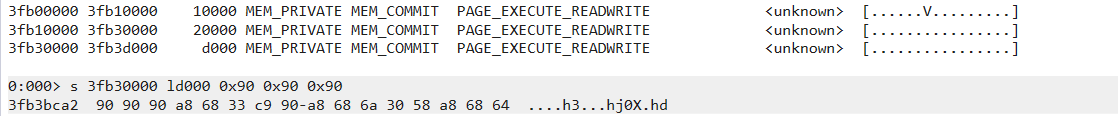


Breakpoint:

bp FoxitPDFReader + 0x0132ADE1

**!address -f:PAGE\_EXECUTE\_READWRITE**

Lối vào trong shellcode sẽ ở vị trí 0xBCA2 trong vùng nhớ có size 0xD000:



Dự đoán base tại: 162c0000

* Shellcode tại: 0x162c0000 + 0xbca2
* Eax đặt giá trị là: 0x162c0000 + 0xbc29

Viết JIT-SPRAY payload:

Đầu tiên, code sẽ tìm hàm Kernel32!GetProcAddressStub

[shellcode2asmjs/WinExec\_cmd.asm at master · rh0dev/shellcode2asmjs · GitHub](https://github.com/rh0dev/shellcode2asmjs/blob/master/asm_payloads/WinExec_cmd.asm)

Giờ ta sẽ viết một cái shellcode dùng để ghi một file .exe vào bộ nhớ của máy nạn nhân thông qua giao thức TCP:



Ý tưởng chèn shell code:

Ta đã có hàm GetProcAddress, giờ ta cần:

Tìm kiếm hàm GetModuleHandleW.

LoadLibraryA(“ucrtbase.dll”) rồi tìm kiếm các hàm: ucrtbase!fopen, ucrtbase!fwrite, ucrtbase!fclose, các hàm khác lần lượt tương tự.

Lưu trữ:

Lệnh mov register, dword ptr [register + value] cost 3 bytes với value < 0x80 nên:

Lấy ebp làm saved local:

[ebp] = GetProcAddress

[ebp + 0x4] = edi = LoadLibraryA(“kernel32.dll”)

[ebp + 0x8] = kernel32!LoadLibraryA

[ebp + 0xc] = kernel32!GetConsoleWindow

[ebp + 0x10] = LoadLibraryA(“ucrtbase.dll”)

[ebp + 0x14] = ucrtbase!fopen

[ebp + 0x18] = ucrtbase!fwrite

[ebp + 0x1c] = ucrtbase!fclose

[ebp + 0x20] = ucrtbase!system

[ebp + 0x24] = fptr

[ebp + 0x28] = ucrtbase! malloc

[ebp + 0x2c] = LoadLibraryA(“user32.dll”)

[ebp + 0x30] = user32!ShowWindow

[ebp + 0x34] = LoadLibraryA(“WS2\_32.dll”)

