Montrehack.ca // 2022-04-21

NORTHSEC 2021 CTF Badge

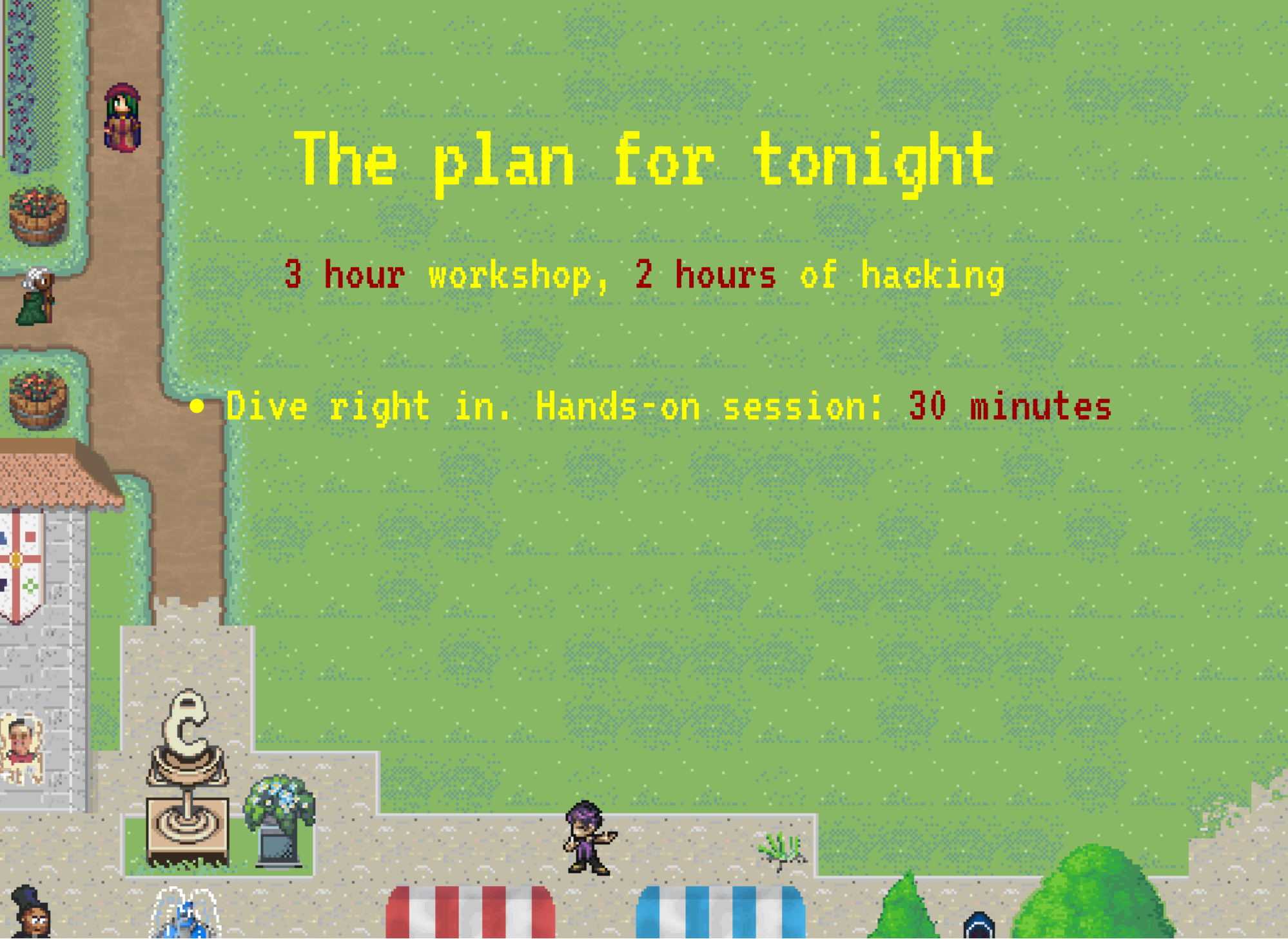
presented by the badge team


The plan for tonight
3 hour workshop, 2 hours of hacking

The plan for tonight

3 hour workshop, 2 hours of hacking

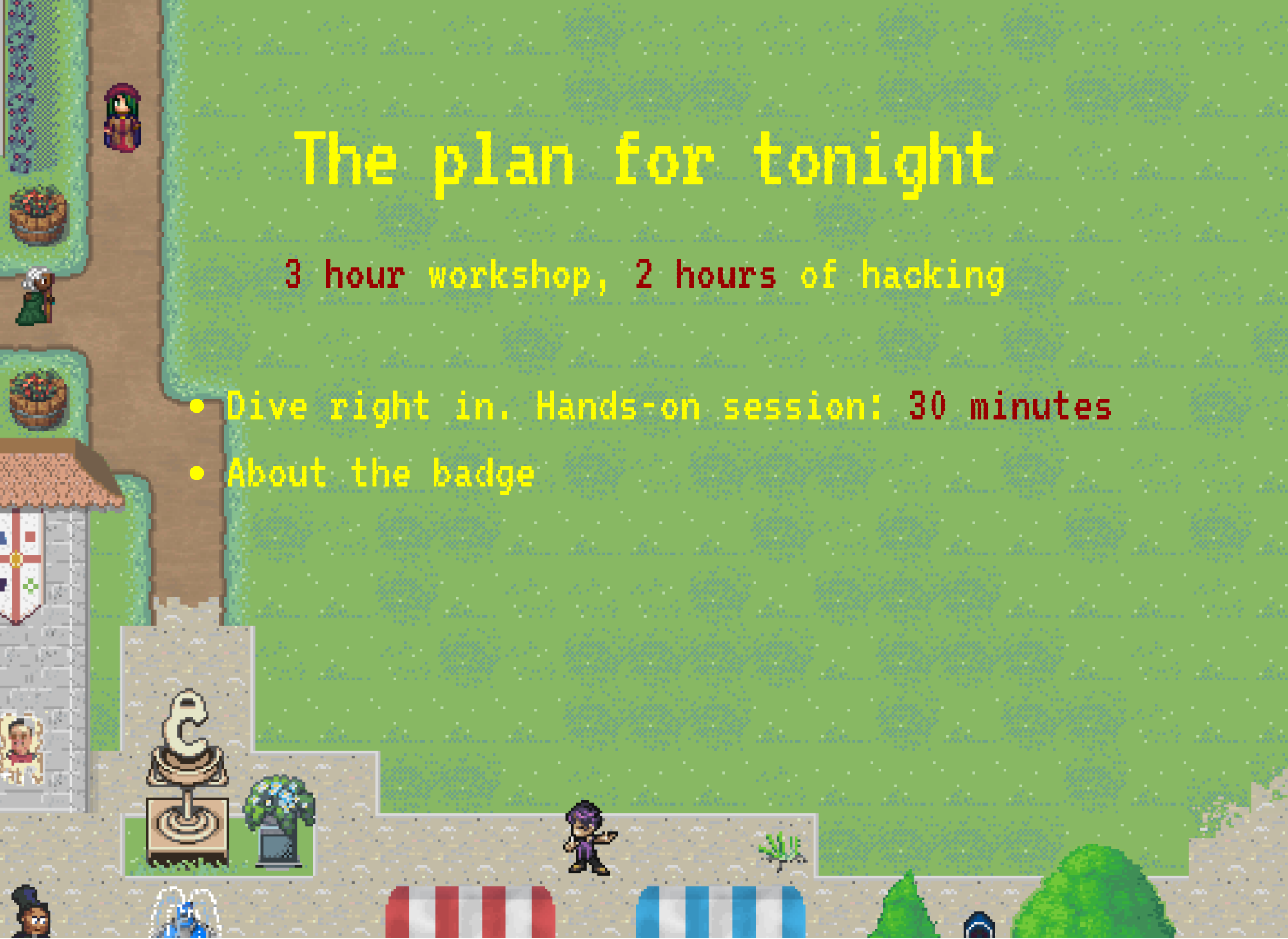
- Dive right in. Hands-on session: 30 minutes



The plan for tonight

3 hour workshop, 2 hours of hacking

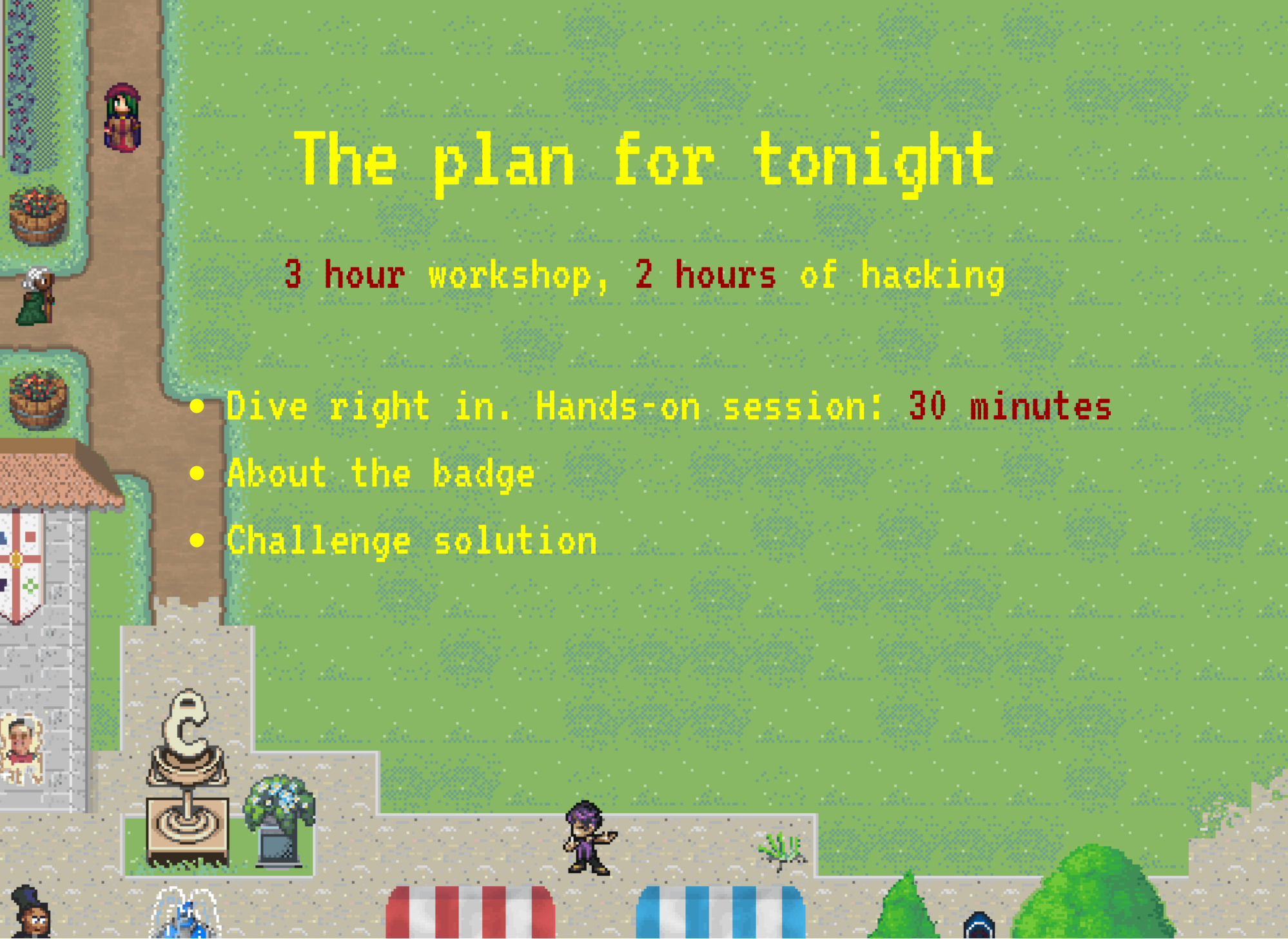
- Dive right in. Hands-on session: 30 minutes
- About the badge



The plan for tonight

3 hour workshop, 2 hours of hacking

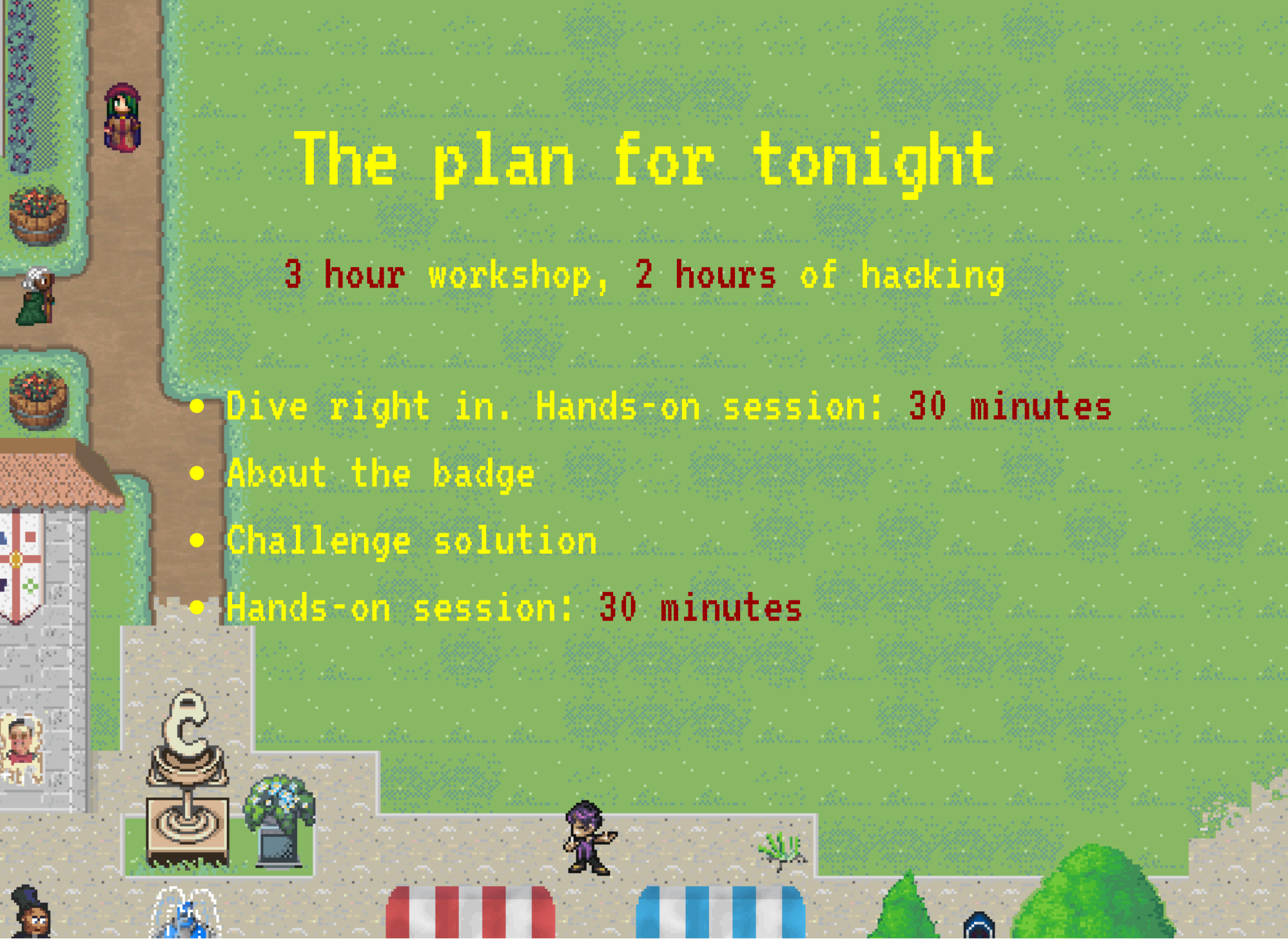
- Dive right in. Hands-on session: 30 minutes
- About the badge
- Challenge solution



The plan for tonight

3 hour workshop, 2 hours of hacking

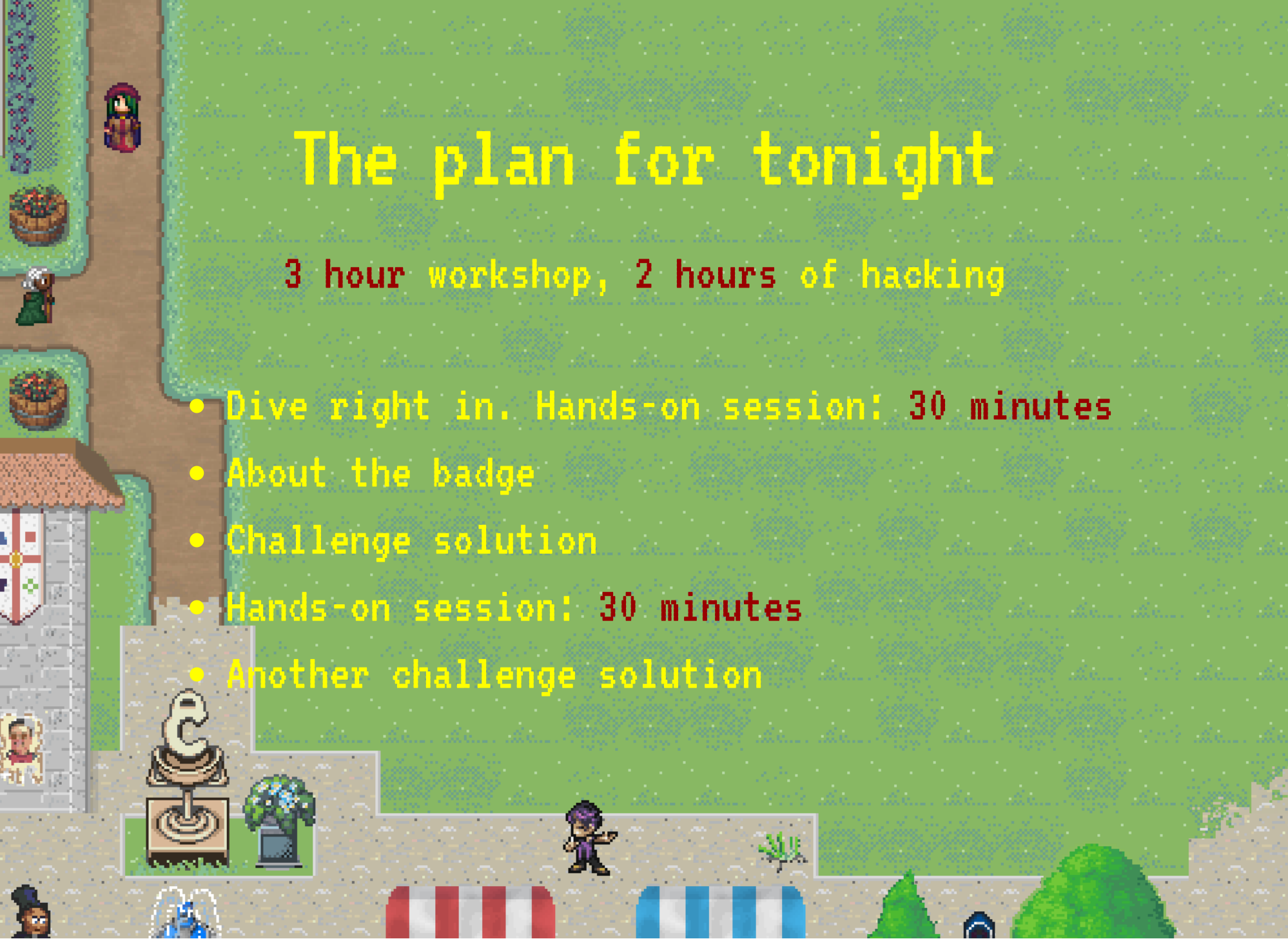
- Dive right in. Hands-on session: 30 minutes
- About the badge
- Challenge solution
- Hands-on session: 30 minutes



The plan for tonight

3 hour workshop, 2 hours of hacking

- Dive right in. Hands-on session: 30 minutes
- About the badge
- Challenge solution
- Hands-on session: 30 minutes
- Another challenge solution



The plan for tonight

3 hour workshop, 2 hours of hacking

- Dive right in. Hands-on session: 30 minutes
- About the badge
- Challenge solution
- Hands-on session: 30 minutes
- Another challenge solution
- [Hands-on session: 15 minutes, followed by solutions]

The plan for tonight

3 hour workshop, 2 hours of hacking

- Dive right in. Hands-on session: 30 minutes
- About the badge
- Challenge solution
- Hands-on session: 30 minutes
- Another challenge solution
- [Hands-on session: 15 minutes, followed by solutions] x4


github.com/nsec/nsec-badge
All flags are in the public source code on Github, but don't go looking for them.
The goal is to learn how to hack an unknown, blackbox device.


Reverse engineering challenges
Two binaries for the reverse-engineering challenge are hosted on the badge and can be downloaded over Wifi.
Download them from Github:
https://github.com/nsec/nsec-badge/tree/master/bin
Because of the guest network setup, connecting to the badge on Wifi may not be possible.

Write-ups

This presentation will include some solutions and images from write-ups made by participants from last year.

You should read them after the workshop because they are very entertaining, go much more in detail and describe the trial and error process of the author.

Write-ups

- https://erichogue.ca/2021/05/NorthSec2021BadgeFirstFlags/
- https://hideandsec.sh/books/ctf/page/northsec-2021-badge-writeup
- https://blog.quantumlyconfused.com/ctf/2021/05/30/nsec2021-badgelife-pt1/
- https://boschko.ca/northsec-2021-badge-writeup/
- https://nsec.io/competition-write-ups/

^ All NorthSec challenge write-ups


Hacking session!


About the badge

About the badge

- A traditional element of the in-person NorthSec conference

About the badge

- A traditional element of the in-person NorthSec conference
- The "game" is not a real game

About the badge

- A traditional element of the in-person NorthSec conference
- The "game" is not a real game
- Designed without any particular goal in mind, the end result is "organic"

About the badge

- A traditional element of the in-person NorthSec conference
- The "game" is not a real game
- Designed without any particular goal in mind, the end result is "organic"
- Beginner-level CTF

About the badge

- A traditional element of the in-person NorthSec conference
- The "game" is not a real game
- Designed without any particular goal in mind, the end result is "organic"
- Beginner-level CTF
- The purpose is to teach you to pay attention to small details

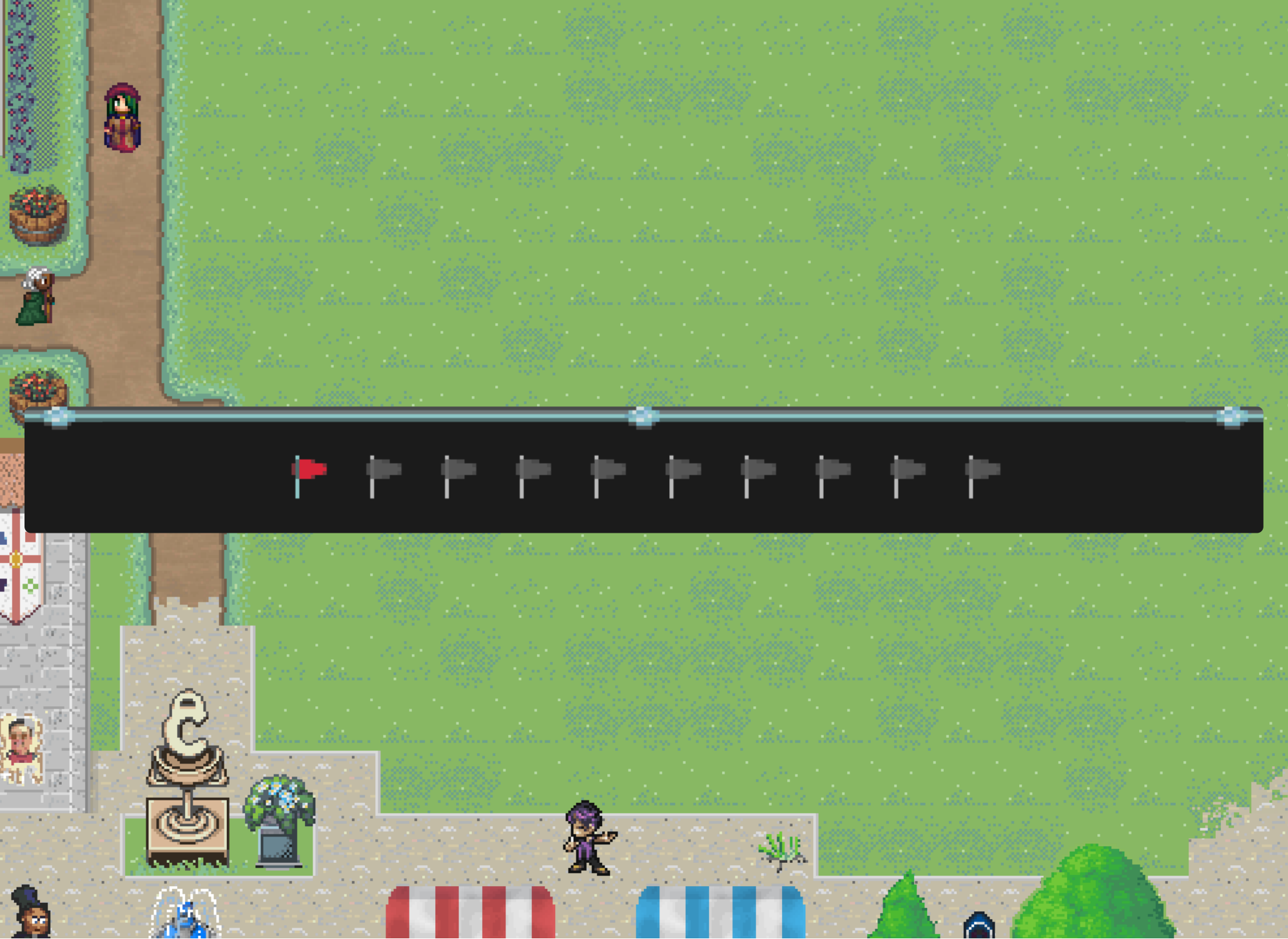
The making

Northsec 2021 Badge Overview

https://www.youtube.com/watch?v=zkiFeA-PHm0

Flagbot


FLAGBOT BOT 05/05/2021
@ is making good progress on badge challenges and is now recognized as a Badge Apprentice!
FLAGBOT BOT 05/05/2021
@ has completed all badge challenges and is now recognized as a Badge Wizard!




