**Encoding vs Hashing vs Encryption**

Legend:

Input Command

Output of the previous command

# EXERCISE 2 - Hashing

## Task 1

echo hello world > file1.txt

echo hello world > file2.txt

md5sum file1.txt

6f5902ac237024bdd0c176cb93063dc4  file1.txt

md5sum file2.txt

6f5902ac237024bdd0c176cb93063dc4  file2.txt

Please note that the MD5 hash of the outputs are identical to each other.

That means when you created the files, they were identical, so were their hashes.

This proves the deterministic nature of using the same hash function, which in this case was md5.

## Task 2

echo hello world! > file3.txt

md5sum file3.txt

c897d1410af8f2c74fba11b1db511e9e  file3.txt

Please note that be inputting different text into file3, that the hash is wildly different that the files in Task 1.  This is due to the Avalanche effect in Hashing.

## Task 3

wget https://github.com/ajay63/BlackTowerAcademy/blob/main/catpicturess.jpg

md5sum catpicturess.jpg

4e89d194f2575c3a1636bcb1091a0f9f catpicturess.jpg

Let's briefly check virustotal

Browse to virus total: https://www.virustotal.com/gui/home/search

Copy the above has for catpicturess.jpg and paste it into the search field. Enter it into the system.

It should return with "No Matches Found" which is GREAT NEWS!

I didn't have you download a virus! 😂

## Task 4

Let's take this up a couple of notches.

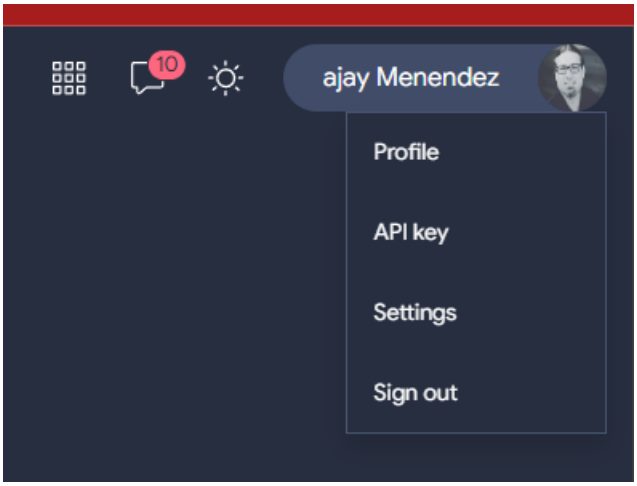
Start in the home folder of your Linux Server

