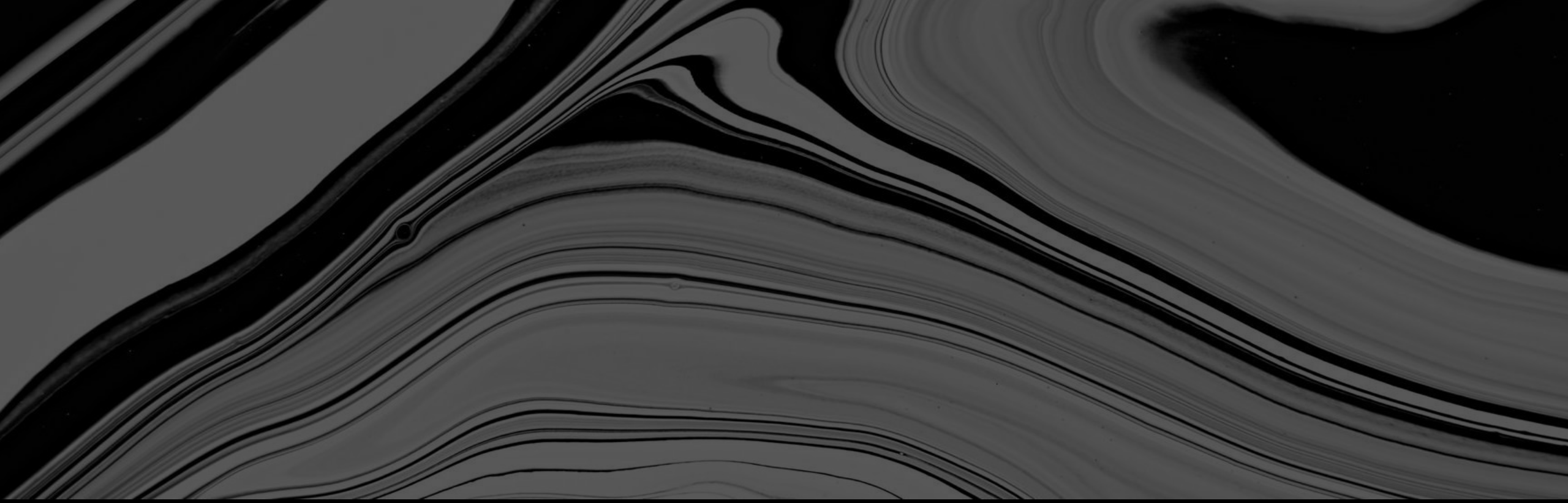
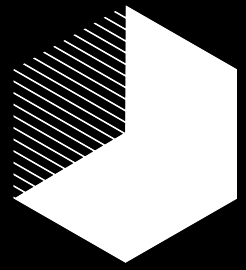