Have you ever heard of
'Konami code'?


Have you ever heard of 'Konami code'?
The Konami Code is a cheat code that appears in many Konami video games. The player has to press a sequence of buttons on the game controller to enable a cheat or other effects.
https://en.wikipedia.org/wiki/Konami_code


Have you ever heard of
'Konami code'?
B A
* some variations of this sequence exist


Are u listening?

Are u listening?

The badge has Wifi connectivity,
so it can be classified as an IoT device.

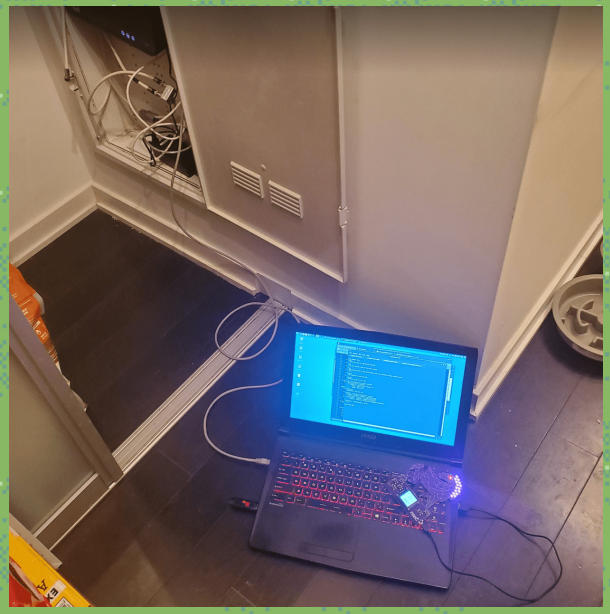

Are u listening?
Does it try to talk to anything?
Does it "call home"?


Are u listening?
** Difficult to do the correct setup
during the workshop.


Are u listening?
For most participants during the CTF,
this was the most difficult flag,
so we'll discuss it first.

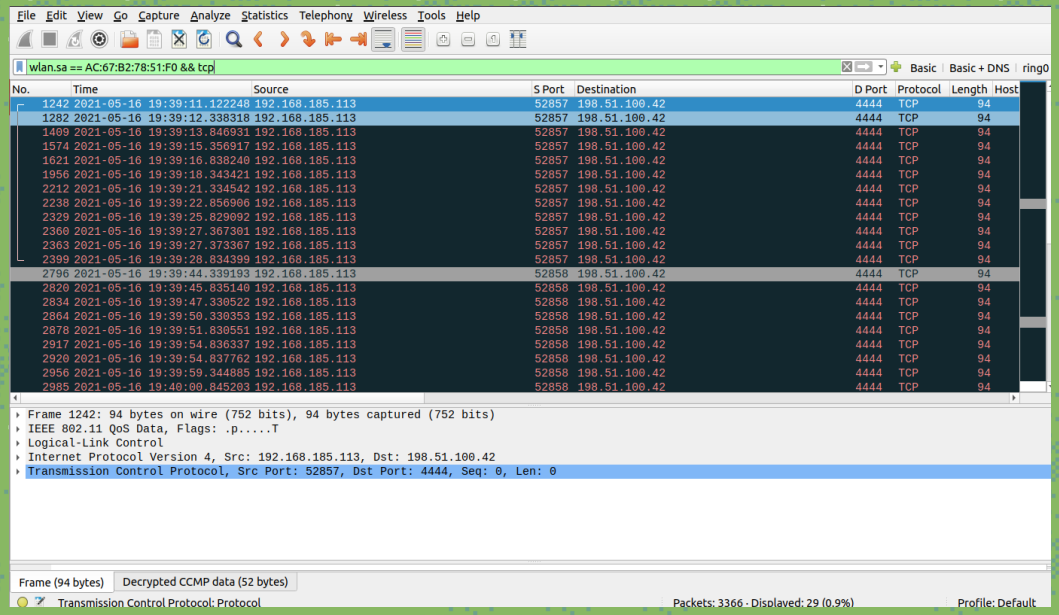

Are u listening?
A hint left on the Discord channel:
If the badge tried to do curl https://nsec.io/flag.txt for example, do you think you would be able to detect this with your current setup?

Are u listening?



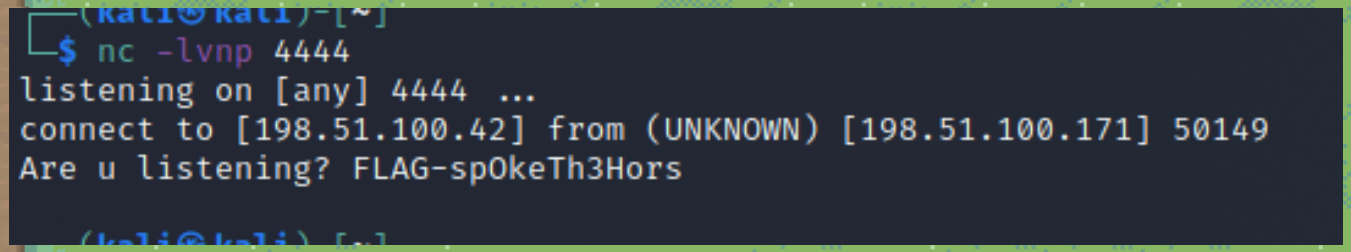
https://hideandsec.sh/books/ctf/page/northsec-2021-badge-writeup

Are u listening?

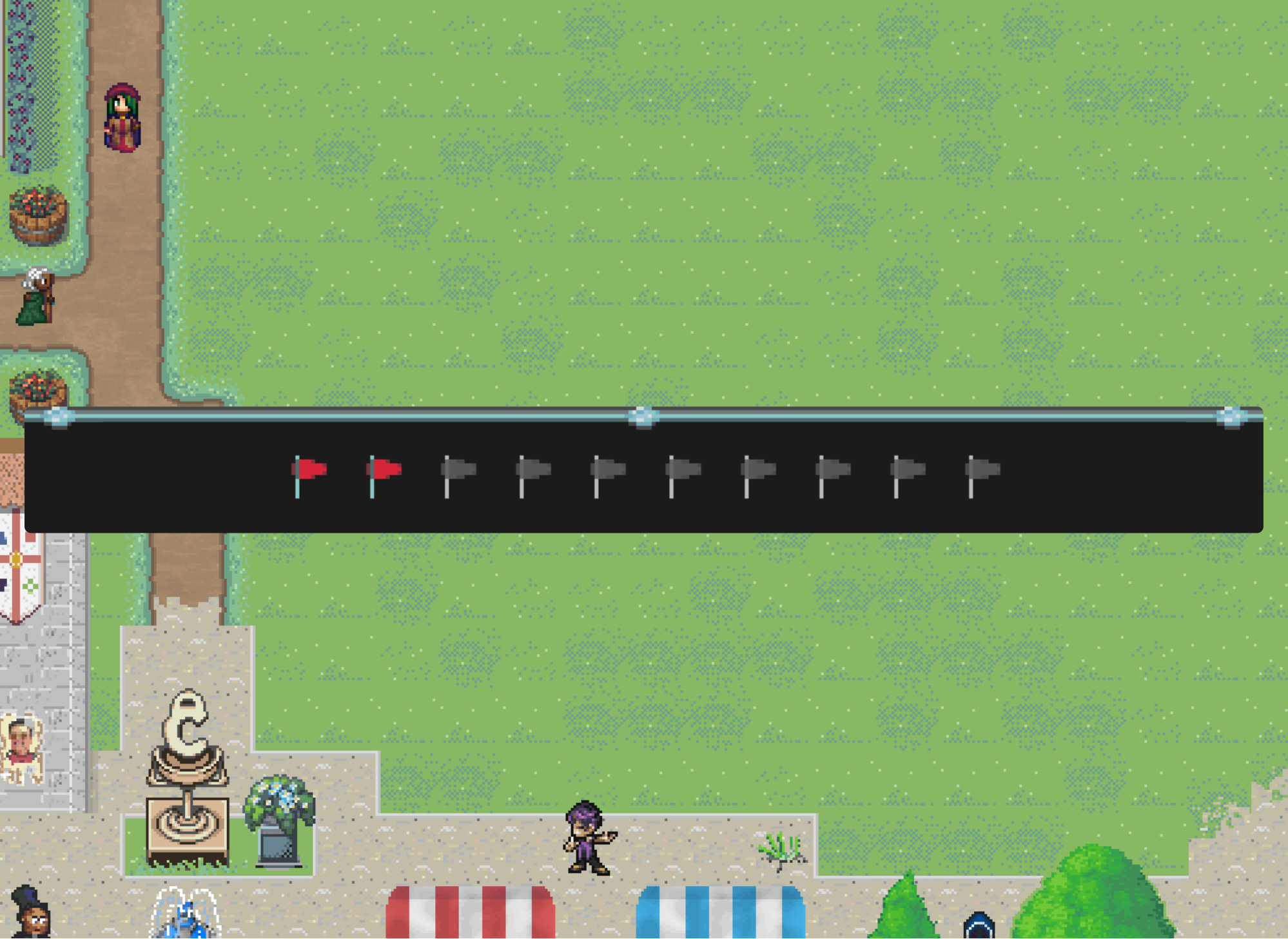


https://erichogue.ca/2021/05/NorthSec2021BadgeNetworking/

Are u listening?



https://blog.quantumlyconfused.com/ctf/2021/05/30/nsec2021-badgelife-pt1/




Hacking session!


Where there are NO flags?


Where there are NO flags?
• On the badge itself (silkscreen, etc.)


Where there are NO flags?
• On the badge itself (silkscreen, etc.)
• In the sound effects


Where there are NO flags?
• On the badge itself (silkscreen, etc.)
• In the sound effects
• In the LED blinking patterns


Where there are NO flags?
• On the badge itself (silkscreen, etc.)
• In the sound effects
• In the LED blinking patterns
• In the weird 6-pin connectors on the body and the head of the horse


Where there are NO flags?
• On the badge itself (silkscreen, etc.)
• In the sound effects
• In the LED blinking patterns
• In the weird 6-pin connectors on the body and the head of the horse
• On the badge webpage


Where there are NO flags?
• On the badge itself (silkscreen, etc.)
• In the sound effects
• In the LED blinking patterns
• In the weird 6-pin connectors on the body and the head of the horse
• On the badge webpage
• In the firmware of the CH340C chip (U3)


Le Lac du Quack

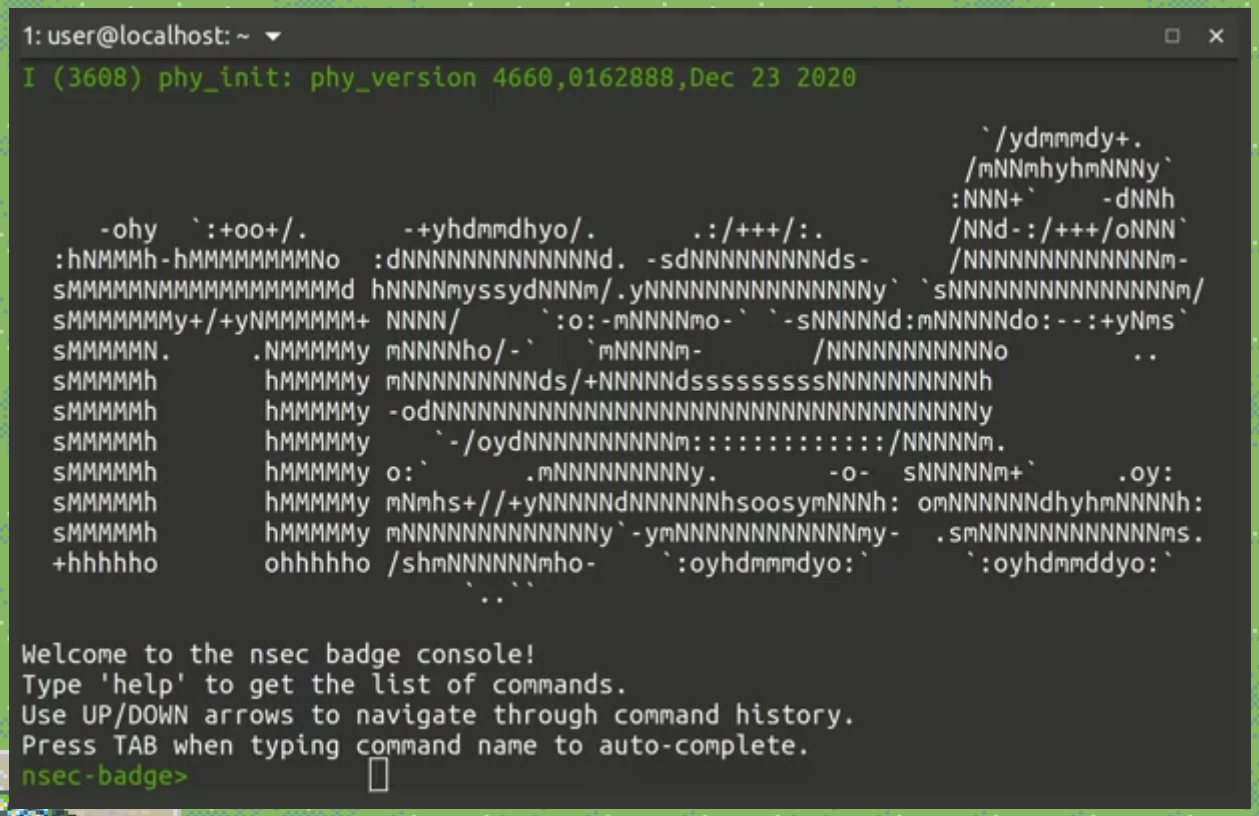

Le Lac du Quack


Le Lac du Quack
This badge has a CLI feature on the serial port. Be sure to connect to it for full experience!
Always look for serial console on the device!
It may output very useful information,
such as debug log or error messages.

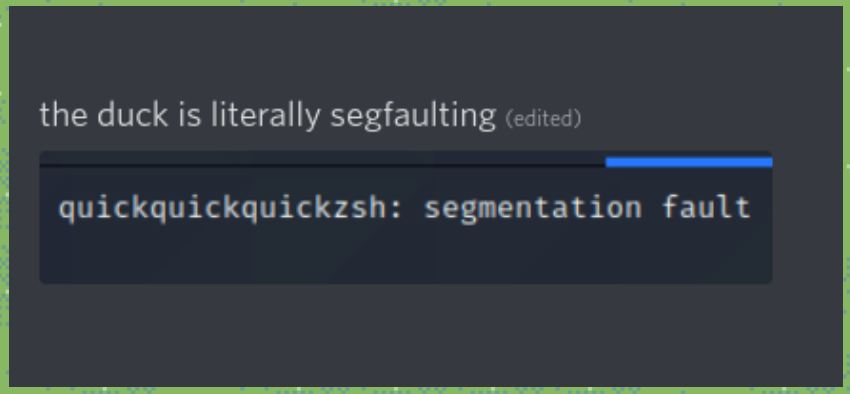
Le Lac du Quack

```
$ picocom -b 115200 /dev/ttyUSB0
picocom v3.1

port is         : /dev/ttyUSB0
flowcontrol     : none
baudrate is     : 115200
parity is       : none
databits are    : 8
stopbits are    : 1
escape is       : C-a
local echo is   : no
noinit is       : no
noreset is      : no
hangup is       : no
nolock is       : no
```

Le Lac du Quack



Le Lac du Quack




Le Lac du Quack
ANSI escape sequences are a standard for in-band signaling to control cursor location, color, font styling, and other options on video text terminals and terminal emulators.
https://en.wikipedia.org/wiki/ANSI_escape_code


Le Lac du Quack
\x1b[?1049h
\x1b[H\x1b[2J\x1b[3J
\x1b[?1049l

Le Lac du Quack

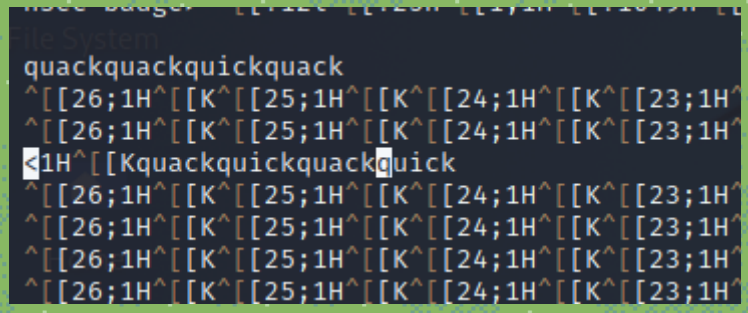
```
quackquackquickquackquickquickquackquickquackquackquickquackquickquackquickqu
ackquickquickquackquackquickquackquackquickquickquickquickquackquackquickquac
kquickquackquickquackquackquackquackquackquackquickquickquackquackquickq
uickquickquickquackquickquickquickquackquackquickquickquickquickquackquickqui
ckquickquickquackquickquickquickquickquickquickquickquickquickquickquack
quickquickquackquickquickquickquackquackquackquickquackquickquickquickquackqu
ickquickquickquickquackquackquackquickquickquackquickquackquickquackquickquac
kquackquackquackquickquickquickquickquackquickquackquackquickquickquick

------

1101001011010101001101100001101010111111100110000100011000010000100000000001
0010001110100010000111001010101111000101100 0

0010110100101010110010011110010101000000011001111011100111101111011111111110
1101110001011101111000110101010000111010011 1

..-.--.-..-.-.-.--..-..----..-.-.-.......--..-----.---..-----.-----.----------.
--.---...-.---.-----...--.-.-.-....---.-..--- 

--.-..-.--.-.-.-..--.--....--.-.-.--------..--....-...--....-....-...........-
..-...---.-...-....---..-.-.-.----...-.--... 

aababbabaababababbaabaabbbbaababababaaaaaaabbaabbbbabbbaabbbbabbbbabbbbbbbbbbba
bbabbbaaabababbbabbbbaaabbabababaaaabbbabaabbb 

bbabaabababbababaabbabbaaaabbababababbbbbbbaabbaaaabaaabbaaaabaaaabaaaaaaaaaaab
aabaaabbbabaaabaaaabbbaabababababbbbaaababbaaa 

211211211111122124211111722413241411112133113143211111143112 3
```

https://blog.quantumlyconfused.com/ctf/2021/05/30/nsec2021-badgelife-pt1/

Le Lac du Quack

https://en.wikipedia.org/wiki/Morse_code

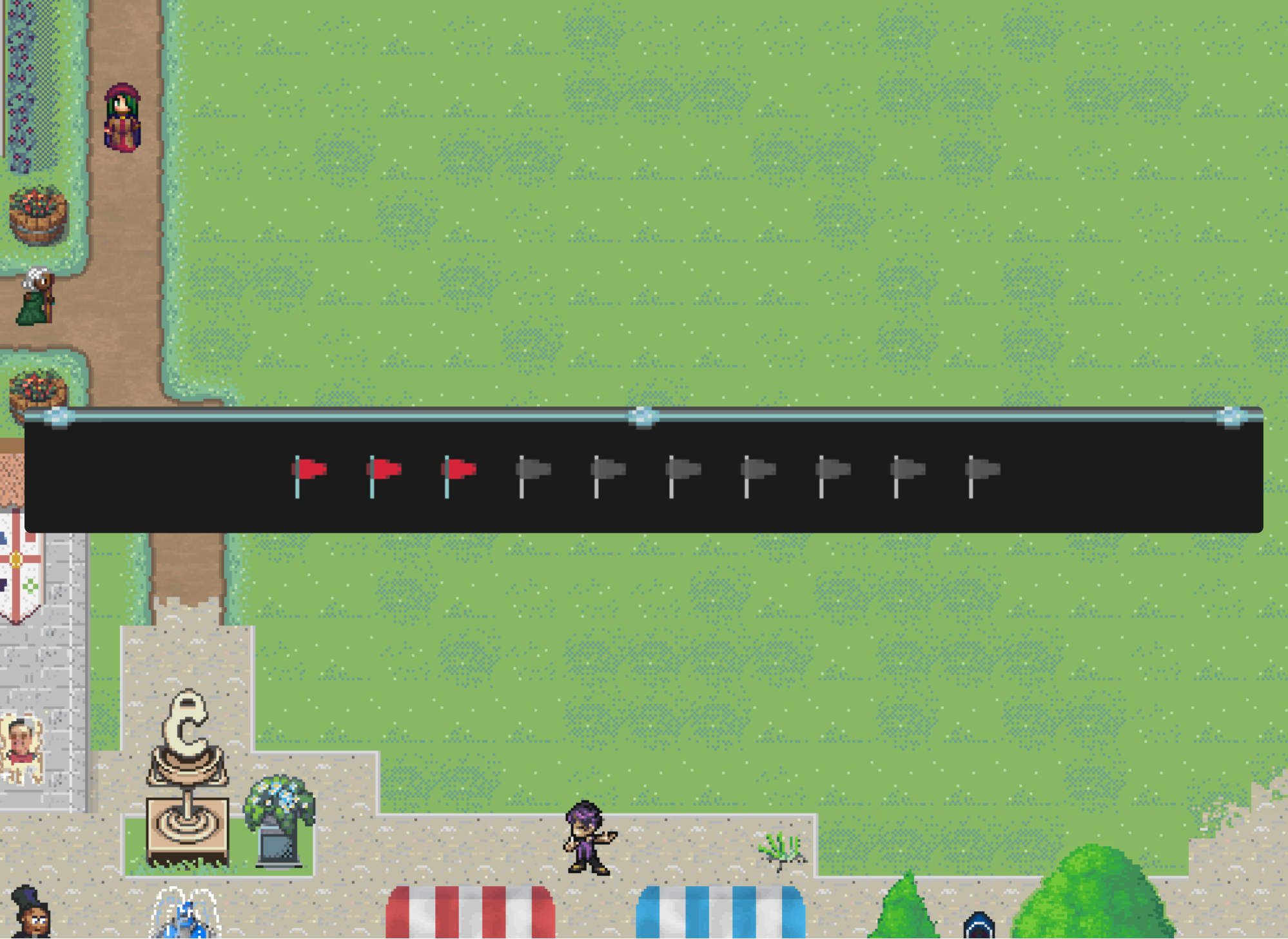
Le Lac du Quack



https://hideandsec.sh/books/ctf/page/northsec-2021-badge-writeup

Le Lac du Quack

```
static constexpr char quack[] =
    "quackquackquickquack quickquickquack quickquack quackquickquackquick
    "quackquickquack quickquick quackquack quickquack quackquickquick "
    "quickquickquack quackquickquackquick quackquickquack quack "
    "quackquackquack quackquackquick quick quack quackquickquickquick "
    "quickquackquick quick quickquack quackquickquick quickquickquack "
    "quickquickquick quick quack quickquickquickquick quickquick "
    "quickquickquick quickquickquackquick quickquackquickquick quickquack
    "quackquackquick quackquickquickquick quackquick quickquickquickquack
    "quackquackquickquick quackquickquack quickquack quickquackquack "
    "quackquack quickquickquick quackquickquackquack quickquickquick";
```




Hacking session!


Up-down-left-right


Up-down-left-right


Up-down-left-right

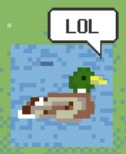

The off-limits island


The off-limits island

The off-limits island

Can't swim. Can't fly.

No choice but to walk.



