

MAL: Researching



MAL: Researching

Understanding checksums, how to generate them and their use throughout malware analysis with online sandboxing & reporting services

[Task 1] Intro

Preface:

Welcome to **2. MP: Research** section of my malware analysis series.

Although we will be covering some cryptography theory, I've kept it relevant and visualised it as best I can. I've provided some resources at the end of the room in the "Further Reading" task for each of the topics covered. You can expect to learn about file checksums, why these values are important in not only day-to-day life but more so how we can utilise them in malware analysis. The first few tasks are theory-heavy, so bear with me. However, towards the end of the room, you will be generating your own checksums, learning how to use online sandboxing, and analysing the reports generated from these.

∴

As always, any feedback regarding the content covered through the series so far is greatly appreciated. I always welcome ideas of topics, tools, and techniques that could be covered.

∴

~CMNatic

#1

Let's go!

No answer needed

[Task 2] Deploy!

Deploy the instance attached to this task by left-clicking the green "Deploy" button on the top-right of this task!

You are not expected to interact with the instance just yet - you will be provided credentials later on throughout the room. But please ensure you are connected to the TryHackMe OpenVPN before proceeding.

#1

I've deployed my instance and ensured I am connected to the THM VPN!

**No answer
needed**

[Task 3] Checksums 101

What

are Checksums?

Checksums are a prominent attribute within the malware analysis community. But moreover, the wider Information Technology (IT) industry. Put simply, these checksums are the result of mathematical operations against an input - where the output is a sequence of characters.

Ultimately, the markup of data on a computer system is binary, merely ones and zeros where each value is a "bit". A cryptographic checksum uses these "bits" as the input for these mathematical operations; the increased complexity of the mathematical operations applied, the more secure a checksum is considered. These checksums are also commonly referred to as "hashes".

Because of how cryptographic algorithms work, regardless of the size of the input - such as a file - the length of the output will remain the same. For example, take an algorithm and apply these mathematical operations against two files listed below:

click me	click me
File Name	File Size
My_Video.mp4	4GB
My_Selfie.png	10MB

Although the file size is vastly different, the length of the output calculated will be the same, albeit its contents different. For example, in this algorithm I have as an example, the output length is 12 characters:

click me	click me	click me
File Name	File Size	The output from the Algorithm
My_Video.mp4	4GB	3DEFAD92D23AD
My_Selfie.png	10MB	FFDE312DAEFF

Whilst the values calculated from the algorithm are different from each file, they remain the same length - irrespective of the file size. Because the two files have different contents, in this case, each output is unique for the file.

The increased complexity of the mathematical operations vastly reduces the chances of two files with different contents from having the same output. If this was to occur, it is known as a "hash collision". Explaining the math behind how this happens is out of scope, however, it is extremely rare. To put into perspective, the hashing algorithm MD5 which is a famous recent example will need 6 billion files to be hashed per second - for 100 years on average. (Kornel., 2008)

Whilst it's pretty safe to say that the probability of a hash collision occurring is pretty low, it is mathematically possible. For example, researchers (Stevens et al., 2017) report on Shattered.io, where they were able to demonstrate a SHA1 hash collision in practice. I greatly encourage taking the time to read through the report to understand how these collisions can be made into proof of concepts but to also appreciate why discoveries like this are important in the world of information security.

It's safe to say it's pretty rare for this error to happen, although it is mathematically possible. We'll visualise this below.

Checksums Continued:

Let's contextualise checksums a bit more. In IT, checksums are used for verifying the integrity of data. Have you ever copied a file onto a USB drive where Windows complains that the file is now corrupt? That is data corruption; binary data was lost somewhere during the transfer process, either due to software or hardware error.

Due to how these cryptographic algorithms work, namely, they produce a result after processing a piece of data at every single bit, checksums are fantastic for verifying if data has completely copied to a new location. Humans can't compare the binary data (which could be millions of values to compare) of two files to ensure they are correct. They can, however, compare two-fixed length values - such as the 12 character length output of the algorithm specified

above. In the real-world, some popular algorithms are SHA1, SHA-256 and in some cases, SHA-512, where the output length of each algorithm is varied based on its security.

We'll visualise how hashing algorithms work below:

File Contents	Hashing Algorithm	Checksum
TryHackMe	MD5	1DEF54326AFF3
TryHackMe	MD5	1DEF54326AFF3
TryHackMe	SHA256	505878F9C2CE48D6DAB95
TryHackeM	MD5	ADE9D844ADEF

See here how the contents of the first two files are the same with "TryHackMe". When using the same algorithm, the generated checksum is the same. This is because the binary data of these two files are identical, so the same algorithm will output the same result. This is not a hash collision. Notice the third file with the same contents of "TryHackMe" resulting in the same binary data has a different checksum. This is because the algorithm, in this case, *SHA256* instead of the previous *MD5*, uses different mathematical operations. The mathematics behind this algorithm is complex in comparison to *MD5*, so the length of the output string is longer. Finally, notice in the screenshot below (the last file in the screenshot above, just cropped below for clarity) that the contents of the file are now "TryHackeM" and not "TryHackMe":

TryHackeM	MD5	ADE9D844ADEF
-----------	-----	--------------

Whilst the same algorithm used on the first two files are also used on this file (*MD5*) because the contents are now different - albeit very similar - the output is now different in comparison.

