All popular games are utilizing kernel anticheat, in this cat and mouse game, the hackers must now enter kernel mode as well. Kernel Anticheat is very effective in preventing usermode cheats. This guide will provide you everything you need to know to start learning how to bypass kernel anticheat. If you have not finished the Guided Hacking Bible, do not waste your time on kernel anticheat, you're not ready.

The information provided in this guide will cover:

- Kernel Mode vs Usermode
- How to learn kernel driver development
- A video tutorial series covering kernel mode cheats
- How to exploit vulnerable drivers
- Common vulnerable drivers & tools
- An overview of the common functionality of kernel anticheats
- Detection of kernel cheats

Anticheats Utilizing Kernel Modules
BattleEye, Xigncode, Easy Anti Cheat, Vanguard

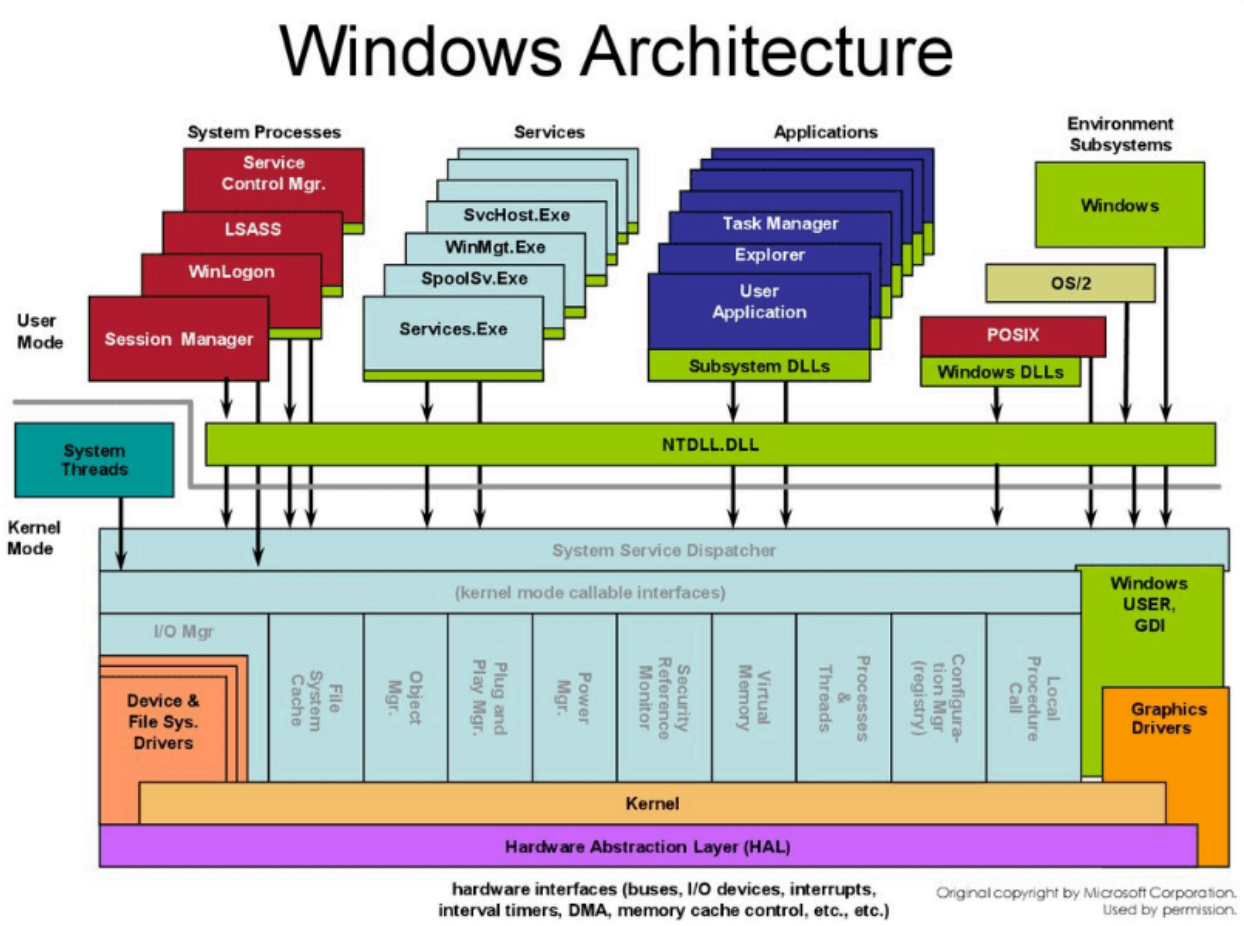
What is a kernel mode driver & Kernel Mode vs User Mode
A processor in a Windows computer has two different modes: kernel mode and user mode. The processor switches between the two modes depending on what type of code is running. Normal .exe programs run in user mode & core operating system components run in kernel mode. The Usermode & Kernelmode construct is built into the CPU. The low level core functionality of the operating system is done in kernel mode, which is a privileged part of memory that is not accessible from user mode and executes with privileged status on the CPU. Drivers are not just limited to Hardware Drivers, you can make a .sys driver to do anything you want in kernel mode, including bypass anticheat and perform cheat functionality.