The off-limits island

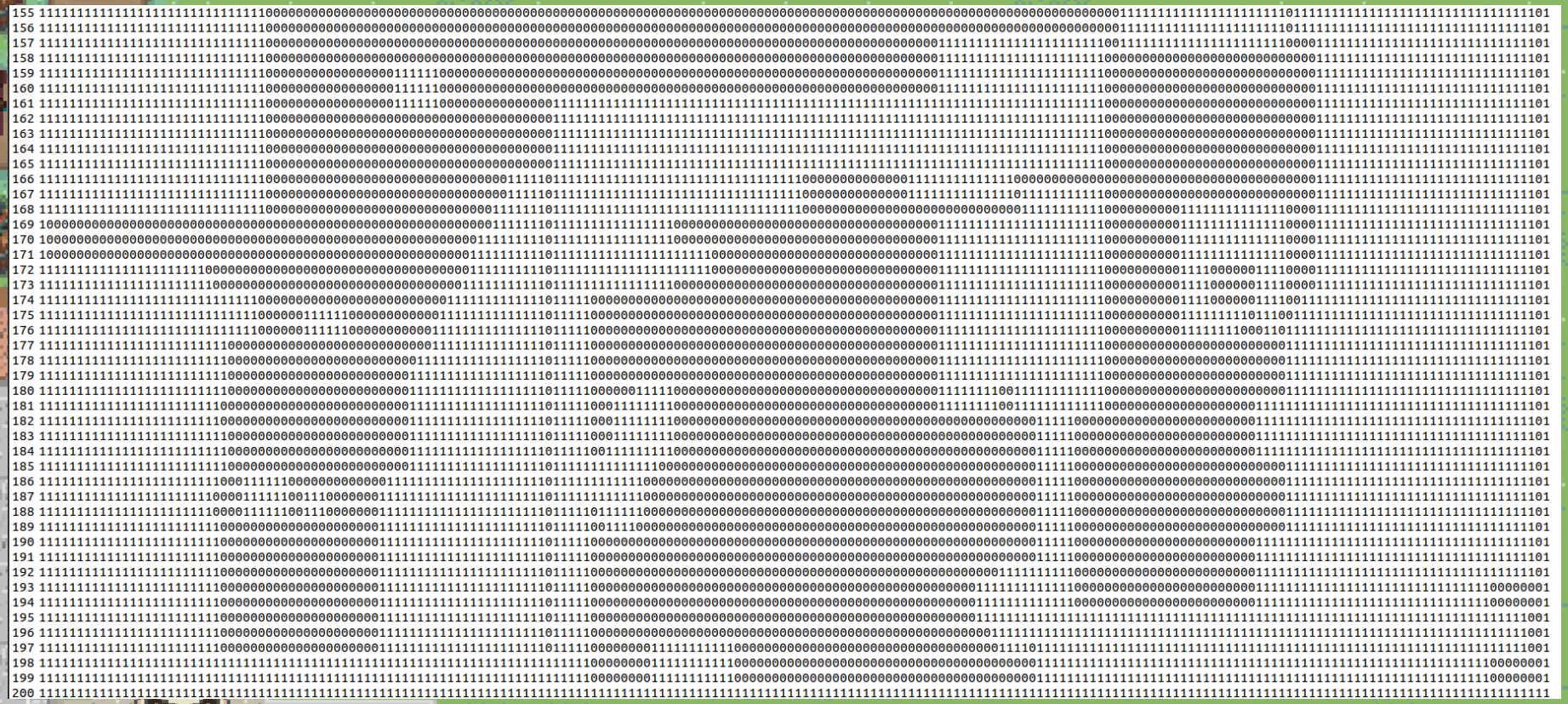
The off-limits island

```
user@linux:esp32/spiffs/rpg$ tree
.
├── main.blocked
└── main.scene

0 directories, 2 files
```

The off-limits island

```
user@linux:esp32/spiffs/rpg$ xxd main.blocked | head
00000000: ffff ffff ffff ffff ffff ffff ffff ffff  ................
00000010: ffff ffff ffff ffff ffff ffff ffff ffff  ................
00000020: ffff ffff ffff ffff ffff ffff ffff ffff  ................
00000030: ffff ffff ffff ffff ffff ffff ffff ffff  ................
00000040: ffff ffff ffff ffff ffff ffff ffff ffff  ................
00000050: ffff ffff ffff ffff ffff ffff ffff ffff  ................
00000060: ffff ffff 81ff ffff 01f8 ffff ffff 1f00  ................
00000070: f8ff ff1f 0000 0000 ffff ffff ff81 ffff  ................
00000080: ff01 f8ff ffff ff1f 00f8 ffff 1f00 0000  ................
00000090: 00ff ffff ffff 9fff ffff 01f8 ffff ffff  ................
```

The off-limits island

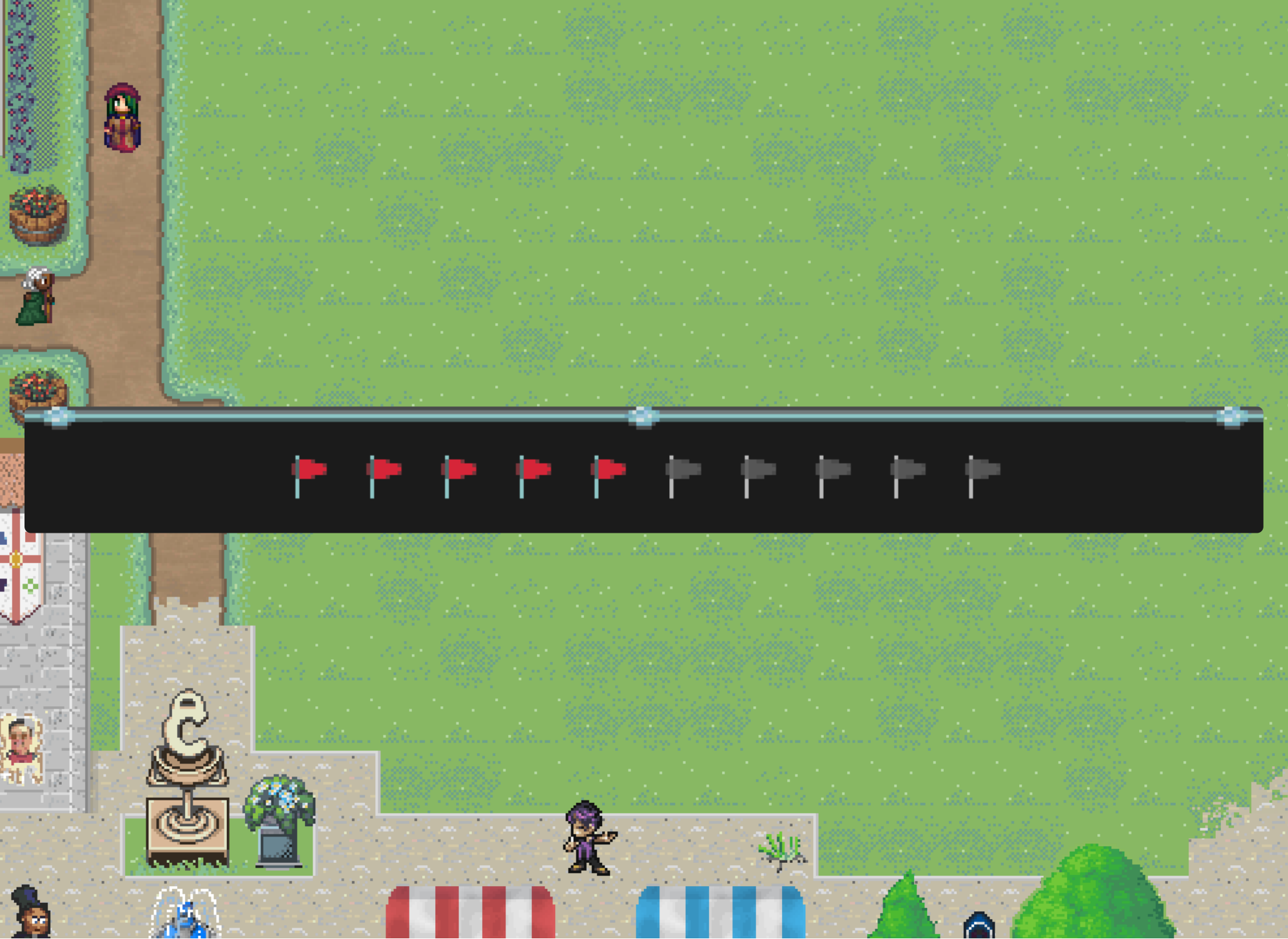


https://erichogue.ca/2021/05/NorthSec2021BadgeTheMap/


How do I get past the gate?


How do I get past the gate?


How do I get past the gate?

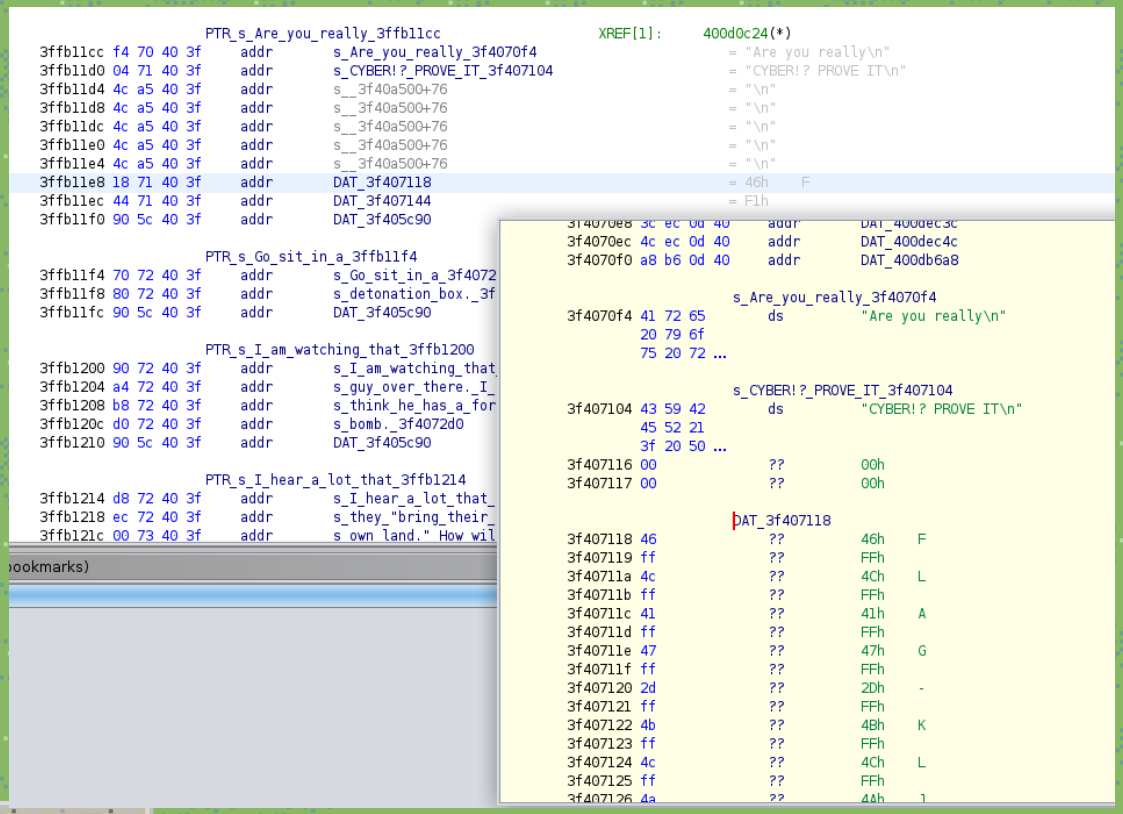



Hacking session!

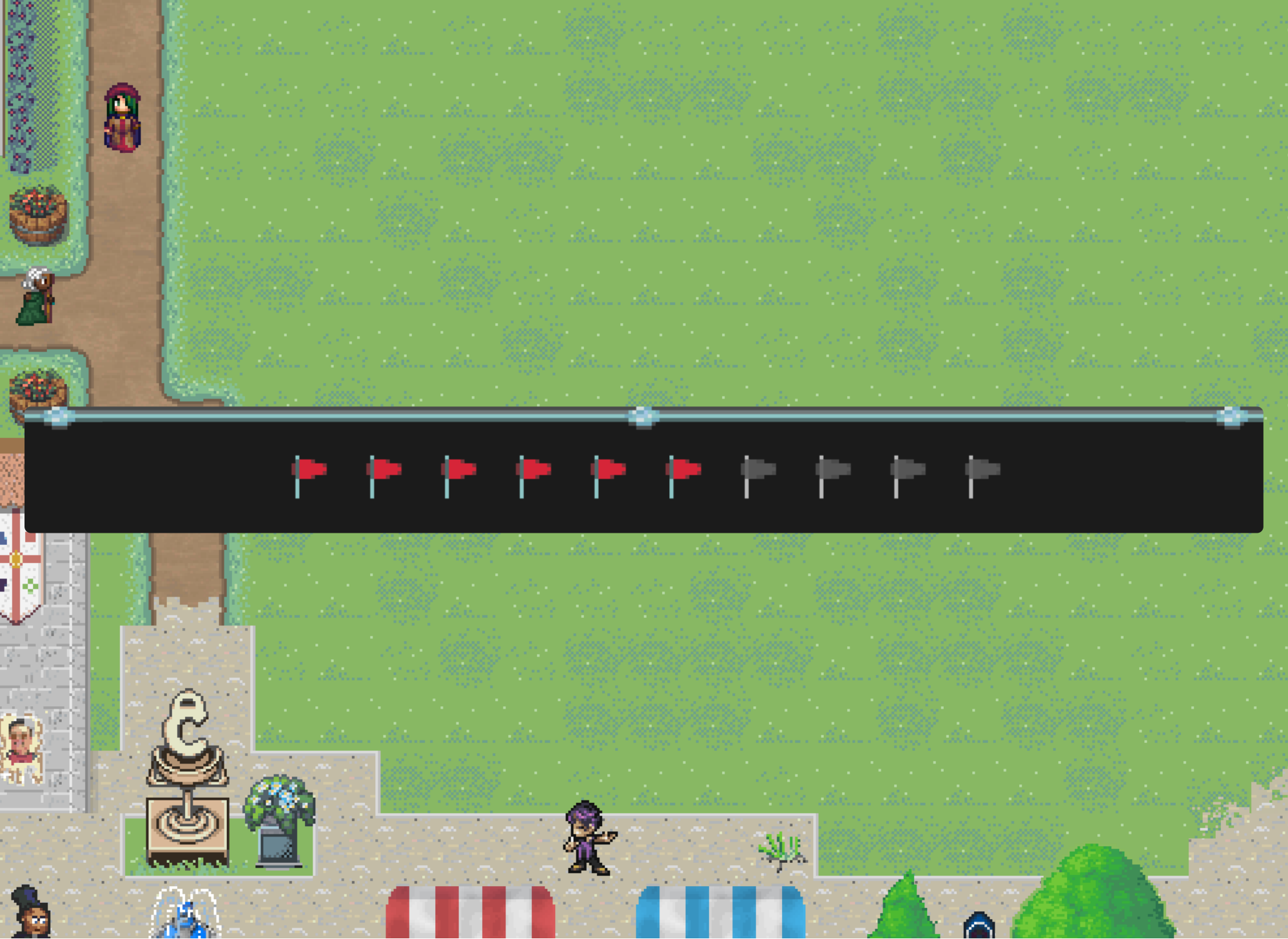

Are you really CYBER!?


Are you really CYBER!?

Are you really CYBER!?

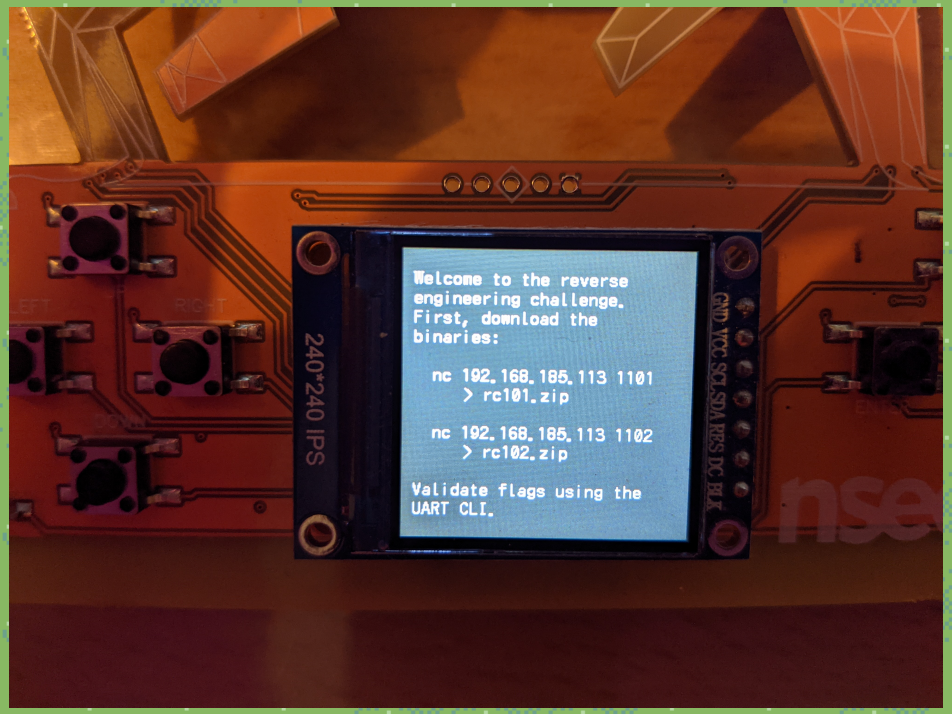


https://blog.quantumlyconfused.com/ctf/2021/05/30/nsec2021-badgelife-pt1/




Hacking session!

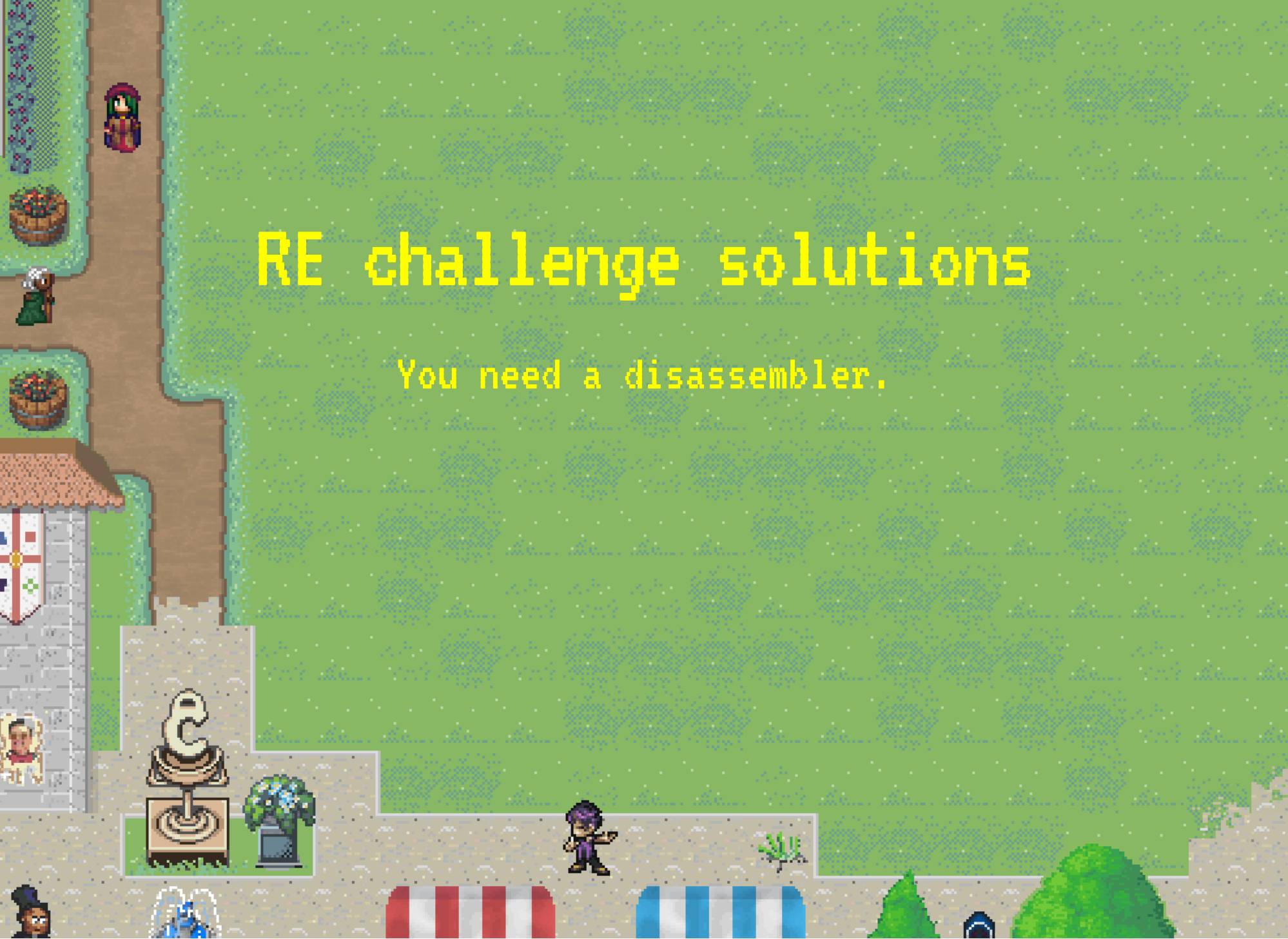
RE challenge solutions

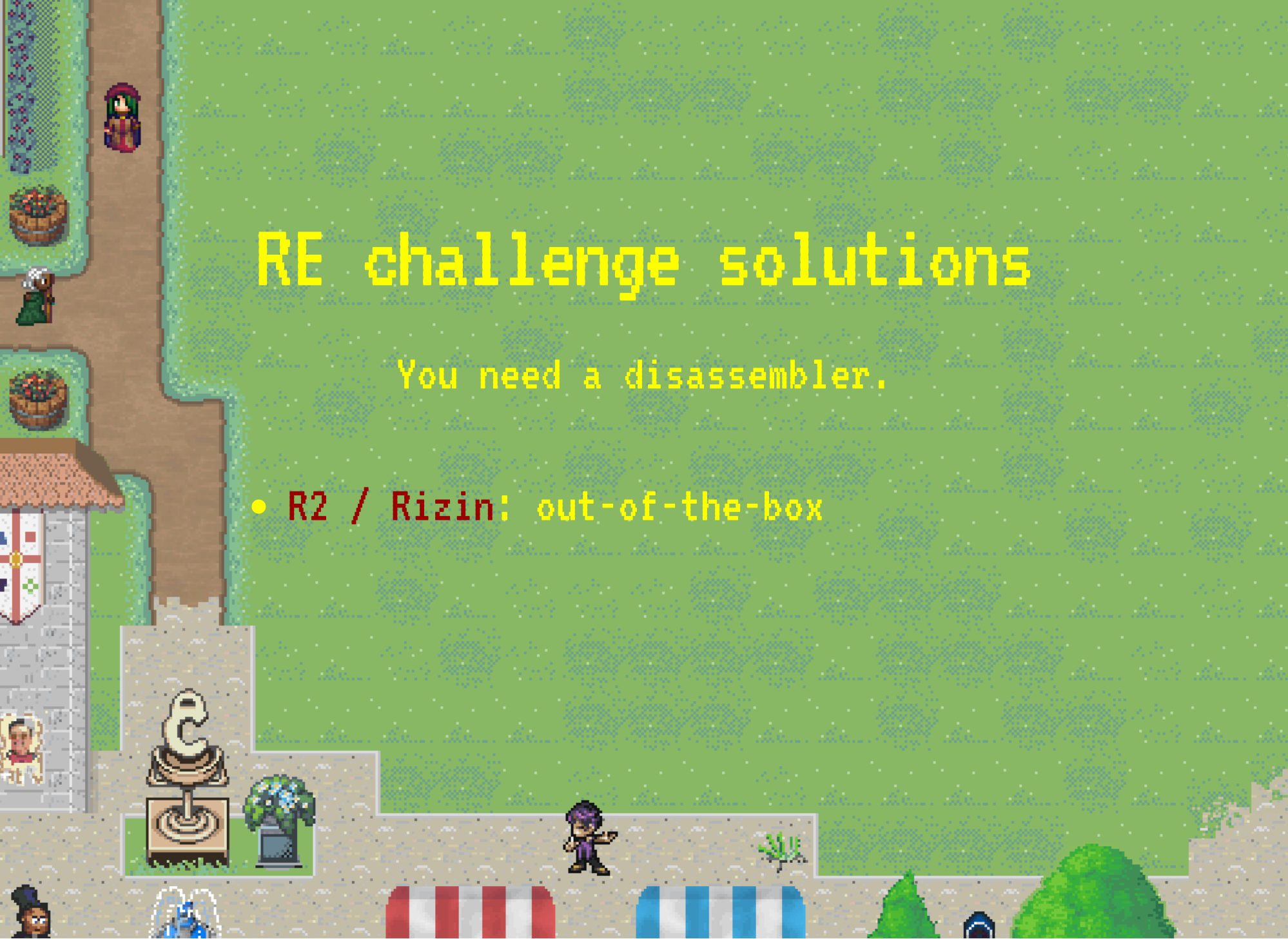


https://erichogue.ca/2021/05/NorthSec2021BadgeReverseEngineeringFlags/

RE challenge solutions

```
$ file re*.elf
re101.elf: ELF 32-bit LSB executable, Tensilica Xtensa,
version 1 (SYSV), statically linked, with debug_info, not stripped
re102.elf: ELF 32-bit LSB executable, Tensilica Xtensa,
version 1 (SYSV), statically linked, with debug_info, not stripped
```


RE challenge solutions
You need a disassembler.