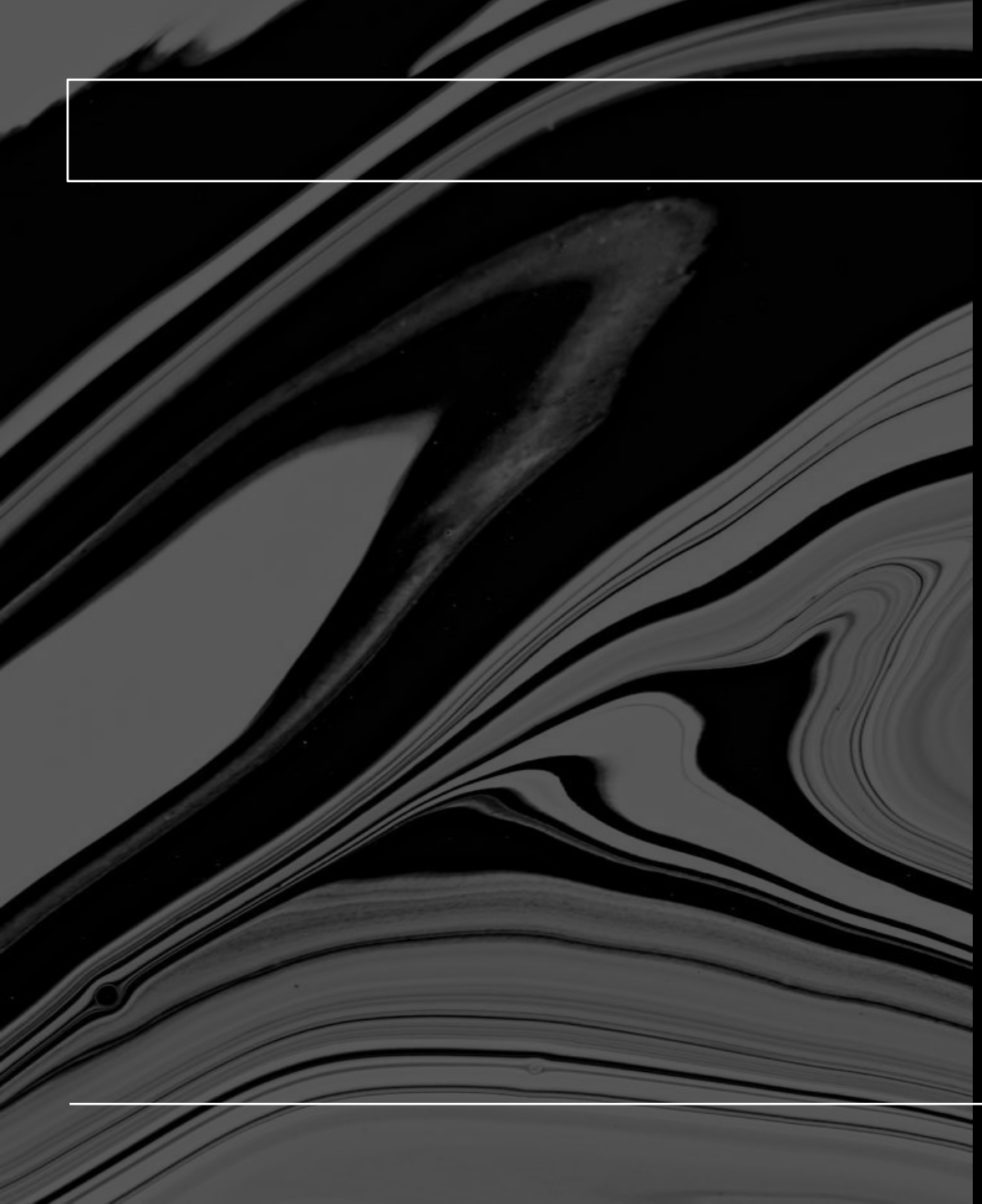




