It’s friday, around 18 o’clock

it’s been a tough week

you’re tired and you don’t really wanna leave the house

so, you launch your favourite video game The Witcher 3™ and start killing monsters

BUT!

after roughly 30 minutes you start thinking...

Is the order argument in the

`std::io::copy`

reader first, **writer second**

or

writer first, **reader second**

?

Now, you could obviously quit or alt-tab the game and check this on the rust docs page...

...but there has to be a better way!

Has this ever happened to you?!?

Introducing…
The Rust Docs Overlay!


Z
C
K
J
I
20/20

How to make your own Steam Overlay! With Rust!

By Jakub Trąd

Disclaimer #1

The project I’m about to talk about is (unfortunately) not open source. I wish it were open source, but that decision is not up to me.

Disclaimer #2

This is pretty much all **unsafe**. The very idea of this project is hacky, to say the least, and due to a lot of interop with Win32 API, there’s a lot of unsafe code. Viewer discretion is advised.

Disclaimer #3

There’s Electron and JS mentioned. Not for the faint of heart

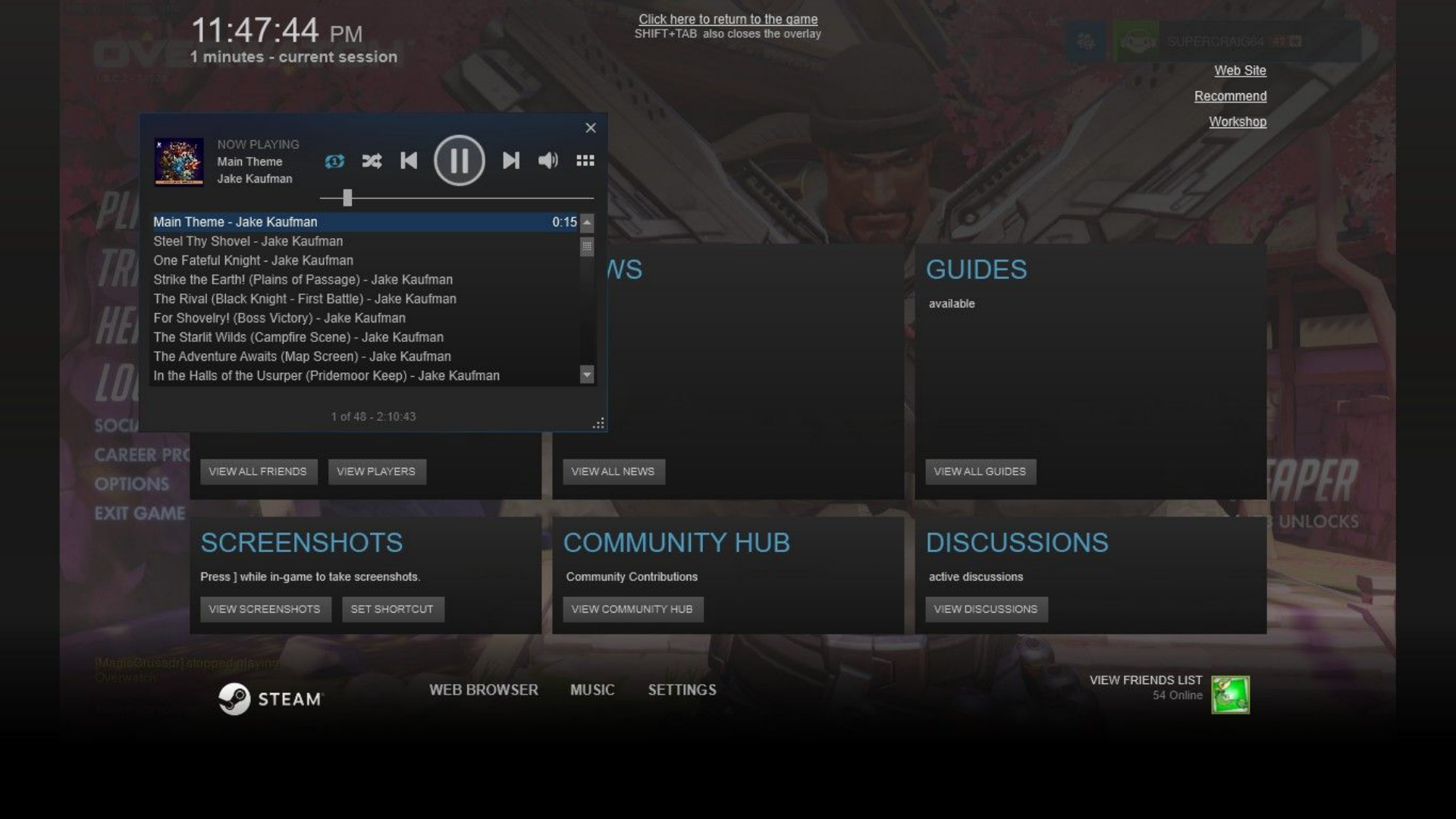
Disclaimer #4

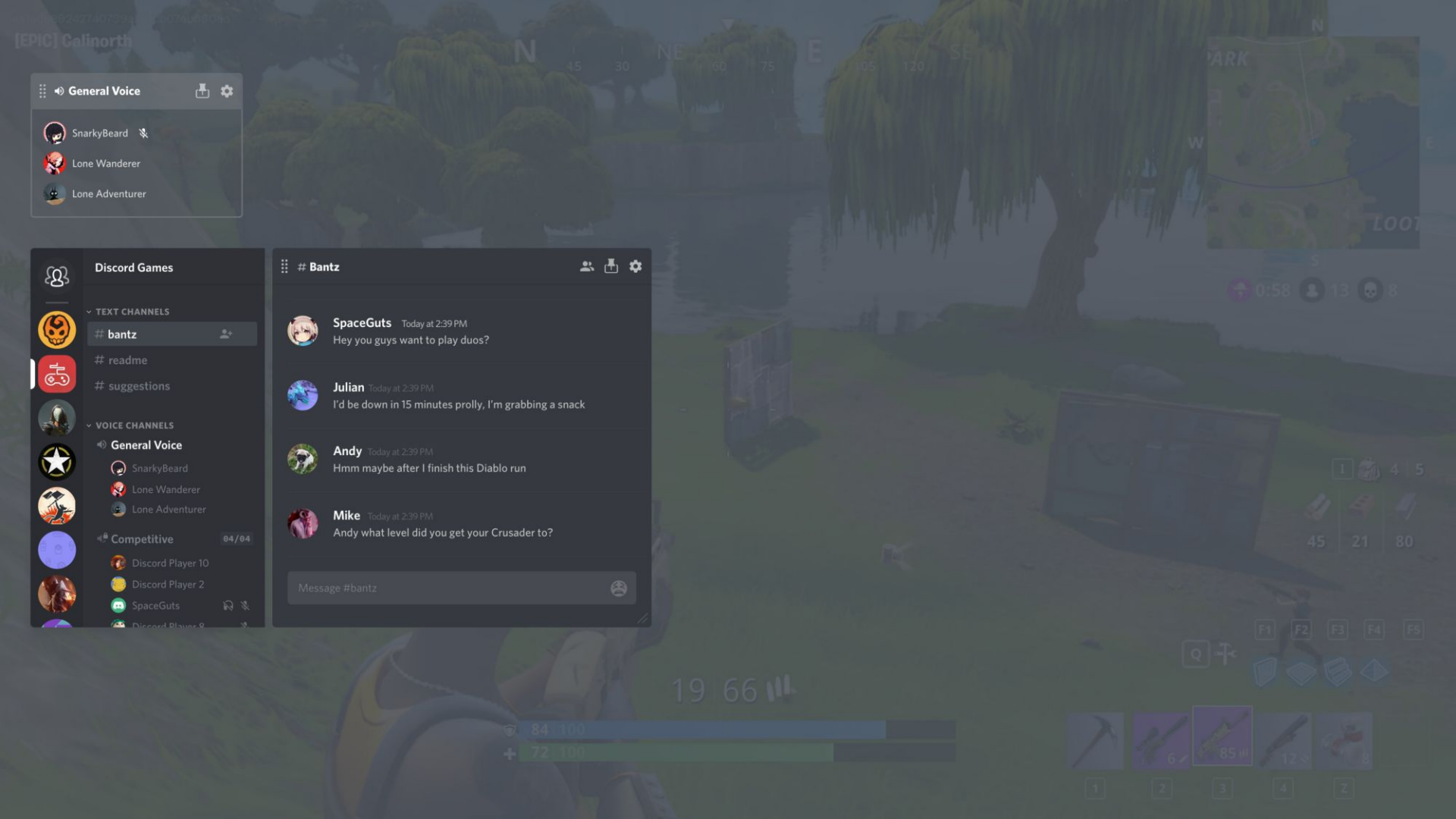
This is all Windows, there’s no Linux or OSX involved

Disclaimer #5

I **will not** be covering the rendering part of this project. As that would basically be equal to doing a “Hello World” example project in DirectX/OpenGL in Rust

What is this “overlay”?


