R2WARS FOR N00BS

BY @CAPTNBANANA

WHO R U MAN

- I do computer stuff!
- I do r2 stuff!
- I spawn calc with r2
- https://bananamafia.dev/tags/r2/

COREWAR

- You program warriors/bots
- They fight each other
- Goal: Cause enemy to crash

COREWAR

- Initial release in 1984
- Runs in MARS (Memory Array Redcode Simulator)
- Multi threading possible

R2WARS IN ONE SLIDE

- Bots written in x86-32/ARM64/ARM32/MIPS ASM
- Arena: 1024 bytes of RXW shared memory
- Separate Stack
- Goal: Cause enemy to crash
  - Corrupt instruction pointer
  - Cause invalid reads/writes
  - Trigger any other exception

BACKGROUND INFORMATION

- Developed by skuater
- Runs in r2's ESIL

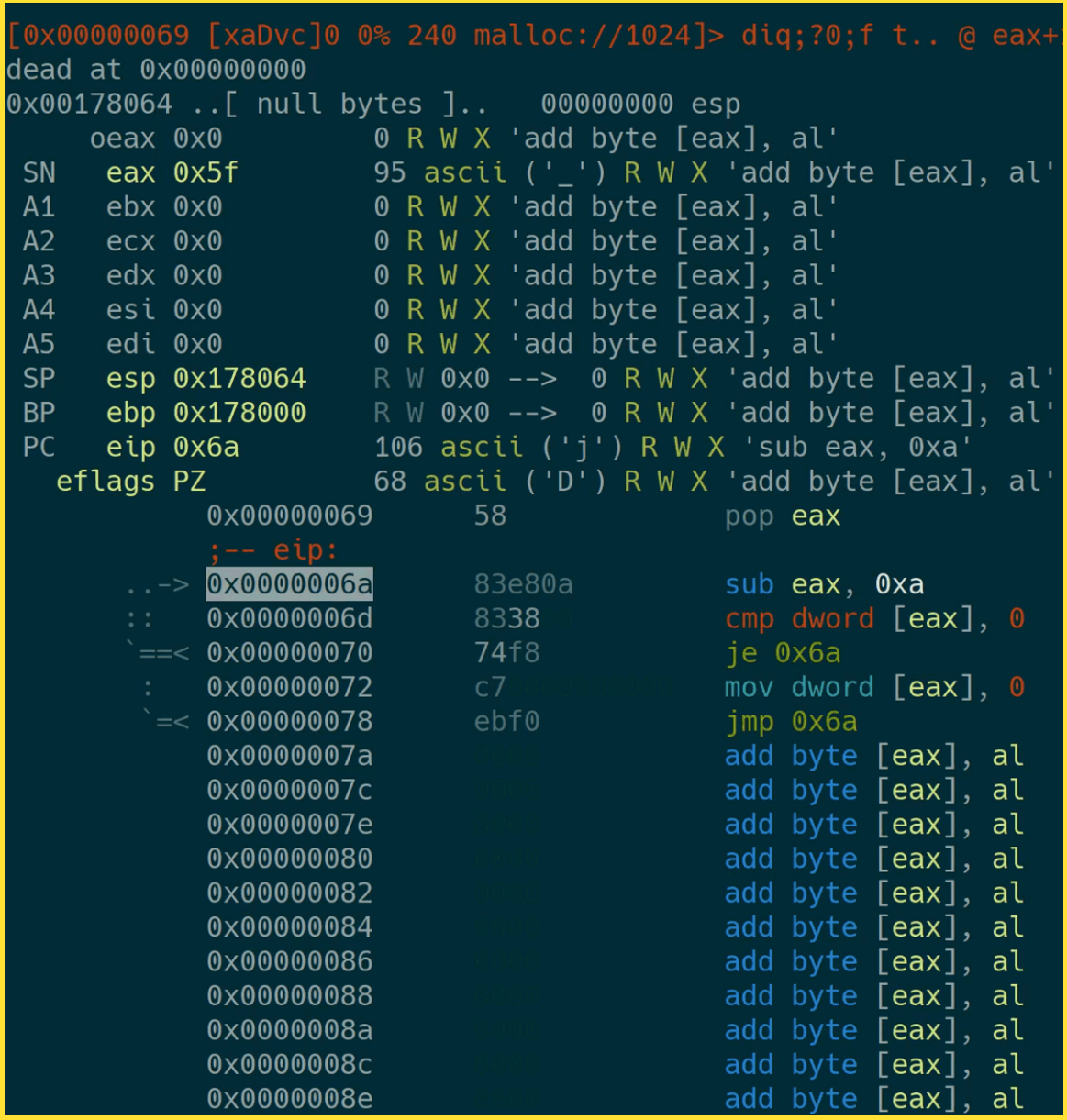
ESIL?