NOT FAIRFIT: BYPASSING ANTI-CHEAT WITH DIRECT MEMORY ACCESS



WHOAMI

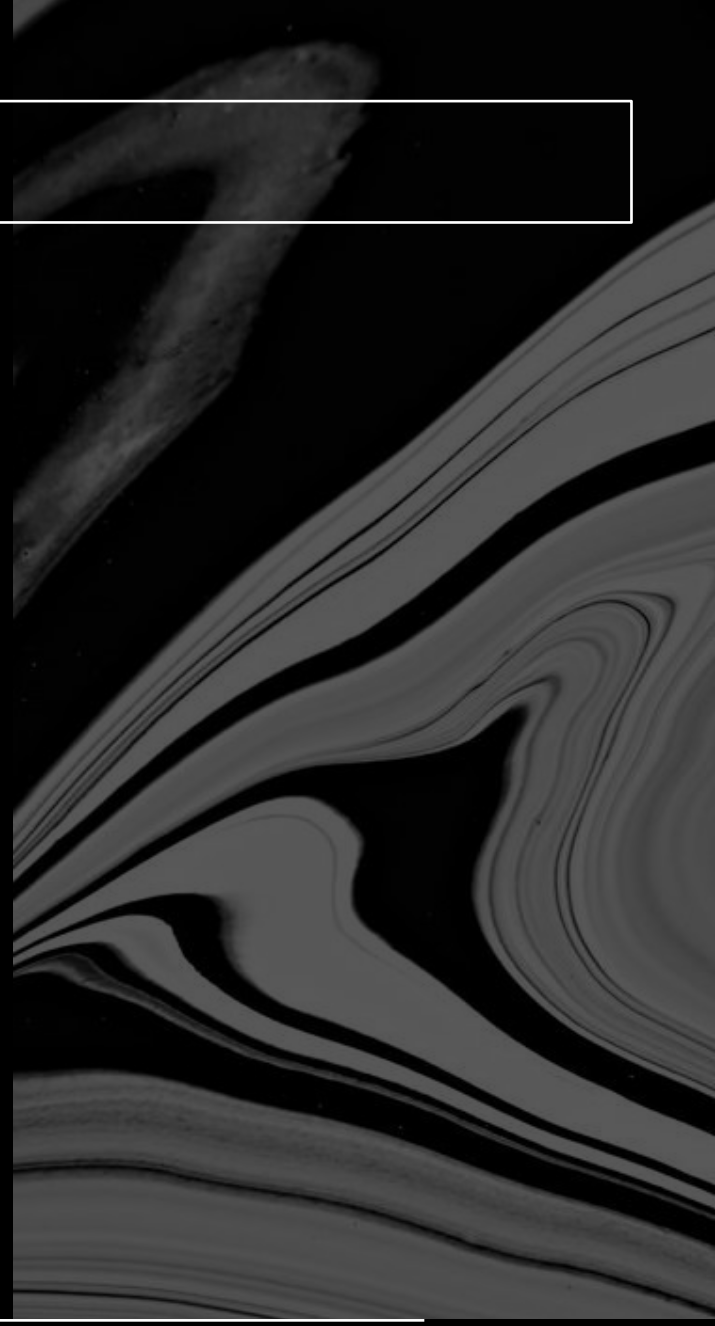
Connor Kastner | GPEN, GSOC

SOC Analyst @ ATC

Big fan of anything involving screens

OVERVIEW

- What is Anti-Cheat (AC)?
- Historical AC Overview
- What is Direct Memory Access (DMA) & PCILeech
- Building Firmware With PCILeech-FPGA
- External Cheats & DMALibrary
- AC's Methodology For Detecting DMA



WHAT IS ANTI-CHEAT (AC)?

- Software designed to detect and prevent players in online games from using tools or methods to gain an unfair advantage
- This can be accomplished through signature, heuristic, and reputation-based detection

LOW TIER ANTI-CHEATS (AC)

PunkBuster (2000-2015)

- User-level application
- Primarily used signature-based detection
- Simple and non-complex

Valve Anti-Cheat (2002-Present)

- User-level application
- Began with signature-based detection
- Incorporates a machine-learning approach to achieve heuristic detection
- Still behind competitors

MID TIER ANTI-CHEATS (AC)

Easy Anti-Cheat (2006-Present)

- Released in 2006 by Kamu
- Acquired by Epic Games in 2018
- Kernel-level application
- Decent signature & heuristic-based detection methods

BattlEye (2006-Present)

- Kernel-level application
- Utilizes signature & heuristic-based detection methods

BATTLEYE

THE ANTI-CHEAT GOLD STANDARD

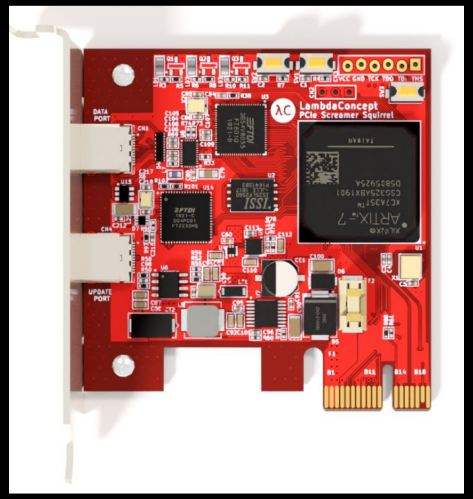
RIOT VANGUARD (2020-PRESENT)

- Released in 2020
- Kernel-level application
- Advanced signature & heuristic-based detection
- One of the strongest ACs on the market

WHAT IS DIRECT MEMORY ACCESS (DMA) & PCILEECH?

Direct Memory Access (DMA)

- Attack allowing an actor to directly access memory independent of the CPU
- Involves using a second device to transfer the data from the victim machine to the attacker machine



