

Forum:[Hack Forums](#)

Thread: [C++ Compile-time Encryption & Runtime Decryption w/ String Clearing](#)

Hackers - March 10, 2024, 9:31 am

This is a simple tool loosely based on skCrypter. It offers encryption of standard strings when compiling, with a decrypt function for runtime. You can then clear the string so it is completely emptied and no longer available in its encrypted or decrypted form.

Example of usage with visual message boxes:

Code #include <windows.h>

#include "include/CompileTimeCrypt.h"

```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {
```

```
    auto encryptedHello = encrypt("Hello, World!"); // Encrypts at compile time
```

```
    MessageBoxA(NULL, encryptedHello.decrypt(), "Decrypted HW Message 1", MB_OK); //
```

Decrypts at runtime

```
    encryptedHello.clear(); // this will write over / erase the original encrypted and decrypted strings
```

```
    MessageBoxA(NULL, encryptedHello.decrypt(), "Decrypted HW Message 2", MB_OK); //
```

because the message was cleared, the window is blank

```
    auto encryptedTestMessage = encrypt("Second Test!"); // Encrypts at compile time
```

```
    MessageBoxA(NULL, encryptedTestMessage.decrypt(), "Test Message 1", MB_OK); // Decrypts
```

at runtime

```
    MessageBoxA(NULL, encryptedTestMessage.decrypt(), "Test Message 2", MB_OK); // Expected
```

to return decrypted string again, because uncleared

```
    encryptedTestMessage.clear(); // this will write over / erase the original encrypted and decrypted
```

strings

```
    MessageBoxA(NULL, encryptedTestMessage.decrypt(), "Test Message 3", MB_OK); // because
```

the message was cleared, the window is blank

```
    MessageBoxA(NULL, encrypt("Inline String Test").decrypt(), "Inline String Test - One Liner", MB_OK); // Encrypts at compile & decrypts at runtime - oneliner
```

```
    // not recommended if you need to clear the data as there will not be a way to do so
```

```
    return 0;
```

```
}
```

Header File:

Code #pragma once

#pragma once

```
#include <cstdint> // For size_t
```

```
template <size_t N>
```

```
class EncryptedString {
```

public:

```
constexpr EncryptedString(const char (&str)[N])
    : encrypted_ {}, isEncrypted_(true), isCleared_(false) {
    for (size_t i = 0; i < N; ++i) {
        encrypted_[i] = str[i] ^ generateKey(i);
    }
}

const char* decrypt() const {

    if (isCleared_) {
        return ""; // Return an empty string if already cleared or decrypted.
    }

    if (!isEncrypted_) {
        return decrypted_;
    }

    for (size_t i = 0; i < N; ++i) {
        decrypted_[i] = encrypted_[i] ^ generateKey(i);
    }
    isEncrypted_ = false; // Mark as decrypted
    return decrypted_;
}
```

```
void clear() {
    if (!isCleared_) {
        // Overwrite both encrypted and decrypted data to ensure it's irrecoverable.
        for (size_t i = 0; i < N; ++i) {
            encrypted_[i] = '\0';
            decrypted_[i] = '\0';
        }
        isEncrypted_ = false;
        isCleared_ = true; // Mark as cleared to block decryption.
    }
}
```

```
bool isEncrypted() const {
    return isEncrypted_;
}
```

```
bool isCleared() const {
    return isCleared_;
}
```

private:

```
mutable char encrypted_[N]; // Mutable to allow clearing
mutable char decrypted_[N]{}; // Mutable for decryption
mutable bool isEncrypted_; // Tracks encryption status
mutable bool isCleared_; // Tracks whether data has been cleared
```

```
static constexpr char generateKey(size_t position) {
    return '\xAA' ^ static_cast<char>((position * 33) % 256);
}
```

```

    }
};

// Helper function to deduce the size of the string at compile time and create an EncryptedString
instance
template <size_t N>
constexpr EncryptedString<N> encrypt(const char (&str)[N]) {
    return EncryptedString<N>(str);
}

```

This is just the first prototype in this thread. I am working on this project here:
<https://github.com/LNodesL/Compile-Time-Encrypt-Runtime-Decrypt-in-CPP>

Give it a star / follow to receive updates for custom XOR improvements, AES-256, and function overloads for different types of strings, as well as size-required string types in an alternate encrypt(string, size) format.

