Módulo de captura, filtragem e redirecionamento de mensagens DirectX

Giovani Chaves

Orientador: Mauro Marcelo Mattos

Sumário

- Introdução
- Objetivos
- DirectX
- DLL Injection
- APIHijack
- Desenvolvimento
- Conclusão
- Extensão

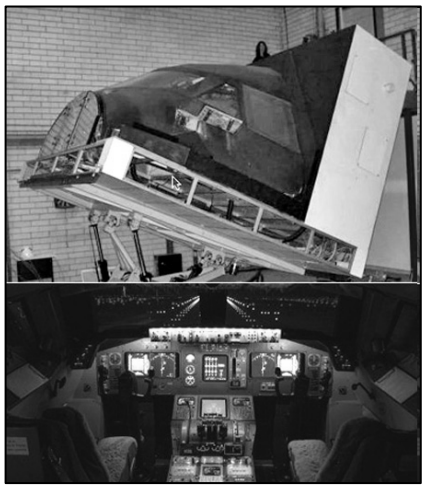
FURB
UNIVERSIDADE DE BLUMENAU

Introdução

- **O principal objetivo do desenvolvimento de gráficos cada vez mais realistas em jogos é proporcionar ao jogador uma maior imersão, dando a impressão de que ele realmente está vivendo a situação apresentada pela problemática do game.**
- **Porém, não apenas gráficos extremamente detalhados e sons de ambientação são recursos utilizados para proporcionar a sensação de se estar dentro do game.**

Introdução

- **Existem cabines e simuladores que abrigam o jogador, para que se sinta de fato pilotando um avião, nave ou carro.**
- **Ao se movimentar, estes equipamentos interagem com outros sentidos do jogador (equilíbrio, tato), resultando em uma resposta mais rápida no reconhecimento do cenário à sua volta.**



Introdução

- **Existem cabines e simuladores que abrigam o jogador, para que se sinta de fato pilotando um**

ário

Introdução

- Existe
  jogad
  avião,
  equipa
  jogad
  respos
  à sua

am o
do um
estes
tidos do
ma
do cenário

Introdução

- **Todavia, há acessórios mais simples e portáteis (como joysticks e volantes que reagem a determinadas ações do jogo oferecendo resistência ao movimento - o chamado Force Feedback, óculos estereoscópicos para simular 3D, pistolas e revólveres para jogos de tiro, entre outros), que são amplamente utilizados por jogadores.**

Introdução

- **Todavia, há acessórios mais simples e portáteis (como joysticks e volantes que reagem a determina lo resistência ao movim lback, óculos estereosc las e revólveres tros), que são amplamer**

Introdução

- **Todavia, há acessórios mais simples e portáteis (como joysticks, e volantes que reagem a deter resistência ao mo ck, óculos estere s e revólv s), que são ampla**

Introdução

- **Todavia, há acessórios mais simples e portáteis (como joysticks e volantes que reagem a deter esistência ao m k, óculos ester e revól), que são ampl**

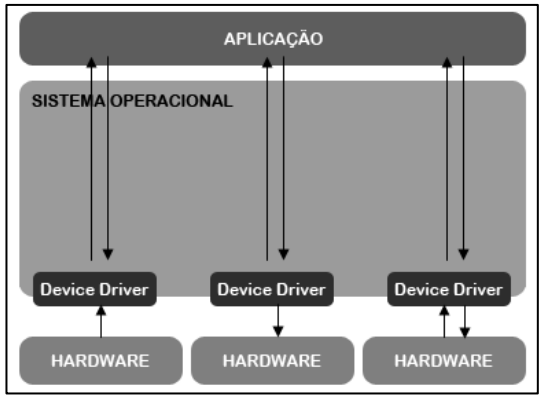
Objetivos

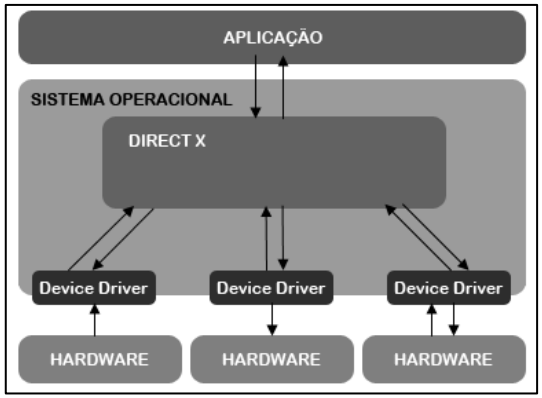
- **O presente trabalho tem como objetivo desenvolver um módulo de captura e filtragem de mensagens DirectX geradas durante a execução de games que se comunicam com dispositivos Human Interface Devices (como mouses e joysticks); e efetuar o redirecionamento destas mensagens.**
- **Os objetivos específicos do trabalho são:**
  - disponibilizar um módulo de interceptação de mensagens do DirectX através de DLL injection e filtragem dos eventos que envolvem Force Feedback no módulo DirectInput;
  - disponibilizar um aplicativo que permita a instalação e remoção do Hook que fará a injeção da DLL customizada no processo alvo.

