Malware Unpacking With Memory Dumps - Intermediate Methods (Pe-Sieve, Process Hacker, Hxd and Pe-bear)

embee-research.ghost.io/unpacking-malware-using-process-hacker-and-memory-inspection/

Matthew November 1, 2023

Intermediate

Demonstrating three additional methods for obtaining unpacked malware samples. Using Process Hacker, Pe-sieve, Hxd and Pe-bear.

In a previous post, we demonstrated a method for unpacking an Asyncrat malware sample by utilising Process Hacker and Dnspy.

We leveraged Process Hacker to identify a suspicious process, then utilised Dnspy to attach to the process and enumerate loaded modules. From there we were able to open a suspicious module from memory, which ultimately obtained the unpacked Asyncrat malware sample.

In this post, we'll go over some additional methods for obtaining the same unpacked payload.

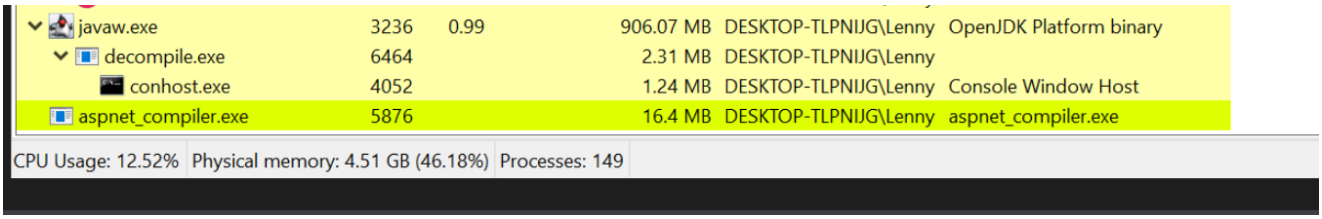
1. Pe-sieve - Directly obtaining the unpacked payload
2. Process Hacker - Monitoring modules and directly dumping memory
3. Process Hacker + X32dbg - Monitoring threads and obtaining the payload using a debugger (x32dbg)

Analysis

We will assume that you have downloaded and unzipped the file from the previous post. You can also obtain the file here.

SHA256: `05c2195aa671d62b3b47ff42630db25f39453375de9cffa92fc4a67fa5b6493b`

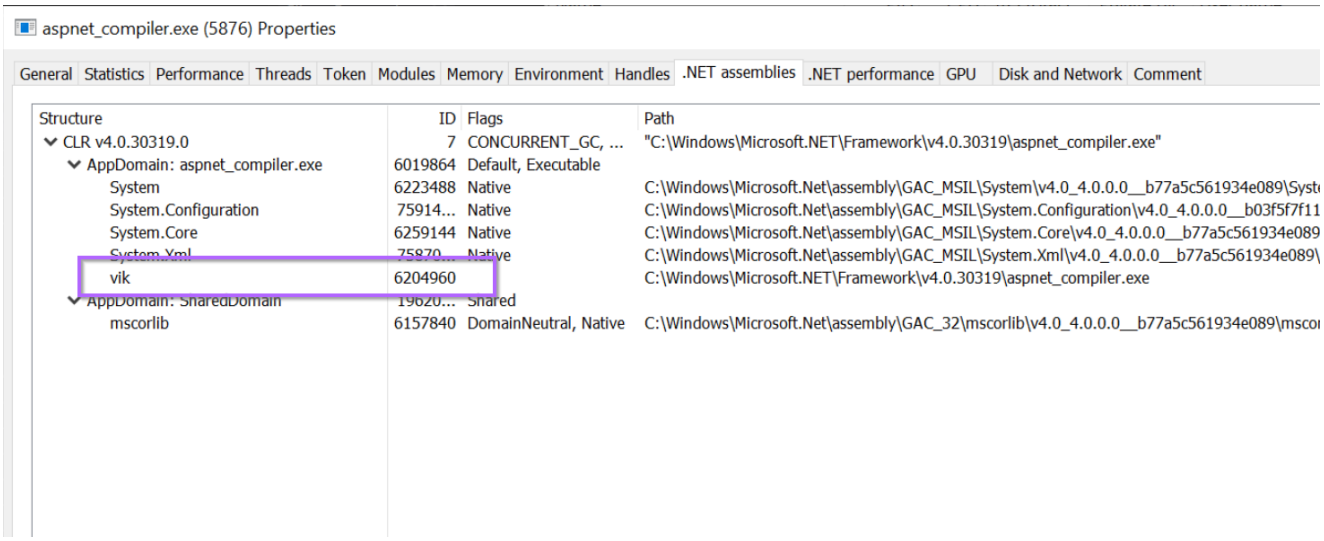
We will also assume that you have executed the file inside of a safe virtual machine, which will result in a running process of `aspnet_compiler.exe`. (This is the file which the malware has injected itself into)



