Security in 3D Graphics Drivers

Submitted By

ShubhamKumar Patel

17MCEI11

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2019

Security in 3D Graphics Drivers

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science & Engineering (Information Network and Security)

Submitted By

Shubhamkumar Patel

(17MCEI11)

Guided By

Prof.Jigna Patel

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2019

Certificate

This is to certify that the Major project entitled **"Security in 3D Graphics Drivers"** submitted by **Shubhamkumar Patel (Roll No: 17MCEI11)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science & Engineering (Information Network and Security) of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Jigna Patel
Guide, Assistant Professor
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Sharada Valiveti
Associate Professor,
Coordinator M.Tech CSE(ISN),
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Madhuri Bhavsar
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr Alka Mahajan
Director,
Institute of Technology,
Nirma University, Ahmedabad

Statement of Originality

I, **Shubhamkumar patel**, Roll.No.**17MCEI11**, give undertaking that the Major Project entitled "**Security in 3D Graphics**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering (Information Network and Security)** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

_____

Signature of Student

Date:

Place:

Endorsed by

Prof. Jigna Patel

(Signature of Guide)

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **prof. Jigna Patel**, Assistant Professor of Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmadabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Madhuri Bhavsar** Hon'ble Head of Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for her kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr. Alka Mahajan**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation she has extended throughout course of this work.

I would also thank the Institution, all faculty members of Information & Technology Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

I am truly grateful to my parents for their blessings and constant motivation which helped me to complete this project successfully. Last but not the least, I would like to thank God for always being there for me.

- Shubhamkumar Patel

17MCEI11

Abstract

User mode driver is software component which is a medium of communication between operating system and a device. Driver plays major in role what we see on display. For rendering 3d graphics on screen it needs to render through specific UMD.

Optimization of debug methods and debug tools will help to reduce the time that we are currently spending over the debugs and the optimization of graphics driver code will help to reduce the number of bugs arriving in graphics driver. Incorporation of such functionality into Direct3D UMD (User Mode Driver) Driver not only improves performance, but also saves a lot of time required to debug the driver for a specific cause. It allows us to find root cause by providing the option to dump commands category wise. Hence improves debug throughput time require to debug the driver and avoid the need to debug the whole driver for any issue and also improves the performance.

For running driver securely through hardware to display final render on screen, needs two kind of security 1) code security 2) hardware security 3) automation corruption. Providing hardware security is part of kernel mode driver (KMD) and automation is kind of work which can improve throughput and prevent easy access for security threats.

Abbreviations

OS	Operating System. .
TDR	Time Detection Recovery.
UMD	User Mode Driver.
KMD	Kernal Mode Driver .
WDDM	Windows Display Driver Model.
CPU	Central Processing Unit
GPU	Graphics Processing Unit.
GTA	Graphics Test Automation.

Contents

List of Figures

Chapter 1

Introduction

1.1 General Introduction to the topic

- **Microsoft Direct 3D:** 3D User mode driver is responsible for the games, scientific applications and any 3D video playback on computer. DirectX is nothing but low-level API which will draw primitives like triangle, square, cube on screen. Basically, many processes and threads are being processed in background but it will be hidden. It is designed to drive separate graphics processor for processing only graphics related content. New GPU has hundreds-thousands of parallel processors for handling multiple processes. Before operation is started bunch of rendering or compute state is set up and checked.

- **Direct3D Rendering Pipeline:** For generating graphics for real-time gaming applications Direct3D pipeline has been designed. Different stages include: Input assembler, Vertex shader, Tessellation, Geometry shader, Rasterizer, Pixel shader, Output merger.

- **Windows Display Driver Model:** Usermode and Kernelmode driver together interacts with Operating system and final output is seen on display, this architecture is called WDDM . Graphics hardware vendor will finally pass both display driver and miniport display driver for last output. dynamic-link library (DLL) is nothing but usermode display driver that is loaded by the Microsoft Direct3D runtime. Mi-

crosoft DirectX graphics kernel subsystem is communicating with miniport display driver in architecture.

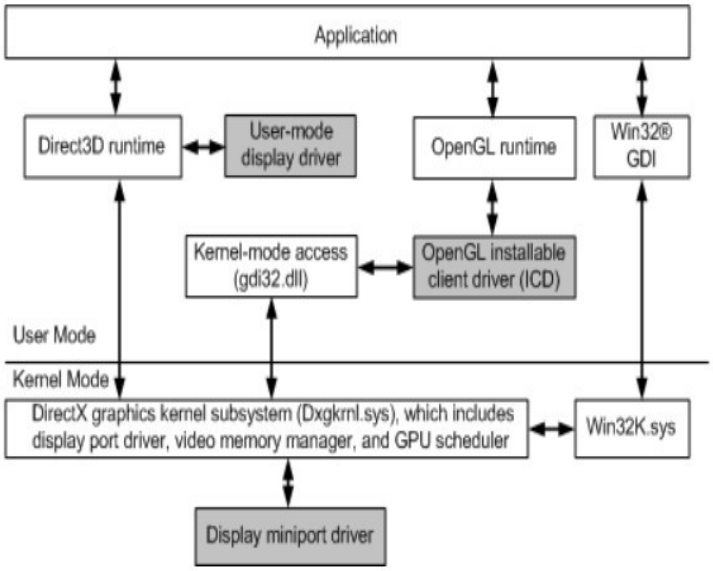

Figure 1.1: WDDM Architecture, Source: Microsoft documentation

- **Resource in Direct3D:** Whatever we see on screen those are resources. To render a scene on screen resources are responsible because they contain all required data. In Direct3D pipeline Resources are areas in memory that can be accessed directly or indirectly. Resources contain : geometry, textures, and shader data. Example of Direct3D resources are buffers and textures. Resources can be strongly typed or type less; one can have read and write, direct or indirect access to resources; they can also be made accessible to only the CPU, GPU, or both.

