

Forum:[Hack Forums](#)

Thread: [Open-source Keylogger](#)

-Tachyon - June 12, 2017, 5:42 am

Hi guys,

Haven't posted in forever so I thought I might as well release this somewhere. I am busy working on an open-source keylogger in C++.

If anyone would like to have a look at it, fork it, or improve features please feel free.

[GitHub Source](#)

Enjoy ?

morph - June 12, 2017, 9:45 am

<http://en.cppreference.com/w/cpp/language/namespace>

[warn]

0xDEAD10CC - June 12, 2017, 10:56 am

Your GetOS function will not work reliably. I think you should read about the problems with GetVersionEx() ever since Windows 8/8.1/10... For what you're doing, there's a reason why Microsoft created a version API. You're also mixing ANSI with Unicode here, which is a very poor idea, and makes me think you took snippets from all over the place for these functions because you don't stay consistent.

(Jun 12, 2017 09:45 AM)n0ctua Wrote: [\[CITE\]](#)

<http://en.cppreference.com/w/cpp/language/namespace>

Providing a link by itself with no direction is hardly helpful. I would assume you're wanting him to look at the notes section about 'using namespace std'.

morph - June 12, 2017, 11:26 am

(Jun 12, 2017 10:56 AM)0xDEAD10CC Wrote: [\[CITE\]](#) Providing a link by itself with no direction is hardly helpful. I would assume you're wanting him to look at the notes section about 'using namespace std'.

Yes, sorry. I forgot to add that

Hoodie+ - June 12, 2017, 11:46 pm

(Jun 12, 2017 10:56 AM)0xDEAD10CC Wrote: [\[CITE\]](#) Your GetOS function will not work reliably... Call the ntdll.dll equivalent of the function and it should work normally.

At least it did when I coded shit.

TouchMyMalware - June 13, 2017, 4:12 am

Saw a memory leak, instantly bad.

maldev0 - June 13, 2017, 9:24 am

(Jun 12, 2017 11:46 PM)Hoodie+ Wrote: [\[CITE\]](#) Call the ntdll.dll equivalent of the function and it should work normally.

At least it did when I coded shit.

yes, RtlGetVersion works.

@OP

As touchmytralala stated you have a memleak:

<https://github.com/TachyonRSA/Axonode/blob/master/Axonode/Axonode/axonUtil.cpp#L76>

<https://github.com/TachyonRSA/Axonode/blob/master/Axonode/Axonode/Axonode.cpp#L101>

Edit:

also here

<https://github.com/TachyonRSA/Axonode/blob/master/Axonode/Axonode/axonUtil.cpp#L240>

<https://github.com/TachyonRSA/Axonode/blob/master/Axonode/Axonode/axonUtil.cpp#L279>

<https://github.com/TachyonRSA/Axonode/blob/master/Axonode/Axonode/axonUtil.cpp#L285>

the pointer to the allocated memory buffer is not freed in any way

OP I suggest you to learn how pointers and memory allocation actually work

Hoodie+ - June 13, 2017, 11:50 am

(Jun 13, 2017 09:24 AM)maldev0 Wrote: [\[CITE\]](#) yes, RtlGetVersionEx works.

@OP

As touchmytralala stated you have a memleak:

<https://github.com/TachyonRSA/Axonode/blob/master/Axonode/Axonode/axonUtil.cpp#L76>

<https://github.com/TachyonRSA/Axonode/blob/master/Axonode/Axonode/Axonode.cpp#L101>

the pointer to the allocated memory buffer is not freed in any way

Omg he calls free after return xD

Rageline - June 13, 2017, 12:41 pm

Dude please go back to books, you have mem leaks, you dont even use rval moves, you dont even know what ur doing.