FURB
UNIVERSIDADE DE BLUMENAU

DirectX

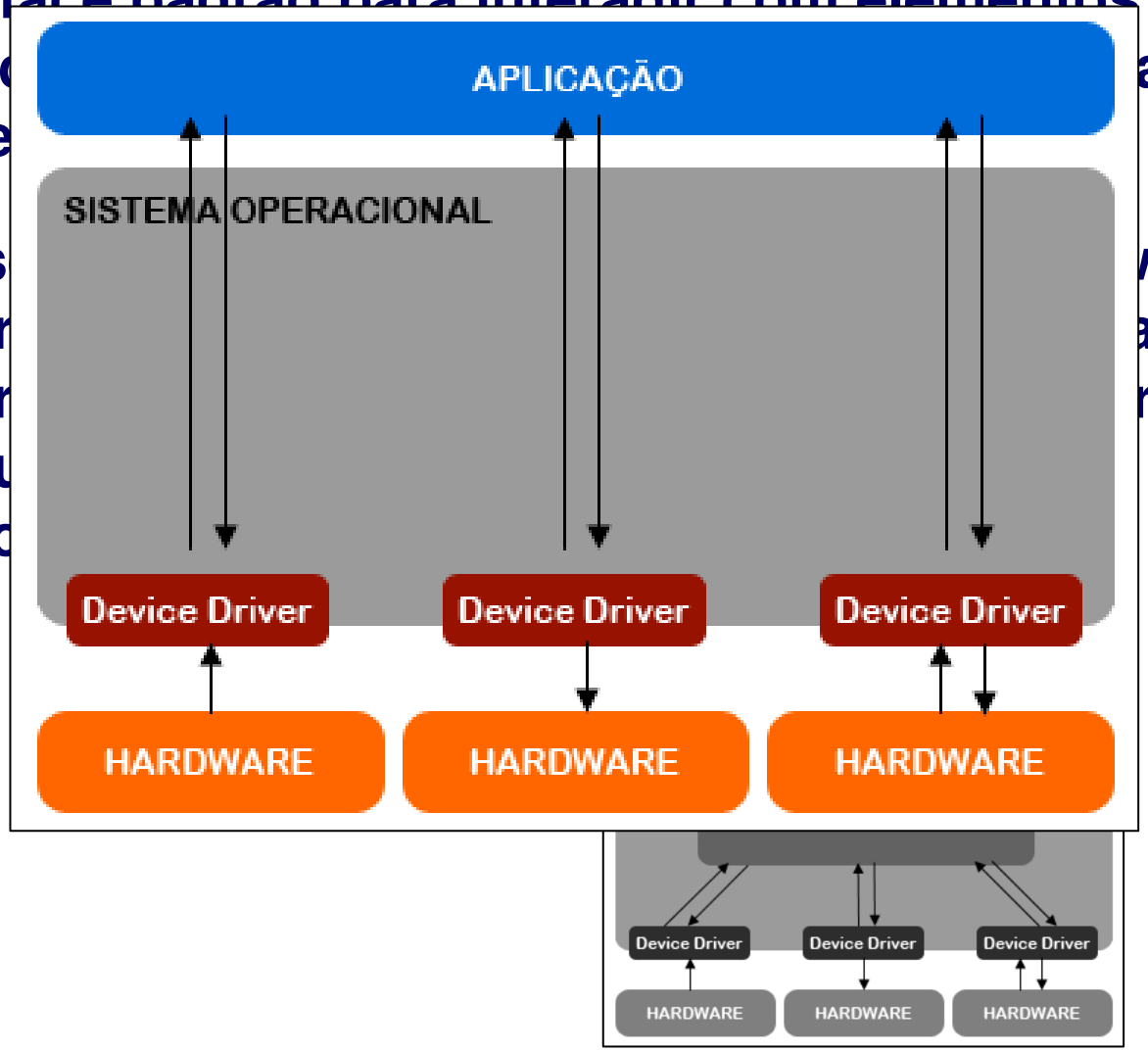
- **O DirectX é uma API multimídia que oferece uma interface padrão para interagir com elementos gráficos, placas de som e dispositivos de entrada, entre outros.**
- **Ele serve como um intermediário entre o software e o hardware, para que os programadores possam abstrair as interações específicas com os diferentes conjuntos de hardware instalados na máquina (placa de som, mouse, teclado, etc).**