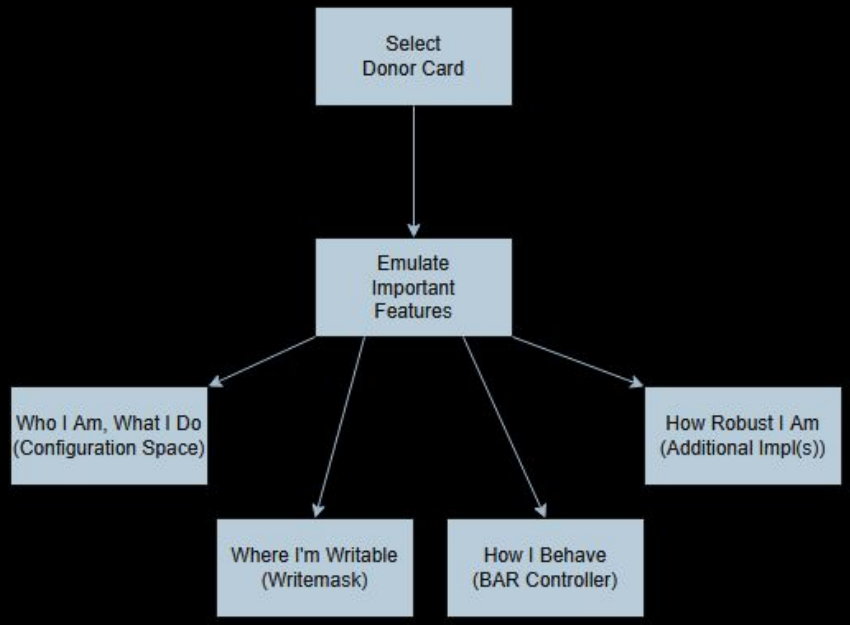
PCILeech & PCILeech-FPGA

- Tool developed by Ulf Frisk to use PCIe hardware to read & write target system memory
- Supports hardware & software-based memory acquisition

BUILDING FIRMWARE WITH PCILEECH-FPGA

Primary components

- Selecting a donor card
- Shadow Configuration Space
- Writemask
- Base Address Register (BAR) Controller
- Additional implementation(s):
  - Interrupt handling, Link state emulation, etc.



SELECTING A DONOR CARD

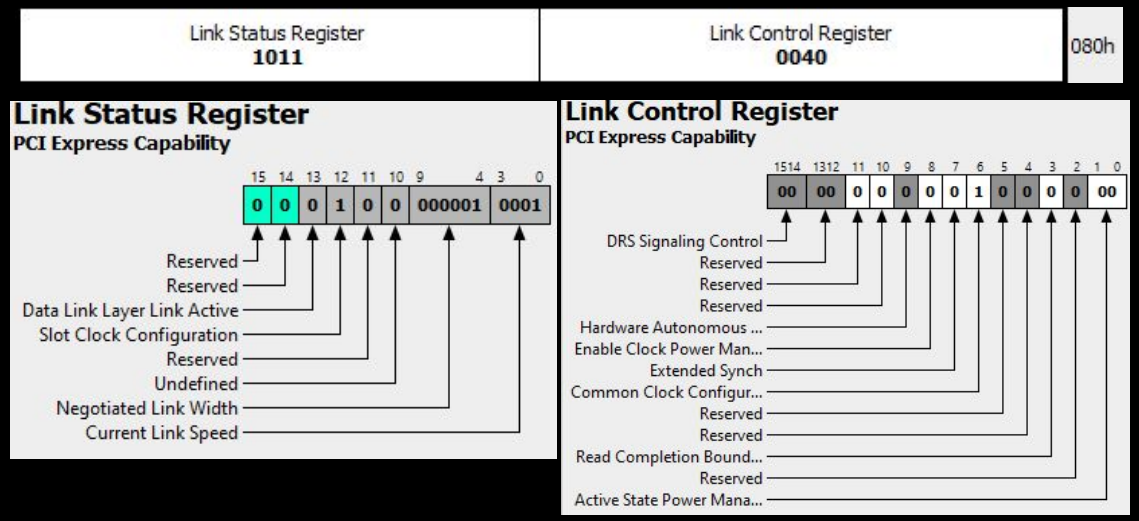
- Research its functionality and emulate it on the DMA card to evade detection
- Any PCIe card is viable (some harder than others)
- I used a Ralink RT5392 Wi-Fi Card

SHADOW CONFIGURATION SPACE

Device ID **5392**		Vendor ID **1814**		000h
Status **0010**		Command **0406**		004h
Class Code **028000**			Revision ID **00**	008h
BIST **00**	Header Type **00**	Latency Timer **00**	Cache Line Size **10**	00Ch
Base Address Register 0 **F78F0000**				010h
Base Address Register 1 **00000000**				014h
Base Address Register 2 **00000000**				018h
Base Address Register 3 **00000000**				01Ch
Base Address Register 4 **00000000**				020h
Base Address Register 5 **00000000**				024h
Cardbus CIS Pointer **00000000**				028h
Subsystem ID **5392**		Subsystem Vendor ID **1814**		02Ch
Expansion ROM Base Address **00000000**				030h
Reserved **000000**			Cap_Ptr **40**	034h
Reserved **00000000**				038h
Max_Lat **00**	Min_Gnt **00**	Int_Pin **01**	Int_Line **00**	03Ch
PMC **01C3**		Next_Item_Ptr **50**	Cap_ID **01**	040h
Data **00**	PMCSR **000000**			044h

