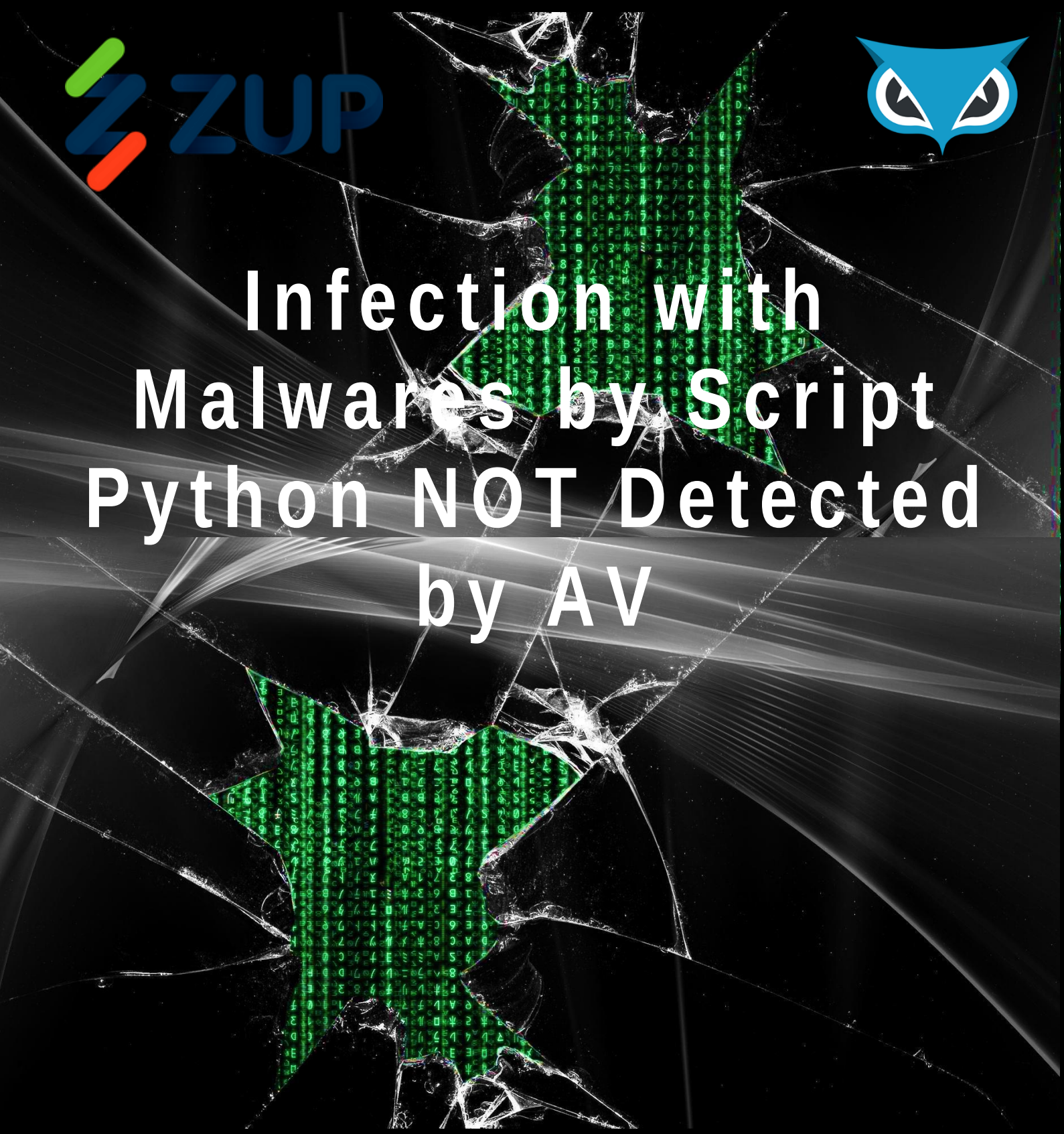


Infection with Malwares by Script Python NOT Detected by AV

ZUP Security Labs at Zup Innovation

Researcher & CyberSecurity Manager: Filipi Pires

1 Introduction

The purpose of this document, it was to execute several efficiency and detection tests in our endpoint solution, provided by Cybereason, this document brings the result of the defensive security analysis with an offensive mindset performing an execution of two python scripts responsible to download some malware in our environment.

Regarding the test performed, the first objective it was to simulate targeted attacks using a python script to obtain a panoramic view of the resilience presented by the solution, with regard to the efficiency in its detection by signatures, NGAV and Machine Learning, running this script, the idea is downloading these artifacts directly on the victim's machine. The second objective consisted in running this script another python script with daily malwares, provide by ***MalwaresBazaar*** by request using API access, in the day of this test we downloaded more than 200 real Malwares (206 Malware exactly).

With the final product, the front responsible for the product will have an instrument capable of guiding a process of mitigation and / or correction, as well as optimized improvement, based on the criticality of risks.

1.1 Scope

The efficiency and detection analysis had as target the Cybereason Endpoint Protection application (`Cybereason Cloud Console`) in **Version 20.1.261.0;**
Installed in the windows machine `Windows 10 Pro`;
Hostname - `Threat-Hunting-Win10`, as you can see in the picture below:

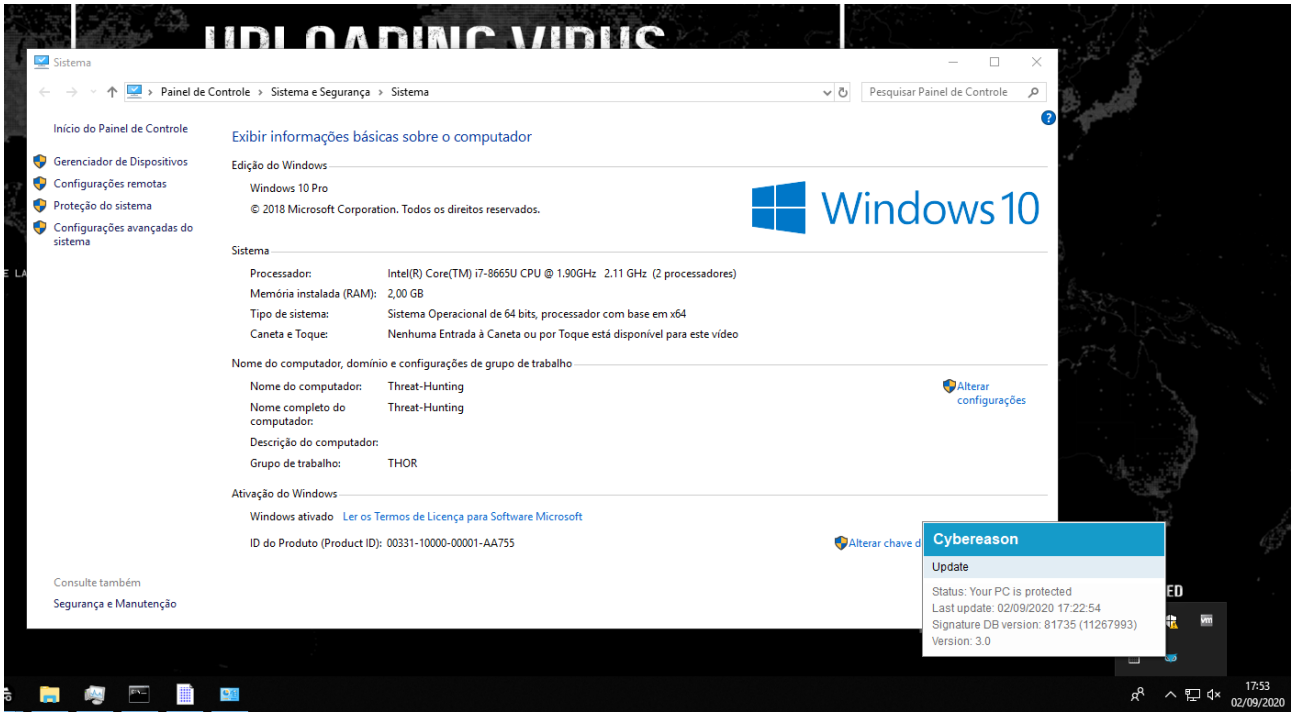


Image 1.1: Windows 10 Education 2019 Virtual Machine

1.2 Project Summary

The execution of the security analysis tests of the Threat Hunting team it was carried out through the performing an execution of two python scripts responsible to download some malware in our environment, in a virtualized environment in a controlled way, simulating a real environment, together with their respective best practices of the security policies applied, the test occurred during **4 day**, without count the weekend, along with the making of this document. The intrusion test started on the **10th of September** of the year 2020 and it was completed on the **14th of September** of the same year.

2 Running the Tests

2.1 Description

A virtual machine with Windows 10 operating system it was deployed to perform the appropriate tests, as well as the creation of a security policy on the management platform (`ZUP - Threat Hunting - Policy`) e and applied to due device.

