Introduction to Reverse Engineering

Reverse Engineering Video Games

How memory is stored and how to find it

Before you get started with memory manipulation you need to understand how values are stored in memory and how to retrieve its pointer

Addresses, Offsets, and Pointers

An easier way of understanding addresses, offsets, and pointers is by using a real life analogy.

Imagine you are a postal worker and need to deliver mail to an apartment. To do so you will need the **address** of the property. But you need to actually locate the apartment, this is where the **offset** comes into use. Offsets are used to read values held by the address. Sometimes there are chains of offsets.

A pointer is the “final route to the apartment” to get the value you’re searching for. Pointers may work together in a chain to get a value.

Addresses and offsets are stored in hexadecimal which is a human friendly way of interpreting binary values.

Application to Manipulate Memory

Cheat Engine

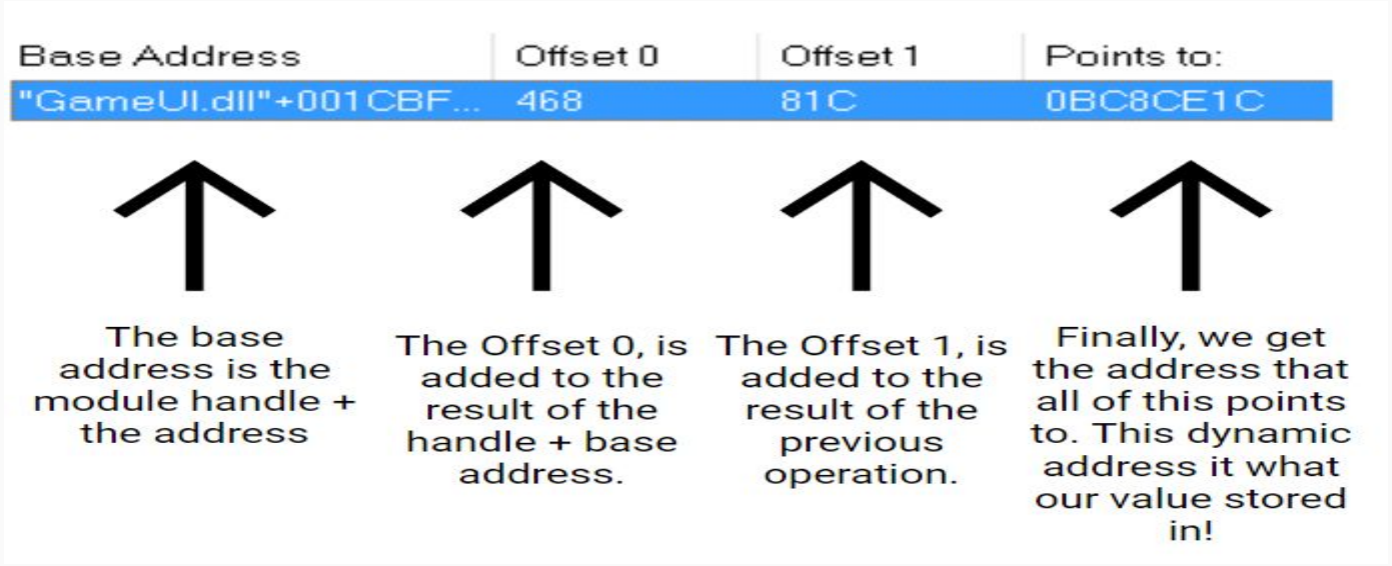
Cheat Engine is a popular way to view and manipulate the values of pointers because it is a free and easy to use application. You can both scan and manipulate pointers with Cheat Engine.

How do I know if I can use Cheat Engine to Exploit Games?

Whenever a video game manages code on the client side, the pointers to sensitive values, such as money or health, become vulnerable to manipulation.

You'd be surprised at how often Cheat Engine actually works on even major gaming titles. For example in popular video game, Grand Theft Auto V, you can exploit the money system quite easily with cheat engine. For example, when you sell a car in the game, you can scan for the price in Cheat Engine, you can then modify the car to change the price so you can narrow down the possible memory pointers and scan again. From there you can just select the narrowed down pointers and change the value to whatever price you want to sell the car for. There are thousands of examples like this in many video games however the core thing to search for is client side pointer management. If it's server-sided you cannot abuse the game this way.

Example of Pointer Path in Cheat Engine



Base Address	Offset 0	Offset 1	Points to:
"GameUI.dll"+001CBF...	468	81C	0BC8CE1C

The base address is the module handle + the address

The Offset 0, is added to the result of the handle + base address.

The Offset 1, is added to the result of the previous operation.

Finally, we get the address that all of this points to. This dynamic address it what our value stored in!

Why the C Family of Languages is Most Prevalent

Any programming language can be used to hack as long as it’s “low-level” meaning it can easily communicate with hardware and OS of the computer.

