

Forum:[Hack Forums](#)

Thread: [Why does my cracker takes time to crack more than 8 password lenght](#)

HiddenEyE - July 16, 2020, 2:47 am

Please has anyone coded a cracker and notice that it takes time to be able to crack more than 8 password length.

Please share your experience and possible solution.

Have tried a lot of optimization to they best of my knowledge.but I think in c/c++ everything is possible.I need help.

Is for educational purposes.thanks

saltycracker - July 16, 2020, 5:49 am

You could post your code and at bit of an explanation to what it does.

Vibe - July 16, 2020, 5:51 am

Could you maybe need more processing power to speed it up?

m8than - July 16, 2020, 11:20 am

Because every time you add a character the total amount of passwords to guess exponentially increases. Think about it, you are adding at least like 35 characters on top of every single previous password checked.

Pawnhub - July 16, 2020, 10:30 pm

Let's say there's 33 possible character combinations you're generating for each character you'd have 1^{33} possible combinations.

if the passwords has a length of 8 you are generating 8^{33} possible combinations.

if 9 it'd be 9^{33} d.

9 characters all possible combinations would equal: 30,903,154,382,632,612,361,920,641,803,529
VS lets say a password with 4 characters: 73,786,976,294,838,206,464

and if the cracker is webbased, it's probably your implementation of requesting to the site and lack of multithreading or parallel processing.

Xora - July 17, 2020, 1:04 am

I just wrote a test prog. It takes me 35 seconds to generate 1 billion 8 character passwords (28.5 million passwords per second) with 40 lines of code. It has no logic though, it just creates random passwords.

What's a good time for this type of thing?

Here's the code, BTW. Pretty shit but maybe you can improve it and use it somehow.

```
Code #define var unsigned int
#define avar __declspec(align(4)) var
const char g_chars[] = {
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*()_+!="
};
```

```

const avar g_char_cnt = sizeof(g_chars);
char g_buff[9];
avar a = 0x499602d2, b = 0x159a55e5, c = 0x1f123bb5, d = 0x05491333;
const var xrand()
{
    register avar t = a ^ (a << 0xb);
    a = b; b = c; c = d;
    d ^= (d >> 0x13) ^ (t ^ (t >> 0x8));
    return 1 + (t % g_char_cnt);
}
#define randchr g_chars[xrand()]
const char * rand_str()
{
    *(avar *)g_buff = ((BYTE)randchr << 24) | ((BYTE)randchr << 16) | ((BYTE)randchr << 8) |
(BYTE)randchr;
    *(avar *)((avar)g_buff + sizeof(var)) = ((BYTE)randchr << 24) | ((BYTE)randchr << 16) |
((BYTE)randchr << 8)
    | (BYTE)randchr;
    return g_buff;
}
#define zero(x) memset(&x, 0, sizeof(x))
#define zeroa(x) memset(&x, 0, sizeof(x) / sizeof(*x))
const avar g_maxpwd = 1000000000;
int main()
{
    LARGE_INTEGER f, s, e, el; zero(f); zero(s); zero(e); zero(el);
    zeroa(g_buff);
    QueryPerformanceFrequency(&f);
    const HANDLE thisthr = GetCurrentThread();
    avar oldpr = GetThreadPriority(thisthr);
    SetThreadPriority(thisthr, THREAD_PRIORITY_TIME_CRITICAL);
    QueryPerformanceCounter(&s);
    for(register avar i = 0; i < g_maxpwd; ++i)
    {
        rand_str();
    }
    QueryPerformanceCounter(&e);
    SetThreadPriority(thisthr, oldpr);
    el.QuadPart = e.QuadPart - s.QuadPart;
    el.QuadPart *= 1000000000;
    el.QuadPart /= f.QuadPart;
    const DOUBLE sec = (DOUBLE)el.QuadPart / 1000000000.0;
    const DOUBLE gps = g_maxpwd / sec;
    printf("generated %d 8 character passwords in:\n\t%lld nanoseconds,\n\t%.1f seconds,\n\t%.1f
passwords/second", g_maxpwd,
        el.QuadPart, sec, gps);
    getchar();
    return 0;
}

```

HiddenEyE - July 17, 2020, 3:33 am

(Jul 17, 2020 01:04 AM)Xora Wrote: [CITE](#) I just wrote a test prog. It takes me 35 seconds to generate 1 billion 8 character passwords (28.5 million passwords per second) with 40 lines of code.