- ESIL = Evaluable Strings Intermediate Language
- Evaulate --> Emulate
- More from @arnaugamez: A Journey Through ESIL (2019)

RUNNING A BOT IN ESIL

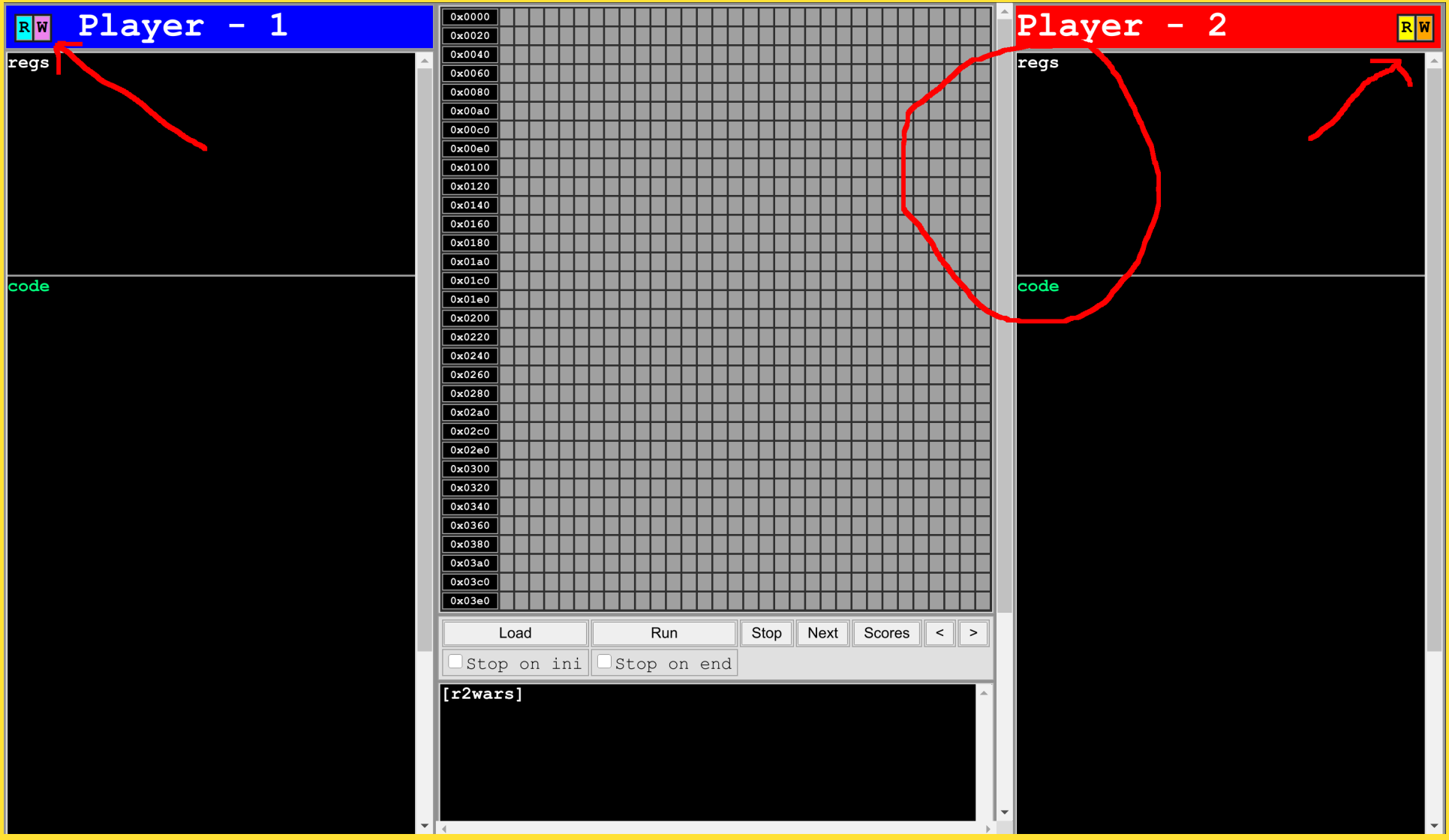
Example for x86-32

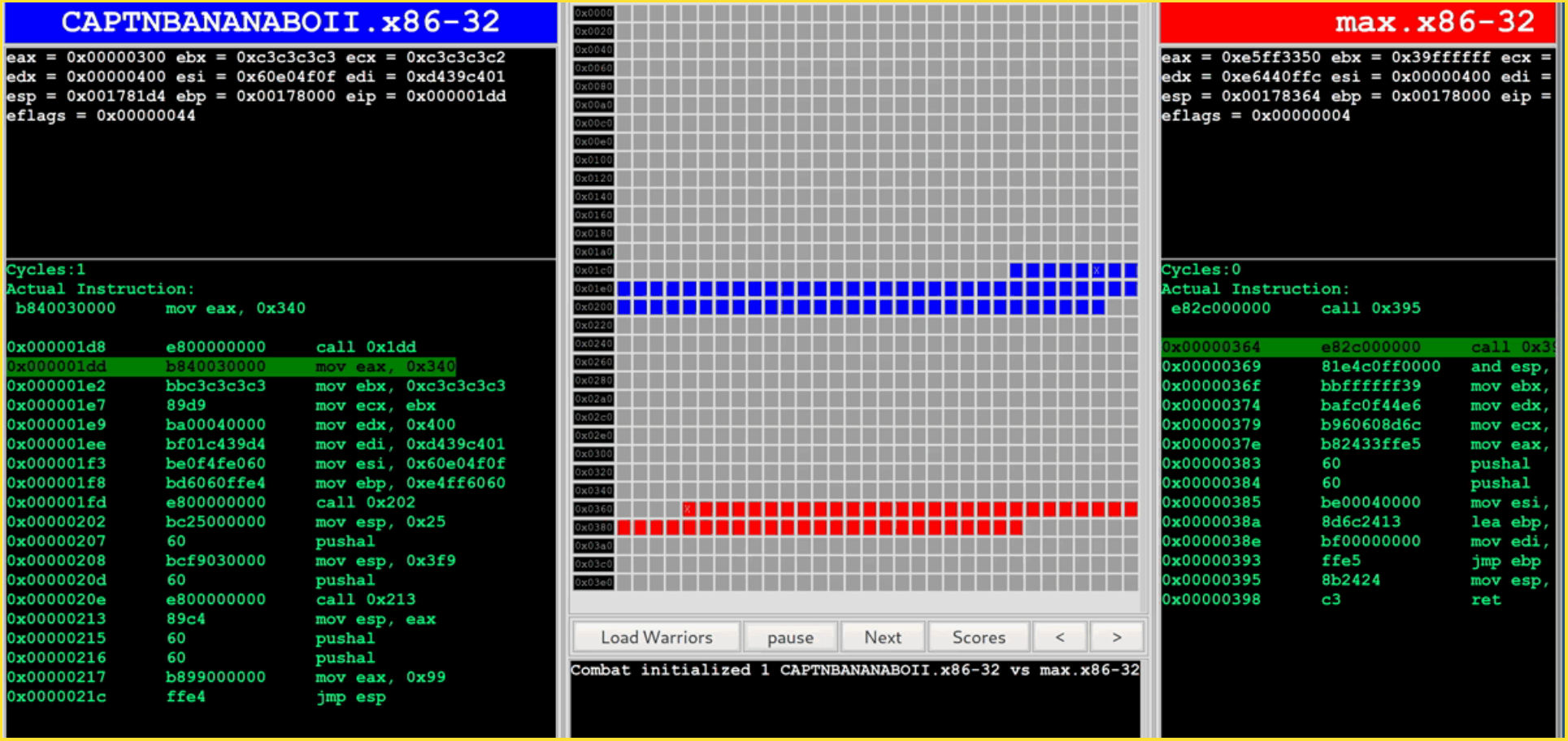
```
$ radare2 -c "
    e asm.arch=x86;
    e asm.bits=32;
    aei; # Initialize ESIL
    aeim; # Initialize even harder!
    wx $( # Assemble and write bot code
        rasm2 -a x86 -b 32 -f yoloBot.x86-32.asm
    ) @100;
    aer PC=100;
    aer SP=SP+100;"
    malloc://1024
```


[0x00000069 [xaDvc]0 0% 240 malloc://1024]> diq;?0;f t.. @ eax+
dead at 0x00000000
0x00178064 ..[null bytes]..   00000000 esp
oeax 0x0 0 R W X 'add byte [eax], al'
SN eax 0x5f 95 ascii ('_') R W X 'add byte [eax], al'
A1 ebx 0x0 0 R W X 'add byte [eax], al'
A2 ecx 0x0 0 R W X 'add byte [eax], al'
A3 edx 0x0 0 R W X 'add byte [eax], al'
A4 esi 0x0 0 R W X 'add byte [eax], al'