PulseLink - April 5, 2024, 12:29 pm

I am looking into other ways to obfuscate rather than encrypt during runtime in c. I like the source of your crypter as well, especially the construction in memory.

I am going to test this out and debug, if you plan on adding XOR please let me know and I will gladly lend a hand, I have been trying to do xor on runtime and get it to work properly while maintaining a small stub size.

Although some of these methods could be modified and possibly used for polymorphism, as its being built during runtime

Hackers - April 10, 2024, 5:52 am

(Apr 5, 2024 12:29 PM)PulseLink Wrote: [\[CITE\]](#) I am looking into other ways to obfuscate rather than encrypt during runtime in c. I like the source of your crypter as well, especially the construction in memory.

I am going to test this out and debug, if you plan on adding XOR please let me know and I will gladly lend a hand, I have been trying to do xor on runtime and get it to work properly while maintaining a small stub size.

Although some of these methods could be modified and possibly used for polymorphism, as its being built during runtime

I do have some XOR examples, but I prefer aes > xor and AFAIK xor is not as secure. However, it's easy to use and I noticed that sometimes just using AES itself can cause a detection or at least suspicion.

Is there a reason you would want to obfuscate rather than encrypt?

nucleo_type09 - April 13, 2024, 2:31 pm

(Mar 10, 2024 09:31 AM)nodes Wrote: [\[CITE\]](#) This is a simple tool loosely based on skCrypter. It offers encryption of standard strings when compiling, with a decrypt function for runtime. You can then clear the string so it is completely emptied and no longer available in its encrypted or decrypted form.

Example of usage with visual message boxes:

```

Code #include <windows.h>
#include "include/CompileTimeCrypt.h"

```

```

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int

```

```
nCmdShow) {
```

```
    auto encryptedHello = encrypt("Hello, World!"); // Encrypts at compile time
```

```
    MessageBoxA(NULL, encryptedHello.decrypt(), "Decrypted HW Message 1", MB_OK); //
```

```
    Decrypts at runtime
```

```
    encryptedHello.clear(); // this will write over / erase the original encrypted and decrypted strings
```

```
    MessageBoxA(NULL, encryptedHello.decrypt(), "Decrypted HW Message 2", MB_OK); //
```

```
    because the message was cleared, the window is blank
```

```
    auto encryptedTestMessage = encrypt("Second Test!"); // Encrypts at compile time
```

```
    MessageBoxA(NULL, encryptedTestMessage.decrypt(), "Test Message 1", MB_OK); // Decrypts
```

```
    at runtime
```

```
    MessageBoxA(NULL, encryptedTestMessage.decrypt(), "Test Message 2", MB_OK); // Expected
```

```
    to return decrypted string again, because uncared
```

```
    encryptedTestMessage.clear(); // this will write over / erase the original encrypted and decrypted
```

```
    strings
```

```
    MessageBoxA(NULL, encryptedTestMessage.decrypt(), "Test Message 3", MB_OK); // because
```

```
    the message was cleared, the window is blank
```

```
    MessageBoxA(NULL, encrypt("Inline String Test").decrypt(), "Inline String Test - One Liner",
```

```
    MB_OK); // Encrypts at compile & decrypts at runtime - oneliner
```

```
    // not recommended if you need to clear the data as there will not be a way to do so
```

```
    return 0;
```

```
}
```

```
Header File:
```

```
Code #pragma once
```

```
#pragma once
```

```
#include <cstdint> // For size_t
```

```
template <size_t N>
```

```
class EncryptedString {
```

```
public:
```

```
    constexpr EncryptedString(const char (&str)[N])
```

```
        : encrypted_{}, isEncrypted_(true), isCleared_(false) {
```

```
        for (size_t i = 0; i < N; ++i) {
```

```
            encrypted_[i] = str[i] ^ generateKey(i);
```

```
        }
```

```
    }
```

```
    const char* decrypt() const {
```

```
        if (isCleared_) {
```

```
            return ""; // Return an empty string if already cleared or decrypted.
```

```
        }
```

```
        if (!isEncrypted_){
```

```
            return decrypted_;
```

```
        }
```

```
        for (size_t i = 0; i < N; ++i) {
```

```

        decrypted_[i] = encrypted_[i] ^ generateKey(i);
    }
    isEncrypted_ = false; // Mark as decrypted
    return decrypted_;
}

void clear() {
    if (!isCleared_) {
        // Overwrite both encrypted and decrypted data to ensure it's irrecoverable.
        for (size_t i = 0; i < N; ++i) {
            encrypted_[i] = '\0';
            decrypted_[i] = '\0';
        }
        isEncrypted_ = false;
        isCleared_ = true; // Mark as cleared to block decryption.
    }
}

bool isEncrypted() const {
    return isEncrypted_;
}

bool isCleared() const {
    return isCleared_;
}

private:
    mutable char encrypted_[N]; // Mutable to allow clearing
    mutable char decrypted_[N]; // Mutable for decryption
    mutable bool isEncrypted_; // Tracks encryption status
    mutable bool isCleared_; // Tracks whether data has been cleared

    static constexpr char generateKey(size_t position) {
        return '\xAA' ^ static_cast<char>((position * 33) % 256);
    }
};

// Helper function to deduce the size of the string at compile time and create an EncryptedString
instance
template <size_t N>
constexpr EncryptedString<N> encrypt(const char (&str)[N]) {
    return EncryptedString<N>(str);
}

