

PWCHECK-PRO

PWCHECK-PRO

Additional Documentation

RACF Password Strength Assessment Tool

Goldis Consulting Services

1 Trowbridge Terrace
Cambridge, MA 02138

Phone: 617-492-4364

Fax: 617-492-1710

www.goldisconsulting.com



PWCHECK-PRO

Introduction

PWCHECK-PRO extends the capabilities of PWCHECK in several ways:

- 1) It optionally displays “cracked” or “unmasked” passwords in clear text.
- 2) It allows controlled “brute-force” attacking of a particular password.
- 3) It allows reporting on all short passwords present on your RACF database, i.e. passwords of four characters or fewer.
- 4) It expands the `USERMASK` and `DFTGROUP` functionality by permitting you to specify users or groups to be excluded from processing. PWCHECK allows you to specify only users or groups to be included.
- 5) It supports “Mixed Case” passwords, a feature available in z/OS 1.7 and later releases which you may desire.

These features are activated using parameter cards in the PWCKPARAM dataset that is recognized when PWCHECK-PRO is installed.

The PWCHECK program recognizes three cards in PWCKPARAM, which are described in the PWCHECK documentation:

```
//PWCKPARAM DD *  
RUNLIMIT=(0000000,9999999)  
USERMASK=(*****)  
DFTGROUP=(*****)
```

PWCHECK-PRO program recognizes several additional cards which are described here.

PWCHECK-PRO

1.0 The RUNTYPE Card

The RUNTYPE card can be specified in one of three ways:

```
RUNTYPE=(CLEAR)    or
RUNTYPE=(BRUTE)    or
RUNTYPE=(CLEAR, BRUTE)
```

RUNTYPE=(CLEAR) specifies that cleartext passwords should be printed in the output report. The report is exactly the same as with PWCHECK, except that cleartext passwords are printed in the "PASSWORD" column instead of the string "GUESSED" or "UNHASHED".

RUNTYPE=(BRUTE) specifies that a "brute force" guessing routine will be executed against a particular userid. This process is governed by parameters specified in the PWCKBRUT dataset, an additional parameter dataset used by PWCHECK-PRO. JCL for the job might look like this:

```
//PWCHECK JOB 1,PG,MSGCLASS=X
// EXEC PGM=PWCHECK
//STEPLIB DD DISP=SHR,DSN=SYS1.LOCAL.LINKLIB
//PWCKPRT1 DD DUMMY
//PWCKPRT2 DD SYSOUT=*
//PWCKLICN DD DISP=SHR,DSN=userid.LIB.CNTL(PWCKLIC)
//PWCKDICT DD *
PASSWORD
SECRET
/*
//PWCKPARAM DD *
RUNLIMIT=(0000000,9999999)
USERMASK=(***** )
DFTGROUP=(***** )
RUNTYPE=(BRUTE)
//PWCKBRUT DD *
RACFID  ???????? ???????? 8??? 00000000 ????????
```

RUNTYPE=(BRUTE) does not print a cleartext password.

RUNTYPE=(CLEAR, BRUTE) specifies that the brute force option will be used and that guessed passwords will appear in clear text in the report.

Brute force report output is written to the PWCKPRT2 dataset.

PWCHECK-PRO

1.1 PWCKBRUT dataset

This parameter dataset contains one record. The fields in the record must appear in correct format in correct columns. Here is the sample record delivered with the package:

```
-----1-----2-----3-----4-----5-----6-----7-----8
RACFID  ???????  ???????  8???  00000000  ????????
```

