

Forum:[Hack Forums](#)

Thread: [\[C++\] Prevent user from closing program \[White hat\]](#)

Xerif - April 1, 2014, 12:02 pm

Hello HF!

I'm making a parental control program, and I need to prevent the user from simply closing it through task manager.

I'm using QT for the GUI. So far I overrode the `closeEvent` function and made it do nothing. That works if you press the close button(s), but you can easily close the program through task manager.

I searched the internet and found some solutions such as hiding the process from the task manager. But from what I read, it seems that it's a difficult thing to do and considered as very malicious.

Are there other methods of preventing the user from closing the program via task manager?

Thanks for the help! ?

Feminist - April 1, 2014, 12:47 pm

No, there is no easy way of preventing the user from killing your process with task manager. You could set it to a critical process, causing a BSOD upon termination (couple of lines of code).

But you are "white hat" so why even do this? It seems like something that does not need to happen.

Natecat33 - April 1, 2014, 1:10 pm

(Apr 1, 2014 12:47 PM)Feminist Wrote: [\[CITE\]](#) No, there is no easy way of preventing the user from killing your process with task manager. You could set it to a critical process, causing a BSOD upon termination (couple of lines of code).

But you are "white hat" so why even do this? It seems like something that does not need to happen. Make two processes, so that one will start the other if it is killed.

jessman420 - April 1, 2014, 1:23 pm

Since it is a white hat program instead of re-spawning the process if killed, maybe log the event, time, logged in user, system state, open windows, and email the parent, and perhaps even force a log off or lock a screensaver. Then they can discipline their kids so they will not close the program again.

This would required either a second monitoring process, or an injected dll, or remote thread. The monitoring process is more whitehat and will cause less issues in a production environment for your purpose.

Just my thoughts

Xerif - April 1, 2014, 10:15 pm

(Apr 1, 2014 12:47 PM)Feminist Wrote: [\[CITE\]](#) No, there is no easy way of preventing the user from killing your process with task manager. You could set it to a critical process, causing a BSOD upon termination (couple of lines of code).

But you are "white hat" so why even do this? It seems like something that does not need to happen. Thanks for the help, but it seems like a person wouldn't want to use a program that intentionally causes a blue screen.

(Apr 1, 2014 01:10 PM)Natecat33 Wrote: [\[CITE\]](#) Make two processes, so that one will start the other if it is killed.

This seems like a good idea! I'm wondering though, are you able to kill 2 process at the same time in task manager?

When I tried to do it, it task manager wouldn't let me select 2 process at the same time.

(Apr 1, 2014 01:23 PM)jessman420 Wrote: [\[CITE\]](#) Since it is a white hat program instead of re-spawning the process if killed, maybe log the event, time, logged in user, system state, open windows, and email the parent, and perhaps even force a log off or lock a screensaver. Then they can discipline their kids so they will not close the program again.

This would required either a second monitoring process, or an injected dll, or remote thread. The monitoring process is more whitehat and will cause less issues in a production environment for your purpose.

Just my thoughts

Logging what is happening at the time the program exits is a great idea! I'll do some research on dll injection and remote threads and see if they would be a good option. I'm thinking a second monitoring process would be my best bet though :D

Thanks everyone for the help! If anyone has any other ideas, please post them here ?

Goodies - April 1, 2014, 10:21 pm

(Apr 1, 2014 10:15 PM)Xerif Wrote: [\[CITE\]](#) This seems like a good idea! I'm wondering though, are you able to kill 2 process at the same time in task manager?

When I tried to do it, it task manager wouldn't let me select 2 process at the same time.

I run linux entirely so I'm not 100% sure when I say this, but I am almost sure than "end process tree" would work on windows 8 (but not windows 7, I BELIEVE). This has to do with some sort of pseudo-virtualization with windows 8...

Masm4ever - April 1, 2014, 10:23 pm

(Apr 1, 2014 12:02 PM)Xerif Wrote: [\[CITE\]](#) I'm making a parental control program, and I need to prevent the user from simply closing it through task manager.

So first thing kids learn in school nowadays is how to kill a process in task manager? Interesting?

Xerif - April 1, 2014, 11:28 pm

(Apr 1, 2014 10:21 PM)Goodies Wrote: [\[CITE\]](#) I run linux entirely so I'm not 100% sure when I say this, but I am almost sure than "end process tree" would work on windows 8 (but not windows 7, I BELIEVE). This has to do with some sort of pseudo-virtualization with windows 8...

Oh okay I see. Thanks!

(Apr 1, 2014 10:23 PM)Masm4ever Wrote: [\[CITE\]](#) So first thing kids learn in school nowadays is how to kill a process in task manager? Interesting? Lol yes, we also learn how to exploit PayPal and get free money ?

Masm4ever - April 2, 2014, 2:39 am

(Apr 1, 2014 11:28 PM)Xerif Wrote: [\[CITE\]](#) (Apr 1, 2014 10:23 PM)Masm4ever Wrote: [\[CITE\]](#) So first thing kids learn in school nowadays is how to kill a process in task manager? Interesting?

Lol yes, we also learn how to exploit PayPal and get free money ?

OK - assuming this is really a legitimate 'parental control' issue:

There are two ways to kill an app from task manager:

- End Task
- End Process

The latter is difficult to block, but the kids need to know the exact name of the process. You can play with that observation ;-)

Nichole - *April 2, 2014, 4:11 am*

You could easily prevent a user from killing the program.

I don't really know or care for the difference between white hat or black hat, it's just slang people use to boost their self esteem.

If you wanted to keep your application alive, you could hook `NtTerminateProcess` inside the Task Manager process.

Or you could hook it globally, so any process that calls `NtTerminateProcess` will fail. You could also hook `NtQuerySystemInformation` to hide your process completely. You could even hook `NtOpenProcess` to prevent the user from grabbing a handle.

Hooking isn't hard, injecting isn't hard. If you really wanted to do it, then drop the whole white hat black hat attitude.