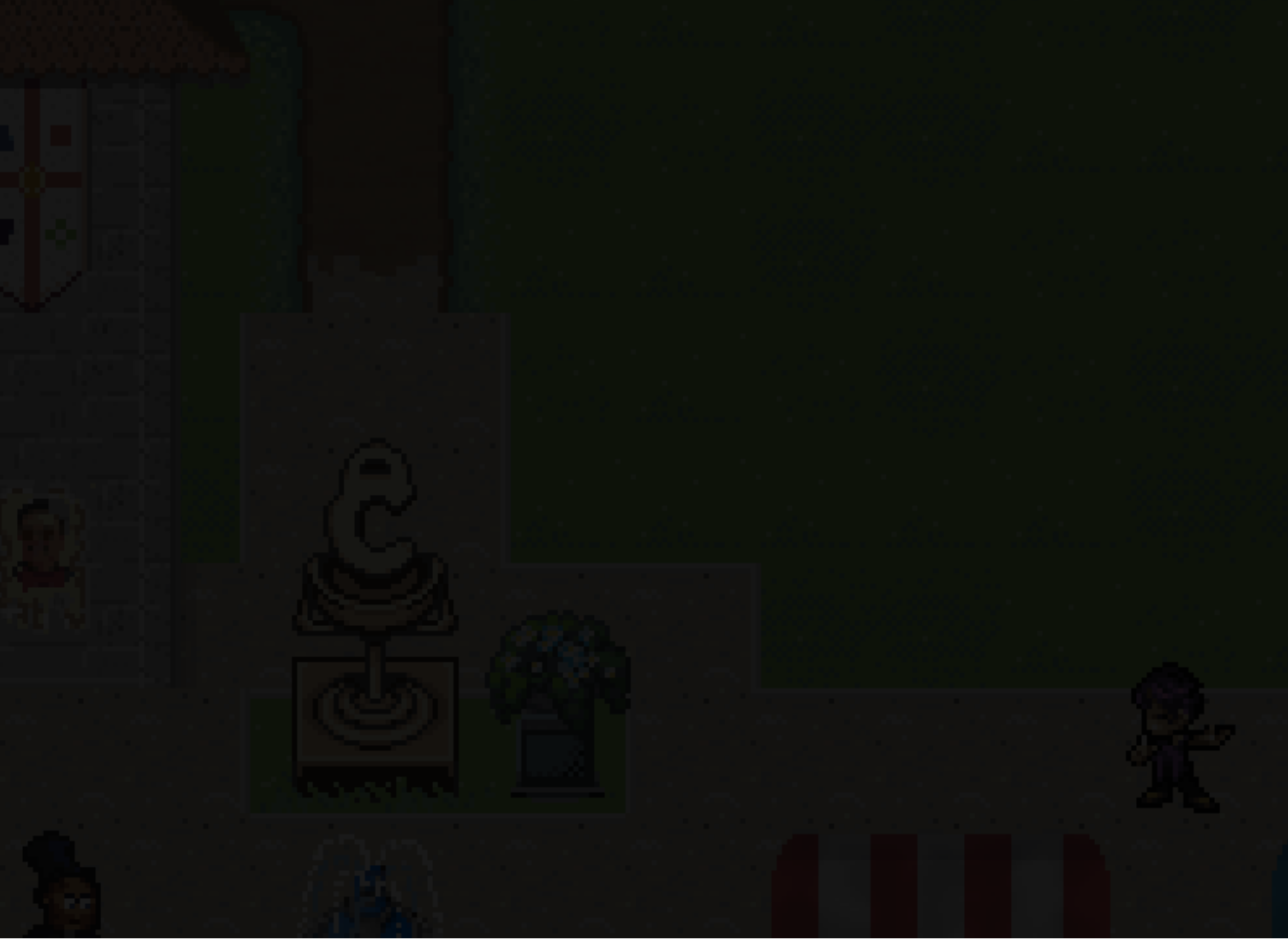

RE challenge solutions
You need a disassembler.
• R2 / Rizin: out-of-the-box

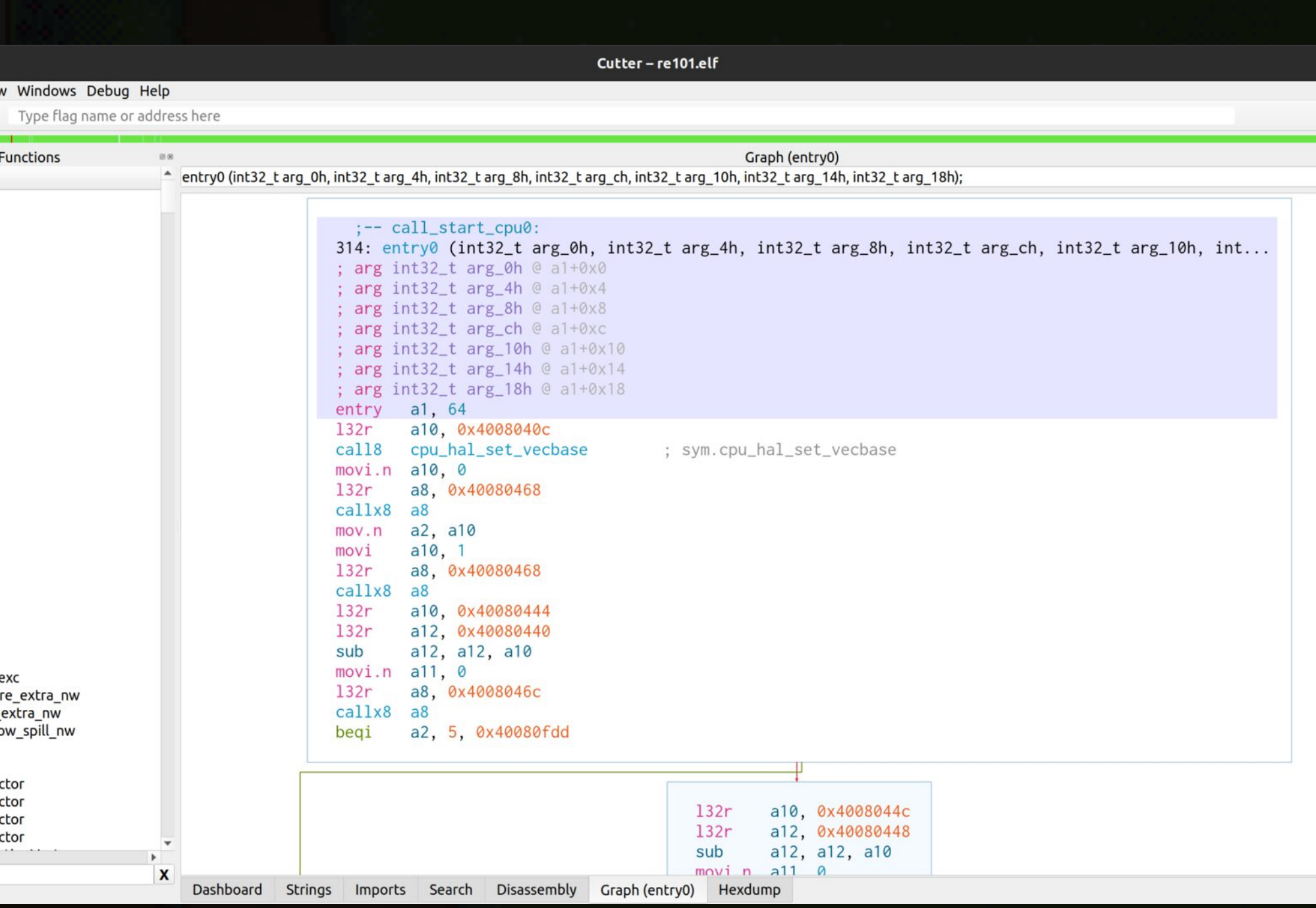

RE challenge solutions
You need a disassembler.
• R2 / Rizin: out-of-the-box
• Ghidra: with a plugin

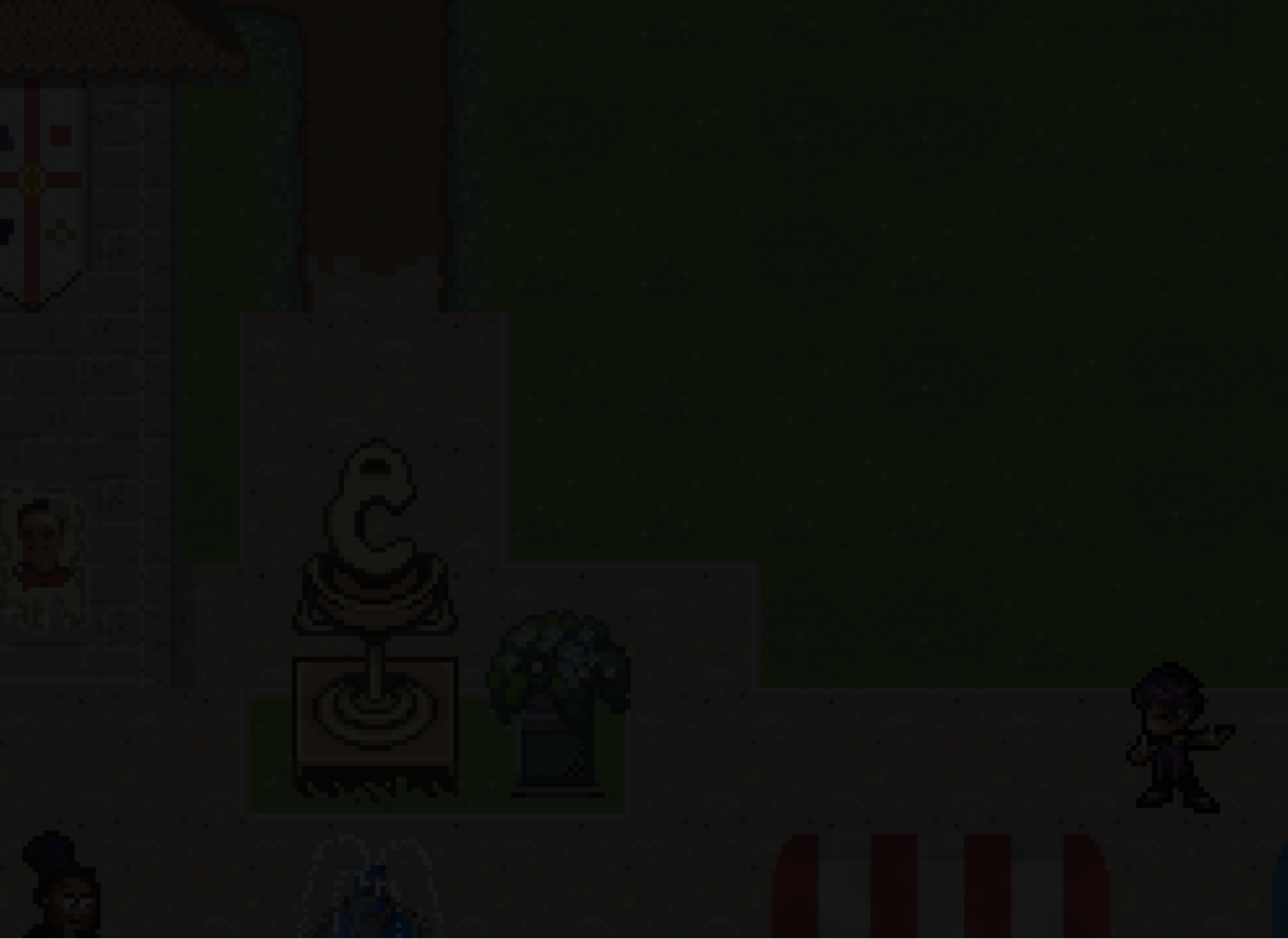

RE challenge solutions
You need a disassembler.
• R2 / Rizin: out-of-the-box
• Ghidra: with a plugin
https://github.com/yath/ghidra-xtensa

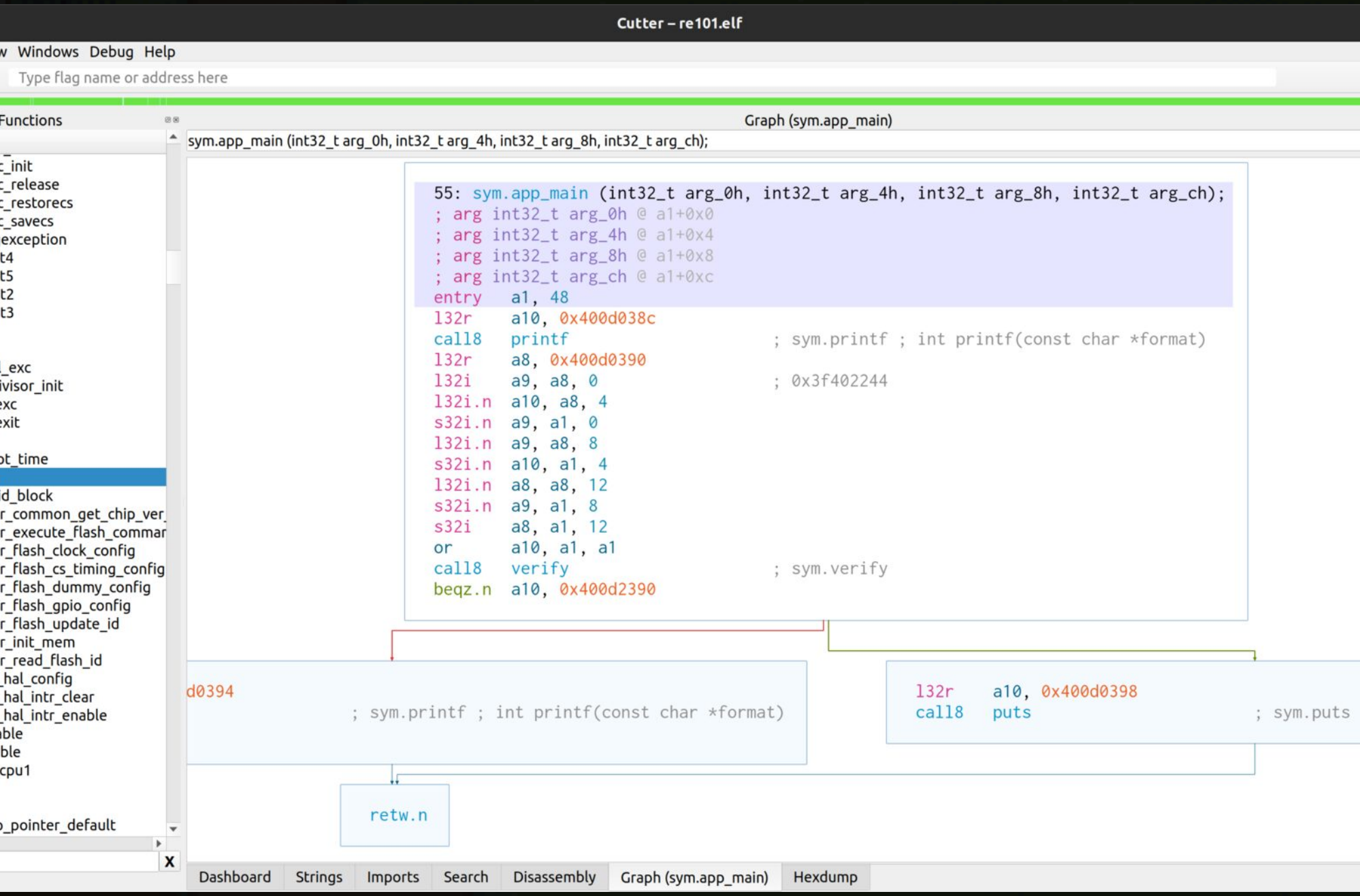

RE challenge solutions
You need a disassembler.
• R2 / Rizin: out-of-the-box
• Ghidra: with a plugin
https://github.com/yath/ghidra-xtensa
• IDA: ???

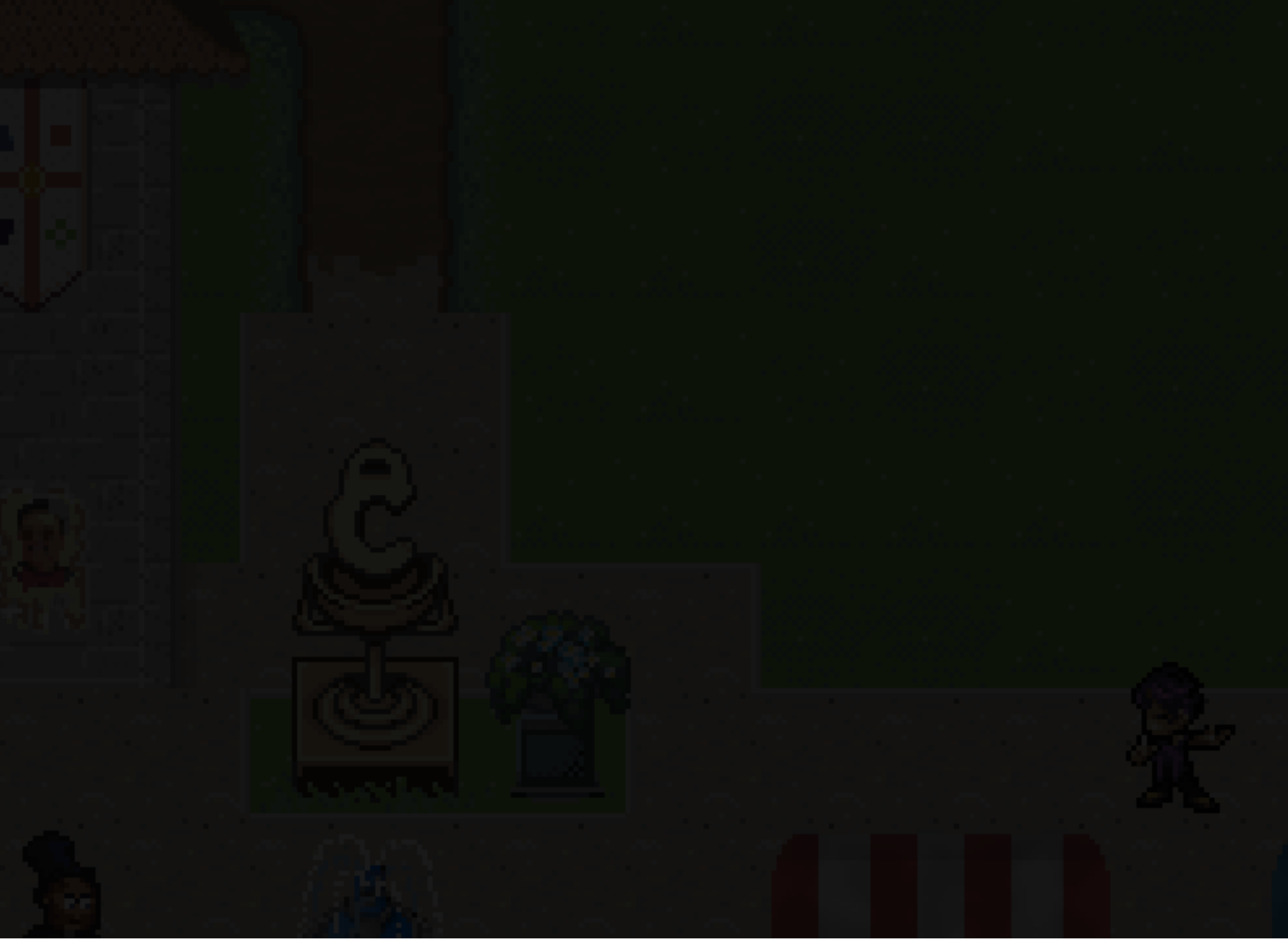

rel01.elf

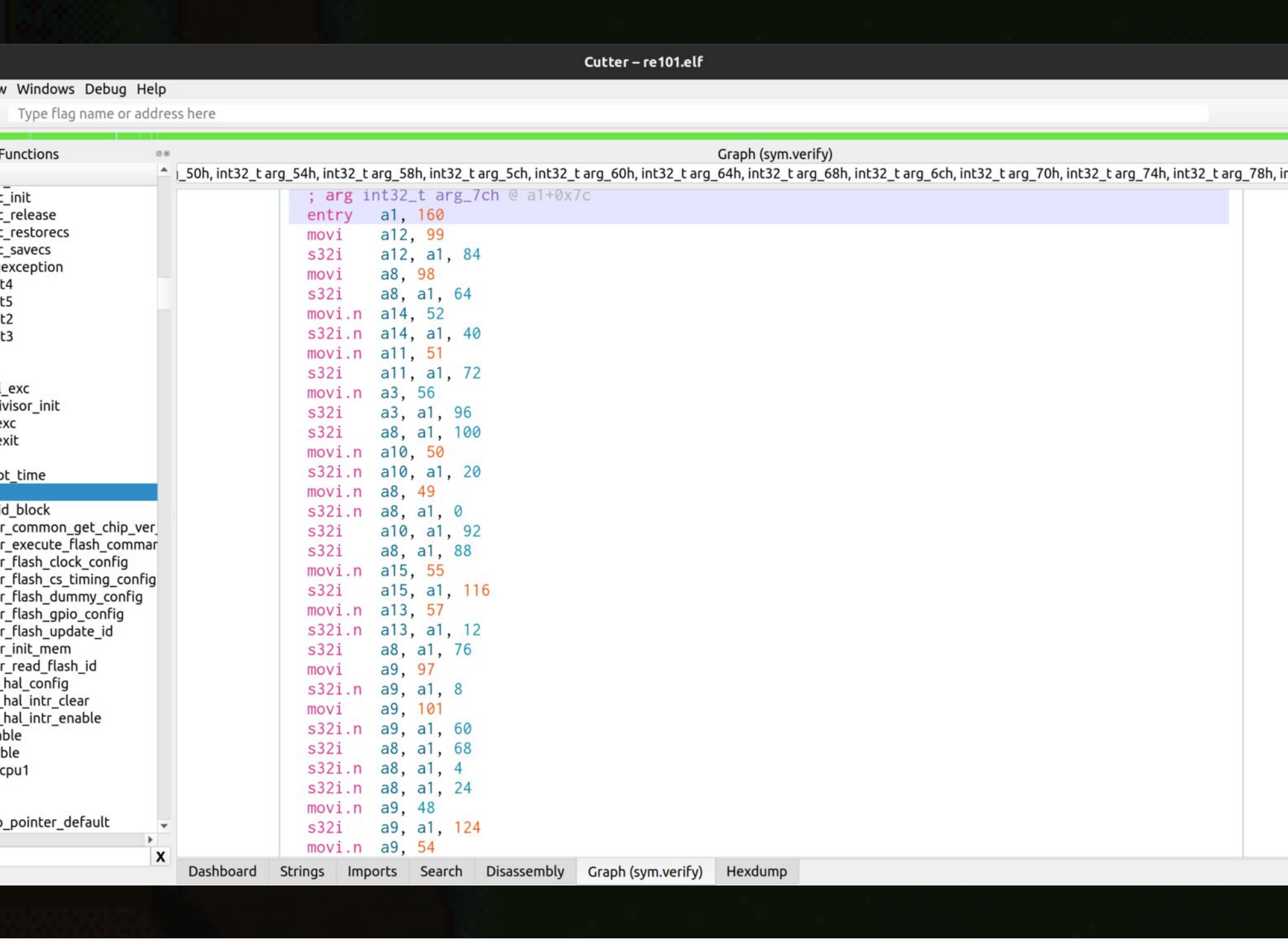


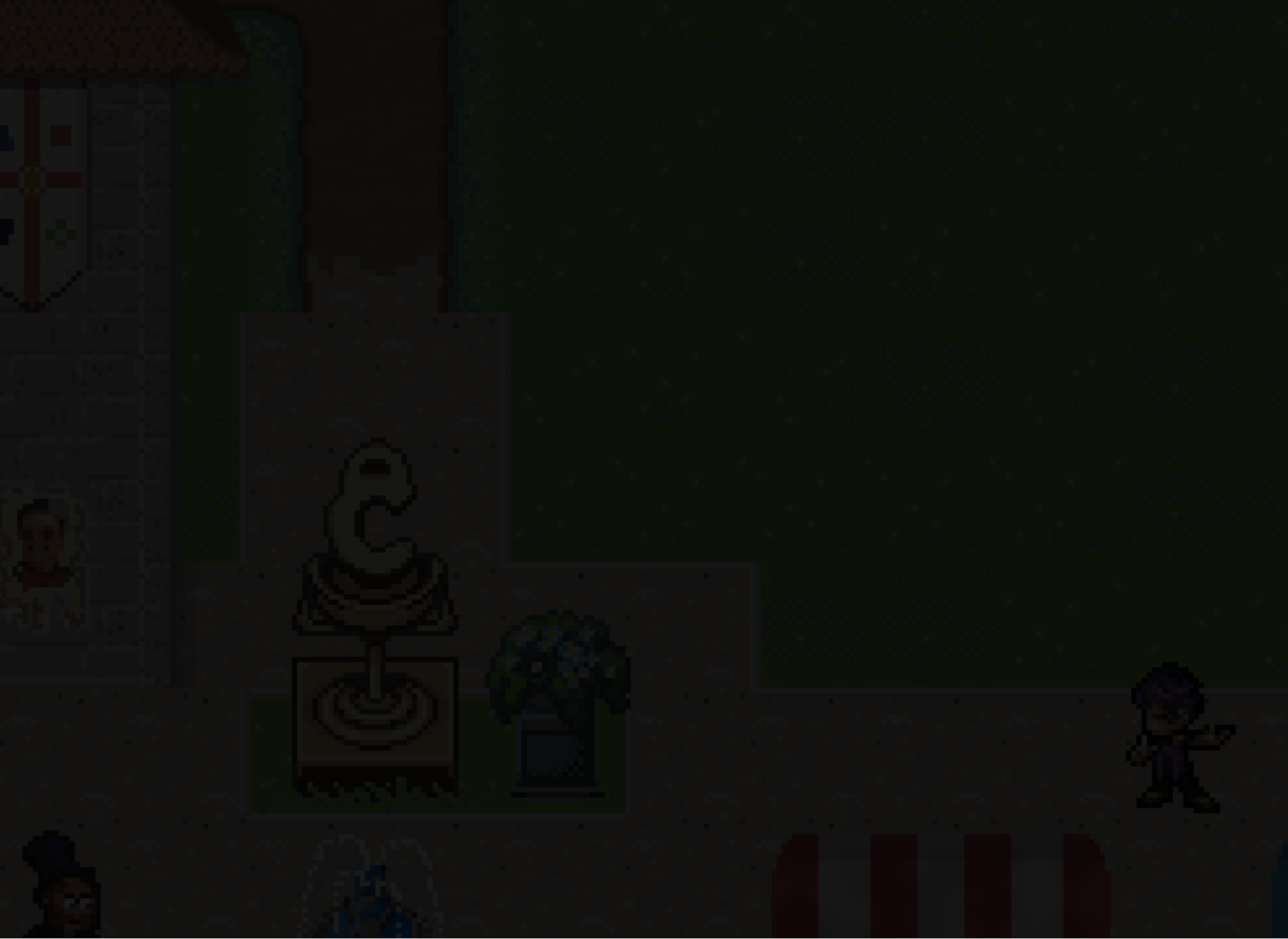

Cutter – re101.elf
w Windows Debug Help
Type flag name or address here
Functions
Graph (entry0)
entry0 (int32_t arg_0h, int32_t arg_4h, int32_t arg_8h, int32_t arg_ch, int32_t arg_10h, int32_t arg_14h, int32_t arg_18h);
;-- call_start_cpu0:
314: entry0 (int32_t arg_0h, int32_t arg_4h, int32_t arg_8h, int32_t arg_ch, int32_t arg_10h, int...
; arg int32_t arg_0h @ a1+0x0
; arg int32_t arg_4h @ a1+0x4
; arg int32_t arg_8h @ a1+0x8
; arg int32_t arg_ch @ a1+0xc
; arg int32_t arg_10h @ a1+0x10
; arg int32_t arg_14h @ a1+0x14
; arg int32_t arg_18h @ a1+0x18
entry a1, 64
l32r a10, 0x4008040c
call8 cpu_hal_set_vecbase ; sym.cpu_hal_set_vecbase
movi.n a10, 0
l32r a8, 0x40080468
callx8 a8
mov.n a2, a10
movi a10, 1
l32r a8, 0x40080468
callx8 a8
l32r a10, 0x40080444
l32r a12, 0x40080440
sub a12, a12, a10
movi.n a11, 0
l32r a8, 0x4008046c
callx8 a8
beqi a2, 5, 0x40080fdd
l32r a10, 0x4008044c
l32r a12, 0x40080448
sub a12, a12, a10
exc
re_extra_nw
extra_nw
ow_spill_nw
ctor
ctor
ctor
ctor
Dashboard
Strings
Imports
Search
Disassembly
Graph (entry0)
Hexdump

