable. **Let's hope that Oracle also relax the policy on security patches only being available if a support contract is held.**
- The second new product is more interesting and is described as an auto hardening tool. The tool will check for database services used by hackers and warn the admin that they should be turned off. It will also identify configuration issues that could lead to security problems. This tool should be available ***free*** in nine months time.

Protecting SQL*Plus

A good few newsgroup threads have discussed the subject of protecting access via SQL*Plus in the past. Whether it is possible

or not or what can be done to limit access in anyway. This is the holy grail to some but probably pointless as chopping off SQL*Plus doesn't stop TOAD or access using ODBC from MS Excel or OO4O from MS Excel or downloading a java thin client….

A way forward is to limit access to the database to certain IP addresses and to log what users are doing using audit features such as standard audit, system triggers, normal user triggers and fine grained audit. A good security policy limiting privileges of all users to the *least privileges* necessary is also a good step. If this is done then even if an errant user connects using a client SQL*Plus session or from MS Access or whatever, then he should not be able to do any more that his normal application user could do through any provided application.

As quite a few people have been talking about securing and protecting SQL*Plus recently what I want to do now is to explore a few ideas to do with limiting SQL*Plus.

Bypassing the product user profile.

I came across an interesting issue with the product_user_profile table recently. I was asked why a user via SQL*Plus was able to execute an alter user command when the ALTER command had been disabled in the product_user_profile for that user (and all others!). After some digging it became evident that the alter user command was executed in an anonymous PL/SQL block with *execute immediate* within a script.

The reason seemed to be that SQL*Plus itself checks the product user profile against what is being executed and stops the command if it is not allowed but anonymous PL/SQL is sent to the server without parsing it to check for disallowed commands.

The solution is to also block PL/SQL in the product user profile table for the particular user. Here is an example demonstrating this issue:

```
SQL> connect system/manager
Connected.

SQL> select count(*)
  2  from product_profile;

  COUNT(*)
----------
         0

SQL> -- create a user for the demo
SQL> grant create session, alter user to
tester identified by tester;

Grant succeeded.

SQL> connect tester/tester
Connected.
SQL> -- test our alter user privileges on the
user pete
SQL> alter user pete identified by
new_passwd;

User altered.

SQL> -- re-connect as system and stop the
user tester from altering users
SQL> -- via SQL*Plus using product user
profile
SQL> connect system/manager
Connected.
SQL> insert into product_profile
  2  (product,userid,attribute,char_value)
  3  values
('SQL*Plus','TESTER','ALTER','DISABLED');

1 row created.

SQL> commit;

Commit complete.

SQL> -- re-connect as tester and try to alter
user pete's password
SQL> connect tester/tester
Connected.
SQL> alter user pete identified by
another_passwd;
SP2-0544: invalid command: alter
SQL> -- OK, it fails - thats good now try
another way
SQL> begin
  2  execute immediate 'alter user pete
identified by secret';
  3  end;
  4  /

PL/SQL procedure successfully completed.

SQL> -- check it altered the password
SQL> connect pete/secret
Connected.
SQL> -- voila, product_profile has been
defeated!
SQL> -- OK, now try to block this loophole.
SQL> connect system/manager
Connected.
SQL> -- we have to restrict "declare" and
"begin" to stop pl/sql
SQL> insert into product_profile
  2  (product,userid,attribute,char_value)
  3
values('SQL*Plus','TESTER','BEGIN','DISABLED'
);

1 row created.

SQL> insert into product_profile
  2  (product,userid,attribute,char_value)
  3
values('SQL*Plus','TESTER','DECLARE','DISABLE
D');

1 row created.

SQL> commit;

Commit complete.

SQL> -- try again to use dynamic pl/sql
SQL> connect tester/tester
Connected.
SQL> begin
SP2-0544: invalid command: begin
SQL> declare
SP2-0544: invalid command: declare
SQL> -- OK, the loophole is blocked
```

The example demonstrates that a user who has been restricted via SQL*Plus from executing an ALTER command can get around this (see bold code above) by using dynamic PL/SQL. The solution to this is to restrict PL/SQL via the *begin* and *declare* keywords. Problems arise if PL/SQL needs to be used for a particular user as then it is not possible to block the loophole. Before Native Dynamic SQL (NDS) was introduced dynamic SQL had to be done with the package dbms_sql or dbms_sys_sql. The problem could be prevented by revoking access on these packages. Now that NDS is built into the language it is not possible to restrict dynamic SQL and block this type of hole if PL/SQL is required.

Note that this issue potentially applies to any other SQL commands that could be restricted via the product user profile where those commands can be done in dynamic PL/SQL.

