

MAL: Malware Introductory



MAL: Malware Introductory

The start of a series of rooms covering Malware Analysis...

[Task 1] What is the Purpose of Malware Analysis?

Malware is such a prevalent topic within Cybersecurity, and often an unfortunately recurring theme among global news today.

Not only is malware analysis a form of incidence response, but it is also useful in understanding how the behaviours of variants of malware result in their respective categorisation. This room will be a practical introduction to the techniques and tools used throughout malware analysis - albeit brief, I hope to expand on these techniques a lot more in-depth within the future.

When analysing malware, it is important to consider the following:

- Point of Entry (PoE) i.e. Was it through spam that our e-mail filtering missed and the user opened the attachment?

Let's review our spam filters and train our users better for future prevention!

- What are the indicators that malware has even been executed on a machine?
Are there any files, processes, or perhaps any attempt of "un-ordinary" communication?
- How does the malware perform? Does it attempt to infect other devices? Does it encrypt files or install anything like a backdoor / Remote Access Tool (RAT)?
- Most importantly - can we ultimately prevent and/or detect further infection?!

You are here amongst the Malware series:

0. MP: Malware Analysis Primer

#1

Ah, now I kinda understand...

No answer needed

[Task 2] Understanding Malware Campaigns

Despite the many variants of malware, attacks can generally be classified into two types: **Targeted** and **Mass Campaign**.

Targeted

A "Targeted" attack is just that - targeted. In most cases, malware attacks that occur this way are created for a specific purpose against a specific target. A great example of this type of purpose could be the DarkHotel malware, whom is designed to steal information such as authentication details from government officials.

Mass Campaign

On the other hand, the "Mass Campaign" classification can be akin to many real life examples, and is the most common type of attacks. The entire purpose of this type of Malware is to infect as many devices as possible and perform whatever it may - regardless of target.

Companies such as Kaspersky to name one, track these campaigns (known as Advanced Persistent Threats (APTs) and often report on their infection rate and indicators, much akin to the real-life spread of a virus from the World Health Organisation (WHO).

Kaspersky report on the "Crouching Yeti (Energetic Bear)" campaign, this campaign specifically targets the following:

- Industrial/machinery
- Manufacturing
- Pharmaceutical
- Construction
- Education
- Information technology

(Kaspersky)

Whilst it this variant is technically targeted, there is a rather large scope of this variant of malware, and as such, can be considered as a "Mass Campaign" attack.

#1

What is the famous example of a targeted attack-esque Malware that targeted Iran?

Stuxnet

#2

What is the name of the Ransomware that used the Eternalblue exploit in a "Mass Campaign" attack?

wannacry

[Task 3] Identifying if a Malware Attack has Happened

Much like any interaction with an Operating System, there is always remnants of such activity even if there is little trace...

Unfortunately, **malware** (dependant on its variant) **is largely obtrusive** - in the sense that it leaves quite an extensive papertrail of evidence...Perfect for us analysts!

The ultimate process of a malware attack can be broken down into a few broad steps:

1. Delivery
2. Execution
3. Maintaining persistence (not always the case!)
4. Propagation (not always!)

These steps will generate lots of data. Namely: network traffic such as communicating with hosts, file system interaction like read/writes and modification.

Malware is essentially classified based upon the behaviours it produces to perform the steps listed above.

For a famous example, Wannacry performs all four of these steps.

1. Delivery

This could be of many methods, to name a few: USB (Stuxnet!), PDF attachments through "Phishing" campaigns or vulnerability enumeration.

2. Execution

Here's the main part of how we classify Malware. What does it actually do? If it encrypts files - it's Ransomware! If it records information like keystrokes or displays adware - we can classify it as Spyware.

We only understand this stage through analysing the sample, which is why analysis is important - and is what we'll be covering.

3. Maintaining Persistence

It wouldn't make much sense for Malware authors to go through all the trouble of developing a piece of code that is capable of execution - just for it to execute and that's it...Gone. Unless you have a very specific agenda (Cerber). I have submitted an extensive report on this specific Malware, amongst others for a University Module on Malware Analysis...however I will wait for it to be marked before sharing.

Imagine a keylogger has been installed, but you then restart your Computer 30 seconds later. Unless you've entered a lot of sensitive information in 30 seconds - that'll be of no use to the author.

4. Persistence

This stage is largely why Malware is so "noisy", Malware employs many techniques, of which we'll be covering in-depth much later on. Essentially, this stage is just to make sure that the "execution" is worth its while.

5. Propagation

Hey...If you can infect one device, why not infect more whilst you're at it? Again, this is another reason why Malware can be so noticeable. Host discovery generates a lot of network traffic, we'll come to this later.

In Summary, there are two categories of fingerprints that malware may leave behind on a Host after an attack:

Host-Based Signatures

These are generally speaking the results of execution and any persistence performed by the Malware. For example, has a file been encrypted? Has any additional software been installed? These can be two of many, many host-based signatures that are useful to know to prevent and check against further infection.

Network-Based Signatures

At an overview, this classification of signatures are the observation of any networking communication taking place during delivery, execution and propagation. For example, in Ransomware, where has the Malware contacted for Bitcoin payments?

Such as in the case of Wannacry, looking for a large amount of "Samba" Protocol communication attempts is a great indication of infection due to its use of "Eternalblue".

#1

Name the first essential step of a Malware Attack?

delivery

#2

Now name the second essential step of a Malware Attack?

execution

#3

What type of signature is used to classify remnants of infection on a host?

host-based signature

#4

What is the name of the other classification of signature used after a Malware attack?

network-based signature

[Task 4] Static Vs. Dynamic Analysis

There are two categories used when analysing malware, these are:

1. Static Analysis
2. Dynamic Analysis

Whilst the methods and tools used for these two categories are vastly different, they are essential in composing an understanding of how a malware behaves.

Static Analysis.

At its brief, "Static Analysis" is used to gain a high-level abstraction of the sample - it can be fairly simple to decide if a piece of code is "malicious" or not with this method alone (but not always, this will be discussed later...). At its core, this method is of the analysis of the sample at the state it presents itself as, without executing the code. Employing the use of techniques such as signature analysis via checksums means quick, efficient (albeit extremely brief) and safe analysis of malware.

Dynamic Analysis

This step is a lot more involved, and is where the abstraction of the sample is largely built upon. "Dynamic Analysis" essentially involves executing the sample and observing what happens. This of course is not safe. If the sample turns out to be "Ransomware" - you've now lost your files. If it is capable of propagating via traversing a network, nice...You've now just infected your Local Area Network (LAN).

Please note that these are extremely simplistic explanations of these techniques, there is a lot more involved which we will go throughout this series.

#1

I understand the two broad categories employed when analysing potential malware!

No answer needed

[Task 5] Discussion of Provided Tools & Their Uses

You will see that some tools will overlap between Static and Dynamic analysis:

Provided Static Analysis Tools:

C:\Users\Analysis\Desktop\Tools\Static\PE Tools

- Dependency Walker (depends)
- PeID
- PE Explorer
- PEview
- ResourceHacker

C:\Users\Analysis\Desktop\Tools\Static\Disassembly

- IDA Freeware
- WinDbg

C:\Users\Analysis\Desktop\Tools\Sysinternalsuite

- ResourceHacker

C:\Users\Analysis\Desktop\Tools\Dynamic

The tools listed here will be used for future tasks, as they involve debugging which is currently out-of-scope for this room...However, will be explored later within the series.

#1

Lets proceed

No answer needed

[Task 6] Connecting to the Windows Analysis Environment (Deploy)

Whilst malware has been known to traverse (spread) over RDP, in this instance, any and all samples on here are not capable of doing so - nor are they capable of performing any destructive action.

This series will teach you the practical knowledge and tool familiarity to allow you to transfer these skills to **actual** samples if you wish too, outside of TryHackMe

With this being a Windows instance specifically, alongside the additional tools and tasks it has to execute, please expect up to wait up towards 10 minutes before being able to access your instance. The average deploy to login took about 7 minutes. Also, please note that the Host will not respond to pings - only the (Remote Desktop Protocol) RDP protocol (3389)

Credentials:

You can either connect via RDP (connect to our network first via OpenVPN), or control the machine in browser (no connection required). Please note that **this Windows "Instance" will take atleast 5 minutes to fully boot - please be patient**. You can view the progress via the progress-bar within the display above.