A user mode process resides in it's own personal virtual address space that is private and doesn't interact with other processes's memory normally. Each application runs in isolation, if a regular program crashes, the crash is limited to that one application. Other applications and the operating system are not affected by the crash.

All code that runs in kernel mode shares a single virtual address space. This means that a kernel-mode driver is not isolated from other drivers and the operating system itself. If a kernel-mode driver accidentally writes to the wrong virtual address, data that belongs

to the operating system or another driver could be compromised. If a kernel-mode driver crashes, the entire operating system crashes.

Many of the privelages things you need to do in game hacking rely on the kernel performing those tasks for you. When you call WriteProcessMemory() for example, that function is exported by NTDLL.DLL and that request to write to the memory of another process is passed on to the kernel through NTDLL. Your application isn't actually doing it, the kernel is, your program is just making the request. View the image below to understand how kernel mode and usermode are separated.



User mode processes don't have access to kernel mode processes and memory. That is how the CPU and Operating System are designed.

How does this apply to bypassing Anticheat?
If you are dealing with a strong usermode anticheat, you can write a kernel mode driver to bypass it. Because you are in the kernel and the anticheat is not, you can modify the

anticheat to stop it's detection or you can hide your usermode module from it entirely. A user mode anticheat has no idea what you're doing in kernel.

If the anticheat has a kernel driver then you must also be in kernel mode, because nothing you do in usermode is going to be able to bypass or hide from a kernel anticheat. Generally speaking, kernel mode drivers are not necessary to hack 99% of games. In fact, kernel mode drivers are very easy to detect by anticheat if not done correctly.

Coding a kernel driver is much more complicated than user mode applications, for which reason your functionality which provides the "bypass" is done in the kernel but in most cases, the actual cheat logic is done in a usermode module. In this situation, you load your driver, enable your "bypass" functionality and then inject your DLL. Alternatively you can write your entire hack to run in kernel mode, which is more difficult.

But Rake, I don't want to learn, I just want to paste some crap and bypass anticheat!
Ok before we go to far, I will give you a simple 6 step process that is the easiest way to paste your way into kernel:

1. Video Tutorial - How to Make a Windows Kernel Mode Driver Tutorial
2. Video Tutorial - Kernel 2 - Usermode Communication - IOCTL Tutorial
3. Video Tutorial - How to Write Memory from Kernel - MmCopyVirtualMemory Tutorial
4. Experiment with this source code Source Code - CSGO Kernel Driver Multihack
5. Use kdmapper to manually map your kernel driver (make sure anticheat is not loaded yet)
6. Start the game and use your usermode application to write to the game memory

With those 5 steps, you can start writing to the memory of games with anticheat, you can easily be detected, but alot of people get started like this. For bypassing VAC & other older 5+ year old anticheats, this is all you need. But EAC and other strong kernel anticheats can detect this easily, so keep reading to learn more.

Kernel Driver Development
To get started with driver development start with these resources:

- Video Tutorial - How to Make a Windows Kernel Mode Driver Tutorial
- User mode and kernel mode - Windows drivers
- Kernel-Mode Driver Architecture Design Guide - Windows drivers
- Getting started with Windows drivers
- Download the Windows Driver Kit (WDK)

- Windows Driver Development - Windows Hardware Dev Center
- Write a universal Hello World driver (KMDF)

Driver Signing & Test Signing
Windows security would certainly be lacking if you could just load any kernel driver you wanted. This is why Windows requires your kernel mode driver to be signed with a security certificate in order for the OS to load it, but don't worry you don't need to pay 200$ for a certificate. You need to enable Test Signing if you want to load a driver you're actively developing.

In the past you could disable Driver Signing by running these commands as admin and rebooting:
C++:
```
bcdedit.exe -set loadoptions DDISABLE_INTEGRITY_CHECKS
bcdedit.exe -set TESTSIGNING ON
```

On Windows 8 and 10 you may need to do this by accessing the Advanced Boot Options menu by pressing F8 during boot. Windows 10 has disabled the F8 hotkey, to re-enable it:
C++:
```
bcdedit /set {default} bootmenupolicy legacy
```

Then reboot, and press F8 before Windows loads and you will see a menu in which you can Disable Driver Signing. Alternatively on Windows 10 you can hold SHIFT when you click Restart, and this menu will appear. But it only works for that one reboot, you need to do it every time because Windows 10 resets it back to default value.

