The next level of Oracle attacks

vonJeek
vonjeek@thc.org

RevMoon *technical editor*
djrevmoon@thc.org

March 25, 2007

Abstract

This paper focuses on new threats on one of the most popular database platforms: Oracle. On 18 October 2005, the SANS institute published *An Assessment of the Oracle Password Hashing Algorithm*[1]. An implementation of the algorithm as described in this document enables attackers to crack passwords. With a cracked password, an attacker can gain unauthorized access to corporate data, or perform unauthorized transactions. But how does it work? And what is the impact of newly discovered weaknesses? The first section of this document describes how Oracle database products generate password hashes[2] which are stored in the database to authenticate users. The second section of this paper demonstrates that a design flaw in the implementation of a key exchange enables an attack on Oracle network authentication and recovery of passwords used in logins over the network.

Contents

1 Attacking the Oracle password hashing algorithm

On 18 October 18 2005, the SANS institute published *An Assessment of the Oracle Password Hashing Algorithm*. This SANS paper describes how Oracle database products generate password hashes which are stored in the database to authenticate users. An implementation of the algorithm enables attackers to crack passwords. With a cracked password, an attacker can gain unauthorized access to corporate data. This chapter focuses on the process and implementation of cracking Oracle password hashes.

1.1 Getting the hashes

In order to crack the hashes, a list containing the username and encrypted passwords must be retrieved from the database. To do so, an attacker can follow different paths, such as:

- Retrieve hashes using authorized access. A personal account of a user or functional application account might allow accessing the table containing the required data if authorizations are not strict enough. This path is typically followed by curious employees or inside attackers.
- Retrieve hashes using unauthorized access. Oracle and products using Oracle databases feature a number of default and well-known username/password combinations[3]. By trying to login using this information or guessing a valid username/password combination, access might be obtained. In an sub-optimally secured environment, such access frequently has the necessary privileges to access the account names and password hashes. Also, access to account names and password hashes might be gained by abusing security flaws in Oracle itself. This category of attack is typically performed by hackers.
- Retrieve hashes from the application layer. By utilising attacks such as SQL-injection, database access can be obtained in some cases. This category of attack is often used by internet hackers.

Extraction techniques fall outside the scope of this paper.

1.2 Cracking hashes

When the account names and password hashes are successfully retrieved, a password cracking attack can be started. The process of cracking is visualized in figure 1, and is a standard dictionary or brute-force attack.

1.3 Oracles password hashing

Oracle hashes are generated using a known, fixed magic number with the value of 0x0123456789ABCDEF, the username, the password, a PREPARE-function

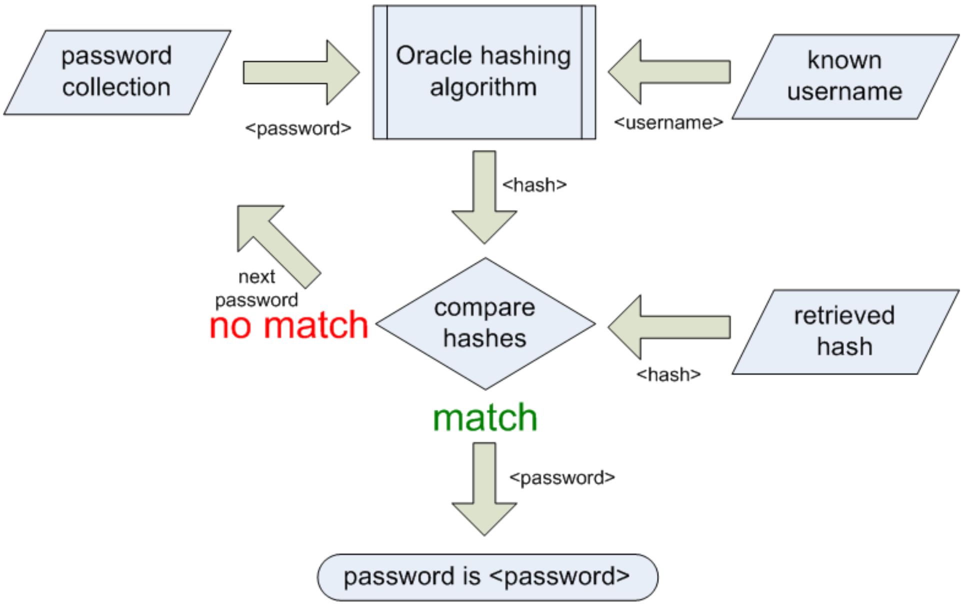

Figure 1: cracking passwords

and the Data Encryption Standard (DES)[4] algorithm in Cypher Block Chaining (CBC)[5] mode.

```
// prepare data
in = PREPARE(username, password)

// first DES CBC
key = DES_CBC_ENCRYPT(in, magic_number)

// second DES CBC
hash = DES_CBC_ENCRYPT(in, key)
```