Recap of Initial Post

In the initial post, we monitored for the creation of `aspnet_compiler.exe` using process hacker.

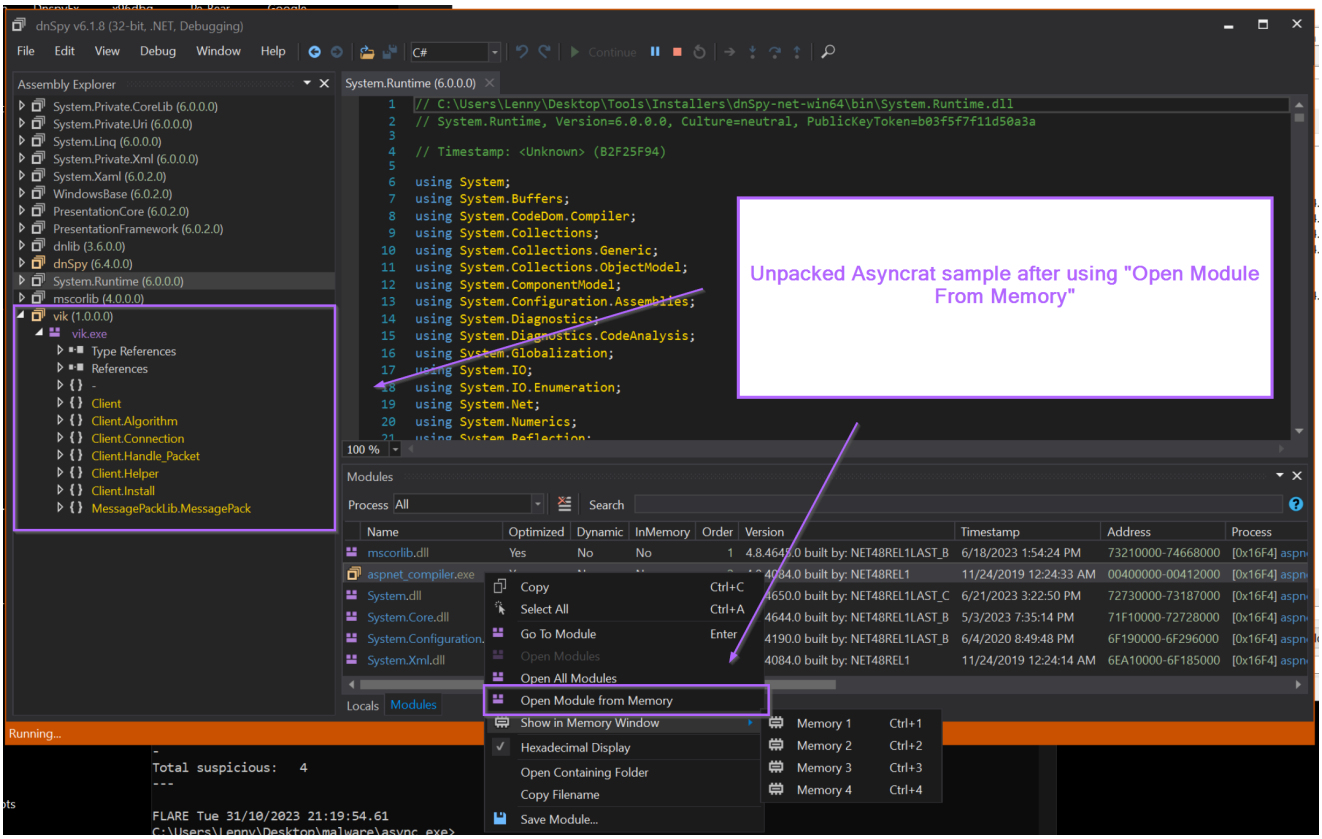
We then used Process Hacker to view loaded .NET assemblies, which resulted in the identification of a suspicious `vik` module, which appeared to have overwritten the original `aspnet_compiler.exe`