Kernel Anticheats Prevent games from loading when Test Signing is enabled
The kernel anticheat developers got wise to this, and now they prevent you from playing the game if Test Signing is enabled. So you're forced to enable Driver Signing.

Then how do you load your driver? Keep reading my young padawan.

Exploiting Kernel Drivers
Kernel drivers are very common not just for hardware drivers, many different types of software utilize them. Driver security is very poor and there are many vulnerable drivers. The drivers expose functions to their usermode applications, to make development easy and cheap, they often expose too much or provide functionality that is too dangerous.

Any driver that takes data from usermode and does something with it in kernel is potentially vulnerable. Many have buffer overflows which can be leveraged, or even worse an arbitrary kernel write vulnerability. These vulnerabilities can be exploited from usermode to execute your code, ideally providing a simple method to load your own driver.

But you can't just load your driver, you need to manually map it because it is not digitally signed. These vulnerable kernel drivers must have valid security certificates. By utilizing a valid & certified driver, you can manually map your unsigned driver without issue. Microsoft or the Certificate Authorities can decide to reject these certificates at any time, making them no longer work, but that is extremely rare.

For learning purposes learn to use KDMapper first, and then learn how to use KDU

KDMapper
KDMapper is used by hundreds of pay cheat providers and for good reason, it's super paste friendly.

- Utilizes an embedded vulnerable Intel driver
- Manually Maps your driver
- Provides a simple command line interface
- You just pass it 1 argument and you're driver is loaded

KDMapper comes embedded with the vulnerable iqvw64e.sys Intel Ethernet diagnostics driver driver. The driver is embedded as a byte array in intel_driver_resource.hpp

The driver was signed in 2013. The vulnerability was officially published in 2015 as CVE 2015 2291 with a severity score of 7.8. Amazingly it's certificate has not been revoked yet.

iqvw64e.sys
Code:
sha256:
B2B2A748EA3754C90C83E1930336CF76C5DF9CBB1E3EEC175164BB01A54A4701
CompanyName: Intel Corporation
FileDescription: Intel(R) Network Adapter Diagnostic Driver
InternalName: iQVW64.SYS
ProductName: Intel(R) iQVW64.SYS
ProductVersion: 1.03.0.7

iqvw64e.sys Main Intel Signature

image
But wait it's not valid after 2015! Wrong! Windows still loads it.

Counter Signer Symantec Time Signature
image

What happens in December 2020? Nothing! Microsoft will continue to load it as long as it is not revoked!

The vulnerability exists due to insufficient input buffer validation when the driver processes IOCTL codes 0x80862013, 0x8086200B, 0x8086200F, 0x80862007 using METHOD_NEITHER and due to insecure permissions allowing everyone read and write access to privileged use only functionality.

KdMapper utilizes IOCTL code 0x80862007 for arbitrary kernel execute
image

KDMapper is very easy to detect by anticheat - The driver is well documented, everyone knows what it is. But it's a good start to get you exposed to kernel hacking. Read more @ Download - KDMapper.

List of vulnerable drivers
There are probably thousands of vulnerable drivers, here are some we know about.
Learn more about this list @ Discuss - New vulnerable kernel drivers

- iqvw64e.sys
- gpcidrv64.sys
- AsUpIO64.sys
- AsrDrv10.sys
- AsrDrv101.sys
- AsrDrv102.sys
- AsrDrv103.sys
- BSMEMx64.sys
- BSMIXP64.sys
- BSMIx64.sys
- BS_Flash64.sys
- BS_HWMIO64_W10.sys
- BS_HWMIo64.sys
- BS_I2c64.sys

- GLCKIO2.sys
- GVCIDrv64.sys
- HwOs2Ec10x64.sys
- HwOs2Ec7x64.sys
- MsIo64.sys
- NBIOLib_X64.sys
- NCHGBIOS2x64.SYS
- NTIOLib_X64.sys
- PhlashNT.sys
- Phymemx64.sys
- UCOREW64.SYS
- WinFlash64.sys
- WinRing0x64.sys
- amifldrv64.sys
- atillk64.sys
- dbk64.sys
- mtcBSv64.sys
- nvflash.sys
- nvflsh64.sys
- phymem64.sys
- rtkio64.sys
- rtkiow10x64.sys
- rtkiow8x64.sys
- segwindrvx64.sys
- superbmc.sys
- semav6msr.sys
- piddrv64.sys
- RTCore64
- Gdrv
- ATSZIO64
- MICSYS
- GLCKIO2
- EneIo
- WinRing0x64
- EneTechIo

Vulnerable Driver Resources

- Discuss - New vulnerable kernel drivers
- Weaponizing vulnerable driver for privilege escalation— Gigabyte Edition!