- **Pre-Silicon process:** Pre-silicon is process in which the devices are tested in a virtual environment with sophisticated simulation without real hardware on emulation, and formal verification tools. Where as Post-silicon validation tests will test actual devices running at-speed in real-world system boards using logic analyzer and assertion-based tools. To reach high confidence and accuracy is very important in pre-silicon verification work- which will require 30% of the overall expense of the implementation and we can use that knowledge for the post-silicon system to

productize valuable product in market.

1.2 Area of Computer Science

- **Computer Graphics:** Computer graphics is a field of Computer Science which studies methods of displaying and modifying screen pixels from hardware to what we see on screen digitally visual content.

- **Hardware Security:** Micro processing unit is the hardware which manages every core of computer hence providing security at low level is very essential.

- **Software Security:** : User mode driver is the component which is designed to give access to users for accessing memory from hardware at unit level, so if that access is not controlled in manner it can cause lot of failures and data loss.

1.3 Hardware and Software Requirements

Hardware requirements:

- Intel next generation platforms.

Software requirements:

- Operating system: Windows10 RS4 and later.
- Programming language: C++

Chapter 2

Specific Objectives and Scope of the Project

2.1 Objectives:

1. To employ concepts of 3D graphics driver to provide security from software as well as hardware side.
2. To build a tool for improvising the debug throughput, and to ease the effort.
3. To provide event traces in code to prevent untraceable exploit.

2.2 Scope:

1. The tool will help in improving the debug throughput.
2. The tool can improve the overall performance.
3. It can be used with Intels next generation graphics platform development.
4. Events tracing can be used to provide fault detection for current/upcoming platform.

Chapter 3

Background

3.1 Direct3D pipeline

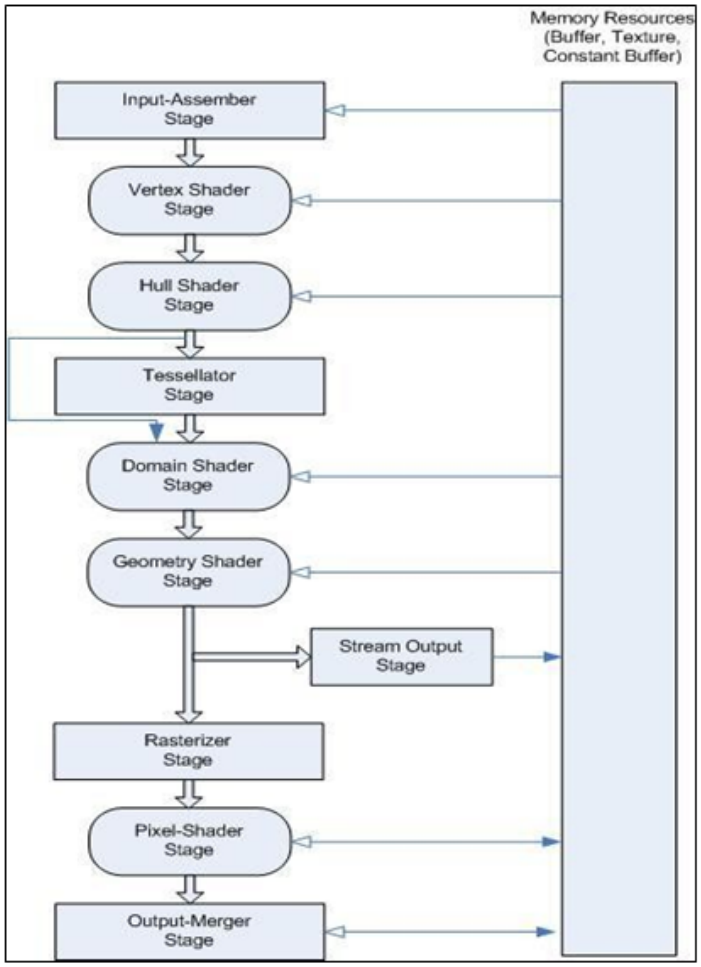

Figure 3.1: Direct 3D pipeline, Source: Microsoft documentation

- **Input-assembler stage:** In this stage 3D execution will start by providing data to pipeline.Data is transferred in form of primitives like triangle, square and rectangle. There are user buffers in which data is collected and processed in later stages . It can be converted into different primitive like line lists, triangle strips, or primitives with adjacency. For making shaders more efficient IA is adding system generated values to primitives.

- **Vertex shader stage:** In this stage mostly operations such as transformations, skinning, morphing and per-vertex lighting will happen on vertices. Fundamentally it will take single vertex as input and produce single vertex as output. They are always run on all vertices, it will run including adjacent vertices as input primitive topologies with adjacency.

- **Tessellation stages:** Main job of this stage is to transfer low level details to high level details on GPU.For rendering Tessellation stage breaks up high order surface into its structural surface.There are 3 stages for tessellation,

    1. **Hull shader stage:** This stage is programmable stage. This shader generates a geometry patch constants corresponding to respective input patch (example :triangle).

    2. **Tessellator stage:** This is fixed function pipeline stage. It generates pattern of samples. That samples shows the geometry patch and it will generate a set of smaller objects (triangles, points, or lines). Those objects will connect to samples in the end.

    3. **Domain shader stage:** A programmable shader stage, which calculates the vertex position corresponding to domain sample.

- **Geometry shader stage:** Vertices as input and vertices as output is the module of this stage It runs on application-based code. Vertex shaders operates on a single

vertex but in geometry shader inputs are full primitive vertices (two vertices for lines, three vertices for triangles, or single vertex for point). Geometry shaders can give the vertex data for the primitives of edge-adjacent.

- **Stream output stage:** Vertex data from the geometry-shader is constantly coming to this stage. Main goal of the stream-output stage is to take that input and process as output(stream) to one or more buffers in memory.

- **Rasterizer stage:** For displaying real time 3D graphics on screen it is very necessary to convert data in raster images, generally those images are comes to stage in vector form. Rasterization is the stage in which pixels are converted from primitives, it interpolats per-vertex values for primitive individually. Rasterization stage also does clipping of vertices to the view. After that division by z to provide view, To invoke the pixel shader mapping of primitive to 2D view takes place .