- **Configuration Space (CFG)**: Defines who this card is and what its capabilities are
- **Shadow CFG**: Allows us to override the standard CFG settings by masking the core

WRITEMASK



- Creates & defines a permission map that tells PCIe devices which bits in its CFG space are writable
- Permission Bit Types:
  - RW – Read/Write // Set to **1**
  - RO – Read Only // Set to **0**
  - RSVDZ – Reserved // Set to **0**
  - RW1CS – Read/Write 1 to Clear Sticky // Set to **0**
  - HWINIT – Hardware Initialized // Set to **0**

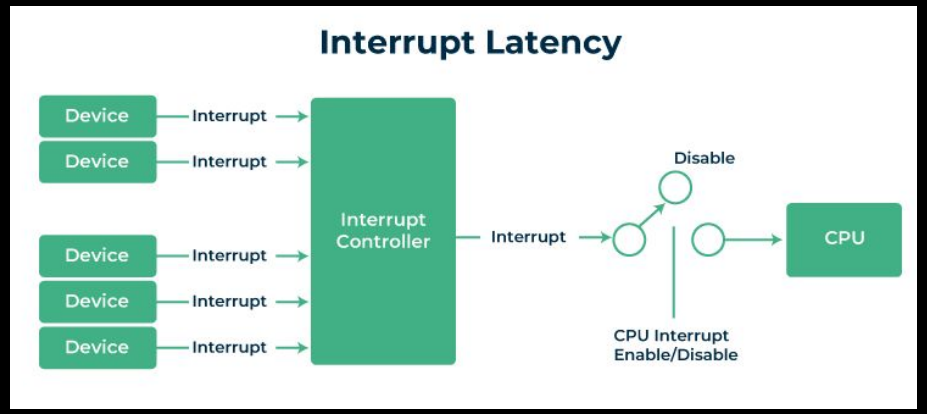
BASE ADDRESS REGISTER (BAR) CONTROLLER

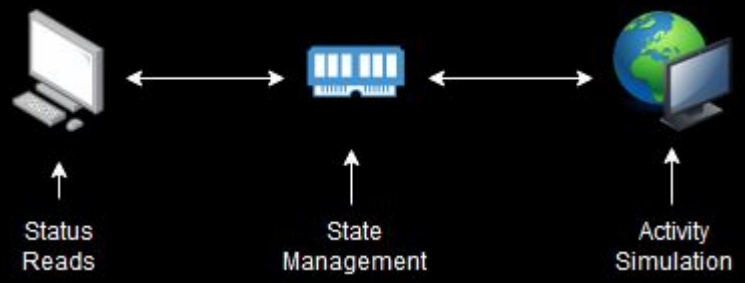
```
always @ (posedge clk) begin
    if (rst)
        number <= 0;

    number          <= number + 1;
    drd_req_ctx     <= rd_req_ctx;
    drd_req_valid   <= rd_req_valid;
    dwr_valid       <= wr_valid;
    drd_req_addr    <= rd_req_addr;
    rd_rsp_ctx      <= drd_req_ctx;
    rd_rsp_valid    <= drd_req_valid;
    dwr_addr        <= wr_addr;
    dwr_data        <= wr_data;

if (drd_req_valid) begin
    case (({drd_req_addr[31:24], drd_req_addr[23:16], drd_req_addr[15:08], drd_req_addr[07:00]} - (base_address_register & ~32'h4)) & 32'hFFFF)
        16'h0598 : begin                        // MAC_ADDR_HIGH
            rd_rsp_data[7:0]   <= 8'h00;        // Vendor prefix
            rd_rsp_data[15:8]  <= 8'h1A;        // Vendor prefix
            rd_rsp_data[23:16] <= 8'hEF;        // Vendor prefix
            rd_rsp_data[31:24] <= ((0 + (number) % (15 + 1 - 0)) << 4) | (0 + (number + 3) % (15 + 1 - 0));
        end
        16'h0594 : begin                         // MAC_ADDR_LOW
            rd_rsp_data[7:0]   <= ((0 + (number + 6) % (15 + 1 - 0)) << 4) | (0 + (number + 9) % (15 + 1 - 0));
            rd_rsp_data[15:8]  <= ((0 + (number + 12) % (15 + 1 - 0)) << 4) | (0 + (number + 15) % (15 + 1 - 0));
            rd_rsp_data[31:16] <= 16'h0000;
        end
        16'h0000 : rd_rsp_data <= 32'h00001C00;  // SYS_CTRL - System control register
        16'h0004 : rd_rsp_data <= 32'h00000010;  // SYS_CFG - System configuration
        16'h0108 : rd_rsp_data <= 32'h14010000;  // PCI_CFG - PCI configuration register
        16'h010C : rd_rsp_data <= 32'h00004C02;  // PCI_STATUS - PCI status register
        16'h0110 : rd_rsp_data <= 32'h00001008;  // PCI_CTRL - PCI control register
```

```
[Host System] <---> [BAR Controller] <---> [Device Memory/Registers]
      |                      |                         |
PCI Requests        Address Translation       Local Resources
```