11:47:44 PM
1 minutes - current session
Click here to return to the game
SHIFT+TAB also closes the overlay
Web Site
Recommend
Workshop
NOW PLAYING
Main Theme
Jake Kaufman
Main Theme - Jake Kaufman 0:15
Steel Thy Shovel - Jake Kaufman
One Fateful Knight - Jake Kaufman
Strike the Earth! (Plains of Passage) - Jake Kaufman
The Rival (Black Knight - First Battle) - Jake Kaufman
For Shovelry! (Boss Victory) - Jake Kaufman
The Starlit Wilds (Campfire Scene) - Jake Kaufman
The Adventure Awaits (Map Screen) - Jake Kaufman
In the Halls of the Usurper (Pridemoor Keep) - Jake Kaufman
1 of 48 - 2:10:43
GUIDES
available
VIEW ALL FRIENDS
VIEW PLAYERS
VIEW ALL NEWS
VIEW ALL GUIDES
SCREENSHOTS
Press] while in-game to take screenshots.
VIEW SCREENSHOTS
SET SHORTCUT
COMMUNITY HUB
Community Contributions
VIEW COMMUNITY HUB
DISCUSSIONS
active discussions
VIEW DISCUSSIONS
STEAM
WEB BROWSER
MUSIC
SETTINGS
VIEW FRIENDS LIST
54 Online


[EPIC] Calinorth
General Voice
SnarkyBeard
Lone Wanderer
Lone Adventurer
Discord Games
TEXT CHANNELS
bantz
readme
suggestions
VOICE CHANNELS
General Voice
SnarkyBeard
Lone Wanderer
Lone Adventurer
Competitive
04/04
Discord Player 10
Discord Player 2
SpaceGuts
Bantz
SpaceGuts Today at 2:39 PM
Hey you guys want to play duos?
Julian Today at 2:39 PM
I'd be down in 15 minutes prolly, I'm grabbing a snack
Andy Today at 2:39 PM
Hmm maybe after I finish this Diablo run
Mike Today at 2:39 PM
Andy what level did you get your Crusader to?
Message #bantz

How does it work?!?

1. Inject code into a game (dll injection)
2. Detect and “hook” into a rendering pipeline (hooking)
3. Generate and render the overlay image

