

Forum:[Hack Forums](#)

Thread: [\[C++ SOURCE CODE\] Multithread brute forcer \(using boost C++ libraries\)](#)

malo1992 - June 29, 2011, 12:56 pm

At first, you may read this(my first brute force code, released yesterday):

<https://hackforums.net/showthread.php?tid=1426822>

Okay dudes, now THIS is fucking awesome...

You may know that the key of a good bruteforcer is its SPEED.

I google'd a lot on multithreading and found this: <http://www.boost.org/> ?

This really drives me crazy. Some godlike nerd called Anthony Williams created a multithread-library(simply called "thread") for this great code collection. I was immediately like: ? and ? !

It was easy to install and apply to my project.

AND NOW COMES THE CLUE!

With only two parallel-running bruteforce engines(this is just the beginning... if you want, you can use 123123123 billion, i think ?), my new code rapes the string "asdf" in LESS THAN A SECOND!!! I'm still so fucking shocked. To compare, my old code took 18 seconds(!!!!) to bust this fuckin simple 4 character string ?

Accordingly, my new multithreaded brute force attack is about 1800 % faster :D

WHAT ABOUT THAT?!?!?! MWAHAHAHA

Edit: aii that wasnt correct xD In fact, the couts slowed the first code radically down. Without couts, it's much faster. Anyway, i stopped the time again. The given string was "asdfgx", first code(now without couts, no multithreading) took 23 seconds, new code(no couts either, but 2 threads at once) took 12 seconds.

Aye, I was already wondering bout that. ~100% speed increasement on 2 threads is more logical than 1800 <.< lol@me. I still have much to learn.

NOTE: COUT SUX!

You'll need the boost-library thing to compile the source code, and its damn big, thus I uploaded the prog right here(I swear that it contains NO MALWARE! would be <3ly if someone could proof this with an virustotal scan report.)

<http://www.mediafire.com/?9wcv8xb0zk5rn27>

(I have a 32bit XP windows, i hope it works on other systems....)

heres the code, I didnt give a fuck on new // comments this time. was too excited. furthermore, I had to cut the bruteword outputs out, cuz cout sux on multithreaded applications and I was too lazy for printf'ing this(i don't even know if printf works correctly).... who cares. you'll still love it.

Code // *** MULTITHREADING C++ BRUTE FORCE EXAMPLE CODE ***

// written by malo1992

//

// You will need the "boost C++ libraries" to compile it.

// <http://www.boost.org/>

// Special thanks to Anthony Williams!

// Have fun!