A5 edi 0x0 0 R W X 'add byte [eax], al'
SP esp 0x178064 R W 0x0 --> 0 R W X 'add byte [eax], al'
BP ebp 0x178000 R W 0x0 --> 0 R W X 'add byte [eax], al'
PC eip 0x6a 106 ascii ('j') R W X 'sub eax, 0xa'
eflags PZ 68 ascii ('D') R W X 'add byte [eax], al'
0x00000069 58 pop eax
;-- eip:
..-> 0x0000006a 83e80a sub eax, 0xa
:: 0x0000006d 8338 cmp dword [eax], 0
`==< 0x00000070 74f8 je 0x6a
: 0x00000072 c7 mov dword [eax], 0
`=< 0x00000078 ebf0 jmp 0x6a
0x0000007a add byte [eax], al
0x0000007c add byte [eax], al
0x0000007e add byte [eax], al
0x00000080 add byte [eax], al
0x00000082 add byte [eax], al
0x00000084 add byte [eax], al
0x00000086 add byte [eax], al
0x00000088 add byte [eax], al
0x0000008a add byte [eax], al
0x0000008c add byte [eax], al
0x0000008e add byte [eax], al

RUNNING R2WARS

- Get radare2 (rlly!)
- Get Mono .NET Runtime
- `$ xbuild r2wars.csproj`
- `$ mono r2wars.exe`
- Open your browser