Visualising a Hash Collision:

Okay, bear with me here. It's been pretty theory-heavy I understand...We're almost there I promise! The example screenshots below outline how the various outputs will differ based upon either the algorithm used or the contents of the file. Here we will go through an example of a hash collision:

File Contents	Hashing Algorithm	Checksum
TryHackMe	MD5	1DEF54326AFF3
TryHackMe	MD5	1DEF54326AFF3
TryHackMe	SHA256	505878F9C2CE48D6DAB95
TryHackeM	MD5	ADE9D844ADEF

The contents of the two files above are very different. In the real world, the differences could be simply adding or removing a character, but I have made dramatised this example to visualise things better.

A hash collision is when two files with different content (such as the two above) have the same output. From a mathematics perspective, these files are identical. However, we know by the contents that they are not at all.

#1

Name the term for an individual piece of binary

bit

#2

What are checksums also known as?

hashes

#3

Name the algorithm that is next in the series after SHA-256

sha-512

#4

According to this task, how long will you need to hash 6 million files before a MD5 hash collision occurs?

100 years

#5

Who developed the MD5 algorithm?

Ronald Rivest

[Task 4] Online Sandboxing

What is online Sandboxing?

Sometimes things are left best to the experts. That's especially true in the case of malware analysis. However, online sandboxing hosts use to a wider audience than just hobbyists.

In the context of information security, sandboxing is the technique used to isolate processes to prevent direct interaction with one another. There are many examples of this. For example, using Virtualbox as a Hypervisor to run the Kali Linux operating system virtually, in parallel on your main computer. The processes within Kali Linux interact only through the means of Virtualbox and has no interference with processes on your main operating system, such as Windows.

In a malware analysis context, analysts employ virtual environments - such as those on TryHackMe, to facilitate analysis of potentially malicious code more securely.

Now, with this being said, malware has been known and can very well escape this virtual environment onto the analyst's host system. Whilst this risk is somewhat limited to sophisticated malware, it's very achievable. For example, CVE-2018-2689 is a CVE for Virtualbox where malware was capable of escaping this restricted virtual environment. CVE's such as these are extremely valuable and seldom disclosed due to the actors who discover them and their intentions, namely malicious malware authors.

I've written more about how malware detects it is in a virtual environment and the possible routes it can take to escape on my blog. What should be taken away from this task is that a virtual environment alone does not protect you from malware. Simply, virtual environments merely provide a convenient platform to analyse code.

Simply, an online sandbox is this virtual environment - but placed online by services such as:

- any.run
- hybrid-analysis

These services are fantastic, as it allows hobbyists to begin understanding how malware behaves with no detriment or risk to themselves. Moreover, online sandboxing platforms are highly sophisticated and are likely to report behaviours that an analyst may have missed.