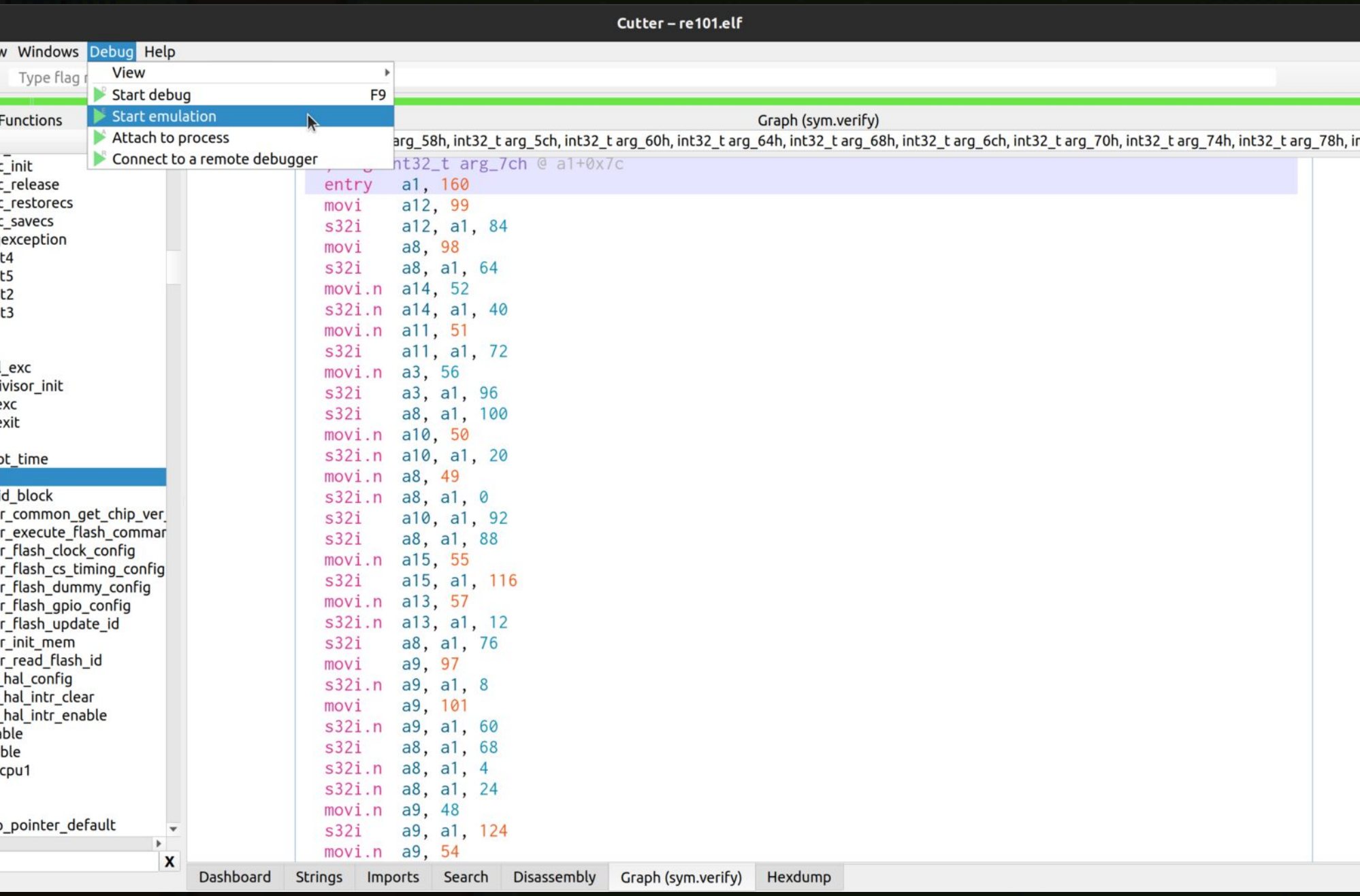


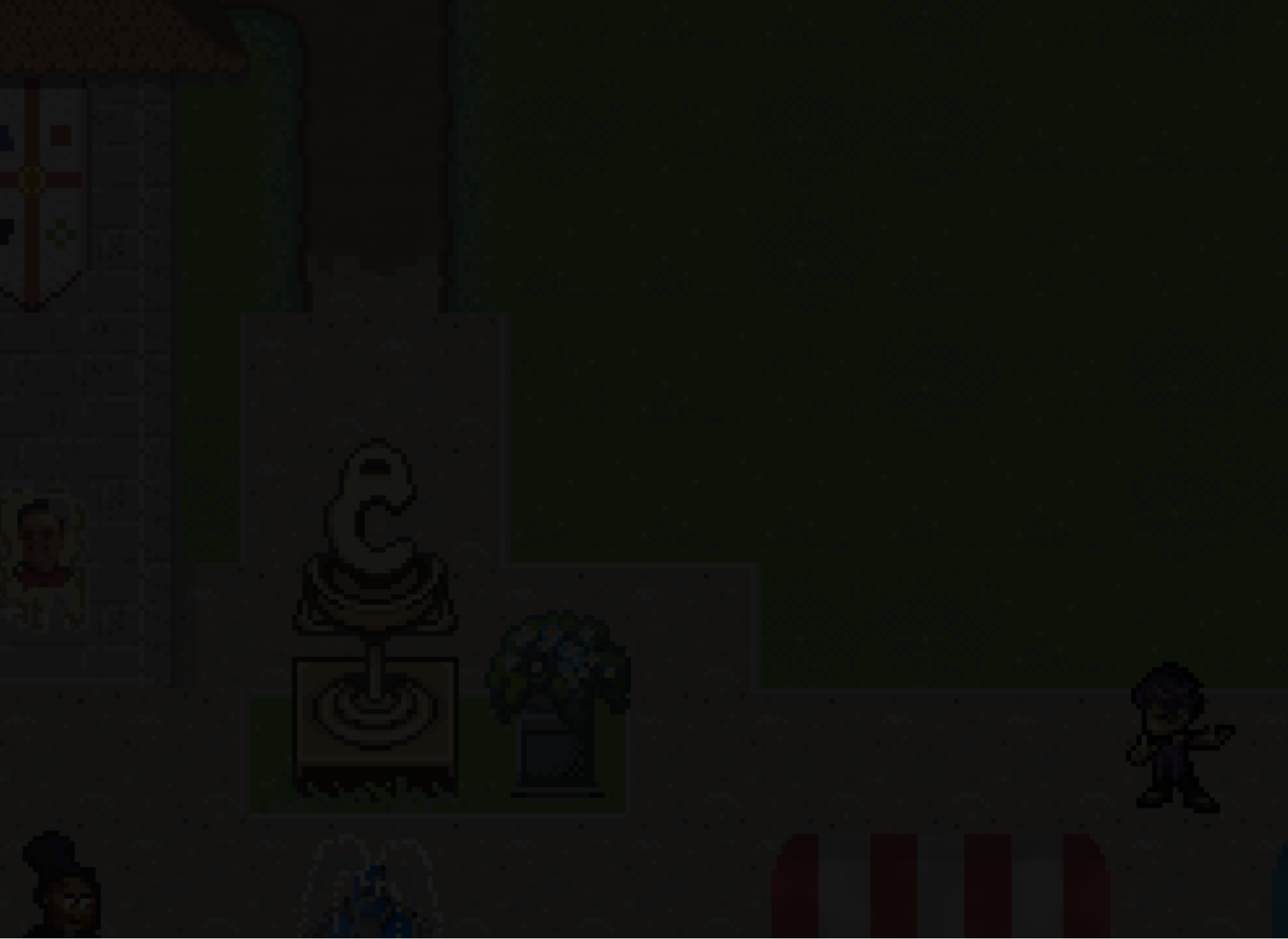

Cutter – re101.elf
w Windows Debug Help
Type flag name or address here
Functions
Graph (sym.app_main)
sym.app_main (int32_t arg_0h, int32_t arg_4h, int32_t arg_8h, int32_t arg_ch);
55: sym.app_main (int32_t arg_0h, int32_t arg_4h, int32_t arg_8h, int32_t arg_ch);
; arg int32_t arg_0h @ a1+0x0
; arg int32_t arg_4h @ a1+0x4
; arg int32_t arg_8h @ a1+0x8
; arg int32_t arg_ch @ a1+0xc
entry a1, 48
l32r a10, 0x400d038c
call8 printf ; sym.printf ; int printf(const char *format)
l32r a8, 0x400d0390
l32i a9, a8, 0 ; 0x3f402244
l32i.n a10, a8, 4
s32i.n a9, a1, 0
l32i.n a9, a8, 8
s32i.n a10, a1, 4
l32i.n a8, a8, 12
s32i.n a9, a1, 8
s32i a8, a1, 12
or a10, a1, a1
call8 verify ; sym.verify
beqz.n a10, 0x400d2390
d0394
; sym.printf ; int printf(const char *format)
l32r a10, 0x400d0398
call8 puts ; sym.puts
retw.n
Dashboard
Strings
Imports
Search
Disassembly
Graph (sym.app_main)
Hexdump

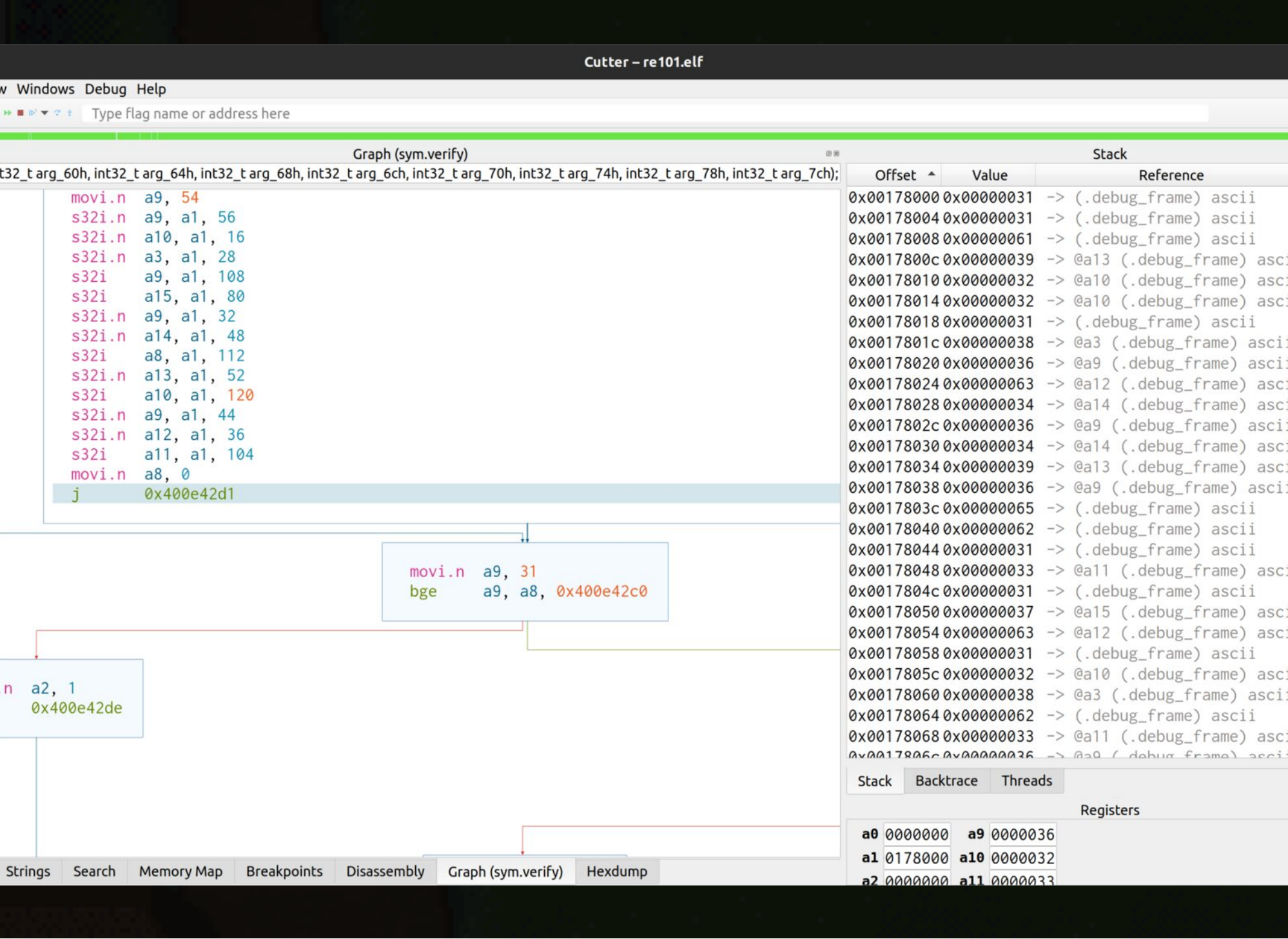


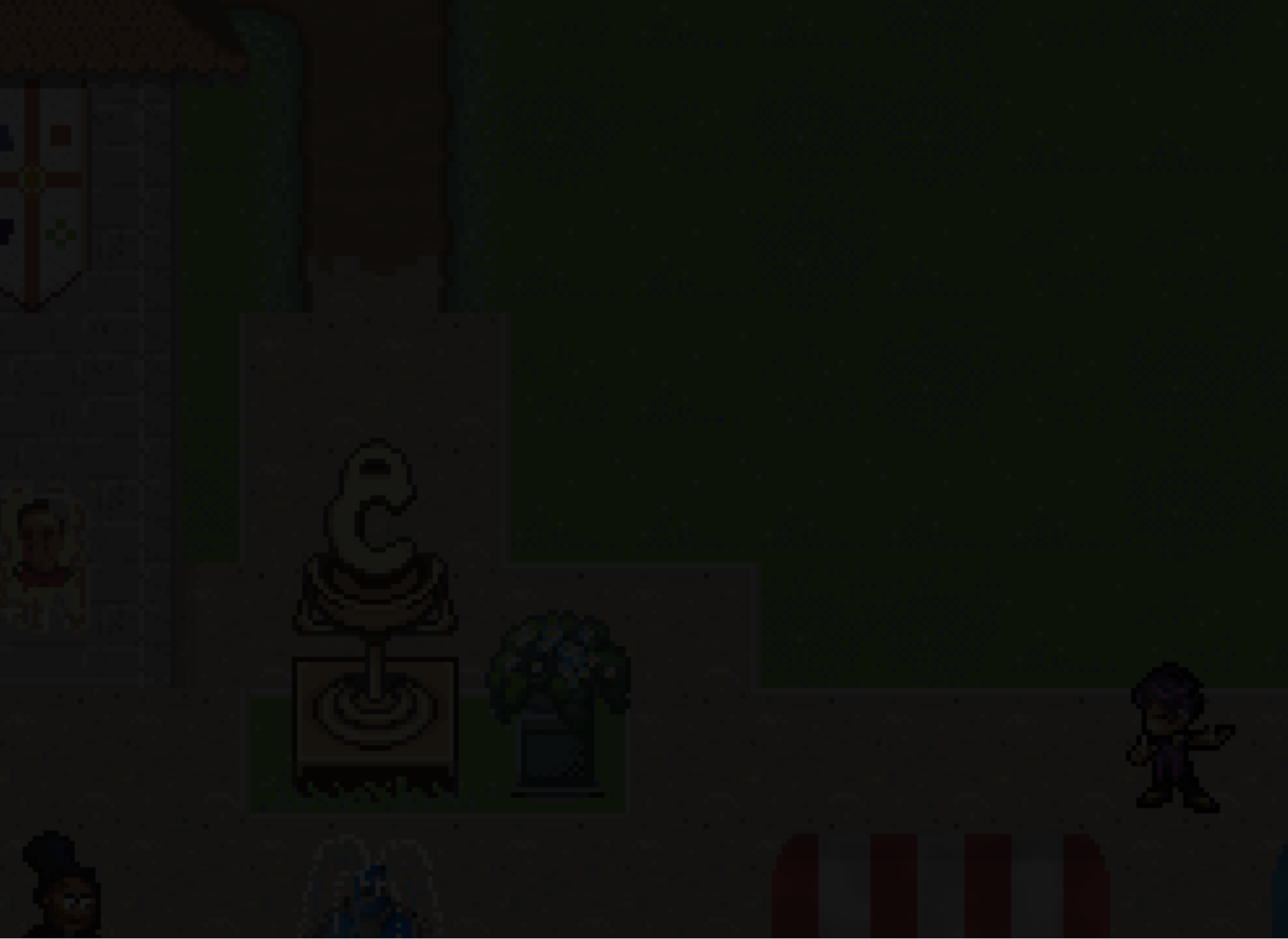

Cutter – re101.elf
w Windows Debug Help
Type flag name or address here
Functions
Graph (sym.verify)
_50h, int32_t arg_54h, int32_t arg_58h, int32_t arg_5ch, int32_t arg_60h, int32_t arg_64h, int32_t arg_68h, int32_t arg_6ch, int32_t arg_70h, int32_t arg_74h, int32_t arg_78h, in
; arg int32_t arg_7ch @ a1+0x7c
entry a1, 160
movi a12, 99
s32i a12, a1, 84
movi a8, 98
s32i a8, a1, 64
movi.n a14, 52
s32i.n a14, a1, 40
movi.n a11, 51
s32i a11, a1, 72
movi.n a3, 56
s32i a3, a1, 96
s32i a8, a1, 100
movi.n a10, 50
s32i.n a10, a1, 20
movi.n a8, 49
s32i.n a8, a1, 0
s32i a10, a1, 92
s32i a8, a1, 88
movi.n a15, 55
s32i a15, a1, 116
movi.n a13, 57
s32i.n a13, a1, 12
s32i a8, a1, 76
movi a9, 97
s32i.n a9, a1, 8
movi a9, 101
s32i.n a9, a1, 60
s32i a8, a1, 68
s32i.n a8, a1, 4
s32i.n a8, a1, 24
movi.n a9, 48
s32i a9, a1, 124
movi.n a9, 54
_init
_release
_restorecs
_savecs
exception
t4
t5
t2
t3
_exc
ivisor_init
xc
xit
ot_time
d_block
r_common_get_chip_ver
r_execute_flash_commar
r_flash_clock_config
r_flash_cs_timing_config
r_flash_dummy_config
r_flash_gpio_config
r_flash_update_id
r_init_mem
r_read_flash_id
_hal_config
_hal_intr_clear
_hal_intr_enable
ble
ble
cpu1
_pointer_default
Dashboard
Strings
Imports
Search
Disassembly
Graph (sym.verify)
Hexdump

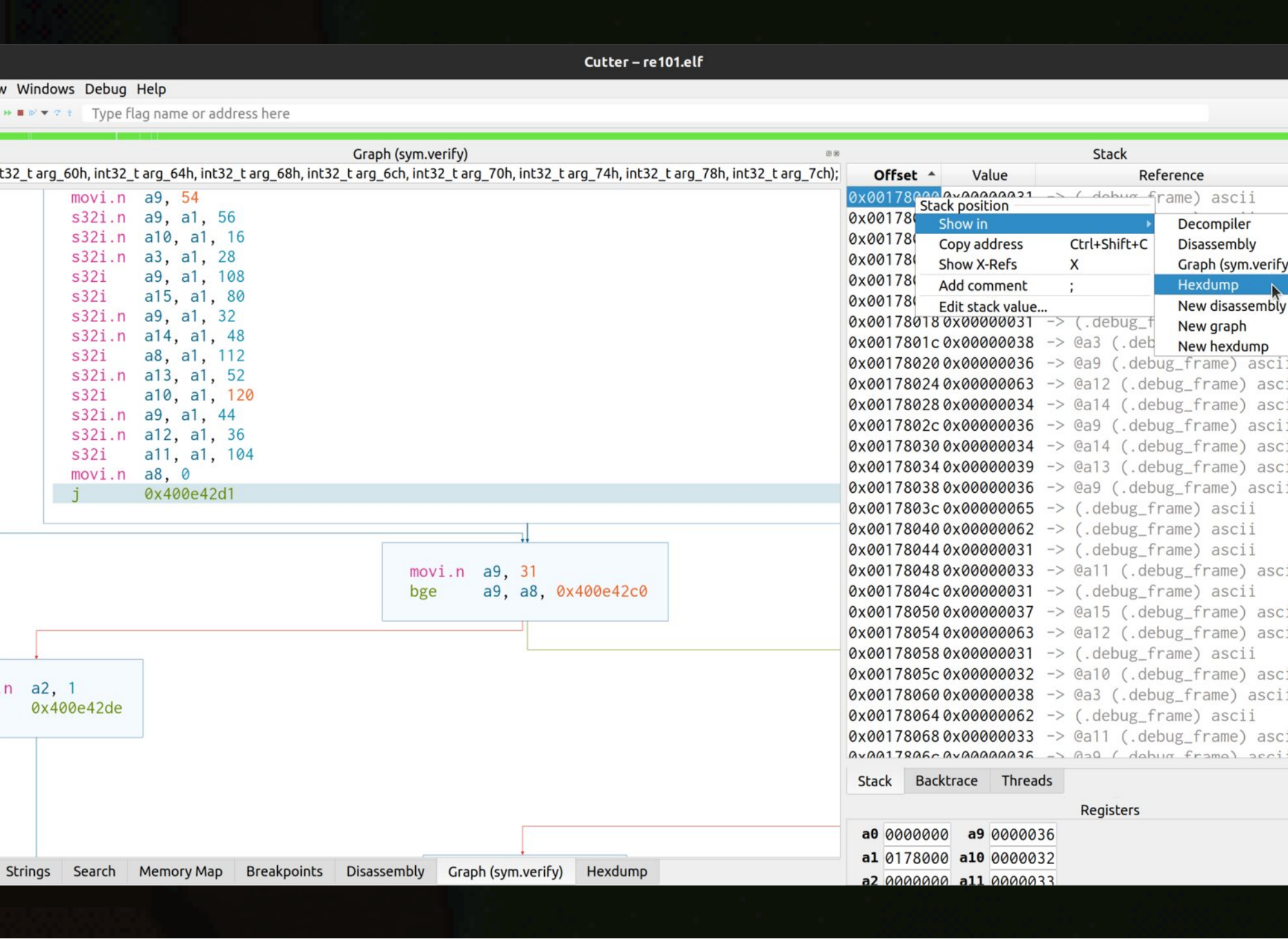


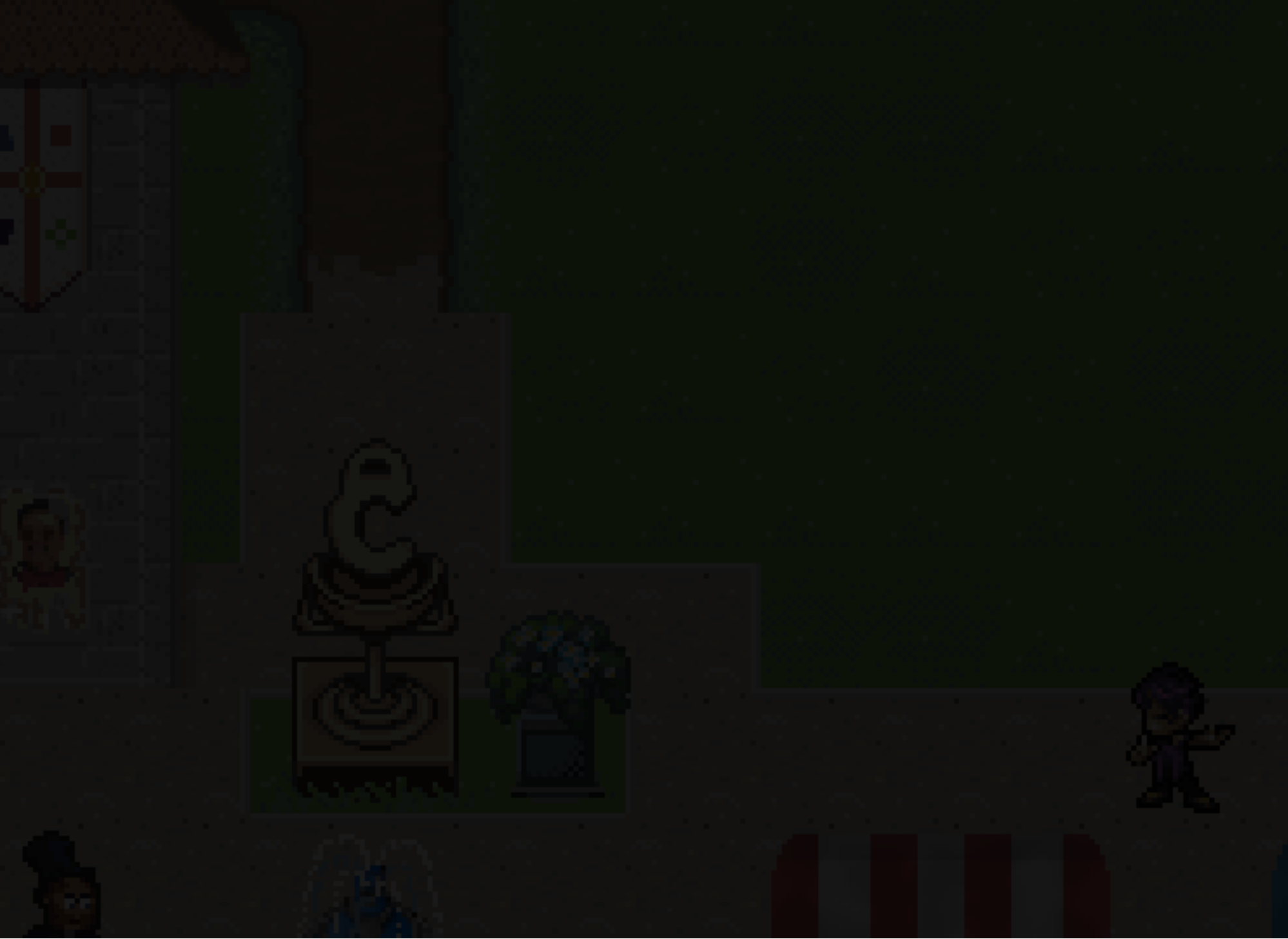

Cutter – re101.elf
Windows
Debug
Help
View
Start debug F9
Start emulation
Attach to process
Connect to a remote debugger
Type flag
Functions
Graph (sym.verify)
arg_58h, int32_t arg_5ch, int32_t arg_60h, int32_t arg_64h, int32_t arg_68h, int32_t arg_6ch, int32_t arg_70h, int32_t arg_74h, int32_t arg_78h, in
nt32_t arg_7ch @ a1+0x7c
entry a1, 160
movi a12, 99
s32i a12, a1, 84
movi a8, 98
s32i a8, a1, 64
movi.n a14, 52
s32i.n a14, a1, 40
movi.n a11, 51
s32i a11, a1, 72
movi.n a3, 56
s32i a3, a1, 96
s32i a8, a1, 100
movi.n a10, 50
s32i.n a10, a1, 20
movi.n a8, 49
s32i.n a8, a1, 0
s32i a10, a1, 92
s32i a8, a1, 88
movi.n a15, 55
s32i a15, a1, 116
movi.n a13, 57
s32i.n a13, a1, 12
s32i a8, a1, 76
movi a9, 97
s32i.n a9, a1, 8
movi a9, 101
s32i.n a9, a1, 60
s32i a8, a1, 68
s32i.n a8, a1, 4
s32i.n a8, a1, 24
movi.n a9, 48
s32i a9, a1, 124
movi.n a9, 54
Dashboard
Strings
Imports
Search
Disassembly
Graph (sym.verify)
Hexdump

