Project Entitled

**Creating Dictionary for Attacks**

# **Introduction:**

In [cryptanalysis](http://en.wikipedia.org/wiki/Cryptanalysis) and [computer security](http://en.wikipedia.org/wiki/Computer_security), a dictionary attack is a technique for defeating a [cipher](http://en.wikipedia.org/wiki/Cipher) or authentication mechanism by trying to determine its decryption key or passphrase by trying hundreds or sometimes millions of likely possibilities, such as words in a dictionary.

A dictionary attack uses a targeted technique of successively trying all the words in an exhaustive list called a dictionary (from a pre-arranged list of values). In contrast with a [brute force attack](http://en.wikipedia.org/wiki/Brute_force_attack), where a large proportion [key space](http://en.wikipedia.org/wiki/Key_space_(cryptography)) is searched systematically, a dictionary attack tries only those possibilities which are most likely to succeed, typically derived from a list of words for example a dictionary. This dictionary attack is guaranteed to succeed.

In this project, we use this approach to crack passwords to access emails, Wireless access points, encrypted zip files etc.

1. **Origin of Proposal:**

The project throws light on the fact that passwords that consist of strings can be cracked by applying passwords, over and over again, until they get in. There has to be one exact string in the dictionary that matches the password to gain access. We have used a simple C++ program to apply strings over and over again.

1. **Objectives of the proposed work:**

It is often said that humans are always the weakest link in the security chain. Social engineering is more frequently successful in penetrating a system than buffer overflow attacks. Similarly, cracking a password is rarely accomplished by breaking the cipher used and more often by exploiting the circumstance that it was generated by a human. This project solely aims on cracking passwords.

1. **Issues Faced:**

**4.1 Issues Faced in Generating the Dictionary**

The major open problems are:

* time taken
* memory space required
* hardware deficiency

4.1.1 Time complexity

O(n)

4.1.2 Memory complexity

O(n2)

4.1.3 Hardware Deficiency

# The high end machines with 8GB ram, quad core CI7, though are considered as expensive and high end machines does not have the required hardware to execute the entire dictionary in a decent amount of time.

# 

**4.2** **Issues faced in applying the dictionary**

* Lack of appropriate tools
* Time required

4.2.1. Lack of Appropriate Tools

Though the dictionary is created with the required passwords, specific tools are required to attack various security gateways.

4.2.2 Time Required

Cracking passwords is a lengthy process which requires a lot of time and patience.

1. **Coding**

The coding is purely done on C++. Following is the code:

#include<iostream.h>

#include<conio.h>

#include<string.h>

#include<fstream.h>

using namespace std;

void main(){

ofstream file;

file.open("test.txt", ios::out);//ye open n save kkardega

//filename<<"this is a test";// ye file me likha jaaega

int x,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p;

char key[17];

//cout<<"Enter number of charcters\n";

//cin>>x;

a=0;b=0;c=0;d=0;e=0;f=0;g=0;h=0;i=0;j=0;k=0;l=0;m=0;n=0;o=0;p=0;

for(a=0;a<17;a++)

key[a]='\0';

a=0;

for(a=32;a<=126;a++)

{

for(b=32;b<=126;b++)

{

for(c=32;c<=126;c++)

{

for(d=32;d<=126;d++)

{

for(e=32;e<=126;e++)

{

for(f=32;f<=126;f++)

{

for(g=32;g<=126;g++)

{

for(h=32;h<=126;h++)

{

for(i=32;i<=126;i++)

{

for(j=32;j<=126;j++)

{

for(k=32;k<=126;k++)

{

for(l=32;l<=126;l++)

{

for(m=32;m<=126;m++)

{

for(n=32;n<=126;n++)

{

for(o=32;o<=126;o++)

{

for(p=32;p<=126;p++)

{

key[0]=(char)p;

cout<<key<<endl;

filename<<key<<endl;

}

key[1]=o;

}

key[2]=n;

}

key[3]=m;

}

key[4]=l;

}

key[5]=k;

}

key[6]=j;

}

key[7]=i;

}

key[8]=h;

}

key[9]=g;

}

key[10]=f;

}

key[11]=e;

}

key[12]=d;

}

key[13]=c;

}

key[14]=b;

}

key[15]=a;

}

getch();

}

1. **The Future Trends**

Though 25-GPU cluster cracks every standard Windows password in <6 hours. A password-cracking expert has unveiled a computer cluster that can cycle through as many as 350 billion guesses per second. It's an almost unprecedented speed that can try every possible Windows passcode in the typical enterprise in less than six hours.

Enhancement of tools that allow the user to upload .cpp files to attack instead of using .txt files. This saves a great amount of space. The .txt files may take up to 250GB of space whereas .cpp files may take up negligible space.

Modern wireless access points have set a minimum of password length of 8 characters. So we will have to change the reference string length to 8-17. This will reduce the execution time by at most a day and reduce the required storage by 150GB.

We have to make modules of each string length as a computer cannot be expected to run continuously for that length of time. So, the modules for each string length can be executed separately.

References:

[1] MD5 online cracking using rainbow tables. http://www.passcracking.com/.

[2] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In Proc. EUROCRYPT ’00, volume 1807 of LNCS, pages 139–155. Springer, 2000.

[3] S. Bellovin and M. Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In Proc. IEEE Security and Privacy Symposium, pages 72–84. IEEE Computer Society, 1992.  
[4] A. Booker. The Nth prime algorithm. http://primes.utm.edu/nthprime/algorithm.php, 2005.