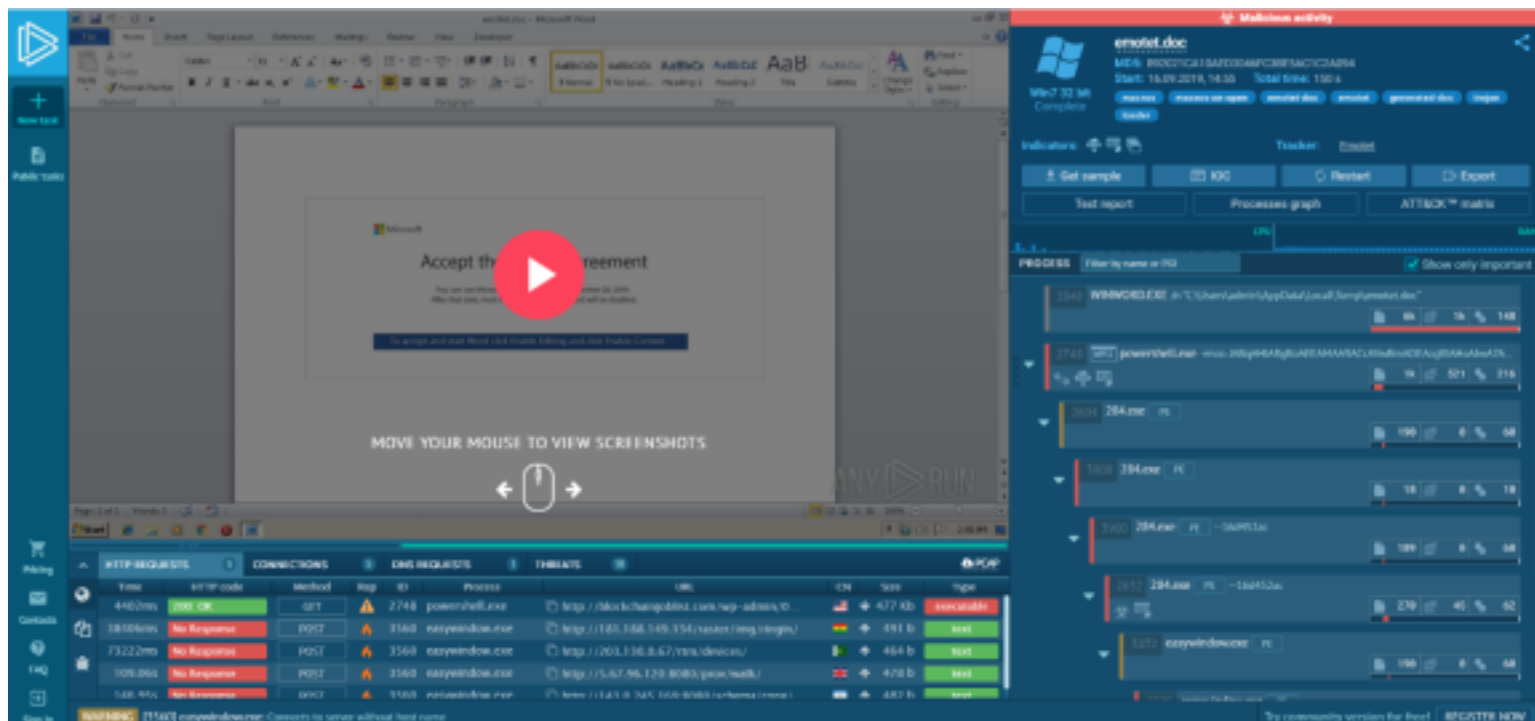
With this said, automated analysis cannot replace the skill and depth that an analyst can exhibit and traverse too. For example, reverse engineering. These platforms are only capable of executing malware and generating reports based upon interactions made with the operating system, any communication attempts and any signatures left behind. For example:

- Contacting a domain name (DNS Lookups, etc)
- Creating registry keys
- Read/Writing files
- Creating system processes
- Maintaining persistent through system startup entries

All of which are all discoverable by an analyst after some time. Therefore, online sandboxes are useful for a precursory inspection of a file.

Interacting with an online sandboxing Service:

In the example screenshots below, this sample was run through the hybrid-analysis service. The sample took a total of 10 minutes and was free of charge. The report detailed an extensive number of behaviours such as networking traffic and the execution chain, this would have taken an analyst a considerable amount of time to of detailed themselves.



Note how the file is still identified only by its "Checksums":

MIME: application/vnd.openxmlformats-officedocument.wordprocessingml.document
 File info: Microsoft Word 2007+
 MD5 B92021CA10AED3046FC3BE5AC1C2A094
 SHA1 0FB1AD5B53CDD09A7268C823EC796A6E623F086F
 SHA256 C378387344E0A552DC065DE6BFA607FD26E0B5C569751C79FBF9C6F2E91C9807
 SSDEEP 3072:/MSKNOK2ER/YR5DPQKAJNDU1CKBWN0PQJFWSQ:ZKOROKDPQZQQKMN0SCR

If an analyst was to now search google with any of these checksums, the report generated by this sandboxing engine will now be provided as a listing for future analysts. Proceeding to read through the report, interesting behaviours are summarised such as those below:

Registry activity



Total events	Read events	Write events	Delete events
2093	1329	763	1

Modification events

PID	Process	Operation	Key	Name	Value
2848	WINWORD.EXE	delete key	HKEY_CURRENT_USER\Software\Microsoft\Office\14.0\Word\Resiliency\StartupItems		
2848	WINWORD.EXE	write	HKEY_CURRENT_USER\Software\Microsoft\Office\14.0\Word\Resiliency\StartupItems	d\1	846021002008000001000000000000000000000000
2848	WINWORD.EXE	write	HKEY_CURRENT_USER\Software\Microsoft\Office\14.0\Common\LanguageResources\EnabledLanguages	1033	Off
2848	WINWORD.EXE	write	HKEY_CURRENT_USER\Software\Microsoft\Office\14.0\Common\LanguageResources\EnabledLanguages	1033	On
2848	WINWORD.EXE	write	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\UserData\S-1-5-18\Products\00004106D300000000000000F01FEC\Usage	WORDFiles	1328545822
2848	WINWORD.EXE	write	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\UserData\S-1-5-18\Products\00004106D300000000000000F01FEC\Usage	ProductFiles	1328545936
2848	WINWORD.EXE	write	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\UserData\S-1-5-18\Products\00004106D300000000000000F01FEC\Usage	ProductFiles	1328545937
2848	WINWORD.EXE	write	HKEY_CURRENT_USER\Software\Microsoft\Office\14.0\Word	MTTT	200B00008E190A58968CD50100000000
2848	WINWORD.EXE	write	HKEY_CURRENT_USER\Software\Microsoft\Office\14.0\Word\Resiliency\StartupItems	jal	

Network activity



HTTP(S) requests	TCP/UDP connections	DNS requests	Threats
5	5	1	18

HTTP requests

To answer the following questions, read through this analysed sample to solve the following questions:

#1

Name the key term for the type of malware that Emotet is classified as

trojan

#2

Research time! What type of emails does Emotet use as its payload?

spam emails

#3

Begin analysing the report, what is the timestamp of when the analysis was made?