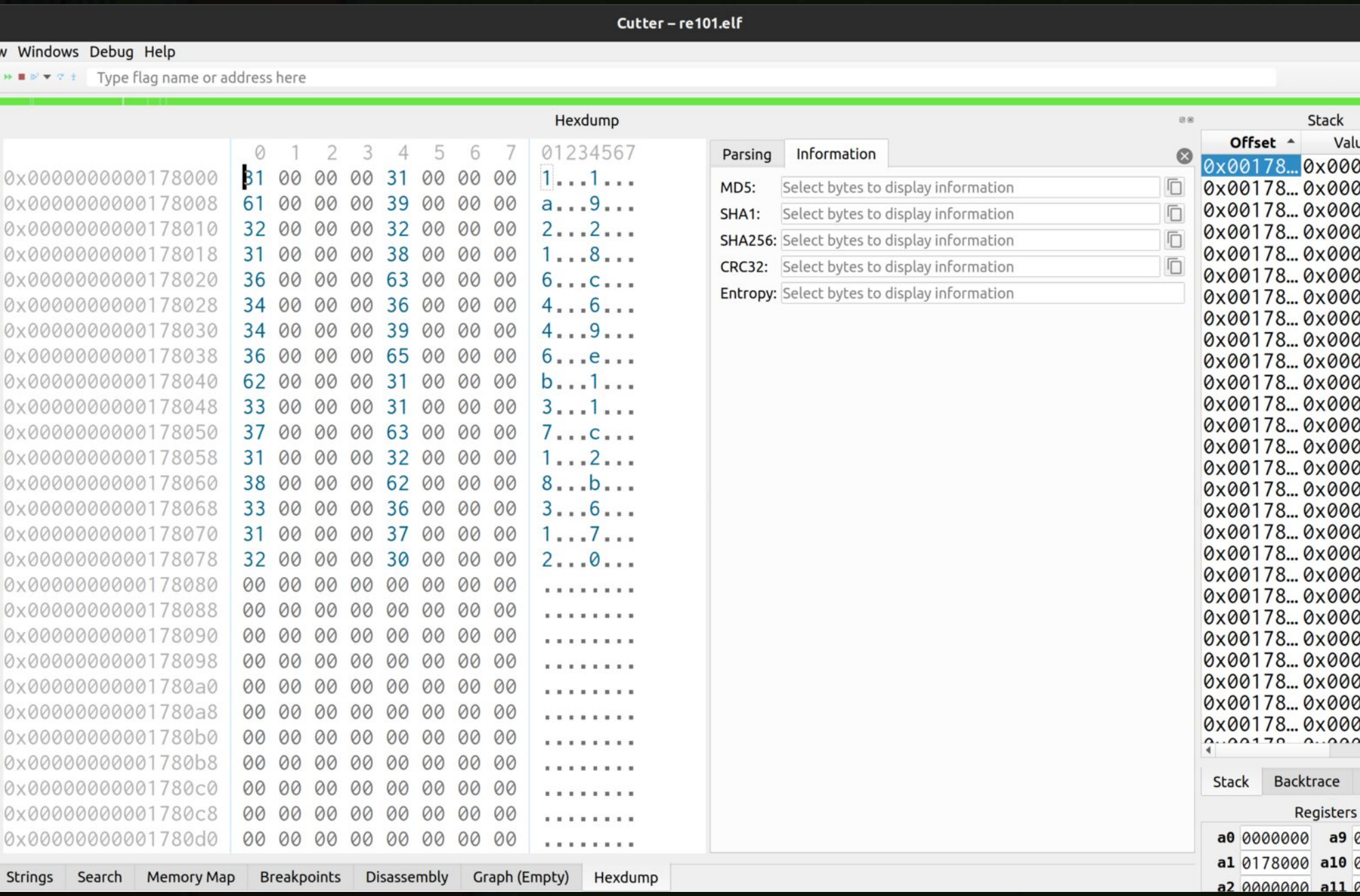


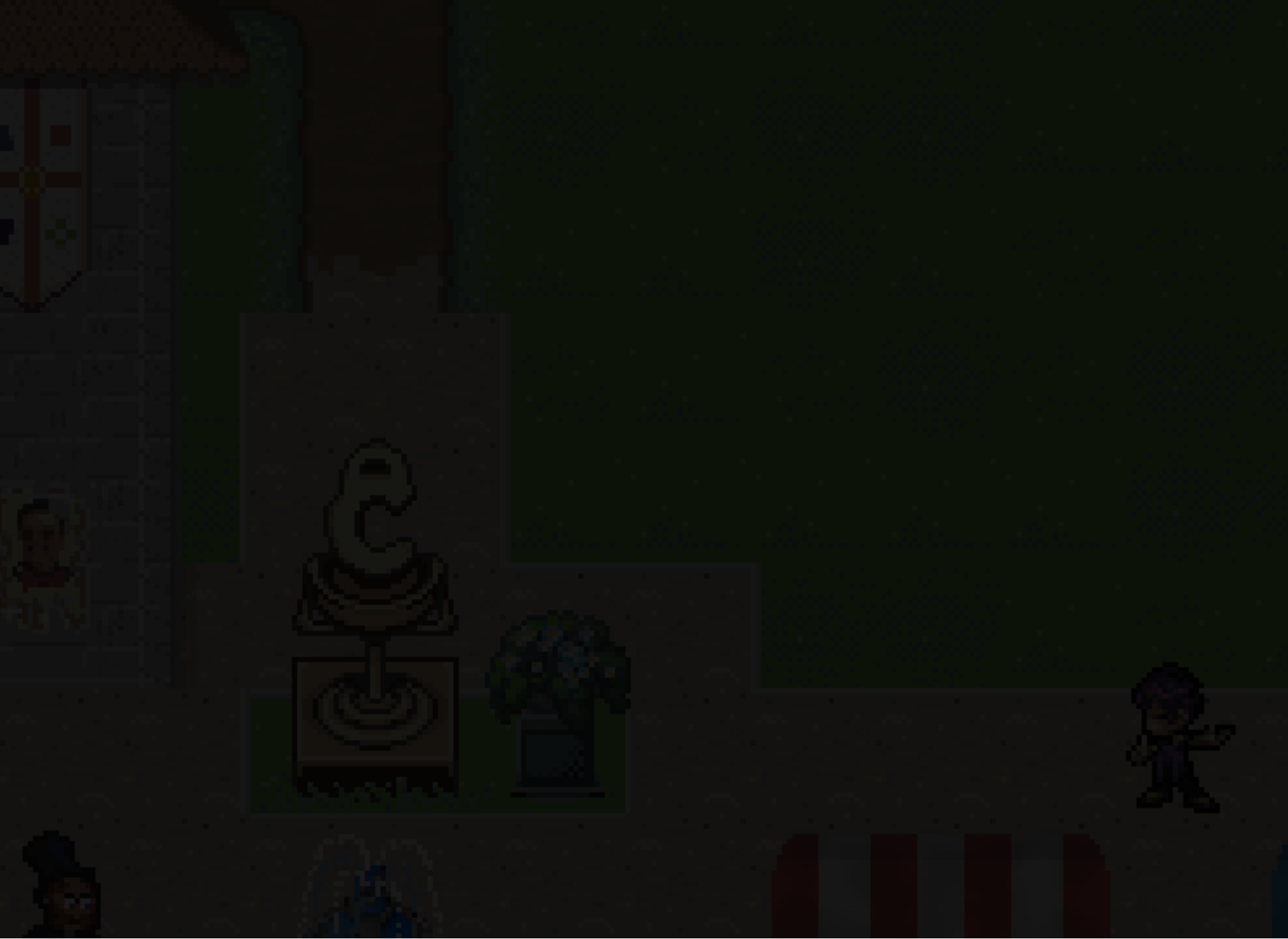

Cutter – re101.elf
w Windows Debug Help
Type flag name or address here
Graph (sym.verify)
t32_t arg_60h, int32_t arg_64h, int32_t arg_68h, int32_t arg_6ch, int32_t arg_70h, int32_t arg_74h, int32_t arg_78h, int32_t arg_7ch);
movi.n a9, 54
s32i.n a9, a1, 56
s32i.n a10, a1, 16
s32i.n a3, a1, 28
s32i a9, a1, 108
s32i a15, a1, 80
s32i.n a9, a1, 32
s32i.n a14, a1, 48
s32i a8, a1, 112
s32i.n a13, a1, 52
s32i a10, a1, 120
s32i.n a9, a1, 44
s32i.n a12, a1, 36
s32i a11, a1, 104
movi.n a8, 0
j 0x400e42d1
movi.n a9, 31
bge a9, a8, 0x400e42c0
n a2, 1
0x400e42de
Strings
Search
Memory Map
Breakpoints
Disassembly
Graph (sym.verify)
Hexdump
Stack
Offset
Value
Reference
0x00178000 0x00000031 -> (.debug_frame) ascii
0x00178004 0x00000031 -> (.debug_frame) ascii
0x00178008 0x00000061 -> (.debug_frame) ascii
0x0017800c 0x00000039 -> @a13 (.debug_frame) asci
0x00178010 0x00000032 -> @a10 (.debug_frame) asci
0x00178014 0x00000032 -> @a10 (.debug_frame) asci
0x00178018 0x00000031 -> (.debug_frame) ascii
0x0017801c 0x00000038 -> @a3 (.debug_frame) ascii
0x00178020 0x00000036 -> @a9 (.debug_frame) ascii
0x00178024 0x00000063 -> @a12 (.debug_frame) asci
0x00178028 0x00000034 -> @a14 (.debug_frame) asci
0x0017802c 0x00000036 -> @a9 (.debug_frame) ascii
0x00178030 0x00000034 -> @a14 (.debug_frame) asci
0x00178034 0x00000039 -> @a13 (.debug_frame) asci
0x00178038 0x00000036 -> @a9 (.debug_frame) ascii
0x0017803c 0x00000065 -> (.debug_frame) ascii
0x00178040 0x00000062 -> (.debug_frame) ascii
0x00178044 0x00000031 -> (.debug_frame) ascii
0x00178048 0x00000033 -> @a11 (.debug_frame) asci
0x0017804c 0x00000031 -> (.debug_frame) ascii
0x00178050 0x00000037 -> @a15 (.debug_frame) asci
0x00178054 0x00000063 -> @a12 (.debug_frame) asci
0x00178058 0x00000031 -> (.debug_frame) ascii
0x0017805c 0x00000032 -> @a10 (.debug_frame) asci
0x00178060 0x00000038 -> @a3 (.debug_frame) ascii
0x00178064 0x00000062 -> (.debug_frame) ascii
0x00178068 0x00000033 -> @a11 (.debug_frame) asci
Stack
Backtrace
Threads
Registers
a0 0000000
a9 0000036
a1 0178000
a10 0000032
a2 0000000
a11 0000033



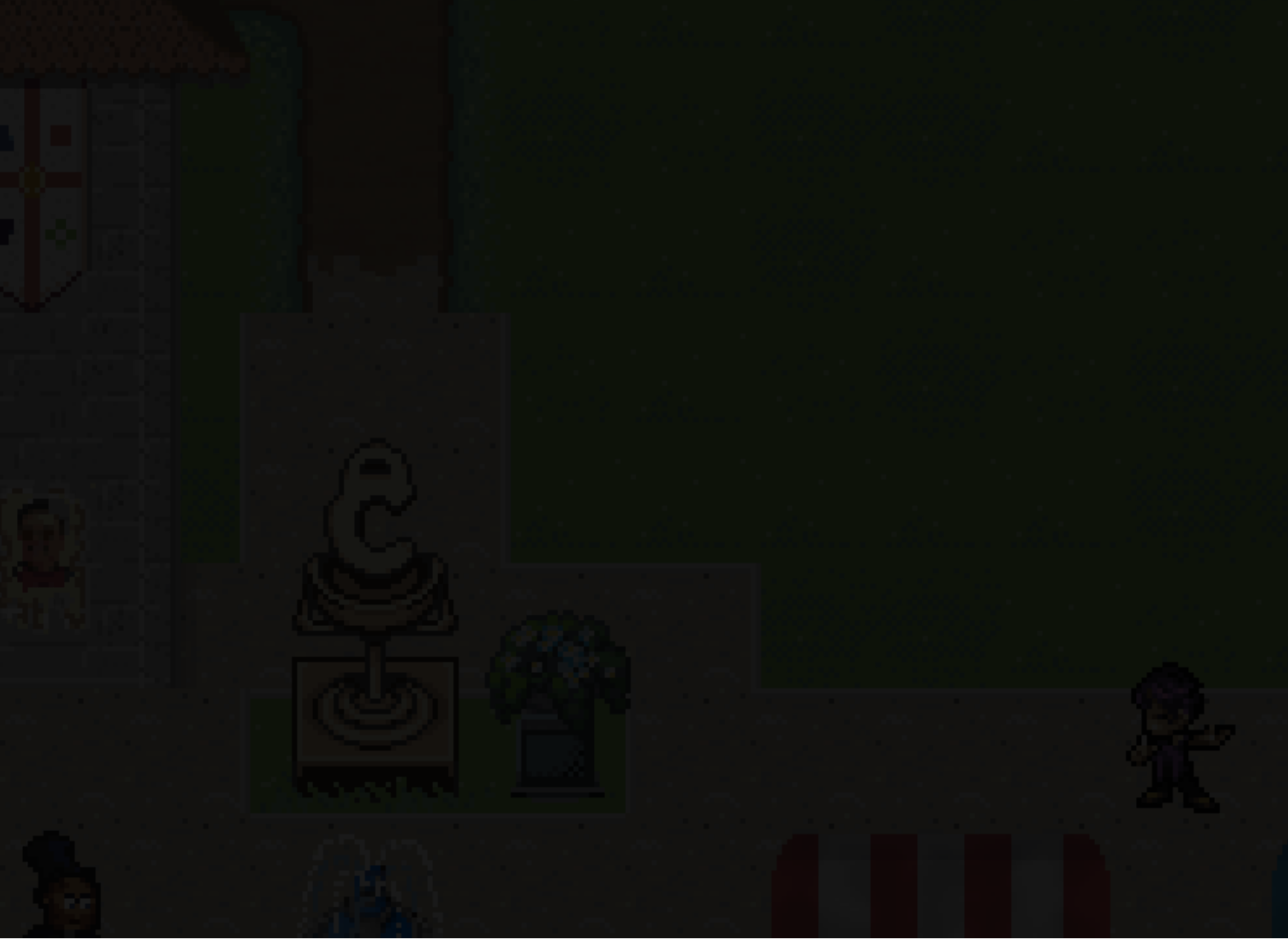

Cutter – re101.elf
w Windows Debug Help
Type flag name or address here
Graph (sym.verify)
t32_t arg_60h, int32_t arg_64h, int32_t arg_68h, int32_t arg_6ch, int32_t arg_70h, int32_t arg_74h, int32_t arg_78h, int32_t arg_7ch);
movi.n a9, 54
s32i.n a9, a1, 56
s32i.n a10, a1, 16
s32i.n a3, a1, 28
s32i a9, a1, 108
s32i a15, a1, 80
s32i.n a9, a1, 32
s32i.n a14, a1, 48
s32i a8, a1, 112
s32i.n a13, a1, 52
s32i a10, a1, 120
s32i.n a9, a1, 44
s32i.n a12, a1, 36
s32i a11, a1, 104
movi.n a8, 0
j 0x400e42d1
movi.n a9, 31
bge a9, a8, 0x400e42c0
n a2, 1
0x400e42de
Strings
Search
Memory Map
Breakpoints
Disassembly
Graph (sym.verify)
Hexdump
Stack
Offset
Value
Reference
Stack position
Show in
Copy address Ctrl+Shift+C
Show X-Refs X
Add comment ;
Edit stack value...
Decompiler
Disassembly
Graph (sym.verify)
Hexdump
New disassembly
New graph
New hexdump
0x00178018 0x00000031 -> (.debug_f
0x0017801c 0x00000038 -> @a3 (.deb
0x00178020 0x00000036 -> @a9 (.debug_frame) asci
0x00178024 0x00000063 -> @a12 (.debug_frame) asc
0x00178028 0x00000034 -> @a14 (.debug_frame) asc
0x0017802c 0x00000036 -> @a9 (.debug_frame) asci
0x00178030 0x00000034 -> @a14 (.debug_frame) asc
0x00178034 0x00000039 -> @a13 (.debug_frame) asc
0x00178038 0x00000036 -> @a9 (.debug_frame) asci
0x0017803c 0x00000065 -> (.debug_frame) ascii
0x00178040 0x00000062 -> (.debug_frame) ascii
0x00178044 0x00000031 -> (.debug_frame) ascii
0x00178048 0x00000033 -> @a11 (.debug_frame) asc
0x0017804c 0x00000031 -> (.debug_frame) ascii
0x00178050 0x00000037 -> @a15 (.debug_frame) asc
0x00178054 0x00000063 -> @a12 (.debug_frame) asc
0x00178058 0x00000031 -> (.debug_frame) ascii
0x0017805c 0x00000032 -> @a10 (.debug_frame) asc
0x00178060 0x00000038 -> @a3 (.debug_frame) asci
0x00178064 0x00000062 -> (.debug_frame) ascii
0x00178068 0x00000033 -> @a11 (.debug_frame) asc
Stack
Backtrace
Threads
Registers
a0 0000000
a1 0178000
a2 0000000
a9 0000036
a10 0000032
a11 0000033