Assume that retrieval of the hash of the account SYS is successful:

```
SYS C648972D2BE43FA4
```

And the value H4X0R is used as the assumed candidate password, the following input data is used:

```
magic_number = 0x0123456789ABCDEF
username = SYS
password = H4X0R
```

First, the PREPARE-function converts all characters to uppercase and concatenates the username and the candidate password. The value of the output string of PREPARE is SYSH4X0R, which is 0x5359534834583052 in ASCII[6]. Then the function stores all characters using 2 bytes per character, zeroing the high bytes; 0x53 becomes 0x0053, 0x59 becomes 0x0059 et cetera. Furthermore, the input is extended to a multiple of 8 bytes, padded with zeros if needed. In this case the input string size is 16 bytes so nothing will happen. The complete operation:

```
// prepare data
in[] = PREPARE(username, password)
     = {0x0053005900530048, 0x003400580300052}
```

DES in CBC mode actually XORs[7] the input of block N with the output of block $N-1$. The example calculation shows this explicit form:

```
// first DES CBC
temp = DES_ENCODE(in[0], magic_number)
     = DES_ENCODE(0x0053005900530048, 0x0123456789ABCDEF)
      = 0x170453E89F8CDA7
in[1] = temp XOR in[1]
      = 0x170453E89F8CDA7 XOR 0x0034005800300052
      = 0x48BB68C2B4C18FD0
key = DES_ENCODE(in[1], magic_number)
    = DES_ENCODE(0x48BB68C2B4C18FD0, 0x0123456789ABCDEF)
    = 0x6B539939C572D9AC

// second DES CBC
temp = DES_ENCODE(in[0], key)
     = DES_ENCODE(0x0053005900530048, 0x6B539939C572D9AC)
     =  0x6517F03B233E4991
in[1] = temp XOR in[1]
      = 0x6517F03B233E4991 XOR 0x0034005800300052
      = 0x6523F063230E49C3
hash = DES_ENCODE(in[1], key)
     = DES_ENCODE(0x6523F063230E49C3, 0x6B539939C572D9AC)
     = 0xC648972D2BE43FA4
```

To see if the password is cracked successfully, the retrieved hash is compared to the hash calculated. Both have the value of 0xC648972D2BE43FA4 indicating a successful guess: the password of SYS is H4X0R.

1.4 Limitations

What happens if this attack is launched in a perfect world? In the perfect world all systems - database and host operating system - are fully patched. No known exploits[8] work. All users have only got the privileges they really need. Unauthorized hash retrieval is not possible. Default username/password combinations do not exist anymore since passwords are changed or accounts are disabled. Therefore unauthorized access is not possible. In a prefect world, if a database administrator selects a view or table containing the hashes[1], it will be recorded in the audit trail and disciplinary action can be taken. In such an environment, the conclusion is that an attacker will not be able to obtain the encrypted passwords. No reference material means no cracking therefore no risk. End of story? No, not at all.

2 Introducing a passive attack

As seen in the previous chapter, an attacker can actively retrieve Oracle password hashes to crack them. In a perfect world access to this information is restricted to prevent an attacker launching an attack. This section focuses on the process

---

[1] DBA_USERS, SYS.USER$

and implementation of network logins of Oracle database clients. What data is transmitted during a network logon? Can this data - which can be retrieved in a passive way - be abused to crack passwords?

2.1 Documentation study

If retrieving hashes from the database directly is not an option, what can be done to obtain passwords? Where are the hashes used besides in the server side login process? In the *Advanced Security Administrator's Guide*[9], Oracle states (fair use quote):

> *The purpose of Authentication Key Fold-in is to defeat a possible third party attack (historically called the man-in-the-middle attack) on the Diffie-Hellman key negotiation. It strengthens the session key significantly by combining a shared secret, known only to the client and the server, with the original session key negotiated by Diffie-Hellman. The client and the server begin communicating using the session key generated by Diffie-Hellman. When the client authenticates to the server, they establish a shared secret that is only known to both parties. Oracle Advanced Security combines the shared secret and the Diffie-Hellman session key to generate a stronger session key designed to defeat a man-in-the-middle attack.*

So there's got to be a key exchange process in place, using a shared secret to encrypt a session key. To identify the key exchange, sample data is required.