- **Pixel shader stage:** The pixel-shader stage (PS) is for applying different techniques such as per- pixel lighting and post-processing. pixel shader will merge all data such as constant variables, texture data, interpolated per-vertex values, and other data to produce per-pixel outputs.

- **Output merger stage:** The output-merger (OM) is responsible for generating final rendered pixel color after processing of pipeline state, the pixel data generated by the pixel shaders, the contents of the render targets, and the contents of the depth/stencil buffers. This stage is the final one , whatever we see on screen will be decided by this stage which pixels to render and which pixels to process in background (like depth stencil).

3.2 Direct3D Application development

Direct3D API is following the D3D pipeline flow so developing and Direct 3D application will be in same flow. Applications are nothing but executables that creates a DirectX device. Examples of direct3D applications: Games, Hardware accelerated desktop applications such as internet explorer, power point, desktop window manager. Knowledge

of windows C/C++ programming, win32 programming, linear algebra and trigonometry and basic graphics concepts is required for this development process.

Direct3D app development:

1. **Initialization of application:**
   Create a Context.
   Create a Device.

2. **Creation of state/resources/shaders**
   Describe Buffer
   Describe and create Swap chain.
   Create Back buffer.
   Create Render target.
   Create Input layout.
   Create Vertex shader.
   Create Pixel shader.

3. **Bind state/resources/shaders**
   Set Render target
   Set input layout
   Set Vertex buffer
   Set Vertex shader
   Set Pixel shader

4. **Draw a primitive - Draw ()**

5. **Present back buffer to the screen Present ()**

3.3 Intel Graphics technology

Intel produces incorporated illustrations processors on a similar bundle or pass on as the focal handling unit (CPU). It was presented in 2010 as Intel HD Graphics.graphics processors in intel are named "Intel Graphics Technology". The Intel designs processors

are separated into ages, with every age getting advancement terms of higher illustrations execution. The original of illustrations innovation was kick-begun with Westmere, where Clarkdale and Arrandale were the processors with Ironlake designs. Sandy scaffold (2011), Ivy Bridge (2012) , Haswell (2012), Broadwell (2013), Braswell (2014), Skylake (2015), Apollo lake, Kaby lake, Coffee lake and Gemini lake came later. Icelake, Whiskey lake, Cannon lake, Icelake, Tiger lake and Alder lake are in line for the discharge.

Microsoft based Direct 3D apis are upheld by intel regarding illustrations drivers which is only UMD . The D3D group at Intel composes client mode show drivers for showcase connectors. The client mode show driver is a dynamic connection library (DLL) that is stacked by the Microsoft Direct 3d runtime.

The client mode drivers bolster the Direct3D adaptation 9 and 10. Client mode drivers are a piece of the Windows show driver model (WDDM-Fig 1) which remains as an extension between the Direct3D illustrations runtime and the Graphics equipment. Returning blunder codes got from runtime capacities, preparing shader codes, changing over Direct3D fixed capacity state, duplicating profundity stencil esteems, approving list esteems, support for numerous processors, taking care of various locks are a portion of the basic errands of client mode drivers.

3.4 Debugging:

- **Draw call minimization** At the point when a 3D object is rendered to the screen, each draw call draws a crude, the draw calls through and through makes up the total item on a scene. At last, the back cradle is exhibited to the screen for presentation. A large number of casings establish a video or diversion, where in each edge comprises of thousands of draw calls. Any disappointment in the designs processor part may result in defilement of the rendered scenes in the video or ongoing interaction. Lighting, textural and, geometrical are a portion of the defilements ordinarily watched.

  While investigating any debasement issue the significant advance to be pursued

is to limit the defilement to a solitary draw call. This can be accomplished by finding the precise casing from which the debasement starts, and later recognize the guilty party draw call. Narrowing down to a draw call can help in finding the main driver of the issue. DX Caps is one of the devices used to discover the offender draw call, likewise to recognize the issues. This designs analyzer apparatus is a useful instrument to investigate pixel level checking and to outwardly check the pipeline organize sources of info and yields, render target properties and some more.

Figure 3.2: A frame without corruption (Left) and frame with corruption (Right)

- **TDR and Page fault:**

  A most generally discovered security issue in illustrations is PC hang or a solidified conduct. However, as a general rule the PC could be preparing an end client order or activity. The PC gives off an impression of being solidified ordinarily in light of the fact that the GPU is occupied with handling concentrated graphical tasks, for the most part amid interactivity. The GPU doesn't refresh the presentation screen, and the PC seems solidified.

  The working framework endeavors to identify circumstances in which PCs seem, by all accounts, to be totally "solidified". The working framework at that point en-

deavors to progressively recuperate from the solidified circumstances so work areas are responsive once more. This procedure of location and recuperation is known as break identification and recuperation (TDR). In the TDR procedure, the working framework's GPU scheduler calls the presentation miniport driver's capacity to reinitialize the driver and reset the GPU. In this manner, end clients are not required to reboot the working framework, which incredibly improves their experience. The GPU scheduler, which is part of the DirectX graphics kernel subsystem (Dxgkrnl.sys), detects that the GPU is taking more than the permitted amount of time to execute a particular task. The GPU scheduler then tries to preempt this particular task. The preempt operation has a "wait" timeout, which is the actual TDR timeout. This step is thus the timeout detection phase of process. The default timeout period in Windows Vista and later operating systems is 2 seconds. On the off chance that the GPU can't finish or appropriate the present errand inside the TDR break period, the working framework analyze that the GPU is solidified. To counteract break identification from happening, it ought to be guaranteed that designs tasks (that is, immediate memory get to (DMA) cradle finish) take close to 2 seconds in end-client situations, for example, profitability and diversion play.

A page issue is a kind of exemption raised by equipment when a running system gets to a memory page that isn't at present mapped by the memory the board unit (MMU) into the virtual location space of a procedure. Indeed, even a page deficiency results in a PC hang/solidify.

At whatever point a page issue or TDR happens initial phase in investigating is to gather the dumps from the Windows live bit reports. Likewise the occasion supervisor helps in following the occasion that set off the TDR. The live portion dumps gathered aides in checking further investigating of the issue. The apparatuses accessible will help in finding the part that is coming up short and causing the issue.