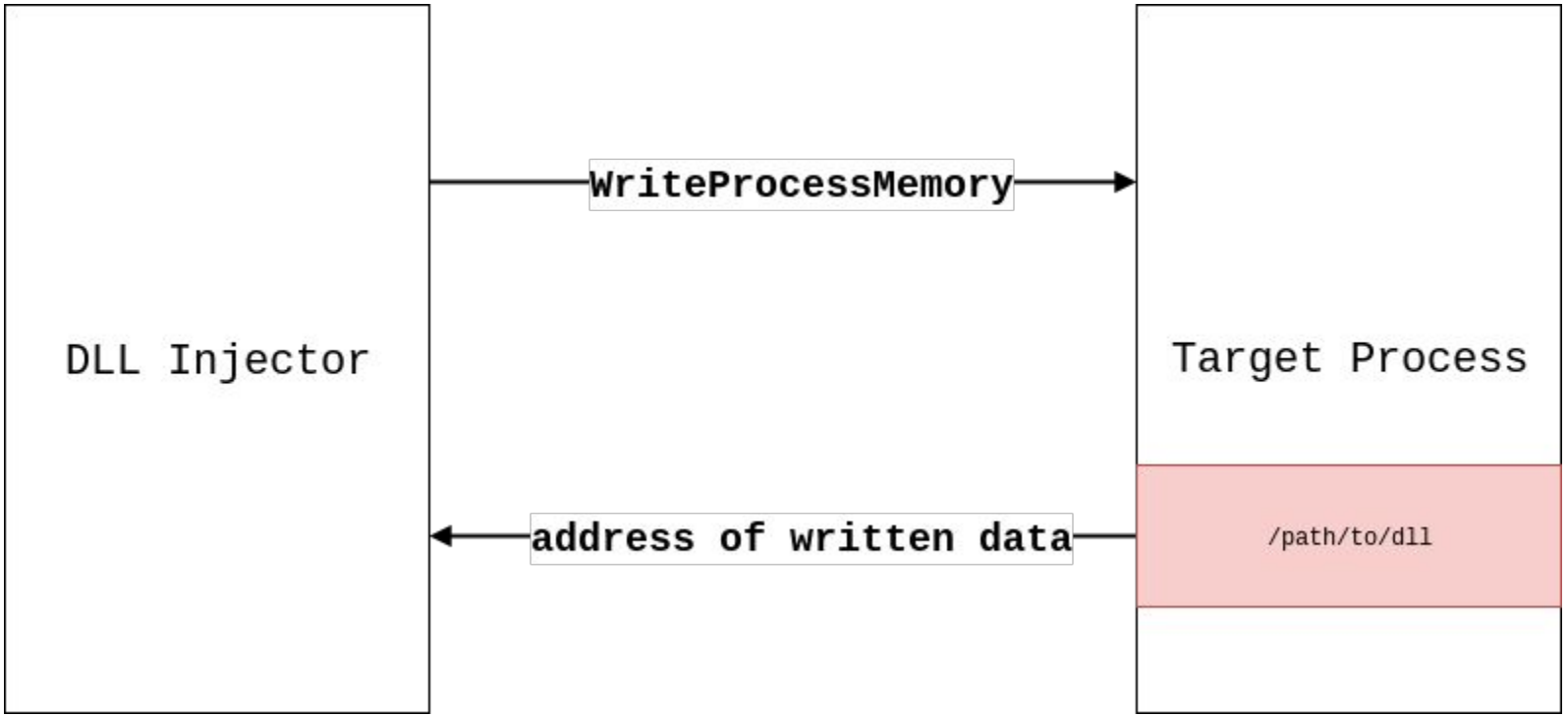
Additionally!

1. Hook into and intercept input methods (input hooking)
2. Communicate with the outside world
