

Forum:[Hack Forums](#)

Thread: [C++ Crypting Project | Diving Into AV/EDR Evasion](#)

OopsIsharded - April 27, 2025, 12:47 am

Hi! I'm going to use this thread to document my learning on crypting and AV/EDR evasion techniques in C++. Going to post my thoughts, ideas, progress, some code snippets as I work my way through building a crypter. You're welcome to comment ideas, methods, or techniques I should explore/consider. I am purely focusing on the evasion side of things currently, I will not be implementing quality of life features like persistence, auto runs etc. Those parts are final touches that I might decide to implement along with a GUI in the end.

For now I've been learning C++ as I am not fluent and the windows API itself. I've been practicing some simple techniques like walking the PEB, process injection, encrypting/decrypting payloads from memory. I'm also reading papers on Hell's Gate and its forks, and syswhispers, writing my own assembly sys call stubs and dynamically finding syscall numbers/unhooking AV's userland hooks. At some point I am going to learn more about compile time functions so I can make the build process more dynamic and prevent signature detection over time. There's a lot of things on my mind and I go from one to another all the time and can overload myself with this information. I have viewed a few threads on here with a lot of great information and some practical open source examples, and also a lot of red team blogs which go into good details on what AV/EDR is looking for in a malicious executable/process.

That's all for now, hopefully I'll have lots more to share and by the end of it a good understanding and an easily modifiable base for crypter that I can build on for future projects.

bigcriminal1337 - April 27, 2025, 5:53 am

Traditional AV's beside maybe 2 (Bitdefender etc.) do not use Ring3 hooks. So syscall is redundant for such work, better do some dynamic call to direct API (AVG / Avast use kernel to detect syscalls - <https://the-deniss.github.io/posts/2022/12/08/hooking-system-calls-in-windows-11-22h2-like-avast-antivirus.htm>)

EDR's on the other hand rely like 60%-90% on ring3 hooks. Most have SYSCALL detections, be it via ETW or another gay measure, like Instrumentation callback, some even have kernel protection, so using System calls (With public garbage methods like Hells) will only lead to more DETECTS. (Then direct API call) - so why?

Unhooking is useless, most EDR's protect against it and will only lead to more flag.

In my opinion you are focusing on things that are deeply irrelevant at your current stage. Focus first on how to create a FILE that does not get classified by AI engines as "suspicious" in traffic, downloadable on chrome/edge reliably. Every EDR/AV that cares nowadays has an AI layer that classifies files. Sentinel one, WD, Crowdstrike etc..

WD sometime even FLAGS real files as malicious. This will be your biggest battle, this alone can take a month of research, probably few of months. 99% of crypters in market FAIL here.

This is 99% more harder then a crap SYSCALL mechanism that everybody and their mother know how to do. No tutorials here ;)

You can be a great king in "EVASION" but if AI classifies your file as "suspicious", your entire work goes to garbage.

I am not even starting to talk about what happens after your file gets classified? How do you

remove detection in least amount of work? This things are 100x harder then any specific AV/EDR evasion you can think of.

OopsISharted - April 27, 2025, 12:55 pm

(Apr 27, 2025 05:53 AM)bigcriminal1337 Wrote: [\[CITE\]](#) Traditional AV's beside maybe 2 (Bitdefender etc.) do not use Ring3 hooks. So syscall is redundant for such work, better do some dynamic call to direct API (AVG / Avast use kernel to detect syscalls - <https://the-deniss.github.io/posts/2022/12/08/hooking-system-calls-in-windows-11-22h2-like-avast-antivirus.htm>)

EDR's on the other hand rely like 60%-90% on ring3 hooks. Most have SYSCALL detections, be it via ETW or another gay measure, like Instrumentation callback, some even have kernel protection, so using System calls (With public garbage methods like Hells) will only lead to more DETECTS. (Then direct API call) - so why?

Unhooking is useless, most EDR's protect against it and will only lead to more flag.

In my opinion you are focusing on things that are deeply irrelevant at your current stage. Focus first on how to create a FILE that does not get classified by AI engines as "suspicious" in traffic, downloadable on chrome/edge reliably. Every EDR/AV that cares nowadays has an AI layer that classifies files. Sentinel one, WD, CrowdStrike etc..

WD sometime even FLAGS real files as malicious. This will be your biggest battle, this alone can take a month of research, probably few of months. 99% of crypters in market FAIL here.

This is 99% more harder then a crap SYSCALL mechanism that everybody and their mother know how to do. No tutorials here ;)

You can be a great king in "EVASION" but if AI classifies your file as "suspicious", your entire work goes to garbage.

I am not even starting to talk about what happens after your file gets classified? How do you remove detection in least amount of work? This things are 100x harder then any specific AV/EDR evasion you can think of.

That's fair, I was reading papers on how EDR detects syscalls and I agree that the public methods would be detected as one paper I read was on how they are detected and that its a dead method. Though there was a few ideas it had that seemed feasible. I know I am most likely way off track but its just how I learn and understand, of course if my sys call implementation is getting it more detections than normal I would obviously revisit it and use other methods. I did do some test runs on kleenscan and I do agree that certain sys call implementations are getting detected more than just using the direct API call.

I will see about making a minimal example and see what edge/chrome says about it.

bigcriminal1337 - April 27, 2025, 8:52 pm

(Apr 27, 2025 12:55 PM)OopsISharted Wrote: [\[CITE\]](#) That's fair, I was reading papers on how EDR detects syscalls and I agree that the public methods would be detected as one paper I read was on how they are detected and that its a dead method. Though there was a few ideas it had that seemed feasible. I know I am most likely way off track but its just how I learn and understand, of course if my sys call implementation is getting it more detections than normal I would obviously revisit it and use other methods. I did do some test runs on kleenscan and I do agree that certain sys call implementations are getting detected more than just using the direct API call.

I will see about making a minimal example and see what edge/chrome says about it.

I suggest forgetting about Kleenscan and stuff like that, it is useless. Kleenscan is doing a scantime check, that is the most basic check that AV's perform since like 2004, a simple sig scan.

Nowdays AV's are a lot smarter and they have emulation engines that can execute payload without "running it", they use it as a flag mechanism, even if your stuff is 100% FUD on kleenscan, it can be 100% detected runtime.

One example of a very strong emulation engine - Avira + ESET.

Walking the PEB/ Running syscall not from NTDLL will get u an instant flag on an emulator, that is why in most cases you must break it.

I am not saying "Syscall" is useless as a whole, but I think you are wasting a lot of time on non relevant stuff at your current stage.

Now of course it is 1337 to do a "syscall", but if you are serious on your goals, I would switch priorities.

I talk form experience, my work uses 0 syscalls and I pass ALL runtime + scantime (All EDR's and AV's) - so obviously, not needed.

The first step in a crypter is creating the stub that will hold the payload, if you can't get it to bypass common AI engines, you are wasting your time, adding "Syscalls", or any other stuff won't help you bypass it.

I will instead focus on:

- Creating the stub
- Invest time into breaking emulators - code it.
- Invest time & research how to hold file offline in stub - code it
- Decide on encryption that keeps entropy intact - code it
- Verify your work passes AI engine of WD - This is basic, chrome and edge is much harder to pass and personally took me few months.
- Make it execute something - be it runpe/loadpe or any custom loading method
- If until this point everything works, start working on poly engine to morph code - This will allow you to create 100% working crypt in 5 minutes.