```

This is just the first prototype in this thread. I am working on this project here:

<https://github.com/LNodesL/Compile-Time-Encrypt-Runtime-Decrypt-in-CPP>

Give it a star / follow to receive updates for custom XOR improvements, AES-256, and function overloads for different types of strings, as well as size-required string types in an alternate `encrypt(string, size)` format.

dude i also wrote an encryption class using c++ time ago, that's what it requires to encrypt and decrypt a string:

----- PSEUDO CODE -----

```
Code #include "lib/viper.cpp"
```

```
using namespace std;
using namespace ViperCipher;
```

```
int main(int argc, char **argv)
{
    string plain = "something to encrypt";
    Viper *viper = new Viper();

    string_view enc = viper->Encrypt(plain);
    string_view dec = viper->Decrypt(enc);

    std::cout << "Encrypted: " << enc << std::endl;
    std::cout << "Decrypted: " << dec << std::endl;

    delete viper;
    return EXIT_SUCCESS;
};
```

the class has a lot of features such as easy SSL key generation, hashing, crack ciphers, etc...
check it out on github, i also made a documentation for people who like documentations :)

and this is how you generate a pair of ssl public/private keys saving them either locally or in a file...

```
Code #include "lib/viper.cpp"
```

```
using namespace std;
using namespace ViperCipher;
```

```
int main(int argc, char **argv) {
    Viper *viper = new Viper();

    auto FlagLocal = ViperCipher::RSA_KEY_FLAG::SCRIPT_COLLECTOR; // save key/s locally
    auto FlagFile = ViperCipher::RSA_KEY_FLAG::FILE_COLLECTOR;    // save key/s to file

    viper->GenRsaPrivateKey("private.pem", Flag); // generate
    string_view get_key = viper->getPrivateKey(); // collect
    delete viper;
    return 0;
};
```

tell me what you think, i shared it because i see you also created something similar...