3. Take screenshots, record videos, measure FPS,
4. and much more...

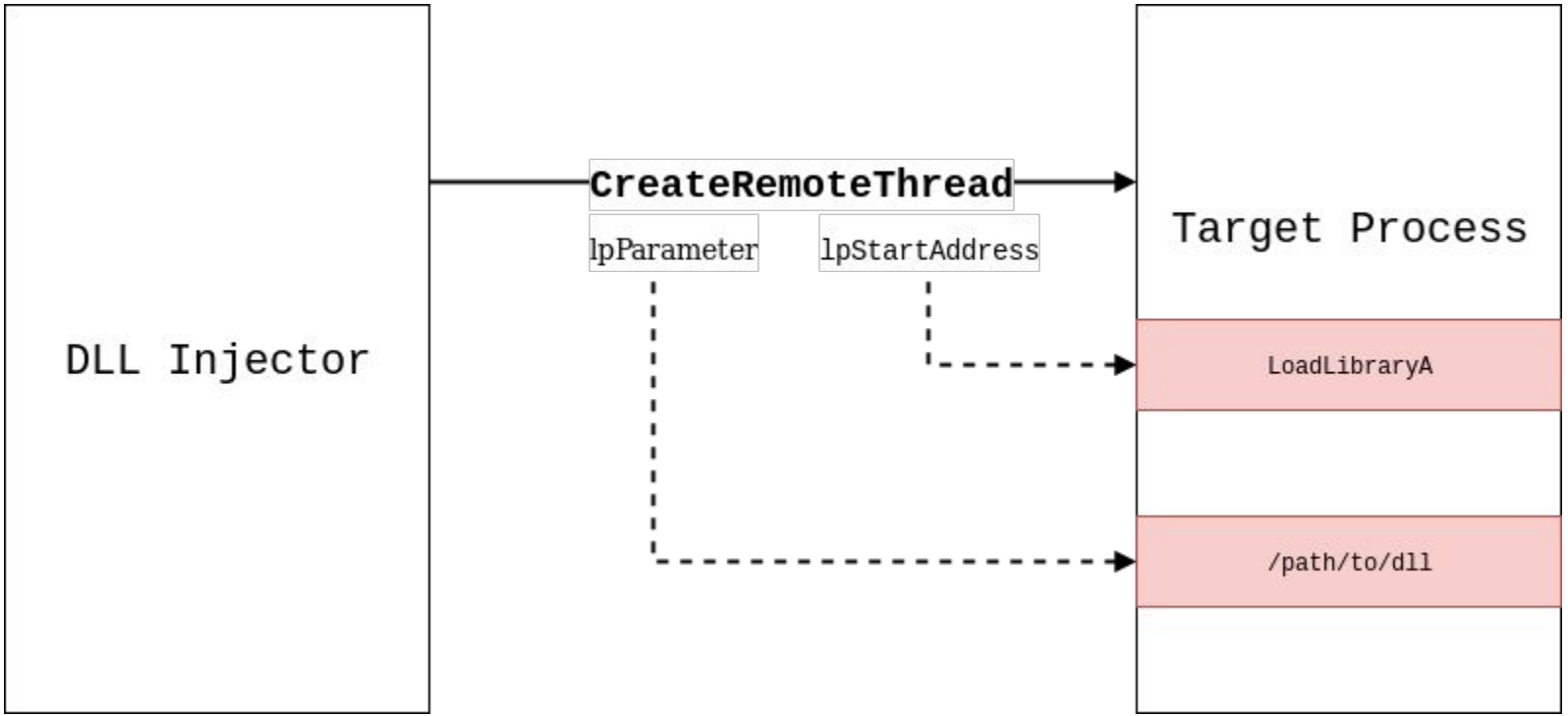
How do we do that?