Column	Field	Contents
1-8	RACFID	The RACF ID of the userid to be subjected to the brute force process.
9	Field separator	space
10-17	Known Characters	When one or more of the characters in the password to be guessed is already known, they may be specified here in order to reduce unnecessary processing. Any character that is not known must be specified by a '?'. All eight positions must contain either a '?' or a valid password character.
18	Field separator	space
19-26	General Knowledge Masks	If something is known in general about any of the password characters, it may be specified in this field. For example if it is known that a particular character is a vowel, a 'V' can be substituted for the default '?' in the appropriate positions. Valid specifications here are: V - Character is a vowel (universe size = 5) C - Character is a consonant (universe size = 21) A - Character is an Alpha (universe size = 26) N - Character is a Numeric (universe size = 10) L - Character is Alpha or Numeric (universe size = 36) ? - Character is unknown (universe size = 39)
27	Field separator	space
28	Password Length	The brute force feature tries all passwords of a particular length, which is specified in this field. Valid values are '4', '5', '6', '7', or '8'. To perform brute force guessing against all passwords of minimum length 6, the program would be run three times, once each with '6', '7' and '8' specified here.
29	Vowel Required	Anything other than '?' in this field causes the program to formulate only guesses with at least one vowel present.
30	Reserved	
31	Numeric Required	Anything other than '?' in this field causes the program to formulate only guesses with at least one numeric present.
32	Field separator	space
33-40	Run Limit	This field is used to limit the number of guesses to be tried. All eight positions must contain a numeric character. When the field is '00000000', no limit is set. This is useful for benchmark trials, and some PWCHECK-PRO messages refer to it as the "benchmark field".
41	Field separator	space
42-49	Resume Password	This field contains either '????????' or a valid "resume" password. A resume password may be useful in cases where the program was cancelled for some reason during a prior run. When this happens, the report shows the last password tried before cancellation. This password can then be used as the resume password in a subsequent run to avoid duplication of effort. It can also be used to run multiple instances of PWCHECK simultaneously on different processors.

PWCHECK-PRO

1.2 Messages That May Appear in Output Report

When the Brute Force Mode is entered, certain error conditions cause an error message to be written to the report output as well as to the console. These typically occur when parsing the brute force parameter input. Examples:

```
'PWCHECK INVALID PASSWORD LENGTH IN PARMS'  
'PWCHECK INVALID BENCHMARK FIELD IN PARMS'  
'PWCHECK INVALID RESUME-PW FIELD IN PARMS'  
'PWCHECK CHARTYPES MUST BE C, A, V, N, L, OR ?'  
'PWCHECK KNOWN CHARS MUST BE VALID PASSWORD CHARS OR ''?''  
'PWCHECK PARAMETERS SAY WE ALREADY KNOW ALL CHARACTERS'
```

1.3 Abnormal Termination

The following additional codes are used by PWCHECK-PRO for abnormal termination:

```
U107 = BAD RETURN FROM PASSWORD ENCRYPTION
```

PWCHECK-PRO

1.4 Sample PWCHECK-PRO Brute Force Report

```
COPYRIGHT GOLDIS CONSULTING SERVICES 2005
PWCHECK VERSION 3.0.1
THIS PROGRAM IS LICENSED TO: GOLDIS CONSULTING SERVICES
LICENSE CODE: 6789L22

THIS PROGRAM WAS RUN AT 00:03 ON 05/05/2005
WE ARE RUNNING ON CPU ID 000001, MODEL 7490
THE SYSID OF THE SYSTEM WE ARE RUNNING ON IS: P390
THE NAME OF THE RACF DATASET IS: SYS1.RACF

INPUT PARAMETERS FOR THIS RUN ARE:
RUNLIMIT=(000000,9999999)
USERMASK=(***** )
DFTGROUP=(***** )
RUNTYPE=(CLEAR,BRUTE)

***** BRUTE FORCE INPUT PARAMETERS:
PGOLDIS M??????? LLLAAAAA 7??? 00001000 ????????

USERID   START-PW LAST-TRY -RESULT- -----CIPHER----- PWS/SEC
PGOLDIS  MAAAAAA M3LANIE GUESSED! 75E1CE8DAEDBD5AC 00100213
```

