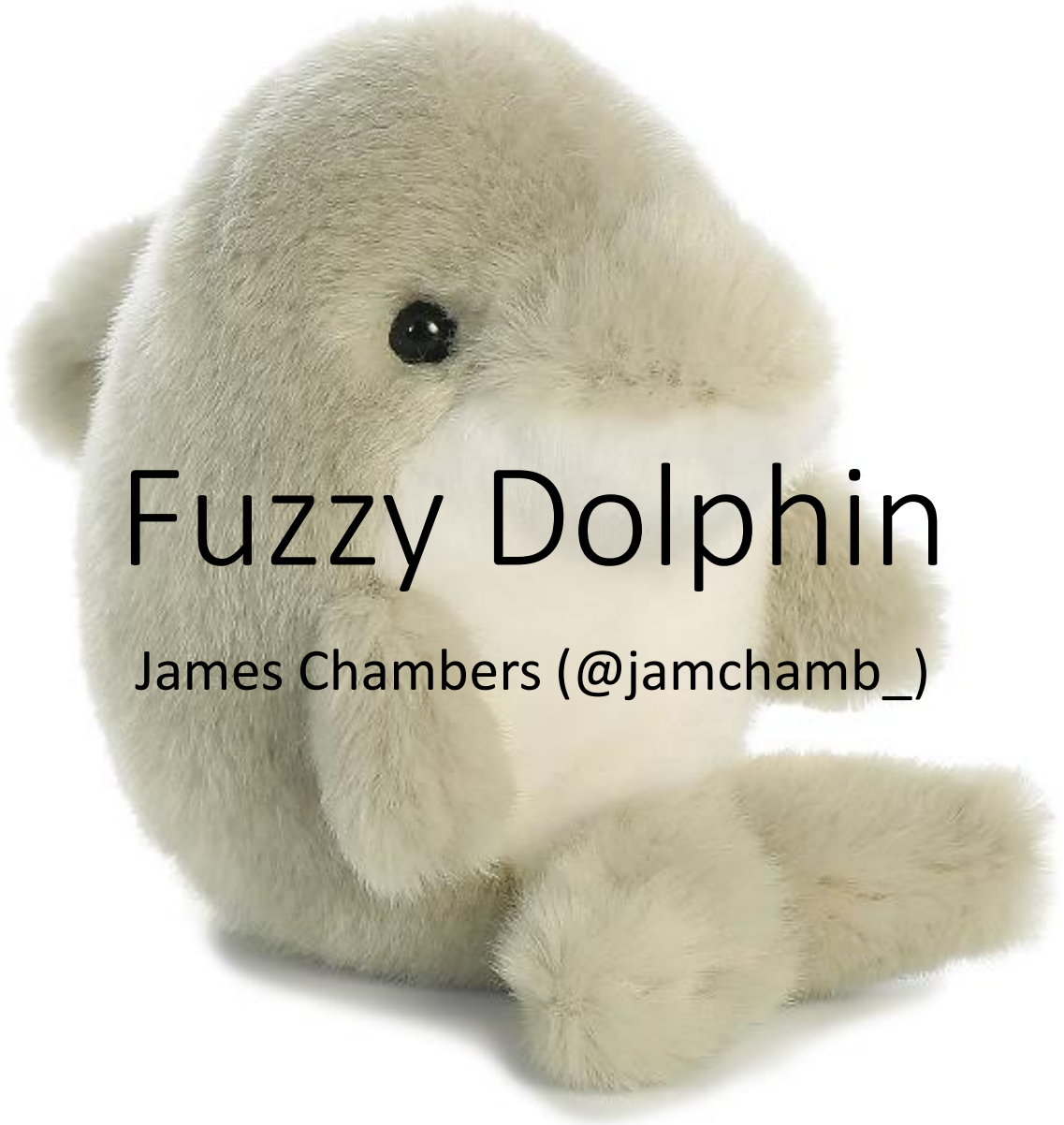
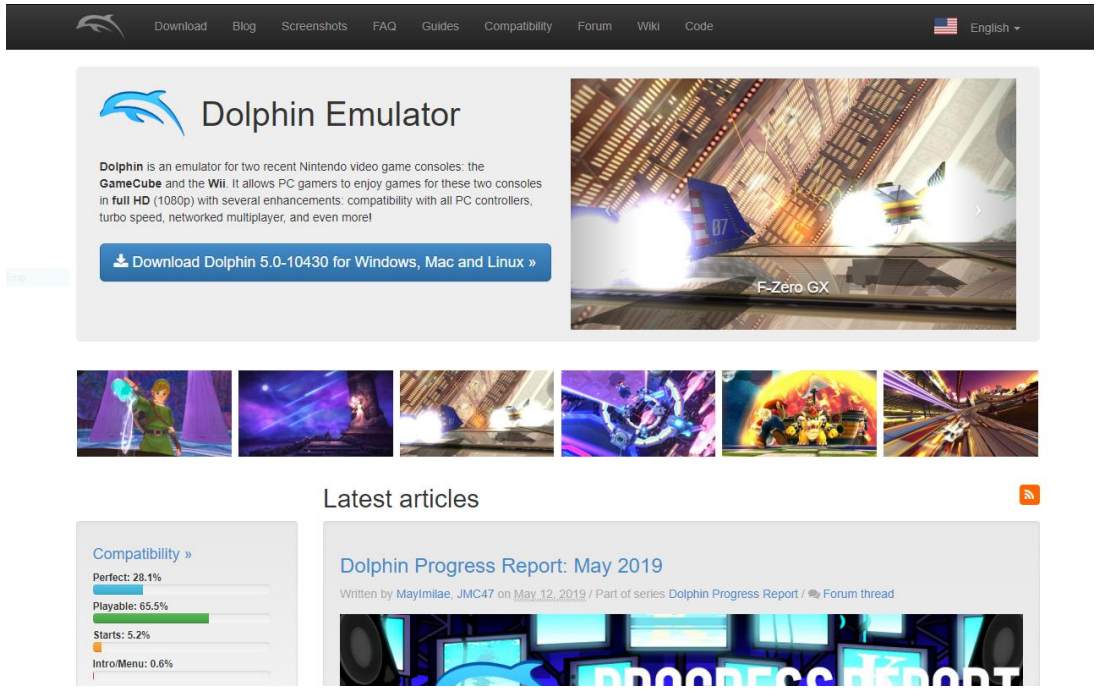