It has no logic though, it just creates random passwords.

What's a good time for this type of thing?

Here's the code, BTW. Pretty shit but maybe you can improve it and use it somehow.

```
Code #define var unsigned int
#define avar __declspec(align(4)) var
const char g_chars[] = {
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*()_+!="
};
const avar g_char_cnt = sizeof(g_chars);
char g_buff[9];
avar a = 0x499602d2, b = 0x159a55e5, c = 0x1f123bb5, d = 0x05491333;
const var xrand()
{
    register avar t = a ^ (a << 0xb);
    a = b; b = c; c = d;
    d ^= (d >> 0x13) ^ (t ^ (t >> 0x8));
    return 1 + (t % g_char_cnt);
}
#define randchr g_chars[xrand()]
const char * rand_str()
{
    *(avar *)g_buff = ((BYTE)randchr << 24) | ((BYTE)randchr << 16) | ((BYTE)randchr << 8) |
(BYTE)randchr;
    *(avar *)((avar)g_buff + sizeof(var)) = ((BYTE)randchr << 24) | ((BYTE)randchr << 16) |
((BYTE)randchr << 8)
    | (BYTE)randchr;
    return g_buff;
}
#define zero(x) memset(&x, 0, sizeof(x))
#define zeroa(x) memset(&x, 0, sizeof(x) / sizeof(*x))
const avar g_maxpwd = 1000000000;
int main()
{
    LARGE_INTEGER f, s, e, el; zero(f); zero(s); zero(e); zero(el);
    zeroa(g_buff);
    QueryPerformanceFrequency(&f);
    const HANDLE thisthr = GetCurrentThread();
    avar oldpr = GetThreadPriority(thisthr);
    SetThreadPriority(thisthr, THREAD_PRIORITY_TIME_CRITICAL);
    QueryPerformanceCounter(&s);
    for(register avar i = 0; i < g_maxpwd; ++i)
    {
        rand_str();
    }
    QueryPerformanceCounter(&e);
    SetThreadPriority(thisthr, oldpr);
    el.QuadPart = e.QuadPart - s.QuadPart;
    el.QuadPart *= 1000000000;
    el.QuadPart /= f.QuadPart;
    const DOUBLE sec = (DOUBLE)el.QuadPart / 1000000000.0;
```

```

const DOUBLE gps = g_maxpwd / sec;
printf("generated %d 8 character passwords in:\n\t%lld nanoseconds,\n\t%.1f seconds,\n\t%.1f
passwords/second", g_maxpwd,
    el.QuadPart, sec, gps);
getchar();
return 0;
}Pm you now mate

```

(Jul 16, 2020 11:20 AM)m8than Wrote: [\[CITE\]](#) Because every time you add a character the total amount of passwords to guess exponentially increases. Think about it, you are adding at least like 35 characters on top of every single previous password checked.

Yes mate Crackers grows exponentially.

The question is that how can one reduce the huge time interval involve in cracking like 24 password length.

(Jul 16, 2020 10:30 PM)UnknownPawn Wrote: [\[CITE\]](#) Let's say there's 33 possible character combinations you're generating for each character you'd have 1^{33} possible combinations. if the passwords has a length of 8 you are generating 8^{33} possible combinations. if 9 it'd be 9^{33} d.

9 characters all possible combinations would equal: 30,903,154,382,632,612,361,920,641,803,529 VS lets say a password with 4 characters: 73,786,976,294,838,206,464

and if the cracker is webbased, it's probably your implementation of requesting to the site and lack of multithreading or parallel processing.