10.10.40.60

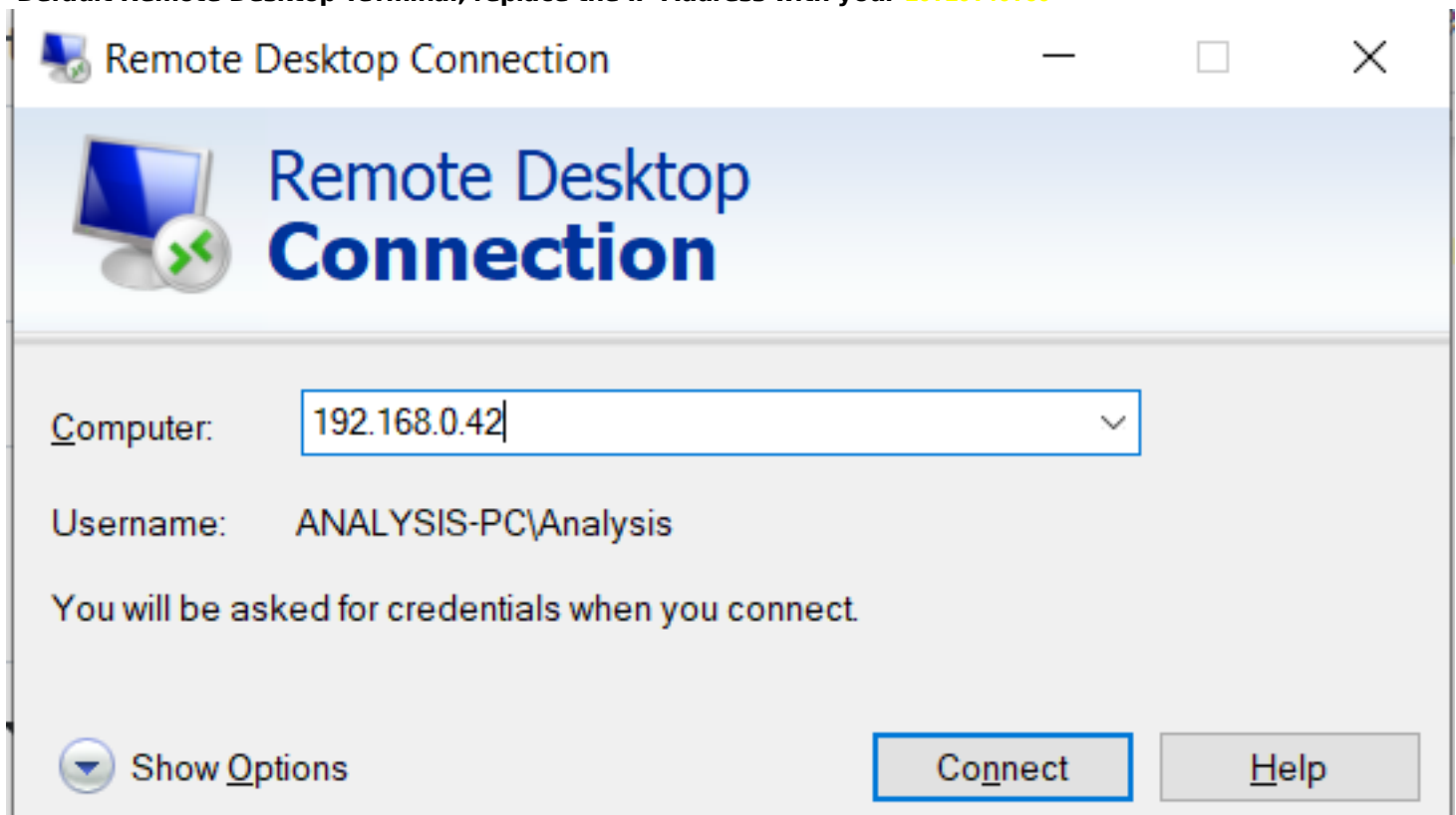
Domain: ANALYSIS-PC

Username: Analysis

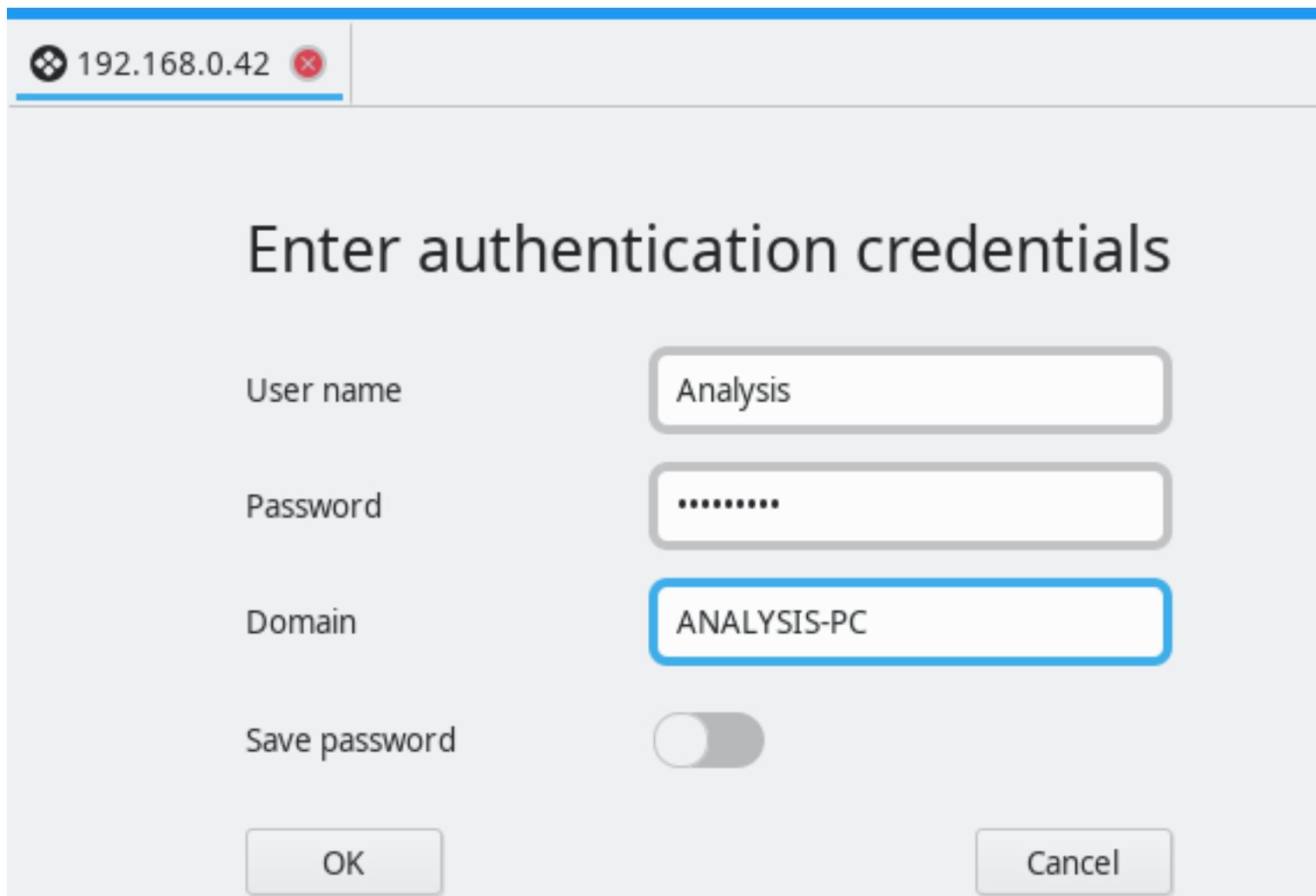
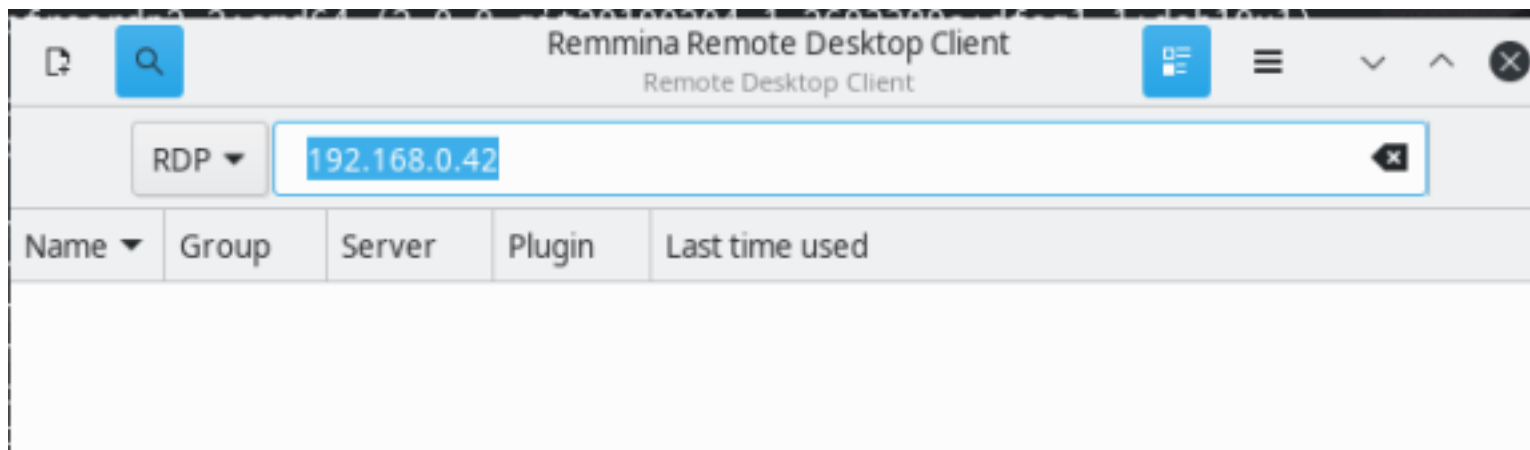
Password: tryhackme

Windows:

- Default Remote Desktop Terminal, replace the IP Address with your 10.10.40.60



Linux: Any compatible RDP client such as Remmina: `sudo apt-get install remmina`
Again, replacing the IP address with your 10.10.40.60



#1

I've logged in!

No answer needed

[Task 7] Obtaining MD5 Checksums of Provided Files

MD5 "Checksums" are a prominent attribute in the malware Community. Because there can be many variants of a family of Ransomware, these MD5 "Checksums" are cryptographic "fingerprints" of the files. This allows a

