

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] 

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



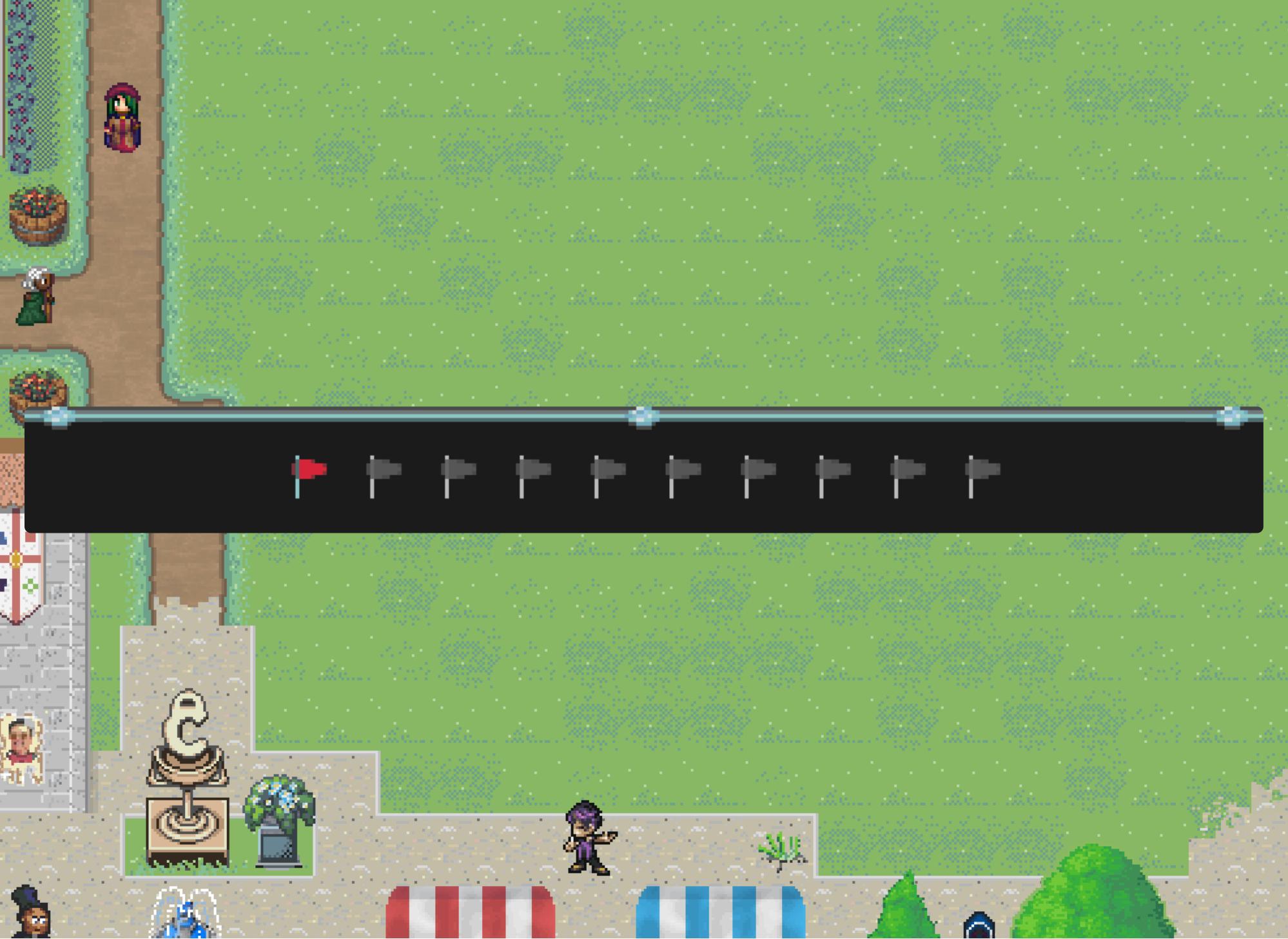
FLAGBOT BOT 05/05/2021

@ 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'?



* 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?

A screenshot of a medieval-themed game interface. On the left, there's a castle wall with various characters like a knight, a wizard, and a woman. In the center, there's a large letter 'C' on a pedestal and some potted plants. A small character is standing near the base of the 'C'. The right side of the screen shows a packet capture window from Wireshark. The title bar says "wlan.sa == AC:67:B2:78:51:F0 && tcp". The main pane shows a list of network packets. The first few packets are highlighted in blue. The details pane at the bottom shows the following information for the selected packet (Frame 1242):

- Frame 1242: 94 bytes on wire (752 bits), 94 bytes captured (752 bits)
- IEEE 802.11 QoS Data, Flags: .p.....T
- Logical-Link Control
- Internet Protocol Version 4, Src: 192.168.185.113, Dst: 198.51.100.42
- Transmission Control Protocol, Src Port: 52857, Dst Port: 4444, Seq: 0, Len: 0

The bottom status bar indicates: Packets: 3366 - Displayed: 29 (0.9%) Profile: Default.