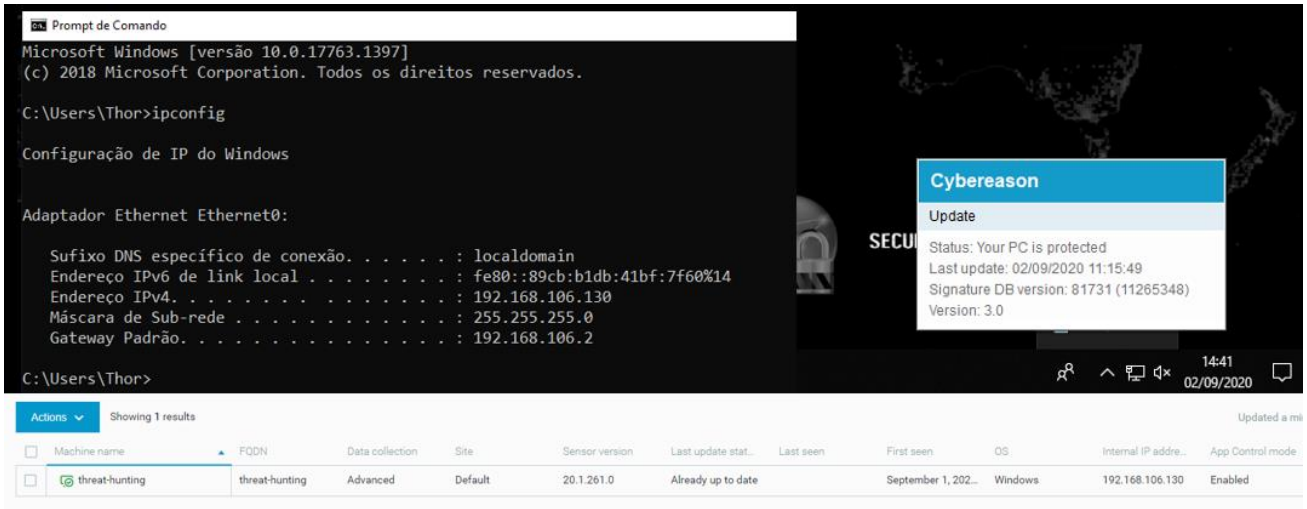


Image 1.2: Virtual Machine with Policy applied

The policy created was named **`ZUP - Threat Hunting`**, following the best practices recommended by the manufacturer, and, for testing purposes, all due actions were based on an aggressive detection method.

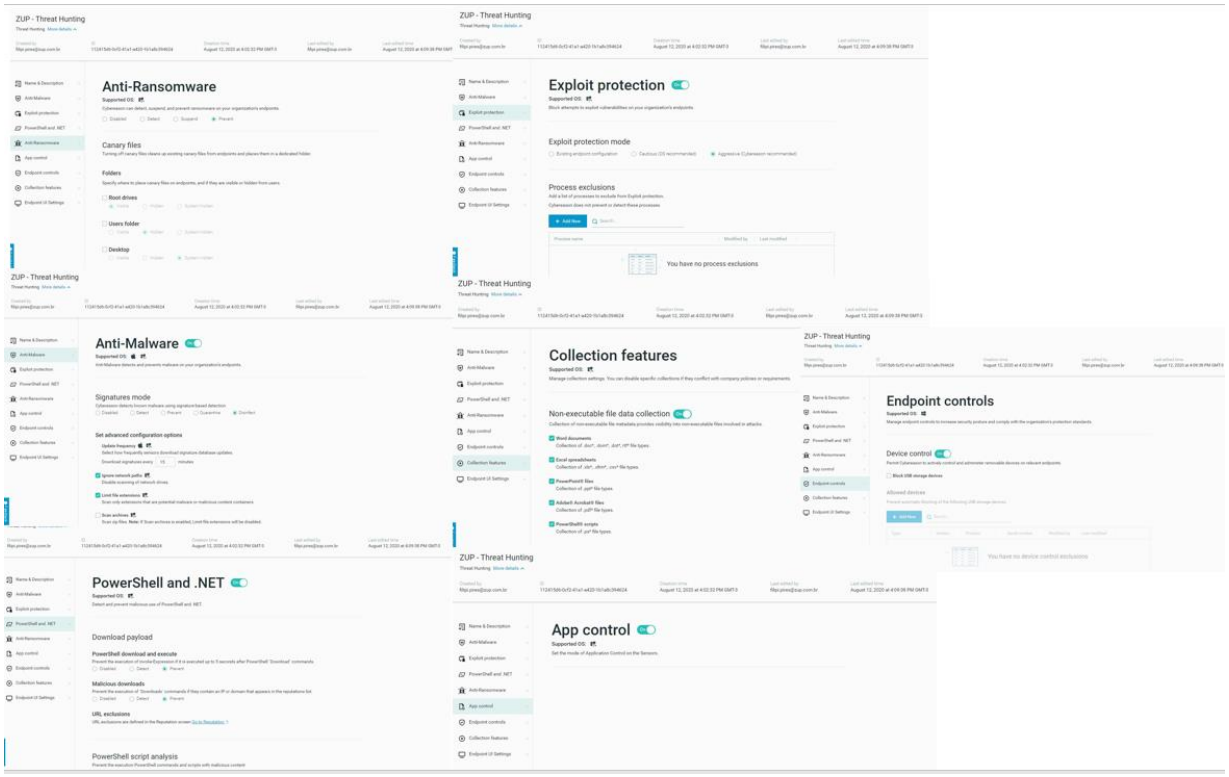


Image 1.3: Policy created by Cybereason Manager

2.2 First Test

Before starting the detection tests, we need to validate if all those scripts are functional.

Bazzar.py (based in MalwareBazzar Documentation)

(https://bazaar.abuse.ch/api/#download)

```
#!/usr/bin/env python3
import requests
import sys
import argparse
import json
import pyzipper

def check_sha256(s):
    if s == "":
        return
    if len(s) != 64:
        raise argparse.ArgumentTypeError("Please use sha256 value instead of '" + s + "'")
    return str(s)

parser = argparse.ArgumentParser(description='Download a malware sample from Malware Bazaa
r by abuse.ch')
parser.add_argument('-s', '--
hash', help='File hash (sha256) to download', metavar="HASH", required=True, type=check_sh
a256)
```

```
parser.add_argument('-u', '--
unzip', help='Unzip the downloaded file', required=False, default=False, action='store_tru
e')
parser.add_argument('-i', '--
info', help='Get information on a hash (do not download file)', required=False, default=Fa
lse, action='store_true')

args = parser.parse_args()

if(args.unzip == True and args.info == True):
    print("Sorry, please select unzip or information display.")
    sys.exit(1)

ZIP_PASSWORD = b'infected'
#ZIP_PASSWORD = "infected"
headers = { 'HERE API provided by MalwareBazaar': '' }

if(args.info == False):
    data = {
        'query': 'get_file',
        'sha256_hash': args.hash,
    }

    response = requests.post('https://mb-
api.abuse.ch/api/v1/', data=data, timeout=15, headers=headers, allow_redirects=True)
    open(args.hash+'.zip', 'wb').write(response.content)

    if(args.unzip == True):
        with pyzipper.AESZipFile(args.hash+".zip") as zf:
            zf.pwd = ZIP_PASSWORD
            my_secrets = zf.extractall(".")

            print("Sample \""+args.hash+"\" downloaded and unpacked.")
    else:
        print("Sample \""+args.hash+"\" downloaded.")
else:
    data = {
        'query': 'get_info',
        'hash': args.hash,
    }
    print(data)
    response = requests.post('https://mb-
api.abuse.ch/api/v1/', data=data, timeout=15, headers=headers)
    print(response.content.decode("utf-8", "ignore"))
```

After to perform this script as we can see below, the hash chosen it was downloaded and extracted in virtual machine.

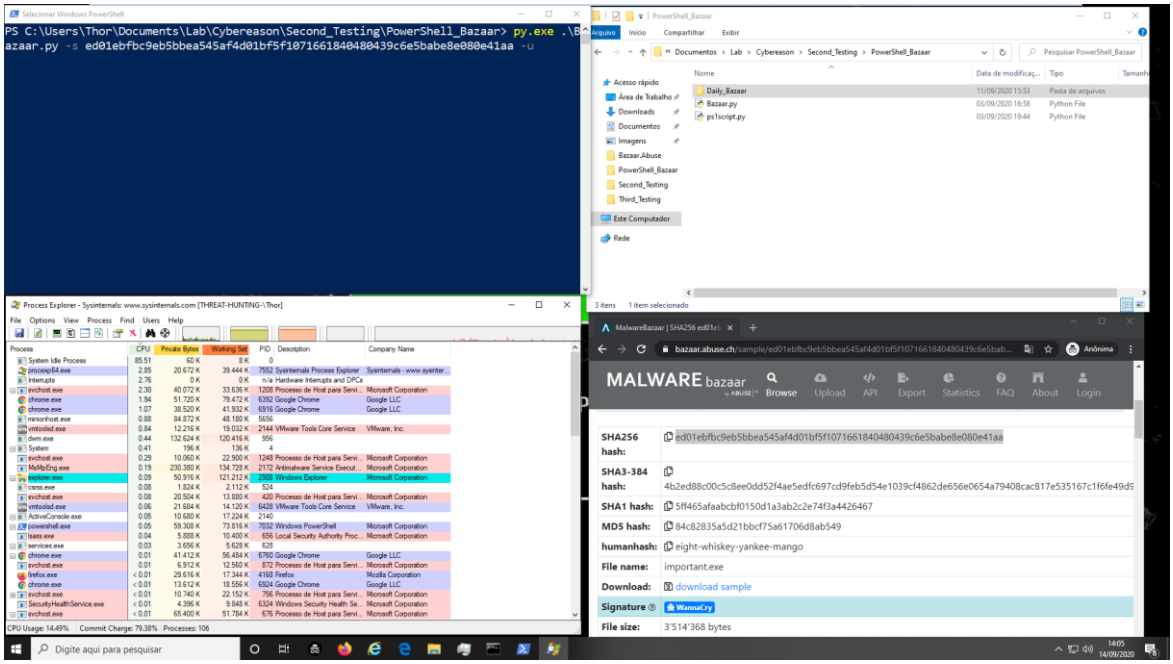


Image 1.4: Python Script execution without policy

As we expected, the file is downloaded from the Malware Bazaar using python script utilizing an API called and then its extracted and executed inside the machine.