Please learn what RAII/unique/shared ptr and how stuff is returned in C++ and what move semantics are before coding something. Whenever you make a function call you copy entire universe because u have 0 clue about anything beyond what WinAPI is. It's people like you who brought the quality of this section down to minecraft level

Rageline - June 13, 2017, 12:43 pm

Also 90% of this code is C, not C++, the only thing thats C++ is "cout"

Sajuuk - June 13, 2017, 2:54 pm

(Jun 13, 2017 12:43 PM)Rageline Wrote: [\[CITE\]](#) Also 90% of this code is C, not C++, the only thing thats C++ is "cout"

How'd you manage to doublepost between a few minutes? ?

Rageline - June 13, 2017, 3:25 pm

(Jun 13, 2017 02:54 PM)Sajuuk Wrote: [\[CITE\]](#) How'd you manage to doublepost between a few minutes? ?

One guy with girl avatar removed his post inbetween

Dostoevsky - June 13, 2017, 11:54 pm

(Jun 13, 2017 12:41 PM)Rageline Wrote: [\[CITE\]](#) Dude please go back to books, you have mem leaks, you dont even use rval moves, you dont even know what ur doing.

Please learn what RAII/unique/shared ptr and how stuff is returned in C++ and what move semantics are before coding something. Whenever you make a function call you copy entire universe

because u have 0 clue about anything beyond what WinAPI is. It's people like you who brought the quality of this section down to minecraft level

"rvalue moves"

What are you talking about?

It's actually pretty rare outwith move constructors (and move assignment operators) or in template functions (universal references) that you'd actually have an rvalue reference parameter. That syntax only exists to lessen ambiguity between the const reference syntax (where you can pass in an rvalue, temporary, and it will assume the lifetime of the immutable reference it's bound to). rvalue references simply make something a candidate for moving. What do you mean by "rvalue moves"? Most moves in reality will simply consist of you using `std::move` when passing to a by-value parameter (and, in some cases, copy elision will be performed implicitly - especially in the case of "how stuff is returned in C++" where RVO will happen on move-only types (and in some other scenarios - read the spec regarding the "triviality" of copying/moving objects) implicitly; hence why `unique_ptr` factories such as `make_unique` need not `std::move` on their return).

I understand your logic behind wanting to introduce move semantics but, after a small look over the code, he's mostly passing around pointers and handles which have little overhead. I seen a case where he's passing a `std::string` by-value but, even in that scenario, introducing move operations will likely still have reasonable overhead when you consider SSO. A move on a `std::string` that has been optimised (where string content is on the stack) will have the same overhead as copying. So, he should prefer passing by const reference in such scenarios.

It is true though that he should probably manage a lot of his memory using a smart pointer like `std::weak_ptr` where moving is required but saying "rval moves" is just vague guidance for OP (and easy to misinterpret).

@OP - there is a few memory leaks in your code and areas where I'm concerned for your understand of C++. For example, in one scenario you use `malloc()` for the destination of `GetWindowTextA` and, in another, you use the new syntax. Make up your mind and make sure you used `delete[]` when freeing. You'd be better off using smart pointers and making using of the syntax introduced in C++14 for creating arrays w/ `std::make_unique` factory (e.g. `std::make_unique<int[]>(length)`) then just making sure the lifetime of this smart pointer object is adequate when you inevitably `.get()` the underlying raw pointer to use it with WinAPI.

TouchMyMalware - June 14, 2017, 4:53 am

(Jun 13, 2017 11:54 PM)Dostoevsky Wrote: [\[CITE\]](#) @OP - there is a few memory leaks in your code and areas where I'm concerned for your understand of C++. For example, in one scenario you use `malloc()` for the destination of `GetWindowTextA` and, in another, you use the new syntax. Make up your mind and make sure you used `delete[]` when freeing. You'd be better off using smart pointers and making using of the syntax introduced in C++14 for creating arrays w/ `std::make_unique` factory (e.g. `std::make_unique<int[]>(length)`) then just making sure the lifetime of this smart pointer object is adequate when you inevitably `.get()` the underlying raw pointer to use it with WinAPI.

This guy is a legend. Listen to him.

Rageline - June 15, 2017, 12:10 am

(Jun 13, 2017 11:54 PM)Dostoevsky Wrote: [\[CITE\]](#) "rvalue moves"

What are you talking about?