DirectX

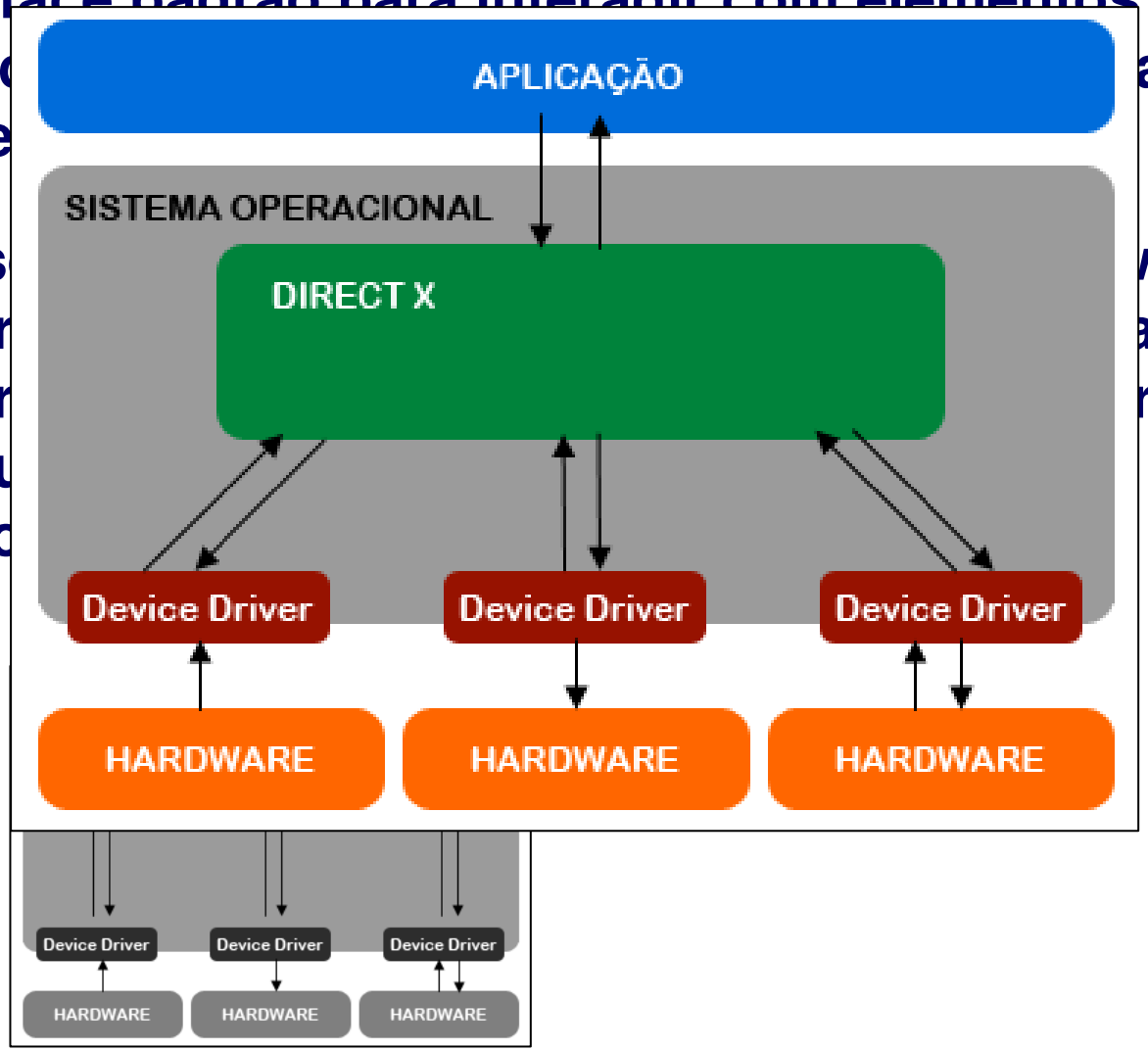
- O DirectX é uma API multimídia que oferece uma interface padrão para interagir com elementos gráfi ada, entre
- Ele s vare e o har am abstr rentes conju (placa de so



FURB
UNIVERSIDADE DE BLUMENAU

DirectX

- O DirectX é uma API multimídia que oferece uma interface padrão para interagir com elementos gráfi ada, entre
- Ele s vare e o har am abstr rentes conju (placa de so



FURB
UNIVERSIDADE DE BLUMENAU

DirectX

- **O DirectX é constituído por diversos módulos com áreas bem definidas, entre eles:**
  - DirectDraw (gráficos 2D)
  - Direct3D (gráficos 3D)
  - DirectSound (entrada e saída de som)
  - DirectMusic (biblioteca de efeitos sonoros)
  - DirectInput (joysticks)
  - DirectPlay (jogos em rede)
  - DirectShow (manipulação de vídeo)

DirectX

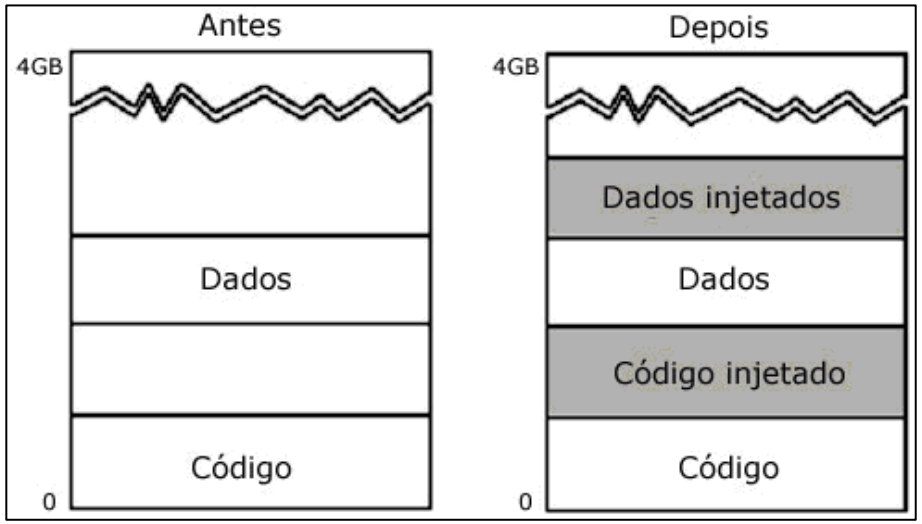
- O módulo DirectInput possui métodos de obtenção dos comandos de controladores de jogo (device objects), bem como o controle da interatividade da tecnologia Force Feedback, que nada mais é do que uma interface de interação háptica com o jogador.
- O DirectInput trabalha diretamente com os device drivers dos dispositivos, criando uma nova camada de abstração entre o dispositivo e o aplicativo.
- Qualquer hardware que não for categorizado como mouse ou teclado, é considerado um joystick.