3.5 Multi-user setup for pre silicon debugging

Wind River Simics is a full framework reenactment innovation that gives programming and framework designers, planners, and test builds an approach to manufacture and uti-

lize a virtual framework or make numerous virtual associated frameworks for different purposes.

Unmodified programming segments keep running in virtual frameworks as though they are running on genuine equipment from the perspective of functionalities. The innovation is adaptable and can recreate a solitary part, a board, numerous sheets, and even the system. Simics IA models are at present utilized and tried by Intel for present and forthcoming items. The innovation is expertly upheld by Wind River, so clients can tailor the answers for meet the one of a kind necessities all alone frameworks.

Multi-user setup in Linux environment:

The prerequisite for an incredible domain to troubleshoot pre silicon issues, required multi-client setup on Linux frameworks with numerous virtual frameworks running in parallel. The necessity is to run the reproduction of various future Intel designs stages on various virtual frameworks parallelly. Instating the diverse parameters to run the virtual machines and begin running them requires a great deal of exertion. The intuitive UIs are very slower in a reproduction, and henceforth needs a great deal of time for finish of a given activity, and subsequently expanding the troubleshooting term. Thus, it is increasingly profitable and favorable to have distinctive virtual machines running in parallel performing diverse errands.

The production of multi-client condition and synchronous various free remote associations with them should be empowered for helpful utilization of the accessible assets. The synchronous free access to the server helps in the correct usage of the figuring power. Synchronous access to a similar framework ought not meddle with the work taken care of by each virtual machine. Every one of the means should be executed in an unequivocal request to set the framework up.

The automation of creating a Linux multi user environment and setting up the parameters and installation of different platforms to simulate the environments to start the virtual machines for pre-silicon debugging is an important task.

3.6 End to end compression and decompression tool for debugging

Assets give information to the pipeline and characterize what is rendered amid a scene. Assets can be stacked from your amusement media or made powerfully at run time. Commonly, assets incorporate surface information, vertex information, and shader information. Most Direct3D applications make and pulverize assets widely all through their life expectancy.

While investigating, assets can't be gotten to and saw direct since the assets are packed. It is required to get to the assets and view them to troubleshoot, subsequently a decompression instrument helps in getting to the assets.

Chapter 4

Methodology

Assets contain the information that gets rendered to finish a scene. These are the territories in memory that can be gotten to by the Direct3D pipeline. Assets contain information like surfaces, shader information, and geometry. There are two different ways to completely determine the format (or memory impression) of an asset:
Typed - fully specify the type when the resource is created.
Typeless - fully specify the type when the resource is bound to the pipeline.

Assets can be specifically or typeless; assets can be controlled to have both perused and compose get to; assets can be influenced available to just to the CPU, GPU, or both. Up to 128 assets can be dynamic for every pipeline arrange. The lifecycle of a Direct3D asset is:
Creation of the resource.
Bind to pipeline.
Deallocate.

Cushions contain information that is utilized for depicting geometry, ordering geometry data, and shader constants. Supports are utilized to store a wide assortment of information, including position vectors, ordinary vectors, surface facilitates in a vertex cushion, lists in a list cradle, or gadget state. Vertex cushion, record cradle and consistent cushion are the cradle asset types upheld. A surface asset is an organized gathering of information intended to store texels. A texel speaks to the littlest unit of a surface that can be perused or written to by the pipeline. In contrast to cushions, surfaces can be

separated by surface samplers as they are perused by shader units.

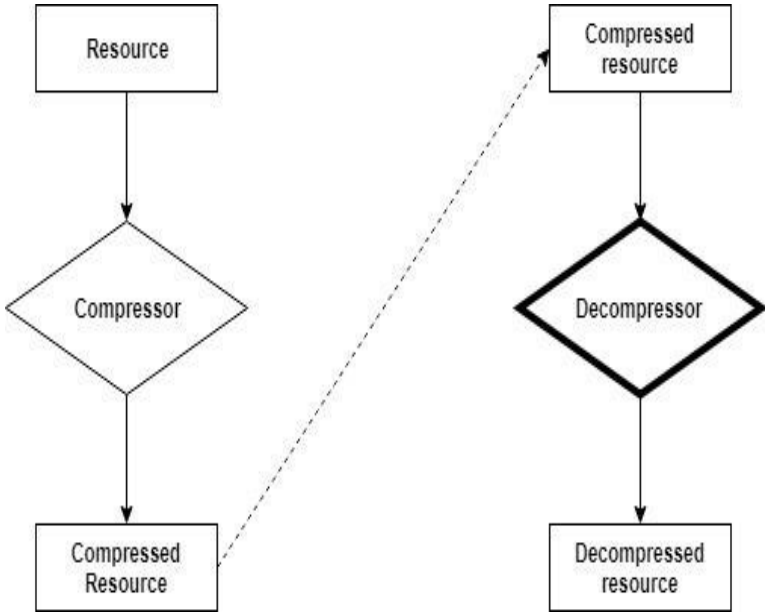

Figure 4.1: Block diagram of resource management

The assets are compacted to be made effectively open on memory. Pressure includes encoding data utilizing lesser space than the first portrayal. Pressure can be lossy pressure or lossless pressure. Lossless pressure diminishes bits by distinguishing and taking out excess. No data is lost in lossless pressure. Lossless pressure calculations for the most part abuse measurable excess to speak to information without losing any data, so the procedure is reversible. Lossless pressure is conceivable on the grounds that most genuine information displays factual excess.

Lossy pressure diminishes bits by evacuating superfluous or less significant data. In lossy pressure, some loss of data is satisfactory. Dropping trivial detail from the information source can spare extra room. The assets when compacted can't be perused straightforwardly from the memory, however must be decompressed first to see. Subsequently a decompression method should be pursued to make the assets accessible amid troubleshooting.

Chapter 5

Implementation details

5.1 Basic 3D App Development

Developing the basic 3D primitive apps, basic primitive in 3D graphics is triangle. Using the triangle, one can develop any complex applications. In this first developed a 3D triangle then the cube and different structures with different movements developed like rotation, scaling and transition with light and color effects was achieved in C++. Fig 5.1 shows is the triangle application developed.