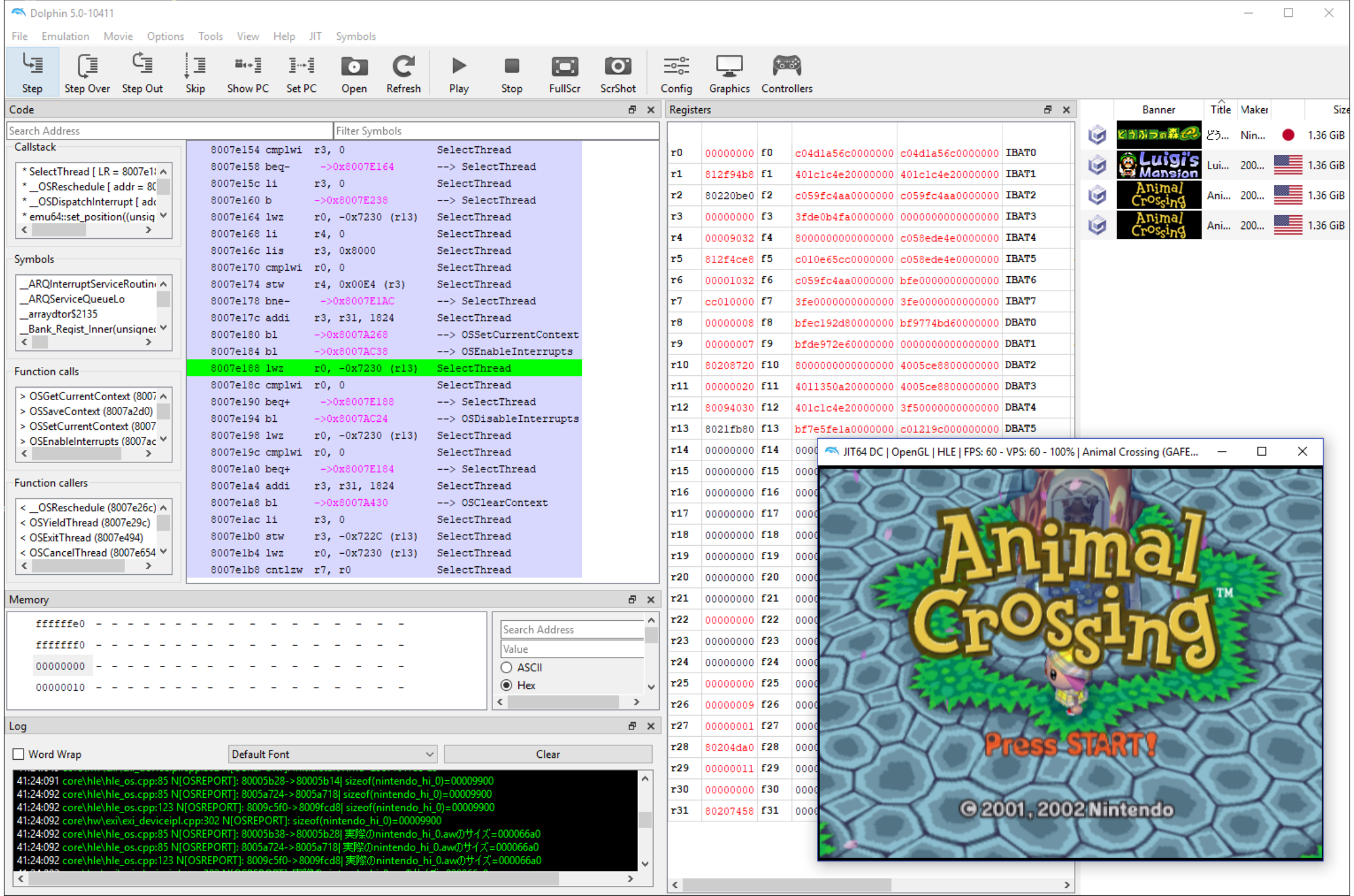
Fuzzy Dolphin

James Chambers (@jamchamb_)

Fuzzy Dolphin

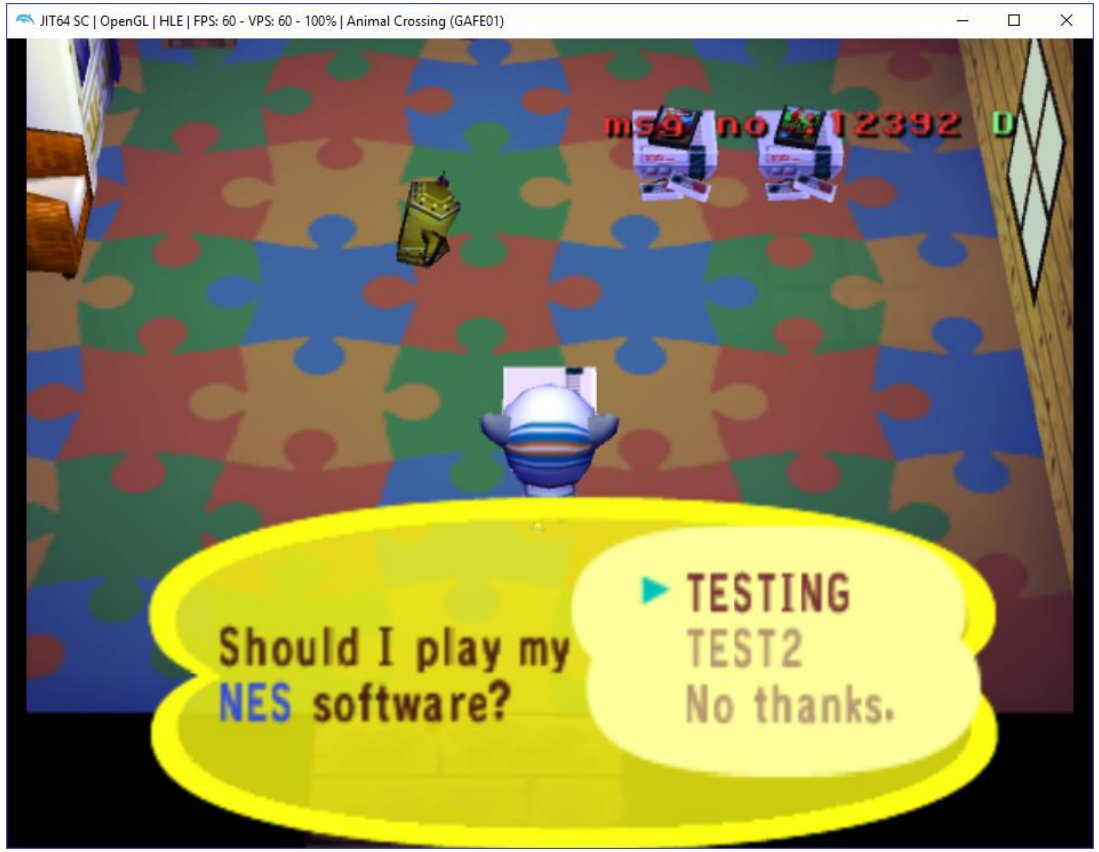
- Fuzzing GameCube (and Wii) games using Dolphin Emulator
- Dolphin is a well made, performant emulator for the Nintendo GameCube and Wii consoles
- It has a nice debugger that's useful for reverse engineering