DLL Injection

- A técnica de injeção de código visa entrar no fluxo normal de execução de um software e introduzir uma rotina de código externa, como se fosse parte do aplicativo original.
- Este procedimento tem vários propósitos: desde adicionar novas funcionalidades ao aplicativo alvo, recolher e analisar dados do processamento, até a disseminação de malwares.

DLL Injection

- **Duas formas de injetar código:**
  - DLL injection
    - Registro do Windows
    - Referenciar DLL em memória
  - Memory injection



APIHijack

- **Wade Brainerd desenvolveu uma API para substituir funções de uma DLL carregada na memória por funções customizadas de uma DLL injetada no processo alvo através de um Hook global do tipo WH_CBT.**
- **Estabelecido o Hook, a DLL customizada passa a ser carregada na memória de todos os processos à medida que forem ocorrendo mensagens relacionadas à CBT. Assim que a DLL é carregada, o método DllMain é executado.**

```
SDLLHook D3DHook =
{
    "DDRAW.DLL",
    false, NULL,    // Default hook disabled, NULL function pointer.
    {
        { "DirectDrawCreate", MyDirectDrawCreate },
        { NULL, NULL }
    }
};
```

```
BOOL APIENTRY DllMain( HINSTANCE hModule, DWORD fdwReason, LPVOID lpReserved)
{
    if ( fdwReason == DLL_PROCESS_ATTACH )  // When initializing....
    {
        hDLL = hModule;

        // We don't need thread notifications for what we're doing.  Thus,
        // get rid of them, thereby eliminating some of the overhead of
        // this DLL
        DisableThreadLibraryCalls( hModule );

        // Only hook the APIs if this is the right process.
        GetModuleFileName( GetModuleHandle( NULL ), Work, sizeof(Work) );
        PathStripPath( Work );

        if ( stricmp( Work, "myhooktarget.exe" ) == 0 )
            HookAPICalls( &D3DHook );
    }

    return TRUE;
}
```

FURB
UNIVERSIDADE DE BLUMENAU

APIHijack

- **Wade Brainerd desenvolveu uma API para substituir funções de uma DLL carregada na memória por funções customizadas de uma DLL injetada no processo alvo através de um Hook global do tipo**
- **ser relacionadas à CBT. Assim que a DLL é carregada, o método DllMain é executado.**

```
SDLLHook D3DHook =
{
    "DDRAW.DLL",
    false, NULL,    // Default hook disabled, NULL function pointer.
    {
        { "DirectDrawCreate", MyDirectDrawCreate },
        { NULL, NULL }
    }
};
```

```
BOOL APIENTRY DllMain( HINSTANCE hModule, DWORD fdwReason, LPVOID lpReserved)
{
    if ( fdwReason == DLL_PROCESS_ATTACH )  // When initializing....
    {
        hDLL = hModule;

        // We don't need thread notifications for what we're doing.  Thus,
        // get rid of them, thereby eliminating some of the overhead of
        // this DLL
        DisableThreadLibraryCalls( hModule );

        // Only hook the APIs if this is the right process.
        GetModuleFileName( GetModuleHandle( NULL ), Work, sizeof(Work) );
        PathStripPath( Work );

        if ( stricmp( Work, "myhooktarget.exe" ) == 0 )
            HookAPICalls( &D3DHook );
    }

    return TRUE;
}
```

FURB
UNIVERSIDADE DE BLUMENAU

APIHijack

- **Wade Brainerd desenvolveu uma API para substituir funções de uma DLL carregada na memória por**

```
BOOL APIENTRY DllMain( HINSTANCE hModule, DWORD fdwReason, LPVOID lpReserved)
{
    if ( fdwReason == DLL_PROCESS_ATTACH )  // When initializing....
    {
        hDLL = hModule;

        // We don't need thread notifications for what we're doing.  Thus,
        // get rid of them, thereby eliminating some of the overhead of
        // this DLL
        DisableThreadLibraryCalls( hModule );

        // Only hook the APIs if this is the right process.
        GetModuleFileName( GetModuleHandle( NULL ), Work, sizeof(Work) );
        PathStripPath( Work );

        if ( stricmp( Work, "myhooktarget.exe" ) == 0 )
            HookAPICalls( &D3DHook );
    }

    return TRUE;
}
```

```
    false, NULL,    // Default hook disabled, NULL function pointer.
    {
        { "DirectDrawCreate", MyDirectDrawCreate },
        { NULL, NULL }
    }
};
```