2.2 Capturing network traffic

Using a network sniffer - a piece of software designed for capturing and analysis of the packets going through the network - , a login session of user SYS with password H4X0R is captured. The client uses the standard Oracle client driver and connects to a default installation of Oracle 8i running on the Microsoft Windows platform[2]:

```
PC CLIENT>>
00000348  00 bb 00 00 06 04 00 00  00 00 03 76 02 30 74 3e ........ ...v.0t>
00000358  04 03 00 00 00 01 00 00  00 00 da 13 00 04 00 00 ........ ........
00000368  00 d0 d7 13 00 c0 db 13  00 03 73 79 73 0d 00 00 ........ ..sys...
00000378  00 0d 41 55 54 48 5f 54  45 52 4d 49 4e 41 4c 07 ..AUTH_T ERMINAL.
00000388  00 00 00 07 56 4f 4e 4a  45 45 4b 00 00 00 00 0f ....VONJ EEK.....
00000398  00 00 00 0f 41 55 54 48  5f 50 52 4f 47 52 41 4d ....AUTH _PROGRAM
000003A8  5f 4e 4d 08 00 00 00 08  54 4f 41 44 2e 65 78 65 _NM..... TOAD.exe
000003B8  00 00 00 00 0c 00 00 00  0c 41 55 54 48 5f 4d 41 ........ .AUTH_MA
000003C8  43 48 49 4e 45 0e 00 00  00 0e 4c 4f 43 41 4c 5c CHINE... ..LOCAL\
000003D8  56 4f 4e 4a 45 45 4b 00  00 00 00 00 08 00 00 00 VONJEEK. ........
000003E8  08 41 55 54 48 5f 50 49  44 09 00 00 00 09 31 34 .AUTH_PI D.....14
000003F8  34 30 3a 31 39 36 34 00  00 00 00                40:1964. ...

ORACLE SERVER<<
```

---

[2] login session to other Oracle database server version, client software and/or platforms might show other data

```
00000245  00 91 00 00 06 00 00 00  00 00 08 01 00 0c 00 00 ........ ........
00000255  00 0c 41 55 54 48 5f 53  45 53 53 4b 45 59 10 00 ..AUTH_S ESSKEY..
00000265  00 00 10 34 33 39 32 31  34 33 42 30 38 30 37 39 ...43921 43B08079
00000275  33 35 44 00 00 00 00 04  01 00 00 00 00 00 00 00 35D..... ........
00000285  00 00 00 00 00 00 00 00  00 40 00 00 00 00 00 00 ........ .@......
00000295  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
000002A5  00 00 02 00 00 00 00 00  00 36 01 00 00 90 d1 1c ........ .6......
000002B5  00 e8 15 1d 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
000002C5  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
000002D5  00                                               .

PC CLIENT>>
00000403  02 1e 00 00 06 04 00 00  00 00 03 73 03 30 74 3e ........ ...s.0t>
00000413  04 03 00 00 00 01 01 00  00 1c e9 13 00 07 00 00 ........ ........
00000423  00 d4 e5 13 00 60 eb 13  00 03 73 79 73 0d 00 00 .....`.. ..sys...
00000433  00 0d 41 55 54 48 5f 50  41 53 53 57 4f 52 44 11 ..AUTH_P ASSWORD.
00000443  00 00 00 11 43 41 44 43  46 46 38 42 35 31 41 45 ....CADC FF8B51AE
00000453  35 41 31 37 33 00 00 00  00 0d 00 00 00 0d 41 55 5A173... ......AU
00000463  54 48 5f 54 45 52 4d 49  4e 41 4c 07 00 00 00 07 TH_TERMI NAL.....
00000473  56 4f 4e 4a 45 45 4b 00  00 00 00 0f 00 00 00 0f VONJEEK. ........
00000483  41 55 54 48 5f 50 52 4f  47 52 41 4d 5f 4e 4d 08 AUTH_PRO GRAM_NM.
00000493  00 00 00 08 54 4f 41 44  2e 65 78 65 00 00 00 00 ....TOAD .exe....
000004A3  0c 00 00 00 0c 41 55 54  48 5f 4d 41 43 48 49 4e .....AUT H_MACHIN
000004B3  45 0e 00 00 00 0e 4c 4f  43 41 4c 5c 56 4f 4e 4a E.....LO CAL\VONJ
000004C3  45 45 4b 00 00 00 00 00  08 00 00 00 08 41 55 54 EEK..... .....AUT
000004D3  48 5f 50 49 44 09 00 00  00 09 31 34 34 30 3a 31 H_PID... ..1440:1
000004E3  39 36 34 00 00 00 00 08  00 00 00 08 41 55 54 48 964..... ....AUTH
000004F3  5f 41 43 4c 04 00 00 00  04 34 34 30 30 00 00 00 _ACL.... .4400...
00000503  00 12 00 00 00 12 41 55  54 48 5f 41 4c 54 45 52 ......AU TH_ALTER
00000513  5f 53 45 53 53 49 4f 4e  f8 00 00 00 fe 40 41 4c _SESSION .....@AL
00000523  54 45 52 20 53 45 53 53  49 4f 4e 20 53 45 54 20 TER SESS ION SET 
00000533  4e 4c 53 5f 4c 41 4e 47  55 41 47 45 3d 20 27 41 NLS_LANG UAGE= ’A
00000543  4d 45 52 49 43 41 4e 27  20 4e 4c 53 5f 54 45 52 MERICAN’  NLS_TER
00000553  52 49 54 4f 52 59 3d 20  27 41 4d 45 52 49 40 43 RITORY=  ’AMERI@C
00000563  41 27 20 4e 4c 53 5f 43  55 52 52 45 4e 43 59 3d A’ NLS_C URRENCY=
00000573  20 27 24 27 20 4e 4c 53  5f 49 53 4f 5f 43 55 52  ’$’ NLS _ISO_CUR
00000583  52 45 4e 43 59 3d 20 27  41 4d 45 52 49 43 41 27 RENCY= ’ AMERICA’
00000593  20 4e 4c 53 5f 4e 55 4d  45 52 49 43 5f 43 48 40  NLS_NUM ERIC_CH@
000005A3  41 52 41 43 54 45 52 53  3d 20 27 2e 2c 27 20 4e ARACTERS = ’.,’ N
000005B3  4c 53 5f 43 41 4c 45 4e  44 41 52 3d 20 27 47 52 LS_CALEN DAR= ’GR
000005C3  45 47 4f 52 49 41 4e 27  20 4e 4c 53 5f 44 41 54 EGORIAN’  NLS_DAT
000005D3  45 5f 46 4f 52 4d 41 54  3d 20 27 44 44 2d 4d 4f E_FORMAT = ’DD-MO
000005E3  38 4e 2d 52 52 27 20 4e  4c 53 5f 44 41 54 45 5f 8N-RR’ N LS_DATE_
000005F3  4c 41 4e 47 55 41 47 45  3d 20 27 41 4d 45 52 49 LANGUAGE = ’AMERI
00000603  43 41 4e 27 20 20 4e 4c  53 5f 53 4f 52 54 3d 20 CAN’  NL S_SORT= 
00000613  27 42 49 4e 41 52 59 27  00 00 00 00 00 00       ’BINARY’ ......
```