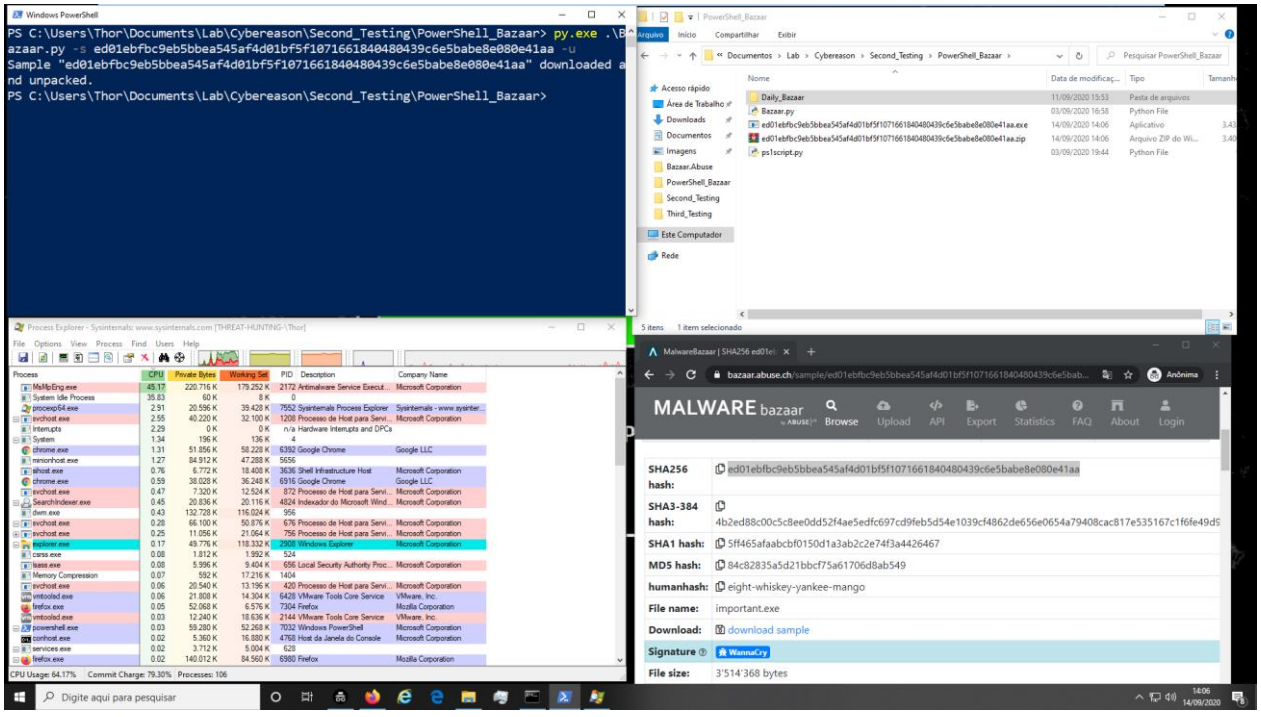


Image 1.5: Python Script works well

After executing the file

"ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa"

manually, we can confirm that the machine has actually been infected with malware of the type WannaCry.

So now, we going to execute the first stage of the tests where we perform a python script with malicious hash provide by ***MalwaresBazaar***, but know we applied all the policy in aggressive mode

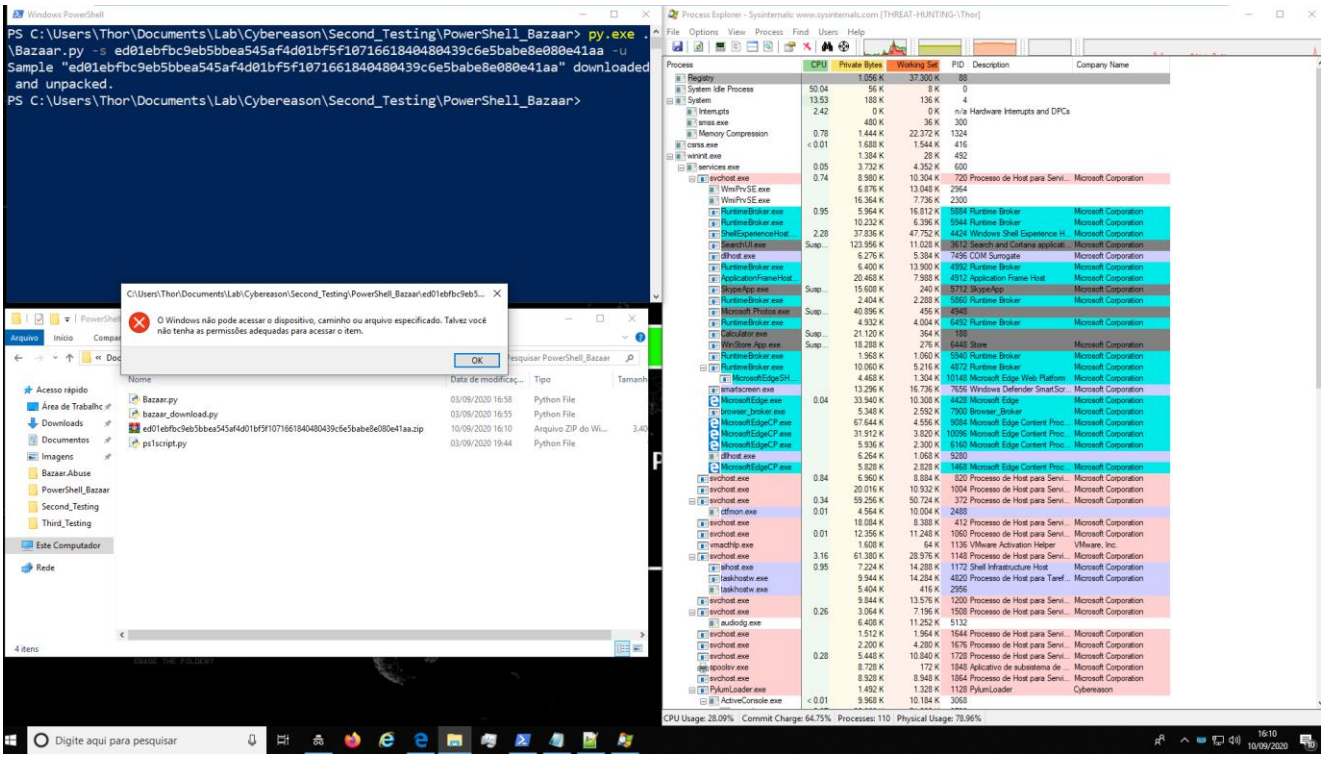


Image 1.6: Python Script with Cybereason Policy applied

As you can see, in this first test, performing **WannaCry Hash**, the sample it was block by Cybereason platform.

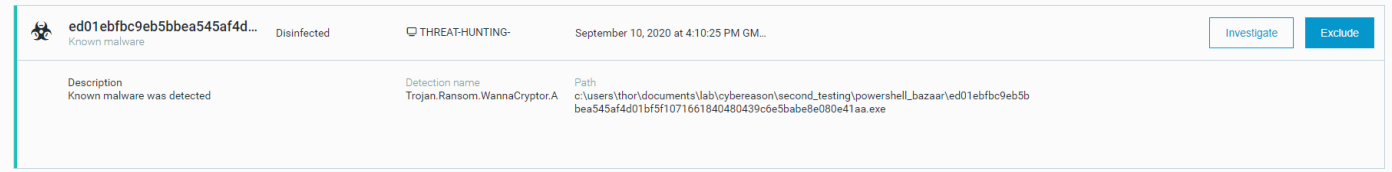


Image 1.7: Cybereason Block

2.3 Second Test

The second test we used another **python script**, responsible to download the daily Malwares collected by Malware Bazaar as you can see below

```
#!/usr/bin/env python3
from datetime import date, timedelta
import urllib.request
import sys
import pyzipper

ZIP_PASSWORD = b'infected'
```

```
print(sys.argv)
if len(sys.argv) == 2:
    datefile = sys.argv[1]
else:
    datefile = (date.today() - timedelta(days=1)).strftime("%Y-%m-%d")

print("Using date: %s" % datefile)
print("Downloading https://mb-api.abuse.ch/downloads/%s.zip dataset..." % datefile)
response = urllib.request.urlopen('https://mb-api.abuse.ch/downloads/%s.zip' % datefile)
print("Download complete!")
open('%s.zip' % datefile, 'wb').write(response.read())
print("Saving dataset... complete!")

with pyzipper.AESZipFile("%s.zip" % datefile) as zf:
    zf.pwd = ZIP_PASSWORD
    my_secrets = zf.extractall(".")
    print("Dataset unpacked!")
```

All this files that it was uploaded from public repository known and maintained by the security community in this web (https://bazaar.abuse.ch/).

> *MalwareBazaar is a project from abuse.ch with the goal of sharing malware samples with the infosec community, AV vendors and threat intelligence providers.*

MalwareBazaar creates daily batches of malware sample). The daily batches are created once a day at midnight (00:00 UTC). Please consider that it takes a few minutes to create the batch. So, I kindly ask you to not fetch the daily batch before 00:15 UTC.