We then used Dnspy to attach to the suspicious `aspnet_compiler.exe` process.

This enabled us to view all loaded modules and open the `aspnet_compiler.exe` file from memory.

By opening the file from memory, we were able to obtain the Asyncrat sample that had overwritten the "real copy" of `aspnet_compiler.exe`



With the recap covered, we will now go over some additional methods that could have been used to obtain the unpacked sample.

These methods work equally as effectively on this particular sample, and also work on samples that are not based on .NET (and hence where Dnspy would not be able to work).

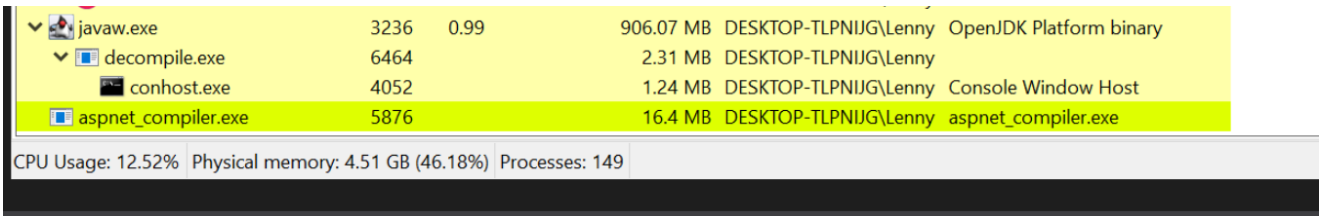
Obtaining the Unpacked Sample Using Pe-sieve

Pe-sieve is one of the quickest and most effective ways to obtain an unpacked sample.

Pe-sieve works by scanning a running process for any suspicious modules that may have been injected or overwritten into memory. If a suspicious module has been identified, pe-sieve will obtain it and save it for you.

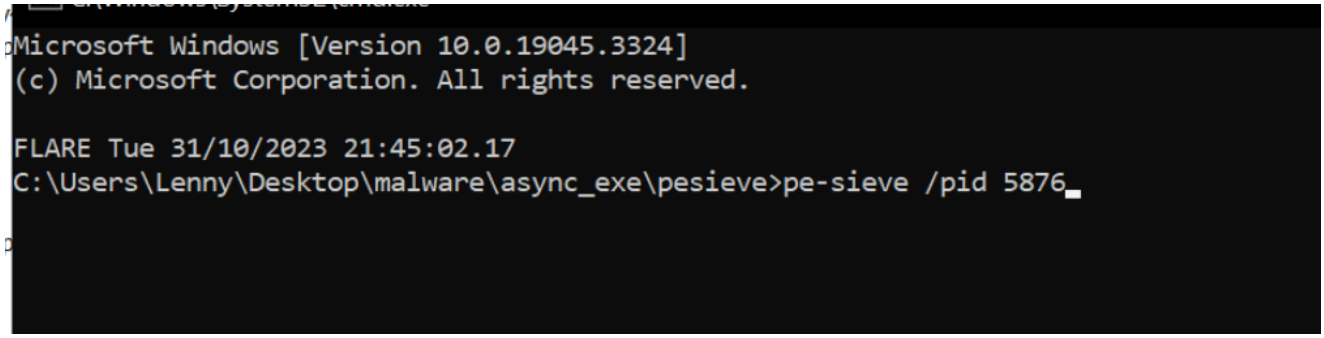
Pe-sieve is an extremely effective and easy-to-use tool.