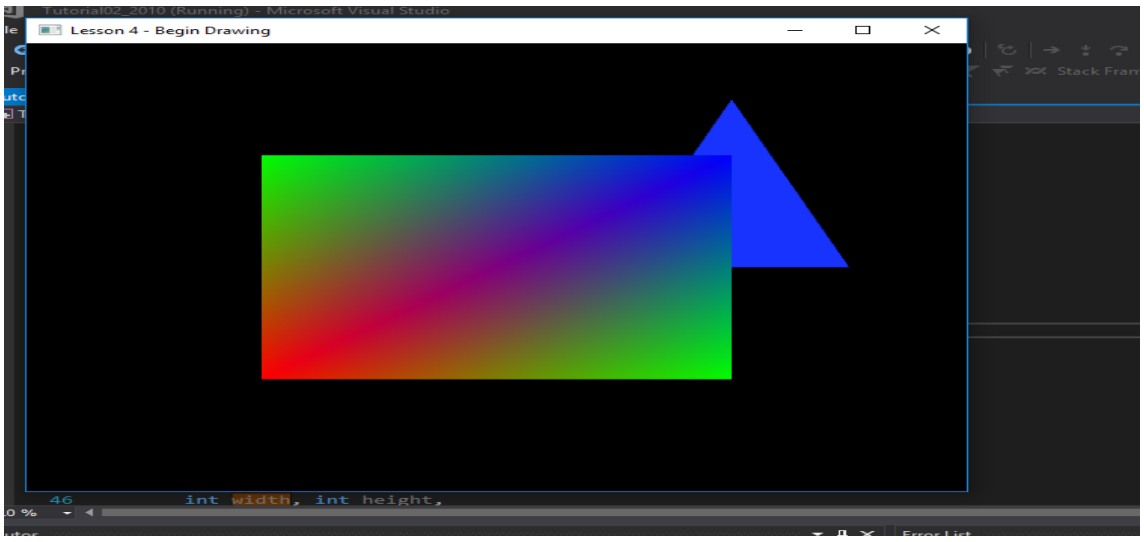

Figure 5.1: Basic triangle app using Directx

5.2 Providing Security Through Tracing

Events will be created for each command but one at a time. Events that exist in manifest file must have their definition declared in this header file. For example, ExampleProvider.h has been created after manifest file creation which contains definition for each event.

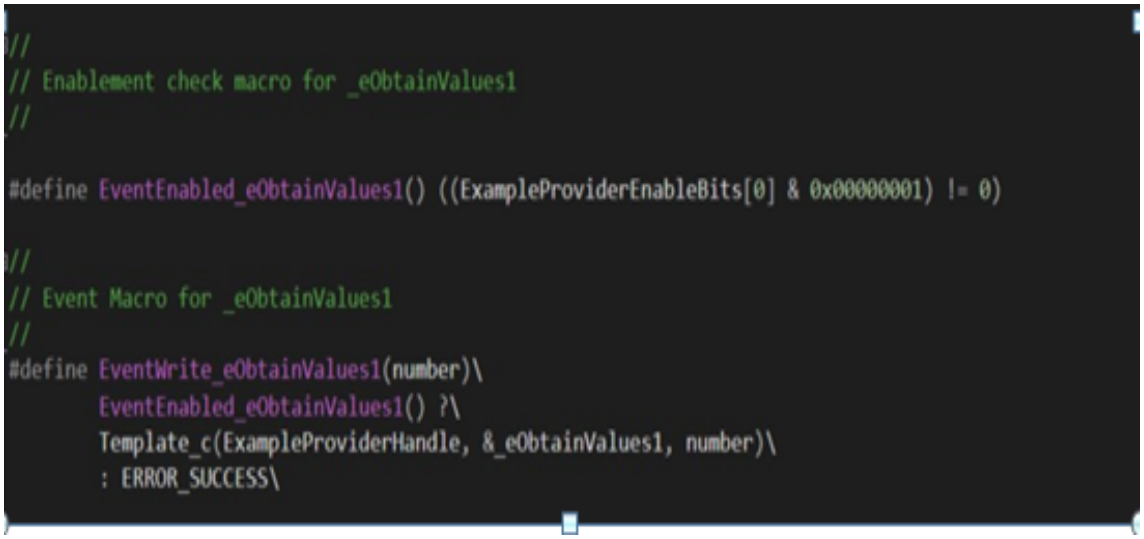

Figure 5.2: providing security using event traces

So this way, we can trace events happening in UMD, so itll be easy to check what goes wrong.

GTA:

Graphics test automation is platform that is used by intel for testing drivers, display and hardware in automated way. this tool is very hugely web-implemented so gathered all information and gave overview to the team.

Code Security :

In UMD code there is always C++ templates which is directly been implemented using functions and pointers.in traditional driver development operating system, driver and component connected to driver is implemented differently in separate groups. But for security purpose we can implement in model structure so whenever there is call, we can find where exactly it is going, which part of model is been called gives exact data.

```
#include "***.h" // including some .h file
#ifndef ***_REG_READ // local defines
  ... ...
#endif
  ... ...
// function prototypes
LOCAL STATUS ***DummyCallback ();
LOCAL void ***InitChannel (**);
LOCAL STATUS ***Ioctl (**, **);
  ... ...
//driver functions
LOCAL SIO_DRV_FUNCS ***SioDrvFuncs =
{
  **Ioctl,
  **TxStartup,
  ... ...
};
//define every function in detail
```

Figure 5.3: code security sample using model based approach

Previously driver code was functional with OOP conceptual coding methodologies. So it was difficult to track different calls and flow.

For providing security, model based approach is implemented. As you can see in the code there are function prototypes Dummy Callback(); is define and declare in one single model.Model is defined using C++ templates for example here SioDrvFuncs is there with body .

Benefit of this model based coding pattern is , its easy to track and flow is secure. Where previously it was really difficult to track the particular function or call coming from OS side and flow was insecure.