Step 1: Injection

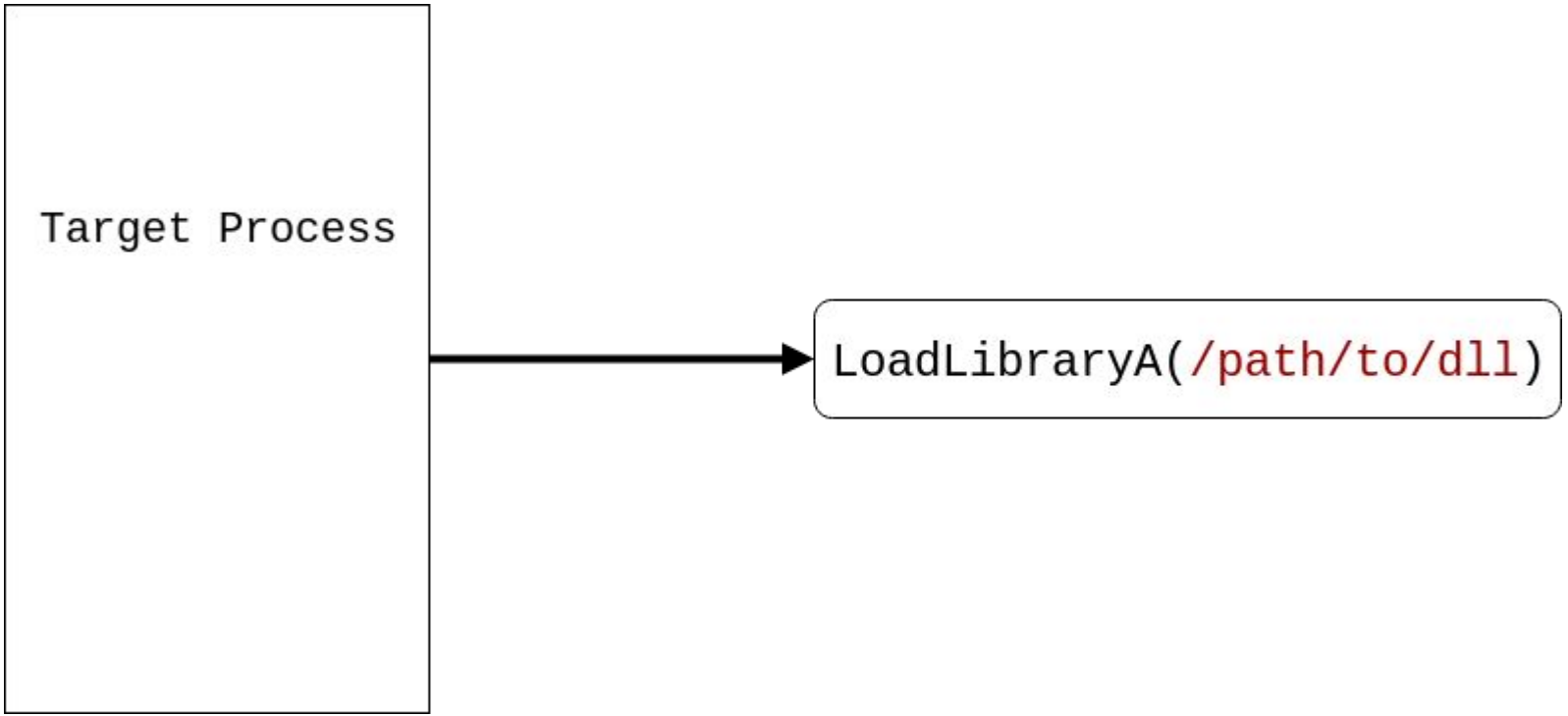
DLL injection



DLL injection



DLL injection



```
HANDLE CreateRemoteThread(
  HANDLE                 hProcess,
  LPSECURITY_ATTRIBUTES  lpThreadAttributes,
  SIZE_T                 dwStackSize,
  LPTHREAD_START_ROUTINE lpStartAddress,
  LPVOID                 lpParameter,
  DWORD                  dwCreationFlags,
  LPDWORD                lpThreadId
);
```

```
HANDLE CreateRemoteThread(
  HANDLE                 hProcess,
  LPSECURITY_ATTRIBUTES  lpThreadAttributes,
  SIZE_T                 dwStackSize,
  LPTHREAD_START_ROUTINE lpStartAddress,
  LPVOID                 lpParameter,
  DWORD                  dwCreationFlags,
  LPDWORD                lpThreadId
);
```

```
DWORD WINAPI ThreadProc(
  _In_ LPVOID lpParameter
);

HMODULE LoadLibraryA(
  LPCSTR lpLibFileName
);
```

```
let kernel32_handle =
    GetModuleHandleA(utils::to_cstr("Kernel32").as_ptr());

let target_process = OpenProcess(
    PROCESS_CREATE_THREAD
        | PROCESS_QUERY_INFORMATION
        | PROCESS_VM_OPERATION
        | PROCESS_VM_WRITE
        | PROCESS_VM_READ,
    FALSE,
    pid,
);
```

```
let remote_lib_path = VirtualAllocEx(
    target_process,
    std::ptr::null_mut(),
    lib_path.len(),
    MEM_RESERVE | MEM_COMMIT,
    PAGE_READWRITE,
);
```

```
let write_res = WriteProcessMemory(
    target_process,
    remote_lib_path,
    lib_path.as_ptr() as *const _,
    lib_path.len(),
    std::ptr::null_mut(),
);
```

```
let injection_process =
    GetProcAddress(
        kernel32_handle,
        utils::to_cstr("LoadLibraryA").as_ptr());
```

```
let thread_handle = CreateRemoteThread(
    target_process,
    std::ptr::null_mut(),
    0,
    std::mem::transmute(injection_process),
    remote_lib_path,
    0,
    std::ptr::null_mut(),
);
```


hacker voice
we're in

```rust
#[no_mangle]
pub unsafe extern "system" fn DllMain(
    _module: HINSTANCE,
    reason: DWORD,
    _: LPVOID
) -> BOOL {
    if reason == DLL_PROCESS_ATTACH {
        thread::spawn(|| { /* .. your Overlay code ... */});
    }

    TRUE
}
```

Remarks

1. Make sure to call `GetModuleHandleExW` soon after `DllMain`, not doing so will immediately unload your overlay DLL.
2. In the same manner call `FreeLibraryAndExitThread` to unload your DLL and terminate your thread
3. I'll mention this later, but a lot of Win32 API calls are thread specific. For instance `ShowCursor` will not work if called from your thread
4. Build your crate with `crate-type = ["dylib"]`

Questions?