- A BAR Controller is used to manage read & write access to the device's memory-mapped regions
- It allows us to configure our DMA card to behave and respond similar to how our donor card would

ADDITIONAL IMPLEMENTATIONS





Interrupt Handling

- Interrupts are used to notify the CPU that an event has occurred
- Implementation of the donor card's sideband signal (INTA#, INTB#, etc.) to send the interrupt request

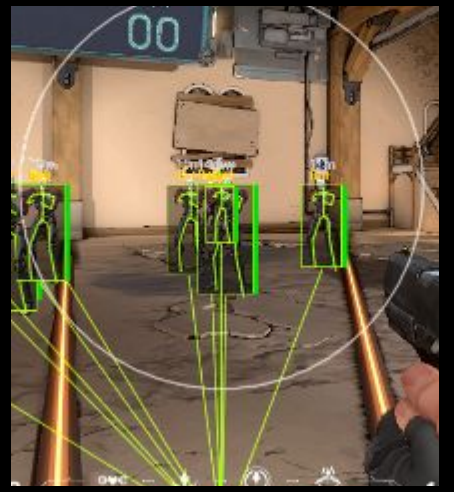
Link State Emulation

- Allows us to simulate the activity that our donor card would perform (active network connection, reading from a USB, etc.)
- Implementation of proper link states, registers, and traffic patterns

EXTERNAL CHEATS & DMALIBRARY

External Cheats

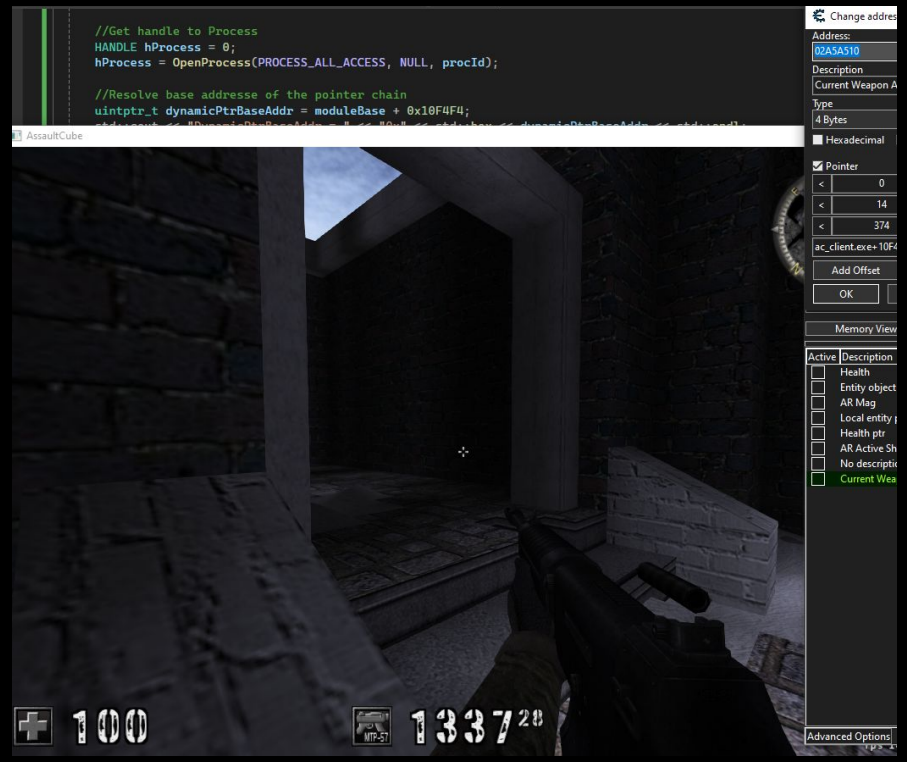
- A separate executable which includes logic to read and write to the memory of another process
- Core C++ Functions For External Cheats:
    - OpenProcess, ReadProcessMemory, WriteProcessMemory, and VirtualProtectEx (Gives permission to modify memory page)