Dolphin 5.0-10411
File Emulation Movie Options Tools View Help JIT Symbols
Step Step Over Step Out Skip Show PC Set PC Open Refresh Play Stop FullScr ScrShot Config Graphics Controllers
Code
Search Address
Filter Symbols
Callstack
* SelectThread [LR = 8007e1
* __OSReschedule [addr = 80
* __OSDispatchInterrupt [ad
* emu64::set_position((unsig
Symbols
__ARQInterruptServiceRoutin
__ARQServiceQueueLo
__arraydtor$2135
__Bank_Regist_Inner(unsigne
Function calls
> OSGetCurrentContext (8007
> OSSaveContext (8007a2d0)
> OSSetCurrentContext (8007
> OSEnableInterrupts (8007ac
Function callers
< __OSReschedule (8007e26c)
< OSYieldThread (8007e29c)
< OSExitThread (8007e494)
< OSCancelThread (8007e654
8007e154 cmplwi r3, 0 SelectThread
8007e158 beq- ->0x8007E164 --> SelectThread
8007e15c li r3, 0 SelectThread
8007e160 b ->0x8007E238 --> SelectThread
8007e164 lwz r0, -0x7230 (r13) SelectThread
8007e168 li r4, 0 SelectThread
8007e16c lis r3, 0x8000 SelectThread
8007e170 cmplwi r0, 0 SelectThread
8007e174 stw r4, 0x00E4 (r3) SelectThread
8007e178 bne- ->0x8007E1AC --> SelectThread
8007e17c addi r3, r31, 1824 SelectThread
8007e180 bl ->0x8007A268 --> OSSetCurrentContext
8007e184 bl ->0x8007AC38 --> OSEnableInterrupts
8007e188 lwz r0, -0x7230 (r13) SelectThread
8007e18c cmplwi r0, 0 SelectThread
8007e190 beq+ ->0x8007E188 --> SelectThread
8007e194 bl ->0x8007AC24 --> OSDisableInterrupts
8007e198 lwz r0, -0x7230 (r13) SelectThread
8007e19c cmplwi r0, 0 SelectThread
8007e1a0 beq+ ->0x8007E184 --> SelectThread
8007e1a4 addi r3, r31, 1824 SelectThread
8007e1a8 bl ->0x8007A430 --> OSClearContext
8007e1ac li r3, 0 SelectThread
8007e1b0 stw r3, -0x722C (r13) SelectThread
8007e1b4 lwz r0, -0x7230 (r13) SelectThread
8007e1b8 cntlzw r7, r0 SelectThread
Registers
r0 00000000 f0 c04d1a56c0000000 c04d1a56c0000000 IBAT0
r1 812f94b8 f1 401c1c4e20000000 401c1c4e20000000 IBAT1
r2 80220be0 f2 c059fc4aa0000000 c059fc4aa0000000 IBAT2
r3 00000000 f3 3fde0b4fa0000000 0000000000000000 IBAT3
r4 00009032 f4 8000000000000000 c058ede4e0000000 IBAT4
r5 812f4ce8 f5 c010e65cc0000000 c058ede4e0000000 IBAT5
r6 00001032 f6 c059fc4aa0000000 bfe0000000000000 IBAT6
r7 cc010000 f7 3fe0000000000000 3fe0000000000000 IBAT7
r8 00000008 f8 bfec192d80000000 bf9774bd60000000 DBAT0
r9 00000007 f9 bfde972e60000000 0000000000000000 DBAT1
r10 80208720 f10 8000000000000000 4005ce8800000000 DBAT2
r11 00000020 f11 4011350a20000000 4005ce8800000000 DBAT3
r12 80094030 f12 401c1c4e20000000 3f5000000000000 DBAT4
r13 8021fb80 f13 bf7e5fe1a0000000 c01219c000000000 DBAT5
r14 00000000 f14
r15 00000000 f15
r16 00000000 f16
r17 00000000 f17
r18 00000000 f18
r19 00000000 f19
r20 00000000 f20
r21 00000000 f21
r22 00000000 f22
r23 00000000 f23
r24 00000000 f24
r25 00000000 f25
r26 00000009 f26
r27 00000001 f27
r28 80204da0 f28
r29 00000011 f29
r30 00000000 f30
r31 80207458 f31
Banner Title Maker Size
どうぶつの森e+ どう... Nin... 1.36 GiB
Luigi's Mansion Lui... 200... 1.36 GiB
Animal Crossing Ani... 200... 1.36 GiB
Animal Crossing Ani... 200... 1.36 GiB
Memory
ffffffe0 - - - - - - - - - - - - - - - -
fffffff0 - - - - - - - - - - - - - - - -
00000000 - - - - - - - - - - - - - - - -
00000010 - - - - - - - - - - - - - - - -
Search Address
Value
ASCII
Hex
Log
Word Wrap
Default Font
Clear
41:24:091 core\hle\hle_os.cpp:85 N[OSREPORT]: 80005b28->80005b14| sizeof(nintendo_hi_0)=00009900
41:24:092 core\hle\hle_os.cpp:85 N[OSREPORT]: 8005a724->8005a718| sizeof(nintendo_hi_0)=00009900
41:24:092 core\hle\hle_os.cpp:123 N[OSREPORT]: 8009c5f0->8009fcd8| sizeof(nintendo_hi_0)=00009900
41:24:092 core\hw\exi\exi_deviceipl.cpp:302 N[OSREPORT]: sizeof(nintendo_hi_0)=00009900
41:24:092 core\hle\hle_os.cpp:85 N[OSREPORT]: 80005b38->80005b28| 実際のnintendo_hi_0.awのサイズ=000066a0
41:24:092 core\hle\hle_os.cpp:85 N[OSREPORT]: 8005a724->8005a718| 実際のnintendo_hi_0.awのサイズ=000066a0
41:24:092 core\hle\hle_os.cpp:123 N[OSREPORT]: 8009c5f0->8009fcd8| 実際のnintendo_hi_0.awのサイズ=000066a0
JIT64 DC | OpenGL | HLE | FPS: 60 - VPS: 60 - 100% | Animal Crossing (GAFE...
Animal Crossing™
Press START!
©2001, 2002 Nintendo

Background

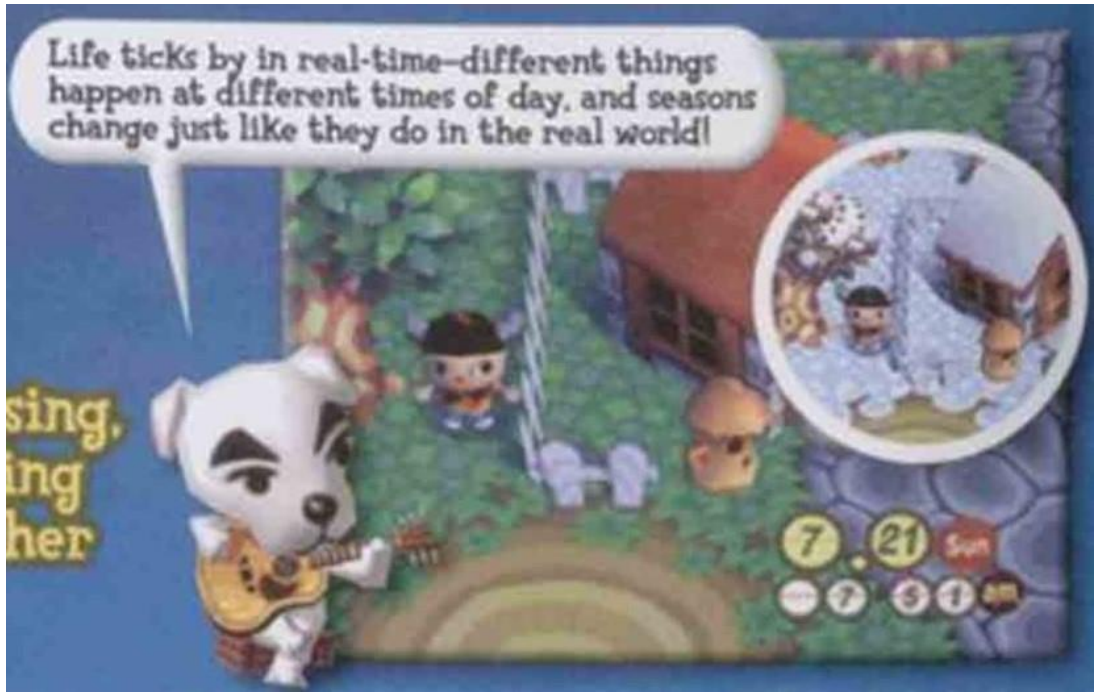
- Presented some reverse engineering work last year on Animal Crossing
- Hoping to find save game exploits
  - Used for modding games on GameCube, older consoles
  - Used for jailbreaking on newer consoles, e.g. 3DS
- Found hidden feature for loading NES ROMs from the memory card
  - ROM metadata format had a “patch” feature that could be used for arbitrary code execution