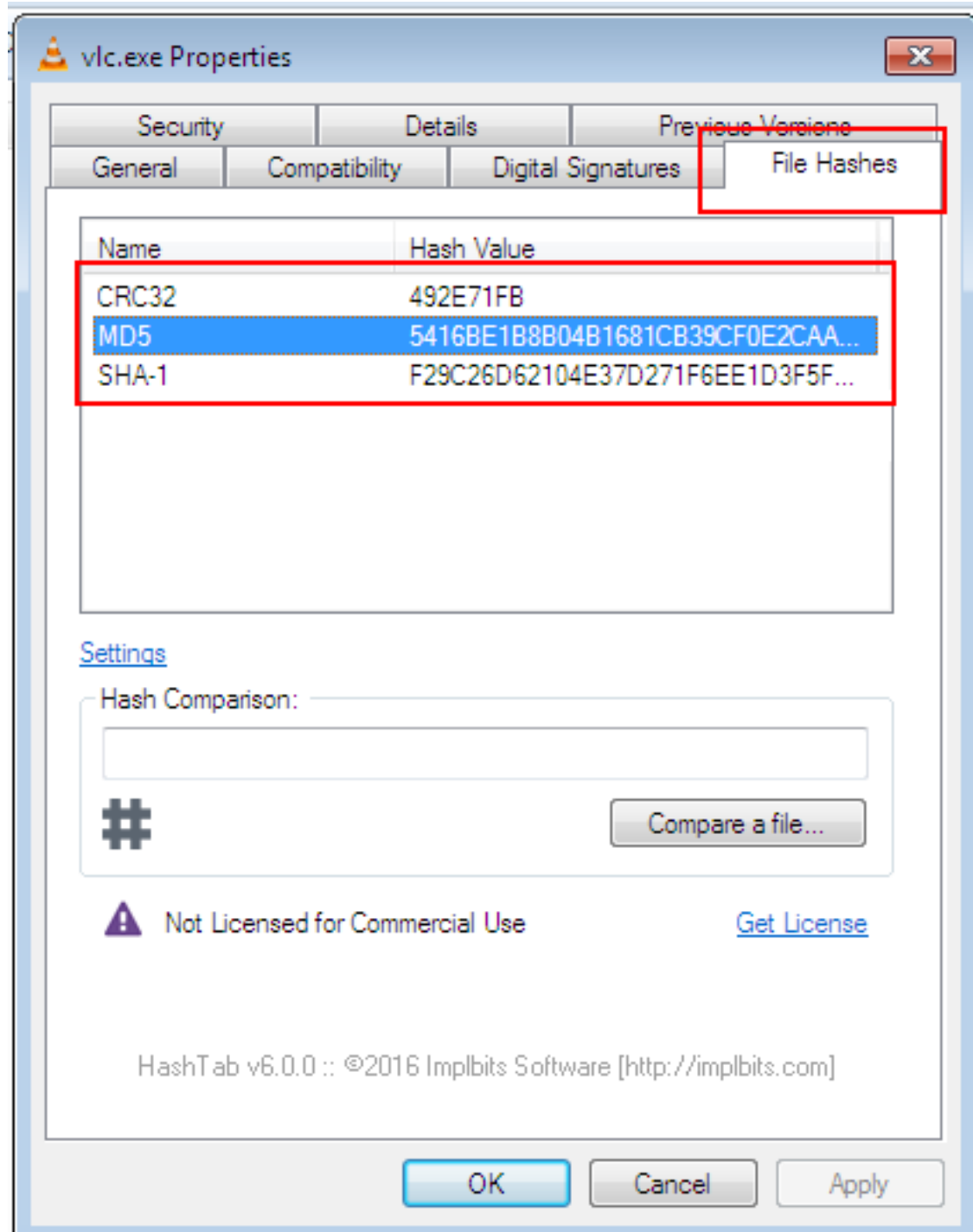
uniformed identification throughout the community - especially with automated Sandboxes.
For example, say you have 20 files of unknown origin, you are able to identify their genus using their MD5 sum against websites such as Virustotal, if that MD5 "Checksum" has been previously analyzed - removing all the legwork for us!

Navigate to the "Tasks" Folder on the Desktop, and then enter the "Task 7" Directory, where there will be three files:

- aws.exe
- NetLog.exe
- vlc.exe

Sure, these are common names of executables, but anyone can name an executable as whatever they like! Just because it says "vlc" doesn't mean it is indeed the VLC application! This is where identifying their MD5 Checksum is useful, as no matter the name - their MD5 reveals its true identity.

I have installed the "HashTab" application, which calculates a files MD5 sum - amongst others, directly within Windows Explorer as if you were inspecting its properties...



YOUR TASK:

Identify the MD5 Checksums of the three files provided in "Task 7" (You can use Ctrl + C & Ctrl + V over RDP)

#1

The MD5 Checksum of aws.exe

D2778164EF643BA8F44CC202EC7EF157

#2

The MD5 Checksum of Netlogo.exe

59CB421172A89E1E16C11A428326952C

#3

The MD5 Checksum of vlc.exe

5416BE1B8B04B1681CB39CF0E2CAAD9F

[Task 8] Now lets see if the MD5 Checksums have been analysed before

Outside of the Remote Windows Environment i.e. Kali or your Windows PC, look up those MD5 "Checksums" on Virustotal to solve this task:

#1

Does Virustotal report this MD5 Checksum / file aws.exe as malicious? (Yay/-Nay)

nay

#2

Does Virustotal report this MD5 Checksum / file Netlogo.exe as malicious? (Yay/-Nay)

nay

#3

Does Virustotal report this MD5 Checksum / file vlc.exe as malicious? (Yay/-Nay)

nay

[Task 9] Identifying if the Executables are obfuscated / packed

There are a few provided tools on this Windows instance that are capable of identifying the compiler / packer of a file. However, PeID has a huge database and is a great tool for this.

Moreover, just because a file doesn't have the ".exe" extension, doesn't mean it isn't an actual executable! For instance, it can have the ".jpg" extension and still be an executable piece of code. This is a tad-bit out of scope for this room specifically, but essentially, files have identifying attributes within its hex - known as file headers.

E.g. The hex value for an executable is always "4D 5A". So if a file with a ".jpg" file has the hex header of "4D 5A",

then it is obviously not a jpg file. You can read more into file headers / trailers here, which are great resources for data carving in file forensics / recovery.

Provided Tools: PeID

Now using "PeID", identify the compiler / packer of the following two files in the Directory "Tasks/Task 9" to answer the questions.

#1

What does PeID propose 1DE9176AD682FF.dll being packed with?

Microsoft Visual C++ 6.0 DLL

#2

What does PeID propose AD29AA1B.bin being packed with?

Microsoft Visual C++ 6.0

[Task 10] What is Obfuscation / Packing?

Theory:

Packing is one form of obfuscation that malware Authors employ to prevent the analysis of programmes. There are both legitimate and malicious reasons as to why the Author of a program will want to prevent the decompiling of their program.

For example, a legitimate reason is the protection of intellectual property! Whilst I'm one for open-source as much as the next person here - alas not every organisation has the same mindset...but let's leave that aside.

In the same token, just because you write a program...Why should everyone have the right to "copy" your project? This is one of the justifiable reasons for obfuscation - it is yours at the end of the day!

However, malware Authors employ obfuscation techniques such as packing - whilst for the same reasons, they do so with the intent to prevent people like us reversing it to understand its behaviours and ultimately with the aims of achieving infection.

How packing works is out of scope for this room, but I hope to be able to delve into topics like these later on within THM, so that you can understand the theory behind the practical skills you'll be using.

Practical:

Your task is to identify whether or not the file "6F431F46547DB2628" located in the Directory of "Tasks\Task 10" is packed using the tool "PeID" akin to the task you just completed!

#1

What packer does PeID report file "6F431F46547DB2628" to be packed with?

FSG 1.0 -> dulek/xt

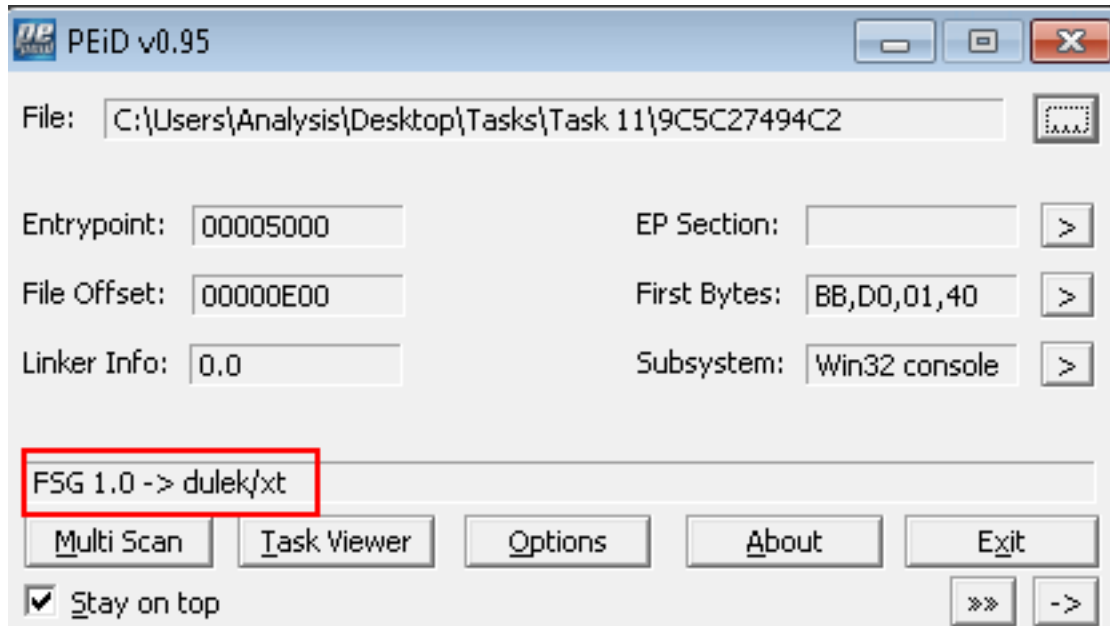
[Task 11] Visualising the Differences Between Packed & Non-Packed Code

This is another example of why PeID is a fantastic little tool. With its huge dataset of known packers, it is arguably one of the better ones at being able to fingerprint the names of packers / obfuscators within an application.

Whilst this tool has a huge database, it doesn't have every packer out there! Especially say if an Author has written their own - PeID will have no way of identifying the packer used. In cases like this, **it's more about what PeID doesn't tell us** - rather than **what it does**.

