

Forum:[Hack Forums](#)

Thread: [malware indicators with pestudio](#)

ttbr0 - April 21, 2025, 1:43 am

malware indicators with pestudio

this is a post for those who are interested in malware analysis. it will involve the pestudio tool which is for analyzing and triaging potential malicious indicators that an executable file might have and explain what they are and how they are relevant to malware but also if they can be false positives. this is a great tool to get a high level overview of executable files to quickly see if there are obvious signs that something is malicious. in this example, i will use a reverse tcp meterpreter payload.

indicators

the first section is the high level indicators. this is good to get a general overview of the file.

file > info: the relevant part here is the entropy value. entropy refers to shannons entropy which measures the "randomness" in a set of data or bytes. the higher the value (max of 8, but realistically you will see a high 7) the more "random" it is. this implies that there is either compressed or encrypted (or both) data which can infer packed malware, especially for crypters, packers, loaders, stagers, etc. this will be expanded in each of the pe file sections.

virustotal > score: shows the rating of the file on virustotal. this is only valid if the file has been uploaded before, pestudio will request the hash of the file from virustotal. this is important because uploading a file during an incident is bad practice because you do not want to alert the threat actors. in my example, i have no internet connection so there is no score.

resource: pe file resources are not present in this sample. resources can contain additional packed data that the executable can use such as icons, images and most importantly, binary data. malware can hide packed payloads in their resource sections. combine this with entropy analysis for each resource to discover potential second stage malware.

entry-point > location: this is where the executable starts execution after process initialization. this value should be in the standard .text code section. if it points anywhere outside of it, it is likely suspicious. in this example, there is a .ntnp section. this is incredibly suspicious because this is not within the .text section and it is not a standard section name.

sections > self-modifying: a section could be marked as both writable and executable meaning that it could rewrite its own code on runtime, hence self-modifying. this can indicate hidden code that can potentially be malicious. packers such as upx have self-modifying sections and it does not inherently mean that it is bad. this is usually a red flag depending on the context.

certificate: if this is a valid and from a well known signer, like from microsoft or google, the file is very likely a known good. compromising the integrity of cryptographically signed data while remaining valid requires a 0day and is very very unlikely to happen. if, however, there is a certificate and it is not valid or does not properly sign the executable, it could be suspicious as it could be trying to appear valid. but just because there is a valid signed file does not mean it is not malicious.

imports > flag: suspicious imported windows api functions. its important to realize that a lot of functions that are "suspicious" can be normal. in this example, virtualalloc is marked as suspicious but a majority of executables use this legitimately because it simply allocates memory to hold data. you should approach these suspicious imports with a bigger picture checking to see if there are multiple imports that are potentially malicious like seeing createprocess, ntunmapviewofsection, virtualallocex, writeprocessmemory, setthreadcontext and resumethread together. some other imports that are legitimate but attributed to malicious behavior includes getprocaddress, isdebuggerpresent, virtualallocex, readprocessmemory, writeprocessmemory, openprocess. there are a few imports that immediately stand out than others such as createreomotethread,

queueuserapc and nt/zwunmapviewofsection.

imphash > md5: the import hash is a hash of the import functions. it can be used to fingerprint similar executables independent of recompilations or other code transformations that would otherwise change other static signatures.

exports: this shows exported functions which is code that can be imported by other executables. imported windows api functions are exported functions in dlls like kernel32.dll and ntdll.dll. malware can export functions and then execute it with rundll32.exe or it may be used with another packaged malware. normal standalone programs don't usually have exports.

overlay: the overlay is additional data appended to the executable and is not part of the pe format. malware may contain overlays to bind secondary stages that are dropped or loaded on runtime.

sections

the pe file format has sections which are used to contain categories of information. each of these sections has its own metadata header that describes its attributes and then there is the section itself which contains the data.

name: the standard ones are .text (code), .data (global variables), .rdata (read only data), .idata (import data), .rsrc (resources) and .tls (thread local storage) but there are some more. there are also section names that can indicate transformations done on the file like upx packing or vmprotect protection. these are not inherently malicious but any names that are not well known such as .ntnp raises red flags immediately.

entropy: as mentioned before, each section has its own entropy score as well. if it is above 7, it is very likely compressed or encrypted.

characteristics: each section should have expected characteristics. code sections should be executable and data sections should be non executable, but readable and/or writable. it makes no sense that any code section should be both executable or writable unless it is self-modifying. self-modifying code is not normal and should raise some red flags. however, it is possible for code to modify its own code sections without the executable or writable characteristics because this can change on runtime with writeprocessmemory or using virtualprotect to modify its page permissions. pestudio also flags the .text section as executable which is completely expected and normal. i'm not sure why this exists, it is a false positive.

entry-point > location: the entry point should be located within the .text section. anything outside is not normal.

libraries

libraries are the executables that are loaded into the process on runtime. these libraries contain additional code that can be used by the program. the code is exported by the libraries and imported by the program to use. as a result, they are dependencies that are needed to run. contrary to popular belief, native programs, not just c# executables, have dependencies.

standard libraries vary depending on the needs of the program. for pretty much all standard programs, they will import some variation of kernel32.dll, kernelbase.dll and ntdll.dll. for this example, despite having no warning marker, it is very suspicious that there is only kernel32.dll. this will be combined with the information about imports in the next section.