Hackers - April 14, 2024, 4:47 am

(Apr 13, 2024 02:31 PM)nucleo_type09 Wrote: [\[CITE\]](#) dude i also wrote an encryption class using c++ time ago, that's what it requires to encrypt and decrypt a string:

----- PSEUDO CODE -----

Code #include "lib/viper.cpp"

```
using namespace std;
using namespace ViperCipher;
```

```
int main(int argc, char **argv)
{
    string plain = "something to encrypt";
    Viper *viper = new Viper();

    string_view enc = viper->Encrypt(plain);
    string_view dec = viper->Decrypt(enc);

    std::cout << "Encrypted: " << enc << std::endl;
    std::cout << "Decrypted: " << dec << std::endl;

    delete viper;
    return EXIT_SUCCESS;
};
```

the class has a lot of features such as easy SSL key generation, hashing, crack ciphers, etc...
check it out on github, i also made a documentation for people who like documentations :)

and this is how you generate a pair of ssl public/private keys saving them either locally or in a file...

Code #include "lib/viper.cpp"

```
using namespace std;
using namespace ViperCipher;
```

```
int main(int argc, char **argv) {
    Viper *viper = new Viper();

    auto FlagLocal = ViperCipher::RSA_KEY_FLAG::SCRIPT_COLLECTOR; // save key/s locally
    auto FlagFile = ViperCipher::RSA_KEY_FLAG::FILE_COLLECTOR;      // save key/s to file

    viper->GenRsaPrivateKey("private.pem", Flag); // generate
    string_view get_key = viper->getPrivateKey(); // collect
    delete viper;
    return 0;
};
```

tell me what you think, i shared it because i see you also created something similar...

Nice, you should make your own thread and post the full source to the Viper class!

Thanks for sharing here though, I always appreciate seeing other examples, it'll help any of us be better the more we can compare & learn from eachother.

nucleo_type09 - April 14, 2024, 5:06 am

(Apr 14, 2024 04:47 AM)nodes Wrote: [\[CITE\]](#) Nice, you should make your own thread and post the full source to the Viper class!

Thanks for sharing here though, I always appreciate seeing other examples, it'll help any of us be better the more we can compare & learn from eachother.

i also created a thread on it but nobody seems to care tho haha, i think people here tend to be more

interested in python coding rather than c++,

Hackers - April 14, 2024, 10:16 am

(Apr 14, 2024 05:06 AM)nucleo_type09 Wrote: [\[CITE\]](#) i also created a thread on it but nobody seems to care tho haha, i think people here tend to be more interested in python coding rather than c++, I will go check it out now - and yeah, a lot more people are going to want a higher level solution, something that feels more manageable, perhaps is easier to setup, is more cross-platform, and is more common. Unfortunately this means higher level languages are always in more demand than lower level, but this phenomenon is also why lower level programmers make more money, can do more advanced things, etc. so it's a balance of pros & cons I guess

nucleo_type09 - April 14, 2024, 10:55 am

(Apr 14, 2024 10:16 AM)nodes Wrote: [\[CITE\]](#) I will go check it out now - and yeah, a lot more people are going to want a higher level solution, something that feels more manageable, perhaps is easier to setup, is more cross-platform, and is more common. Unfortunately this means higher level languages are always in more demand than lower level, but this phenomenon is also why lower level programmers make more money, can do more advanced things, etc. so it's a balance of pros & cons I guess
what do you prefer? lower or higher?

Hackers - April 14, 2024, 11:04 am

(Apr 14, 2024 10:55 AM)nucleo_type09 Wrote: [\[CITE\]](#) what do you prefer? lower or higher?
Lower for my long-term career plans, higher for small projects that don't require low level.
Depends on the project, team skills, etc.

For example at work I have brought in some of my C++. One was a utility tool for starting up some server processes. Another is to let a codebase have different versions of certain files, but shared files for the rest - like having site1 use 'page1-A' and site2 uses 'page1-B' but both share the same code for 'page2'. And then I did a couple small things, but other than that our company doesn't use C/C++ at all for anything other than what I've chosen to bring over in last few months. So if someone asked what language would be best for a project at my current job - probably always a high level language because no other devs mess around with low level except me currently, even if it would make sense to.

nucleo_type09 - April 14, 2024, 11:20 am

(Apr 14, 2024 11:04 AM)nodes Wrote: [\[CITE\]](#) Lower for my long-term career plans, higher for small projects that don't require low level.
Depends on the project, team skills, etc.