JUTReport 1337
msg no :12395 D
Hello world!
Even more text!
Z
Lights
Camera
time: 2372463449935925 1

Background

- Wanted to find an exploit that could run earlier
  - as close to save load as possible
- There is a big chunk of data copied directly from the save file to memory that contains global game state
- A lot of the data is processed during startup while the game simulates events that happened while you were away



Background

- Common pattern for state machine type abstractions in the game is to use a state value as an index into a function table
- Many of these indices are not bounds checked before being used to load a function pointer from the table
  - Could load unintended integers in memory as code addresses
- Some of the indices are located in the area loaded from the memory card

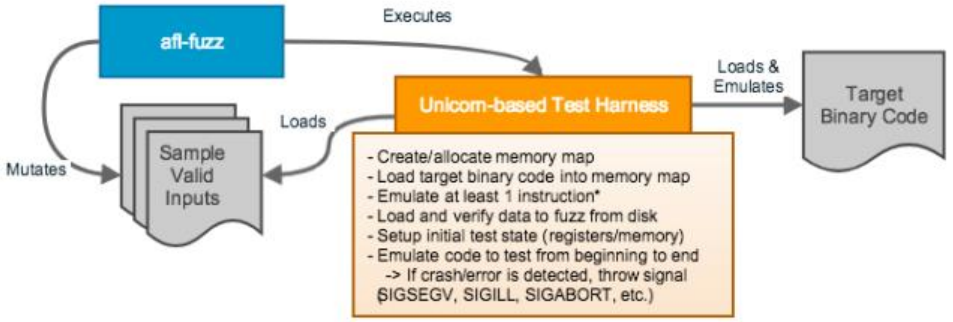
```
.globl Kabu_decide_price_without_sunday # weak
Kabu_decide_price_without_sunday:

.set arg_4,  4

stwu      r1, -0x10(r1) # Store Word with Update
mfspr   r0, LR          # Move from sprg,
lis       r4, common_data@ha # Load Immediate Shifted
lis       r3, process_454@h # Load Immediate Shifted
addi      r4, r4, common_data@l # Add Immediate
stw       r0, 0x10+arg_4(r1) # Store Word
addis     r4, r4, 2     # Add Immediate Shifted
addi      r3, r3, process_454@l # Add Immediate
lhz       r0, (common_data+0x48E - common_data)(r4) # Load Half Word and Zero
slwi      r0, r0, 2     # Shift Left Immediate
lwzx      r12, r3, r0   # Load Word and Zero Indexed
mtspr   CTR, r12        # Move to sprg,
bctrl                   # Branch unconditionally
lwz       r0, 0x10+arg_4(r1) # Load Word and Zero
mtspr   LR, r0          # Move to sprg,
addi      r1, r1, 0x10  # Add Immediate
blr                     # Branch unconditionally
```

Background

- This pattern is used everywhere
- Don’t want to analyze every function that looks like it might grab an index from save data
- Most tables only have a dozen entries at most…
  - If only I could do a fuzz test setting each byte in the save file to 0xFF and hope it crashes when used as an index

Background

- Used afl-unicorn before, but it requires a lot of set up and analysis to get emulation working correctly
  - Iterative process of weeding out false positive crashes caused by emulation before fuzzer is useful
- Dolphin project already did the hard work of implementing emulation, I wish I could just use that to fuzz the game
  - Use save states and debugger to implement simple fuzzer?



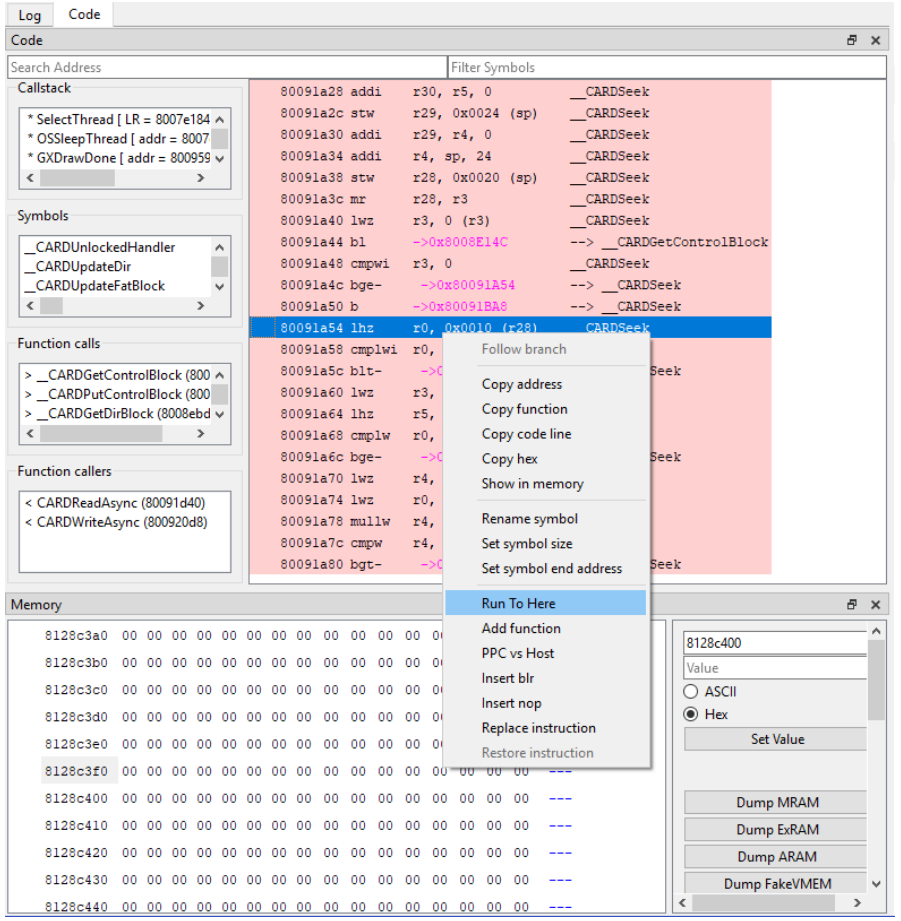
The only addition to normal AFL use is the Unicorn-based test harness