1.5 Elements of the Brute Force Report

The Environment Information is the same as in the usual PWCHECK-PRO report. The brute force feature is designed to “attack” a single RACFID which is specified in the PWCKBRUT parameter dataset, so instead of detail records for each ID successfully processed, the brute force report contains one line with information about the RACFID which was the subject of the run:

```
USERID   START-PW LAST-TRY -RESULT- -----CIPHER----- PWS/SEC
PGOLDIS  M3LANAA  M3LANIE  GUESSED! 75E1CE8DAEDBD5AC 00100213
```

- USERID** The RACF ID that was the “target” of this run.
- START-PW** The first password tried during this run, based on input parameters.
- LAST-TRY** The last password tried by the program. When “GUESSED!” appears in the RESULT field, this is the password of the RACFID in question. When “ABENDED” appears in the RESULT field (e.g if the program was cancelled), this was the last password tried before the abend. This password could be placed in the RESUME field of the input parameters for a subsequent run to avoid guesses that have already failed before.
- RESULT** Why the program terminated:
GUESSED! = The password was successfully “brute-forced”
NO HIT = The password was not guessed during this run.
UNMASKED = The password existed in hashed form.
ABENDED = The program was cancelled, e.g.

PWCHECK-PRO

CIPHER	The enciphered value of the password as it appeared on the RACF database.
PWS/SEC	The guess rate for this run. This is the approximate number of guesses made per CPU-second of execution time.

PWCHECK-PRO

1.6 Additional Notes on Brute Force Execution

1) When `RUNTYPE=(BRUTE)` is specified in the PWCHECK-PRO input parameters, the output contains '*****' in the `LAST-TRY` field.

```
USERID   START-PW  LAST-TRY  -RESULT-  -----CIPHER-----  PWS/SEC
PGOLDIS  MAAAAAA  *****  GUESSED!  75E1CE8DAEDBD5AC  00100213
```

This means that “RESUME” processing is not possible in subsequent runs. In order to see a clear password in this field, it is necessary to specify `RUNTYPE=(CLEAR, BRUTE)`. RESUME processing is then possible using the contents of the `LAST-TRY` field.

2) The input parameters allow great flexibility in tailoring brute force execution. This flexibility means it is possible to mistakenly specify conflicting parameters in various ways.

Example 1:

```
PGOLDIS  M3??????  AAAAAAAA  7???  00000000  M3BQ22N
```

The character masks say to try only alpha characters, but the resume password contains numerics. In this case, the first password composed by PWCHECK-PRO (the one in the “START-PW” field) would be M3BQAAA. This is a composite made from the “Known Characters” (‘M’ and ‘3’), the ‘B’ and ‘Q’ of the resume password, and ‘AAA’. This is because PWCHECK-PRO detected the conflict while initializing the resume password and abandoned the process at that point, defaulting to the “character masks” for the remaining characters.

Example 2:

```
PGOLDIS  ?????????  AAAAAAAA  7??1  00000000  ?????????
```

Input parms specify that only alpha characters will be used for guesses, but the `7??1` in the input also requires that all guesses must contain at least one numeric. In this case, the program will end with ‘NO HIT’ in the `RESULT` field. The `PWS/SEC` field is `00000000`.

The program will formulate guesses using alpha characters only, but none of these guesses will pass the internal “Quality Assurance” standard for the run since none have a numeric character as is required by the input parms.

3) When `RUNTYPE=(CLEAR, BRUTE)` or `RUNTYPE=(BRUTE)` is specified, all other parameters specified in the PWCKPARAM parameter input dataset are ignored.

PWCHECK-PRO

2.0 The SHORT card

If you would like PWCHECK-PRO to determine whether four-character (or shorter) passwords are in use, you can add the SHORT parameter card to the PWCKPARM input dataset. There is no way to simply query RACF as to the length of someone's password, so when SHORT is specified, PWCHECK-PRO adds a "brute-force" routine to its normal processing.

This has the effect of adding all possible passwords of four characters or shorter to the "dictionary" of password guesses that will be compared. This feature is useful if there is a concern that some users, e.g. legacy CICS users, may be using old, four-character passwords. PWCHECK-PRO will reveal such cases when SHORT is specified.