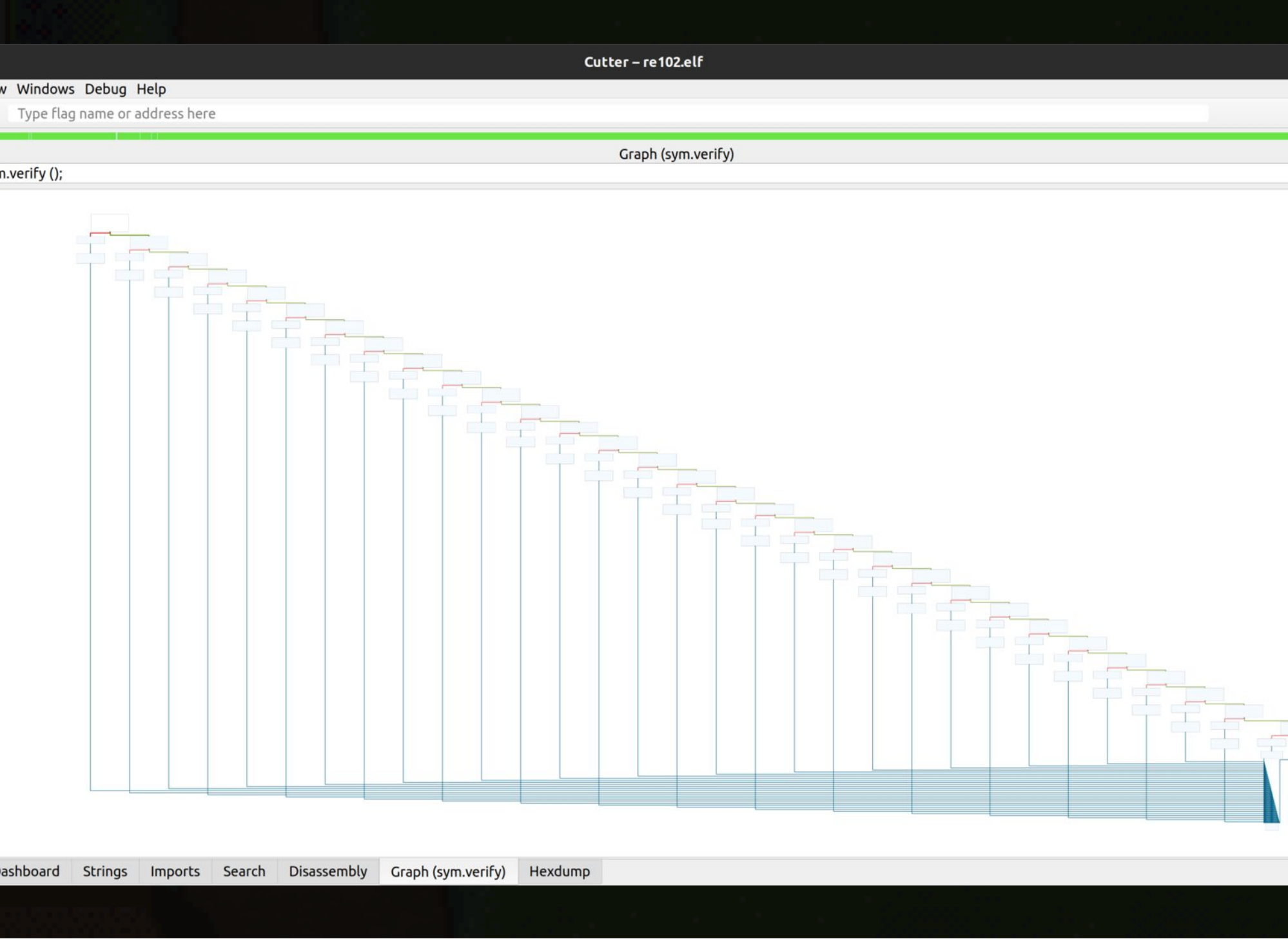


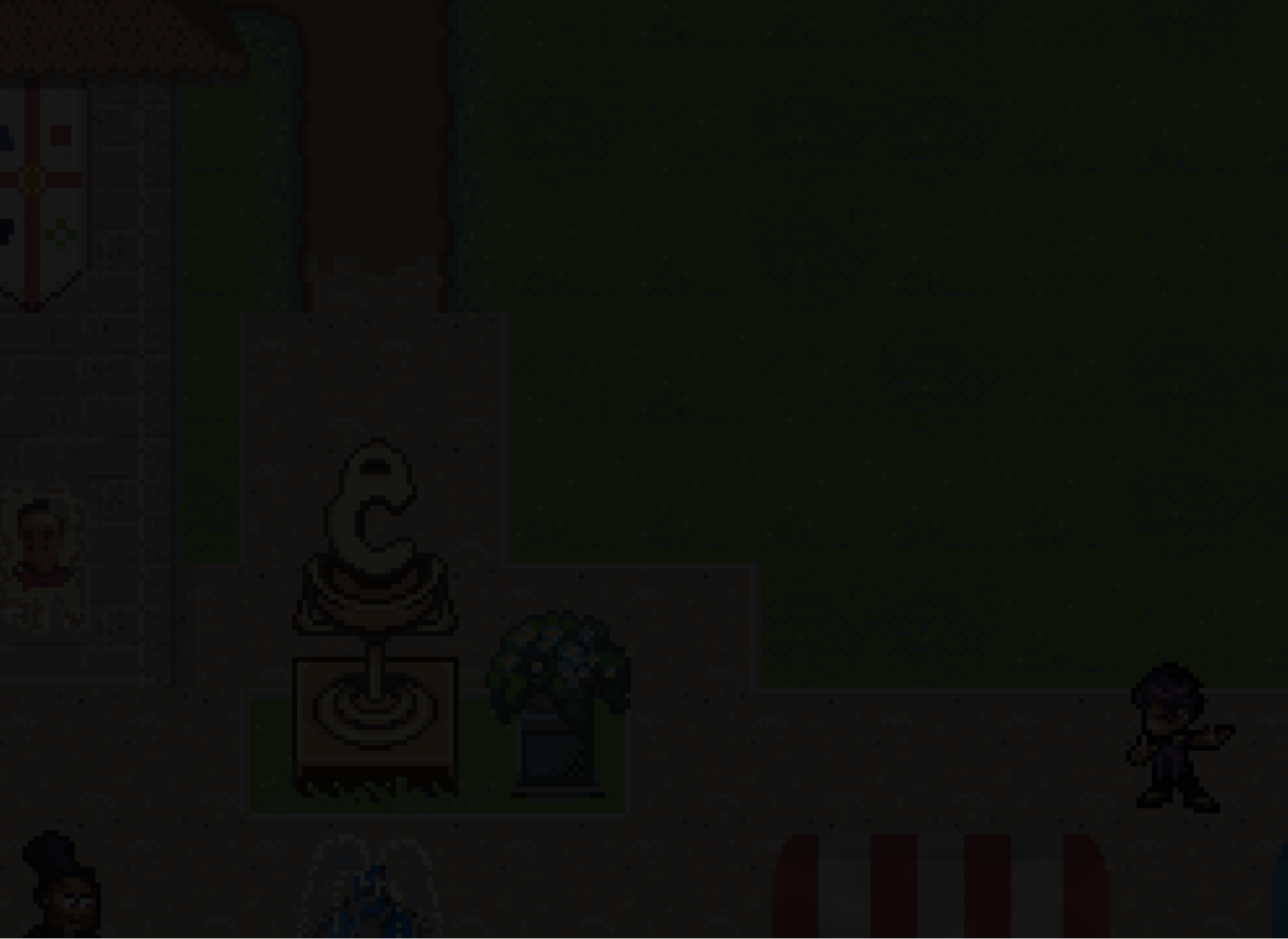

Cutter – re101.elf
w Windows Debug Help
Type flag name or address here
Hexdump
0 1 2 3 4 5 6 7 01234567
0x0000000000178000 31 00 00 00 31 00 00 00 1...1...
0x0000000000178008 61 00 00 00 39 00 00 00 a...9...
0x0000000000178010 32 00 00 00 32 00 00 00 2...2...
0x0000000000178018 31 00 00 00 38 00 00 00 1...8...
0x0000000000178020 36 00 00 00 63 00 00 00 6...c...
0x0000000000178028 34 00 00 00 36 00 00 00 4...6...
0x0000000000178030 34 00 00 00 39 00 00 00 4...9...
0x0000000000178038 36 00 00 00 65 00 00 00 6...e...
0x0000000000178040 62 00 00 00 31 00 00 00 b...1...
0x0000000000178048 33 00 00 00 31 00 00 00 3...1...
0x0000000000178050 37 00 00 00 63 00 00 00 7...c...
0x0000000000178058 31 00 00 00 32 00 00 00 1...2...
0x0000000000178060 38 00 00 00 62 00 00 00 8...b...
0x0000000000178068 33 00 00 00 36 00 00 00 3...6...
0x0000000000178070 31 00 00 00 37 00 00 00 1...7...
0x0000000000178078 32 00 00 00 30 00 00 00 2...0...
0x0000000000178080 00 00 00 00 00 00 00 00 ....
0x0000000000178088 00 00 00 00 00 00 00 00 ....
0x0000000000178090 00 00 00 00 00 00 00 00 ....
0x0000000000178098 00 00 00 00 00 00 00 00 ....
0x00000000001780a0 00 00 00 00 00 00 00 00 ....
0x00000000001780a8 00 00 00 00 00 00 00 00 ....
0x00000000001780b0 00 00 00 00 00 00 00 00 ....
0x00000000001780b8 00 00 00 00 00 00 00 00 ....
0x00000000001780c0 00 00 00 00 00 00 00 00 ....
0x00000000001780c8 00 00 00 00 00 00 00 00 ....
0x00000000001780d0 00 00 00 00 00 00 00 00 ....
Parsing
Information
MD5: Select bytes to display information
SHA1: Select bytes to display information
SHA256: Select bytes to display information
CRC32: Select bytes to display information
Entropy: Select bytes to display information
Stack
Offset
Valu
0x00178... 0x000
Stack
Backtrace
Registers
a0 0000000
a9
a1 0178000
a10
a2 0000000
a11
Strings
Search
Memory Map
Breakpoints
Disassembly
Graph (Empty)
Hexdump

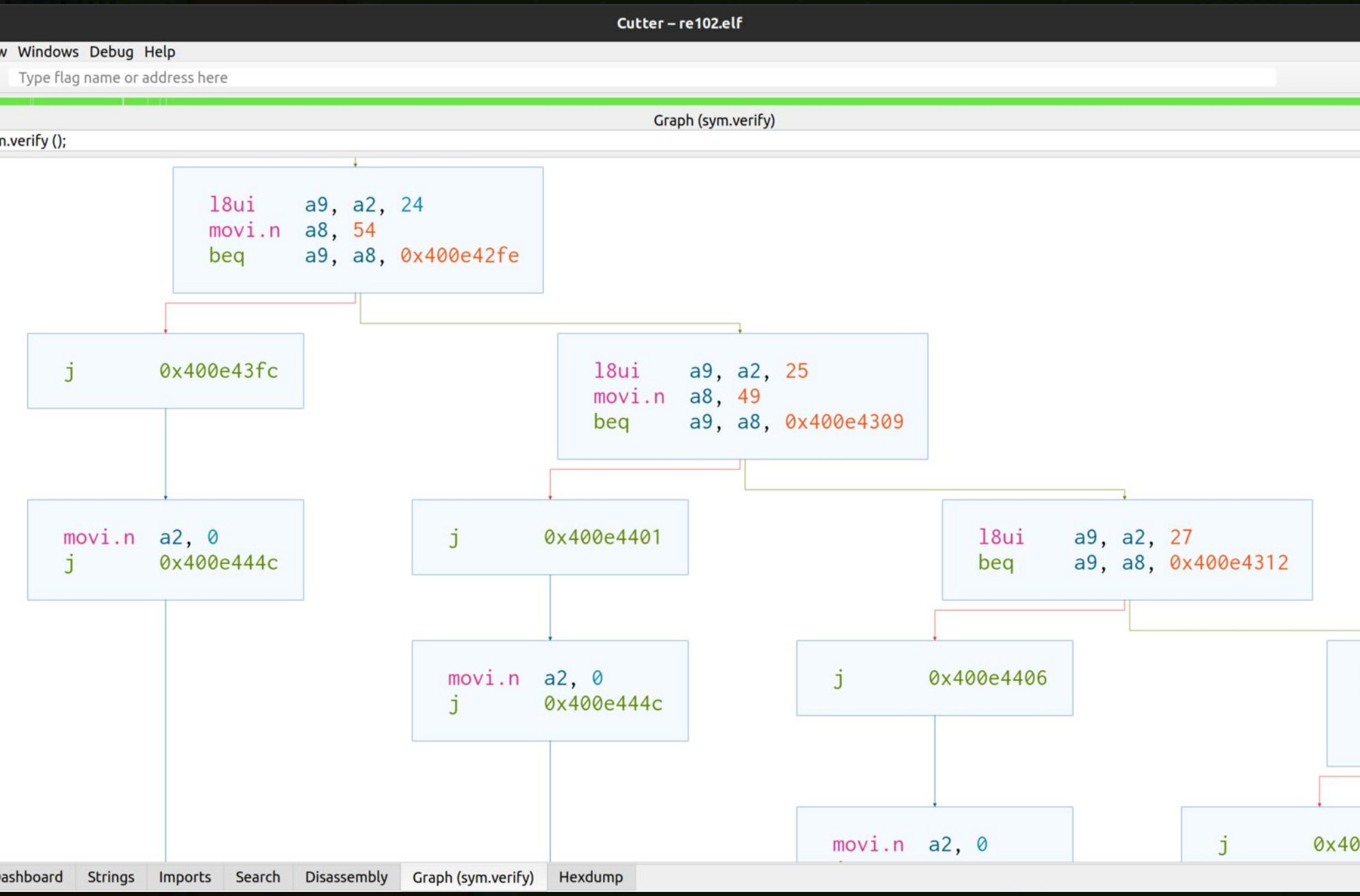


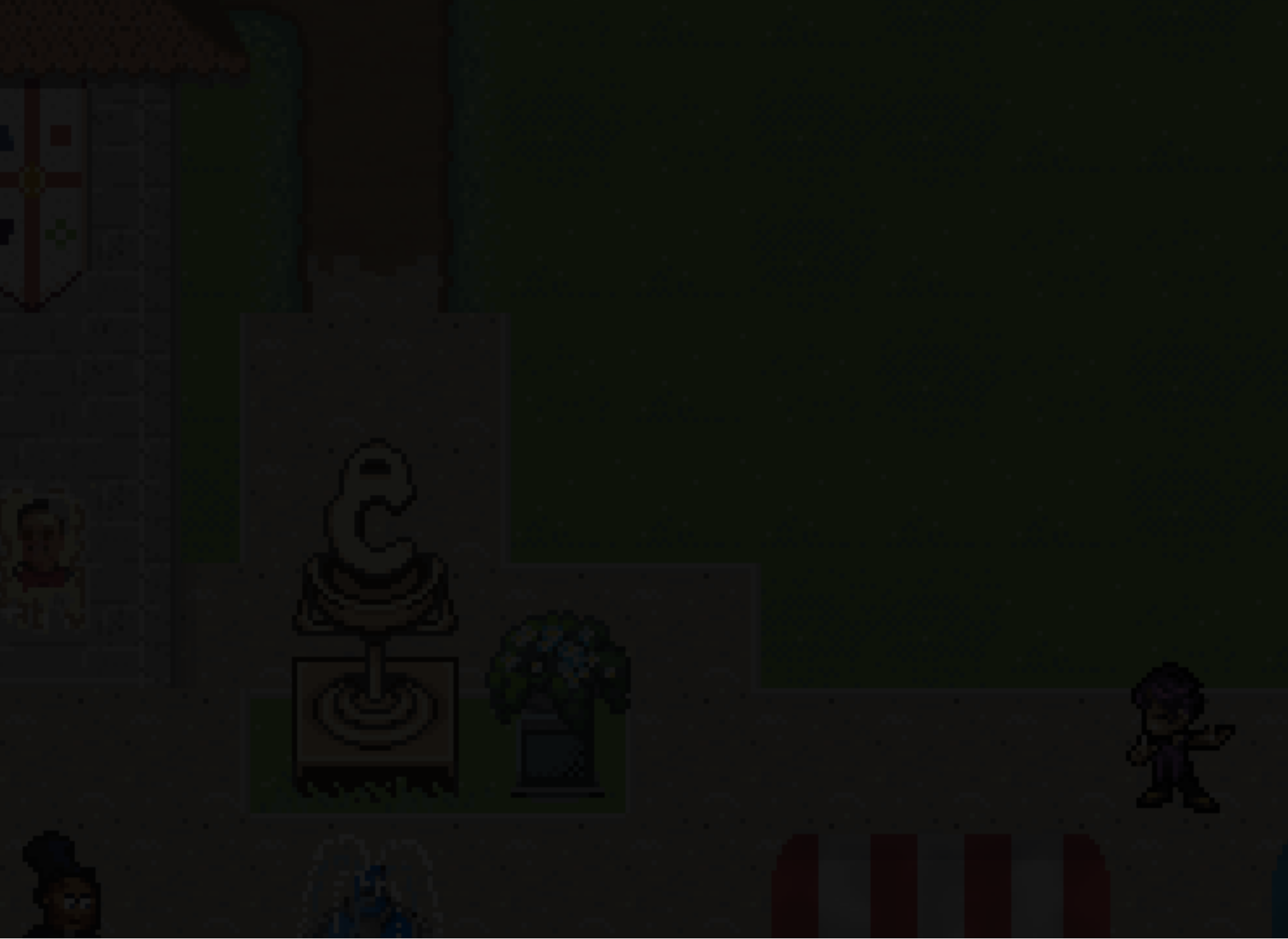

rel02.elf

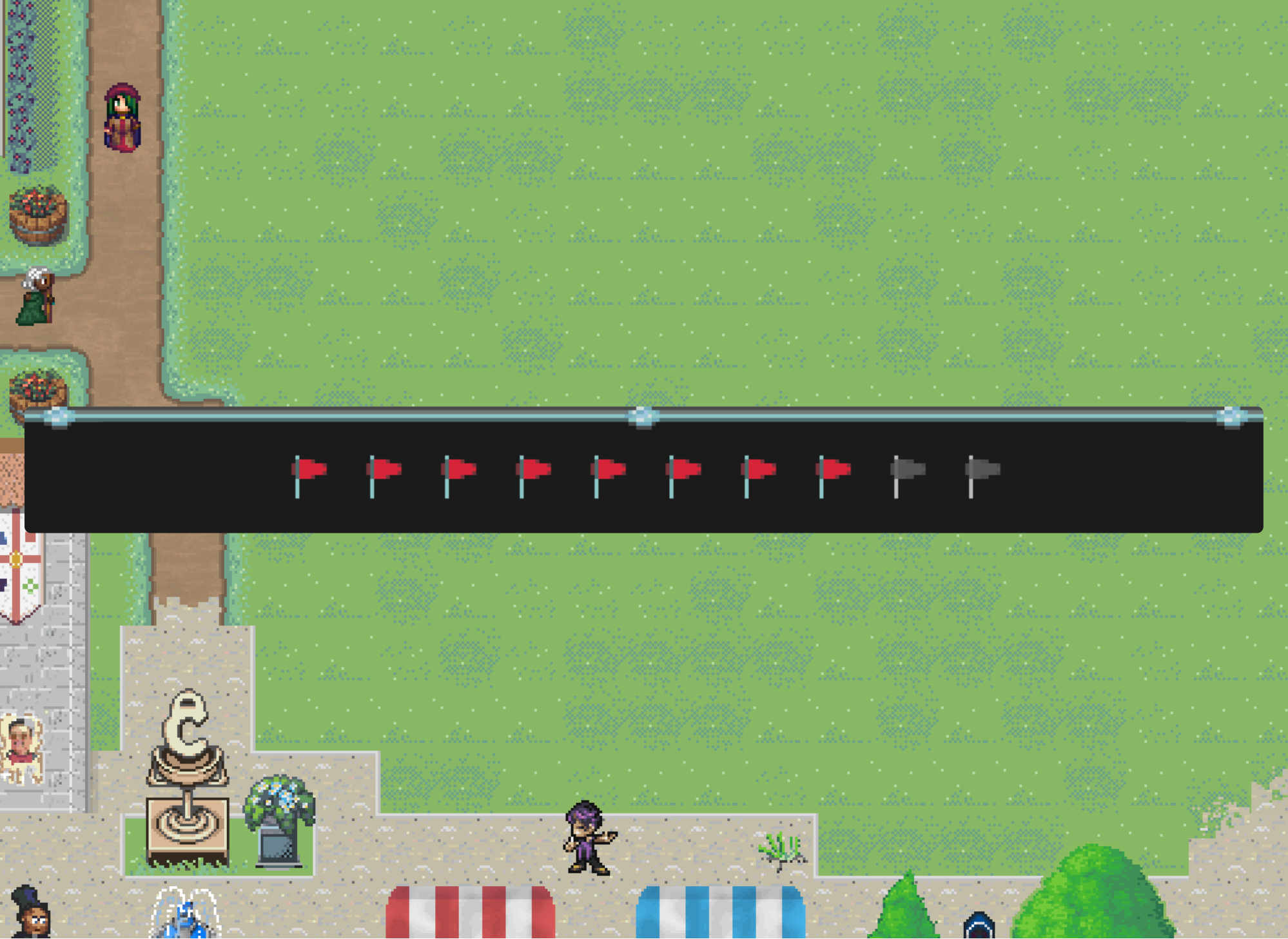



Cutter – re102.elf
w Windows Debug Help
Type flag name or address here
Graph (sym.verify)
n.verify ();
ashboard
Strings
Imports
Search
Disassembly
Graph (sym.verify)
Hexdump




Cutter – re102.elf
w Windows Debug Help
Type flag name or address here
Graph (sym.verify)
n.verify ();
l8ui a9, a2, 24
movi.n a8, 54
beq a9, a8, 0x400e42fe
j 0x400e43fc
l8ui a9, a2, 25
movi.n a8, 49
beq a9, a8, 0x400e4309
movi.n a2, 0
j 0x400e444c
j 0x400e4401
l8ui a9, a2, 27
beq a9, a8, 0x400e4312
movi.n a2, 0
j 0x400e444c
j 0x400e4406
movi.n a2, 0
j 0x40
ashboard
Strings
Imports
Search
Disassembly
Graph (sym.verify)
Hexdump






About the 'bonus' flag


About the 'bonus' flag
• Intended to be more difficult than other flags.
• Not counted by the FLAGBOT, you can get the full Badge Wizard role with other 9 flags.
• There is no actual flag, you have to make it.
• No intended solution - you are on your own.


About the 'bonus' flag
• You need to modify the flash image to write it back to the badge.
• You can damage the firmware if you do this incorrectly.
• Because the firmware wasn't released until after NSEC, this means you risk "bricking" the badge.

About the 'bonus' flag

Dump the firmware and run strings:

```
static const char firmware_dump_flag[] =
    "Have you dumped the firmware? Here is your flag "
    "[FLAG-JTAGPower0verwhelming]. Now flip the right bit in memory to "
    "activate the last (10th) flag icon in the status bar on screen.";
```


About the 'bonus' flag
Try to dump the full contents of the flash during the last hacking session.
It can be tricky to get it right...


Hacking session! (final)


Auto-complete my flag...
1: user@localhost: ~


Auto-complete my flag...
Welcome to the nsec badge console!
Type 'help' to get the list of commands.
Use UP/DOWN arrows to navigate through command history.
Press TAB when typing command name to auto-complete.


Auto-complete my flag...
1 Welcome to the nsec badge console!
2 Type 'help' to get the list of commands.
3 Use UP/DOWN arrows to navigate through command history.
4 Press TAB when typing command name to auto-complete.


Auto-complete my flag...
1 Welcome to the nsec badge console!
2 Type 'help' to get the list of commands.
3 Use UP/DOWN arrows to navigate through command history.
4 Press TAB when typing command name to auto-complete.
Just try it for yourself...


The 'bonus' flag

The 'bonus' flag

```
$ esptool.py read_flash --help
usage: esptool read_flash [-h] [--spi-connection SPI_CONNECTION]
                          [--no-progress] address size filename

positional arguments:
  address                 Start address
  size                    Size of region to dump
  filename                Name of binary dump

optional arguments:
  -h, --help              show this help message and exit
  --spi-connection SPI_CONNECTION, -sc SPI_CONNECTION
                          ESP32-only argument. Override default SPI Flash
                          connection. Value can be SPI, HSPI or a comma-sepa
                          list of 5 I/O numbers to use for SPI flash
                          (CLK,Q,D,HD,CS).
```

https://github.com/espressif/esptool

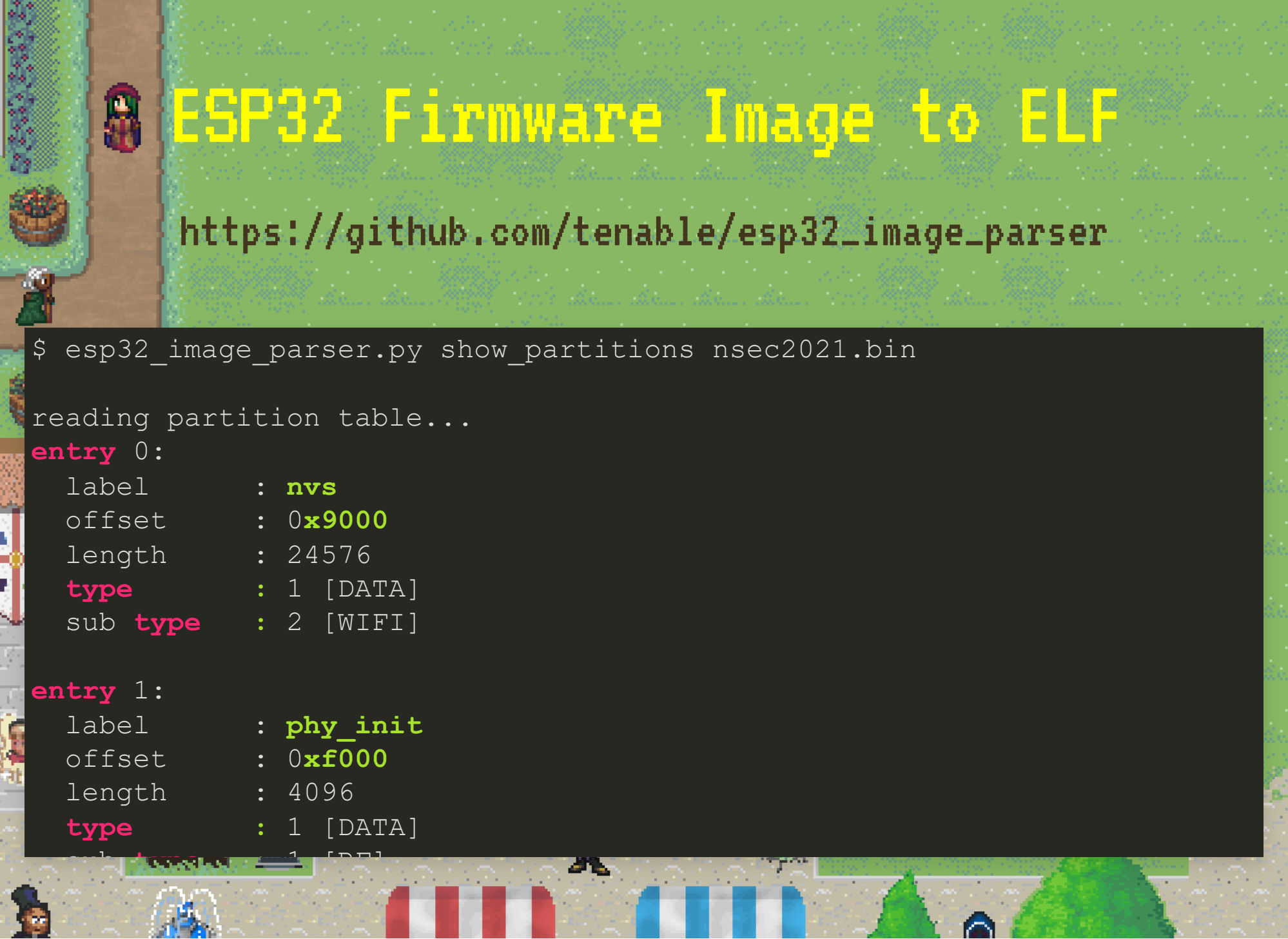

The 'bonus' flag
Alternatively, download from Github:
https://github.com/nsec/nsec-badge/releases/download/nsec21/nsec2021.bin
* wasn't an option during the CTF.


You have the flash image,
now what?


You have the flash image,
now what?
Extracting an ELF From an ESP32 -
Chris Lyne and Nick Miles (Shmoocon 2020)


You have the flash image,
now what?
Extracting an ELF From an ESP32 -
Chris Lyne and Nick Miles (Shmoocon 2020)
https://www.youtube.com/watch?v=w4_3vwN_2dI


You have the flash image,
now what?
Extracting an ELF From an ESP32 -
Chris Lyne and Nick Miles (Shmoocon 2020)
https://www.youtube.com/watch?v=w4_3vwN_2dI
This video was discovered by participants.


ESP32 Firmware Image to ELF
https://github.com/tenable/esp32_image_parser
$ esp32_image_parser.py show_partitions nsec2021.bin
reading partition table...
entry 0:
label : nvs
offset : 0x9000
length : 24576
type : 1 [DATA]
sub type : 2 [WIFI]
entry 1:
label : phy_init
offset : 0xf000
length : 4096
type : 1 [DATA]

Dump NVS partition

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/nvs_flash.html

```
$ esp32_image_parser.py dump_nvs nsec2021.bin -partition nvs | wc -l
2176

$ esp32_image_parser.py dump_nvs nsec2021.bin -partition nvs | grep Key
      Key : storage
      Key : misc
      Key : log
      Key : log
      Key : nvs.net80211
      Key : opmode
      Key : sta.ssid
      Key : sta.ssid
      Key : sta.authmode
      Key : sta.pswd
      Key : sta.pswd
```


Dump NVS partition
What is the correct key?


Dump NVS partition
If you downloaded the stock flash image from Github,
you are out of luck.


Dump NVS partition
If you downloaded the stock flash image from Github,
you are out of luck.
The badge stores the flags in the save key,
which is created on the fly.


Dump NVS partition
If you downloaded the stock flash image from Github,
you are out of luck.
The badge stores the flags in the save key,
which is created on the fly.
The only way to know this is dump the flash, activate some
flag, dump it again, then compare two files.