https://hackernoon.com/afl-unicorn-fuzzing-arbitrary-binary-code-563ca28936bf

Building the fuzzer

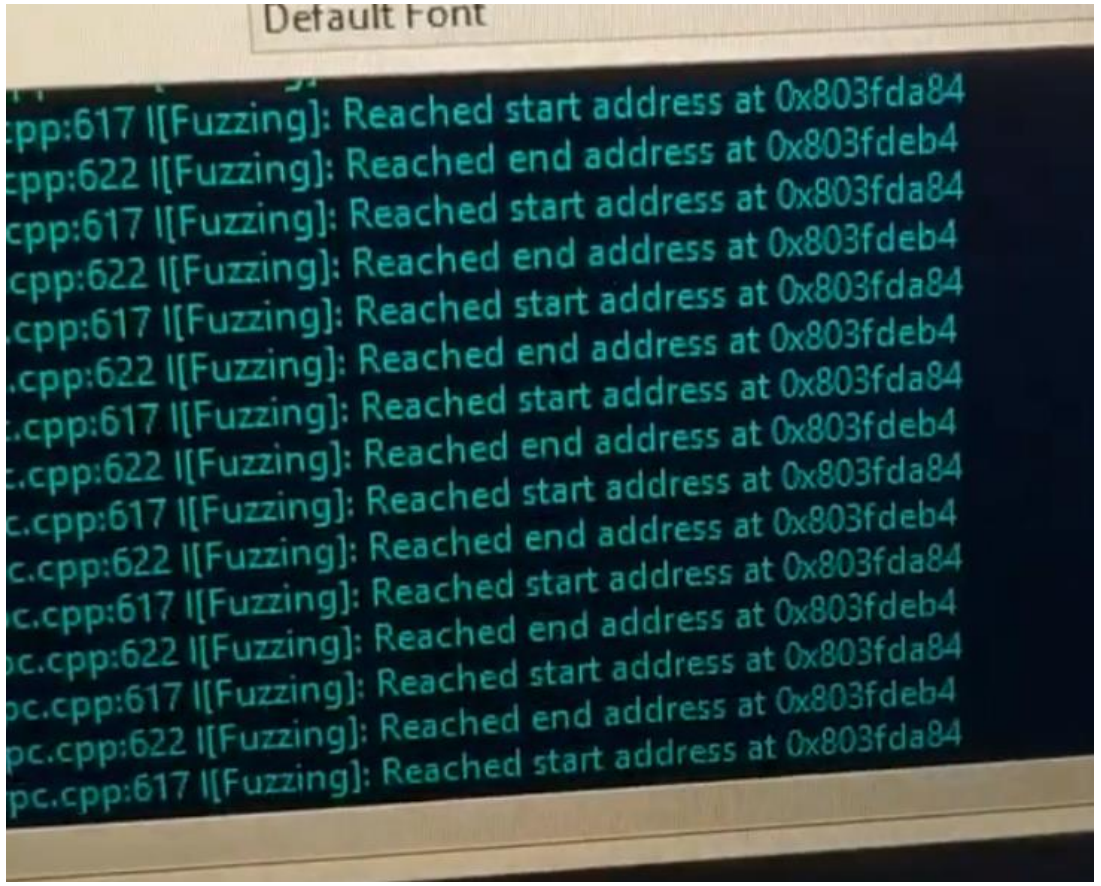
Building the fuzzer

- Start by looking at how breakpoints work in the debugger
  - Fuzzing start/stop location will work similarly and have the same kind of user interface
- Three emulation modes:
  - Interpreted
  - Cached interpreter
  - JIT recompile
- Dolphin switches between modes during execution, e.g. when hitting a breakpoint and going into step mode
  - Switches to interpreted mode



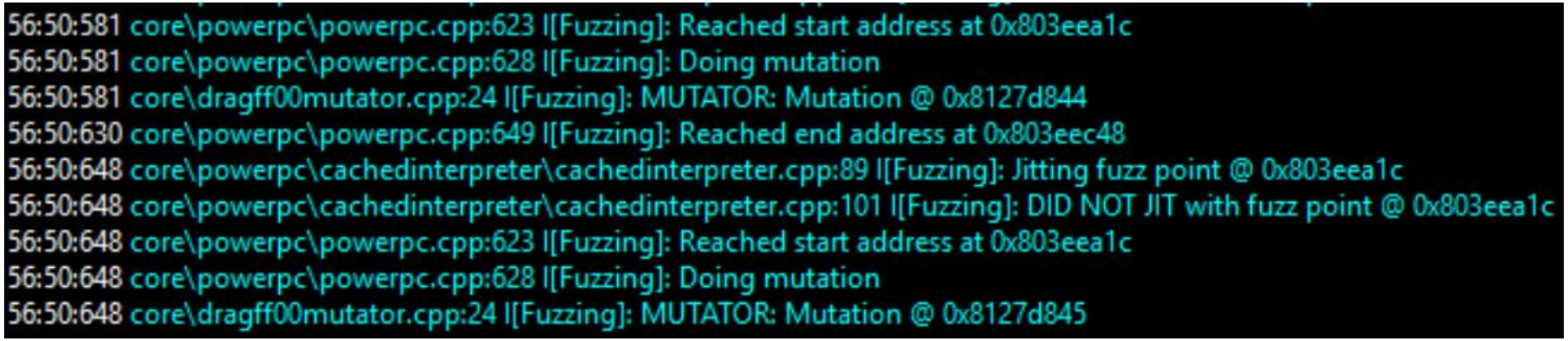
Building the fuzzer

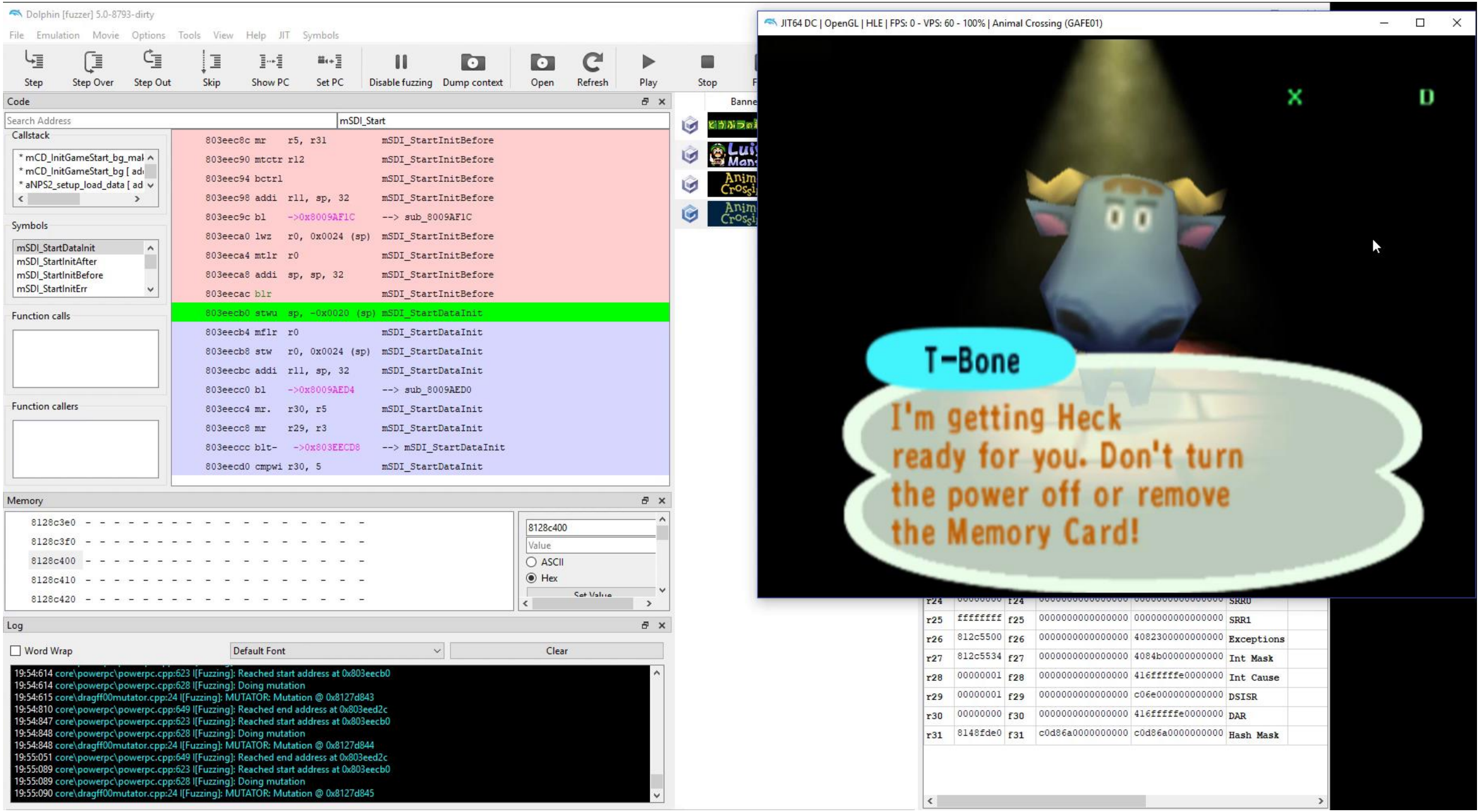
- Added checks similar to “is breakpoint set here” to check for “fuzz points”
- Create in-memory save state when hitting the fuzz start point and then reload it after hitting the fuzz stop point
- After working out threading and synchronization issues, got initial proof of concept working where I could repeatedly restart and run a section of code