The day chosen for this test it was **2020-09-10**

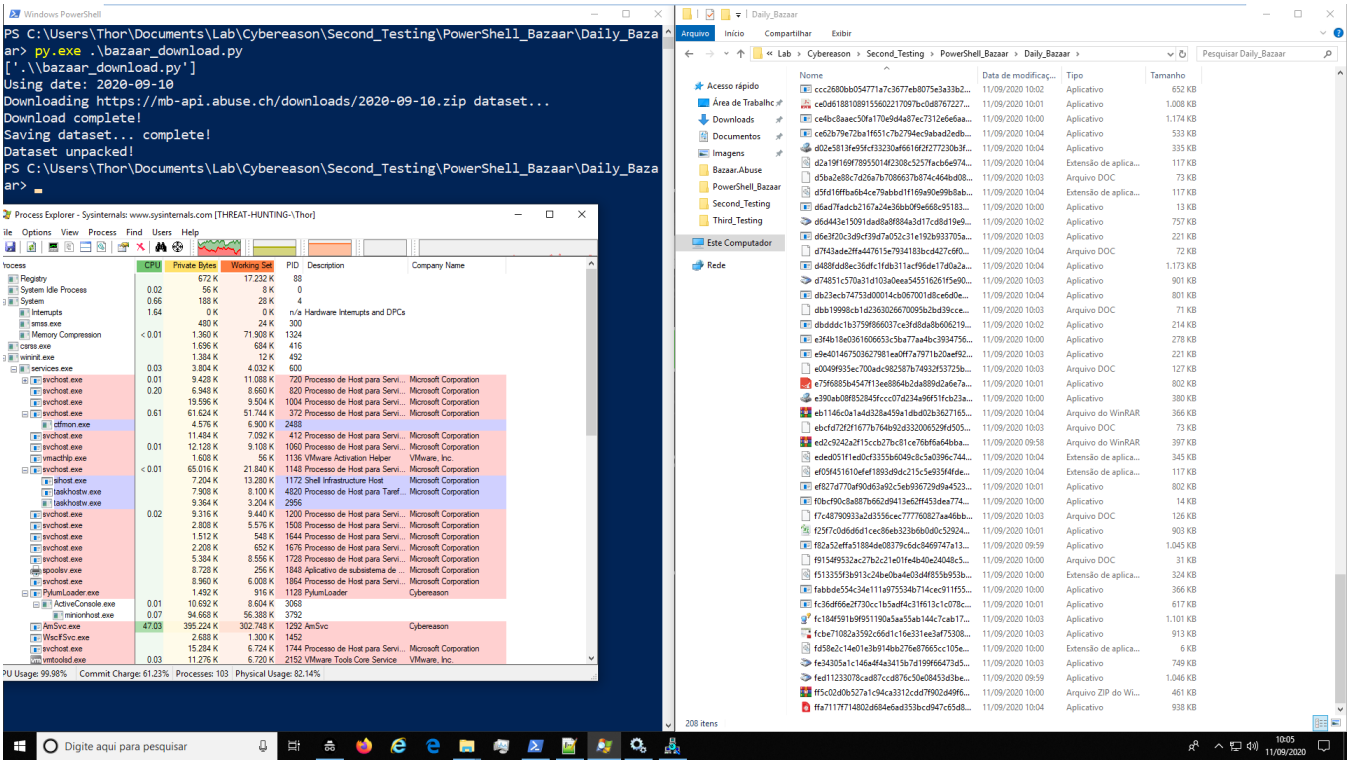


Image 1.8: Python Script works well

The purpose of this test, it was to simulate the same process when the user receives any email with a python script and after that he could even click in this script and he'll be downloading a zipped file (`.zip`) and will perform the extraction of these artifacts in their own environment.

During this test, one thing called my attention:

- **First Detection** happened on **September 11, 2020 at 9:58:55 AM GMT-3**
- **Last Detection** happened on **September 11, 2020 at 3:53:50 PM GMT-3**

That is, we have a time gap with almost **four hours** between the first and the last detection, that was the time it took for malwares to be detected.

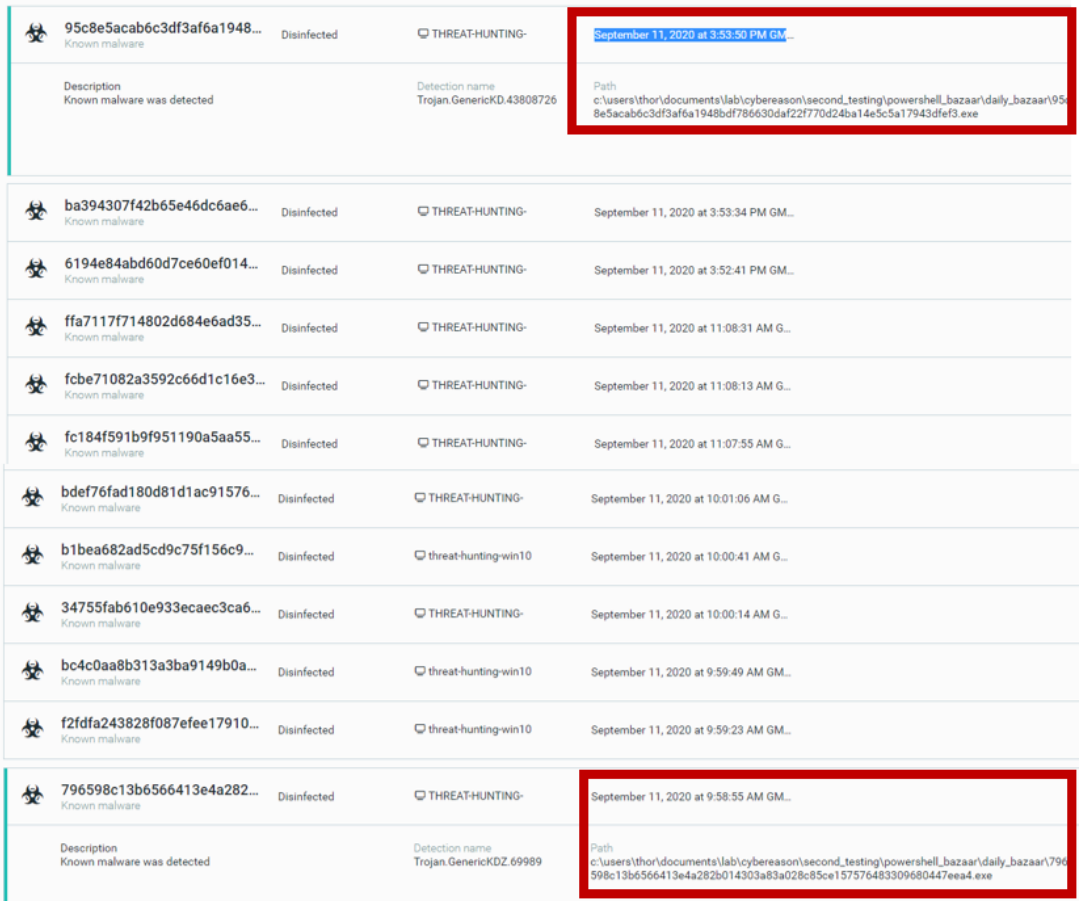


Image 1.5: Malware Alerts Detection by Cybereason

After performing the action of extracting the files, it was possible to verify in the *Cybereason* "***Malware Alerts***" logs that many malwares were detected, however it was possible to verify that there are currently **47 (forty-five) Malwares** that, when executed inside the environment, could perform an infection.

When we look in some files, with some researches we can find information about them, below **.z** file.

`5ed7822936ff5e5517eae4f6fd60c7d1c629f5010fd55d04a67ae773d5837983.z`

In Virus Total reference we can see that those files are known like **Malicious file**, by many different antivirus engines.

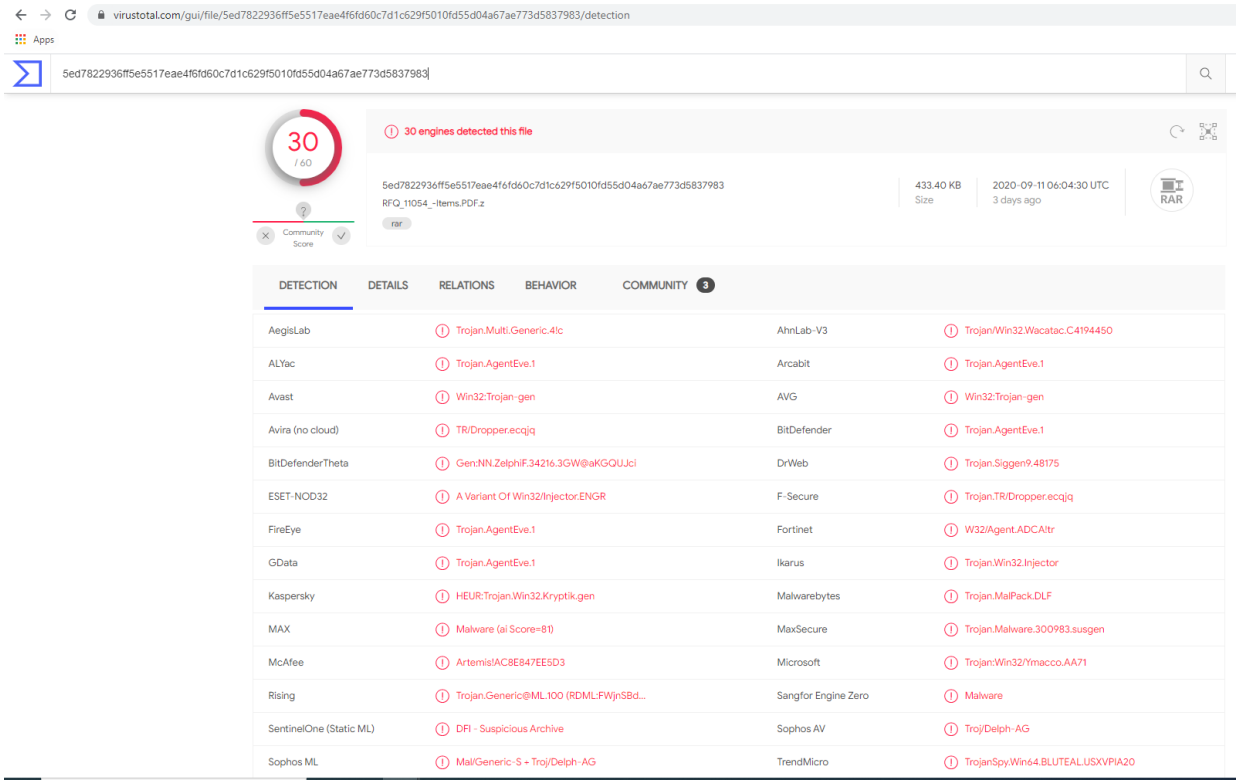


Image 1.8: VirusTotal Detection

```
Basic Properties
MD5    ac8e847ee5d389f6cd3a43011413514c
SHA-1  c44570fd660105ce3294010d3dc42c55eb5d2961
SHA-256      5ed7822936ff5e5517eae4f6fd60c7d1c629f5010fd55d04a67ae773d5837983
SSDEEP 12288:JzzyzBLNVy7N6z2OygOHE9NzrV0dhNRINL+:wLNMN6z2Oyg6EN0dnRi+

Names
RFQ_11054_-Items.PDF.z
RFQ 11054 -Items.PDF.z
```

Other example below with **.rar** file :

`026b46ce13b69a6e48fea6e47360d4680d61dbd075a10c52f4d48591af113bb1`

In Virus Total reference we can see that those files are known like **Malicious file**, more than 30 different antivirus engines.