Practical:

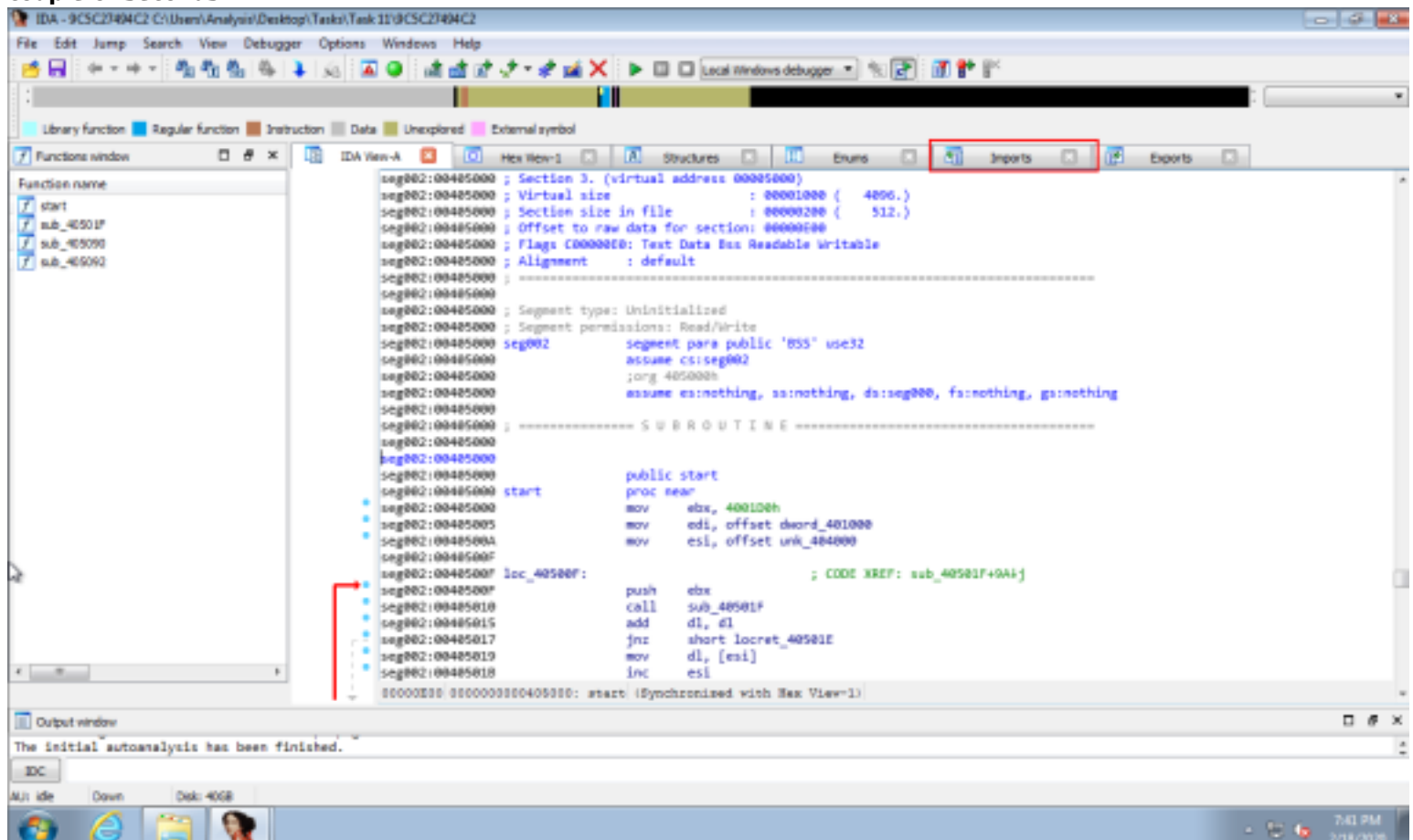
You can try this yourself by navigating to **directory "Tasks/Task 11"** and dragging and dropping that file into PeID. What does it tell us?

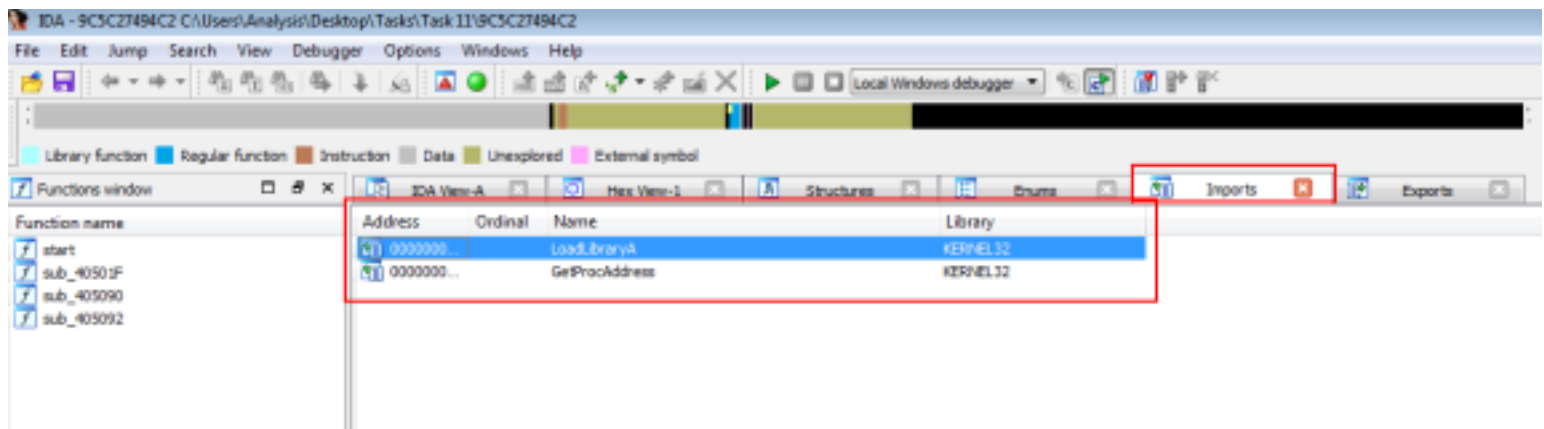


In this instance, PeID is able to detect what packer has been used to obfuscate the code. Whilst PeID is capable of detecting the possibility of packers being used, it is not able to automatically de-obfuscate them. This is a process we will have to do manually - at a later stage.

After confirming that this file is indeed packed, let's open it up with a tool called **IDA Freeware**. This is located in the **"Tools/Static/Disassembly"** directory (or you can search for it through Windows Toolbar). Notice how there is a very minimal amount of information provided to us? For example, the "Imports" tab is practically empty! Little odd...right?

When opening the file, a few dialogue boxes may appear - its just IDA Freeware processing the file, it'll take a couple of seconds.





Address	Ordinal	Name	Library
00000000...		LoadLibraryA	KERNEL32
00000000...		GetProcAddress	KERNEL32

We can see there's only two "**Imports**" in this program! That's a bit bizarre...
 ...The use of "IDA Free" will be developed upon as it is an advanced bit of kit. But essentially, we can see the flow of how the program executes - indicated by the arrows. The problem? There's very little here! There's a few more characteristics that indicate its packed, but this is also beyond the scope for this room.

```

; Section 3. (virtual address 00005000)
; Virtual size           : 00001000 ( 4096.)
; Section size in file   : 00000200 ( 512.)
; Offset to raw data for section: 00000E00
; Flags C00000E0: Text Data Bss Readable Writable
; Alignment              : default

; Segment type: Uninitialized
; Segment permissions: Read/Write
seg002 segment para public 'BSS' use32
assume cs:seg002
;org 405000h
assume es:nothing, ss:nothing, ds:seg000, fs:nothing, gs:nothing

public start
start proc near
mov     ebx, 4001D0h
mov     edi, offset dword_401000
mov     esi, offset unk_404000

```

```

loc_40500F:
push    ebx
call    sub_40501F
add     dl, dl
jnz     short locret_40501E

```

```

mov     dl, [esi]
inc     esi
adc     dl, dl

```

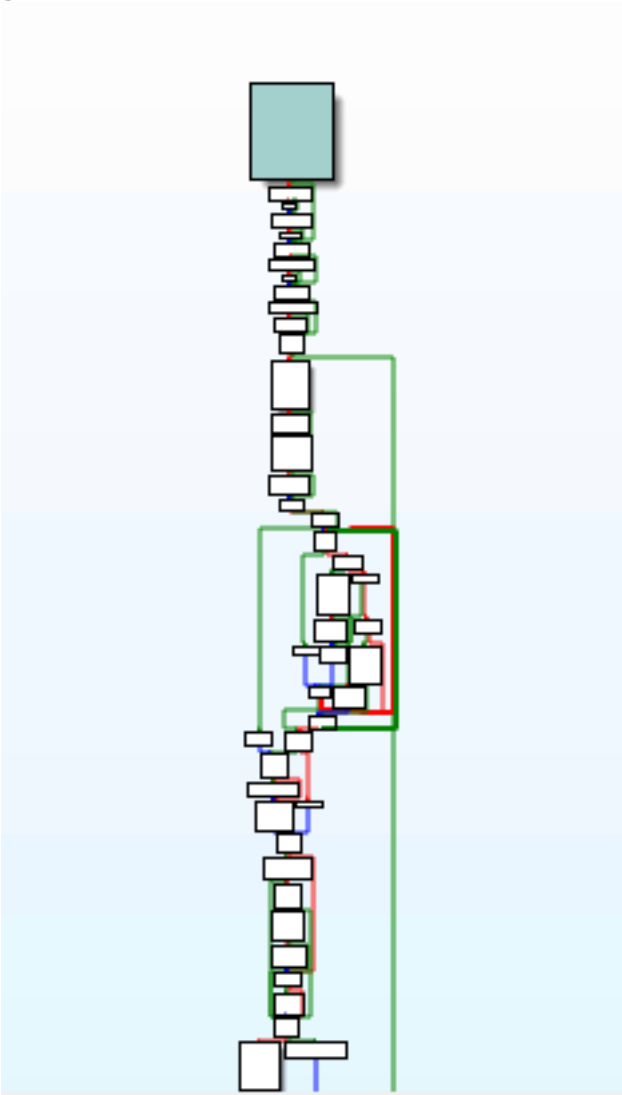
```

locret_40501E:
retn
start endp

```

Whereas, if we was to open a file that isn't obfuscated, we should expect to see a much larger import count and

graph/flowchart, like this:



IDA View-A			
Hex View-1			
Structures			
Enums			
Imports			
Address	Ordinal	Name	Library
00000000...		libvlc_add_intf	libvlc
00000000...		libvlc_new	libvlc
00000000...		libvlc_playlist_play	libvlc
00000000...		libvlc_release	libvlc
00000000...		libvlc_set_app_id	libvlc
00000000...		libvlc_set_user_agent	libvlc
00000000...		libvlc_wait	libvlc
00000000...		CryptAcquireContextA	ADVAPI32
00000000...		CryptGenRandom	ADVAPI32
00000000...		CryptReleaseContext	ADVAPI32
00000000...		RegOpenKeyExW	ADVAPI32
00000000...		RegQueryValueExW	ADVAPI32
00000000...		CloseHandle	KERNEL32
00000000...		CreateDirectoryW	KERNEL32
00000000...		CreateFileW	KERNEL32
00000000...		CreateSemaphoreW	KERNEL32
00000000...		CreateThread	KERNEL32
00000000...		DeleteCriticalSection	KERNEL32
00000000...		DeleteFileW	KERNEL32
00000000...		DuplicateHandle	KERNEL32
00000000...		EnterCriticalSection	KERNEL32
00000000...		FindClose	KERNEL32
00000000...		FindFirstFileW	KERNEL32
00000000...		FindNextFileW	KERNEL32
00000000...		FreeLibrary	KERNEL32
00000000...		GetCommandLineW	KERNEL32

See how there's so much more information here? Obfuscated code is much harder to analyze at least at the static level, as we're presented with very little information!