2.3 Key exchange: the information flow

The complete identification and authentication process from client to server takes places in the following steps:

PC client:

- Sends username as client’s public value. The username can be seen at offset 00000372-00000374 and has the value SYS.

Oracle server:

- Looks up username’s private value.

- Generates a session key.
- Sends session key encrypted with username's private value. The session key encrypted with SYS's private value is called AUTH_SESSKEY and has got the value 4392143B0807935D (offset 00000257-00000277).

PC client:

- Calculates the session key by decrypting the value of the encrypted session key using the username's private value.

Now, both the server and the client know the value of the session key and can encrypt communication using (derivations of) this value. The first thing the client does is sending the password, encrypted with the session key. This variable is called AUTH_PASSWORD and has the value CADCFF8B51AE5A17 (offset 00000435-00000456).

2.4 Key exchange: assumptions made

Since key values are identified, assumptions about the algorithm used for exchanging the session key and the password can be made. Also, an assumption can be made concerning the keys used.

1. Key to exchange the session key: using a username and a password, the Oracle password hash is an obvious candidate; the Oracle server knows the value, the client can calculate the value for any given username/password combination.
2. Algorithm: since 8 byte - 64 bits - values are used, an obvious candidate for the algorithm is DES.
3. Key to exchange password: Oracle stated that secret information is exchanged *by combining a shared secret, known only to the client and the server*. Therefore, the assumption is made that the session key - the decrypted value of AUTH_SESSKEY - is used for encrypting passwords.

2.5 Key exchange: verification

We will verify these assumptions by calculating the session key followed by calculating the password:

```
// 1, CLIENT SIDE CALCULATION
HASH = ORACLEHASH(USERNAME, PASSWORD)
// 2, CLIENT SIDE CALCULATION
SESSION = DES_DECRYPT(SESSION_ENCRYPTED, HASH)
// 3, CLIENT SIDE CALCULATION
GUESSED_PASSWORD = DES_DECRYPT(PASSWORD_ENCRYPTED, SESSION)
```

If the value of GUESSED_PASSWORD equals PASSWORD the guess is successful. To verify the assumption, the following input data is used:

```
USERNAME = SYS
PASSWORD = H4X0R
SESSION_ENCRYPTED = 0x4392143B0807935D (= AUTH_SESSKEY)
PASSWORD_ENCRYPTED = 0xCADCFF8B51AE5A17 (= AUTH_PASSWORD)
```