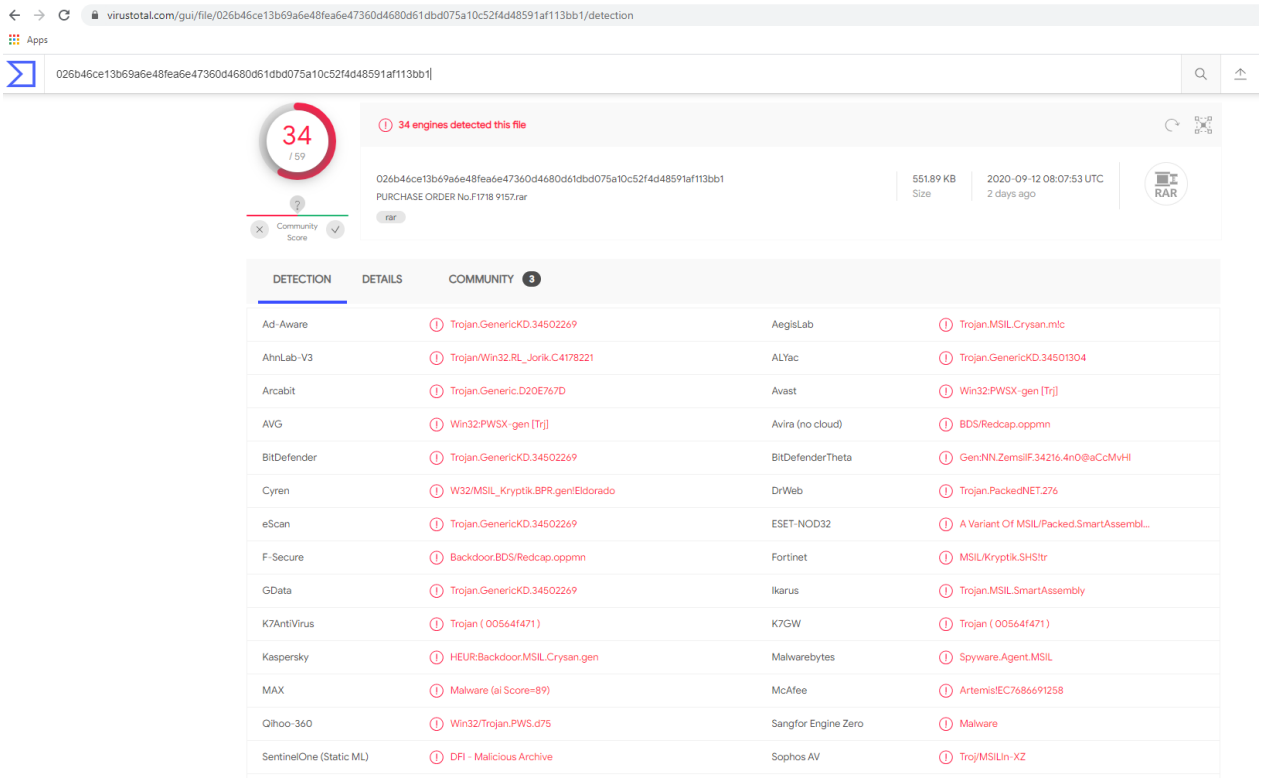


Image 1.9: VirusTotal Detection

```
Basic Properties
MD5    ec76866912584f5abde7f6c6bcc6b281
SHA-1  49cd4ca3556dbdfbe5c506a4276f1d66dd3e0887
SHA-256      026b46ce13b69a6e48fea6e47360d4680d61dbd075a10c52f4d48591af113bb1
SSDEEP 12288:PUWyh1KB90dB07xkqF1L6Smza9rNxfeQZTaugNQARH:PUWyhQ90dBkL6SmONB7FautqH

Names
PURCHASE ORDER No.F1718 9157.rar
ec76866912584f5abde7f6c6bcc6b281.file
```

Another example below with **.gz** file:

```
63affc23e20b0fdfeeb5a85cc35a8c5613de7bc4a0b0a7647a1cfff26b8f89c7
```

In Virus Total reference we can see that those files are known like **Malicious file**, more than 30 different antivirus engines.

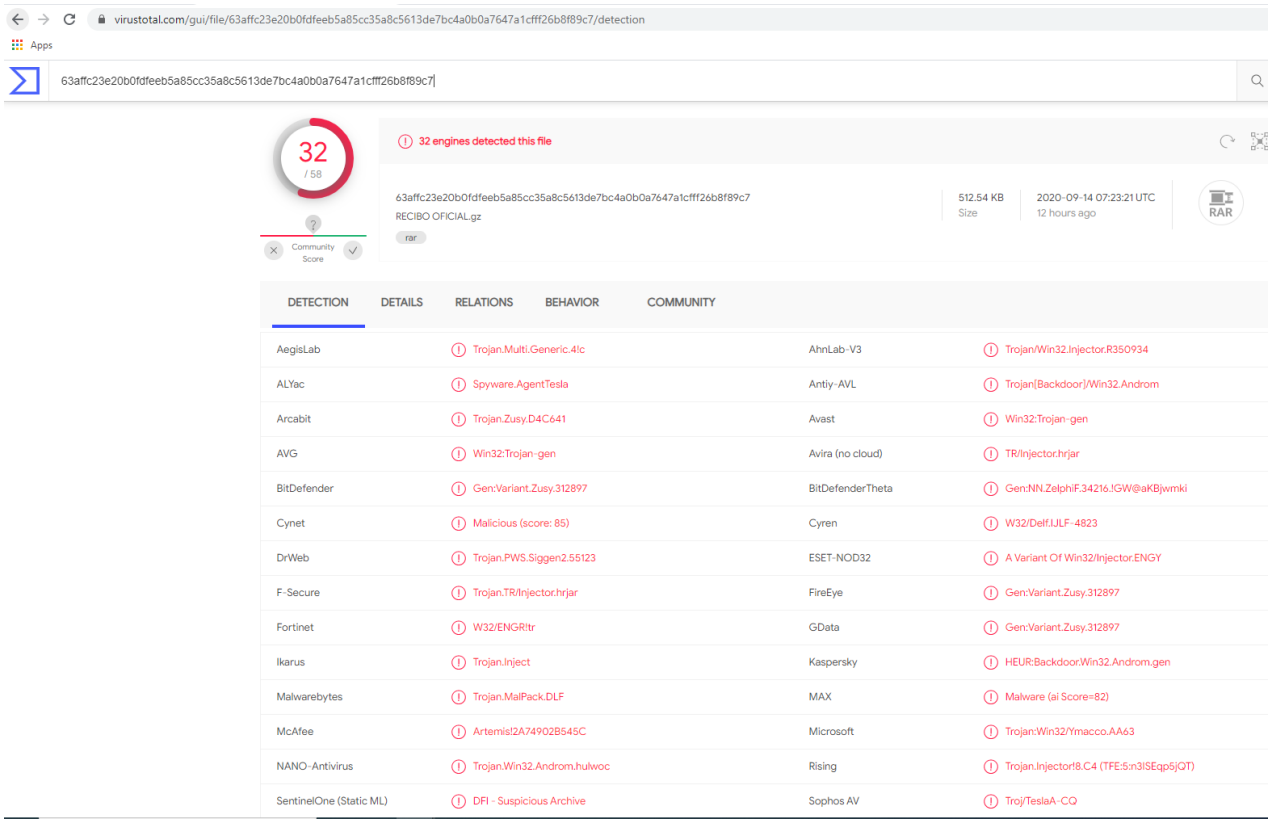


Image 1.10: VirusTotal Detection

```
Basic Properties
MD5    2a74902b545c562adeaa3bbed5792222
SHA-1  f49274ee50dfb6cc86d087e6879606385248be38
SHA-256      63affc23e20b0fdfeeb5a85cc35a8c5613de7bc4a0b0a7647a1cfff26b8f89c7
SSDEEP 12288:m0z4En01NGwGpmPKtZbBRBDTZ+qijIjQ1m:m0e3SpcKL1RHsVm

Names
RECIBO OFICIAL.gz
```

Another example below with **ELF** file:

```
96ead4fa8bf37eb8933285466b0f3985ab55438702000f678fac150ab3ea9703
```

In Virus Total reference we can see that those files are known like **Malicious file**, more than 25 different antivirus engines.