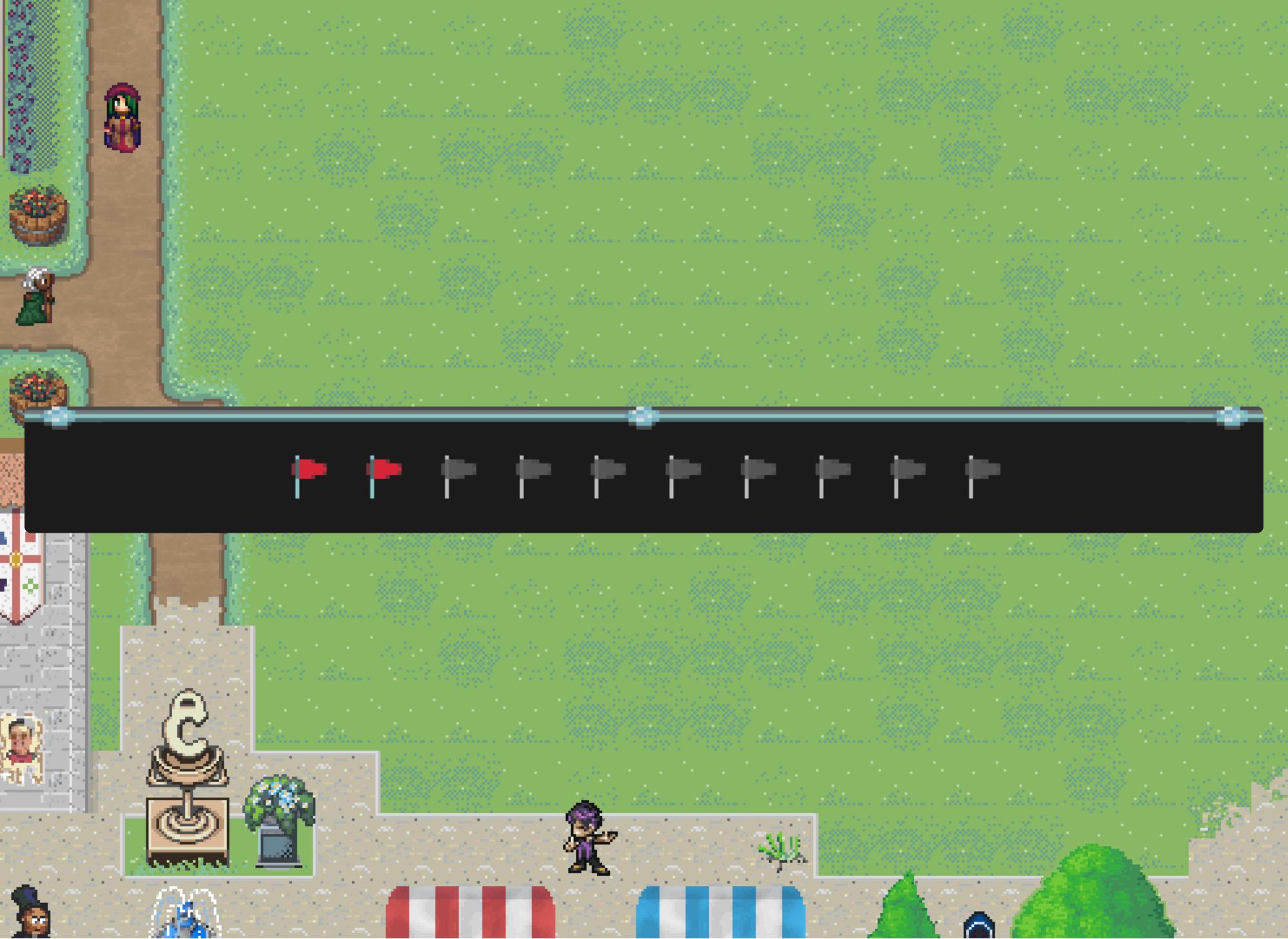
No.	Time	Source	S Port	Destination	D Port	Protocol	Length	Host
1242	2021-05-16 19:39:11.122248	192.168.185.113	52857	198.51.100.42	4444	TCP	94	
1282	2021-05-16 19:39:12.338318	192.168.185.113	52857	198.51.100.42	4444	TCP	94	
1409	2021-05-16 19:39:13.846931	192.168.185.113	52857	198.51.100.42	4444	TCP	94	
1574	2021-05-16 19:39:15.356917	192.168.185.113	52857	198.51.100.42	4444	TCP	94	
1621	2021-05-16 19:39:16.838240	192.168.185.113	52857	198.51.100.42	4444	TCP	94	
1956	2021-05-16 19:39:18.343421	192.168.185.113	52857	198.51.100.42	4444	TCP	94	
2212	2021-05-16 19:39:21.334542	192.168.185.113	52857	198.51.100.42	4444	TCP	94	
2238	2021-05-16 19:39:22.856966	192.168.185.113	52857	198.51.100.42	4444	TCP	94	
2329	2021-05-16 19:39:25.829692	192.168.185.113	52857	198.51.100.42	4444	TCP	94	
2360	2021-05-16 19:39:27.367381	192.168.185.113	52857	198.51.100.42	4444	TCP	94	
2363	2021-05-16 19:39:27.373367	192.168.185.113	52857	198.51.100.42	4444	TCP	94	
2399	2021-05-16 19:39:28.834399	192.168.185.113	52857	198.51.100.42	4444	TCP	94	
2796	2021-05-16 19:39:44.339193	192.168.168.185.113	52858	198.51.100.42	4444	TCP	94	
2820	2021-05-16 19:39:45.835148	192.168.185.113	52858	198.51.100.42	4444	TCP	94	
2834	2021-05-16 19:39:47.330522	192.168.185.113	52858	198.51.100.42	4444	TCP	94	
2864	2021-05-16 19:39:50.330353	192.168.185.113	52858	198.51.100.42	4444	TCP	94	
2878	2021-05-16 19:39:51.830551	192.168.185.113	52858	198.51.100.42	4444	TCP	94	
2917	2021-05-16 19:39:54.836337	192.168.185.113	52858	198.51.100.42	4444	TCP	94	
2920	2021-05-16 19:39:54.837762	192.168.185.113	52858	198.51.100.42	4444	TCP	94	
2956	2021-05-16 19:39:59.344885	192.168.185.113	52858	198.51.100.42	4444	TCP	94	
2985	2021-05-16 19:40:09.845293	192.168.185.113	52858	198.51.100.42	4444	TCP	94	