Building the fuzzer

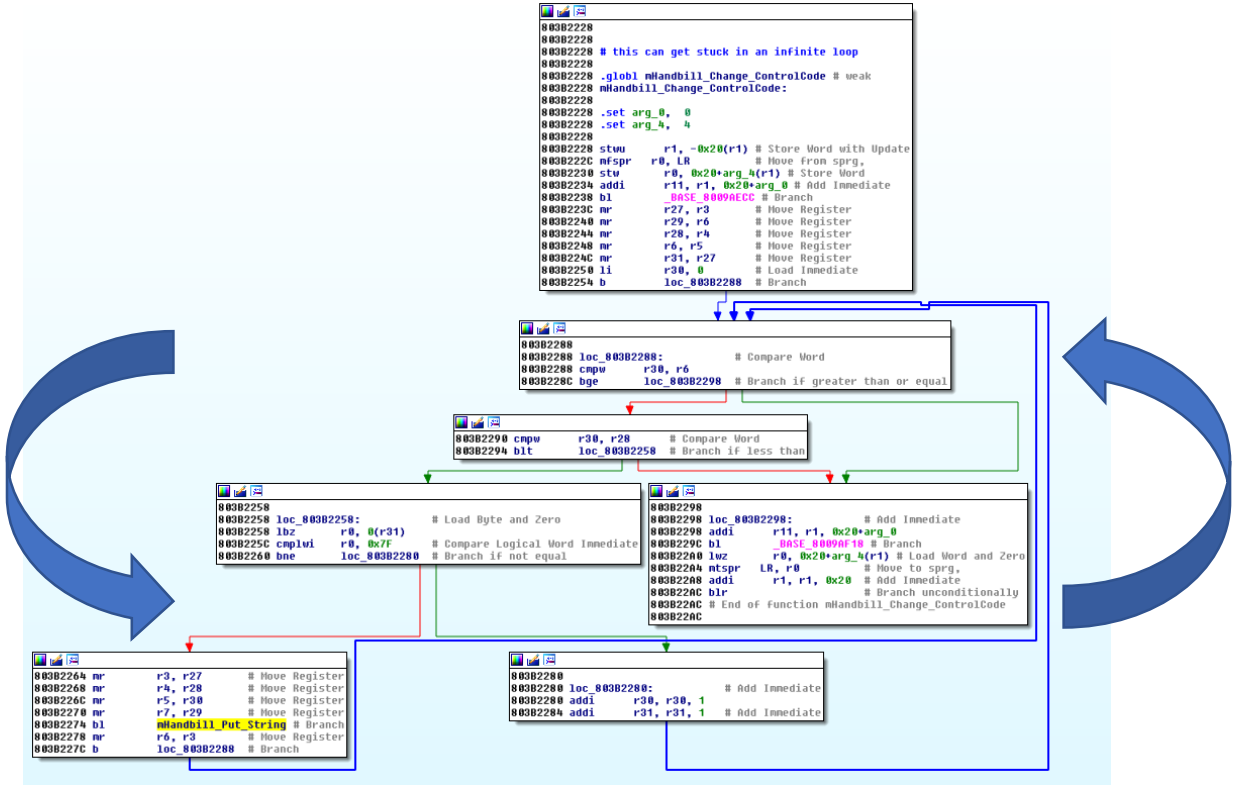
- Need to mutate save data: let user select memory region to operate on
- Set up a fuzzing state machine to handle mutating data each time the target code runs
  1. Fuzzer hits start point, creates initial save state
  2. Asks mutators for next mutation on selected memory region and applies it
  3. Runs the code until stop point is reached
  4. Stop execution, load save state and repeat
- Mutators implemented by user
  - First one just changed each byte in selected region to 0xFF, one at a time




Dolphin [fuzzer] 5.0-8793-dirty
File Emulation Movie Options Tools View Help JIT Symbols
Step Step Over Step Out Skip Show PC Set PC Disable fuzzing Dump context Open Refresh Play Stop
Code
Search Address
mSDI_Start
Callstack
* mCD_InitGameStart_bg_mal
* mCD_InitGameStart_bg [ad
* aNPS2_setup_load_data [ad
Symbols
mSDI_StartDataInit
mSDI_StartInitAfter
mSDI_StartInitBefore
mSDI_StartInitErr
Function calls
Function callers
803eec8c mr r5, r31 mSDI_StartInitBefore
803eec90 mtctr r12 mSDI_StartInitBefore
803eec94 bctrl mSDI_StartInitBefore
803eec98 addi r11, sp, 32 mSDI_StartInitBefore
803eec9c bl ->0x8009AF1C --> sub_8009AF1C
803eeca0 lwz r0, 0x0024 (sp) mSDI_StartInitBefore
803eeca4 mtlr r0 mSDI_StartInitBefore
803eeca8 addi sp, sp, 32 mSDI_StartInitBefore
803eecac blr mSDI_StartInitBefore
803eecb0 stwu sp, -0x0020 (sp) mSDI_StartDataInit
803eecb4 mflr r0 mSDI_StartDataInit
803eecb8 stw r0, 0x0024 (sp) mSDI_StartDataInit
803eecbc addi r11, sp, 32 mSDI_StartDataInit
803eecc0 bl ->0x8009AED4 --> sub_8009AED0
803eecc4 mr. r30, r5 mSDI_StartDataInit
803eecc8 mr r29, r3 mSDI_StartDataInit
803eeccc blt- ->0x803EECD8 --> mSDI_StartDataInit
803eecd0 cmpwi r30, 5 mSDI_StartDataInit
Memory
8128c3e0 - - - - - - - - - - - - - - - -
8128c3f0 - - - - - - - - - - - - - - - -
8128c400 - - - - - - - - - - - - - - - -
8128c410 - - - - - - - - - - - - - - - -
8128c420 - - - - - - - - - - - - - - - -
8128c400
Value
ASCII
Hex
Set Value
Log
Word Wrap
Default Font
Clear
19:54:614 core\powerpc\powerpc.cpp:623 I[Fuzzing]: Reached start address at 0x803eecb0
19:54:614 core\powerpc\powerpc.cpp:628 I[Fuzzing]: Doing mutation
19:54:615 core\dragff00mutator.cpp:24 I[Fuzzing]: MUTATOR: Mutation @ 0x8127d843
19:54:810 core\powerpc\powerpc.cpp:649 I[Fuzzing]: Reached end address at 0x803eed2c
19:54:847 core\powerpc\powerpc.cpp:623 I[Fuzzing]: Reached start address at 0x803eecb0
19:54:848 core\powerpc\powerpc.cpp:628 I[Fuzzing]: Doing mutation
19:54:848 core\dragff00mutator.cpp:24 I[Fuzzing]: MUTATOR: Mutation @ 0x8127d844
19:55:051 core\powerpc\powerpc.cpp:649 I[Fuzzing]: Reached end address at 0x803eed2c
19:55:089 core\powerpc\powerpc.cpp:623 I[Fuzzing]: Reached start address at 0x803eecb0
19:55:089 core\powerpc\powerpc.cpp:628 I[Fuzzing]: Doing mutation
19:55:090 core\dragff00mutator.cpp:24 I[Fuzzing]: MUTATOR: Mutation @ 0x8127d845
JIT64 DC | OpenGL | HLE | FPS: 0 - VPS: 60 - 100% | Animal Crossing (GAFE01)
T-Bone
I'm getting Heck ready for you. Don't turn the power off or remove the Memory Card!
r24 f24 SRR0
r25 ffffffff f25 0000000000000000 0000000000000000 SRR1
r26 812c5500 f26 0000000000000000 4082300000000000 Exceptions
r27 812c5534 f27 0000000000000000 4084b00000000000 Int Mask
r28 00000001 f28 0000000000000000 416fffffe0000000 Int Cause
r29 00000001 f29 0000000000000000 c06e000000000000 DSISR
r30 00000000 f30 0000000000000000 416fffffe0000000 DAR
r31 8148fde0 f31 c0d86a0000000000 c0d86a0000000000 Hash Mask