#1

Cursed obfuscation!

No answer needed

[Task 12] Introduction to Strings

Theory:

"Strings" are essentially the ASCII / Text contents of a program...this could be anything from passwords for self-extracting zips, to bitcoin addresses in ransomware samples.

Such as that in the example above, when analysing the contents of these strings, we can sometimes paint a fairly indicative picture of the behaviours of the programme - bitcoin wallets being used in ransomware.

Task:

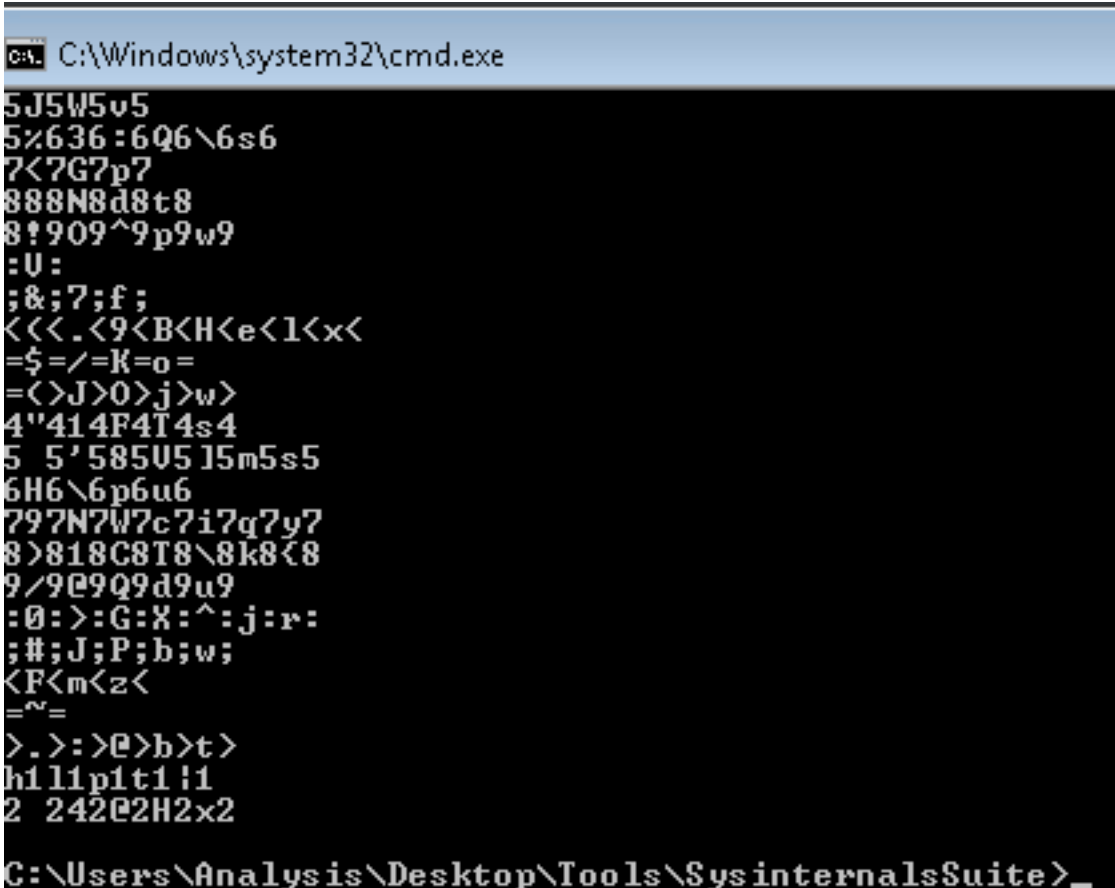
Open a Command prompt on the Windows Machine and navigate to the directory "**Tools\SysinternalsSuite**"
cd C:\Users\Analysis\Desktop\Tools\SysinternalsSuite

Keep this terminal open.

We're going to use Microsoft's Sysinternals "Strings" program to output the retained strings within the specified file in "Task 12". We can do this by:

strings "C:\Users\Analysis\Desktop\Tasks\Task 12\67844C01"

You will receive a whole load of text, most of it looks like nonsense...But there is some text in there that is valuable. Scroll up!



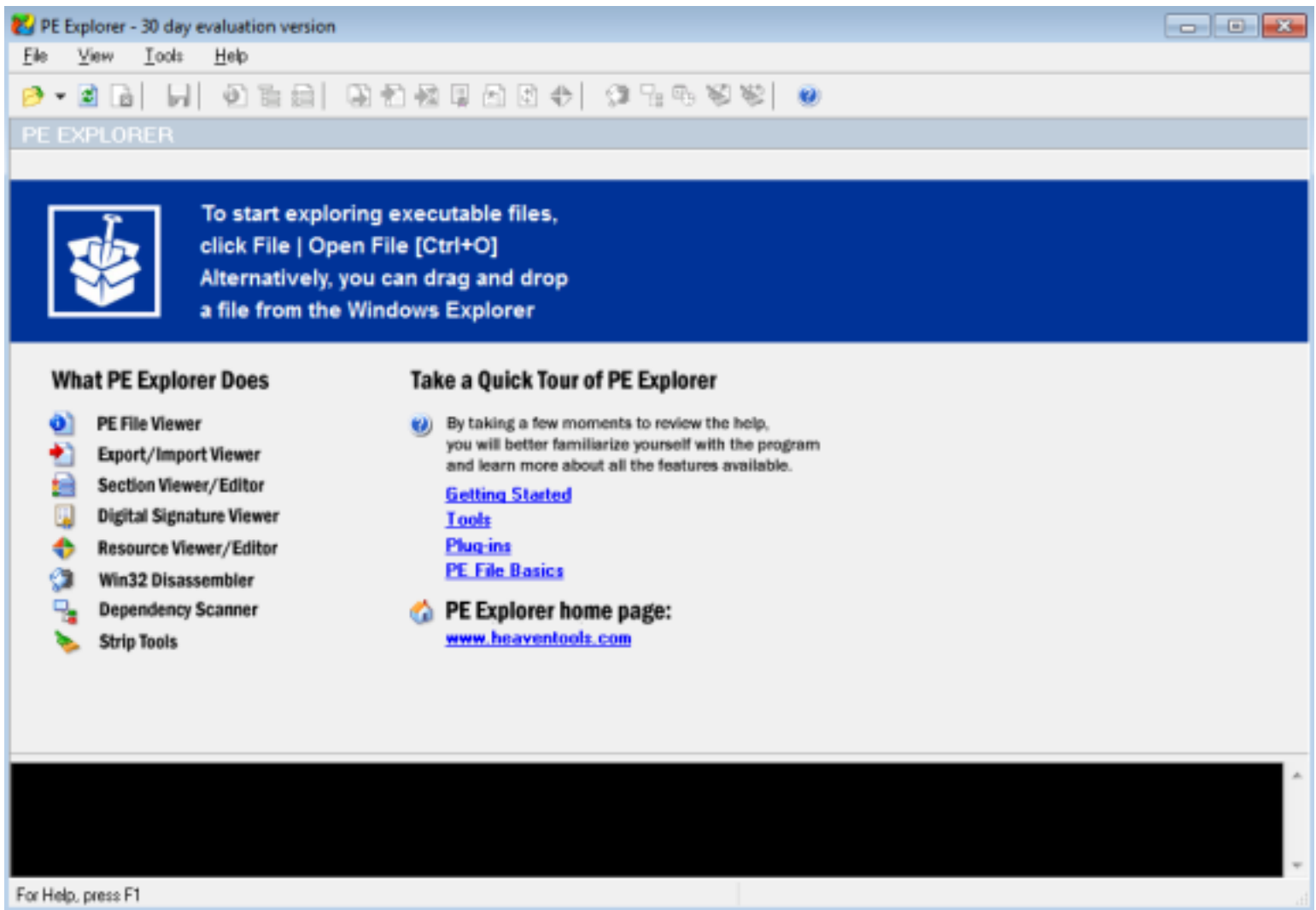
```
C:\Windows\system32\cmd.exe
5J5W5v5
5%636:6Q6\6s6
7<7G7p7
888N8d8t8
8!909^9p9w9
:U:
;&;7;f;
<<<.<9<B<H<e<l<x<
= $ = / = K = o =
= < > J > O > j > w >
4"414F4I4s4
5 5'585U5l5m5s5
6H6\6p6u6
797N7W7c7i7q7y7
8>818C8T8\8k8<8
9/9@9Q9d9u9
:0:>:G:X:^:j:r:
;#;J;P;b;w;
<F<m<z<
=~=
>.>:>@>b>t>
h1l1p1t1!1
2 242@2H2x2
C:\Users\Analysis\Desktop\Tools\SysinternalsSuite>
```

Proceed to answering Question 1.

