

Forum:[Hack Forums](#)

Thread: [AVs detecting the encrypted payload](#)

myselfandi - July 20, 2019, 9:09 am

My sample keeps getting detected as Gen:Trojan.Heur.FU, by 10/12 of AVs

I tried to put the encrypted payload as resource, as another section or as an array of bytes each they get detected by at least 10 AVs in all three cases,

I removed the injection and decryption functions from the executable, so there is no way that the emulator could've found my decryption function nor the key, and the detection rate doesn't change. But when i didn't include the encrypted payload the detection drops to 2 AVs detection. from that I concluded that they are detecting my sample through just static analysis, as of that I tried changing the encryption algorithm XOR+ADD, RC4 same, I tried also messing up with size of the executable, changing the payload, but that doesn't seem to effect the detection.

I'd like to hear you thoughts about what might giving me up. or any other ideas to try out. thanks.

Legal Coding - July 20, 2019, 9:41 am

good xor now more secure nice work

Friedrich Nietzsche - July 20, 2019, 9:44 am

The problem is most likely entropy.

Jean-Jacques Rousseau - July 20, 2019, 4:59 pm

(Jul 20, 2019 09:44 AM)Friedrich Nietzsche Wrote: [\[CITE\]](#) The problem is most likely entropy. How would you go to remove these detections?

Friedrich Nietzsche - July 20, 2019, 9:37 pm

(Jul 20, 2019 04:59 PM)Supercut Wrote: [\[CITE\]](#) How would you go to remove these detections? By reducing the overall entropy of your file?

Twistd - July 20, 2019, 10:08 pm

This is a behavior detection aka heuristic. Entropy is very unlikely, never had any problems with that even tho my Entropy levels were through the roof. Anyways, it could be your IAT or codeflow you can try to rearrange the general codeflow of the stub. Other than that I would suggest that you start commenting code to figure out what's causing it like any other regular detection. Adding a couple fake api calls may work, adding a string table to resources may also help in some situations. Just rip a couple tables from another file, most executables has one.

I'm talking from a native perspective though, .Net may be different. However, if you really want to reduce the entropy of your file just encode your payload with something like ascii85 after encryption.

myselfandi - July 21, 2019, 6:20 am

(Jul 20, 2019 10:08 PM)Twistd Wrote: [\[CITE\]](#) This is a behavior detection aka heuristic. Entropy is very unlikely, never had any problems with that even tho my Entropy levels were through the roof. Anyways, it could be your IAT or codeflow you can try to rearrange the general codeflow of the stub. Other than that I would suggest that you start commenting code to figure out what's causing it like any other regular detection. Adding a couple fake api calls may work, adding a string table to resources may also help in some situations. Just rip a couple tables from another file, most

executables has one.

I'm talking from a native perspective though, .Net may be different. However, if you really want to reduce the entropy of your file just encode your payload with something like ascii85 after encryption. Like I said i removed the decryption and injection functions and the detection doesn't change, so at the moment my executable walk the doubly linked list and parse ntdll.dll and kernel32.dll by their hashes and set their base addresses into a global variables and then create mutex(CreateMutex) by walking the export section of the module and parse the the function by comparing the hashes. thus the IAT is pretty much empty.

regarding commenting code, that basically what I did to end up at the conclusion that the encrypted payload is being detected rather than the STUB. (re-read my post please)

Yes, I'm talking about native, anyway, thanks I'll try to implement your ideas.

Twistd - July 21, 2019, 6:58 am

(Jul 21, 2019 06:20 AM)myselfandi Wrote: [\[CITE\]](#) regarding commenting code, that basically what I did to end up at the conclusion that the encrypted payload is being detected rather than the STUB. (re-read my post please)

Yes, I'm talking about native, anyway, thanks I'll try to implement your ideas.

Ok im going to say this once more, heuristic = behavior detection and its all about codeflow. Yes i know you just said that you removed shit that will prevent the malware from executing but just because its a heuristic detection doesn't mean its because of emulation. Your blob of encrypted data can trigger red flags in combination with others and eventually get you detected.

A gen as you all seem to enjoy calling it, is in fact a code detection, or in other words an heuristic/behaviour detection. Because that's the right word for it.

Most people here add a huge icon to its resources and blame the icon, which is kinda funny. You just happen to add size(ico) to the file which makes the resource data larger and sometimes makes the file unrecognizable to its static behavior analysis. The built in behavior detection analyzer recognizes and matches known malware patterns.

Example: Bitdefender Antivirus Plus - Gen:Trojan.Heur.FU.hxW@aAzHytay

Like I said -> heuristic.

Quote: Heuristic analysis is a method employed by many computer antivirus programs designed to detect previously unknown computer viruses, as well as new variants of viruses already in the "wild".[1]

Heuristic analysis is an expert based analysis that determines the susceptibility of a system towards particular threat/risk using various decision rules or weighing methods. MultiCriteria analysis (MCA) is one of the means of weighing. This method differs from statistical analysis, which bases itself on the available data/statistics.

[Heuristic analysis - Wikipedia](#)

You can fud a gen detection like any other detection by just changing the code flow a little.

It's not like some magic detection that happens just because you got a "non-HQ" icon like so many here likes to think..

If you struggle with gen.heuristic detections, you can sometimes make the code look different and avoid triggering a detection by adding an icon or other random resources.

As long as you use proper assembly you shouldn't even need an icon attached to your file to get rid

of heuristics.

There are however some antiviruses that detect icons of course(especially Avira), but not in this case.

In my builder i wrote some shit that generated random version information, which was all a waste, bifdefender engine didn't like it at all and the heuristics kicked in hard so i had to clone file every time. Splitting it up in to several parts works good and so does adding some random string tables.

So heres a couple pointers for you:

1. Codeflow
2. Assembly information (CRUCIAL)
3. Adding some random resources like a string table can be good.
4. Splitting the payload resource
5. Change storage? MAKEINTRESOURCE(20) and up to 24 i recall were good. (Don't use your own string name on the resources.)
6. Encoding your payload with ascii85 will make your encrypted payload datas look less suspicious and lower your entropy levels, worth trying.