9/16/2019, 15:54:48

#4

Name the file that is detected as a "Network Trojan"

easywindow.exe

#5

What is the PID of the first HTTP GET request?

2748

#6

What is the only DNS request that is made after the sample is executed?

blockchainjoblist.com

[Task 5] Practical: Calculating & Reporting Checksums

Calculating the MD5 Checksums of provided material:

You will be able to interact with your instance using the in-browser functionality, however, you may connect via RDP using the details below - ensuring you are connected to the TryHackMe VPN beforehand.

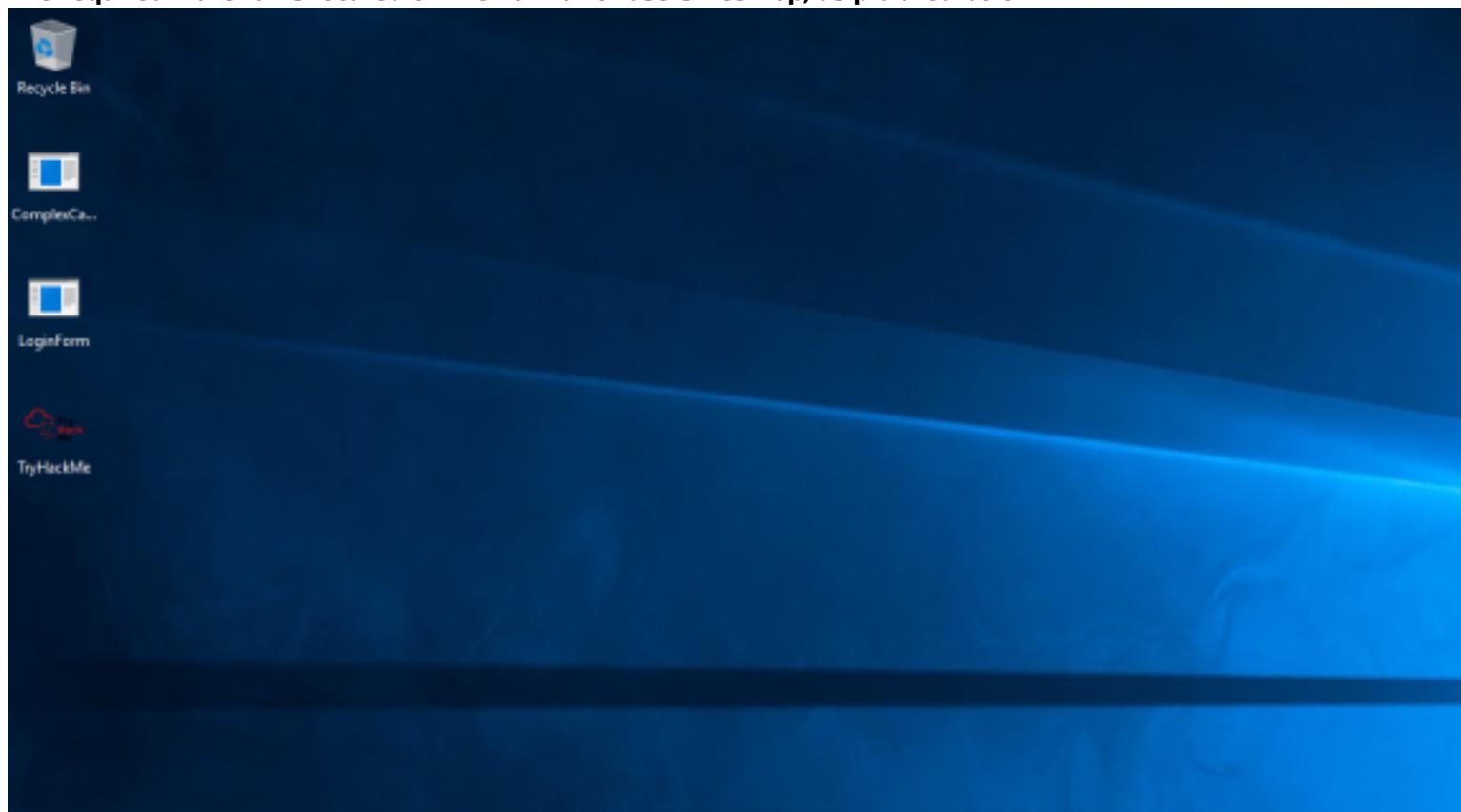
IP Address: MACHINE_IP

Username: **cmnatic**

Password: **Tryhackm3!**

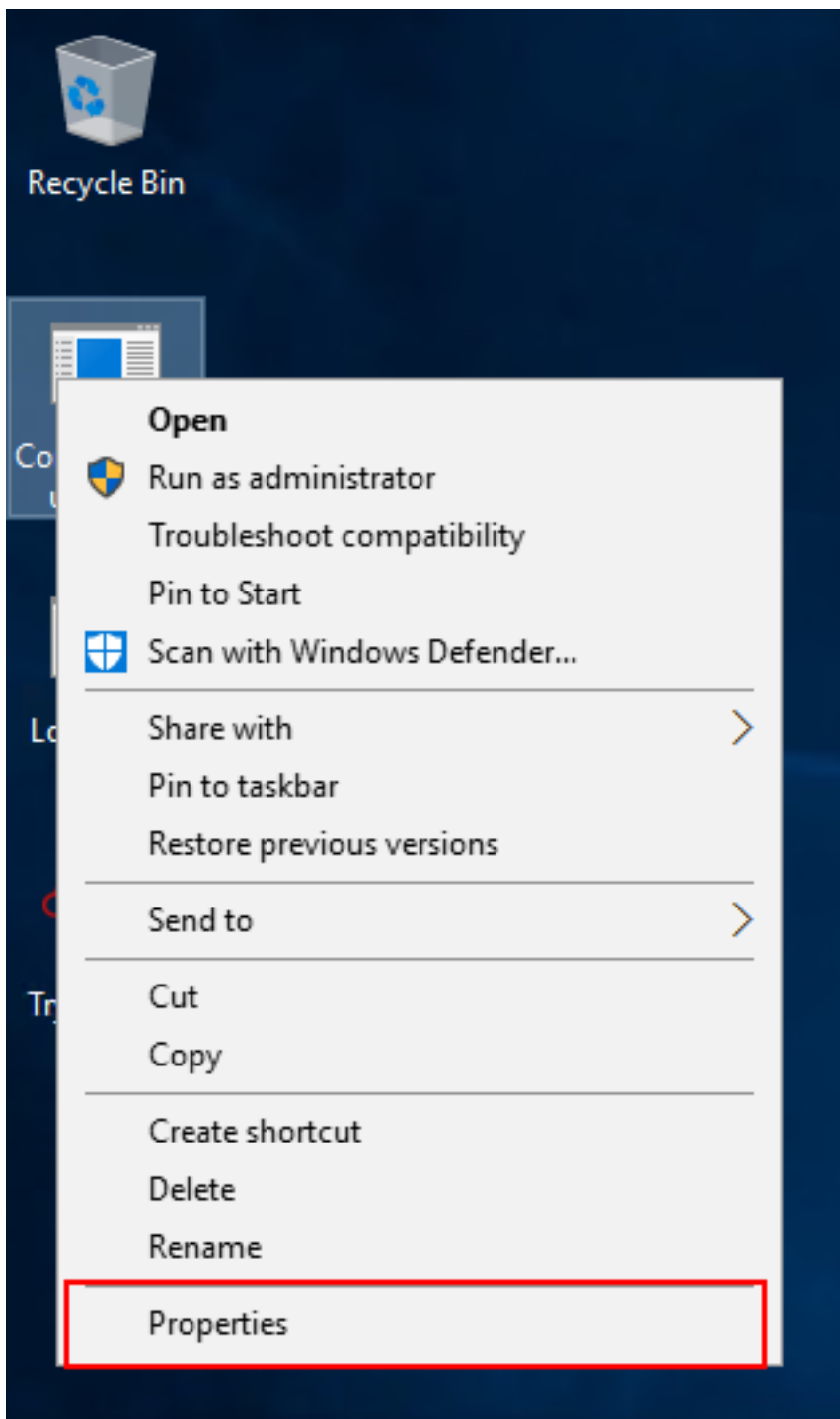
I have provided the tools and materials on this Instance for you to complete the questions. I will go through obtaining the MD5 Checksum of a file using two methods - you will apply these techniques to answer the questions for this task.

The required material is located on the "cmnatic" users Desktop, as pictured below:

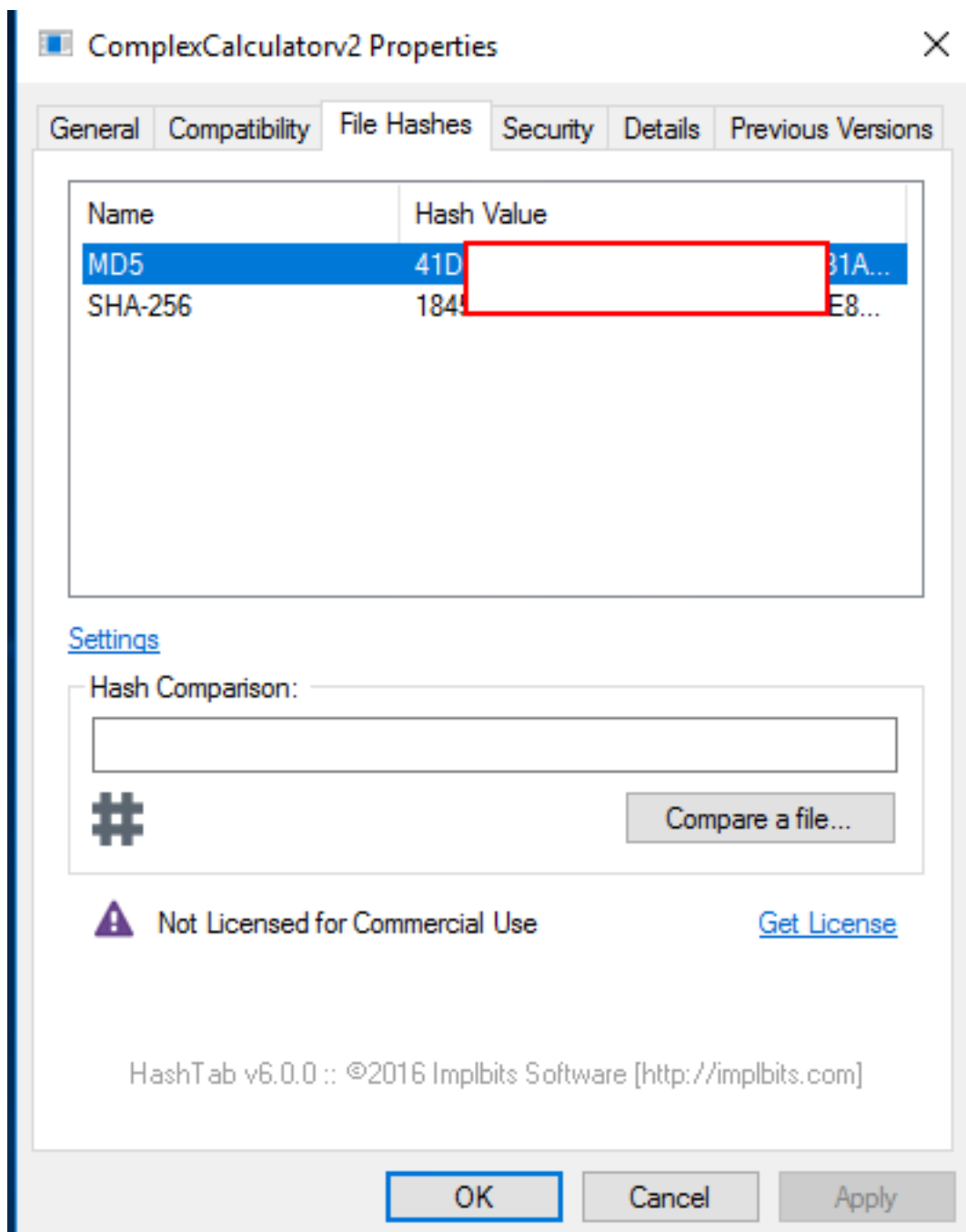


Using the 3rd-party application "HashTab":

1. Right-click the file you wish to retrieve the checksum of. I will be using "ComplexCalculatorv2" in this example.
2. Left-click the "Properties" title in the drop-down.