By this methodologies calls coming from OS will go to particular model so control of access is given to only OS not any third party component hence it is very secure.

5.3 Corruption Automation Tool

When any graphics workload is processing on hardware it will be having many draw calls and present calls. Every workload has different pass through hence it goes to different part of driver. Workload can be games, media , benchmarks and 3D applications.

So while drawing on screen loading of pixels or geometry can be improper for different workloads. So finally, it will have individual visual on screen and if any of the pixels are not properly drawn then it will end up in corruption.

Whoso ever is aware about hardware will know the exact flow of drawing on screen.so it is like exploit path for hackers. Now finding the corrupt draw call or minimized cause of corruption is bit time consuming.

Finding corrupted draw is bit lengthy process and sometime tricky too.for finding corrupted draw one need to understand what workload is doing in the user mode driver.starting point is running workload on specific driver and finding the calls which happening inside the driver.so from opening adapter to drawing corrupted draw is long process.

Flow is :

- Open adapter
- Create device
- Create context
- Create resource
- Write buffer
- Dump Frames
- Dump draws

This is very generic flow reaching to the draw level. It can vary as different application go to different area of driver.so innovation here is to create automated way of dumping frames. Implementation of tool is described. Tool will take care of all the steps required to dump corrupted draw.

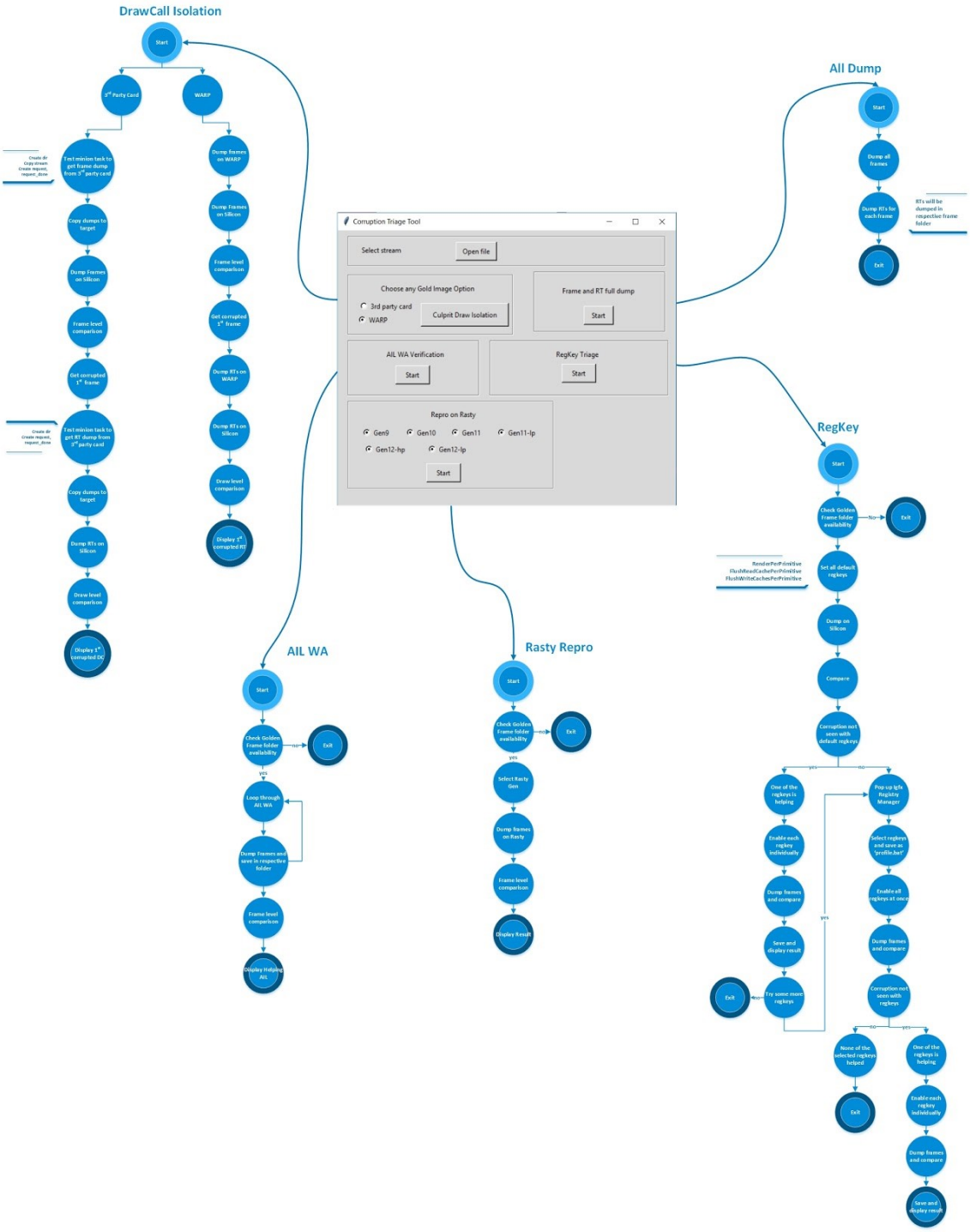

Figure 5.4: Flow chart of corruption automation tool

Modules:

1. **Select stream :** In this module user needs to give input as captured workload.
2. **Golden images options :** In this module selection of golden image platform is taken from user. Golden images are the images which without corruption.
3. **Frame and Render target dump :** If user want to dump only frames and render targets then this module will provide that facility.
4. **Regkey triage :** Registry keys can enable disable functionality for expected output. In this module we have 12 regkeys enable options by default.
5. **AIL WA :** This module basically enable/disable internal workaround of intel specific software.
6. **Rasty :** Again it is intel specific software on which we can reproduce the corruption platform wise.

Outcomes:

- Automating this process seems to be very useful for finding culprit draw for corruption.
- Prevents from hacker to enter the corruption state.
- Reduce the work from 21 days to 1-2 days.