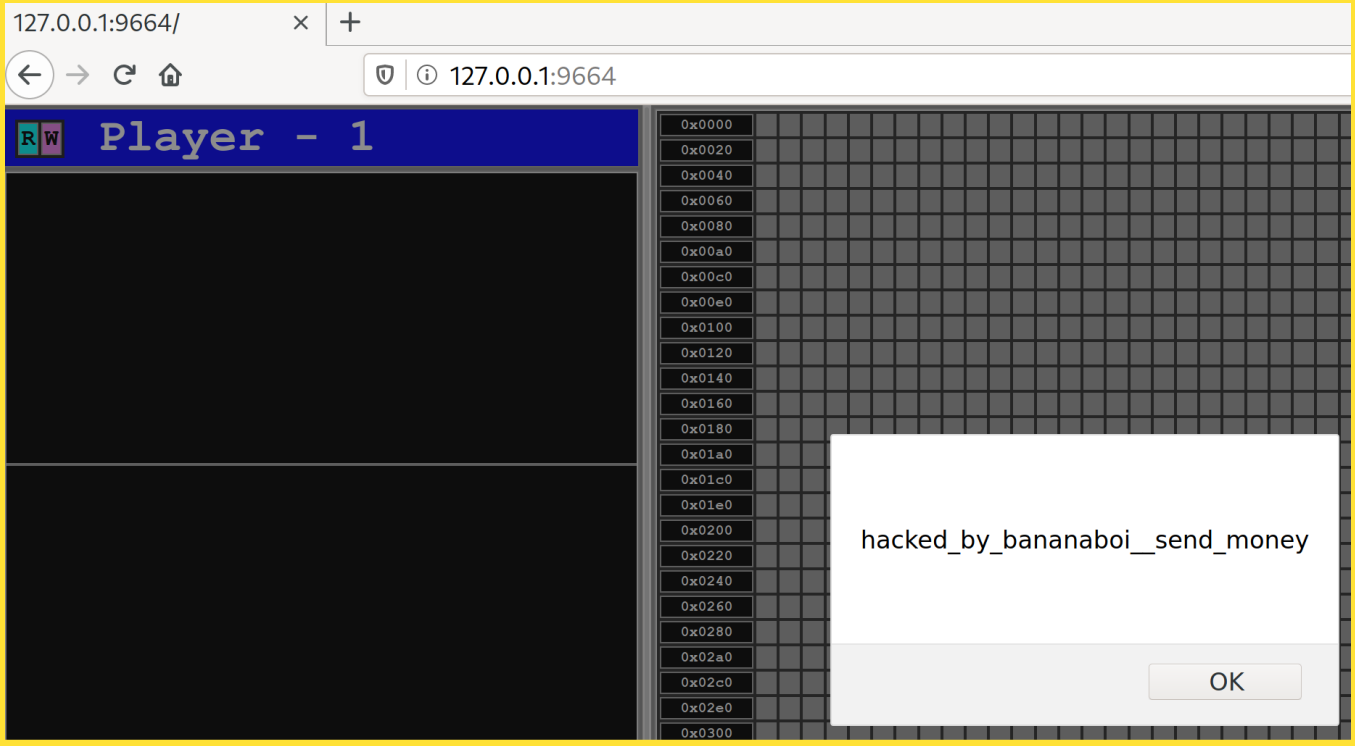

R W
Player - 1
regs
code
0x0000
0x0020
0x0040
0x0060
0x0080
0x00a0
0x00c0
0x00e0
0x0100
0x0120
0x0140
0x0160
0x0180
0x01a0
0x01c0
0x01e0
0x0200
0x0220
0x0240
0x0260
0x0280
0x02a0
0x02c0
0x02e0
0x0300
0x0320
0x0340
0x0360
0x0380
0x03a0
0x03c0
0x03e0
Load
Run
Stop
Next
Scores
<
>
Stop on ini
Stop on end
[r2wars]
Player - 2
R W
regs
code

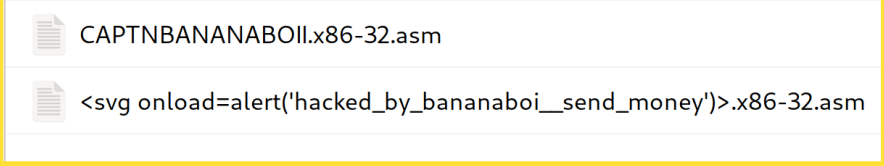

CAPTNBANANABOII.x86-32
eax = 0x00000300 ebx = 0xc3c3c3c3 ecx = 0xc3c3c3c2
edx = 0x00000400 esi = 0x60e04f0f edi = 0xd439c401
esp = 0x001781d4 ebp = 0x00178000 eip = 0x000001dd
eflags = 0x00000044
Cycles:1
Actual Instruction:
b840030000 mov eax, 0x340
0x000001d8 e800000000 call 0x1dd
0x000001dd b840030000 mov eax, 0x340
0x000001e2 bbc3c3c3c3 mov ebx, 0xc3c3c3c3
0x000001e7 89d9 mov ecx, ebx
0x000001e9 ba00040000 mov edx, 0x400
0x000001ee bf01c439d4 mov edi, 0xd439c401
0x000001f3 be0f4fe060 mov esi, 0x60e04f0f
0x000001f8 bd6060ffe4 mov ebp, 0xe4ff6060
0x000001fd e800000000 call 0x202
0x00000202 bc25000000 mov esp, 0x25
0x00000207 60 pushal
0x00000208 bcf9030000 mov esp, 0x3f9
0x0000020d 60 pushal
0x0000020e e800000000 call 0x213
0x00000213 89c4 mov esp, eax
0x00000215 60 pushal
0x00000216 60 pushal
0x00000217 b899000000 mov eax, 0x99
0x0000021c ffe4 jmp esp
Load Warriors
pause
Next
Scores
<
>
Combat initialized 1 CAPTNBANANABOII.x86-32 vs max.x86-32
max.x86-32
eax = 0xe5ff3350 ebx = 0x39ffffff ecx =
edx = 0xe6440ffc esi = 0x00000400 edi =
esp = 0x00178364 ebp = 0x00178000 eip =
eflags = 0x00000004
Cycles:0
Actual Instruction:
e82c000000 call 0x395
0x00000364 e82c000000 call 0x3
0x00000369 81e4c0ff0000 and esp,
0x0000036f bbffffff39 mov ebx,
0x00000374 bafc0f44e6 mov edx,
0x00000379 b960608d6c mov ecx,
0x0000037e b82433ffe5 mov eax,
0x00000383 60 pushal
0x00000384 60 pushal
0x00000385 be00040000 mov esi,
0x0000038a 8d6c2413 lea ebp,
0x0000038e bf00000000 mov edi,
0x00000393 ffe5 jmp ebp
0x00000395 8b2424 mov esp,
0x00000398 c3 ret

