FORTRA

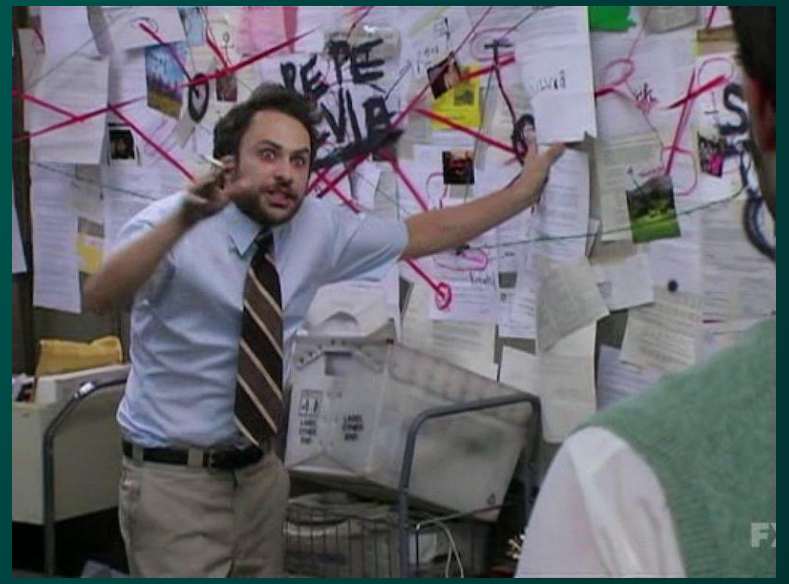
Starting at 15:00 Orchestrating Modern Implants

With LLVM

FORTRA

Agenda

- Why should you care?
- Motivations
- Introduction to LLVM
- The Architecture of LLVM
- Examples – Case Study
- Examples I want to implement
- Demo



FORTRA

Why should you care?

- End of the day, trying to deal with static/Behavorial signatures is a pain for commercial frameworks
- Some tradecraft implementations can be implemented as a pass so its less work to implement without it.
- 50 disjointed implementations of what I want to achieve when I can just write a pass for one thing that can be applied to everything.

FORTRA

Motivations

- (Most!) existing research & frameworks don’t go beyond obfuscation
- Majority are not designed with the idea of a flat-executable ”Position Independent Code” (PIC), and rely on the notion of having a R/W region of memory within the same address space, rather than assuming the only permissions I have are execute and read.
- For Cobalt Strike having our own LLVM tooling is of huge benefit for future development / research capabilities where otherwise it would've been a significant pain or impossible

Introduction to LLVM

- The “Low Level Virtual Machine” also known as ”LLVM” is a modern compiler framework developed by the University of Illinois student Chris Lattner as apart of his master thesis with a more modern Intermediary Representation known as “LLVM IR”.
- Eventually Chris developed a front-end called “Clang” short for `C-lang` while at Apple that translated the C/C++ language into this language-neutral intermediary IR.
- Its often used by more modern languages like Rust, Swift, Zig as its easier to develop a frontend for an already established backend. Henri Nurmi developed a PoC of this himself and got far enough to write his own BOFs.

FORTRA

Architecture

- Lexer/etc to translate into LLVM IR, which is the universal intermediary before optimization occurs
- Optimization/target lowering then inlines / removes dead code etc.
  - Can implement Module or Function passes
- Backend translates into target-specific intermediary before assembling into machine code.
  - Can implement your own Machine Function per target architecture

FORTRA

FORTRA

Demo:
Generating IR from C using Clang

FORTRA

Examples of Module/Function passes – Case Study

- Most of the major “Obfuscation Frameworks” out there are implemented as module / function passes.
- Think like Obfuscation-LLVM, Hikari(-LLVM15), Polaris, Pluto Obfuscator, xVMP, Goron
  - https://github.com/eshard/obfus[illegible]llvm
- Or, even your own virtual machine translation as seen with Secret Club’s Risc-VM a couple of years ago and PSXExecute.
  - https://github.com/thesecretclub/riscy-business
  - https://github.com/eversinc33/PSXecute

FORTRA

Examples of Module/Function Passes I want to implement

- Assist Position Independent Code Development – Make it (almost) as easy as writing a normal executable.
- Mutation/Polymorphic engines on the target level so “signatures” became a non issue as well as a engine for hindering RE efforts
- ...Stuff :)

FORTRA

Demo:
String / Integer Constant Mutations

FORTRA

FORTRA

Questions/Comments/Concern?