- **ser**

- **, o**

FURB
UNIVERSIDADE DE BLUMENAU

APIHijack

- **Após encontrar o executável que será injetado, o método HookAPICalls() é chamado para procurar a DLL.**

```
DWORD importRVA = pExeNTHdr->OptionalHeader.DataDirectory
                    [IMAGE_DIRECTORY_ENTRY_IMPORT].VirtualAddress;
if ( !importRVA )
    return false;

// Convert imports RVA to a usable pointer
PIMAGE_IMPORT_DESCRIPTOR pImportDesc = MakePtr( PIMAGE_IMPORT_DESCRIPTOR,
                                                hModEXE, importRVA );

// Save off imports address in a global for later use
g_pFirstImportDesc = pImportDesc;

// Iterate through each import descriptor, and redirect if appropriate
while ( pImportDesc->FirstThunk )
{
    PSTR pszImportModuleName = MakePtr( PSTR, hModEXE, pImportDesc->Name);

    if ( lstrcmpi( pszImportModuleName, Hook->Name ) == 0 )
    {
        OutputDebugString( "Found " );
        OutputDebugString( Hook->Name );
        OutputDebugString( "...\n" );

        RedirectIAT( Hook, pImportDesc, (PVOID)hModEXE );
    }

    pImportDesc++;  // Advance to next import descriptor
}
```

FURB
UNIVERSIDADE DE BLUMENAU

APIHijack

- **Assim que é encontrada a DLL que contém as funções a serem substituídas, o método RedirectIAT() é chamado para iterar entre as funções exportadas pela DLL.**

```
// These next few lines ensure that we'll be able to modify the IAT,
// which is often in a read-only section in the EXE.
DWORD flOldProtect, flNewProtect, flDontCare;
MEMORY_BASIC_INFORMATION mbi;

// Get the current protection attributes
VirtualQuery( pIAT, &mbi, sizeof(mbi) );

// remove ReadOnly and ExecuteRead attributes, add on ReadWrite flag
flNewProtect = mbi.Protect;
flNewProtect &= ~(PAGE_READONLY | PAGE_EXECUTE_READ);
flNewProtect |= (PAGE_READWRITE);

if ( !VirtualProtect(   pIAT, sizeof(PVOID) * cFuncs,
                        flNewProtect, &flOldProtect) )
{
    return false;
}
```

APIHijack


```
// Scan through the IAT, completing the stubs and redirecting the IAT
// entries to point to the stubs
pIteratingIAT = pIAT;

while ( pIteratingIAT->u1.Function )
{
    void* HookFn = 0;  // Set to either the SFunctionHook or pStubs.

    if ( !IMAGE_SNAP_BY_ORDINAL( pINT->u1.Ordinal ) )  // import by name
    {
        PIMAGE_IMPORT_BY_NAME pImportName = MakePtr( PIMAGE_IMPORT_BY_NAME
                                  , pBaseLoadAddr, pINT->u1.AddressOfData );

        // Iterate through the hook functions, searching for this import.
        SFunctionHook* FHook = DLLHook->Functions;
        while ( FHook->Name )
        {
            if ( lstrcmpi( FHook->Name, (char*)pImportName->Name ) == 0 )
            {
                OutputDebugString( "Hooked function: " );
                OutputDebugString( (char*)pImportName->Name );
                OutputDebugString( "\n" );

                // Save the old function in the SFunctionHook structure and get the new one.
                FHook->OrigFn = reinterpret_cast<void*>(pIteratingIAT->u1.Function);
                HookFn = FHook->HookFn;
                break;
            }

            FHook++;
        }
```

FURB
UNIVERSIDADE DE BLUMENAU

APIHijack

Gravando função customizada na IAT

```
// Replace the IAT function pointer if we have a hook.
if ( HookFn )
{
    // Cheez-o hack to see if what we're importing is code or data.
    // If it's code, we shouldn't be able to write to it
    if ( IsBadWritePtr( (PVOID)pIteratingIAT->u1.Function, 1 ) )
    {
        pIteratingIAT->u1.Function = reinterpret_cast<DWORD>(HookFn);
    }
    else if ( osvi.dwPlatformId == VER_PLATFORM_WIN32_WINDOWS )
    {
        // Special hack for Win9X, which builds stubs for imported
        // functions in system DLLs (Loaded above 2GB).  These stubs are
        // writeable, so we have to explicitly check for this case
        if ( pIteratingIAT->u1.Function > 0x80000000 )
            pIteratingIAT->u1.Function = reinterpret_cast<DWORD>(HookFn);
    }
}
```

Voltando IAT para somente leitura

```
// Put the page attributes back the way they were.
VirtualProtect( pIAT, sizeof(PVOID) * cFuncs, flOldProtect, &flDontCare);
```

FURB
UNIVERSIDADE DE BLUMENAU

Desenvolvimento

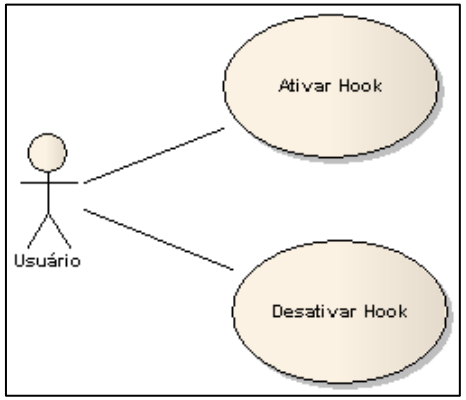
Especificação

- o aplicativo deverá interceptar, filtrar e identificar os eventos de Force Feedback do módulo DirectInput;
- deverá apresentar uma lista dos joysticks instalados e permitir que o usuário selecione para qual deles os eventos serão redirecionados;
- disponibilizar um campo de seleção de arquivo com filtro para somente executáveis, para que o usuário selecione o executável que sofrerá injeção;
- o código injetado deverá permanecer ativo até que o usuário solicite seu desligamento através de um botão na tela.