In the previous screenshot, we identified the suspicious process `aspnet_compiler.exe`, and we can see that it's process id (pid) is `5876`.



To scan the process and obtain the unpacked payload, we can run pe-sieve and pass the pid parameter of `5876` (or whichever the pid is in your situation).

To pass the parameter, we can run the command `pe-sieve /pid 5876`



After running the command, you may see a bunch of text come up on the screen. You can largely ignore the text and skip straight to the end.

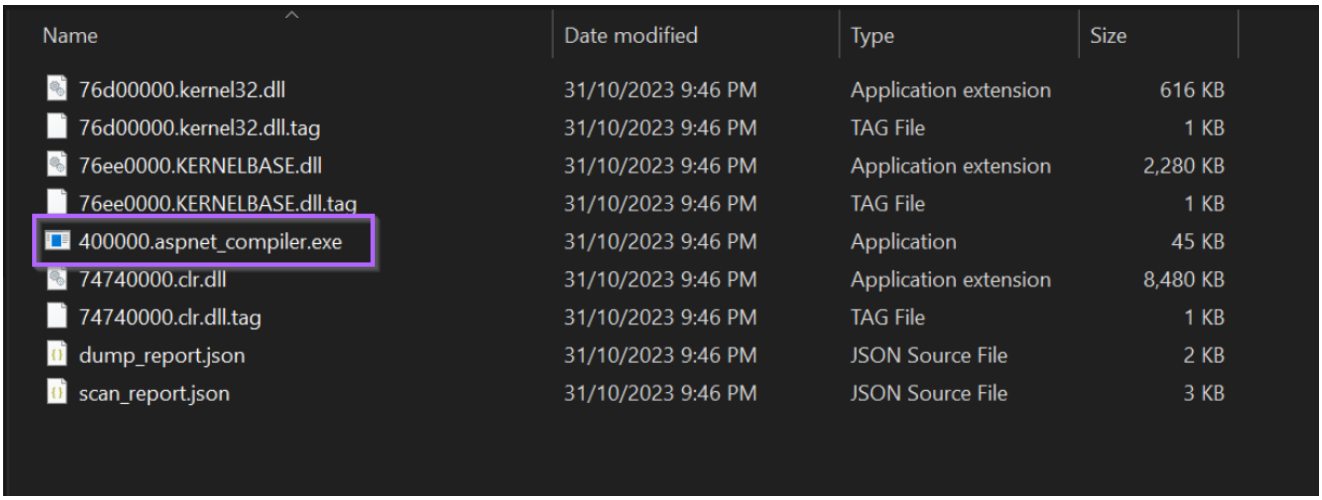
Here we can see the scan summary, indicating that 52 modules were scanned and 1 "implanted PE" was identified.

```
---
PID: 5876
---
SUMMARY:

Total scanned:      52
Skipped:            0
-
Hooked:             3
Replaced:           1
Hdrs Modified:      0
IAT Hooks:          0
Implanted:          1
Implanted PE:       1
Implanted shc:      0
Unreachable files:  0
Other:              1
-
Total suspicious:   6
---

FLARE Tue 31/10/2023 21:46:52.70
C:\Users\Lenny\Desktop\malware\async_exe\pesieve>
```

A new folder `process_5876` will be created from where you ran the command.