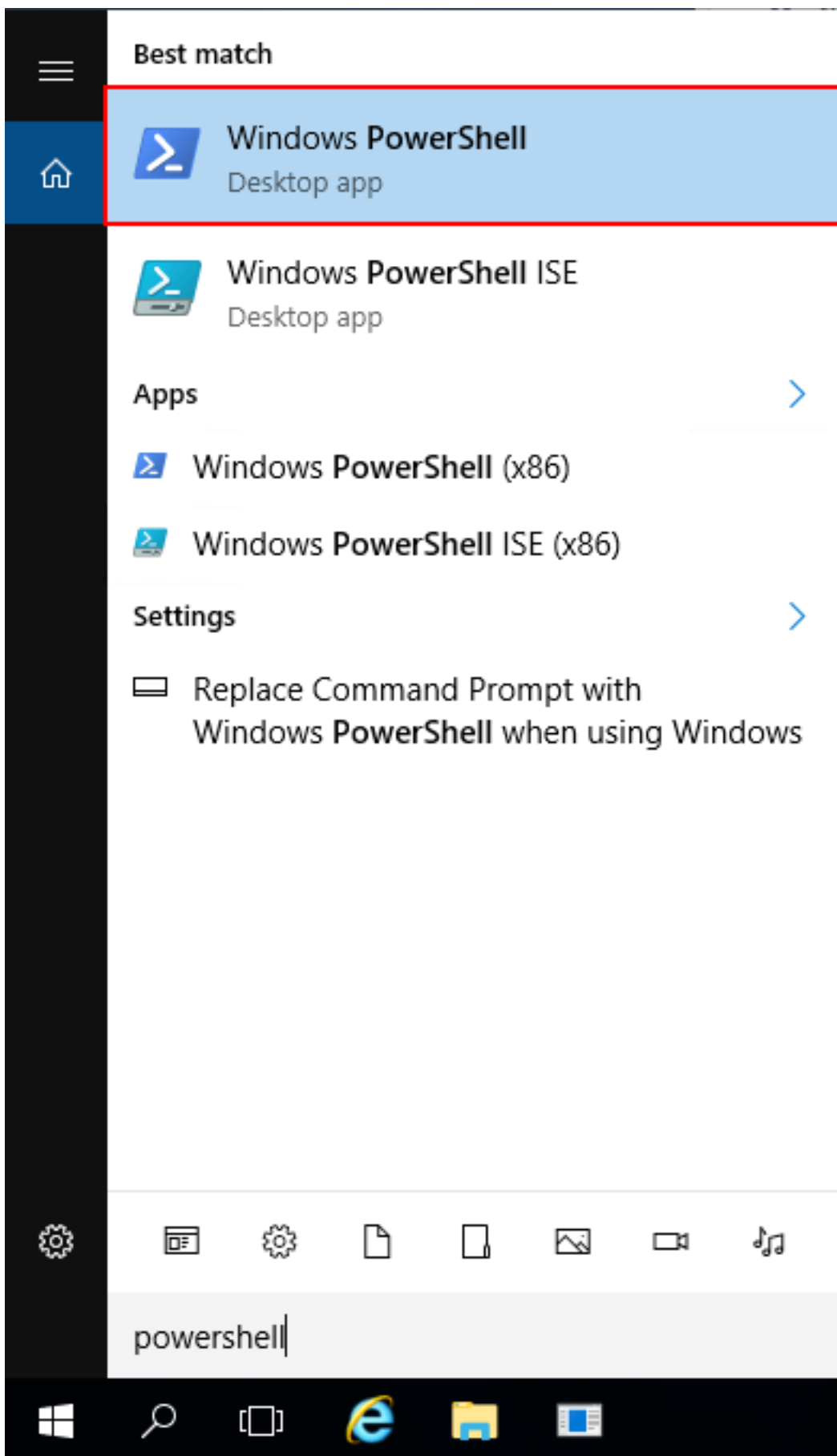


3. In the popup, navigate to the "File Hashes" tab, where you will see a screenshot akin to the one below. Note that this tab is not present on a default Windows installation:



You can now answer question 1.
Using Windows' "Powershell":

1. Firstly we will need to open up "Powershell". You can do this opening the Windows Search bar.



2. Next, change the directory the users Desktop by using `cd Desktop`
3. Verify you are in the right directory by using `dir` to list the files in the directory. You should see the three below:

```
Windows PowerShell
PS C:\Users\cmnatic> cd Desktop
PS C:\Users\cmnatic\Desktop> dir

Directory: C:\Users\cmnatic\Desktop

Mode                LastWriteTime         Length Name
----                -
-a----          21/05/2020   11:19             73216 ComplexCalculatorv2.exe
-a----          21/05/2020   11:20             17408 LoginForm.exe
-a----          28/02/2020   12:25          6789120 TryHackMe.exe

PS C:\Users\cmnatic\Desktop> _
```

Powershell has both CertUtil and File-Hash commands that allow us to retrieve various checksums of files, including MD5, SHA1, SHA2, and SHA-256. I will detail the syntax for both below, calculating the MD5 checksum of a file.

Using CertUtil:

CertUtil -hashfile ComplexCalculatorv2.exe MD5|SHA256|SHA512 or "CertUtil -hashfile <filename> <algorithm>" such as in the example below:

```
PS C:\Users\cmnatic\Desktop> CertUtil -hashfile ComplexCalculatorv2.exe MD5
MD5 hash of file ComplexCalculatorv2.exe:
41 [REDACTED] d2
CertUtil: -hashfile command completed successfully.
PS C:\Users\cmnatic\Desktop> _
```

Using FileHash:

Get-FileHash file_name -Algorithm MD5|SHA256|SHA512

```
PS C:\Users\CMNatic\Desktop\dis> Get-FileHash malware_dataset.csv -Algorithm MD5

Algorithm      Hash                                     Path
-----
MD5            8EE918EAB25C0697B004D49B6774E956      C:\Us

PS C:\Users\CMNatic\Desktop\dis>
```

Now you can proceed to answer the remaining questions!

#1

Using the HashTab tool, what is the MD5 checksum for "LoginForm.exe"?

FF395A6D528DC5724BCDE9C844A0EE89

#2

Using Get-FileHash in Powershell, retrieve the SHA256 of "TryHackMe.exe"

6F870C80361062E8631282D31A16872835F7962222457730BC55676A61A

#3

What would be the syntax to retrieve the SHA256 checksum of "TryHackMe.exe" using CertUtil in Powershell?

CertUtil -hashfile TryHackMe.exe SHA256

[Task 6] VirusTotal

Another online service that utilises these checksums is Virustotal. Virustotal acts as an indexer and aggregator for various Anti Virus (AV) engines. When a checksum is submitted to Virsutotal, fellow malware analysts can view the AV reports attributed to that file.