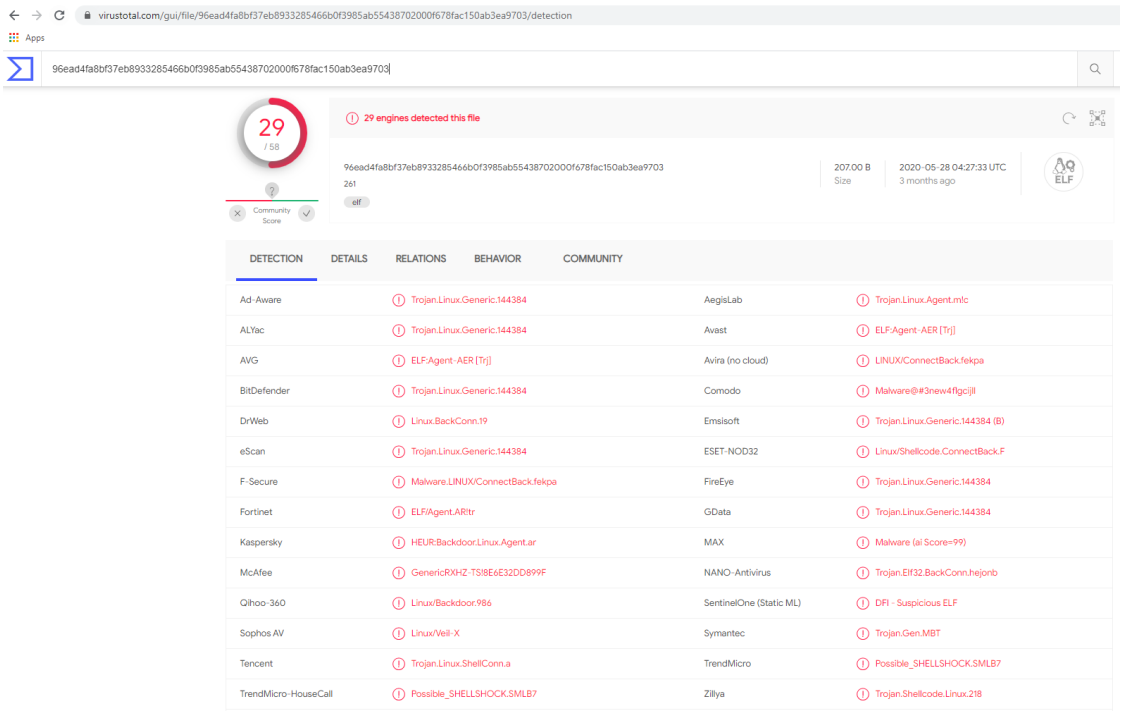


Image 1.11: VirusTotal Detection

```
Basic Properties
MD5   8e6e32dd899f09e83df531ec9f54ae8b
SHA-1 1d0e4c5ba903194d3cdeb08af6afde5f9c5f5e8b
SHA-256      96ead4fa8bf37eb8933285466b0f3985ab55438702000f678fac150ab3ea9703
Vhash 2710bdcf7969ecc632f3f0b2c321e711
SSDEEP
      3:Bkkk/tMlwXll/O/slrCs4X1lFrSwfijvRM8IPNioOHyUvwGcV5QfE2:Btk/tMl//E2s4uzRKQXSEwhV5Qf
E2

File type    ELF
Magic ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked,
corrupted section header size

Names
261
272
```

Another example below with **.Dll** file:

`6b2eb4a0767e551244e0d8e253550332dad808025ebfae65a31ef4376df94deb`

In Virus Total reference we can see that those files are known like **Malicious file**, more than 3 0different antivirus engines.

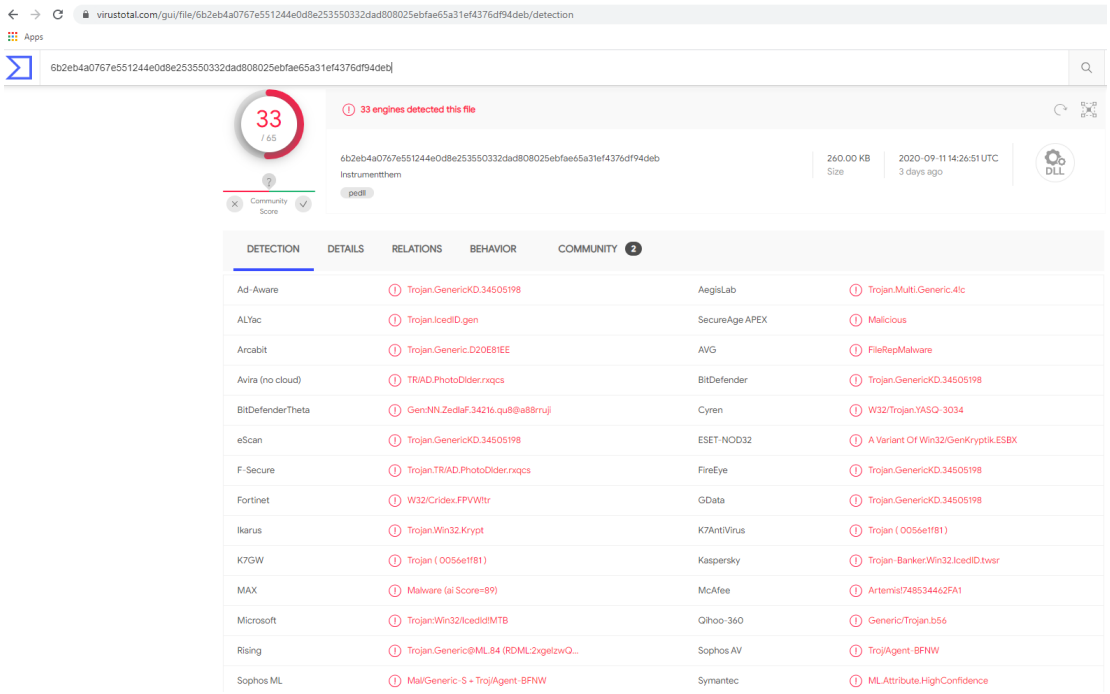


Image 1.12: VirusTotal Detection

```
Basic Properties
MD5    748534462fa1cdc4903739d077ab8364
SHA-1  6f4279bd80894b1057fcbe6cca2fe38a788638c1
SHA-256      6b2eb4a0767e551244e0d8e253550332dad808025ebfae65a31ef4376df94deb
Vhash  125056655d15156az4fpz3dz
Authentihash dc61067e43565af267d8b67159da30e724efa0c4f61eea7b5377fad3ac71ef4d
Imphash      b01dbc6a1382714766d899fb941df022
SSDEEP 6144:Iu+CL4X3w+d95Yw8v7wVc0HyKoUzAOvSUzNd:YCL4/96w8v7MUUztDNd
File type    Win32 DLL
Magic  PE32 executable for MS Windows (DLL) (GUI) Intel 80386 32-bit

Names
Instrumentthem
death.dll
```

Another example below with **.exe** file:

`d02e5813fe95fcf33230af6616f2f277230b3f9f636420c9c0a979540922c6bb`

In Virus Total reference we can see that those files are known like **Malicious file**, more than 3 0different antivirus engines.

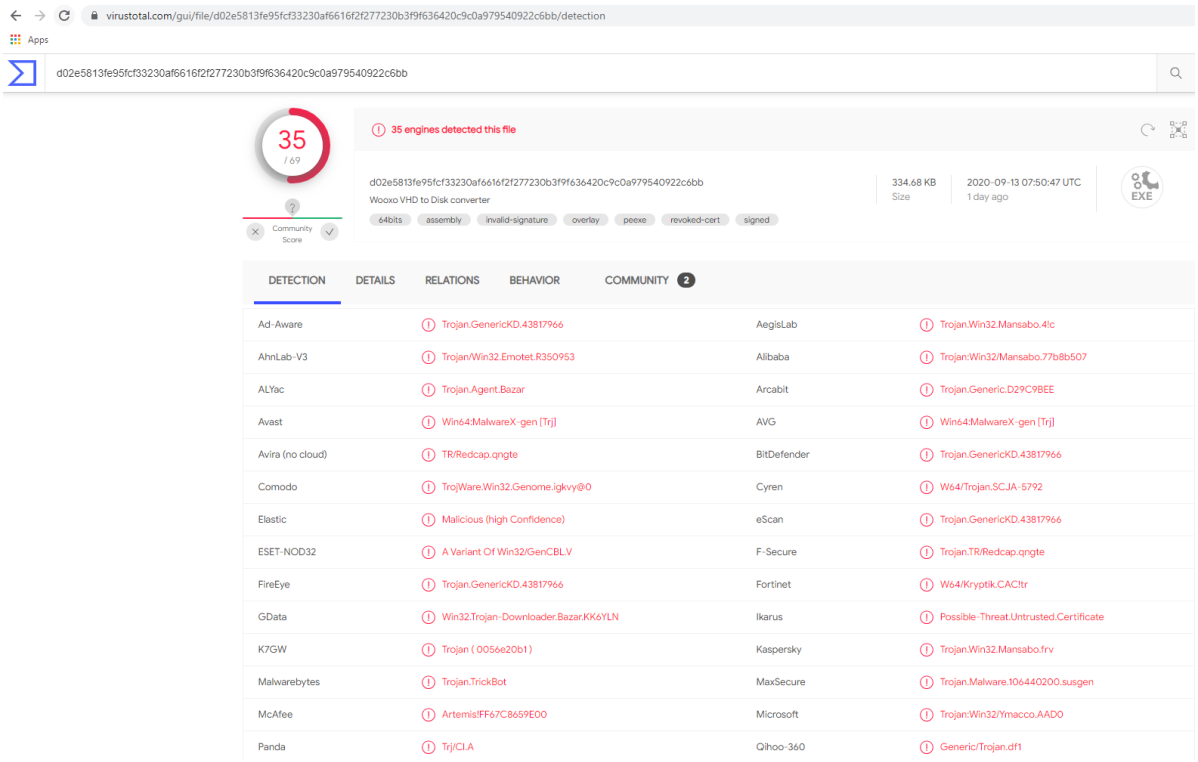


Image 1.13: VirusTotal Detection

```
Basic Properties
MD5    ff67c8659e00bdff07d216c45edf5498
SHA-1  d213782ba7746676a9322733b31e3505600e68ec
SHA-256      d02e5813fe95fcf33230af6616f2f277230b3f9f636420c9c0a979540922c6bb
Vhash  035056551d151573z11z70022hz12z29az23001
Authentihash aad8efe1348f7fc9382e4d7b50929c419fa7f8bbc42be5a2f0490aa1db99d651
Imphash      4e87a8ef84d8e7fc292278e3888dc5b9
SSDEEP 6144:DUuDJZ2+YfLLTw55iZTlTYIpJzHHq2KxPybHSLvNo+:DUuDJZpYEinTRDHCxabyLvt
File type    Win32 EXE
Magic  PE32+ executable for MS Windows (GUI) Mono/.Net assembly

Names
Vhd2disk
Print.exe
tmp.download
```

Other perspective is to try find any information in another kind of platforms, or others databases and to try find any reputation for this domain, I made an testing in **JoySandbox** platform that detects and analyzes potential malicious files and URLs on Windows, Android, Mac OS, Linux, and iOS for suspicious activities. It performs deep malware analysis and generates comprehensive and detailed analysis reports, as you can see in the *prescreen* below provide by **JoySandbox** Platform.