Screenshot:

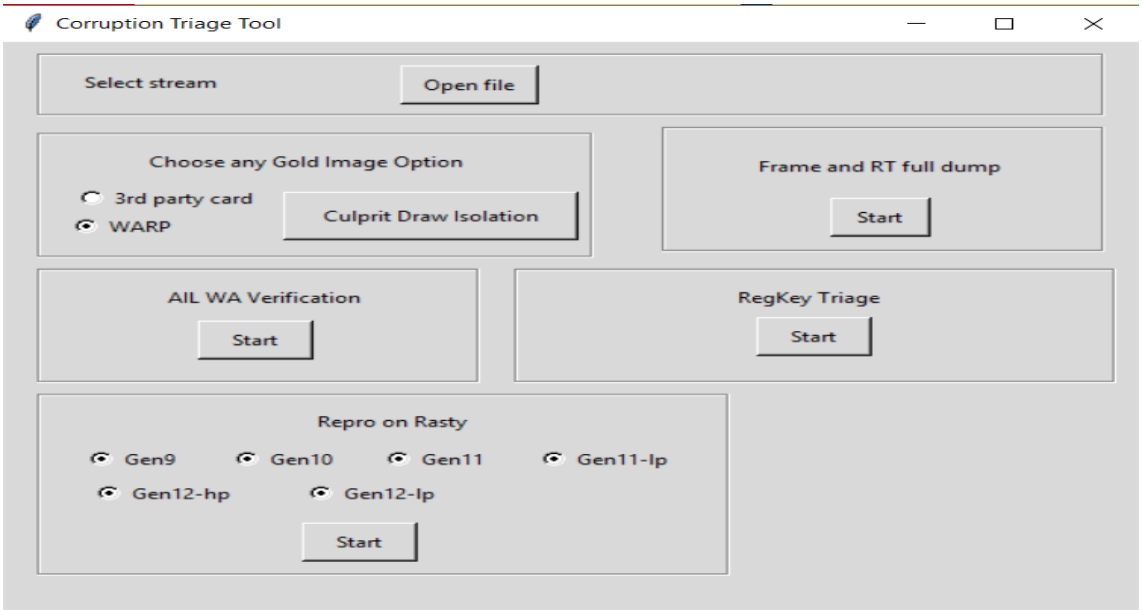

Figure 5.5: corruption automation tool

5.4 MMIO Blacklisted operation

- Memory mapped I/O is used for performing input /output methods on CPU.
- MMIO is most used component in hardware for read/write operations.
- There are lot of read/write operations happening in MMIO but which operations are authentic and which are not there is no mechanism to find that out.
- There are two kind of MMIO 1)whitelisted 2)blacklisted.
- as per the names whitelisted registers are those on which write operations are allowed and blacklisted are dangers ones.
- It is very important to stop access of blacklisted registers for security reasons.
- If hacker can get access of blacklisted registers , access violation of entire software as well as hardware can happen.
- Blacklisted register access can lead hardware and software to exploit
- If attacker is able to find out blacklisted register , then he can attack on that register.

- He can try to read/write register and based on information available, he can launch attack.
- It can be hardware damage also.
- Assembled all MMIO access happening from 3D side.

After this experiment found some blacklisted register access happening from 3D graphics driver. So it was essential to stop that access to reduce the security risk. Finding each and every access is very time consuming and very difficult.

So it was very necessary to find all access and stop them and doing it manually was not a doable task. It was clear security bug which needs to be taken care of.

Blacklisted	**Whitelisted**
No direct access	Direct access
Only read access	Read / write access
Harmful for hardware	No harm
If software is accessing it, then it creates entrance for hackers	Accessing it is fine until and unless its unauthorized access.

MMIO Script

- Implemented script to provide security and interrupt hardware and software from Accessing blacklisted MMIO.
- From next gen onwards whenever any component will try to access blacklisted MMIO because of this change in code it will throw a warning and will prevent from the attack.
- It was a clear security vulnerability which was not exposed before.

5.5 Hardware Security

Security is essential in both hardware and software. When we are referring to microprocessor chip vendors like INTEL AMD and discrete graphics card like NVIDIA follows

different security patterns.so there is no best and waste kind of thing while it comes to security techniques there are different techniques for software and hardware. you can understand every technique theoretically, but which technique is best for us we can only come to know after applying all the techniques and then doing comparison between them so, that we can find out which technique is giving more accuracy and more efficient output in this case.

A processor runs on a hardware with all operations managed by Operating system.so kernel mode is the most trusted mode of any particular OS, it will do very trusted operations from OS side.so basically it is very low-level structure to control connected devices. so, kernel mode are basically the control point or we can say interface between hardware and software, they can completely stop running your PC as well as they can release your halted PC.

Every data has attributes and instances with respect to subject or requirement of that hardware components. Every bit of information will require some component in hardware to store and process the data. Every register can process limited data at a time. We can select that attributes and make our work easy. For finding principal components we need to do some calculations . For that calculations firstly, we need to assign some virtual address to the input so mapping between virtual address and physical address will take place to render content on device.

Any circumstance where subjective client controlled information is handled by believed code exhibits the likelihood of the information being contorted with a certain goal in mind to such an extent that the code carries on in manners that were not planned (in qualification to compositional blemishes). Any administrator mode interface presented to client mode must have vigorous info approval.

Past that, piece modules are standard PE records with in Import Directory (on the off chance that they import any images). Despite the fact that drivers commonly just connection with the NT portion, or with win32k or videoprt for showcase drivers and their miniports, the NT bit bolsters 'piece mode DLLs' all the more formally alluded to