The calculation:

```
HASH = ORACLEHASH(SYS, H4X0R)
     = 0xC648972D2BE43FA4
SESSION = DES_DECRYPT(0x4392143B0807935D, 0xC648972D2BE43FA4)
        = 0xF06BBCAE024A2B2B
GUESSED_PASSWORD = DES_DECRYPT(CADCFF8B51AE5A17, 0xF06BBCAE024A2B2B)
                 = 0x4834583052000000
```

The result, GUESSED_PASSWORD, is padded with zeros. To get the actual password, all trailing zeros can be dropped resulting in 0x48, 0x34, 0x58, 0x30, 0x52. Converted to ASCII, the value of the result is H4X0R: the guess is successful.

Given the 64-bit limit of the session key, we investigate what will happen if an encrypted password with length $N > 8$ is sent over the network. This data obviously does not fit in an 8 character (64-bit) array. Observations show that the process stays in place: the only difference is the length of the encrypted password data which is transmitted over the network. This length will be a multiple of eight characters. A logon using user SYS and password H4X0RH4X0R shows the following values:

```
USERNAME = SYS
PASSWORD = H4X0RH4X0R
SESSION_ENCRYPTED = 0x64BAFAB43AD56EE5 (= AUTH_SESSKEY)
PASSWORD_ENCRYPTED[] = {0x0D41AD693A7B92D5, 0x6B0CCA9485935942} (= AUTH_PASSWORD)
```

Analysis shows that $PASSWORD[N]$ is XORed with $PASSWORD\_ENCRYPTED[N-1]$ where $N > 0$. Another calculation:

```
HASH = ORACLEHASH(SYS, H4X0RH4X0R)
     = 0x11FBDF0625C06252
SESSION = DES_DECRYPT(0x64BAFAB43AD56EE5, 0x11FBDF0625C06252)
        = 0xDA688F9F780AF080
GUESSED_PASSWORD[0] = DES_DECRYPT(PASSWORD_ENCRYPTED[0], SESSION)
                    = DES_DECRYPT(0x0D41AD693A7B92D5, 0xDA688F9F780AF080)
                    = 0x4834583052483458 (in ASCII: "H4X0RH4X")
// an attacker can verify the guess here already!

GUESSED_PASSWORD[1] = DES_DECRYPT(PASSWORD_ENCRYPTED[1], SESSION)
                    = DES_DECRYPT(0x6B0CCA9485935942, 0xDA688F9F780AF080)
                    = 0x3D33AD693A7B92D5

// the XOR
GUESSED_PASSWORD[1] = GUESSED_PASSWORD[1] XOR PASSWORD_ENCRYPTED[0]
                    = 0x3D33AD693A7B92D5 XOR 0x0D41AD693A7B92D5
                    = 0x3052000000000000 (in ASCII: "0R")

GUESSED_PASSWORD = GUESSED_PASSWORD[0] + GUESSED_PASSWORD[1]
                 = 0x4834583052483458 + 0x3052000000000000

ASCII(GUESSED_PASSWORD) = H4X0RH4X0R
```

2.6 Attacking: theory

The assumption made earlier appears to be correct. Before verification, the keyspace of the session key SESSION had the fixed size of 2^{64}. Using the knowledge of the mechanism used, complexity can be reduced in a number of cases. Let's take another look at the algorithm:

```
HASH = ORACLEHASH(USERNAME, PASSWORD)
SESSION = DES_DECRYPT(SESSION_ENCRYPTED, HASH)
PASSWORD = DES_DECRYPT(PASSWORD_ENCRYPTED, SESSION)
```

The weakness of the mechanism used is the implementation of the shared secrets. the value of private key is predictable since the the mechanism of generating a hash is available in the public domain. As a result, the keyspace can be reduced from a fixed value of 2^{64} to a value directly related to the password length:

- The keyspace of all passwords generated from character set C with length N is C^N which is variable;
- Using the Oracle password hashing algorithm, this will result in C^N password hashes;
- Using these hashes, all candidate session keys for this keyspace can be calculated. The result will consist of C^N candidate session keys.

If the length of the password used is $\leq N$, one of the candidate session keys will match the real session key. So, as long as $C^N < 2^{64}$ the keyspace is reduced. Specifically, a small value of N reduces the keyspace significantly.

Field observations show that most Oracle accounts use the basic Oracle character set for composing passwords. This set consists of [A-Z][0-9]#$_ where a password must start with an alpha character, resulting in a 26 out of 39 ratio for valid passwords ($\frac{2}{3}$). Therefore, the keyspace for password length N = $\frac{2}{3} \cdot 39^N$. This keyspace is smaller than 2^{64} until $N = 64 \cdot \log 2^{39} = 12$. So using these parameters, the attack is superior as long as $N \leq 12$. A illustration visualizing this information van be found in figure 2.