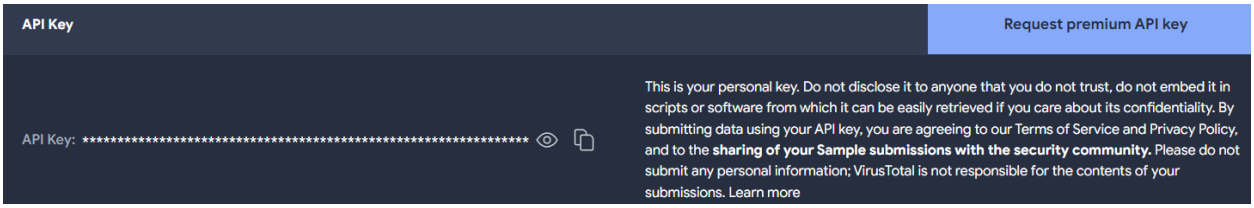
Step 1

Open a browser to https://www.virustotal.com

Create an Account

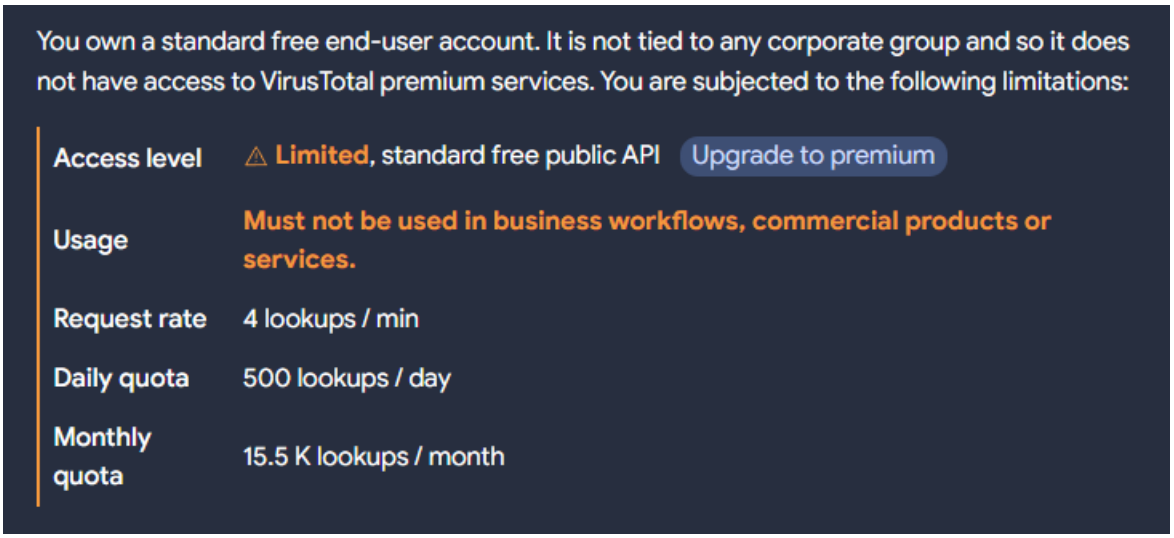
Once you create an account, navigate to the API key






You'll need to copy the API key into the VirusTotal software later, don't navigate away from this page.

It has some limitations unless you are willing to pay for it, but for our educational purposes it absolutely provides exactly what we need.




Step 2

In your Linux Server

We have to download the pre-compiled VirusTotal Application.

Use the following command to download the zipped file.

wget https://github.com/VirusTotal/vt-cli/releases/download/1.0.0/Linux64.zip

Output:

A bunch of stuff and

Linux64.zip
100%[==========================================================================>] 6.79M 21.4MB/s in 0.3s

2024-02-21 23:20:55 (21.4 MB/s) - ‘Linux64.zip’ saved [7120233/7120233]

This means that it was downloaded successfully.

List the directory to check.

ls

and Linux 64.zip should be there.

Now, to unzip this zipped file, we’ll need to install unzip. Online Guide

sudo apt install unzip -y

Time to unzip the VirusTotal application.

unzip Linux64.zip

Archive: Linux64.zip

inflating: vt

Now run ls and you should see the vt application in your folder. (Hopefully your home folder)

Time to initialize the vt or VirusTotal Application. Please note you’ll need to use dot slash before the application to get it to work.

./vt init


VirusTotal Command-Line Interface: Threat Intelligence at your fingertips.

Enter your API key:

Now go copy the API key from the Virustotal website, and paste it here right clicking on your mouse to paste.

Your API key has been written to config file /home/ajay/.vt.toml

NOW! We are ready to start using Virus Total from the CLI!\

Step 3

md5sum catpicturess.jpg

4e89d194f2575c3a1636bcb1091a0f9f  catpicturess.jpg

Now we are going to check this hash in virustotal from the CLI.  Please note you'll need to use dot slash before the application to get it to work.

Understand the syntax  ./  then vt  then file and then the hash.

./vt file 4e89d194f2575c3a1636bcb1091a0f9f

File "4e89d194f2575c3a1636bcb1091a0f9f" not found

HURRAY! The cat picture is still not a virus, according to Virus Total.