Building the fuzzer

- First bug it found was actually an infinite loop
  - Skipped over it in the debugger to continue fuzzing
- When it finally causes a crash, the emulator just halts when an exception occurs
- Good proof of concept, but not useful for automatic fuzzing yet



Building the fuzzer

- Luckily, Dolphin already had timer callbacks based on emulated system clock
  - Set a callback for half-second or one second's worth of clock ticks
  - If it's called while the test case is still running, consider it a timeout
- Also detects system exceptions based on standard PowerPC exception handling vectors
  - DSI: data memory access cannot be performed
  - ISI: instruction fetch cannot be performed

Vector Offset (hex)	Exception	
0 0000	Reserved	
0 0100	System Reset	Power-on, Hard & Soft Resets
0 0200	Machine Check	Eabled through MSR [ME]
0 0300	Data Access	Data Page Fault/Memory Protection
0 0400	Instruction Access	Instr. Page Fault/Memory Protection
0 0500	External Interrupt	$\overline{\text{INT}}$
0 0600	Alignment	Access crosses Segment or Page
0 0700	Program	Instr. Traps, Errors, Illegal, Privileged
0 0800	Floating-Point Unavailiable	MSR[FP]=0 & F.P. Instruction encountered
0 0900	Decrementer	Decrementer Register passes through 0
0 0A00	Reserved	
0 0B00	Reserved	
0 0C00	System Call	'sc' instruction
0 0D00	Trace	Single-step instruction trace
0 0E00	Floating-Point Assist	A floating-point exception

The basic PowerPC vector table

Building the fuzzer

- Use timeouts/exceptions to trigger handler in the fuzzer
- Saves serialized mutation patch, address of the instruction where the exception occurred, and copy of the original save state
  - This file can be used to reproduce and examine the crash in the Dolphin debugger

Example

Fuzzing Animal Crossing’s initial save data processing

Example

- 0x26000 bytes copied directly from save file to first part of a massive data structure called "`common_data`" – the global game state
- `mSDI_StartDataInit` function processes many pieces of `common_data` for initial startup of the game
  - Anything within the first 0x26000 bytes can probably be arbitrary user input

```
.globl Kabu_decide_price_without_sunday # weak
Kabu_decide_price_without_sunday:

.set arg_4,  4

stwu      r1, -0x10(r1) # Store Word with Update
mfspr   r0, LR          # Move from sprg,
lis       r4, common_data@ha # Load Immediate Shifted
lis       r3, process_454@h # Load Immediate Shifted
addi      r4, r4, common_data@l # Add Immediate
stw       r0, 0x10+arg_4(r1) # Store Word
addis     r4, r4, 2     # Add Immediate Shifted
addi      r3, r3, process_454@l # Add Immediate
lhz       r0, (common_data+0x48E - common_data)(r4) # Load Half Word and Zero
slwi      r0, r0, 2     # Shift Left Immediate
lwzx      r12, r3, r0   # Load Word and Zero Indexed
mtspr   CTR, r12        # Move to sprg,
bctrl                   # Branch unconditionally
lwz       r0, 0x10+arg_4(r1) # Load Word and Zero
mtspr   LR, r0          # Move to sprg,
addi      r1, r1, 0x10  # Add Immediate
blr                     # Branch unconditionally
```

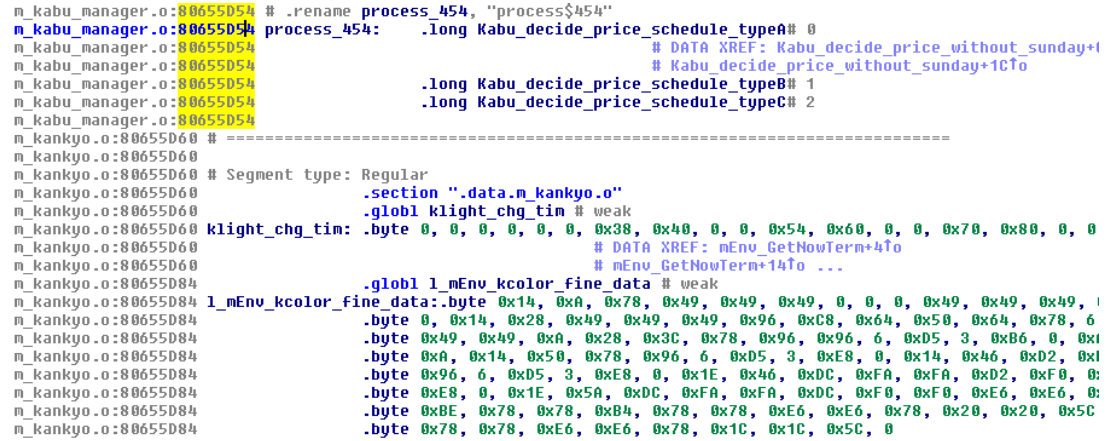
- Get 16 bit index from `common_data + 0x2048E`
- Load func pointer from `process_454[index]`
  - No bounds check on index
- Call function