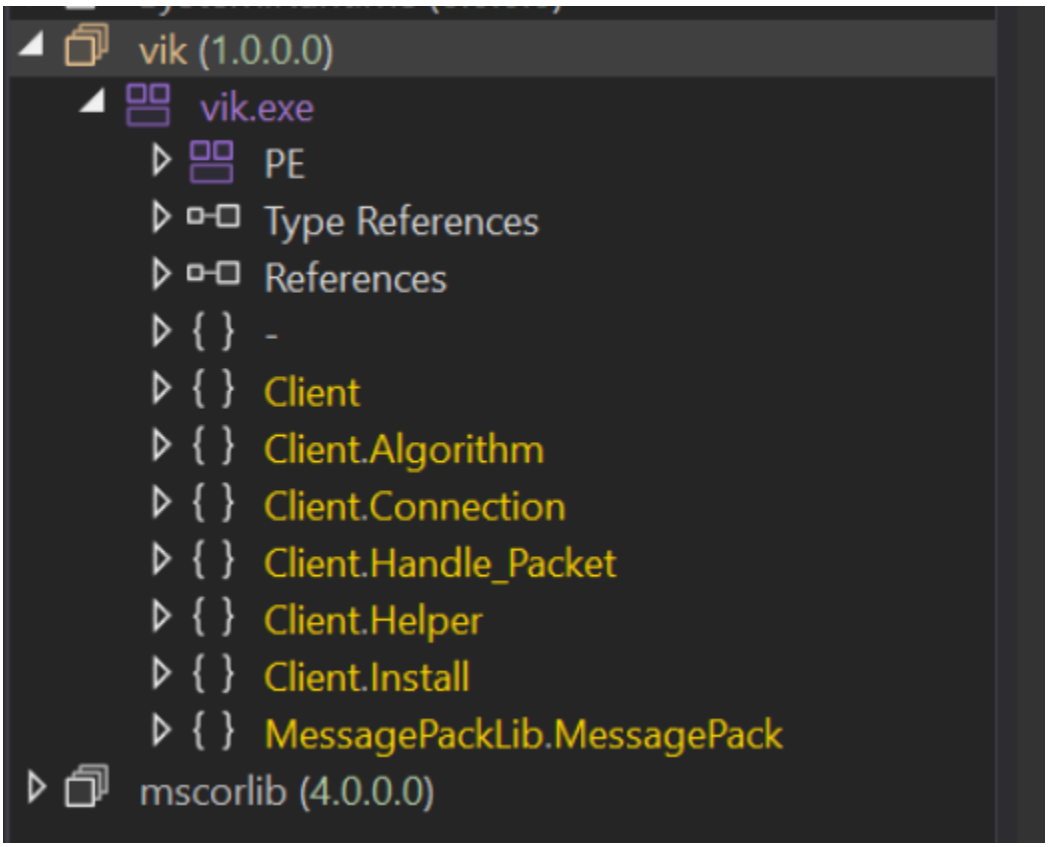
Inside this folder contains a series of files that pe-sieve obtained from memory.

One of these files corresponds to `aspnet_compiler.exe`. Which we previously identified as potentially being overwritten by malware.



By opening the `40000.aspnet_compiler.exe` inside of Dnspy, we can see the unpacked payload.

This is the same `vik` file as identified in the initial post. In this case, we have obtained the same file by using pe-sieve.



Additional Methods for Analysis - Members Section

If you enjoyed this section, you may enjoy the next two sections which are available for paid members of the site.

Becoming a paid member grants you access to all future bonus content. And helps support the creation of more blogs. You will also get access to a discord server where you can ask questions and receive guidance and help.

In the next two sections, you can learn how to

- Perform a memory dump with Process Hacker
- Identify a broken memory dump using a hex editor
- Identify and Correct a broken memory dump using pe-bear
- Identify a suspicious thread with Process Hacker
- Map a thread to a memory region and obtain it using X32dbg.

This post is for paying subscribers only