Analyze suspicious files and URLs to detect types of malware,
automatically share them with the security community

FILE

URL

SEARCH

A magnifying glass icon with a fingerprint pattern inside the lens, indicating a search function.

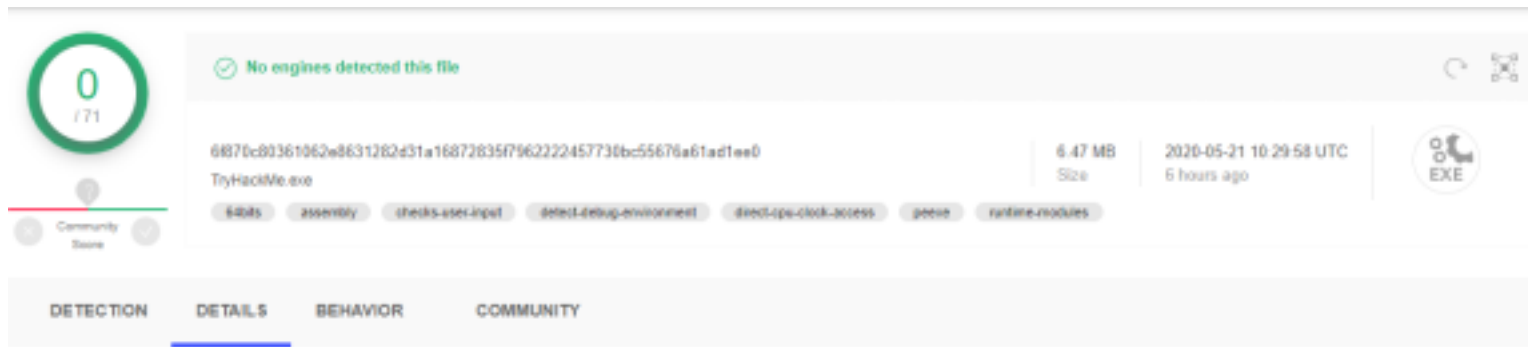
URL, IP address, domain, or file hash

By submitting data below, you are agreeing to our [Terms of Service](#) and [Privacy Policy](#), and to the sharing of your Sample submission with the security community. Please do not submit any personal information; VirusTotal is not responsible for the contents of your submission. [Learn more.](#)

Much like a search engine, you can search for reports by a few characteristics, for example:

- The IP Addresses that samples communicate with
- Checksums
- The file itself

In the screenshot below, I have uploaded the "TryHackMe.exe" executable to Virustotal. If you were to browse to VirusTotal, you would be able to discover this report by entering the files' checksum. I have provided the report for you here.



The screenshot shows the VirusTotal interface for a file named 'TryHackMe.exe'. At the top left, there is a green circle with the number '0' and '/71' below it, indicating the number of engines that have scanned the file. To the right of this, a green checkmark icon is followed by the text 'No engines detected this file'. Below this, the file's SHA-256 hash is displayed: '68876c80361062e8631282d31a16872835f796222457738bc55676a61ad1ee0'. The file name 'TryHackMe.exe' is shown below the hash. To the right of the file name, the file size is listed as '6.47 MB' and the upload date is '2020-05-21 10:29:58 UTC', with '6 hours ago' below it. On the far right, there is a circular icon with 'EXE' inside. Below the file name, there are several tags: '64bits', 'assembly', 'checks-user-input', 'detect-debug-environment', 'detect-ops-clock-access', 'pcode', and 'runtime-modules'. At the bottom of the interface, there are four tabs: 'DETECTION', 'DETAILS', 'BEHAVIOR', and 'COMMUNITY'. The 'DETAILS' tab is currently selected and highlighted with a blue underline.

A THM Contributor, Darkstar7471 has a fantastic room on using both Volatility, a memory analysis framework and Virustotal. I highly recommend checking it out as you extract files and interact with VirusTotal to determine their maliciousness from the aggregated AV ratings.

Read the report provided (here) to answer the questions provided.

#1

Navigate to the "Details" tab, what is the other filename and extension reported as present? f

HxD.exe

#2

In the same "Details" tab, what is the reported compilation timestamp?

2020-02-28 11:16:36

#3

What is the THM{} formatted flag on the report?

THM{TryHackMe_Malware_Series_Research_Flag}

[Task 7] Future Reading (References)

Cryptography and Checksums:

A Meaningful MD5 Hash Collision Attack - (Narayana D. Kashyap., 2008)

Cryptography & Network Security - (Behrouz A. Forozuan., 2007)

The first collision for full SHA-1 - (Stevens et al., 2017) / (Shattered.io)

Blog (Selfless Promo)

So you want to analyse malware? - (CMNatic., 2020)

Sandboxing Engines:

any.run

hybrid-analysis

#1

Thanks! I'll stay tuned for more

No answer needed