63 Cracking Techniques

"He who stands firm to the end will be saved."

"*Hacker*" is an enthusiastic programmer. "*Cracker*" is the one who does illegal operations like stealing data, passwords etc through programming. So the Cracker might be a Hacker, and the Hacker need not be a Cracker. But in India, both "Hacking" and "Cracking" are interchangeably used.

Password cracking techniques can basically be classified into:

1. Brute force technique
2. Dictionary attack

63.1 Brute force technique

In brute force technique, all combinations of valid characters are tried until we get the right password. For example, if the length of the password is 1, we have to try 'A', 'B'…'Z' or '0','1'…'9', and the process has to continue until the right password is found. If the application uses case-sensitive passwords or special symbols as valid characters, then we have to try 'a', 'b'…'z' and '~', '$'… too. And so from programming point of view, brute force technique is considered to be very time-consuming technique.

I have written the following program to generate word list of length 2. It accepts the file name in which the strings are to be added as an argument.

```
/* File name: Brute.c */
#include <stdio.h>

char
Valid_Chars[]="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
               " 1234567890!@#$%^&*()-=_+`~[]\\{}|;':\",./<>?";

int main( int argc, char *argv[] )
{
    unsigned long i, j, n;
    char str[5];
    FILE *fp;

    if ( argc<2 )
      {
         printf( "Syntax: BRUTE <output file name> \n\a" );
         exit(1);
      }
    if ( (fp=fopen( argv[1], "w"))==NULL )
      {
         perror( "Error" );
         exit(1);
      }
    printf( "Strings are being generated... \n" );
    n = 0;
    for ( i=0 ; i<strlen( Valid_Chars ) ; ++i )
      for ( j=0; j<strlen( Valid_Chars ); ++j )
          {
             str[0] = Valid_Chars[i];
             str[1] = Valid_Chars[j];
             str[2] = '\0';
             fprintf( fp, "%s \n", str );
             ++n;
          }
    fclose( fp );
    printf( "No. of strings written to %s is %ul \a\n", argv[1], n );
    return(0);
}
```

When you run the above program as

```
C:\> BRUTE words.lst
```

You would get about 90 thousand words! All the words are with length 2. So it is more time consuming. You can add more *for* loops to get words of length other than 2. But it won't be an efficient implementation, you need to try another method. Optimized implementation of generating words list using brute force technique is left to the reader as an exercise. You may change the `Valid_Chars` table if you don't require all the characters.

63.2 Dictionary attack

In this technique, all words that are expected to be the right password are tried. But, there is a difference… it won't directly try those passwords with the software as in brute force technique. The software's encrypting technique like hash values etc will be performed on those passwords and if there is a match in the *key*, it recognizes it as the right password. Mostly people prefer this technique, because it is not much time consuming compared to brute force.