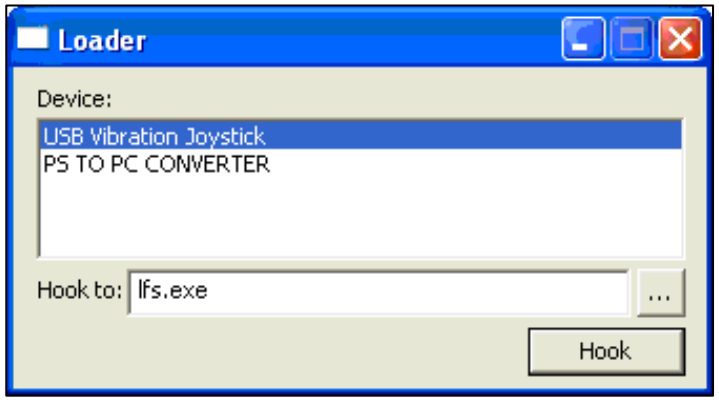
Desenvolvimento

Especificação

Casos de uso



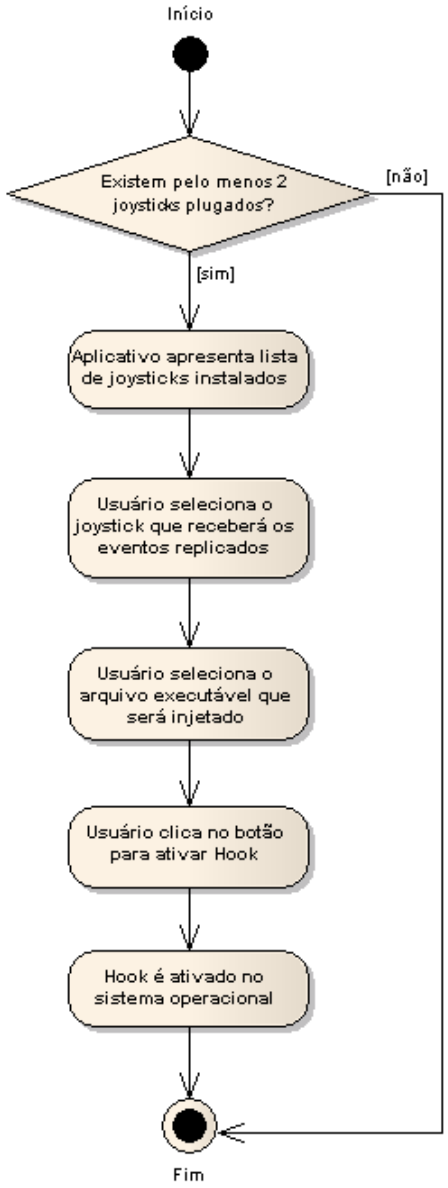
Interface do usuário



FURB
UNIVERSIDADE DE BLUMENAU

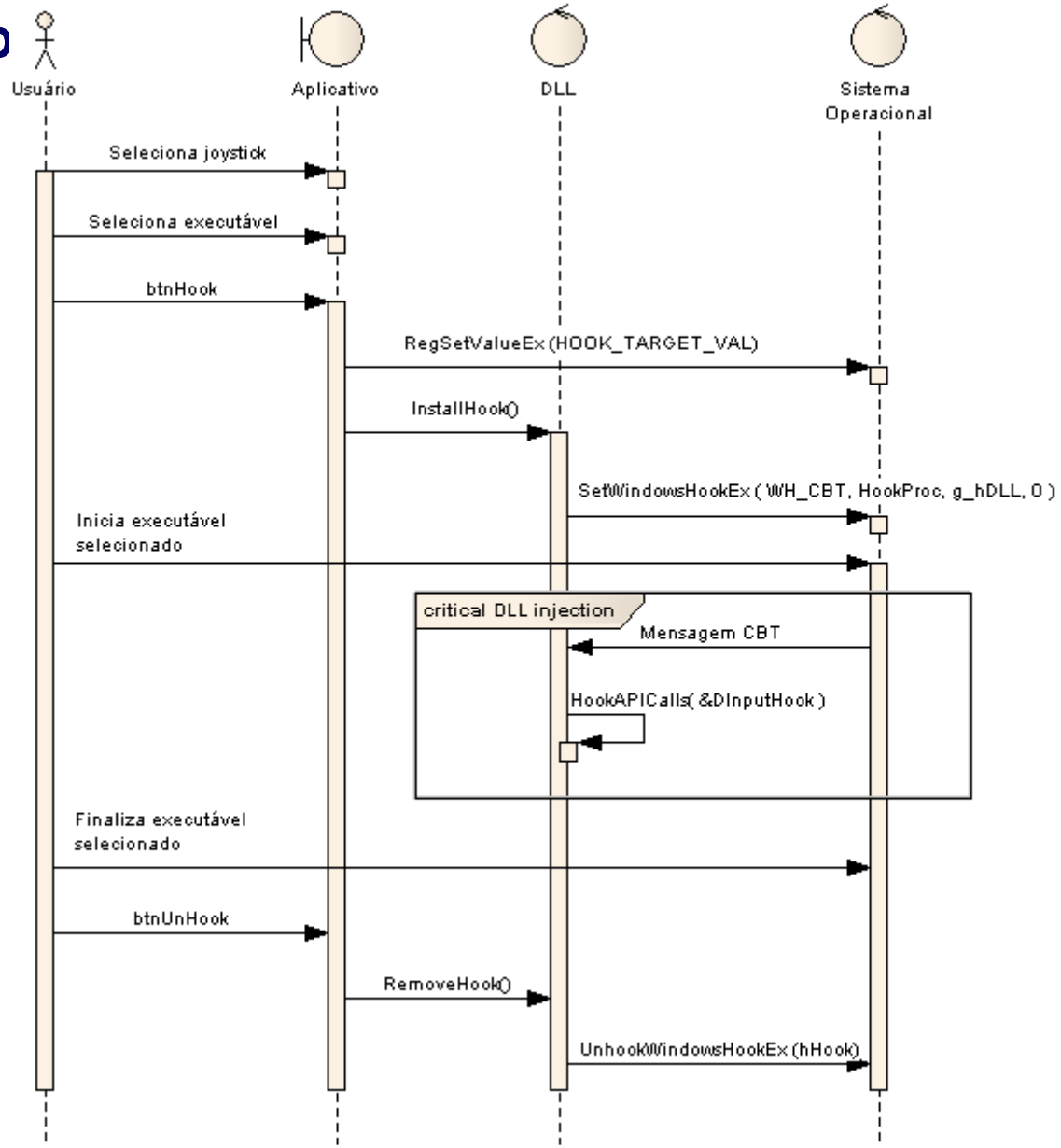
Desenvolvimento

Especificação



Desenvolvimento

Especificação



FURB
UNIVERSIDADE DE BLUMENAU

Desenvolvimento

- **Implementação**
  - Linguagem C++
  - Microsoft Visual Studio 2008
  - APIHijack
  - DirectX SDK

Desenvolvimento

Implementação – Hook

```
DIHOOK_API LRESULT CALLBACK HookProc( int nCode, WPARAM wParam, LPARAM lParam )
{
    return CallNextHookEx( hHook, nCode, wParam, lParam);
}

extern "C"
{
    DIHOOK_API void __stdcall InstallHook()
    {
        hHook = SetWindowsHookEx( WH_CBT, HookProc, g_hDLL, 0 );
    }

    DIHOOK_API void __stdcall RemoveHook()
    {
        UnhookWindowsHookEx( hHook );
    }
}
```

FURB
UNIVERSIDADE DE BLUMENAU

Desenvolvimento

Implementação – DllMain

```
BOOL APIENTRY DllMain( HMODULE hModule, DWORD dwReason, void* lpReserved )
{
    if( dwReason == DLL_PROCESS_ATTACH )
    {
        g_hDLL = hModule;

        HKEY hAppKey;

        if( RegOpenKeyEx( HKEY_CURRENT_USER, HOOK_REG_KEY, 0, KEY_QUERY_VALUE, &hAppKey ) == ERROR_SUCCESS )
        {
            TCHAR szTargetProcess[512];
            DWORD dwCount = sizeof( szTargetProcess );

            if( RegQueryValueEx( hAppKey, HOOK_TARGET_VAL, NULL, NULL, (BYTE*)szTargetProcess, &dwCount )