Step 4

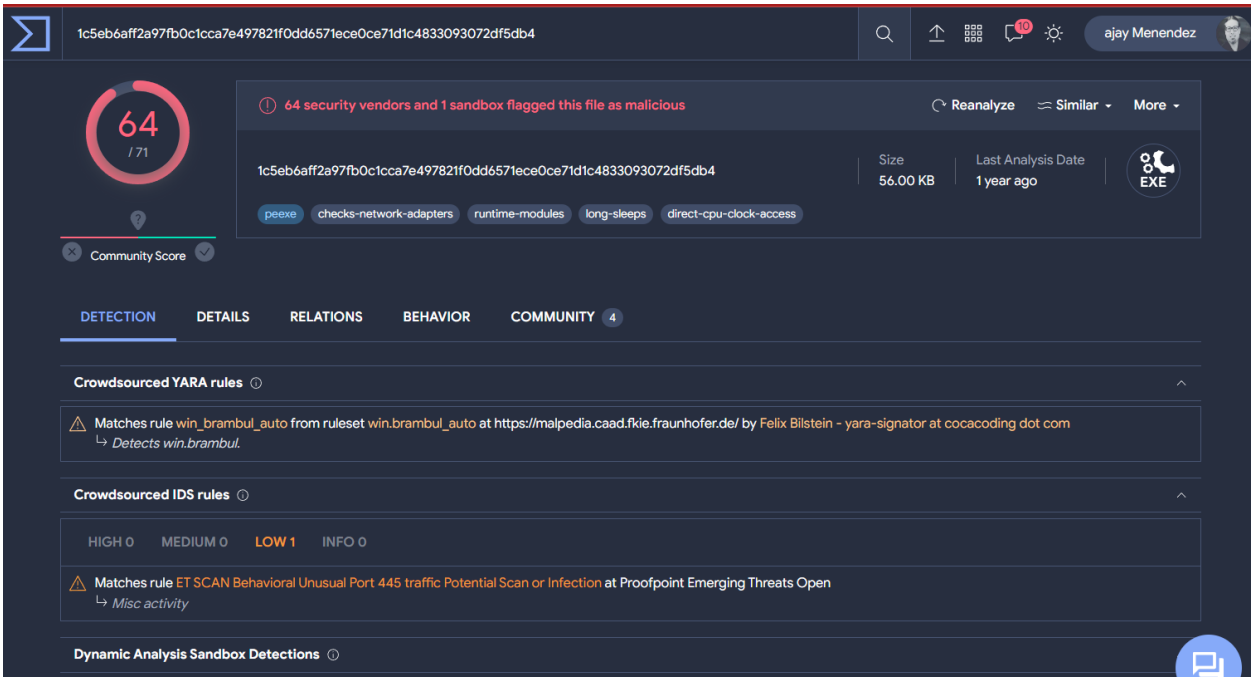
Let's use a known bad hash.

Let's pull up our Virus Total Web page, and paste the following known bad hash:

00434c7dabe90c49dfcb78038e7595e1cfb87851


And let's press the search icon on the right.

Oh my!




So… that's really bad.

Lots of detections

| | | | |
|---|---|---|---|
| Kingsoft | Win32.Troj.Agent.(kcloud) | Lionic | Trojan.Win32.Brambul.tn9U |
| Malwarebytes | Trojan.Agent | MAX | Malware (ai Score=100) |
| McAfee | Downloader-CUZ | McAfee-GW-Edition | BehavesLike.Win32.Generic.qh |
| Microsoft | Trojan:Win32/Brambul.A!dha | NANO-Antivirus | Trojan.Win32.Agent.bmgds |
| Palo Alto Networks | Generic.ml | Panda | Generic Malware |
| QuickHeal | Trojan.BrambulRI.S18867290 | Rising | Backdoor.Win32.Mnless.diy (CLASSIC) |
| Sangfor Engine Zero | [ARMADILLO V1.71] | SecureAge | Malicious |
| SentinelOne (Static ML) | Static AI - Malicious PE | Sophos | ML/PE-A + Mal/Spy-Y |
| SUPERAntiSpyware | Trojan.Agent/Gen-Downloader | Symantec | W32.Brambul |
| TACHYON | Trojan-Spy/W32.Agent.57344.KT | Tencent | Trojan.Win32.Agent.spy |
| Trapmine | Suspicious.low.ml.score | Trellix (FireEye) | Generic.mg.d7613345d28327ea |
| TrendMicro | WORM_MYDOOM.SMB | TrendMicro-HouseCall | WORM_MYDOOM.SMB |
| VBA32 | BScope.Worm.Agent | VIPRE | Trojan.GenericKDZ.90234 |
| VirIT | Backdoor.RBot.BZ | ViRobot | Trojan.Win32.Agent.57344.UI |

But, let's see how this looks like doing it via the command line. =)

./vt file 4e89d194f2575c3a1636bcb1091a0f9f


- _id: "1c5eb6aff2a97fb0c1cca7e497821f0dd6571ece0ce71d1c4833093072df5db4"