GOALS

- Create the nerdiest game ever
- Learn ASM
- Have fun and win beer
- Eternal fame for the winner
- Listen to pancake's selection of chiptune songs
- Find bugs in r2 and ESIL

OR BUGS IN R2WARS





GENERAL RULES

- You send in bots
- Each one will fight against the others: 1vs1
- Rank is based on wins
- Ever r2con day: 1 practice run
- Last day of r2con: Final run (win prizes!)

IMPORTANT THINGS TO KNOW

- Max 2000 cycles: Timeout
- IO and syscalls are ignored
- Program counter and stack pointer: Randomized
- Memory doesn't wrap around
- Turns based on instruction cost

CHOOSE YOUR ARCHITECTURE/STRATEGY

- You will compete vs. other architectures
- Different architectures: (Dis)Advantages
- - Suggestion: Choose x86-32 if you're not sure
  - Many examples available
- You will compete vs. various strategies
- Good bot: Works well against most of them

STRATEGIES

COREWAR STRATEGY GUIDE

- Opening/Midgame/Endgame Strategies
- Choose yours or come up with your own
- Note: There's no threading in r2wars (yet?)

IMP

- Loop: Copy current instruction to next address
- One task: Survival

SCANNER

```
call go

go:
    pop eax ; Get EIP

loop:
    add eax, 1
    cmp [eax], 0
    je loop
    mov [eax], 0xc3 ; ret
    jmp loop
```

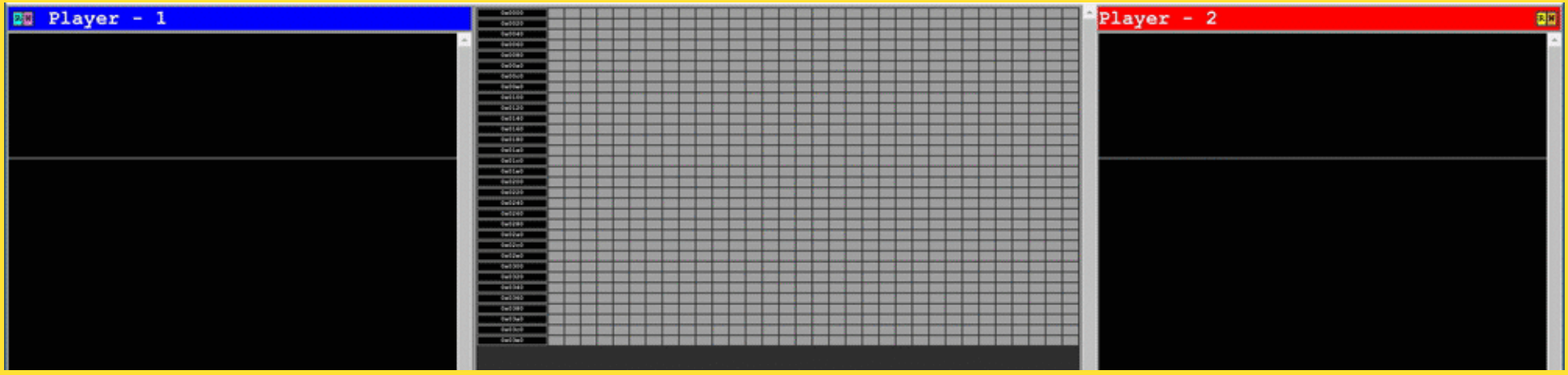
BETTER SCANNER

```
call go

go:
    pop eax ; Get EIP

loop:
    add eax, 10 ; <- changed
    cmp [eax], 0
    je loop
    mov [eax], 0xc3 ;ret
    jmp loop
```

EVEN BETTER SCANNER

```
call go

go:
    mov edx, 0x400
    mov ecx, 0x0
    pop eax
    add eax, 0x20 ; dont scan in own code

loop:
    add eax, 0x20
    cmp eax, edx
    cmovg eax, ecx ; enter arena from start - we are outside
    cmp [eax], 0
    je loop
```