                == ERROR_SUCCESS )
            {
                TCHAR szProcess[512];
                GetModuleFileName( GetModuleHandle( NULL ), szProcess, _countof( szProcess ) );
                PathStripPath( szProcess );

                if( _strnicmp( szTargetProcess, szProcess, _countof( szProcess ) ) == 0 &&
                    LoadSeatGuid( hAppKey, &g_guidSeat ) )
                {
                    MakeLog( "Initializing hook\r\n" );
                    HookAPICalls( &DInputHook );
                }
            }

            RegCloseKey( hAppKey );
```

FURB
UNIVERSIDADE DE BLUMENAU

Desenvolvimento

Implementação – Função customizada

```
SDLLHook DInputHook =
{
    "DINPUT8.DLL", false, NULL,     // Default hook disabled, NULL function pointer.
    {
        { "DirectInput8Create", &MyDirectInput8Create },
        { NULL, NULL }
    }
};

// Hook function.
HRESULT WINAPI MyDirectInput8Create( HINSTANCE hinst, DWORD dwVersion, REFIID riidltf
                                   , void** ppvOut, LPUNKNOWN punkOuter )
{
    PFN_DirectInput8Create pfnOldFunc = static_cast<PFN_DirectInput8Create>
        ( DInputHook.Functions[0].OrigFn );

    HRESULT hr = pfnOldFunc( hinst, dwVersion, riidltf, ppvOut, punkOuter );

    if( SUCCEEDED( hr ) )
    {
        *ppvOut = new MyDirectInput8( reinterpret_cast<IDirectInput8*>( *ppvOut ) );
    }

    return hr;
}
```