It should be noted that protecting SQL*Plus is pretty pointless in this day and age of ODBC, thin Java clients and the whole array of other products that support Oracle connectivity. The other point worth mentioning is those users connecting as SYS, SYSTEM and privileged connections "AS SYSDBA" and "AS SYSOPER" bypass the product user profile functionality altogether.

Renaming SQL*Plus

A while ago someone on the comp.databases.oracle.server newsgroup was talking about restricting the use of SQL*Plus on a client and forcing all users to access the database via their application. One poster suggested checking *v$session* and using a logon trigger. Another poster suggested that this could be easily spoofed by renaming the SQL*Plus binary. I tried an experiment to see if this was true. Here are the results.

First a test from a Windows 98 client to an Oracle 8.1.7 database on Solaris:

```
C:\Oracle\Ora81\BIN>sqlplus pete/pete@plsq
```

```
SQL*Plus: Release 8.1.5.0.0 - Production on
Wed Jul 9 16:15:18 2003

(c) Copyright 1999 Oracle Corporation.  All
rights reserved.


Connected to:
Oracle8i Enterprise Edition Release 8.1.7.0.0
- Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

SQL>
SQL> select username,program,module
  2  from v$session
  3  where username='PETE';

USERNAME                       PROGRAM
------------------------------ --------------
---------------------------------
MODULE
------------------------------------------------
---
PETE
SQL*Plus


SQL>
SQL> EXIT
Disconnected from Oracle8i Enterprise Edition
Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

C:\Oracle\Ora81\BIN>copy sqlplus.exe
hacker.exe
        1 file(s) copied

C:\Oracle\Ora81\BIN>hacker pete/pete@plsq

SQL*Plus: Release 8.1.5.0.0 - Production on
Wed Jul 9 16:19:00 2003

(c) Copyright 1999 Oracle Corporation.  All
rights reserved.


Connected to:
Oracle8i Enterprise Edition Release 8.1.7.0.0
- Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

SQL>
SQL> select username,program,module
  2  from v$session
  3  where username='PETE';

USERNAME                       PROGRAM
------------------------------ --------------
---------------------------------
MODULE
------------------------------------------------
---
PETE
SQL*Plus


SQL>
```

From the client point of view renaming SQL*Plus doesn't seem to be a trivial way to trick *v$session*. Let's try on the server as well:

```
oracle:venus> sqlplus pete/pete@plsq

SQL*Plus: Release 8.1.7.0.0 - Production on
Wed Jul 9 16:34:30 2003

(c) Copyright 2000 Oracle Corporation.  All
rights reserved.


Connected to:
Oracle8i Enterprise Edition Release 8.1.7.0.0
- Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

SQL> select username,program,module
  2  from v$session
  3  where username='PETE';

USERNAME                       PROGRAM
------------------------------ --------------
---------------------------------
MODULE
------------------------------------------------
---
PETE                           sqlplus@venus
(TNS V1-V3)
SQL*Plus


SQL> exit
Disconnected from Oracle8i Enterprise Edition
Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

oracle:venus> cp $ORACLE_HOME/bin/sqlplus
./hacker
oracle:venus> ./hacker pete/pete@plsq

SQL*Plus: Release 8.1.7.0.0 - Production on
Wed Jul 9 16:36:49 2003

(c) Copyright 2000 Oracle Corporation.  All
rights reserved.


Connected to:
Oracle8i Enterprise Edition Release 8.1.7.0.0
- Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

SQL> select username,program,module
  2  from v$session
  3  where username='PETE';

USERNAME                       PROGRAM
------------------------------ --------------
---------------------------------
MODULE
------------------------------------------------
---
PETE                           sqlplus@venus
(TNS V1-V3)
SQL*Plus


SQL>
```

This produces the same result as renaming the SQL*Plus binary on the client.

Therefore using a logon trigger and restricting SQL*Plus based on what is stored in *v$session* does seem like a viable first base solution, it is not foolproof but would stop casual and normal business users.

More than likely the module column is set by *dbms_application_info.set_module* called from the SQL*Plus binary on the users behalf. A quick use of trace and tkprof shows this:

```
BEGIN DBMS_APPLICATION_INFO.SET_MODULE(:1,NULL);
END;


call     count       cpu    elapsed       disk
query    current        rows
------- ------  -------- ---------- ---------- -----
----- ----------  ----------
Parse        3      0.00       0.00          0
0          0           0
Execute      3      0.00       0.00          0
0          0           3
Fetch        0      0.00       0.00          0
0          0           0
------- ------  -------- ---------- ---------- -----
----- ----------  ----------
total        6      0.00       0.00          0
0          0           3
```

```
Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: SYS
```

Of course a more determined user could with skill alter the Oracle networking packets being passed to the server and hide the fact SQL*Plus was being used or possibly alter the strings held in the binary with a hex editor. This would not be trivial as the module would need to be set correctly to *not* SQL*Plus and also the *program* field would need to be changed.

