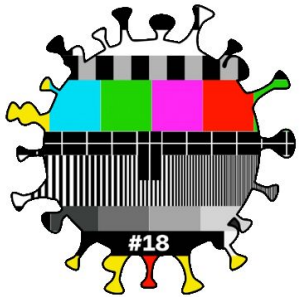
First Steps with Ghidra libsla.a



@guedou

sstic2020 - rumps

Ghidra?

See https://ghidra-sre.org/

Open-Source RE developed by the NSA

revealed in the Vault7 leak

released in March 2019

https://github.com/NationalSecurityAgency/ghidra

many features

disassembly, graphing, scripting, extensions, decompiling...

SLEIGH?

language derived from SLED

Specification Language for Encoding and Decoding
architecture independent disassembler ~~& assembler~~

ease defining instructions decoding & semantics

data-flow & decompilation analysis
semantics converted to Ghidra IR (aka P-CODE)

also a command-line tool

$GHIDRA_HOME/support/sleigh

Mandatory Processor Architecture Files Structure

guedou/ghidra-processor-mep

```
$ tree Ghidra/Processors/MEP_C4/
Ghidra/Processors/MEP_C4/
├── data
│   └── languages
│       ├── mep_c4.cspec
│       ├── mep_c4.ldefs
│       ├── mep_c4.pspec
│       ├── mep_c4.sla
│       └── mep_c4.slaspec
└── Module.manifest

2 directories, 6 files
```

SLEIGH Specification File -.slaspec

See https://docs.google.com/presentation/d/1b955DV2ii-Dgv6YR4kUrJtjGugEqXD3FffTHRfvVSYo

compiled to.sla with the sleigh command

XML version of.slaspec with P-CODE

this format is only useful in Ghidra

is it?

liblsla.a?

C++ bindings

easily disassemble & emulate using a .sla file

already included in Ghidra

a different implementation than the Java one

Compiling liblsla.a

1. build the **ghidra-builder** Docker image
   easily disassemble & emulate using a .sla file
2. build Ghidra
   go for a long walk
3. compile libsla & the example
   a pending Ghidra PR automates this

Step by Step

```
$ git clone https://github.com/dukebarman/ghidra-builder
$ cd ghidra-builder
$ ./docker-tpl/build

$ cd workdir
$ ../docker-tpl/run bash
$ ./build_ghidra.sh

$ cd ghidra/Ghidra/Features/Decompiler/src/decompile/cpp
$ git fetch origin pull/1677/head:sstic2020
$ git checkout sstic2020
$ make sleighexamp_dir
$ cd sleigh-*/
$ make sleighexample
```

./sleighexample demo

Perspective: Rust & Python Bindings

```
$ ./sori 10cc0100
instruction.size=4
instruction.buf_asm=add3 r12, r1, #0x1
instruction.code=
(unique,0x740,4) = INT_SEXT (const,0x1,2)
(register,0x1030,4) = INT_ADD (register,0x1004,4) (unique,0x740,4)
```

Questions?
Virtual Beers?