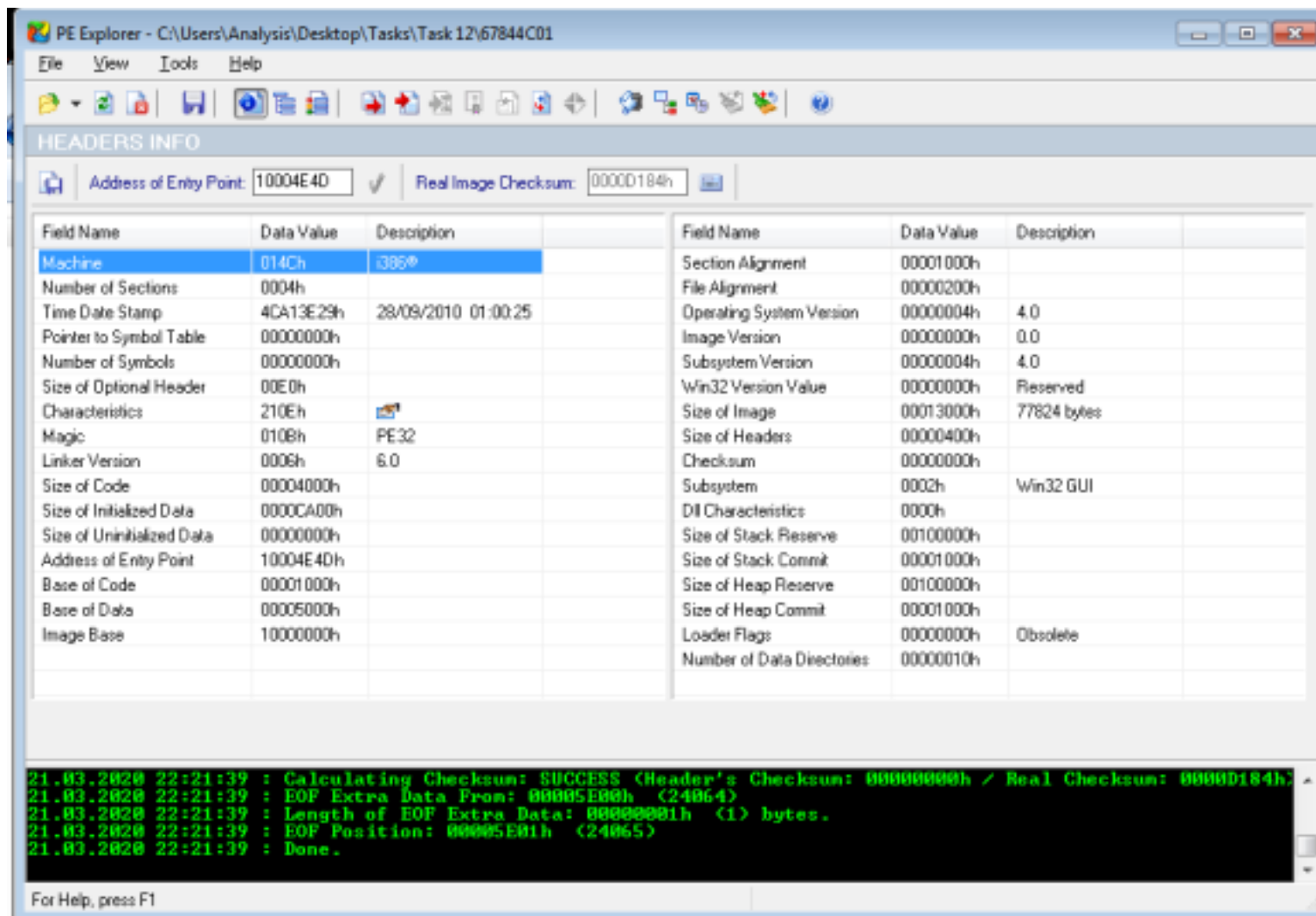
You'll find that programs often contain large amount of strings and using the "strings" tool from sysinternals may only display 10% of these...

...That and it's not exactly practical scrolling up through a terminal for stuff like this - we are on Windows afterall! There's a GUI tool for that.

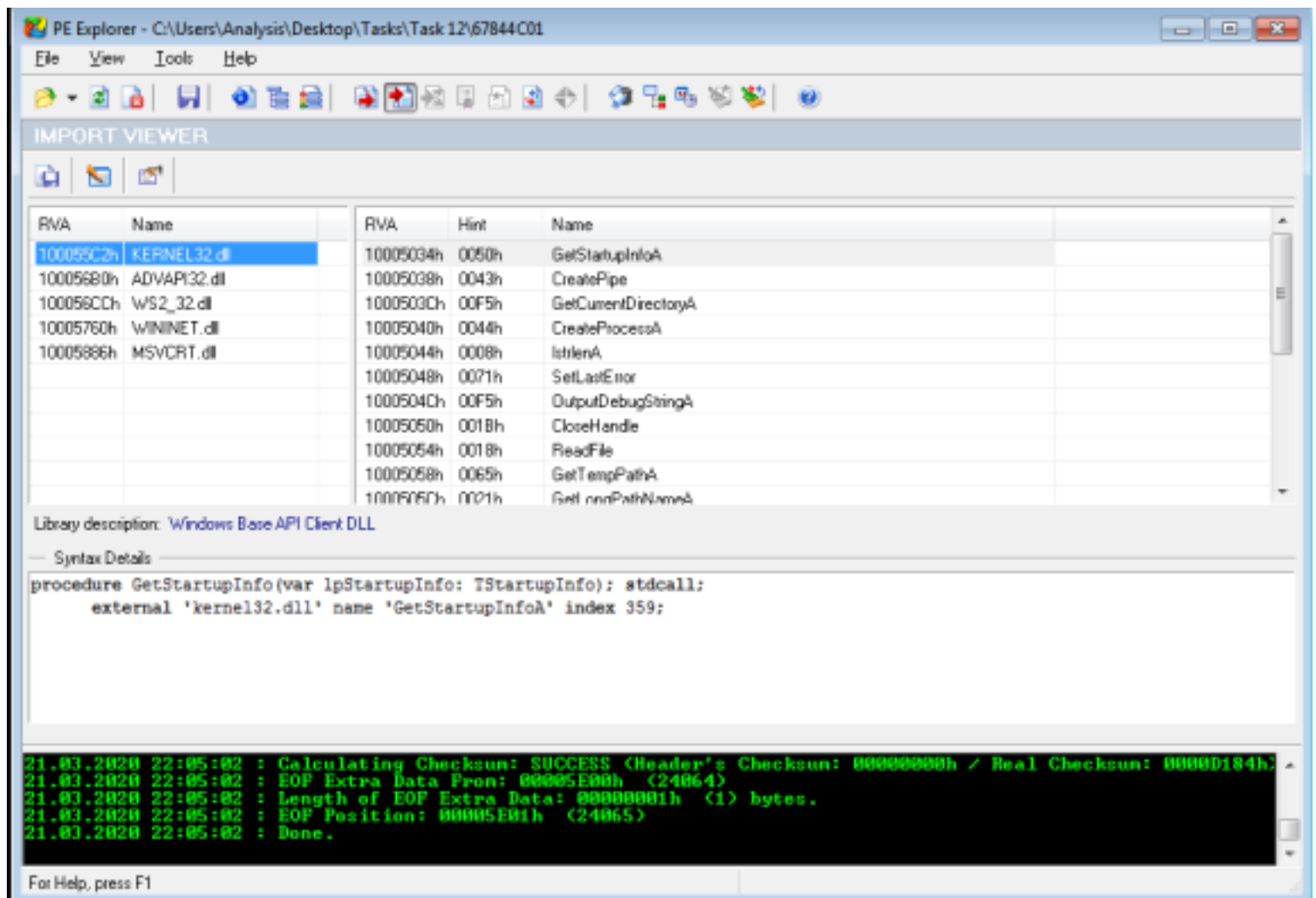
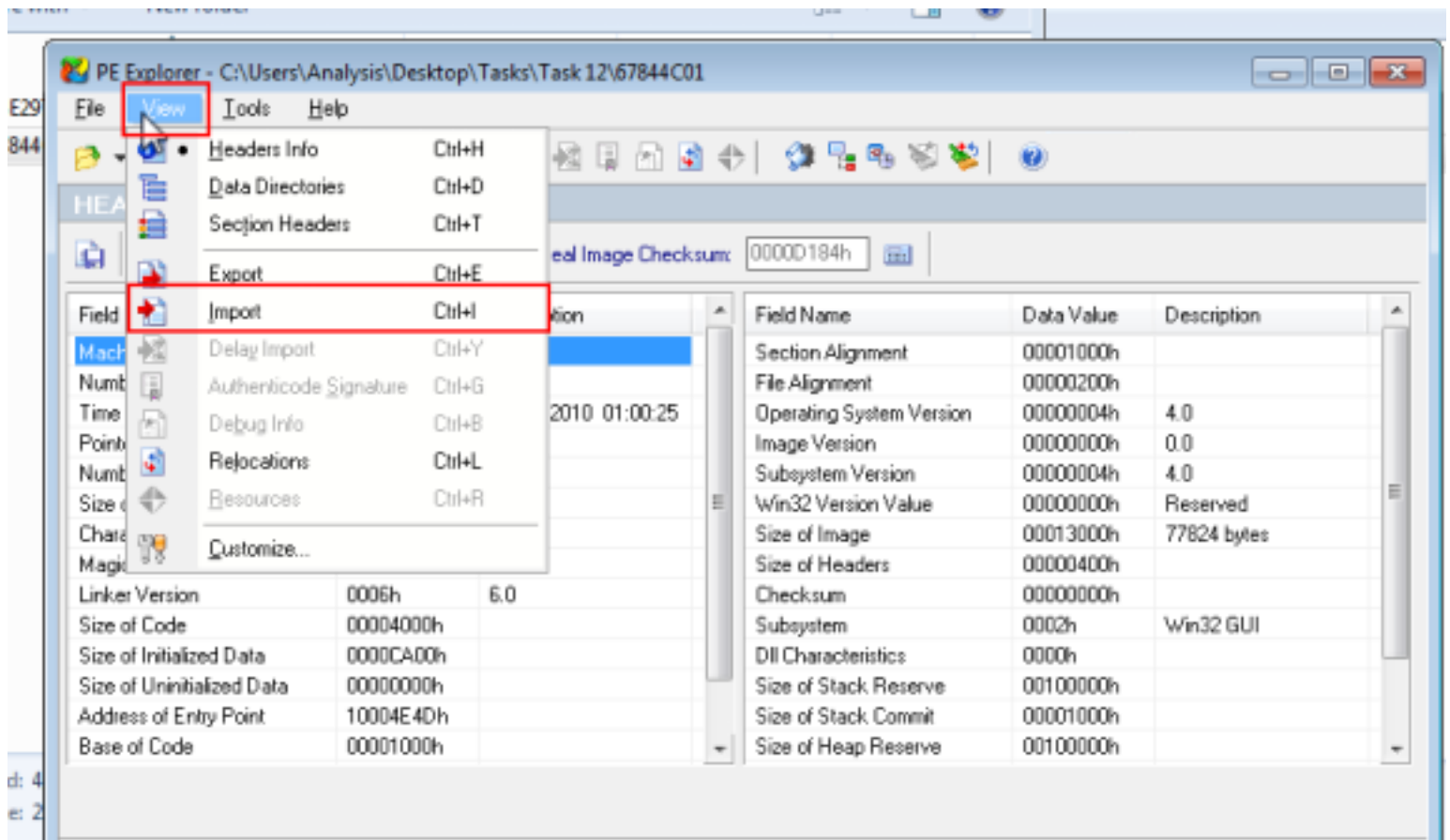
Launch the application within "**Tools/Static/PE Tools/PE Explorer**" and **drag and drop** the same file "**67844C01**" from the previous question into the application.