[ebp + 0x38] = WSAStartup

[ebp + 0x3c] = socket

[ebp + 0x40] = WSACleanup

[ebp + 0x44] = inet\_addr

[ebp + 0x48] = htons

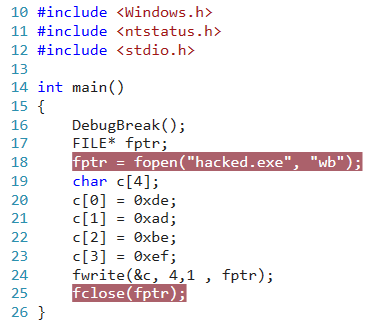
[ebp + 0x4c] = connect

[ebp + 0x50] = closesocket

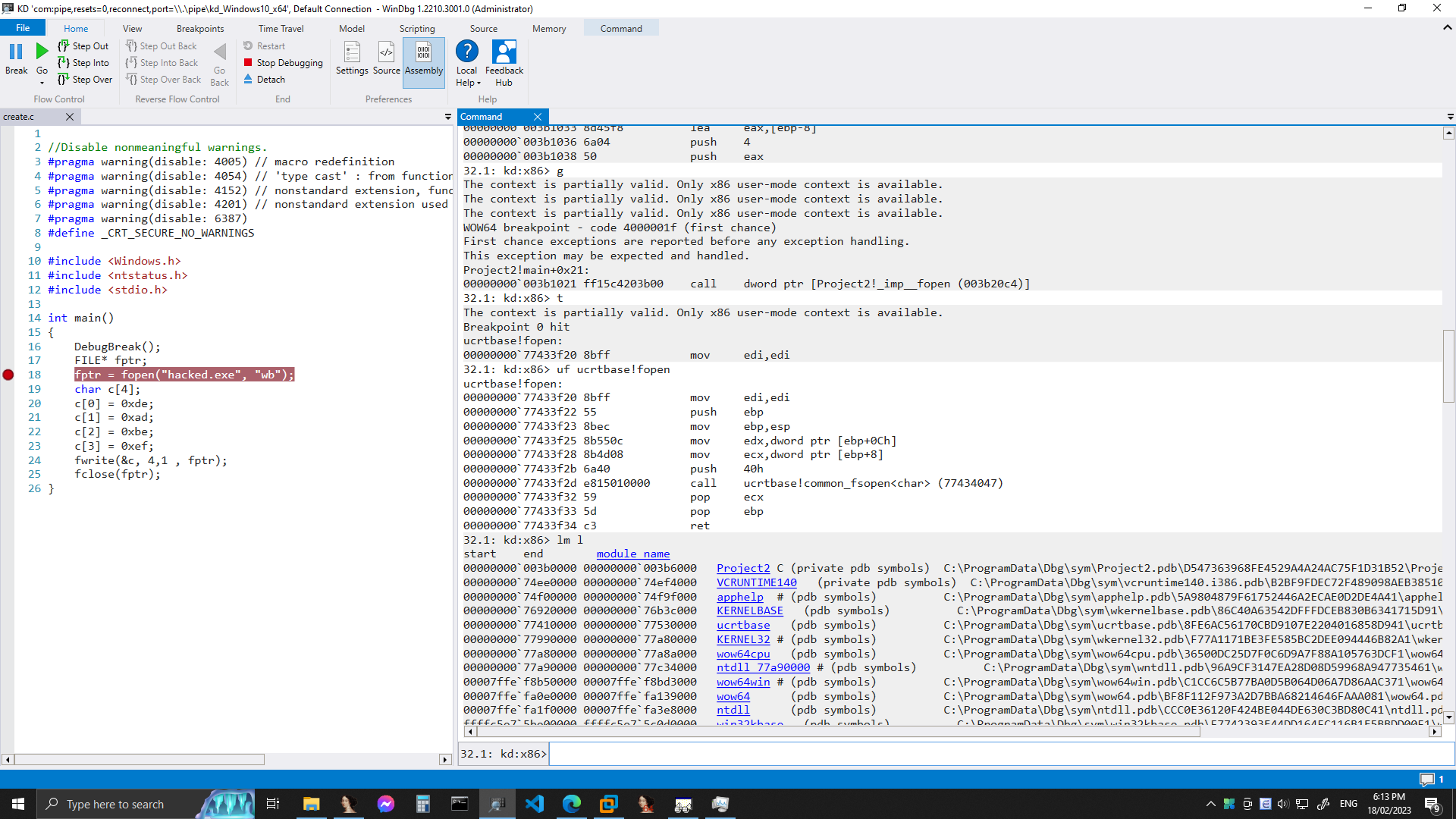
[ebp + 0x54] = recv

[ebp + 0x58] = (char\* )data (biến phục vụ cho hàm recv())

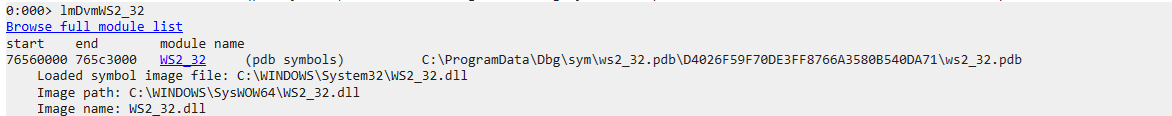
[ebp + 0x5c] = s (biến phục vụ cho hàm socket())



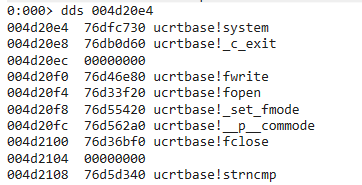
Sau đó system(“hacked.exe”) để chạy file tấn công.



Load trực tiếp WS2\_32.dll là ta có thể tìm được các hàm cho winsock:



Load ucrtbase.dll cho các hàm sau:



Và cả malloc()

Load kernel32.dll:



Load user32.dll:



Gốc rễ:

Breakpoint tại:

Nơi bị crash:

bp FoxitPDFReader + 0x01334610

Khởi tạo một heap size 0x64 khi gọi setFocus của field\_10:

bp FoxitPDFReader + 0x01329070

Hàm được gọi khi đóng document:

bp FoxitPDFReader + 0x0132B3A0

Hàm dùng để free heap size 0x64 khi hàm this.deletePages được gọi, sau khi free heap bằng hàm này, hàm FoxitPDFReader + 0x01334610 lại sử dụng nó dẫn đến việc bị crash:

bp FoxitPDFReader + 0x0133A900