//

// Germany, June 2011

```
#include <iostream>
#include <string>
#include <boost/thread.hpp>          // THIS IS THA SHIT!!!!
// #include <Windows.h>    -> only 4 debug reasons
using namespace std;

/* CHARACTER LISTS */
char lower[27]={ 'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','\0'};
char upper[27]={ 'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','\0'};
char numbs[11]={ '0','1','2','3','4','5','6','7','8','9','\0'};
char symbls[31]={ '!', '?', ':', ';', ',', '-', '_', '#', '$', '*', '+', '%', '&', '/', '\\', '(', ')', '[', ']', '@', '=', ':', '{', '}', '<', '>', '|', '^', '"', '~', '\0'};
char alphabet[96];          // This will contain the brute force alphabet

/* VARIABLES */
string password = "";       // We will brute-force this
string bruteword = "";      // The string we want to be =password
string pw = "";             // Just a help variable; contains the matching bruteword
string letters = "";        // Letters that will be added to the current character(alphabet[X]) in order to
create the bruteword
bool pw_found = false;      // Password ain't found yet.
int maxChars = 15;          // Maximum length of the bruteword (can be changed by the user)
int minChars = 1;           // Minimum length o. t. BW ( " )
int alph_length = 0;        // Length of the char alphabet

/* VOIDS */
void asdf();                // Nonsense, but funny ;P
void createAlph();          // Creates brute alphabet
void check1(string bruteword);
void check2(string bruteword);
void check3(string bruteword);
void check4(string bruteword);    // Checks if bruteword = password
void brute1 (int charsInBW, int instance, string letters);    // BRUTE BRUTE BRUTE
void brute2 (int charsInBW, int instance, string letters);
void brute3 (int charsInBW, int instance, string letters);
void brute4 (int charsInBW, int instance, string letters);

/* PROGRAM CODE */
int main()
{
    // START ROUTINE //
    cout << endl;
    cout << "Enter a password to attack: ";
    cin >> password;

    cout << endl;
    cout << "Minimum length of the bruteword(default 1): ";
    cin >> minChars;
    cout << "Maximum length of the bruteword(default 15): ";
    cin >> maxChars;

    // CREATING THE ALPHABET ARRAY //
```

```
createAlph();
```

```
// Right before we start...
```

```
cout << endl;
```

```
cout << "***" << endl;
```

```
cout << "Password string that will be attacked: " << password << "" << endl;
```

```
alph_length = strlen(alphabet);
```

```
cout << "Brute force alphabet: " << alphabet << endl;
```

```
cout << "Characters: " << alph_length << endl;
```

```
asdf(); // Some useless extra feature, I was bored
```

```
cout << "***" << endl << endl;
```

```
cout << "Press ENTER to start brute forcing..." << endl;
```

```
cin.get();cin.get(); // Forces the application to wait for user input
```

```
cout << "Bruteforcing .." << endl;
```

```
cout << "Please wait. " << endl << endl;
```

```
// LET'S GO! //
```

```
for( int i=minChars; i<=maxChars ; i++) // Bruteword needs [i] characters.
```

```
{ // ( e.g. the brutewords with 1 character don't match? Then use 2 characters etc etc ... )
```

```
    boost::thread bf_core1(brute1, i, 0, letters);
```

```
    boost::thread bf_core2(brute2, i, 0, letters);
```

```
    bf_core1.join();
```

```
    bf_core2.join();
```

```
}
```

```
// FOR()-LOOP STOPPED, PASSWORD FOUND OR NOT?
```

```
if ( pw_found )
```

```
{ // match! :D
```

```
    cout << "Bruteforcing stopped." << endl << endl;
```

```
    cout << "PASSWORD MATCH!!! -> " << pw << endl << endl;
```

```
    cout << "Press ESCAPE to close the application...";
```

```
    cin.get(); // Forces the application to wait for user input
```

```
}
```

```
else
```

```
{ // no match ... :(
```

```
    cout << "Bruteforcing stopped." << endl << endl;
```

```
    cout << "*** No matching password has been found." << endl << endl;
```

```
    cout << "Press ESCAPE to close the application...";
```

```
    cin.get(); // Forces the application to wait for user input
```

```
}
```

```
return 0;
```

```
}
```

```
void brute2(int charsInBW, int instance, string letters)
```

```
{
```

```
    if ( instance+1 != charsInBW )
```

```
    {
```

```
        for(int b=0; b<alph_length; b++)
```

```
        {
```

```
            brute2(charsInBW, instance+1, alphabet[b]+letters);
```

```
        }
```

```
    }
```

```
    else
```

```
    {
```

```

    for(int b=1; b<alph_length ; b=b+2 )    // There are [alph_length] characters in char array
    alphabet( e.g.'a'=position 0 and 'z'=position 25).
    {    // Now, alphabet[b] can display every single character in the chosen alphabet.
        if( !pw_found )    // Password still unknown?
        {    // ... with alphabet[b]+(letters of other instances) in the "letters"-string has been
started.
            check2(alphabet[b]+letters);    // This is the last instance which checks ...
        }    // ... if (current character)+(letters of all other instances) is =password.
        else    // A matching password has been found!
        {
            b=alph_length;    // This stops the for()-loops in void brute...
            maxChars = pw.size();    // ... and in int main().
        }
    }
}
}
}

```

```

void brute1(int charsInBW, int instance, string letters)
{
    if ( instance+1 != charsInBW )
    {
        for(int b=0; b<alph_length; b++)
        {
            brute1(charsInBW, instance+1, alphabet[b]+letters);
        }
    }
    else
    {
        for(int b=0; b<alph_length-1 ; b=b+2 )    // There are [alph_length] characters in char array
        alphabet( e.g.'a'=position 0 and 'z'=position 25).
        {    // Now, alphabet[b] can display every single character in the chosen alphabet.
            if( !pw_found )    // Password still unknown?
            {    // ... with alphabet[b]+(letters of other instances) in the "letters"-string has been
started.
                check1(alphabet[b]+letters);    // This is the last instance which checks ...
            }    // ... if (current character)+(letters of all other instances) is =password.
            else    // A matching password has been found!
            {
                b=alph_length;    // This stops the for()-loops in void brute...
                maxChars = pw.size();    // ... and in int main().
            }
        }
    }
}
}
}

```

```

void check2(string bruteword)
{
    //Sleep(1000);
    //cout << bruteword << endl;
    if ( bruteword == password ) // Bruteword matches?
    {
        pw_found = true;    // Stops the creation of instances in void brute.
        pw = bruteword;    // Writes a global string for displaying the matching bruteword in int main.
    }
}