- EvanMcBroom/PoCs
- Escaping SMEP Hell: Exploiting Capcom Driver In a Safe Manner
- can1357/safe_capcom
- notscimmy/libcapcom
- Bypassing Anti-Cheats - Part 1 - Exploiting Razer Synapse Driver - Niemand - Cyber Security
- Mother of All Drivers - New Vulnerabilities Found in Windows Drivers - Eclypsium

Everything from hfiref0x is amazing
hfiref0x - Overview specifically -> hfiref0x/KDU
The Vault
@hFireF0X

WOW LOOK AT ME, I BYPASSED KERNEL ANTICHEAT!

You literally did nothing except paste. Stop saying "I have a bypass", you have the same bypass that another 100,000 people are using and all you did was download kdmapper. You're not special, so just shut up please, we're not impressed. Saying "I have a bypass" when you're using kdmapper is like saying "I have Cheat Engine".

General Functionality of Kernel Anticheats
- All the normal usermode detections
- Blocking / stripping of process handles
- Detection of test signing
- Detection of usermode hooks
- Detection of injected modules
- Detection of manually mapped modules
- Detection of kernel drivers
- Detecting of traces of manually mapped drivers
- Detection of virtual machines and emulation

Manually Mapped Driver Detection
You must bypass these things, clear PiDDBCacheTable & MmUnloadedDrivers, and stop the enumeration of your own system pools & threads.
PiDDBCacheTable & MmUnloadedDrivers
system pool detection
system thread detection
Source Code - How to Clear PiDDBCache Table / PiDDBLock

PatchGuard
PatchGuard detects patches in the kernel, you can't just patch the anticheat's kernel driver

What Next?
So you can manually map your driver, and you can read and write memory, what do you do next?

Well you didn't really bypass the anticheat. All you did was load a cheat they didn't detect yet, and now it's very likely they have seen your modules. If the same modules are detected on multiple machines, you may find yourself in the next ban wave. Just making a driver and mapping it doesn't bypass anything. Kernel anticheats are incredibly invasive and they can detect everything that's happening on your system. If you're doing something that looks malicious, they can easily detect it and ban you.

Kernel Anticheat typically are used in combination with a usermode module, which is manually mapped into the game and obfuscated. Your next step is to dump both the kernel module and the usermode module and reverse engineer them. Then you will have a very good idea of how they operate, and what else you need to do completely bypass the anticheat.

Remember, you can't patch the kernel anticheat, so you need to go around it.

Next you want to patch all the usermode detections so you can attach a debugger, especially Cheat Engine & Reclass so you can start reversing the game.

From kernel you can patch or hook all the detection mechanisms in the anticheat's usermode module, and you can use your own kernel module to protect & hide your own usermode module. Essentially you want to block the anticheat from accessing any of your modules address range. Once you've taken care of all of that, you can inject your usermode module without any trouble.

Detection of Kernel Cheats
It's super easy for them to detect vulnerable drivers, the anticheat devs have the same list of vulnerable drivers that we have and they are actively scanning for the most popular ones. If they find your module they will upload it to their server, analyze it and build detection for it.

EAC for example has some very good detection methods, regardless of which anticheat you're trying to bypass you should read our EAC thread to learn more.

A manually mapped driver cannot be detected using the normal methods, but mapping your driver does leave traces behind. Make sure you clear PiDDBCacheTable and anything else your driver leaves behind.

Guided Hacking Kernel Videos
Video Tutorial - How to Make a Windows Kernel Mode Driver Tutorial
Video Tutorial - Kernel 2 - Usermode Communication - IOCTL Tutorial
Video Tutorial - How to Write Memory from Kernel - MmCopyVirtualMemory Tutorial

GH Resources
Download - KDMapper - Manually Map Kernel Drivers CVE-2015-229
Guide - How to Bypass EAC - Easy Anti Cheat
Tutorial - MTA: SA's kernel mode anticheat is a joke (information)
Guide - Anticheat Battleye Bypass Overview
Guide - How to bypass XignCode Anticheat Guide - XignCode3
Source Code - CSGO Kernel Driver Multihack
Tutorial - MTA: SA's kernel mode anticheat is a joke (information)
Guide - How anti-cheats detect system emulation
Download - GamersClub Anti-Cheat Information (Driver + user mode module)
External Resources
All secret.club articles
https://back.engineering/
https://github.com/hfiref0x/TDL
https://github.com/hfiref0x/KDU
https://github.com/hacksysteam/HackSysExtremeVulnerableDriver
https://github.com/Zer0Mem0ry/ntoskrnl
https://github.com/FuzzySecurity/Capcom-Rootkit
https://github.com/tandasat/ExploitCapcom
https://github.com/SamLarenN/CapcomDKOM
https://github.com/BlueSkeye/CapcomDriver
https://github.com/zerosum0x0/ShellcodeDriver