It's actually pretty rare outwith move constructors (and move assignment operators) or in template functions (universal references) that you'd actually have an rvalue reference parameter. That syntax only exists to lessen ambiguity between the const reference syntax (where you can pass in an rvalue, temporary, and it will assume the lifetime of the immutable reference it's bound to). rvalue references simply make something a candidate for moving. What do you mean by "rvalue moves"? Most moves in reality will simply consist of you using `std::move` when passing to a by-value

parameter (and, in some cases, copy elision will be performed implicitly - especially in the case of "how stuff is returned in C++" where RVO will happen on move-only types (and in some other scenarios - read the spec regarding the "triviality" of copying/moving objects) implicitly; hence why `unique_ptr` factories such as `make_unique` need not `std::move` on their return).

I understand your logic behind wanting to introduce move semantics but, after a small look over the code, he's mostly passing around pointers and handles which have little overhead. I seen a case where he's passing a `std::string` by-value but, even in that scenario, introducing move operations will likely still have reasonable overhead when you consider SSO. A move on a `std::string` that has been optimised (where string content is on the stack) will have the same overhead as copying. So, he should prefer passing by const reference in such scenarios.

It is true though that he should probably manage a lot of his memory using a smart pointer like `std::weak_ptr` where moving is required but saying "rval moves" is just vague guidance for OP (and easy to misinterpret).

@OP - there is a few memory leaks in your code and areas where I'm concerned for your understand of C++. For example, in one scenario you use `malloc()` for the destination of `GetWindowTextA` and, in another, you use the new syntax. Make up your mind and make sure you used `delete[]` when freeing. You'd be better off using smart pointers and making using of the syntax introduced in C++14 for creating arrays w/ `std::make_unique` factory (e.g. `std::make_unique<int[]>(length)`) then just making sure the lifetime of this smart pointer object is adequate when you inevitably `.get()` the underlying raw pointer to use it with WinAPI.
ur right i could've written 500 words to OP what to do but when he would look at cppreference link of anything his brain would shut down permanently.

Dostoevsky - June 15, 2017, 3:02 am

(Jun 15, 2017 12:10 AM)Rageline Wrote: [\[CITE\]](#) ur right i could've written 500 words to OP what to do but when he would look at cppreference link of anything his brain would shut down permanently. True.

Still kills me inside when people make threads containing lots of WinAPI shit and label it under "C++" just because they use like one or two STL includes.

All these HF C++ developers who know `<iostream>`, `<string>`, and `<vector>`. C++ and STL experts.

Sajuuk - June 15, 2017, 8:38 am

(Jun 15, 2017 03:02 AM)Dostoevsky Wrote: [\[CITE\]](#) True.

Still kills me inside when people make threads containing lots of WinAPI shit and label it under "C++" just because they use like one or two STL includes.

All these HF C++ developers who know `<iostream>`, `<string>`, and `<vector>`. C++ and STL experts. I'm sitting here giving advice and posting links trying to keep in mind how I once was the same way. ?

-Tachyon - June 18, 2017, 11:31 pm

(Jun 13, 2017 11:54 PM)Dostoevsky Wrote: [\[CITE\]](#) @OP - there is a few memory leaks in your code and areas where I'm concerned for your understand of C++. For example, in one scenario you use `malloc()` for the destination of `GetWindowTextA` and, in another, you use the new syntax. Make up your mind and make sure you used `delete[]` when freeing. You'd be better off using smart pointers and making using of the syntax introduced in C++14 for creating arrays w/ `std::make_unique` factory (e.g. `std::make_unique<int[]>(length)`) then just making sure the lifetime of this smart pointer object is adequate when you inevitably `.get()` the underlying raw pointer to use it with WinAPI.

(Jun 15, 2017 08:38 AM)Sajuuk Wrote: [\[CITE\]](#) I'm sitting here giving advice and posting links trying

to keep in mind how I once was the same way. ?

Thank you for the feedback guys, I do realize it is pretty shit at the moment. Still quite new to C++, learning as I go. Criticism is always appreciated, only way to learn ?