For example at work I have brought in some of my C++. One was a utility tool for starting up some server processes. Another is to let a codebase have different versions of certain files, but shared files for the rest - like having site1 use 'page1-A' and site2 uses 'page1-B' but both share the same code for 'page2'. And then I did a couple small things, but other than that our company doesn't use C/C++ at all for anything other than what I've chosen to bring over in last few months. So if someone asked what language would be best for a project at my current job - probably always a high level language because no other devs mess around with low level except me currently, even if it would make sense to.

c++ idk about others, but to me it just boosts my brain activity at the highest level because you know it requires you to think alot about what you writing, i like the feeling, that's why i prefer to use it over higher level haha,

did you worked with networking in c++? maybe some socket programming? what are your opinions about it compared to higher level langs when working with networking? are higher level langs faster to work with but maybe lower quality or just the same?

Hackers - April 14, 2024, 1:49 pm

(Apr 14, 2024 11:20 AM)nucleo_type09 Wrote: [\[CITE\]](#) c++ idk about others, but to me it just boosts my brain activity at the highest level because you know it requires you to think alot about what you writing, i like the feeling, that's why i prefer to use it over higher level haha,

did you worked with networking in c++? maybe some socket programming? what are your opinions about it compared to higher level langs when working with networking? are higher level langs faster to work with but maybe lower quality or just the same?

I only started writing C++ a few months ago, I'm still very new to it. I've done some networking things, but only through win API so far & common functions. I haven't gone too low level with anything networking related really, and haven't done too much with sockets. I had GPT whip up some examples for it, as well as some wininet functionality and they were easy to work with.

I think higher level langs can be faster to work with in many cases (for dev speed, not runtime speed) but if the project becomes more and more complicated, using a higher level lang may end up being a mistake in the long run.

Lately I'm a fan of lower > higher, at least for personal dev. Anything I do outside of work is in C/C++ for the most part, for the sole sake of learning more, but also the final programs are smaller & faster from what I've seen.

nucleo_type09 - April 14, 2024, 2:01 pm

(Apr 14, 2024 01:49 PM)nodes Wrote: [\[CITE\]](#) I only started writing C++ a few months ago, I'm still very new to it. I've done some networking things, but only through win API so far & common functions. I haven't gone too low level with anything networking related really, and haven't done too much with sockets. I had GPT whip up some examples for it, as well as some wininet functionality and they were easy to work with.

I think higher level langs can be faster to work with in many cases (for dev speed, not runtime speed) but if the project becomes more and more complicated, using a higher level lang may end up being a mistake in the long run.

Lately I'm a fan of lower > higher, at least for personal dev. Anything I do outside of work is in C/C++ for the most part, for the sole sake of learning more, but also the final programs are smaller & faster from what I've seen.

for me is a bit different, im not working in the field, actually, i don't really have a job yet, i didn't even try to get one so far, i always thought that if you intersect hobby and job, you get annoyed by both haha, from your experience working and hobbying with the same subject, how is it? does it feel better?

Hackers - April 14, 2024, 3:39 pm

(Apr 14, 2024 02:01 PM)nucleo_type09 Wrote: [\[CITE\]](#) for me is a bit different, im not working in the field, actually, i don't really have a job yet, i didn't even try to get one so far, i always thought that if you intersect hobby and job, you get annoyed by both haha, from your experience working and hobbying with the same subject, how is it? does it feel better?

You have to be careful bout burnout, but you also get double the efficiencies. And skill improvements over time. Imagine if I coded at work and knitted at home. I'd be good at coding and knitting, but if you code at work and code at home ... well, you get the point

itcorp - April 14, 2024, 3:50 pm

interesting approach bud,what would be interesting is to add and ensure compatibility across different platforms.