Step 2: Hooking

```rust
fn hello_world() {
    println!("Hello, World!");
}
```

```rust
fn new_hello_world() {
    println!("Suprise!!!");
}
```

```rust
use detour::RawDetour;

// ...

fn main() {
    hello_world(); // prints "Hello, World!"
    unsafe {
        let detour = RawDetour::new(
            hello_world as *const (),
            new_hello_world as *const ()
        ).unwrap();
        detour.enable().unwrap();
        hello_world(); // prints "Suprise!!!"

        let trampoline: fn()
            = std::mem::transmute(detour.trampoline());
        trampoline(); // prints "Hello, World!"
    }
    hello_world(); // prints "Hello, World!"
}
```

How to hook into DirectX?

```
HRESULT IDXGISwapChain::Present(
  UINT SyncInterval,
  UINT Flags
);
```

```rust
pub type PresentHookType
    = unsafe extern "system" fn(
        *mut IDXGISwapChain,
        UINT,
        UINT
    ) -> HRESULT;

pub unsafe extern "system" fn _hooked_present(
    swap_chain: *mut IDXGISwapChain,
    sync_interval: UINT,
    flags: UINT,
) -> HRESULT {
    // your rendering code
    // ...

    let trampoline: PresentHookType
        = std::mem::transmute(HOOK.trampoline());

    trampoline(swap_chain, sync_interval, flags)
}
```

Uh Oh!

`IDXGISwapChain::Present`

is a C++ COM method!

Which means it’s somewhere in an object’s vtable!!!

http://www.directxtutorial.com/Lesson.aspx?lessonid=11-4-2

```
// you'll be using these quite a bit

let null_ptr = std::ptr::null_mut();

// a struct initialized to 0, like in C!
let zeroed_struct = std::mem::zeroed();
```

```
pub unsafe extern "system" fn D3D11CreateDeviceAndSwapChain(
    pAdapter: *mut IDXGIAdapter,
    DriverType: D3D_DRIVER_TYPE,
    Software: HMODULE,
    Flags: UINT,
    pFeatureLevels: *const D3D_FEATURE_LEVEL,
    FeatureLevels: UINT,
    SDKVersion: UINT,
    pSwapChainDesc: *const DXGI_SWAP_CHAIN_DESC,
    ppSwapChain: *mut *mut IDXGISwapChain,
    ppDevice: *mut *mut ID3D11Device,
    pFeatureLevel: *mut D3D_FEATURE_LEVEL,
    ppImmediateContext: *mut *mut ID3D11DeviceContext
) -> HRESULT
```

```
pub unsafe extern "system" fn D3D11CreateDeviceAndSwapChain(
    pAdapter: *mut IDXGIAdapter,
    DriverType: D3D_DRIVER_TYPE,
    Software: HMODULE,
    Flags: UINT,
    pFeatureLevels: *const D3D_FEATURE_LEVEL,
    FeatureLevels: UINT,
    SDKVersion: UINT,
    pSwapChainDesc: *const DXGI_SWAP_CHAIN_DESC,
    ppSwapChain: *mut *mut IDXGISwapChain,
    ppDevice: *mut *mut ID3D11Device,
    pFeatureLevel: *mut D3D_FEATURE_LEVEL,
    ppImmediateContext: *mut *mut ID3D11DeviceContext
) -> HRESULT
```

```
let vtable = swap_chain as *mut &[*mut c_void];
let present: *mut c_void = (*vtable)[8];
```

