Weird EDR Bypass's

Shhhhh secret stuff

Who am I

Fawaz Almutairi

Security researcher and red teamer in the banking sector

Twitter: @Q8fawazo

What is an EDR?

Endpoint Detection and Response - An endpoint protection that combines real-time continuous monitoring, data analytics and rule based automated response.

https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-endpoint-detection-and-response/

How does it work?

EDRs mostly work by hooking into specific Windows APIs, then depending on which API endpoints are being called it will estimate if a process if legitimate or malicious

EDRs will also have specific detection rules, for example if process A.exe was spawning process svchost.exe without being a service, then that is a red flag and an EDR will flag it as a malicious action and will respond accordingly to how the EDR is configured.

Weird stuff

Disclaimer:

- I did not reverse engineer any of the EDRs used for testing.
- I did not share any of the findings due to EDR companies being famous for retaliation.
- To call any Windows API I use PIC (Position Independent Code) hashed with DJB2 (I will share examples at the end)
- No EDR will be named (kinda)

Most of my findings have either been:

1. Logical
2. Brute Forced
3. Stupid

What did I do?

Well for starters the whole Idea was about creating a Matrix of Remote Process Injection methods that actually work on different EDR solutions

So I created my own Stage0 C2 which does that by providing it shellcode and the loading method, it will use different techniques to inject shellcode and after trial and error I can come to a conclusion regarding which methods work and which don’t against each EDR

AirStrike

Refresh Clear Download Configurations Logout

Available Sessions

ID	Status	User	Domain	Machine	Process	Version	Arch	PID	Last Seen	IP	EDR	Actions
1	Alive	fawaz	DESKTOP-[redacted]	DESKTOP[redacted]	[redacted]ca6f8efc390043d7ac9f0ac4b00679a6.exe	Windows 10 Pro	AMD64	61300	2 seconds	37.[redacted]	X	Kill / Load

Load

- Sideload
- Self
- Thread
- Com Hijack
- APCQueue
- EarlyBird
- ModuleStomp
- DllStomp
- ENV hijack

EDR (1)

Even though this EDR is one of the industries top products and most security engineers swear by it.

It has a fatal flaw when it comes to generating alerts from API calls with “Delays” between them

---

In theory you can do a plain and simple CreateRemoteThread injection while adding various Sleep(1000) API calls between each API call and achieve a remote code injection to another process without causing any alerts!

EDR (2)

this EDR is also one of the industries top products, It gained a lot of fame during the last couple of years.

It also has a fatal killer flaw that completely blinds alerts from this EDR causing you to spawn processes or inject remote shellcode without detection

---

The flaw is by patching the EtwEventWrite Windows API (ETW patching)

The EDR will stop any userland operations when it comes to that process, meaning it’s free for all!

PS: For remote code injection you need to patch ETW for the remote process too

EDR (3)

This is the default EDR most companies go to medium or big
It’s very good, scary good.

It uses ETW, ETWTi, AMSI and other kernel structures

The trick is simple! COM Hijacking
What is COM hijacking? well that is a big subject that I can’t cover right now, but think of it as registering a new (malicious) extension to an existing Microsoft software or registering a non-found extension
PS: This will only trigger after the computer is restarted and can be considered a type of persistence too

EDR (X)

One of the previously mentioned EDRs (no numbers) can also be unhooked.

By unhooking I mean removing the hooks installed by the EDR on a syscall.

That can be done by patching the syscall ID by comparing the “live” version of a DLL to the original unpatched state of the DLL.

Doing so can also blind an EDR from specific actions that you take

PS: This has to be done on remote process also.

EDR (*)

All the mentioned EDRs suffer from the same problem, which is self injection.

By self injection I mean retrieving malicious code either via reading a file or downloading shellcode etc...

Then loading it straight into the memory of the current process without injecting it into another process.

This is the best safe bulletproof solution to injection but it suffers from the ability to persist and defenders can easily identify this weird process making internet connection.

Thank you!