SCANNER VS. SCANNER



INVALID INSTRUCTIONS

```
$ rasm2 -a x86 -b 32 -d 0000
add byte [eax], al

$ rasm2 -a arm -b 32 -d 0000
invalid
```

BOMBER

- Write "bombs" to random locations
- Beware: don't kill yourself

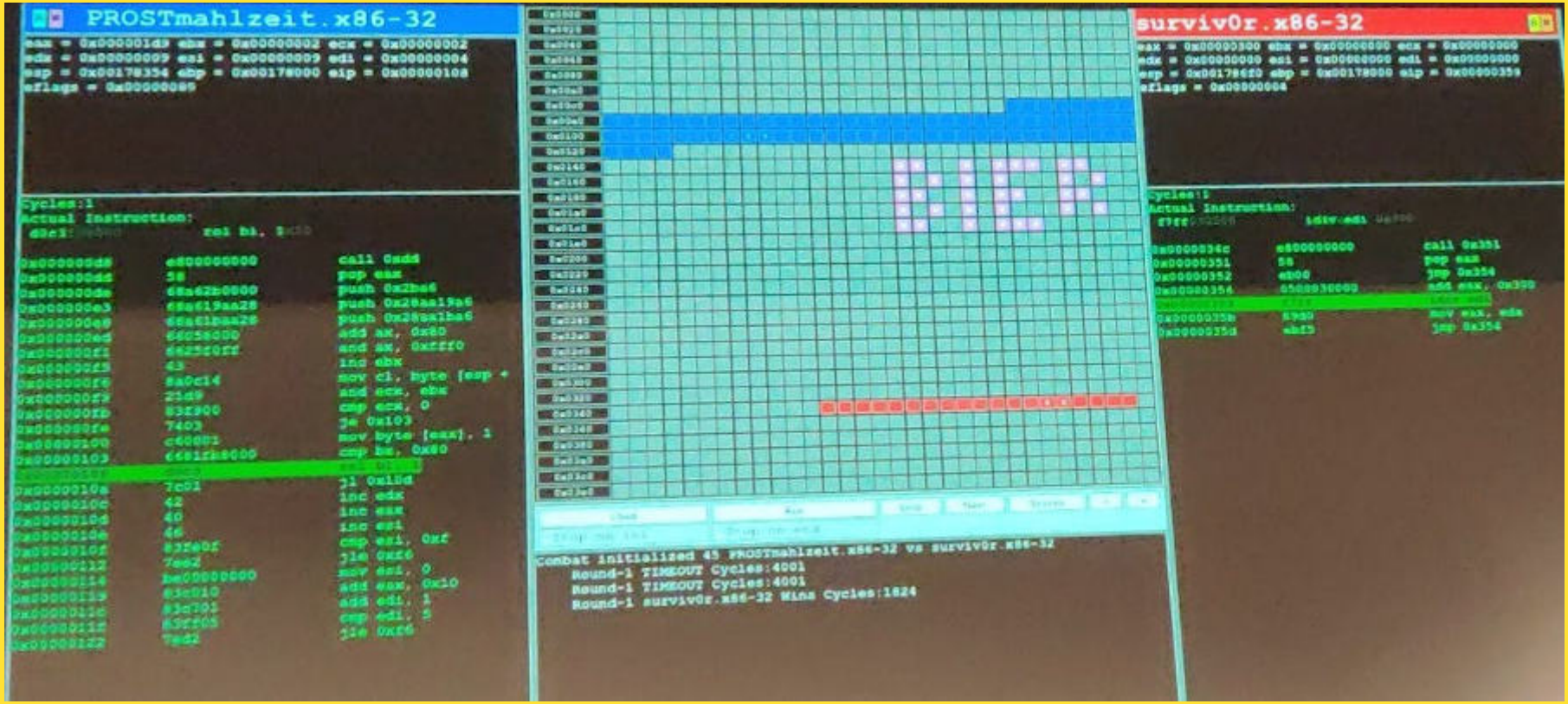
VAMPIRE

- Write JMP instructions
- "Capture" enemy

REPLICATOR

- Contains second stage code in registers
- Preparation: Load code into registers
- Loop:
  1. Push registers
  2. Execute pushed code: JMP ESP
  3. (Adjust stack pointer)

BEER STRATEGY (1337)