Finding the saved flags


Finding the saved flags
• Compare the two dumped flash images


Finding the saved flags
• Compare the two dumped flash images
• Locate the key that has changed

Finding the saved flags

- Compare the two dumped flash images
- Locate the key that has changed

```
Flag 1
00 00 00 00 01 00 00 00 01 01 00 00 01 00 19 00 01 00 00 00 FF 00 FF 00 01 00 00 00 00 00 00 00 00 00 00 00

Flag 2
00 00 00 00 01 00 00 00 01 01 00 00 01 00 19 00 01 00 00 00 FF 00 FF 00 01 01 00 00 00 00 00 00 00 00 00 00
```

https://hideandsec.sh/books/ctf/page/northsec-2021-badge-writeup

Finding the saved flags

Dump the partition with

```
$ esp32_image_parser.py dump_partition -partition nvs nsec2021.bin
Dumping partition 'nvs' to nvs_out.bin
```

Locate the correct byte

and change its value in a HEX editor.

Write the partition back

We extracted only one partition from the full flash image, so we cannot simply write it back - this will corrupt the firmware.

```
$ stat -c"%n %s" *bin
nsec2021.bin 16777216
nvs_out.bin  24576
```

We need to find the correct offset.

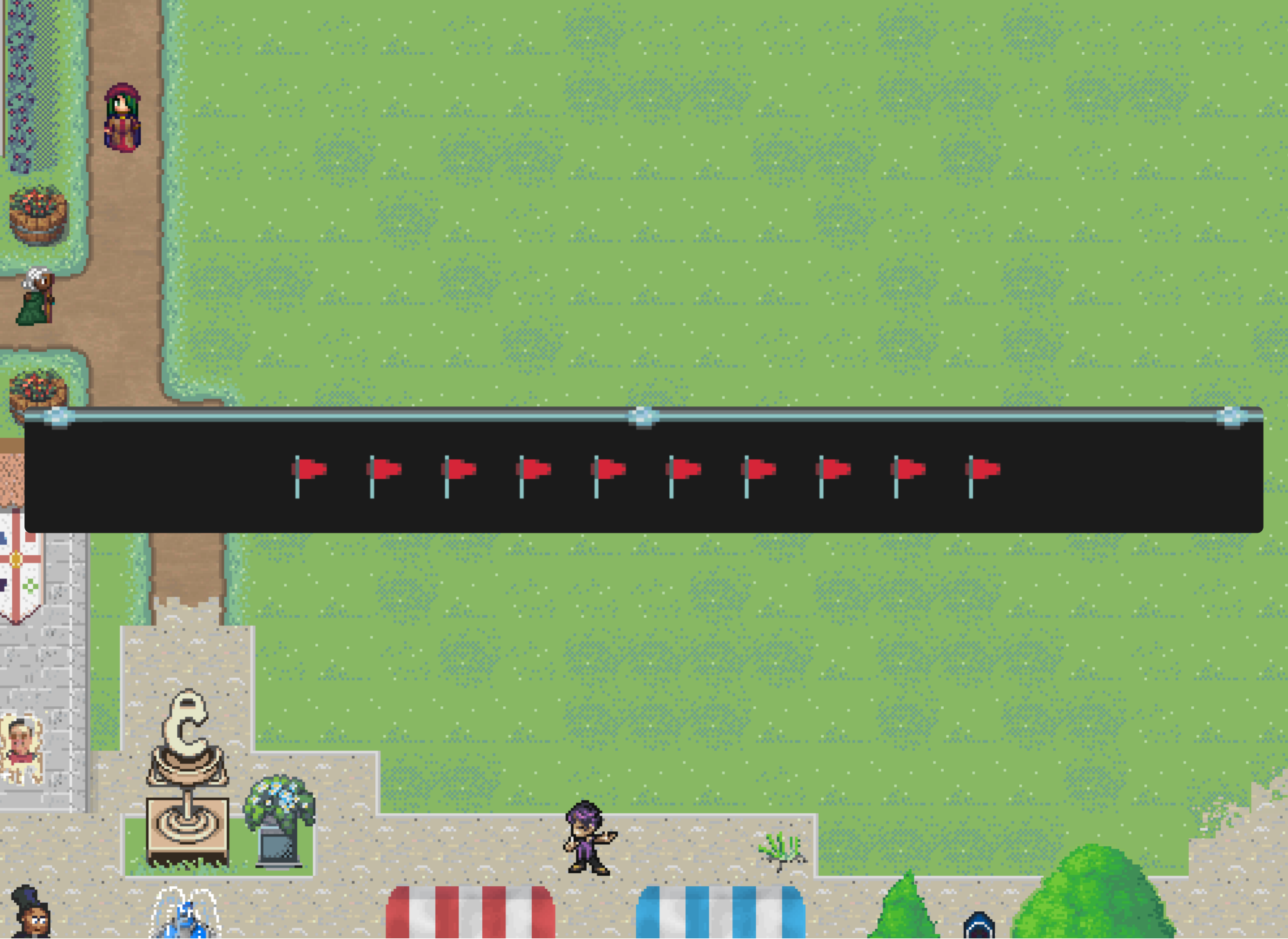
Write the partition back

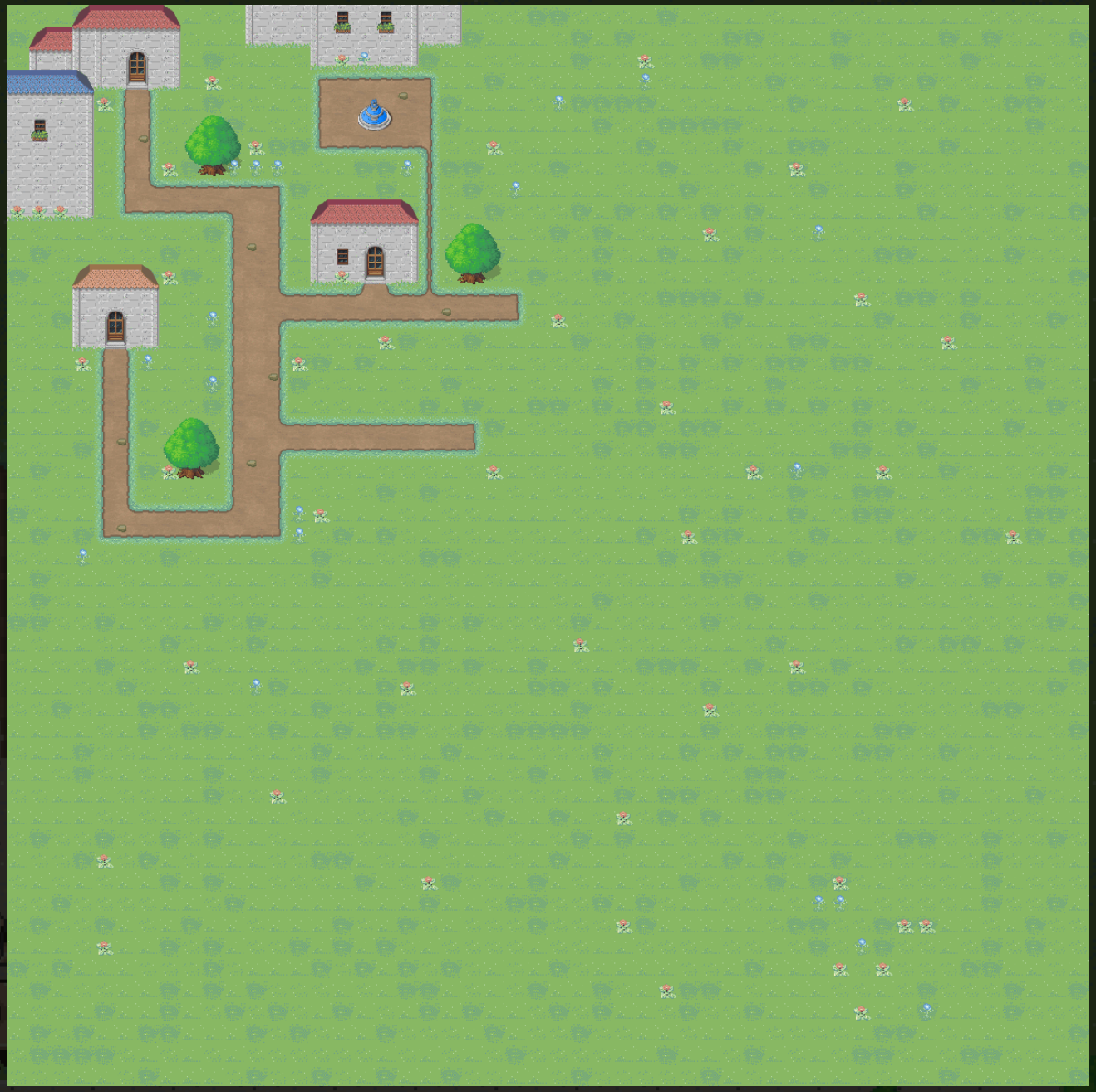
```
$ esp32_image_parser.py show_partitions nsec2021.bin

reading partition table...
entry 0:
  label       : nvs
  offset      : 0x9000
  length      : 24576
  type        : 1 [DATA]
  sub type    : 2 [WIFI]

...
```

Write the partition back

```
┌──(root💀olivier)-[~/esp32_image_parser]
└─# python3 esptool/esptool.py --chip esp32 --port /dev/ttyUSB0 -b 115200 write_flash 0x9000 nvs_out.bin
esptool.py v3.2-dev
Serial port /dev/ttyUSB0
Connecting....
Chip is ESP32-D0WD-V3 (revision 3)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 40:f5:20:57:89:a0
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x00009000 to 0x0000efff...
Compressed 24576 bytes to 1858...
Wrote 24576 bytes (1858 compressed) at 0x00009000 in 0.4 seconds (effective 479.9 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

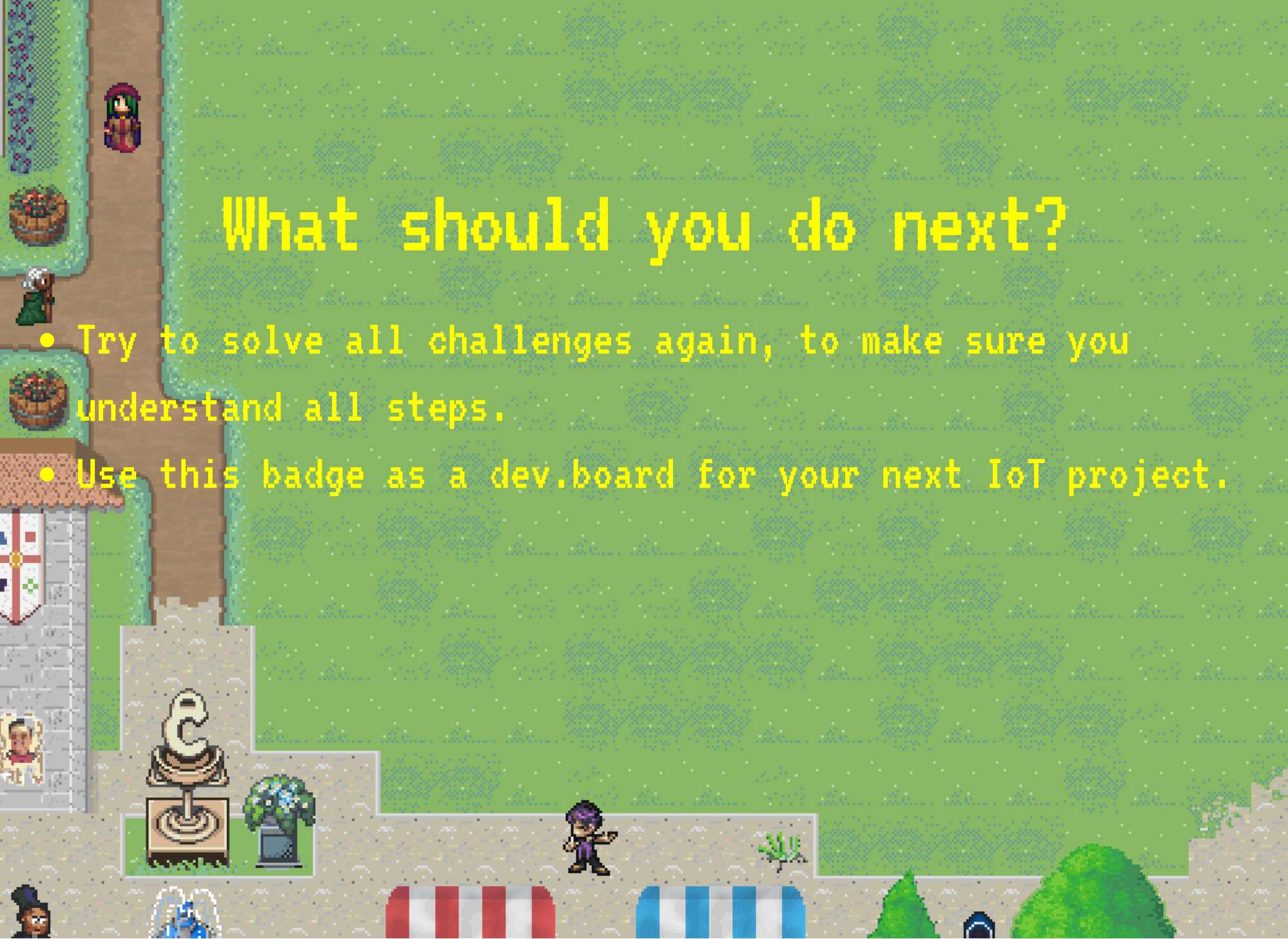
https://hideandsec.sh/books/ctf/page/northsec-2021-badge-writeup

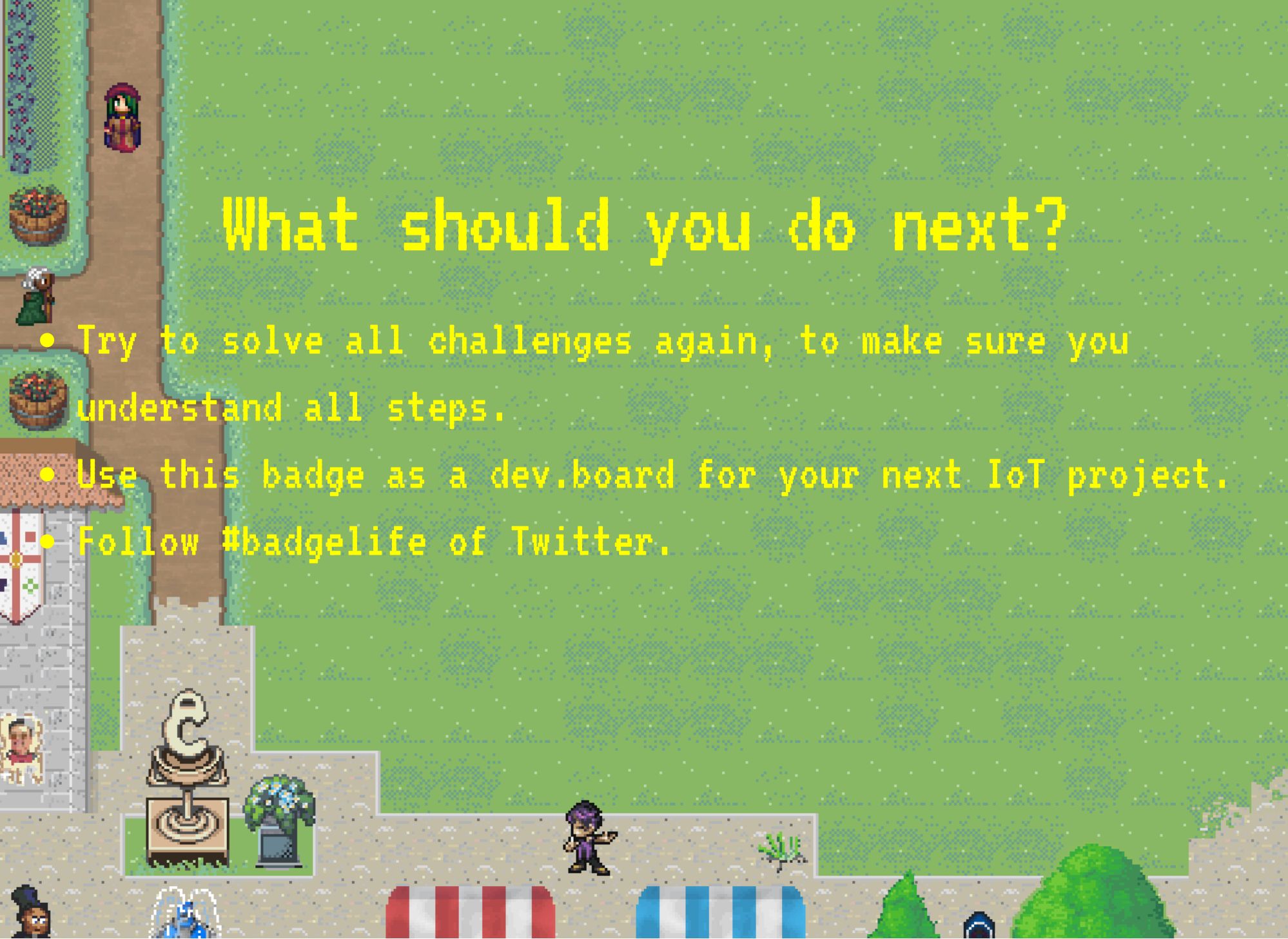


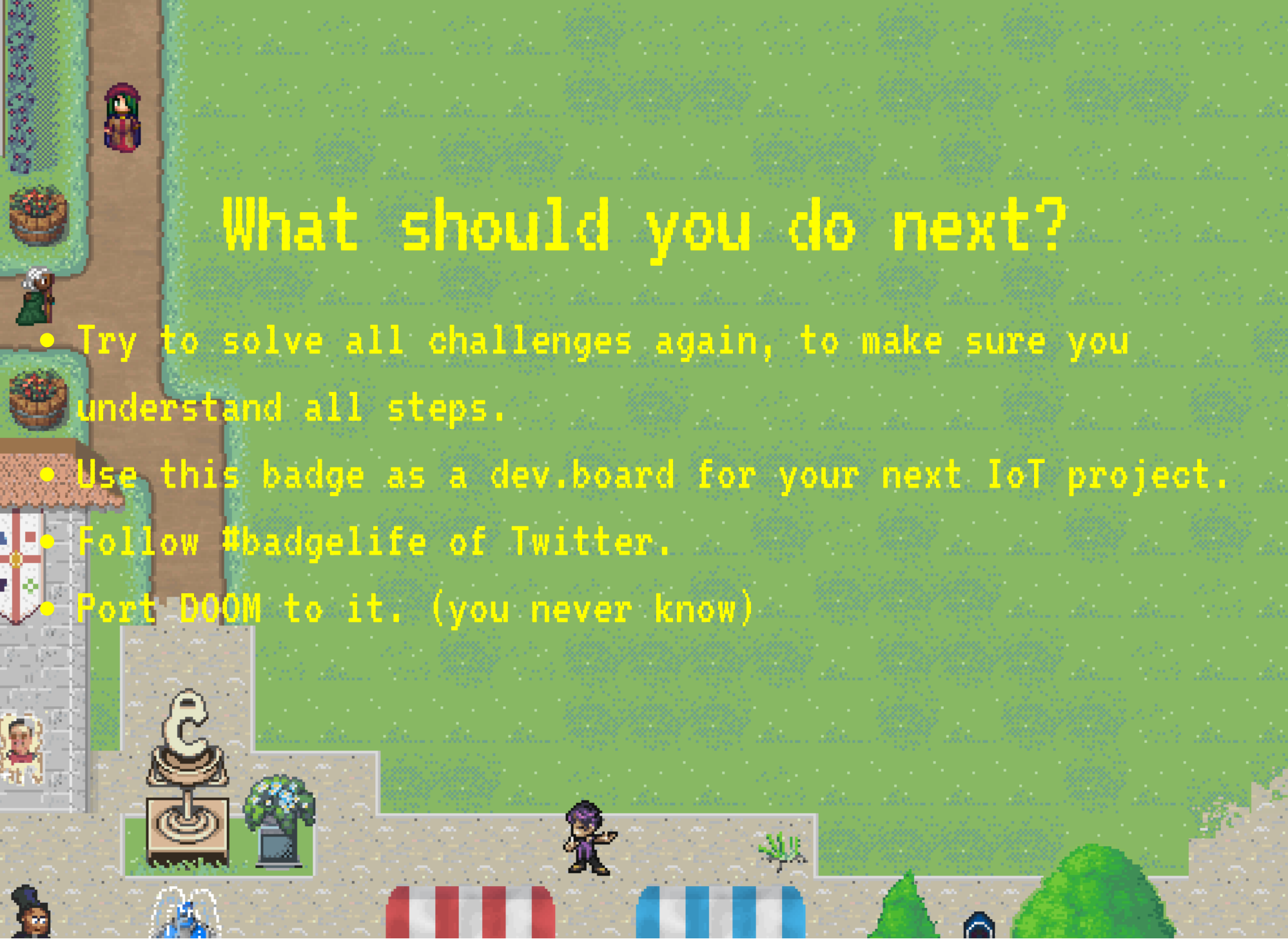


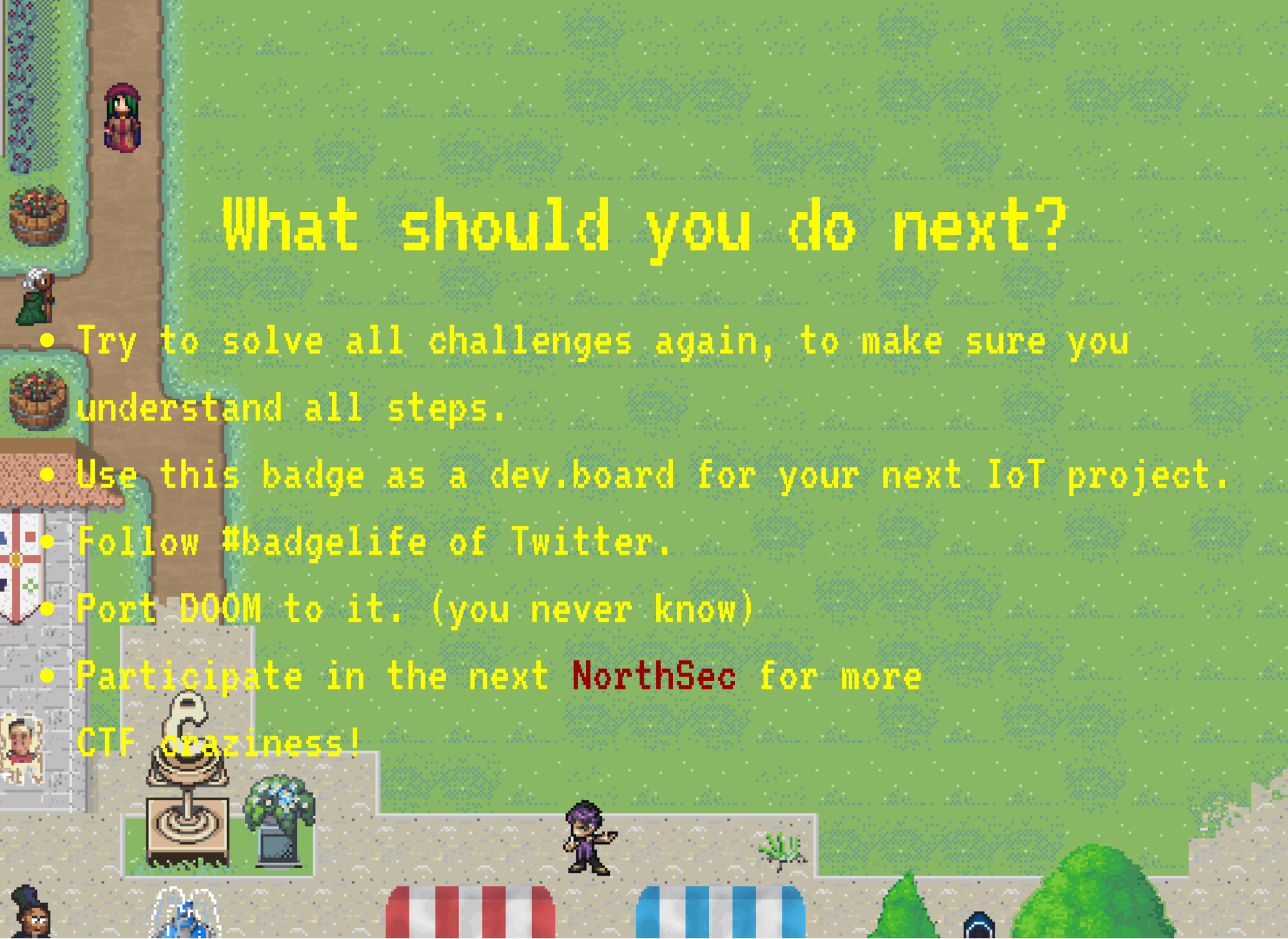

What should you do next?


What should you do next?
• Try to solve all challenges again, to make sure you understand all steps.


What should you do next?
• Try to solve all challenges again, to make sure you understand all steps.
• Use this badge as a dev.board for your next IoT project.


What should you do next?
• Try to solve all challenges again, to make sure you understand all steps.
• Use this badge as a dev.board for your next IoT project.
• Follow #badgelife of Twitter.


What should you do next?
• Try to solve all challenges again, to make sure you understand all steps.
• Use this badge as a dev.board for your next IoT project.
• Follow #badgelife of Twitter.
• Port DOOM to it. (you never know)


What should you do next?
• Try to solve all challenges again, to make sure you understand all steps.
• Use this badge as a dev.board for your next IoT project.
• Follow #badgelife of Twitter.
• Port DOOM to it. (you never know)
• Participate in the next NorthSec for more CTF craziness!


NSEC
Thank you for listening!
https://nsec.io/
https://montrehack.ca/