Yes I know mate. I need method being used by others in reducing the time interval.

m8than - July 17, 2020, 7:05 am

(Jul 17, 2020 03:33 AM)HiddenEyE Wrote: [\[CITE\]](#) Pm you now mate

Yes mate Crackers grows exponentially.

The question is that how can one reduce the huge time interval involve in cracking like 24 password length.

Yes I know mate. I need method being used by others in reducing the time interval.

I mean you can just do micro optimisations, every little counts but bruteforcing a long password is probably impossible for modern computers.

Xora - July 18, 2020, 12:16 am

(Jul 17, 2020 03:33 AM)HiddenEyE Wrote: [\[CITE\]](#) Pm you now mate

Unfortunately, mate, it seems I am not l33t enough to figure out how to generate all possible combinations using a list of characters, so I can only help you if you (with optimizing the code) or someone else provides me with a clear code that demonstrates how to do it and does not use any STL or some other external library.

Pernat1y - July 20, 2020, 2:15 pm

(Jul 17, 2020 03:33 AM)HiddenEyE Wrote: [\[CITE\]](#) The question is that how can one reduce the huge time interval involve in cracking like 24 password length. Get quantum computer. It helps.

Goodies - July 20, 2020, 3:35 pm

1 character is 128
2 characters is 128×128
3 characters is $128 \times 128 \times 128$

etc...

so it's just 128^n where n =the number of characters.

Every additional character is 2 orders of magnitude greater in terms of average time taken to crack.

HiddenEyE - July 21, 2020, 1:29 am

(Jul 20, 2020 03:35 PM)Goodies Wrote: [\[CITE\]](#) 1 character is 128

2 characters is 128×128

3 characters is $128 \times 128 \times 128$

etc...

so it's just 128^n where n =the number of characters.

Every additional character is 2 orders of magnitude greater in terms of average time taken to crack.
What are they work around for reducing time spent in cracking 24 password length

Goodies - July 21, 2020, 1:42 am

(Jul 21, 2020 01:29 AM)HiddenEyE Wrote: [\[CITE\]](#) What are they work around for reducing time spent in cracking 24 password length

Rainbowtables are the most efficient way of searching for previously cracked passwords, far more efficient than hashing/checking

iolaka - July 24, 2020, 10:53 pm

Are you talking about a web cracker or what? This question has no info.

Laggy - July 28, 2020, 12:46 pm

I'd assume you either need a more powerful computer or more efficient code. Every time you add another character to a string, it becomes exponentially more difficult to crack, hence why having a password of length 10 that consists of purely random characters is the strongest you can have, it'd take far too long for any brute force to crack.

ViRusRat - July 28, 2020, 2:05 pm

This question has no info.

Semmenial - July 28, 2020, 3:55 pm

A cracker is a cracker. It's gonna take time for it to compute all those hashes etc, you can't expect it to be fast unless you have a dedicated server for the sole purpose of cracking hashed passwords.

HiddenEyE - August 1, 2020, 11:02 pm

(Jul 21, 2020 01:42 AM)Goodies Wrote: [\[CITE\]](#) Rainbowtables are the most efficient way of searching for previously cracked passwords, far more efficient than hashing/checking

I will try that out. Thanks so far

(Jul 28, 2020 03:55 PM)Semmenial Wrote: [\[CITE\]](#) A cracker is a cracker. It's gonna take time for it to compute all those hashes etc, you can't expect it to be fast unless you have a dedicated server for the sole purpose of cracking hashed passwords.

I will consider this but actually need it done without spending some \$\$

Klonopin - August 2, 2020, 5:28 pm

Whats your computer's specs looking like, might effect the speed.

HRRHHRRH - August 6, 2020, 1:11 am

(Aug 2, 2020 05:28 PM)Klonopin Wrote: [\[CITE\]](#) Whats your computer's specs looking like, might effect the speed.

Before moving to changing your computer parts, it's important you fully optimize your program. I imagine OP is probably using strings, vectors, maybe recursion or he simply does not know enough about how the CPU works or a lot of other things that will greatly slow down the program to begin with.