pic src: @Aissn

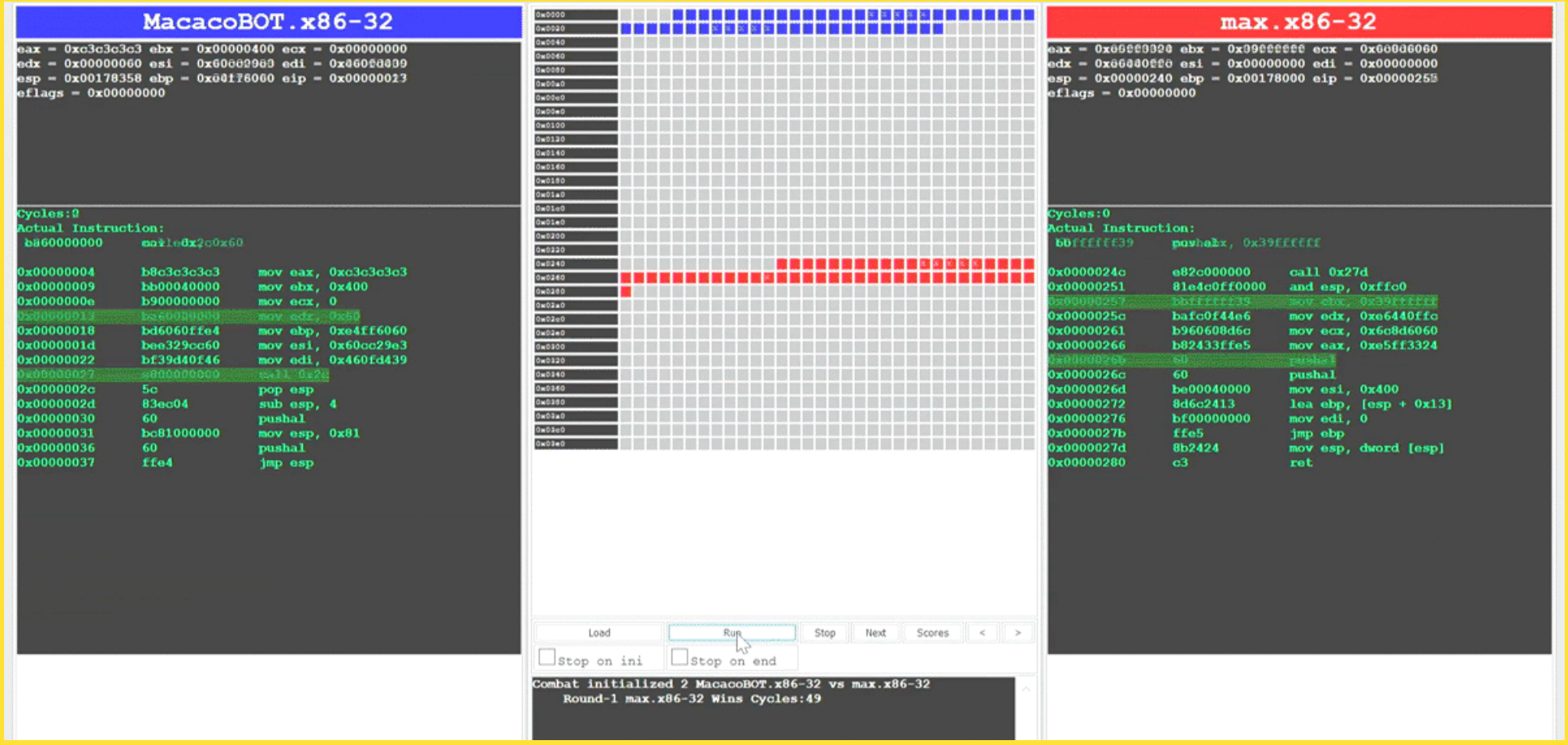
THINGS THAT MAKE BANANA SAD

- Suicidal bots
- Timeouts

RANDOM TIPS

- Combine various Strategies -> Stages
- Test against real bots: See references
- Find exploits, e.g. stack escape (fixed)
- Reverse your opponents: "The challenge for next year would be to automate this task with OCR or Google Lens"

TUNING


MacacoBOT.x86-32
eax = 0xc3c3c3c3 ebx = 0x00000400 ecx = 0x00000000
edx = 0x00000060 esi = 0x60cc29e3 edi = 0x460fd439
esp = 0x00178358 ebp = 0xe4ff6060 eip = 0x00000023
eflags = 0x00000000
Cycles:2
Actual Instruction:
0x00000004 b8c3c3c3c3 mov eax, 0xc3c3c3c3
0x00000009 bb00040000 mov ebx, 0x400
0x0000000e b900000000 mov ecx, 0
0x00000018 bd6060ffe4 mov ebp, 0xe4ff6060
0x0000001d bee329cc60 mov esi, 0x60cc29e3
0x00000022 bf39d40f46 mov edi, 0x460fd439
0x0000002c 5c pop esp
0x0000002d 83ec04 sub esp, 4
0x00000030 60 pushal
0x00000031 bc81000000 mov esp, 0x81
0x00000036 60 pushal
0x00000037 ffe4 jmp esp
Load
Run
Stop
Next
Scores
Stop on ini
Stop on end
Combat initialized 2 MacacoBOT.x86-32 vs max.x86-32
Round-1 max.x86-32 Wins Cycles:49
max.x86-32
eax = 0x00000000 ebx = 0x00000000 ecx = 0x60000060
edx = 0x00000000 esi = 0x00000000 edi = 0x00000000
esp = 0x00000240 ebp = 0x00178000 eip = 0x00000255
eflags = 0x00000000
Cycles:0
Actual Instruction:
0x0000024c e82c000000 call 0x27d
0x00000251 81e4c0ff0000 and esp, 0xffc0
0x0000025c bafc0f44e6 mov edx, 0xe6440ffc
0x00000261 b960608d6c mov ecx, 0x6c8d6060
0x00000266 b82433ffe5 mov eax, 0xe5ff3324
0x0000026c 60 pushal
0x0000026d be00040000 mov esi, 0x400
0x00000272 8d6c2413 lea ebp, [esp + 0x13]
0x00000276 bf00000000 mov edi, 0
0x0000027b ffe5 jmp ebp
0x0000027d 8b2424 mov esp, dword [esp]
0x00000280 c3 ret