The result of this test, brought to us that this file **(*.exe* as already mention in VirusTotal) is Malicious**

`d02e5813fe95fcf33230af6616f2f277230b3f9f636420c9c0a979540922c6bb`

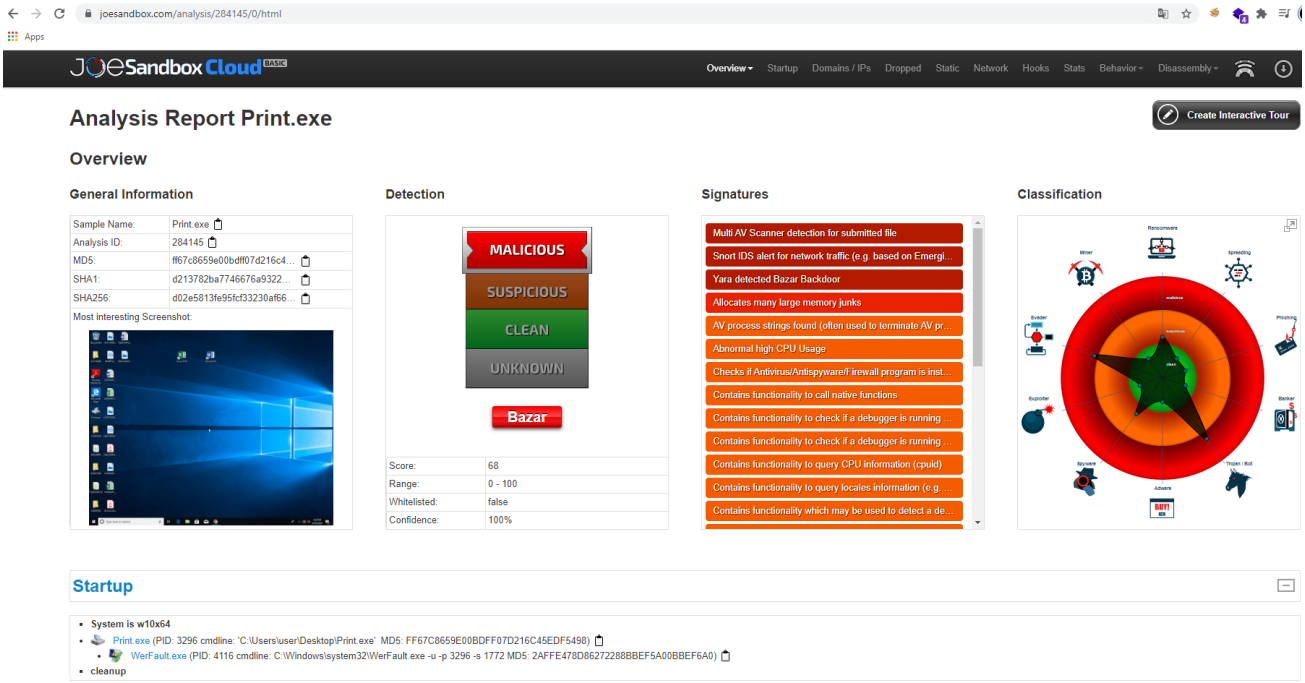


Image 1.14: JoySandbox Detection

Another test realized in JoySandbox show us as we can see in the *prescreen* below provided by **JoySandbox** Platform.

The result of this test, brought to us that this file **(*.dll* as already mention in VirusTotal) is Malicious**

`d02e5813fe95fcf33230af6616f2f277230b3f9f636420c9c0a979540922c6bb`

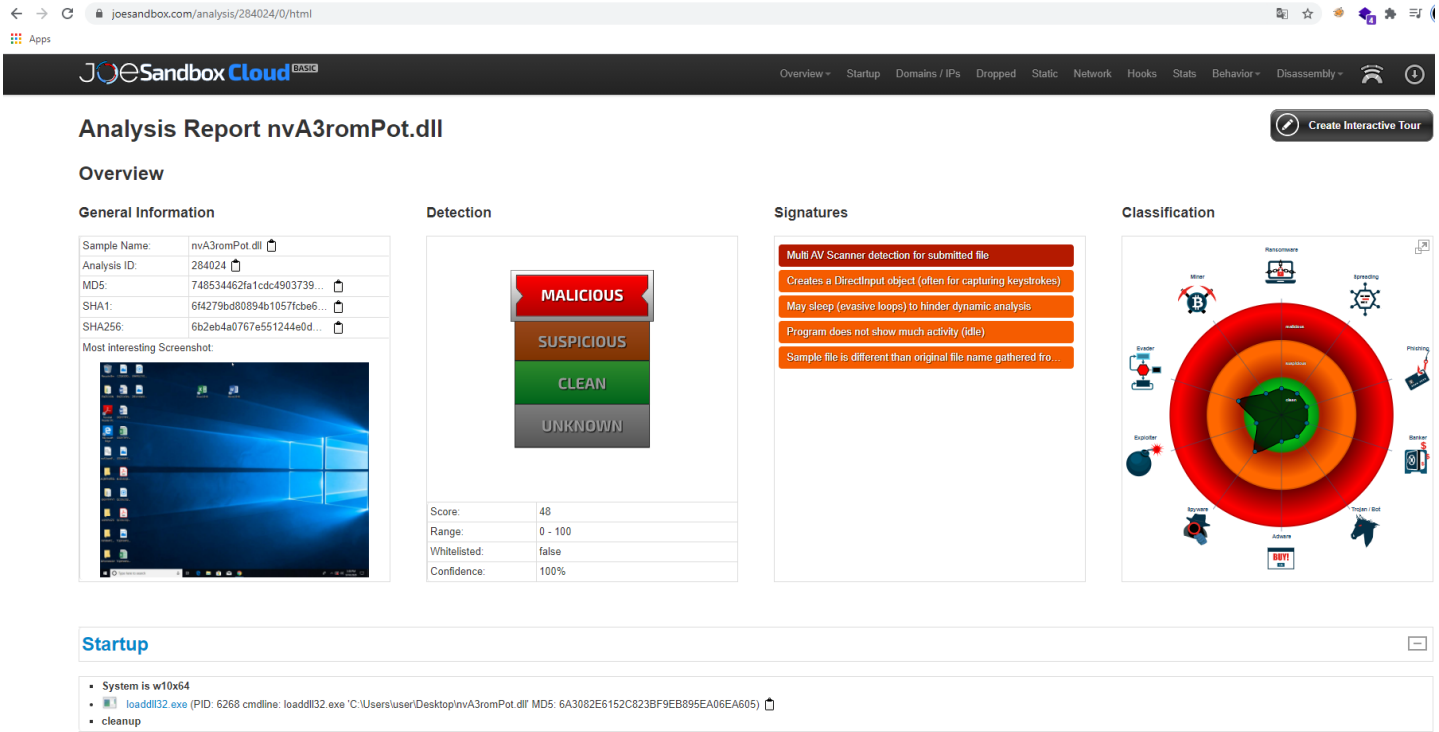


Image 1.15: JoySandbox Detection

Other perspective is to try find any information in another kind of platforms, or others databases and to try find any reputation for this domain, I made an testing in **Maltiverse** platform, It is born as a service oriented to get used by cybersecurity analysts to research on indicators of compromise (IOCs), as you can see in the *prescreen* below provide by **Maltiverse** Platform.

The result of this test, brought to us that this file **(*.dll* as already mention in VirusTotal) is Malicious**