`process_454` function table with 3 entries

```
m_kabu_manager.o:80655D54 # .rename process_454, "process$454"
m_kabu_manager.o:80655D54 process_454:    .long Kabu_decide_price_schedule_typeA# 0
m_kabu_manager.o:80655D54                                       # DATA XREF: Kabu_decide_price_without_sunday+
m_kabu_manager.o:80655D54                                       # Kabu_decide_price_without_sunday+1C↑o
m_kabu_manager.o:80655D54                 .long Kabu_decide_price_schedule_typeB# 1
m_kabu_manager.o:80655D54                 .long Kabu_decide_price_schedule_typeC# 2
m_kabu_manager.o:80655D54
m_kankyo.o:80655D60 # ============================================================================
m_kankyo.o:80655D60
m_kankyo.o:80655D60 # Segment type: Regular
m_kankyo.o:80655D60                 .section ".data.m_kankyo.o"
m_kankyo.o:80655D60                 .globl klight_chg_tim # weak
m_kankyo.o:80655D60 klight_chg_tim: .byte 0, 0, 0, 0, 0, 0, 0x38, 0x40, 0, 0, 0x54, 0x60, 0, 0, 0x70, 0x80, 0, 0
m_kankyo.o:80655D60                                       # DATA XREF: mEnv_GetNowTerm+4↑o
m_kankyo.o:80655D60                                       # mEnv_GetNowTerm+14↑o ...
m_kankyo.o:80655D84                 .globl l_mEnv_kcolor_fine_data # weak
m_kankyo.o:80655D84 l_mEnv_kcolor_fine_data:.byte 0x14, 0xA, 0x78, 0x49, 0x49, 0x49, 0, 0, 0, 0x49, 0x49, 0x49,
m_kankyo.o:80655D84                 .byte 0, 0x14, 0x28, 0x49, 0x49, 0x49, 0x96, 0xC8, 0x64, 0x50, 0x64, 0x78, 6
m_kankyo.o:80655D84                 .byte 0x49, 0x49, 0xA, 0x28, 0x3C, 0x78, 0x96, 0x96, 6, 0xD5, 3, 0xB6, 0, 0x
m_kankyo.o:80655D84                 .byte 0xA, 0x14, 0x50, 0x78, 0x96, 6, 0xD5, 3, 0xE8, 0, 0x14, 0x46, 0xD2, 0x
m_kankyo.o:80655D84                 .byte 0x96, 6, 0xD5, 3, 0xE8, 0, 0x1E, 0x46, 0xDC, 0xFA, 0xFA, 0xD2, 0xF0, 0
m_kankyo.o:80655D84                 .byte 0xE8, 0, 0x1E, 0x5A, 0xDC, 0xFA, 0xFA, 0xDC, 0xF0, 0xF0, 0xE6, 0xE6, 0
m_kankyo.o:80655D84                 .byte 0xBE, 0x78, 0x78, 0xB4, 0x78, 0x78, 0xE6, 0xE6, 0x78, 0x20, 0x20, 0x5C
m_kankyo.o:80655D84                 .byte 0x78, 0x78, 0xE6, 0xE6, 0x78, 0x1C, 0x1C, 0x5C, 0
```

Example

- Function table index variables that are not bounds checked should be easy to find by setting their value to 0xFF
  - Could be char/word/long, so have to try 0xFF, 0x00FF, 0x000000FF at each position
- Game will most likely crash when invalid function pointer is loaded

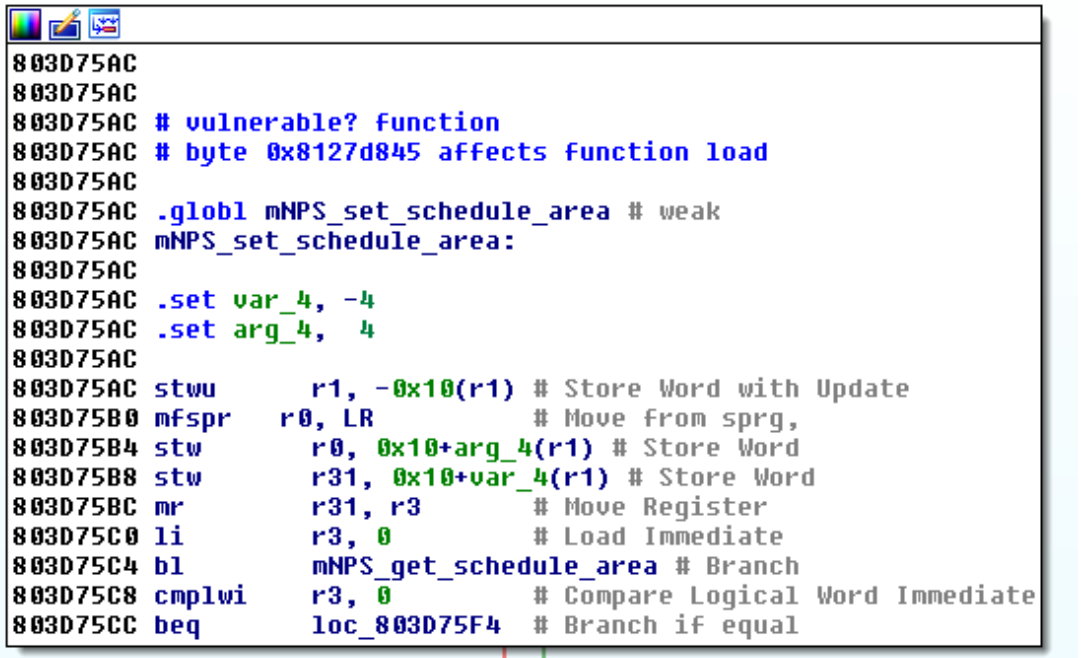


Example

- Write a mutator to apply 0xFF mutation at each position in the selected memory region
- Set fuzzer start address to first instruction of mSDI_StartDataInit, end address to last instruction
- Set fuzzer memory region start to beginning of common_data, end to common_data+0x26000

```
bool DragFF00Mutator::HasMoreMutations()
{
  return m_fuzz_position < GetRegionEnd();
}

std::shared_ptr<MutatorPatch> DragFF00Mutator::DoNextMutation()
{
  if (!IsInitialized()) return nullptr;

  INFO_LOG(FUZZING, "MUTATOR: Mutation @ 0x%x",
    m_fuzz_position);

  // Change the byte at current position
  void* patch_buf = malloc(1);
  if (patch_buf == nullptr) return nullptr;
  memset(patch_buf, 0xff, 1);

  std::shared_ptr<MutatorPatch> new_patch(
    new MutatorPatch(m_fuzz_position, patch_buf, 1));

  m_fuzz_position++;

  return new_patch;
}

void DragFF00Mutator::InitializeInner()
{
  m_fuzz_position = GetRegionStart();
}

void DragFF00Mutator::ResetInner()
{
  m_fuzz_position = 0;
}
```

demo



Example crash case

- Gets function pointer from `mNPS_schedule` table based on value in `common_data`
- The maximum address it can be tricked to load the function pointer from is `8065B7C0 + (0xFF x 4) = 8065BBBC`
- `common_data` load area is `81266400 - 8128C400`
  - Can’t easily influence branch address
- Does show ability to discover unsafe function table lookups

Future work

- Improve speed
  - Currently 1-2 executions per second
  - Doesn’t maximize CPU usage
- Use JIT to implement code coverage feedback
  - Work towards afl-like zero configuration fuzzing (in terms of data mutation)
- Auto-detect location of save data in memory after it’s loaded from the emulated memory card
  - Ease initial analysis of code and memory regions to target

Questions?

https://github.com/jamchamb