A logon Trigger

A useful protection against SQL*Plus would be to use a *logon trigger* to check for the application being used and then to error if the conditions are not met.

A warning: This is a good solution for stopping casual access with SQL*Plus a clever hacker will beat it. From above we saw that re-naming SQL*Plus will not fool this method easily but if this method were to be applied to a home grown application things would not be the same. Renaming another application to masquarade as your business app may work!

A sample logon trigger is given in the SANS guide (action 5.7.3) for capturing logon info from the client, this can be altered to block a user from logging on if they are using SQL*Plus as follows:

```
create or replace trigger check_logon
  after logon on database
declare
     cursor c_check is
     select
sys_context('userenv','session_user')
username,
             s.module,
             s.program
     from v$session s
     where
sys_context('userenv','sessionid')=s.audsid;
     --
     lv_check c_check%rowtype;
begin
     open c_check;
     fetch c_check into lv_check;
     if lv_check.username in ('SYS','SYSTEM')
then
        null;
     elsif upper(lv_check.module) like
('%SQL*PLUS%') or
              upper(lv_check.program) like
('%SQLPLUS%') then
        close c_check;
        raise_application_error(-
20100,'sqlplus banned');
     end if;
     close c_check;
end;
/
```

After installing this and then trying to connect as a user with SQL*Plus we get:

```
oracle:venus> sqlplus pete/pete@plsq

SQL*Plus: Release 8.1.7.0.0 - Production on
Thu Jul 10 20:30:12 2003

(c) Copyright 2000 Oracle Corporation.  All
rights reserved.

ERROR:
ORA-00604: error occurred at recursive SQL
level 1
ORA-20100: sqlplus banned
ORA-06512: at line 20
```

Delete the PUP tables?

Geoff Ingram in his book *High Performance Oracle* offers a solution to the problem of stopping access by SQL*Plus or MS Access that involves moving the audit table aud$ into the system schema and adding a trigger onto it to capture -942 errors. These are generated because product_privs has been removed. SQL*Plus queries this table on start-up. The idea is that SQL*Plus will generate an error in the audit log and then the trigger will fire and send an alert to a waiting daemon that will kill the process.

Whilst the idea is good there are some flaws. The first is that Oracle will not support the moving of the AUD$ table to another schema. The second is that this system could easily be broken by someone in the future inadvertently installing the pup tables and views again. The other issue is as stated above that these tables are not queried by privileged connections so this method would not capture all accesses anyway.

Other Ideas

Another option, is to use role based security to stop users logging on with applications such as SQL*Plus. Strangely as I write this a newsgroup thread has just started where one of the posters (Jacques Kilchoer) suggests a solution based on this idea.

The idea is to grant only create session to all the application users and then grant a role that allows them access to do the tasks they need but this role is not a default role and is protected by a password. The application on start-up connects and submits a *set role* command and uses the password to enable the role. There are a number of ways to try to protect the password. Some application writers obfuscate the password, some encrypt it (as Jacques suggests) and some read it from a secure file or table in the database. Because the user doesn't know the role password she cannot just logon with SQL*Plus.

There are flaws with this method, the first is that any encryption or obfuscation scheme if broken will render the security useless, the second is that the password can be easily read from the wire with tools such as *snoop* or by using SQL*Net trace set to SUPPORT level to capture packet contents. An application user can set client SQL*Net trace and login and then find the password from the trace file.

Again this type of scheme would not stop a determined hacker or malicious user.

Wrapping up on SQL*Plus

In summary I believe that trying protect SQL*Plus whilst useful to stop casual user connections with this tool rather than your application is somewhat futile considering the multitude of other methods of connecting to the database.

From what we have seen with a number of ideas and issues with blocking SQL*Plus it is very difficult to keep out just this tool. One further issue is the plethora of Oracle client CD's in existance. It would be easy for a casual user to obtain one and install an Oracle client for herself!! – happy hacking!!

How to subscribe and unsubscribe

To subscribe to the PeteFinnigan.com Limited newsletter simply send a blank email to news@petefinnigan.com to un-subscribe send a blank email to nonews@petefinnigan.com.

Company number 4664901

Phone: 0044 (0) 1757 701840
Fax: 0044 (0) 1757 701840
Email: sales@petefinnigan.com
Web Site: http://www.petefinnigan.com

Registered Office
PeteFinnigan.com Limited
3 Rowan Close
North Yorkshire
YO8 9FJ
England