```

```

    }
}

void check1(string bruteword)
{
    //Sleep(1000);
    //cout << bruteword << endl;
    if ( bruteword == password ) // Bruteword matches?
    {
        pw_found = true;    // Stops the creation of instances in void brute.
        pw = bruteword;     // Writes a global string for displaying the matching bruteword in int main.
    }
}

void createAlph()    // Should be self explanatory ...
{
    cout << endl;
    string answ = "";    // This will store the user's answers during alphabet construction

    cout << "Attack with lower case letters(Y/N)? ";
    cin >> answ;
    if ( answ == "y" || answ == "Y" ) strcat_s(alphabet,sizeof(alphabet),lower,sizeof(lower));
    else if ( answ == "n" || answ == "N" ) cout << "**** No lower case letters." << endl;
    else cout << "**** Wrong input." << endl << "Lower case letters have NOT been added to the
brute force alphabet." << endl;
    answ = "";    // Reset for possible rewrite

    cout << "Attack with numbers(Y/N)? ";
    cin >> answ;
    if ( answ == "y" || answ == "Y" ) strcat_s(alphabet,sizeof(alphabet),numbs,sizeof(numbs));
    else if ( answ == "n" || answ == "N" ) cout << "**** No numbers." << endl;
    else cout << "**** Wrong input." << endl << "Numbers have NOT been added to the brute force
alphabet." << endl;
    answ = "";

    cout << "Attack with upper case letters(Y/N)? ";
    cin >> answ;
    if ( answ == "y" || answ == "Y" ) strcat_s(alphabet,sizeof(alphabet),upper,sizeof(upper));
    else if ( answ == "n" || answ == "N" ) cout << "**** No upper case letters." << endl;
    else cout << "**** Wrong input." << endl << "Upper case letters have NOT been added to the
brute force alphabet." << endl;
    answ = "";

    cout << "Attack with symbols(Y/N)? ";
    cin >> answ;
    if ( answ == "y" || answ == "Y" ) strcat_s(alphabet,sizeof(alphabet),symls,sizeof(symls));
    else if ( answ == "n" || answ == "N" ) cout << "**** No symbols." << endl;
    else cout << "**** Wrong input." << endl << "Symbols have NOT been added to the brute force
alphabet." << endl;
    answ = "";
}

void asdf()

```

```

{
    string first = "";
    string last = "";
    char cFirst=alphabet[0];
    char cLast=alphabet[alph_length-1];

    for(int i=1; i<=minChars;i++)
    {
        first=first+cFirst;
    }

    for(int i=1; i<=maxChars;i++)
    {
        last=last+cLast;
    }

    cout << "Going to attack from " << first << " to " << last << "!" << endl;
}
//
// END OF FILE

```

cicaglisa - February 10, 2013, 2:39 pm

Can you clarify few things please?

you have 2 completely same brute functions running in 2 threads.. And you are passing same chars to it, and both threads start from length=1...

So, if I am not mistaken, you are bruteforcing the same chars combinations and same string lengths in both threads..

Therefore, it would look like you are doing same thing as there was only one thread, since you aren't putting different combinations in each thread?!

And if I am mistaken, is it possible to just make FOR loop and put like 1000 threads in it? Because right now, CPU usage is very low with 2 threads running..

Drug Lord. - February 11, 2013, 4:35 pm

[i] What does it mean by, "Enter a password to attack"? Isn't already attacking passwords?

oprichniki - February 11, 2013, 10:57 pm

(Feb 10, 2013 03:39 PM)cicaglis[a](#) Wrote: [\[CITE\]](#) Can you clarify few things please?

you have 2 completely same brute functions running in 2 threads.. And you are passing same chars to it, and both threads start from length=1...

So, if I am not mistaken, you are bruteforcing the same chars combinations and same string lengths in both threads..

Therefore, it would look like you are doing same thing as there was only one thread, since you aren't putting different combinations in each thread?!

And if I am mistaken, is it possible to just make FOR loop and put like 1000 threads in it? Because right now, CPU usage is very low with 2 threads running..If it were me I would try a number roughly equal to the number of CPU cores. If you have a hyperthreaded I7, I would try 8. My aircrack bruteforcer uses 6 threads on my quad core I5, but my OS will automatically schedule threads to the GPU if it isn't busy. It can vary sometimes, but too many threads won't do much good other than waste memory, and possibly bog down the scheduler.

Of course one should always be careful creating threads in a loop. If you just naively fork in a 100

iteration loop, you'll end up with 100! threads.

cicaglis - February 12, 2013, 12:39 am

His last visit was just a month ago, so it's possible he will see this...

(Feb 11, 2013 05:35 PM)NasHF Wrote: [\[CITE\]](#) [i] What does it mean by, "Enter a password to attack"? Isn't already attacking passwords?

Well, currently, this is just a skeleton for bruteforcing... It just checks how long does program take to generate password like the one you specified, starting from character with length=1 and increasing..

You can put your own function instead of this:

```
if ( bruteword == password )
```

Like for ex. you can put:

```
LPTSTR bruteword='MD5_hash_to_be_cracked';  
if(CreateMD5hashFromString(password)==bruteword){pass=found}
```

...to bruteforce MD5 for some hash...

Also, I think I found an error:

Brute1() and Brute2() have 1 difference here:

```
for(int b=1; b<alph_length ; b=b+2 )
```

and

```
for(int b=0; b<alph_length-1 ; b=b+2
```

EDIT: here is one more similar bruteforce program has SSE2 optimization etc.:

<https://code.google.com/p/lavernasbrute/source/browse/#svn%2Ftrunk%253Fstate%253Dclosed>

Irritable - February 12, 2013, 1:26 am

Comparing an entire string to generate an option is very ill-advised.

You should use a different scheme, IMO. Such as an enumeration (C++) or #define (C), and use a switch statement.

From the looks of your question: I don't see much of a question.

You are saying to repeat the same work. In essence: You'd be burning 50% of the CPU power on 'nothing' (progress wise).

The only logic behind multiple threads in a brute forcer would be to manipulate the OS into giving a bigger time-share.

If you were serious about brute forcing: Look into rainbow cracking. Much, MUCH more efficient. It is a space-versus-time problem.