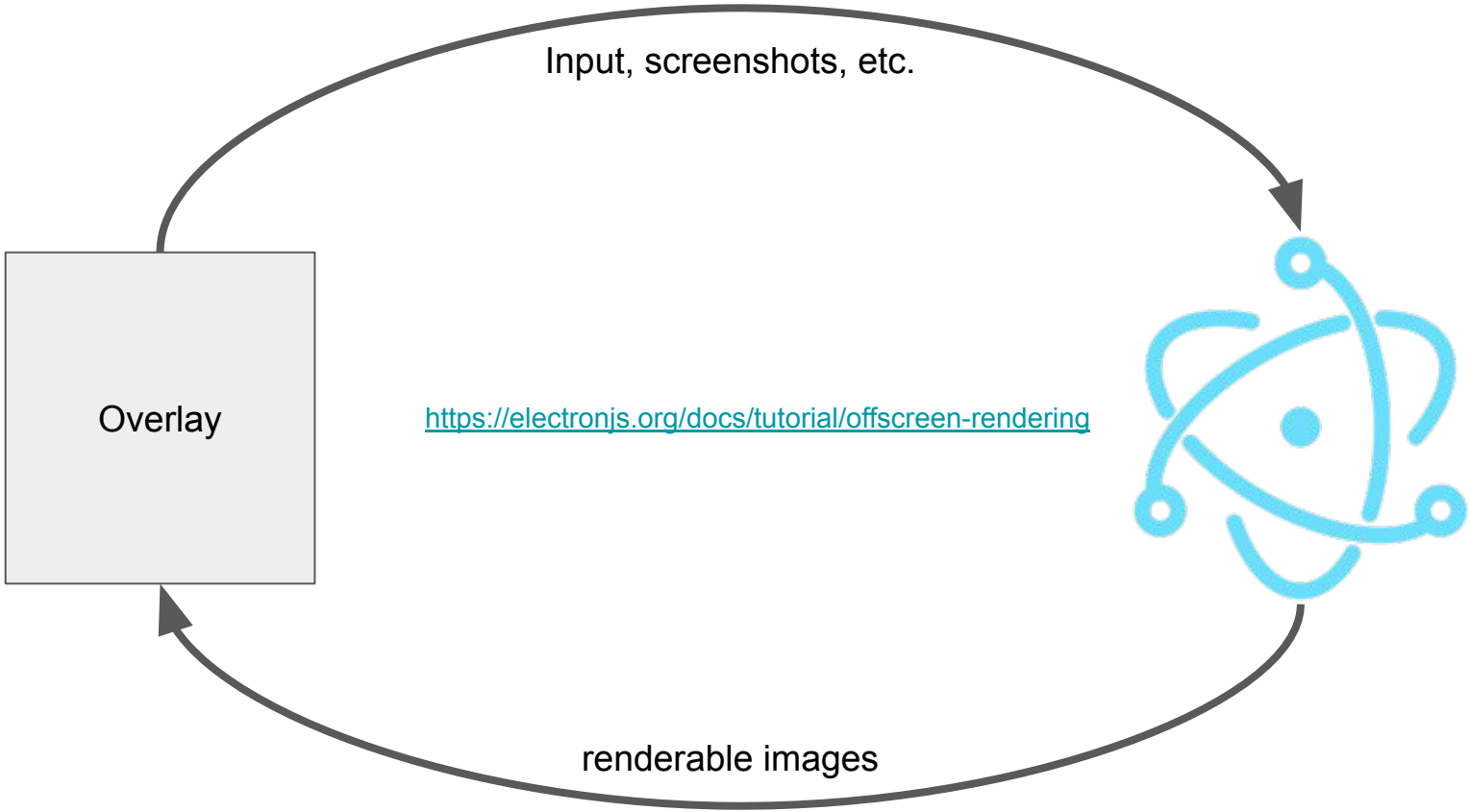
```
// HOOK is global/static and mutable, maybe AtomicPtr?
HOOK = RawDetour::new(
    present as *const (),
    _hooked_present as *const (),
)
```

And voilà!

don’t forget to release your COM interfaces!

Is that it?

Of course not!


Input, screenshots, etc.
Overlay
https://electronjs.org/docs/tutorial/offscreen-rendering
renderable images

WARNING:
The next slide contains JS code!

```
const { app, BrowserWindow } = require('electron')

app.disableHardwareAcceleration()

let win

app.once('ready', () => {
  win = new BrowserWindow({
    webPreferences: {
      offscreen: true
    }
  })

  win.loadURL('http://github.com')
  win.webContents.on('paint', (event, dirty, image) => {
    // dirty is a Rect of which part of the screen is updated
    // image contains the pixels of ENTIRE screen
  })
  win.webContents.setFrameRate(30)
})
```

But how does it talk to Electron?

Protocol Buffers!

over a TCP socket

```
message Image {
    uint32 x = 1;
    uint32 y = 2;
    uint32 width = 3;
    uint32 height = 4;

    bytes pixels = 5;
}
```

A lot of inter-thread plumbing...

And that’s where Rust really shines!

I want to make my own overlay!
How do I start?

Open source overlays

1. https://github.com/hiitiger/gelectron
2. https://github.com/mumble-voip/mumble

Dll injectors

1. https://github.com/amcarthur/hammer
2. https://github.com/segfo/dllinjector-rs

Useful crates

1. https://crates.io/crates/winapi
2. https://crates.io/crates/detour
3. https://crates.io/crates/crossbeam

Assorted reading

1. https://www.codeproject.com/Articles/4610/Three-Ways-to-Inject-Your-Code-into-Another-Proces
2. https://electronjs.org/docs/tutorial/offscreen-rendering
3. https://docs.microsoft.com/en-us/windows/win32/api/
4. http://www.directxtutorial.com/LessonList.aspx?listid=11

~fin~