

Forum:[Hack Forums](#)

Thread: [Offsets changing?](#)

Meterpreter - January 6, 2018, 3:34 pm

I'm trying to make a basic cheat for The Hunter CotW, problem is when I find the offset, find what writes to it, blah blah I am supposed to have the offset that doesn't change, but it does on restart...

From my research I believe it is code changing? So instead of it having a set address, it creates the offset based on the base address.

I added a GetModule function to my cheat to return the base address, just confused on how to use CE to get the offsets, and how to combine them with the base address. (Base address is not hex, plain int e.g "931917419").

Any help would be very much appreciated! Thanks guys

Just to add, I have found the address to be almost the same every time, as it ends in C820A0 but the first 4/5 num/alph characters change....

Base address currently is - 2019164160
Pointer is - 21074C41298+A0 or 21074C820A0

Pointer changes when I restart game. Meaning it must rely on theHunterCotW_F.exe+OFFSET
Pointer always ends in C820A0.

Nomad.Null - January 6, 2018, 10:19 pm

What you are experiencing is referred to as Address Space Layout Randomization (ASLR). It is a commonplace anti-exploit mechanism to make it very difficult to execute code in linear assembly by way of common exploit techniques such as buffer overflows.

This is something you will have to learn how to bypass on your own. Not easily done but very possible. Hacking: The Art of Exploitation is one such book that details how to perform these techniques.

Meterpreter - January 7, 2018, 12:52 am

(Jan 6, 2018 11:19 PM)True Demon Wrote: [\[CITE\]](#) What you are experiencing is referred to as Address Space Layout Randomization (ASLR). It is a commonplace anti-exploit mechanism to make it very difficult to execute code in linear assembly by way of common exploit techniques such as buffer overflows.

This is something you will have to learn how to bypass on your own. Not easily done but very possible. Hacking: The Art of Exploitation is one such book that details how to perform these techniques.

Can I not disable ASLR through registry? Or by running the program in Windows XP compatibility mode?

Also curious to know where the first 5 letter/numbers of the pointer come from because like I said the pointer always ends in C820A0 (*****C820A0).

If I could find how to predict those first few digits (or find them), I could simply add it on. :/

This disables ASLR for a particular program:

```
Code #include <Windows.h>
#include <ImageHlp.h>
#include <iostream>

using namespace std;

#pragma comment(lib, "ImageHlp.lib")

bool Flag(LPCSTR path, bool ASLR, bool DEP)
{
    LOADED_IMAGE PE;
    if (MapAndLoad(path, 0, &PE, 0, 0))
    {
        if (ASLR) // Enable address space layout randomization
            PE.FileHeader->OptionalHeader.DllCharacteristics =
IMAGE_DLLCHARACTERISTICS_DYNAMIC_BASE;
        else // Disable address space layout randomization
            PE.FileHeader->OptionalHeader.DllCharacteristics = NULL;
        UnMapAndLoad(&PE);
        return true;
    }
    return false;
}

int main()
{
    Flag("theHunterCotW_F.exe", false, false);
}
```

Has to be in the same directory. Works well, can't be bothered with this stupid game anyway.

movsx - January 7, 2018, 2:33 pm

You need to provide more information. Disabling ASLR on the executable isn't a good solution if you want to distribute the cheat or worry about anticheat detection.

What are you trying to change? Where is the pointer accessed? Show us the list of loaded modules, etc.

jammy47 - January 8, 2018, 1:08 am

getmodulehandle to get base address of process after injecting dll.

basically get a handle to the process, inject dll, make dll run code, get base address of process and add the offset.

Meterpreter - January 8, 2018, 3:44 am

(Jan 8, 2018 02:08 AM)jammy47 Wrote: [\[CITE\]](#) getmodulehandle to get base address of process after injecting dll.

basically get a handle to the process, inject dll, make dll run code, get base address of process and add the offset.

Yep I have made a GetModule function, and I know how to use it, however I am unsure how to obtain the offset to add to the base.

For example, I can find the pointer, but I cannot find the base+offset address. It's always in the form of numbers, never "theHunterCotW_F.exe+*****"

jammy47 - January 9, 2018, 4:47 am

i think you can get that from the peb -> imagebaseaddress, but you'll have to check.

surenhohn - January 9, 2018, 2:47 pm

Usually it works like this

Code // get base of code as an absolute reference
DWORD dwBase = (DWORD)GetModuleHandle("game.exe/dll");
Then from there on you need to find pointers to whatever structure/class/function you're looking for (or use pattern search).
Usually there things are classes and/or created dynamically (e.g. class constructor, new/HeapAlloc/VirtualAlloc)... or like virtual class members / functions volatile by itself. Here you need to do the following:

A) find a pointer to the structure of you cannot hard-code the offset

Code {\$base + someoffset1} call VirtualAlloc.... // resulting ptr to new memory in eax
{\$base + someoffset2} mov dword ptr [someoffset], eax
Steal the pointer by referencing that immediate constant (someoffset) like

Code DWORD dwPtrINeed = (DWORD)(dwBase + *(PDWORD)((\$base + someoffset) - \$base)))
where \$base is the default base (needed if ASLR is enabled).

Sometimes all you'll find is a pointer to a pointer that will hold the offsets/data you're looking for.
Example:

Code DWORD dwPtrINeed = **(PDWORD*)(....

B) If there is no fixed reference you can hijack OR if you need to intercept the game at a certain point of time (like hooking / redirecting a syscall),
try to hook (detour code) whatever is being executed right "before" that. Often you may want to grap a copy of the D3D Interface ptr, like this

Code HRESULT WINAPI CreateDevice_Hook(void* pDirect3D, unsigned int uiAdapter, D3DDEVTYPE pDeviceType, HWND hFocusWindow, unsigned long ulBehaviorFlags, D3DPRESENT_PARAMETERS* pPresentationParameters, LPDIRECT3DDEVICE9* ppReturnedDeviceInterface)
{
 HRESULT hrReturn = pCreateDevice(pDirect3D, uiAdapter, pDeviceType, hFocusWindow, ulBehaviorFlags, pPresentationParameters, ppReturnedDeviceInterface);

if (hrReturn == D3D_OK)
{
 dwDevice = (DWORD*)(DWORD*)*ppReturnedDeviceInterface;
 pDrawIndexedPrimitive = (DrawIndexedPrimitive_t)dwDevice[82];
 ...

All pseudo-code... or short snippets from cheats I made over the years... just to give you an example.

Hope it helps.