The curve on other Oracle key foldins is expected to show similar characteristics; only the brute force key space will presumably be 2^{128} keeping the this attack superior until $N = 128 \cdot log 2^{39} = 24$.

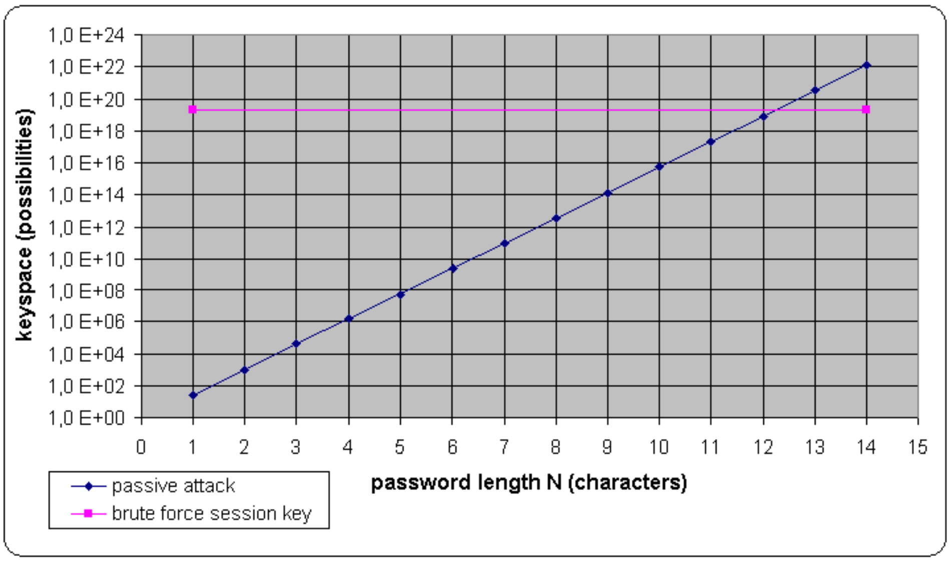

Figure 2: keyspaces

2.7 Attacking: in the field

To perform a successful attack, the values of USERNAME, SESSION_ENCRYPTED and PASSWORD_ENCRYPTED must be sniffed first. Acquiring this data is possible everywhere on the path in between the client and the database server:

Points of attack from left to right. Required information might be tapped:

- In a client's subnet by sniffing directly in a shared, or unsecured wireless environment (inside or outside attacker).
- In a client's subnet by an ARP poisoning attack[11] - using for e.g. Cain[12] - by any device in the same OSI layer 2 domain[13] (every device connected to the network).
- At the local router by for example monitoring the router's uplink or tunneling traffic to other places[14] by a person capable of configuring network equipment (local IT administrator/hacker).
- At an ISP router, for instance by monitoring the router's uplink or tunneling traffic to other places by a person capable of configuring network equipment (ISP administrator/hacker).
- At an ASP router, for example by monitoring the router's uplink or tunneling traffic to other places by a person capable of configuring network equipment (ASP administrator/hacker).

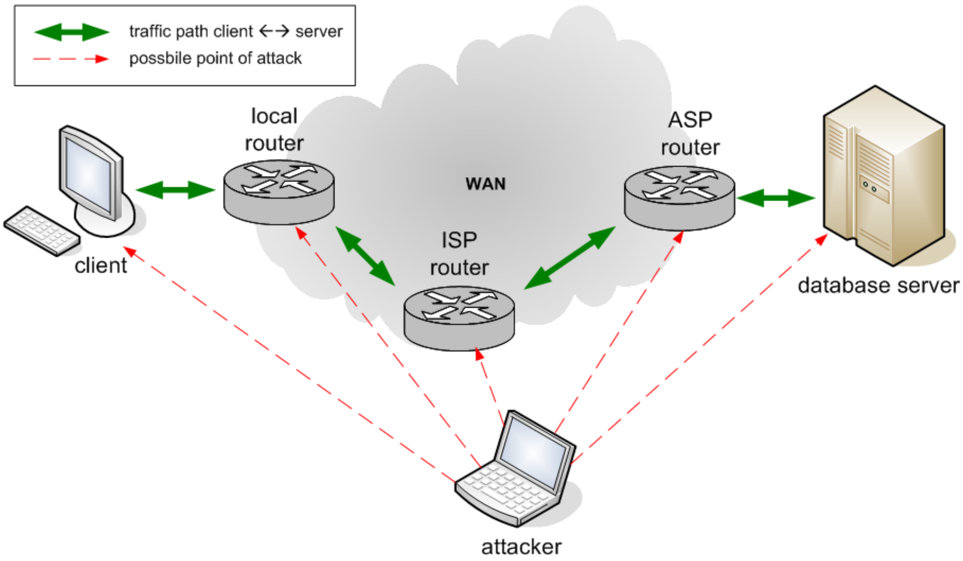

Figure 3: points of attack for sniffing data

- In the server's subnet by an ARP poisoning attack any device in the same OSI layer 2 domain (every device connected to the network).