DMALibrary

- Publicly available C++ library to implement into your cheat to enable your DMA card to read and process memory from your target game
- Uses PCILeech components (FTD3XX.dll, LeechCore.dll, VMM.dll)

ANATOMY OF CHEAT DEVELOPMENT



- Locating the process and its module base address
- Dynamically finding offsets within the memory region
- Resolving pointer chains to important objects and manipulating them
- Decrypting pointers through reverse engineering (VGK)

DMALIBRARY IN ACTION

```
unsigned char abort2[4] = {0x10, 0x00, 0x10, 0x00};

bool Memory::SetFPGA()
{
    ULONG64 qwID = 0, qwVersionMajor = 0, qwVersionMinor = 0;
    if (!VMMDLL_ConfigGet(this->vHandle, LC_OPT_FPGA_FPGA_ID, &qwID) && VMMDLL_ConfigGet
    (this->vHandle, LC_OPT_FPGA_VERSION_MAJOR, &qwVersionMajor) && VMMDLL_ConfigGet
    (this->vHandle, LC_OPT_FPGA_VERSION_MINOR, &qwVersionMinor))
    {
        LOG("[!] Failed to lookup FPGA device, Attempting to proceed\n\n");
        return false;
    }

    if ((qwVersionMajor >= 4) && ((qwVersionMajor >= 5) || (qwVersionMinor >= 7)))
    {
        HANDLE handle;
        LC_CONFIG config = {.dwVersion = LC_CONFIG_VERSION, .szDevice = "existing"};
        handle = LcCreate(&config);
        if (!handle)
        {
            LOG("[!] Failed to create FPGA device\n");
            return false;
        }

        LcCommand(handle, LC_CMD_FPGA_CFGREGPCIE_MARKWR | 0x002, 4, reinterpret_cast<PBYTE>
        (&abort2), NULL, NULL);
        LOG("[-] Register auto cleared\n");
        LcClose(handle);
    }

    return true;
}
```

- Load the three primary libraries from PCILeech (FTD3XX.dll, LeechCore.dll, VMM.dll)
- Create a handle to the existing FPGA/DMA device
- Initializes and sets required registers for the DMA card

BYPASSING POINTER ENCRYPTION (VGK)

```
// Credit to Jouh on UnKnoWnCheaTs for source code
__forceinline __int64 Decrypt_UWorld(const uint32_t key, const uintptr_t* state)
{
    unsigned __int64 v19; // R11
    unsigned __int64 v20; // R8
    unsigned __int64 v21; // R9
    unsigned int v22; // ER10
    unsigned __int64 v23; // RCX
    unsigned __int64 v24; // RDX
    unsigned __int64 v25; // RCX
    int v26; // EBX
    unsigned int v27; // ECX
    __int64 v28; // RAX
    unsigned __int64 v29; // R8
    unsigned __int64 v30; // R8
    unsigned __int64 v31; // RCX
    unsigned __int64 v32; // RDX
    unsigned __int64 v33; // RCX

    v19 = 268582165773633871 7i64
        * ((unsigned int)key ^ (unsigned int)(key << 25) ^ (((unsigned int)key ^ ((unsigned __int64)(unsigned int)key >> 15)) >> 12))
        % 7;
    v20 = state[v19];
    v21 = (268582165773633871 7i64
        * ((unsigned int)key ^ (unsigned int)(key << 25) ^ (((unsigned int)key ^ ((unsigned __int64)(unsigned int)key >> 15)) >> 12))) >> 32;
    v22 = (unsigned int)v19 % 7;
Vector2 w2s(const Camera& camera, Vector3 world_location) {
    XMMATRIX temp_matrix = to_matrix(camera.Rotation);

    XMVECTOR vaxisx = temp_matrix.r[0];
    XMVECTOR vaxisy = temp_matrix.r[1];
    XMVECTOR vaxisz = temp_matrix.r[2];

    Vector3 vdelta = world_location - camera.Location;
    XMVECTOR vdelta_vec = XMVectorSet(vdelta.x, vdelta.y, vdelta.z, 0.0f);

    float vtransformed_x = XMVectorGetX(XMVector3Dot(vaxisy, vdelta_vec));
    float vtransformed_y = XMVectorGetX(XMVector3Dot(vaxisz, vdelta_vec));
    float vtransformed_z = XMVectorGetX(XMVector3Dot(vaxisx, vdelta_vec));

    if (vtransformed_z < 1.0f) vtransformed_z = 1.0f;

    int screen_center_x = static_cast<int>(settings::width) / 2;
    int screen_center_y = static_cast<int>(settings::height) / 2;

    float fov = camera.FieldOfView;
    float tan_fov = tanf(fov * static_cast<float>(XM_PI) / 360.0f);

    float screen_x = screen_center_x + (vtransformed_x / tan_fov) * screen_center_x / vtransformed_z;
    float screen_y = screen_center_y - (vtransformed_y / tan_fov) * screen_center_x / vtransformed_z;

    return { screen_x, screen_y };
}
```