imports

imports are the function names of the code in the associated library. these must be exported by the library to be discovered on linking during process initialization so its function pointer can be patched into the import address table for the code to use it.

for normal programs, there are usually many entries depending on the size and functionality of the

program. in this example, despite having no warning marker, only having 2 functions is extremely suspicious. just because there are no imports doesn't mean that the program does not use it. there are dynamic ways to get an import such as with `loadlibrary + getprocaddress`. most stealthy malware up to industry standards will not show `loadlibrary` or `getprocaddress` imports at all so be aware.

strings

strings can show suspicious data like import names, library names, ip addresses, urls, registry keys and other such information. this is an obvious one, there's not much to explain. the other thing to note is just because there are no plaintext strings doesn't mean that it's not malicious. it is trivial to obfuscate strings and every off the shelf stealthy malware or crypter will do this.

import functions and section names are also strings so you will see them here. in this example, there is one particular string that is quite suspicious called "PAYLOAD" but keep in mind that this terminology is used beyond malware and can be legitimate.

other

this sample doesn't have any other metadata. such an empty file could be potentially malicious due to its minimalism. it could mean that it's a crypter or some other delivery malware that is trying to maintain a small footprint. but it's not directly obvious.

conclusion

these indicators are only supposed to help give some guidance to what could be potentially malicious. they are never really obviously malicious standalone. a well rounded analyst should understand each of these sections to fully know what can be present and importantly what isn't present, which can also be an indication. combine several of these indicators to give yourself an initial malicious score and then dive deeper into what looks suspicious to verify your hypotheses.

Bastille - April 21, 2025, 2:03 am

This is definitely something I'm going to read, thanks for the share, I've gone through PE studio few times on TryHackMe but didn't go in depth, and it was not always applied to malware analysis. Having concrete examples, with a specific analysis (malware) is interesting.

You don't want to share this with other members, like in the White Hat section ?

ttbr0 - April 21, 2025, 2:07 am

(Apr 21, 2025 02:03 AM)Bastille Wrote: [\[CITE\]](#) This is definitely something I'm going to read, thanks for the share, I've gone through PE studio few times on TryHackMe but didn't go in depth, and it was not always applied to malware analysis. Having concrete examples, with a specific analysis (malware) is interesting.

You don't want to share this with other members, like in the White Hat section ?

yeah it's honestly a great tool to get an idea of a file before reversing it. i might share this with other members but for now i will post them in here for the time being, at least for mrt-x members to get premium access to the information and review it before it goes public, if ever. let me know your thoughts about it and if there are any improvements or other suggestions.

Bastille - April 25, 2025, 12:37 am

Just finished to read the thread and took some notes on notion. I learned many things, will try to practice on the next weird piece of software shared by a newly created account on HF haha, classic

Besides, I'm currently doing the Level 2 SOC Analyst path on tryhackme, and there is a module

about Malware Anaysis with rooms about PE analysis, it will definitely help to improve my knowledge, such as : <https://tryhackme.com/room/dissectingpeheaders>

Thanks again [ttbr0](#) !

Pr0m3th3us - April 25, 2025, 8:36 am

Never knew of this tool. I'll download it and check it out on some files. Very handy tool thanks for the share.

JoseMar - April 27, 2025, 3:46 am

Amazing tutorial, very well explained step-by-step. [ttbr0](#)

The way you break down the PeStudio analysis and how to interpret the indicators is perfect, even for beginners in malware analysis.

Great contribution, one of the best practical resources I've seen lately!

Bastille - May 10, 2025, 6:48 am

Hey guys, [Pr0m3th3us](#) [JoseMar](#)

Just found out a room is available on tryhackme if you want to dive into PE Headers (free room, no need premium subscription)

<https://tryhackme.com/room/dissectingpeheaders>

JoseMar - May 10, 2025, 8:43 am

(May 10, 2025 06:48 AM)Bastille Wrote: [\[CITE\]](#) Hey guys, [Pr0m3th3us](#) [JoseMar](#)

Just found out a room is available on tryhackme if you want to dive into PE Headers (free room, no need premium subscription)

<https://tryhackme.com/room/dissectingpeheaders>

Oh, that's nice! I'll definitely take a look at this in the coming days. Thank you for sharing; I appreciate it.

ttbr0 - May 10, 2025, 6:13 pm

(May 10, 2025 06:48 AM)Bastille Wrote: [\[CITE\]](#) Hey guys, [Pr0m3th3us](#) [JoseMar](#)

Just found out a room is available on tryhackme if you want to dive into PE Headers (free room, no need premium subscription)

<https://tryhackme.com/room/dissectingpeheaders>

nice link. i hate to write my own posts on these things because its so boring and takes so much effort

Pr0m3th3us - May 10, 2025, 10:42 pm

(May 10, 2025 06:48 AM)Bastille Wrote: [\[CITE\]](#) Hey guys, [Pr0m3th3us](#) [JoseMar](#)

Just found out a room is available on tryhackme if you want to dive into PE Headers (free room, no need premium subscription)

<https://tryhackme.com/room/dissectingpeheaders>

I will check it out, thank you!