Using this information, is it feasible that an attacker can gain unauthoprised access to the database and the corporate data?

2.8 Proof of concept

To check the feasibility of this attack, a proof of concept application was developed. Running on a Pentium 4 - 3.2 GHz PC, the non-optimized application can test over 500.000 passwords per second. Table 1 shows the time required for cracking passwords with length N using the basic Oracle character set and by trying all possible session keys. For probability P, value 0.5 is set (meaning that fully random passwords are used):

For a password with length $N = 8$, the size of the keyspace is $3.57 \cdot 10^{12}$. This is more than 5 million times smaller than the original 2^{64}. It is likely that cracking a random password will take about 41 days. The use of non-random passwords - like words from the dictionary or permutations of dictionary words - might speed up the process even more; the keyspace of the entire *Oxford English Dictionary*[15] is 'just' about $5.9 \cdot 10^7$ which, at 500.000 tries/second, can be fully checked within 2 minutes. Using a cluster of 50 PCs, cracking of any password with length $N = 8$ can be done within a day.

N	keyspace ($\frac{2}{3} \cdot 39^N$)	$P = 0.5$	hours	days
1	$2.60 \cdot 10^1$	$1.30 \cdot 10^1$	0.00	0.00
2	$1.01 \cdot 10^2$	$5.07 \cdot 10^2$	0.00	0.00
3	$3.95 \cdot 10^4$	$1.98 \cdot 10^4$	0.00	0.00
4	$1.54 \cdot 10^5$	$7.71 \cdot 10^5$	0.00	0.00
5	$6.01 \cdot 10^7$	$3.01 \cdot 10^7$	0.02	0.00
6	$2.35 \cdot 10^9$	$1.17 \cdot 10^9$	0.65	0.03
7	$9.15 \cdot 10^{10}$	$4.57 \cdot 10^{10}$	25.41	1.06
8	$3.57 \cdot 10^{12}$	$1.78 \cdot 10^{12}$	991.11	41.30
9	$1.39 \cdot 10^{13}$	$6.96 \cdot 10^{13}$	38653.40	1610.56
10	$5.43 \cdot 10^{15}$	$2.71 \cdot 10^{15}$	1507482.61	62811.78
11	$2.12 \cdot 10^{17}$	$1.06 \cdot 10^{17}$	58791821.73	2449659.24
12	$8.25 \cdot 10^{18}$	$4.13 \cdot 10^{18}$	2292881047.33	95536710.31
13	$3.22 \cdot 10^{20}$	$1.61 \cdot 10^{20}$	89422360845.83	3725931701.91
14	$1.26 \cdot 10^{22}$	$6.28 \cdot 10^{21}$	3487472072987.36	145311336374.47

Table 1: time required for attack

2.9 Limitations

All published information was gathered using an out-of-the-box Oracle 8i database server. More recent Oracle database versions show other behavior out-of-the-box when using native Oracle drivers. However, we have found this attack to be applicable to all Oracle database versions providing non-Oracle and/or non-native drivers are used to set up a connection. For example, the Oracle JDBC Thin Driver[16] including version 10g Release 2 forces Oracle 9i and 10g databases to fall back to the mechanism as described in this paper. Note that this driver is commonly used in Java based application servers environments like IBM WebSphere[17], Apache TomCat[18] and BEA WebLogic[19] for connecting to Oracle databases. It is also known that it's possbile to downgrade the Oracle native authentication type by actively injecting packets on the network[20].

This attack is therefore not limited to the investigated Oracle 8i version, but, within limitations, also applies to other Oracle versions up to and including 10g.

3 Conclusion

The cracking of Oracle passwords entered a new era after publication of the Oracle password hashing algorithm on 18 October 2005 by the SANS institute. A new threat was introduced. Fortunately, the critical first step of retrieving password hashes from a database can be prevented in simple ways by hardening the database server and applying strict schemes for access control, which under normal circumstances would have been implemented anyway.