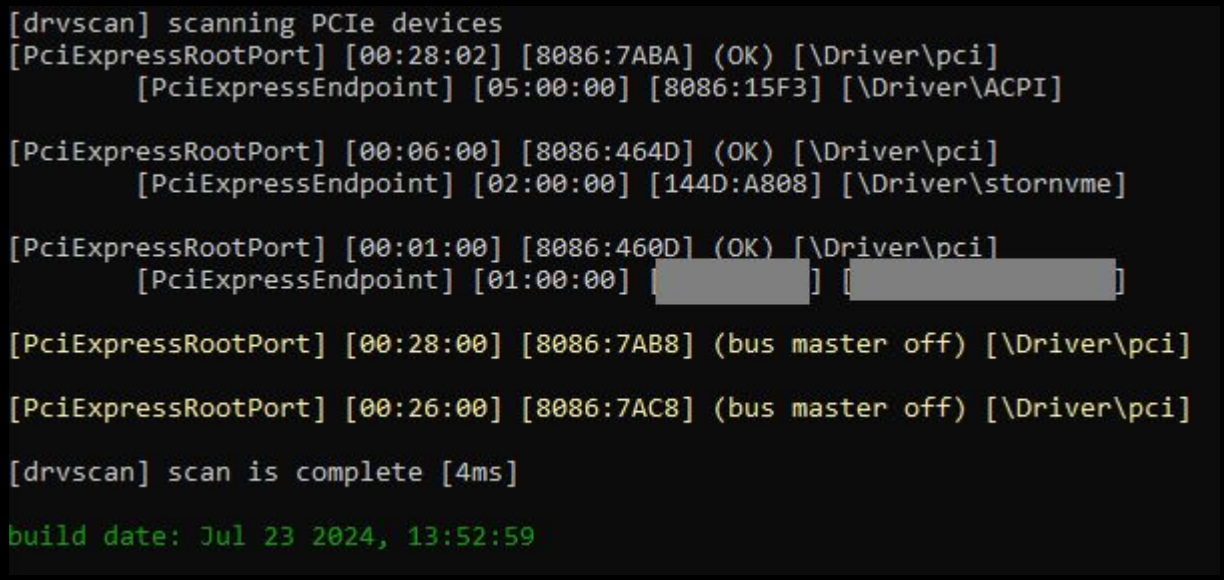
- Valorant utilizes encryption to protect its critical pointers (Uworld, Actor Health, etc.)
- Use DMALibrary to read these values directly from physical memory
- Implement a reverse-engineered decryption algorithm to reveal true memory location(s)
- Once values are retrieved, use DirectX & W2S to outline actors

DMA IN ACTION

Editor Note: There was a video demo here, please reach out if you'd like to see it

AC'S METHODOLOGY FOR DETECTING DMA

- How the hell do you detect any of this?
- Human error is the downfall in many cases
- Illegal devices (devices that don't follow industry standards) are blocked and flagged



THANK YOU!

Questions?

Contact:

- LinkedIn: Connor Kastner (/in/connorkas)
- https://ret2c[.]com
  - LinkedIn, GitHub, Twitter, Signal

Thank You:

- Simonrak
- Desilvered
- Dzul
- Noah Tongate (Ap3x)
- Anonymous Contributor(s)

Resources:

- Simonrak[.]se
- DMA-CFW-Guide by Desilvered on GitHub
- OpenSecurityTraining[.]info
- Getting Started with FPGAs: Digital Circuit Design, Verilog, and VHDL for Beginners
- GitHub Repositories of PCILeech firmware