The C family of languages are the most widely used when creating cheats as they are low-level languages and do not need many external dependencies. This allows for C languages to directly and efficiently manipulate memory through pointers.

External, Kernel, and Internal

There are three main ways to make a cheat function on a target: External, Internal, and Kernel.

External

External cheats typically use a .exe(executable) file and open a handle to the process you wish to target. They are ring-3(next slide.)

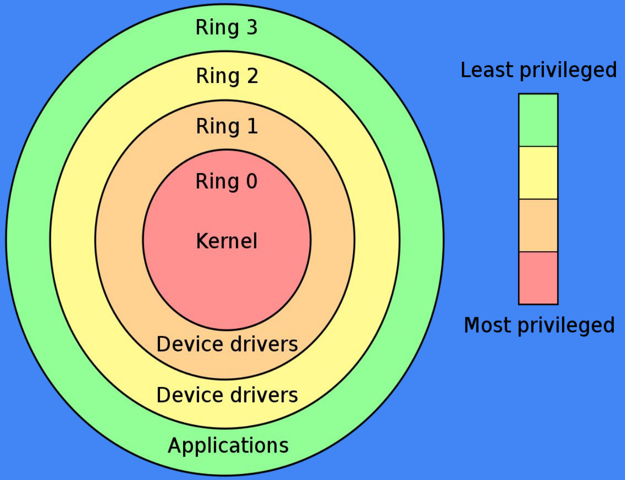
Internal

Internal cheats do not use an external executable file but instead inject .dll files into the game’s process in order to change the code. Internal cheats do not need a handle to the process to access memory.

Kernel

Kernel cheats are located in “Kernel Land” and are ring-0. You only need the process ID to access memory.

Rings of Privilege



In the previous slides I mentioned various rings of privilege and kernel land.

Ring-3 is the least privileged ring. Most of the time applications such as executables and DLLs are ring-3.

Ring-0, or the Kernel, is the most privileged ring. This has control over all other rings and is the most powerful on the PC. Although, ring-0 hacks can still be detected if the kernels aren't all signed, or if you haven't found a way to bypass the required signatures.

The PID

https://learn.microsoft.com/en-gb/windows/win32/toolhelp/taking-a-snapshot-and-viewing-processes

Process ID

Each process in Windows is allocated a unique identifier when it’s executed. It can be up to 5 numbers long and is read as a DWORD.

How do I get a PID in C++?

To get the PID of a process, you can take a snapshot of all running processes by using the command “CreateToolHelp32Snapshot()” and passing TH32CS_SNAPPROCESS as the parameter. This returns a handle to a snapshot of all running processes which you can enumerate with further code to find the process you’re looking for.

How do I open a handle to a process?

In order to access memory and manipulate it, you need to open a handle to the process. A handle to the process can be found with the OpenProcess() function.

This function is located in the Windows header file, so in order to use it you must import Windows.h. The function takes three parameters. If the function succeeds then it will return an open handle, if it fails it will return null.

https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-openprocesshttps://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-openprocess

Reading and Writing Memory

https://docs.microsoft.com/en-us/windows/desktop/api/memoryapi/nf-memoryapi-readprocessmemory
https://docs.microsoft.com/en-us/windows/desktop/api/memoryapi/nf-memoryapi-writeprocessmemory

The function used to read memory is “ReadProcessMemory().” The function takes in four parameters. After executing the function, the memory you are trying to read is copied into a buffer, which is defined by the parameter IpBuffer. If it succeeds it returns a non-zero value, if it fails it returns zero.

The function used to write to memory is “WriteProcessMemory().” This function also takes in several parameters. IpBuffer contains the data you wish to write to memory. The function returns a non-zero value if it succeeds and returns zero if it doesn’t.

Closing the Handle

To close the handle we use the CloseHandle() function. This function only requires one parameter, hObject, which is the handle we’re trying to close. We close the handle when we’re finished modifying or reading values or exiting the cheat. Leaving a handle open would make it easier for a cheat to be detected.

Bypassing an Anti-Cheat

Now to the hardest and most abstract part of reverse engineering

Anti-Cheats and How to Bypass One

Common anti-cheats

The three main anti-cheats used today are Battle Eye, Easy Anti Cheat, and VAC. Game companies use these software to prevent unauthorized access to memory. Some anti-cheats are better at detecting externals for example than others so the type of cheat you create will depend on the anticheat's weakness.

How to bypass anti-cheats?

Creating a bypass is very abstract and case to case. You can't just copy and paste a bypass and expect it to work. Most public bypass methods are detected by anticheats quite easily. Bypassing also requires intermediate coding knowledge. However once you learn how to code you can search terms like "injection bypass" on Github and look at the projects there for inspiration.

Thank You For Watching!