  _type: "file"

  authentihash: "ff3530eb0fcd6f36777a9dd5c2fa211b8841ced09736c673645ee803db73eb7e"

  creation_date: 1255524354  # 2009-10-14 12:45:54 +0000 UTC

  crowdsourced_ids_results:

  - alert_context:

    - dest_ip: "132.206.96.7"

      dest_port: 445

    alert_severity: "low"

    rule_category: "Misc activity"

    rule_id: "1:2001569"

    rule_msg: "ET SCAN Behavioral Unusual Port 445 traffic Potential Scan or Infection"

    rule_raw: "alert tcp $HOME_NET any -> any 445 (msg:\"ET SCAN Behavioral Unusual Port 445 traffic Potential Scan or Infection\"; flow:to_server; flags: S,12; threshold: type both, track by_src, count 70 , seconds 60; reference:url,doc.emergingthreats.net/2001569; classtype:misc-activity; sid:2001569; rev:15; metadata:created_at 2010_07_30, former_category SCAN, updated_at 2017_05_11;)"

    rule_references:

    - "https://doc.emergingthreats.net/2001569"

    rule_source: "Proofpoint Emerging Threats Open"


And it goes on and on and on, wow! That is a huge output! How can we control or manage this?

Step 5

Let’s redirect the output to a file, so we can control what we are seeing.

./vt file 00434c7dabe90c49dfcb78038e7595e1cfb87851 > ebil.txt

Now, we can use any of the Linux Text READERS to check out the file.

Less, More, Cat, Head, Tail?

What seems like it would be the most useful?  The author is going to use less.

less ebil.txt

Now by using the command line interface text connection to virus total, I can bring in automatable log enrichment and information that can be used with other code to provide good threat intelligence and way to populate cybersecurity systems.

# Hashing

Hashing is a process that takes an input (or 'message') and returns a fixed-size string of bytes. The output, known as the hash value or hash code, is typically a hexadecimal number that represents the content of the input. Hash functions are designed to be a one-way function, meaning it should be computationally infeasible to reverse the process and retrieve the original input from the hash output.

In the given example, two files (`file1.txt` and `file2.txt`) contain the exact same content, "hello world". When the `md5sum` command is applied to both files, they produce the same hash value (`6f5902ac237024bdd0c176cb93063dc4`).

This demonstrates a fundamental property of hash functions: consistency or determinism. If the input doesn't change, the hash value will always be the same, regardless of how many times the hash function is executed.

However, when even a small change is made to the input, as seen with `file3.txt` where an exclamation mark is added to "hello world" (resulting in "hello world!"), the hash value changes significantly (`c897d1410af8f2c74fba11b1db511e9e`). This illustrates another critical property of hash functions: sensitivity to input changes. A tiny change in the input data (even just adding a punctuation mark) results in a completely different output, which is a characteristic known as the avalanche effect.

The avalanche effect in hashing refers to a desirable property of cryptographic hash functions where a small change in the input data (even just a single bit) results in a significantly different output hash. This means that the hash values of two similar but not identical inputs will appear completely unrelated. The effect ensures that the hash function is highly sensitive to input changes, enhancing security by making it impractical to predict the hash value of a slightly altered input based on the hash value of the original input.

For instance, if you have two input strings that differ by just one character or bit, their respective hash values will be vastly different from each other. This characteristic is crucial for security applications, as it prevents attackers from guessing the input based on the hash and makes it extremely difficult to find two different inputs that produce the same output hash (a situation known as a collision). The avalanche effect thus plays a vital role in ensuring data integrity, authenticity, and the overall robustness of cryptographic systems.

## Key Points About Hashing:

- **Deterministic**: The same input will always produce the same output.
- **Fixed Size**: No matter the size of the input data, the hash value size is constant.
- **Efficiency**: The function should be capable of returning the hash value in a short amount of time, making it practical for various applications.
- **Avalanche Effect**: A minor change in the input should produce a significantly different output.
- **Infeasibility of Finding Collisions**: It should be extremely difficult to find two different inputs that produce the same output hash.

Hash functions like MD5 are widely used in various applications, including data integrity verification, password storage, and digital signatures. However, due to vulnerabilities discovered in MD5, it's no longer recommended for cryptographic security purposes. More secure algorithms like SHA-256 are preferred for applications requiring high security.