Where you will be presented with the following, indicating that it has successfully imported:



After import. Navigate to "View -> Imports"



You can now answer Question #2!

#1

What is the URL that is outputted after using "strings"

practicalmalwareanalysis.com

#2

How many unique "Imports" are there?

5

[Task 13] Introduction to Imports

Theory:

The classification of IDA Freeware is arguable as the tool can be used for both static and dynamic analysis. Without going too in-depth regarding the differences, there are two classifications of tools like IDA Freeware:

- Disassemblers
- Debuggers

I'll allow you to explore the differences between these two types of tools and their use cases in your own time, but for contextual sake - Disassemblers reverse the compiled code of a program from machine code to human-readable instructions (assembly). This is limited to how the program represents itself in its current state! I.e. If the contents of an executable changes during execution - "Disassemblers" will not reflect this.

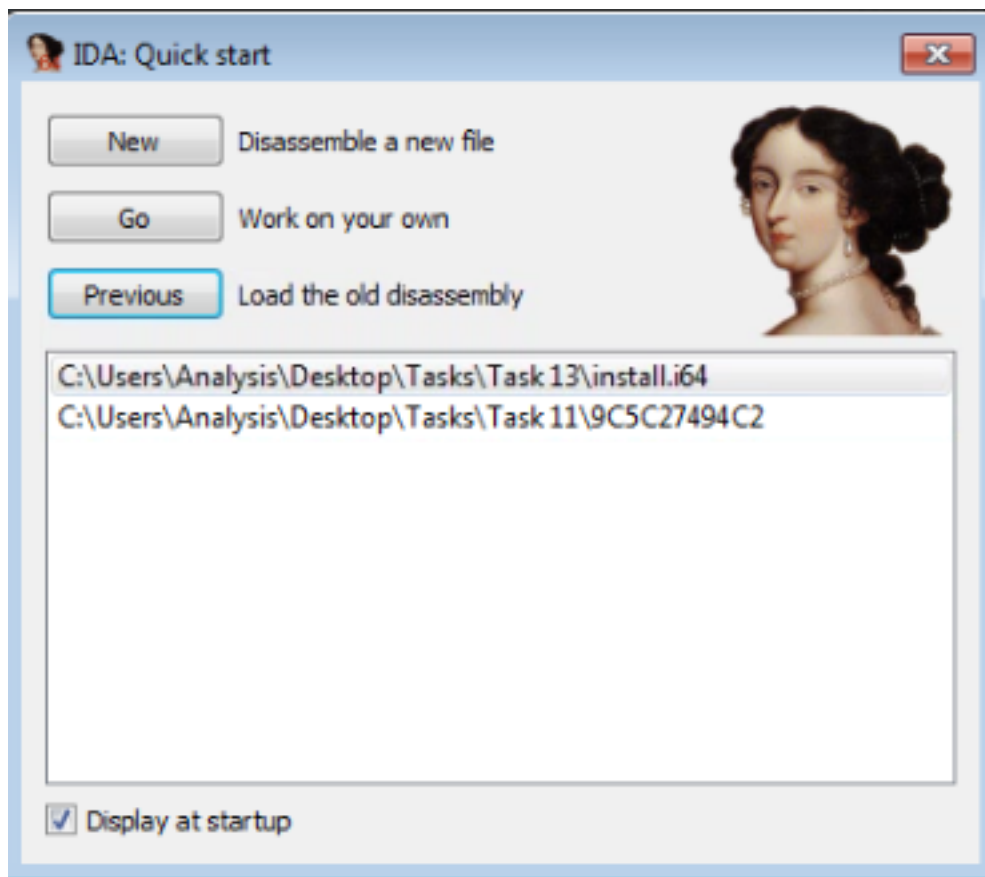
Whilst Debuggers deploy the same techniques used by "Disassemblers", "Debuggers" essentially facilitate execution of the program - where the analyser can view the changes made throughout each "step" of the program. These tools are great because a true picture of the program presents itself. However, if it is indeed malicious, you have now infected yourself.

With enough understanding, an analyser can introduce "breakpoints" (or pauses) at various stages of a program, where the program will execute up until a breakpoint. For example, sticking with the idea of Ransomware, an analyser can create a "breakpoint" within the application prior to the actual stage of encryption of files. This facilitates an analyser to view the various changes of a program during execution (such as unpacking or connecting to a remote server such as that in a botnet) up until the point of malicious infection.

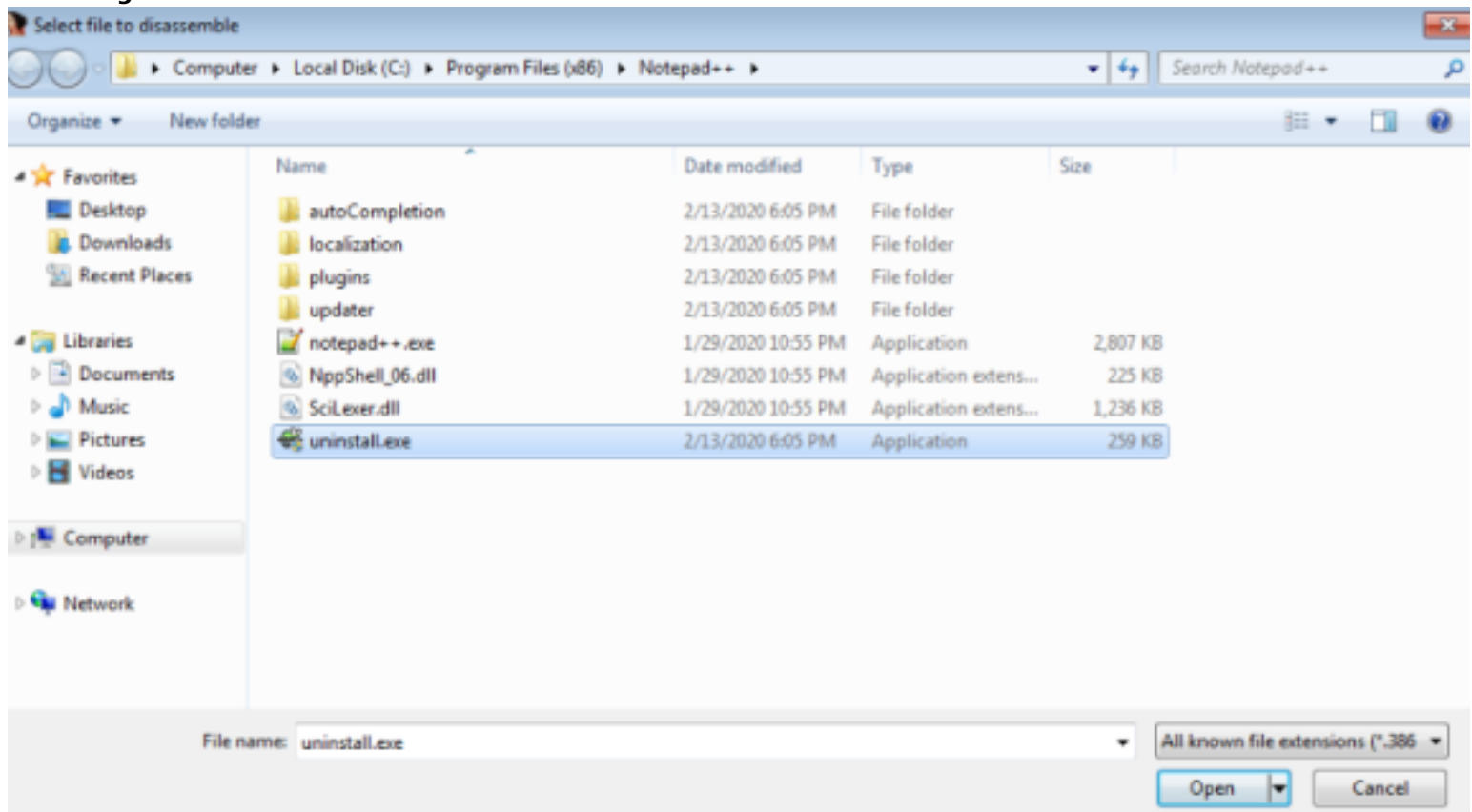
Practical:

For this room, we will be using IDA Freeware within the context of statistical analysis. I'll walkthrough how to import an executable into IDA Freeware below.