It should be understood, however, that using the SHORT parameter can greatly increase the execution time of PWCHECK-PRO, particularly if a large segment of the RACF database is to be tested at once or if the PWCHECK-PRO job executes at a low service level in the MVS environment.

The SHORT parameter adds over two million password "guesses" for each user tested when only uppercase alphabetic characters are used. If your system supports mixed case passwords and you specify the MIXEDCASE parameter at execution time, PWCHECK-PRO adds over 18 million password guesses for each user tested.

It is suggested, therefore, that you exercise caution when using the SHORT parameter. We recommend doing some benchmark checks to determine how PWCHECK-PRO will behave in your environment when SHORT is specified.

For example, you might specify SHORT when only a single userid is tested, and then note the execution time of the job. On subsequent tests, you could include a broader userid sample until you become confident of PWCHECK-PRO execution times.

Example 1:

Testing all short, uppercase passwords for a single userid, "IBMUSER".

:

```
RUNLIMIT=(0000000,9999999)
USERMASK=(IBMUSER*)
DFTGROUP=(******)
SHORT
```

Example 2:

Testing all short passwords including upper and lower case characters for a single user.

```
RUNLIMIT=(0000000,9999999)
USERMASK=(IBMUSER*)
DFTGROUP=(******)
SHORT
MIXEDCASE
```

PWCHECK-PRO

Example 3:

Testing the first 100 users found on the RACF database, using all upper and lower case characters

```
RUNLIMIT=(0000000,0000100)
USERMASK=(******)
DFTGROUP=(CICSTEST)
SHORT
MIXEDCASE
```

Example 4:

Testing all userids that have a Default Group of CICSTEST. Again, all upper and lower case passwords are tried for each user tested.

```
RUNLIMIT=(0000000,9999999)
USERMASK=(******)
DFTGROUP=(CICSTEST)
SHORT
MIXEDCASE
```

PWCHECK-PRO

3.0 The MIXEDCASE card

As of z/OS 1.7, RACF supports the use of both upper and lower case characters in passwords. If your system has been specified to use mixed case passwords, you can instruct PWCHECK-PRO to test for them by specifying the MIXEDCASE parameter in the PWCKPARAM input dataset.

When MIXEDCASE is specified, all the entries in the PWCKDICT datasets are processed exactly as they appear, using both upper and lower case characters. This is different from the default behavior of PWCHECK, which is to convert all entries in the PWCKDICT datasets to uppercase characters before testing.

Example 1:

To test and report on the passwords in a system where mixed case passwords are present, specify:

```
RUNLIMIT=(0000000,9999999)
USERMASK=(******)
DFTGROUP=(******)
MIXEDCASE
```

Note that only passwords specified in the “dictionary” of guesses (i.e. the PWCKDICT dataset) will be tested. If all of the entries in the dictionary are uppercase-only, then no mixed case passwords will be tried, even though the MIXEDCASE parameter may have been specified.

If you want to test for the passwords “SECRET1”, “secret1”, and “Secret1”, then you must include all three in the dictionary when MIXEDCASE is specified. If you include only “secret1” in the dictionary, then “SECRET1” will not be tested.

If MIXEDCASE is not specified in the parameter input, then “SECRET1” will be tested when “Secret1” or “secret1” is found in the dictionary.

PWCHECK-PRO

4.0 The EXCLUDEMASKS card

The “EXCLUDEMASKS” parameter changes the function of the USERMASK and the DFLTGRP parameters. A match on anything specified in either of these parameters will NOT be reported on by PWCHECK if EXCLUDEMASKS has been specified.

If EXCLUDEMASKS is specified and both USERMASK and DFTGROUP is specified as ‘*****’, PWCHECK will test and report on all userids as if EXCLUDEMASKS had not been specified.

Example 1:

To test and report on all userids except those whose default group is SYS1, specify

```
RUNLIMIT=(0000000,9999999)
USERMASK=(******)
DFTGROUP=(SYS1****)
EXCLUDEMASKS
```