Nice share bud, I like how you are developing day by day.

nucleo_type09 - April 14, 2024, 4:12 pm

(Apr 14, 2024 03:39 PM)nodes Wrote: [\[CITE\]](#) You have to be careful bout burnout, but you also get double the efficiencies. And skill improvements over time. Imagine if I coded at work and knitted at home. I'd be good at coding and knitting, but if you code at work and code at home ... well, you get the point

bro lot of people are sending me popularity points, i DO really appreciate what u just did, it really shows how you people here in this community are well structured, but i think the job and hobby thing, it also depends don't u think? because CSS developers and system developers are both IT paths but the level on knowledge and advanced concepts implementation are on different sides, so yeah, i would say with honesty, i do thing working and hobbying with it is advantageous ... but it also depends a bit, don't u agree? :)

Hackers - April 15, 2024, 7:44 am

(Apr 14, 2024 04:12 PM)nucleo_type09 Wrote: [\[CITE\]](#) bro lot of people are sending me popularity points, i DO really appreciate what u just did, it really shows how you people here in this community are well structured

No problem! You can thank Benevolence group, the group is set up to spread kindness, encourage HQ content, etc!

(Apr 14, 2024 04:12 PM)nucleo_type09 Wrote: [\[CITE\]](#) but i think the job and hobby thing, it also depends don't u think? because CSS developers and system developers are both IT paths but the level on knowledge and advanced concepts implementation are on different sides, so yeah, i would say with honesty, i do thing working and hobbying with it is advantageous ... but it also depends a bit, don't u agree? :)

I do agree and there are times where I burn out.

What I do is I mostly write C/C++ and Python at home, and at work I'm mostly involved with JS based projects or websites/web apps/etc.

For me, this is enough of a difference

However, I also make sure to log off & close the computer too. I don't spend 100% of my freetime coding and if I'm burnt out from work I won't code at home. I have other hobbies too to balance things out & I spend time with my fiancé relaxing and doing non-tech things

nucleo_type09 - April 15, 2024, 8:13 am

(Apr 15, 2024 07:44 AM)nodes Wrote: [\[CITE\]](#) No problem! You can thank Benevolence group, the group is set up to spread kindness, encourage HQ content, etc!

I do agree and there are times where I burn out.

What I do is I mostly write C/C++ and Python at home, and at work I'm mostly involved with JS based projects or websites/web apps/etc.

For me, this is enough of a difference

However, I also make sure to log off & close the computer too. I don't spend 100% of my freetime coding and if I'm burnt out from work I won't code at home. I have other hobbies too to balance things out & I spend time with my fiancé relaxing and doing non-tech things

here another question, how do you find the relation with what you're doing and your fiancé? do u have time for both? or u split 50/50?

Hackers - April 28, 2024, 9:41 pm

(Apr 15, 2024 08:13 AM)nucleo_type09 Wrote: [\[CITE\]](#) here another question, how do you find the relation with what you're doing and your fiancé? do u have time for both? or u split 50/50? good, I make time for her but some days I should make more.

She is good at communicating when she feels I am not balanced enough and I usually instantly make a point of changing that when I can, especially if it's non-work stuff taking up my time that day/week.

nucleo_type09 - April 29, 2024, 5:01 am

(Apr 28, 2024 09:41 PM)nodes Wrote: [\[CITE\]](#) good, I make time for her but some days I should make more.

She is good at communicating when she feels I am not balanced enough and I usually instantly make a point of changing that when I can, especially if it's non-work stuff taking up my time that day/week.

thanks bro i appreciate you replied to that, im kind of similar situation, but im not so good a making time for both :(

mordock1998 - May 1, 2024, 9:49 am

(Apr 13, 2024 02:31 PM)nucleo_type09 Wrote: [\[CITE\]](#) dude i also wrote an encryption class using c++ time ago, that's what it requires to encrypt and decrypt a string:

----- PSEUDO CODE -----

```
Code #include "lib/viper.cpp"
```

```
using namespace std;
using namespace ViperCipher;
```

```
int main(int argc, char **argv)
{
    string plain = "something to encrypt";
    Viper *viper = new Viper();

    string_view enc = viper->Encrypt(plain);
    string_view dec = viper->Decrypt(enc);

    std::cout << "Encrypted: " << enc << std::endl;
    std::cout << "Decrypted: " << dec << std::endl;

    delete viper;
    return EXIT_SUCCESS;
};
```

the class has a lot of features such as easy SSL key generation, hashing, crack ciphers, etc... check it out on github, i also made a documentation for people who like documentations :)

and this is how you generate a pair of ssl public/private keys saving them either locally or in a file...

```
Code #include "lib/viper.cpp"
```

```
using namespace std;
using namespace ViperCipher;
```

```
int main(int argc, char **argv) {
    Viper *viper = new Viper();
```

```
    auto FlagLocal = ViperCipher::RSA_KEY_FLAG::SCRIPT_COLLECTOR; // save key/s locally
```

```

auto FlagFile = ViperCipher::RSA_KEY_FLAG::FILE_COLLECTOR;           // save key/s to file

viper->GenRsaPrivateKey("private.pem", Flag); // generate
string_view get_key = viper->getPrivateKey(); // collect
delete viper;
return 0;
};

```

tell me what you think, i shared it because i see you also created something similar...
good but this would not work for all systems, you should make it cross-platform, the unistd library in your header file will prevent other systems to use that, it will only work on linux, why don't you split it into SARBs?

Hackers - November 19, 2024, 12:51 pm

(May 1, 2024 09:49 AM)mordock1998 Wrote: [\[CITE\]](#) good but this would not work for all systems, you should make it cross-platform, the unistd library in your header file will prevent other systems to use that, it will only work on linux, why don't you split it into SARBs?
it's a shame you are closed, you brought up some good points.

If anyone else has questions, just chime in here or msg me! :)

Stanley - March 16, 2026, 8:20 am

I'm Stanley HF's helper bot. The OP (original poster) of this thread wants everyone to see this page and has elected to bump it to the top of the showthread listing. Since HF doesn't want a ton of LQP (Low Quality Posts) this is the best method of bumping your content. There is a small ?yte fee for each bump. Any OP can bump any of their threads on HF from the linked icon on the top of their first post.

If you haven't already read the first post of this thread we encourage you to do so.

Thanks,
Stanley

PELock - April 10, 2026, 2:15 am

Check out my project StringEncrypt you can automate polymorphic string encryption from VSCode extension generating polymorphic mess like this one

```

Code // encrypted with https://www.stringencrypt.com (v1.5.0) [C/C++]
// secret = "C++ polymorphic string encryption"
wchar_t secret[34] = { 0x14A8, 0x296B, 0xB88D, 0x63FD, 0x2133, 0xB99A, 0x6001, 0x8223,
                      0xD58B, 0x9A28, 0x336F, 0xD0D3, 0x2685, 0xCB2F, 0x4C1F, 0x2367,
                      0xD4B8, 0x3D74, 0x05C7, 0x0205, 0x7CE0, 0xB619, 0xB7EE, 0x97D7,
                      0x15D7, 0xB6F1, 0x9E74, 0xBC6A, 0x0AA4, 0x4236, 0x2A02, 0x1EBA,
                      0x3B08, 0x40CE };

for (volatile unsigned int fKzsw = 0, AFoyk = 0; fKzsw < 34; fKzsw++)
{
    AFoyk = secret[fKzsw];
    AFoyk = ((AFoyk & 0xFFFF) + 1) & 0xFFFF;
    for (unsigned int hHGVF = 0; hHGVF < 2; hHGVF++)
    {
        AFoyk = (AFoyk ^ ((AFoyk & 0xFFFF) >> 11)) & 0xFFFF;
        for (unsigned int xReOp = 0; xReOp < 2; xReOp++)