In this article a new way of attacking Oracle is introduced using passive techniques. From a database point of view, it is very difficult if not impossible to detect the passive attack: there are many possible points of access to network traffic. Preventing the attack is also quite complex to accomplish: secure tunneling or port security on switches is required to prevent an attacker from getting required network access. To ensure that it will stay like that, security policies of network equipment must be in place to keep eavesdroppers out, total control over the path from user (endpoint) to data (database) must be enforced. This is seldom the case.

All data used in this paper was gathered using an out-of-the-box Oracle 8i database server. More recent Oracle database versions show other behavior out of the box when using native Oracle drivers. However, this attack is also applicable to other Oracle database versions when non-Oracle and/or non-native drivers are used to set up a connection.

For both active and passive attacks, the most important line of defense is the password policy. Even if an attacker obtains password hashes in one way or another, an adequate password policy mitigates the risk of cracked accounts. Using a non trivial 10 position password attackers might be kept out; it's too complex to find the password in a reasonable amount of time.

4 Timeline

End of 2005 - PoC coded
Beginning of 2006 - Vendor contacted
Beginning of 2006 - Vendor response
Spring of 2007 - *finally* released the article ;)

5 Future research

Interesting future research might include questions such as:

- How does the session variable generator of Oracle work? Is it possible to reduce the keyspace using this knowledge?
- What is the relation between the session key and encryption keys used for encrypting all client communication?
- How can this information be used to decrypt encrypted (DES, 3DES, AES, RC4) Oracle 8i/9i/10g network traffic?

6 Contributors

6.1 Author

vonJeek is a security consultant for one of the largest professional services firms in the world. He focuses on the technical side of information security, especially on network, Microsoft Windows and database security. vonJeek has over 5 years of experience in network security, ethical hacking, host based security, intrusion detection and developing security tools.

6.2 Technical editor

DJ RevMoon is head of consultancy for a global security firm and has extensive experience performing network audits, penetration tests and other professional security services. He is author of several tools including THC-amap. ■

References

[1] "An Assessment of the Oracle Password Hashing Algorithm", *SANS Institute,* `http://www.sans.org/rr/special/index.php?id=oracle_pass`

[2] "Hash function", *Wikipedia, the free encyclopedia,* `http://en.wikipedia.org/wiki/Hash_function`

[3] "Oracle Default Password List", *Pete Finnigan - Oracle and Oracle security information,* `http://www.petefinnigan.com/default/default_password_list.htm`

[4] "DATA ENCRYPTION STANDARD (DES)", *FIPS PUB 46-2,* `http://www.itl.nist.gov/fipspubs/fip46-2.htm`

[5] "DES MODES OF OPERATION", *FIPS PUB 81,* `http://www.itl.nist.gov/fipspubs/fip81.htm`

[6] "ASCII", *Wikipedia, the free encyclopedia,* `http://en.wikipedia.org/wiki/ASCII`

[7] "Exclusive or", *Wikipedia, the free encyclopedia,* `http://en.wikipedia.org/wiki/Xor`

[8] "Oracle Exploits / Exploit", *Red Database Security,* `http://www.red-database-security.com/exploits/oracle_exploits.html`

[9] "Oracle Advanced Security Administrator's Guide", *Oracle Corporation,* `http://otn.oracle.com/pls/db10g/db10g.to_pdf?pathname=network.101%2Fb10772.pdf&remark=portal+%28Administration%29`

[10] "New Directions in Cryptography", *Martin E. Hellman Home Page,* `http://www-ee.stanford.edu/~hellman/publications/24.pdf`

[11] "ARP spoofing", *Wikipedia, the free encyclopedia,* `http://en.wikipedia.org/wiki/ARP_poisoning`

[12] "Cain & Abel password recovery tool", *oxid.it - Cain & Abel,* `http://www.oxid.it/cain.html`

[13] "How OSI Works", *Howstuffworks,* `http://computer.howstuffworks.com/osi1.htm`

[14] "Exploiting Cisco Routers: Part 1", *SecurityFocus,* `http://www.securityfocus.com/infocus/1734`

[15] “Dictionary facts”, *Oxford English Dictionary,* `http://www.oed.com/about/facts.html`

[16] “SQLJ/JDBC Download Page”, *Oracle Technology Network,* `http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html`

[17] “WepSphere homepage”, *IBM WebSphere Software,* `http://www.ibm.com/software/websphere/`

[18] “Apache Tomcat homepage”, *Apache Tomcat,* `http://tomcat.apache.org/`

[19] “BEA WebLogic Product Family”, *BEA Systems,* `http://www.bea.com/framework.jsp?CNT=index.htm\&FP=/content/products/weblogic/`

[20] “Downgrading the Oracle native authentication”, *Price Waterhouse Coopers,* `http://www.pwc.com/extweb/service.nsf/docid/3AC99308583CCE398025727400391E31/$file/oraauthdg_pub.pdf`