FURB
UNIVERSIDADE DE BLUMENAU

Desenvolvimento

Implementação – Interfaces DirectX

```
#include "MyDirectInputDevice.h"

class MyDirectInput8 : public IDirectInput8
{
public:
    MyDirectInput8( IDirectInput8* dip ) : m_dip( dip )
    {
    }
    /*** DirectInput8 methods ***/
    STDMETHOD(CreateDevice)( REFGUID rguid, IDirectInputDevice8** device, LPUNKNOWN unknown )
    {
        if( rguid == g_guidSeat )
        {
            // How the game got this GUID since we're filtering it on enumDevices?
            return E_UNEXPECTED;
        }

        HRESULT hr = m_dip->CreateDevice( rguid, device, unknown );

        std::wstring sGuid = MyStringFromCLSID( rguid );
        MakeLog( "m_dip->CreateDevice, rguid = %S, hr = 0x%08X\r\n", sGuid.c_str(), hr );

        if( SUCCEEDED( hr ) )
        {
            IDirectInputDevice8* pSeat = NULL;
            hr = m_dip->CreateDevice( g_guidSeat, &pSeat, unknown );

            sGuid = MyStringFromCLSID( g_guidSeat );
            MakeLog( "m_dip->CreateDevice, SEAT = %S, hr = 0x%08X\r\n", sGuid.c_str(), hr );

            // Return our device
            *device = new MyDirectInputDevice8( *device, pSeat );
```

FURB
UNIVERSIDADE DE BLUMENAU

Desenvolvimento

Implementação – Interfaces DirectX

```
#include "MyDirectInputEffect.h"

class MyDirectInputDevice8 : public IDirectInputDevice8
{
public:
    MyDirectInputDevice8( IDirectInputDevice8* did, IDirectInputDevice8* seat ) :
      m_did( did ), m_seat( seat )
    {
    }
    /*** IDirectInputDevice8 methods ***/
    STDMETHOD(Initialize)( HINSTANCE instance, DWORD version, REFGUID rguid )
    {
        m_seat->Initialize( instance, version, rguid );
        return m_did->Initialize( instance, version, rguid );
    }

    STDMETHOD(CreateEffect)( REFGUID rguid, LPCDIEFFECT effect_params,
                             LPDIRECTINPUTEFFECT* effect, LPUNKNOWN unknown )
    {
        HRESULT hr = m_did->CreateEffect(rguid, effect_params, effect, unknown);

        if( SUCCEEDED( hr ) )
        {
            LPDIRECTINPUTEFFECT pMyEffect;
            HRESULT hrSeat = m_seat->CreateEffect( rguid, effect_params, &pMyEffect, unknown );

            // Return our device
            *effect = new MyDirectInputEffect( *effect, pMyEffect );
        }

        return hr;
    }
```

FURB
UNIVERSIDADE DE BLUMENAU

Desenvolvimento

Implementação – Interfaces DirectX

```
class MyDirectInputEffect : public IDirectInputEffect
{
public:
    MyDirectInputEffect(IDirectInputEffect* die, IDirectInputEffect* seat) :
      m_die(die), m_seat(seat)
    {
    }
    /*** IDirectInputEffect methods ***/
    STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion, REFGUID rguid) {
        m_seat->Initialize(hinst, dwVersion, rguid);
        MakeLog( "m_die->Initialize\r\n" );
        return m_die->Initialize(hinst, dwVersion, rguid);
    }
    STDMETHOD(GetParameters)(THIS_ LPDIEFFECT peff, DWORD dwFlags) {
        return m_die->GetParameters(peff, dwFlags);
    }

    STDMETHOD(SetParameters)(THIS_ LPCDIEFFECT peff, DWORD dwFlags) {
        HRESULT hrSeat = m_seat->SetParameters(peff, dwFlags);
        MakeLog("SetParameters, hrSeat = 0x%08X\r\n", hrSeat );

        HRESULT hrDie = m_die->SetParameters(peff, dwFlags);
        MakeLog("SetParameters, hrDie = 0x%08X\r\n", hrDie );

        return hrDie;
    }

    STDMETHOD(Start)(THIS_ DWORD dwIterations, DWORD dwFlags) {
        MakeLog( "m_die->Start\r\n" );
        m_seat->Start(dwIterations, dwFlags);
        return m_die->Start(dwIterations, dwFlags);
    }
```

FURB
UNIVERSIDADE DE BLUMENAU

Conclusão

- O processo de DLL injection é executado com êxito pelo aplicativo, carregando as funções da DLL injetada no processo e substituindo na IAT as funções da DLL escolhida.
- A função desenvolvida filtra eventos de Force Feedback e duplica para outro controlador, mas outros tipos de uso podem ser feitos, bastando alterar as interfaces DirectX.

Extensão

- **Fazendo uso da disponibilidade de redirecionamento dos eventos de Force Feedback, foi desenvolvido um acessório para simular o cockpit de um carro de corrida.**
- **Dessa forma, o assento do piloto reage vibrando juntamente com o volante no caso de uma colisão ou de sair da pista, assim como ocorre na vida real.**

Apresentação do protótipo BodyStruck

Giovani Chaves
gchaves@inf.furb.br
giovanichaves@gmail.com