```

```

{
    AFoyk = (AFoyk ^ fKzsw) & 0xFFFF;
}
for (unsigned int DtJMT = 0; DtJMT < 4; DtJMT++)
{
    for (unsigned int USTcG = 0; USTcG < 4; USTcG++)
    {
        AFoyk = (AFoyk ^ (((AFoyk & 0xFFFF) << 11) & 0xFFFF)) & 0xFFFF;
    }
    AFoyk = (AFoyk ^ 0x66E3) & 0xFFFF;
}
}
AFoyk = ((AFoyk & 0xFFFF) - fKzsw) & 0xFFFF;
for (unsigned int vkCKD = 0; vkCKD < 3; vkCKD++)
{
    AFoyk = (AFoyk ^ 0xA7C6) & 0xFFFF;
    for (unsigned int ZTmjB = 0; ZTmjB < 2; ZTmjB++)
    {
        for (unsigned int QtWpG = 0; QtWpG < 3; QtWpG++)
        {
            AFoyk = ((AFoyk & 0xFFFF) + fKzsw) & 0xFFFF;
            AFoyk = (((~(AFoyk & 0xFFFF)) & 0xFFFF) + 1) & 0xFFFF;
        }
        AFoyk = (AFoyk ^ 0xCAC0) & 0xFFFF;
        AFoyk = ((AFoyk & 0xFFFF) + fKzsw) & 0xFFFF;
    }
}
}
AFoyk = (((AFoyk & 0xFFFF) >> 9) | ((AFoyk & 0xFFFF) << 7)) & 0xFFFF;
for (unsigned int hguLx = 0; hguLx < 2; hguLx++)
{
    AFoyk = ((AFoyk ^ 0xFFFF) & 0xFFFF);
    AFoyk = ((AFoyk & 0xFFFF) + 0x6883) & 0xFFFF;
    for (unsigned int CPkIM = 0; CPkIM < 3; CPkIM++)
    {
        AFoyk = ((AFoyk & 0xFFFF) + fKzsw) & 0xFFFF;
        AFoyk = (((AFoyk & 0xFFFF) << (fKzsw % 16)) | ((AFoyk & 0xFFFF) >> (16 - (fKzsw %
16)))) & 0xFFFF;
        for (unsigned int KZidk = 0; KZidk < 2; KZidk++)
        {
            AFoyk = (AFoyk ^ (((AFoyk & 0xFFFF) << 12) & 0xFFFF)) & 0xFFFF;
            AFoyk = ((AFoyk & 0xFFFF) + 0x1336) & 0xFFFF;
            AFoyk = (((AFoyk & 0xFFFF) << (fKzsw % 16)) | ((AFoyk & 0xFFFF) >> (16 - (fKzsw %
16)))) & 0xFFFF;
        }
    }
}
}
AFoyk = ((AFoyk & 0xFFFF) - fKzsw) & 0xFFFF;
for (unsigned int zBfSo = 0; zBfSo < 4; zBfSo++)
{
    AFoyk = ((AFoyk & 0xFFFF) + 0xF766) & 0xFFFF;
    for (unsigned int guGek = 0; guGek < 4; guGek++)
    {
        AFoyk = (AFoyk ^ ((AFoyk & 0xFFFF) >> 14)) & 0xFFFF;
    }
}
}

```

```

    }
    AFoyk = ((AFoyk & 0xFFFF) - 0x2D6D) & 0xFFFF;
}
for (unsigned int LiTeO = 0; LiTeO < 3; LiTeO++)
{
    AFoyk = (((AFoyk & 0xFFFF) << (fKzsw % 16)) | ((AFoyk & 0xFFFF) >> (16 - (fKzsw % 16))))
& 0xFFFF;
    AFoyk = ((AFoyk ^ 0xFFFF) & 0xFFFF);
}
AFoyk = (AFoyk ^ ((AFoyk & 0xFFFF) >> 11)) & 0xFFFF;
AFoyk = ((AFoyk & 0xFFFF) + 1) & 0xFFFF;
AFoyk = (((AFoyk & 0xFFFF) << (fKzsw % 16)) | ((AFoyk & 0xFFFF) >> (16 - (fKzsw % 16)))) &
0xFFFF;
    secret[fKzsw] = AFoyk;
}

```

Also its resistant to string extraction tools like flare-floss.