`f513355f3b913c24be0ba4e03d4f855b953bb8a616368c5a217d33988742f871`

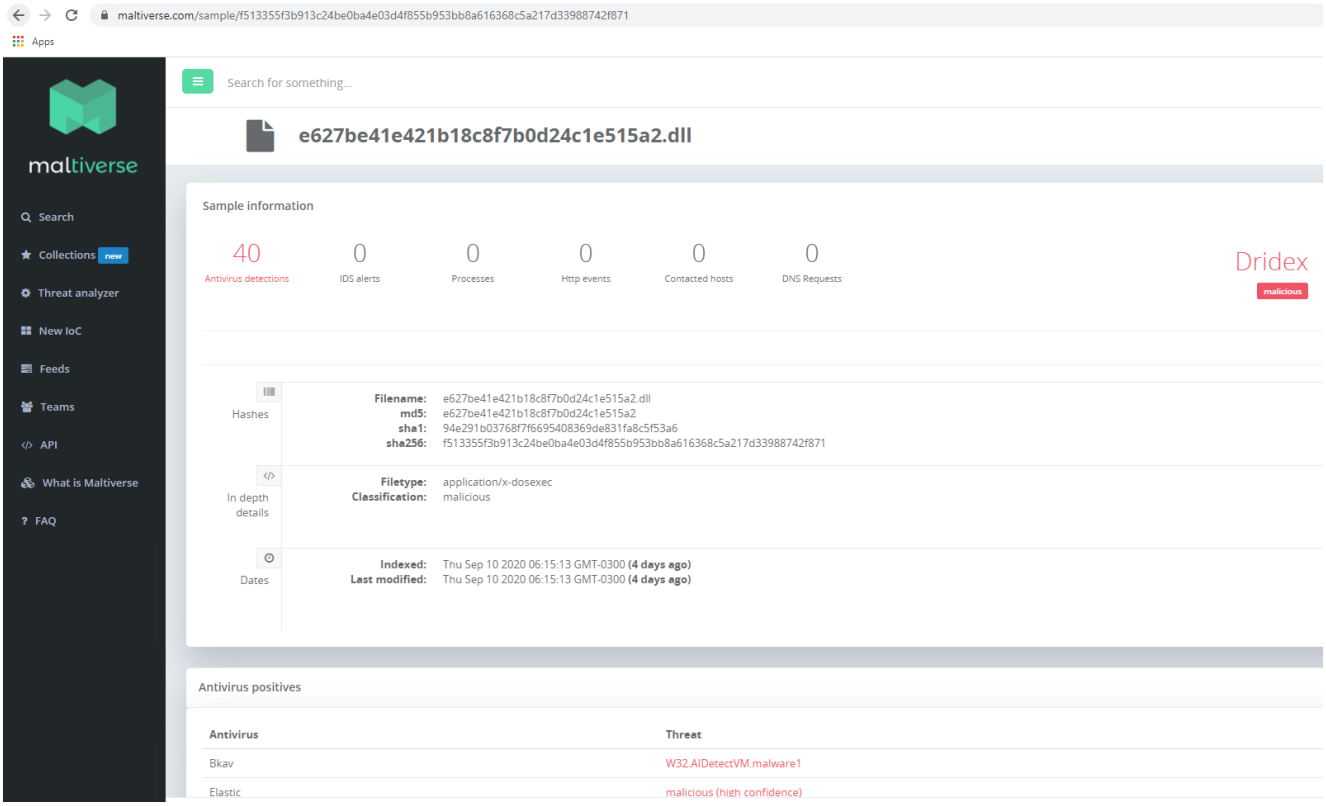


Image 1.9: Maltiverse Detection

All those files you can find and download from Malware Bazaar in the *url* below.

hxxps://mb-api.abuse.ch/downloads/2020-09-10.zip

So, the question here is, how is it works? Detection by pattern? Signature? NGAV? ML?

3 Impact

At the end of this test, it was possible to verify that there were **47 forty-seven malwares** that, when executed inside the environment, may perform an infection.

- **Problem during the first test - unzipping ZIP file (detection time)**
  - During this test it was possible to see that the Cybereason Endpoint Solution took almost 4 hours to realize all detections in our environment test, that is, if the attack happened in the same time in the victim, this user could click in anyone of the samples and could be infected, because it's not clear how works the prevalence, maybe priority of the engine in the detection flow.
- **Malicious .RAR, .7z,.Zip in others compress files NOT Detected**
  - As we can see in many samples (.RAR, .7z, .Zip, etc) it's not detected like a Malicious, we used many different sources to prove that is sample it's malicious.
- **Malicious Files (.RAR, .7z,.Zip) without necessity of password can be executed NOT Detected**
  - As we can see in many samples (.RAR, .7z, .Zip, etc) without password, that is, anyone can extract those files and execute the same in our environment test, it was not detected like a Malicious
- **Malicious EXE files Not Detected**
  - PE files not detected even though malicious; it was not detected.
- **Malicious DLL files Not Detected**
  - *DLL* files not detected even though malicious; it was not detected.
- **Malicious ELF files Not Detected**
  - *ELF* file not detected even though malicious; In our test environment, wouldn't be dangerous, because our environment it was Windows, but should be block but it was not detected.

4 Recommendatios Actions

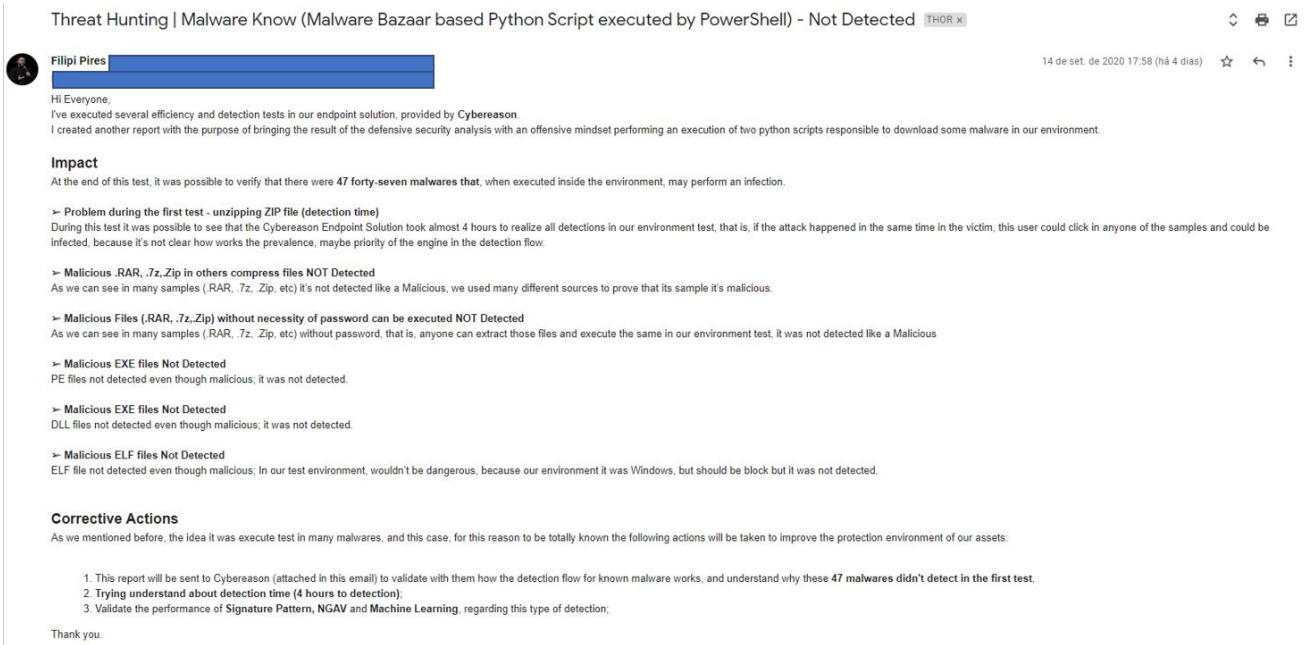
As we mentioned before, the idea it was execute test in many malwares, and this case, for this reason to be totally known the following actions will be taken to improve the protection environment of our assets:

- This report it was to send to Cybereason to validate with them how the detection flow for known malware works, and understand why these 47 malwares didn't detect in the first test, and to try understand the detection time;
- It was requested a validation of the performance of NGAV and Machine Learning, regarding this type of detection, and to try understand (again) the flow detection, as well as the priority of the engines;
- The best practices of the configurations will be requested by Cybereason team;

5 Answers from Cybereason Company

As we mentioned before, the idea it was execute test in many malwares, and this case to bring the result of the defensive security analysis with an offensive mindset running scripts python with daily malwares, provide by ***MalwaresBazaar*** by request using API access, in the day of this test we downloaded more than 200 real Malwares (206 Malware exactly) in our environment.

We opened a support case with the Cybereason support team on **September 14th** as you can see below



Threat Hunting | Malware Know (Malware Bazaar based Python Script executed by PowerShell) - Not Detected THOR x

Filipi Pires [redacted] 14 de set. de 2020 17:58 (há 4 dias)

Hi Everyone,
I've executed several efficiency and detection tests in our endpoint solution, provided by **Cybereason**.
I created another report with the purpose of bringing the result of the defensive security analysis with an offensive mindset performing an execution of two python scripts responsible to download some malware in our environment.

Impact
At the end of this test, it was possible to verify that there were **47 forty-seven malwares that**, when executed inside the environment, may perform an infection.

➢ **Problem during the first test - unzipping ZIP file (detection time)**
During this test it was possible to see that the Cybereason Endpoint Solution took almost 4 hours to realize all detections in our environment test, that is, if the attack happened in the same time in the victim, this user could click in anyone of the samples and could be infected, because it's not clear how works the prevalence, maybe priority of the engine in the detection flow.

➢ **Malicious .RAR, .7z,.Zip in others compress files NOT Detected**
As we can see in many samples (.RAR, .7z, .Zip, etc) it's not detected like a Malicious, we used many different sources to prove that its sample it's malicious.

➢ **Malicious Files (.RAR, .7z,.Zip) without necessity of password can be executed NOT Detected**
As we can see in many samples (.RAR, .7z, .Zip, etc) without password, that is, anyone can extract those files and execute the same in our environment test, it was not detected like a Malicious

➢ **Malicious EXE files Not Detected**
PE files not detected even though malicious; it was not detected.

➢ **Malicious EXE files Not Detected**
DLL files not detected even though malicious; it was not detected.

➢ **Malicious ELF files Not Detected**
ELF file not detected even though malicious; In our test environment, wouldn't be dangerous, because our environment it was Windows, but should be block but it was not detected.

Corrective Actions
As we mentioned before, the idea it was execute test in many malwares, and this case, for this reason to be totally known the following actions will be taken to improve the protection environment of our assets:

1. This report will be sent to Cybereason (attached in this email) to validate with them how the detection flow for known malware works, and understand why these **47 malwares didn't detect in the first test**,
2. **Trying understand about detection time (4 hours to detection)**;
3. Validate the performance of **Signature Pattern, NGAV** and **Machine Learning**, regarding this type of detection;

Thank you.

Image 1.10: Report to manufacturer

And this case is **Totally Critical**, the answer should happen in 3 hours as how it was aligned in some conversation with Customer Success Managers, Support Managers, Director Customers and VP from Cybereason but unfortunately, we didn't receive any answer to solve this problem with 47 undetected malware, even though they were known..

The only answer received it was:

Hi Filipi,

Thank you for the information. We are currently working with your CSM, @CustomerManager, to ensure that this information gets escalated to the Support/Detection teams who can provide better insight on malware detections.

Please let us know if you have any questions or concerns.