METRICS

- Ratio: Average bytes written / cycle
- x86-32 `pushal` vs. ARM `stmia`

MOAR METRICS

- Count of cycles to first write
- Length of bot main loop
- Credits to Anisse

INSTRUCTION COST

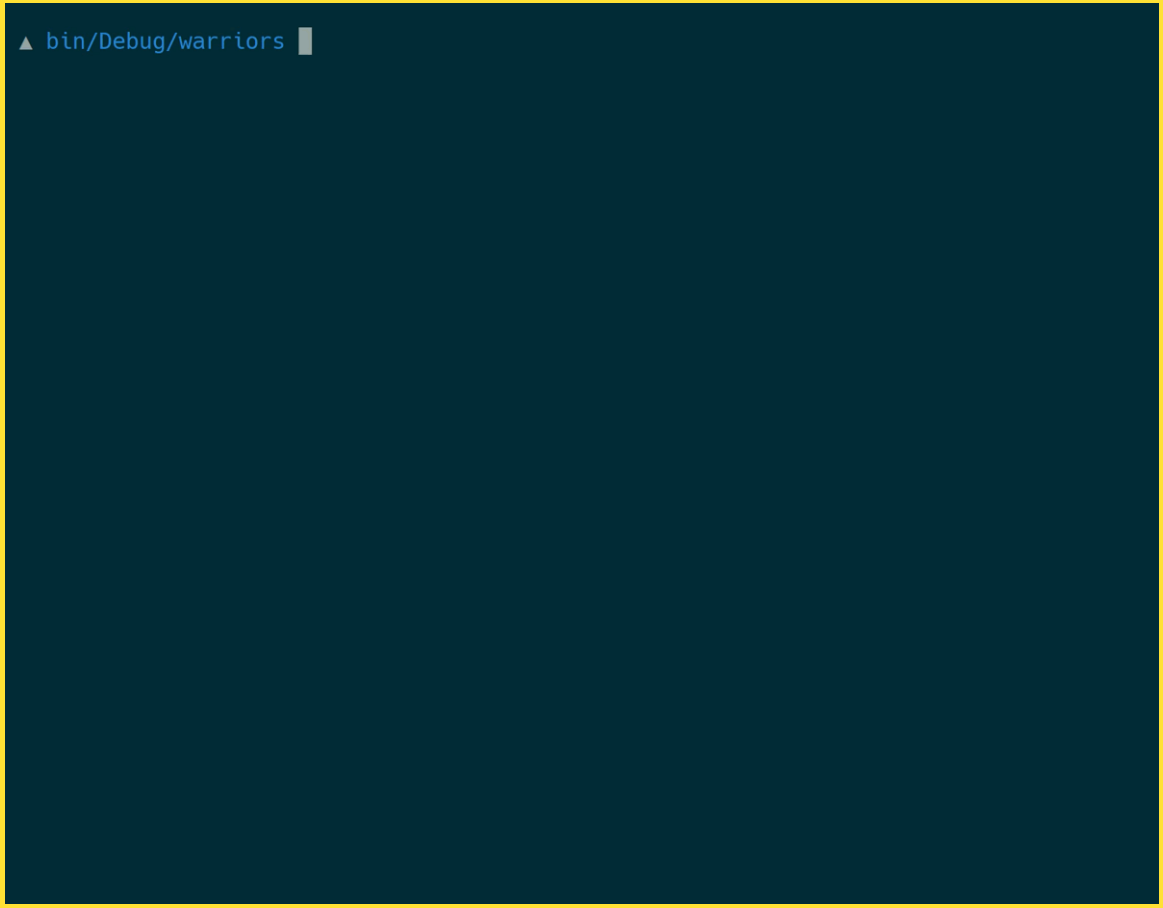
```
[0x0000007a]> pd 2
0x0000007a 60 pushal
0x0000007c ffe4 jmp esp

[0x0000007a]> ao 2~cycles
cycles: 1
cycles: 2
```

PROFILING

```
$ radare2 -c "
    e asm.arch=x86;
    e asm.bits=32; aei;
    aeim;
    wx $(rasm2 -a x86 -b 32 -f yoloBot.x86-32.asm) @100;
    aer PC=100;
    aer SP=SP+100;
    150ds; <- execute 150 instructions
    s 0x0; V\!"
    malloc://1024
```

PROFILING

Example: Memory contents after 50 instructions



HOW TO PARTICIPATE

- Join the r2wars channel
- Send in your bots in a Telegram DM to @sanguinawer
- (he's in the channel)
- Follow the naming scheme: See next slide

NAMING SCHEME

- You have to follow it
- Otherwise everything goes boom
- x86-32 bot: `cakepan.x86-32.asm`
- arm-32 bot: `muchos_fightos.arm-32.asm`

@CaptnBanana

REFERENCES

- r2wars Source Code
- My Blog Post
- CoreWar Strategy Guide
- Example Bots
- Initial r2wars PoC and even more bots
- Infos and Hints
- Writeups from @Aissn: 2018 and 2019
- Another writeup, from 2017