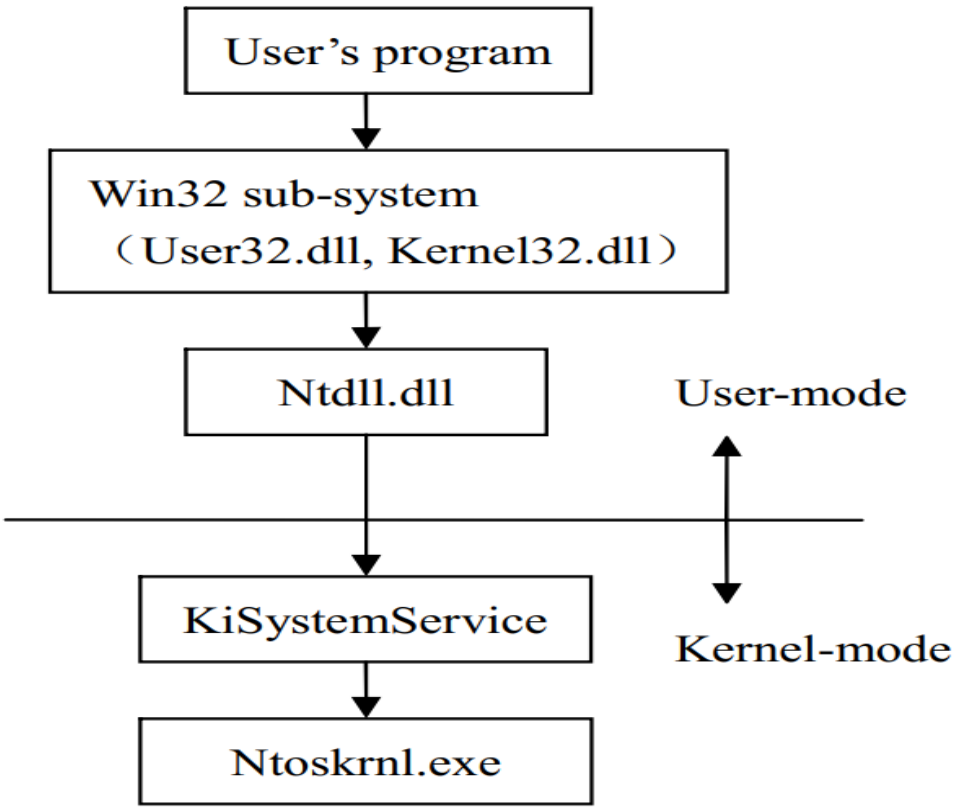

Figure 5.6: flow in system services

as fare drivers. Ordinary drivers can be statically connected to send out drivers, and the conditions of any driver are recursively settled amid the picture stacking process. The section purpose of a fare driver isn't executed, and all things considered it doesn't have to contain any code and is just required as the assemble procedure necessitates that capacity to be available. In any case, on the off chance that it sends out DllInitialize, at that point that will be called to permit whatever introduction the driver requires to be completed. The empty routine is DllUnload.[1]

The plan of such an apparatus raises significant issues on the off chance that all potential information is given in the meantime, at that point on a fundamental level each restrictive bounce might possibly be taken (however code could be composed whereby explicit contingent hops are never taken). To boost code inclusion, the state at each contingent hop would should be put away and if important connected to a parent state (in the event that it is in a subfunction, for instance). For a modestly estimated bit of code, the measure of room required to store these states would all around rapidly turned

out to be substantial.

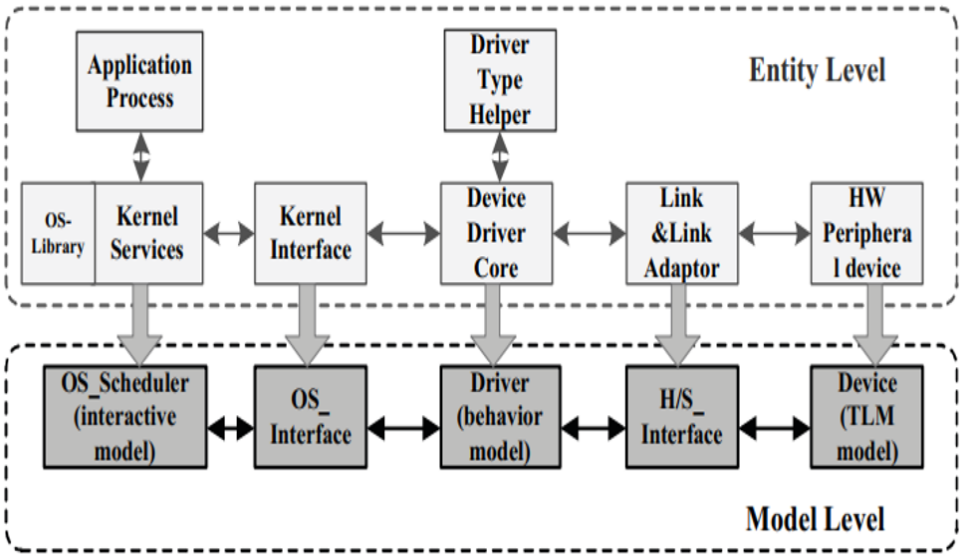

Figure 5.7: Integrated secure model for driver

Application process interacts with OS library. Os calls with pfn function to kernel. Kernel services also interacts with kernel interface. OS interface calls kernel interface to access some memory.so there this driver behavior model that will create device and it will become model to help driver type.[2]

So whenever there is any application process is running some part of memory will be used from RAM. For accessing that memory driver needs to interact with hardware surface to do write and read operations.

Chapter 6

Conclusion

Providing security in user mode driver is very important for future because number of vendors are increasing now a days and hence change in hardware architecture will take place .so providing low level security is also essential. Providing hardware security is not easy task because different content use different components .

Kernel is the module which directly talks to hardware as well as software.so kernel is the key player here, which is must implemented with security. Kernel is very low level, so programming is also in assembly languages like C, C++. Which requires understanding of hardware memory allocation.

We can use both modules to provide security so that we can close as many doors as possible for hackers. If we are providing security separately in different modules it will be very difficult for anyone to exploit. That is like providing highest security from low level but still no one has control over hardware and memory so future challenge is how to take complete control of hardware so that no one can access single bit of unauthenticated information.

In this term attempt to provide security using code security, hardware security and using automation tool is described. This can play role in fault detection, fault prevention and exploit prevention. So final outcome for this module is now hardware and software is protected before and also automation can reduce manual effort and prevent hackers to enter hardware and software for 3D graphics.

Bibliography

[1] "Techniques of user-mode detecting system service descriptor table r.,"

[2] "Proceedings of the 2009 13th international conference on computer supported cooperative work in design.,"