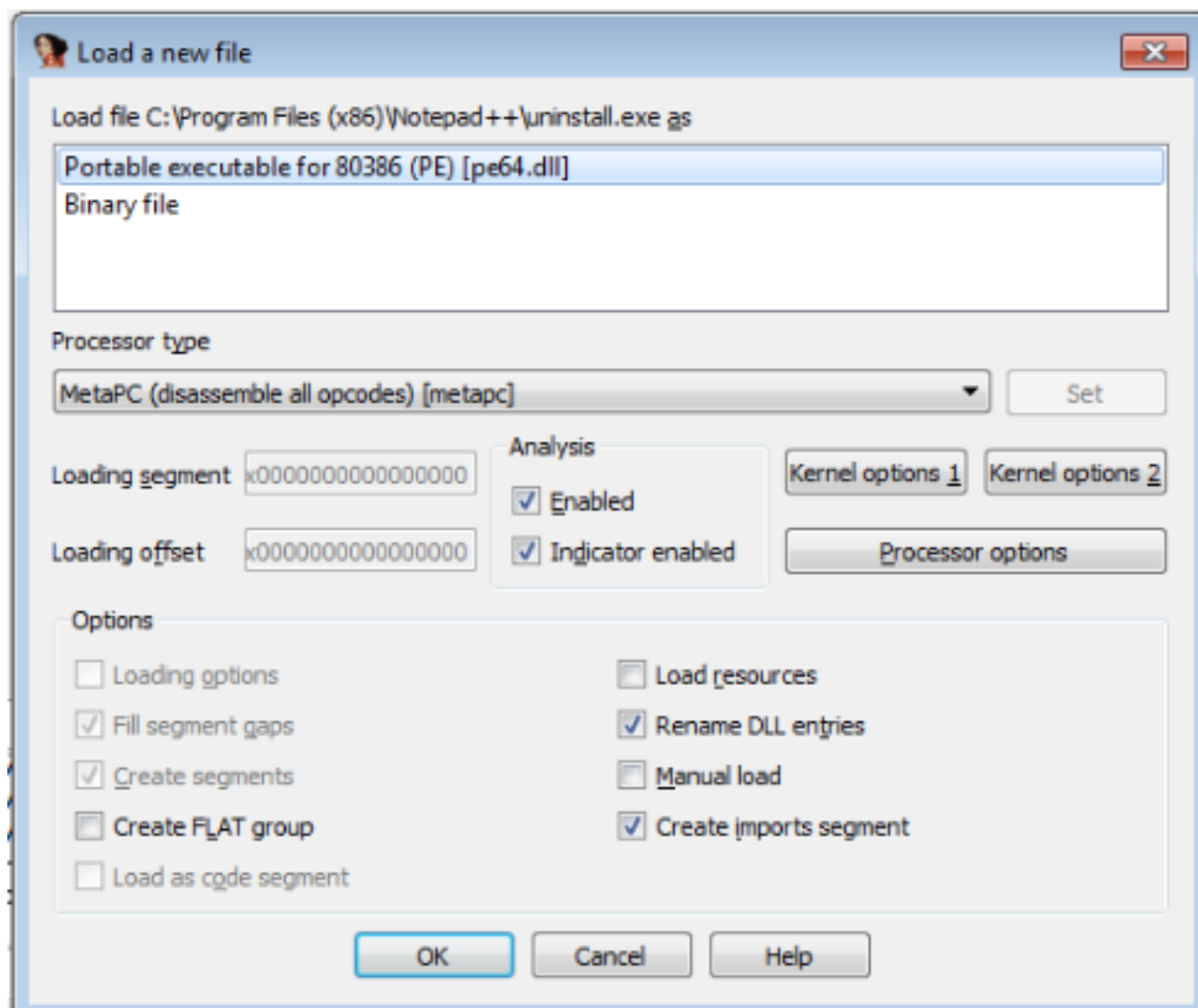
1. Lets launch "IDA Freeware" and select the file to import, in this case we'll be using "uninstall.exe"



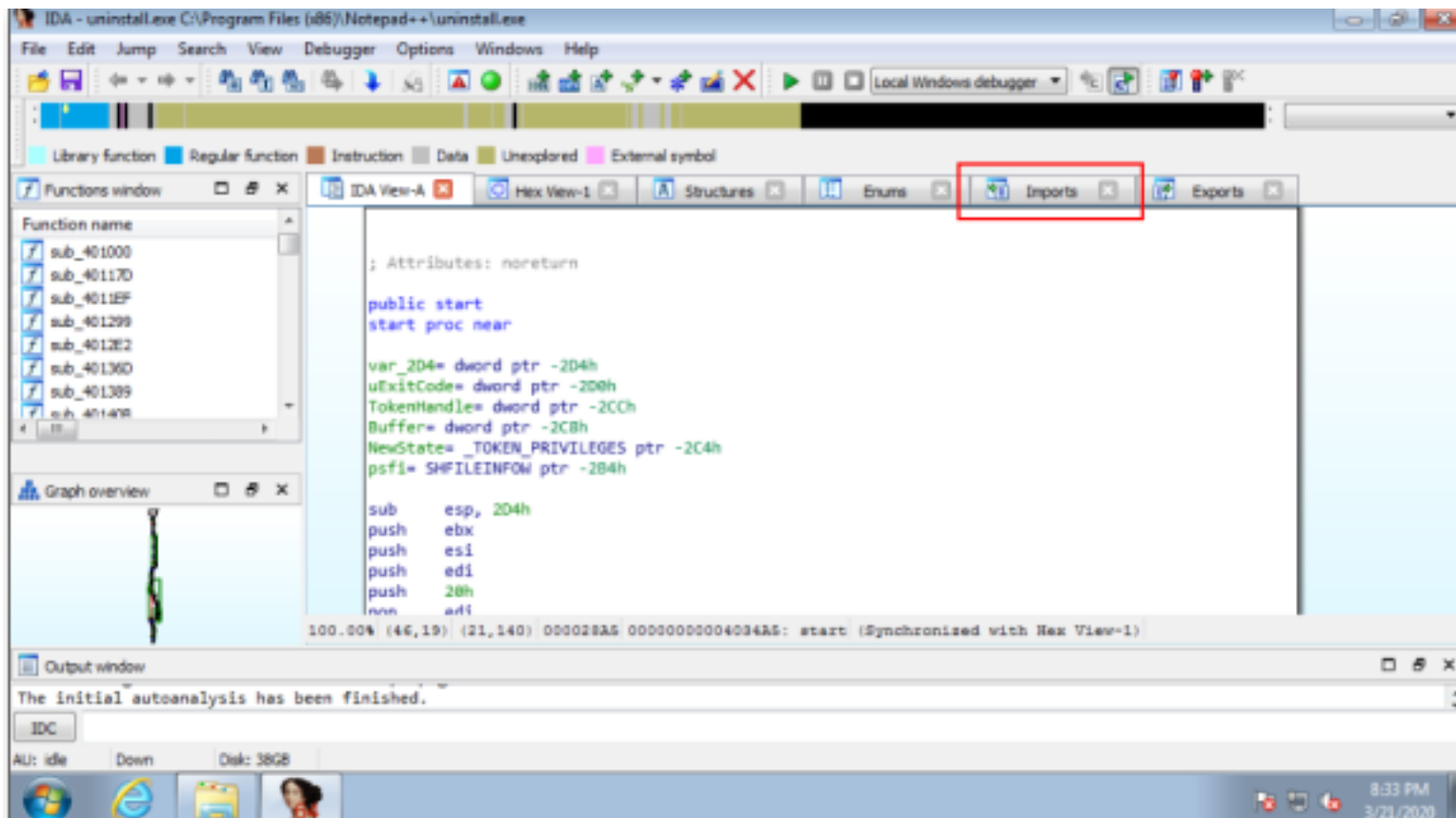
And navigate to the file...



2. Since we know it is an executable file, we select "Portable executable for 80386 (PE) [pe64.dll]"



3. After pressing "OK" the application will load. Allow a few minutes for the executable to be decompiled.



There are various tabs, similar to what we saw in "PE Explorer" i.e. "Imports" and "Exports".

Task:
 Navigate to the directory "**Tasks/Task 13**" and open "**install.exe**" with IDA Freeware, just like we did in the example above. Again, this may take a few seconds to a couple of minutes to compute dependant upon the size of the application. For this task expect roughly ~20 seconds.
 You can now answer the question:

#1
How many references are there to the library "msi" in the "Imports" tab of IDA Freeware for "install.exe"

9

[Task 14] Practical Summary

I'm not going to walk you through this one, but you have done all the necessary steps above to achieve this. GL HF :^
 If you struggle, revisit the techniques you used above. Moreover, if you're still stuck, visit the TryHackMe Discord! The file specified for analysis is "**ComplexCalculator.exe**" in the Directory "**Tasks/Task 14**". I'll leave it up to you to figure out what tool(s) out of what we've used above is best!

#1
What is the MD5 Checksum of the file?

F5BD8E6DC6782ED4DFA62B8215BDC429

#2
Does Virustotal report this file as malicious? (Yay/Nay)

yay

#3

Output the strings using Sysinternals "strings" tool.

What is the last string outputted?

d:h:

#4

What is the output of PeID when trying to detect what packer is used by the file?

Nothing found *

[Task 15] Future Aspirations

I'd love for this topic to be well received on TryHackMe. It is a very different yet prevalent topic within Cybersecurity that is a totally different and equally as dangerous context to "Booting and Rooting" a box.

However, because of its complexity and diversity, its extremely hard to approach - let alone decide how to introduce it to TryHackMe, so here are my future aspirations for this topic:

- Develop this into a series, where topics such as Strings, Imports and the differences between Static and Dynamic analysis can be discussed in dedicated rooms (rather than just briefly outlined throughout tasks)
- I've got some real interesting practical material planned that I'd love to share such as:
 - Network PCAP analysis of genuine infections that I've simulated/ran on my own environments outside of TryHackMe where the tasks will be practical analysis of this.
 - File system change (forensics) post-infection again from my dedicated environments
 - Dedicated analysis of samples such as the infamous Wannacry, to Stuxnet and Cryptolocker (to name a few)

#1

Hope you enjoyed the room!

No answer needed