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

Are u listening?

```
(kali㉿kali)-[~]
$ nc -lvpn 4444
listening on [any] 4444 ...
connect to [198.51.100.42] from (UNKNOWN) [198.51.100.171] 50149
Are u listening? FLAG-sp0keTh3Hors
```

<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

1: user@localhost: ~ ▾

I (3608) phy_init: phy_version 4660, 0162888, Dec 23 2020

```
 `ydmmdy+.
 /mNmhymNNy`  
:NNN+` -dNNh  
/NNd-:/+++oNNN`  
/NNNNNNNNNNNNNm/  
`sNNNNNNNNNNNNNm/  
/NNNNNNNNNNNNNm/  
-ohy `:+oo+/ . -+yhdmmddy/ . .:/+++/:.  
:hNMMMH-hMMMMMMMNNo :dNNNNNNNNNNNNNd. -sdNNNNNNNNNs- /NNNNNNNNNNNNNm/  
sMMMMMNMMMMMMMMMMMd hNNNNmyssydNNNm/.yNNNNNNNNNNNNNy` `sNNNNNNNNNNNNNm/  
sMMMMMMMy/+yNMMMMMM+ NNNN/ `:o:-mNNNNmo-` `sNNNNNd:mNNNNdo:--:+yNMs`  
sMMMMMN. .NMMMMMy mNNNNho/-` mNNNNm- /NNNNNNNNNNNo ..  
sMMMMNh hMMMMMy mNNNNNNNNNs/+NNNNNdssssssssssNNNNNNNNNm/  
sMMMMNh hMMMMMy -odNNNNNNNNNNNNNNNNNNNNNNNNNNNNNm/  
sMMMMNh hMMMMMy `-/oydNNNNNNNNNm:::/:/:/:/:/NNNNNm.  
sMMMMNh hMMMMMy o: .mNNNNNNNNNy. -o- sNNNNNm+ .oy:  
sMMMMNh hMMMMMy mNmhs/+/yNNNNNdNNNNNhsosymNNNh: omNNNNNNdhyhmNNNh:  
sMMMMNh hMMMMMy mNNNNNNNNNNNNNy` -ymNNNNNNNNNNNm- .smNNNNNNNNNNNm.  
+hhhhho /shmNNNNNho- `:oyhdmmddy: ` `:oyhdmmddy: `
```

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.
nsec-badge>

Le Lac du Quack

the duck is literally segfaulting (edited)

quickquickquickzsh: segmentation fault

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

International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	• -
B	• • -
C	• - -
D	• -
E	•
F	• • - -
G	• - - -
H	• • -
I	• •
J	• - - -
K	• - -
L	• - - -
M	• - -
N	• - -
O	• - - -
P	• - - -
Q	• - - -
R	• - -
S	• • -
T	-

U	• - -
V	• - - -
W	• - - - -
X	• - - - -
Y	• - - - -
Z	• - - - -

1	• - - - -
2	• - - - -
3	• - - - -
4	• - - - -
5	• - - - -
6	• - - - -
7	• - - - -
8	• - - - -
9	• - - - -
0	• - - - -

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

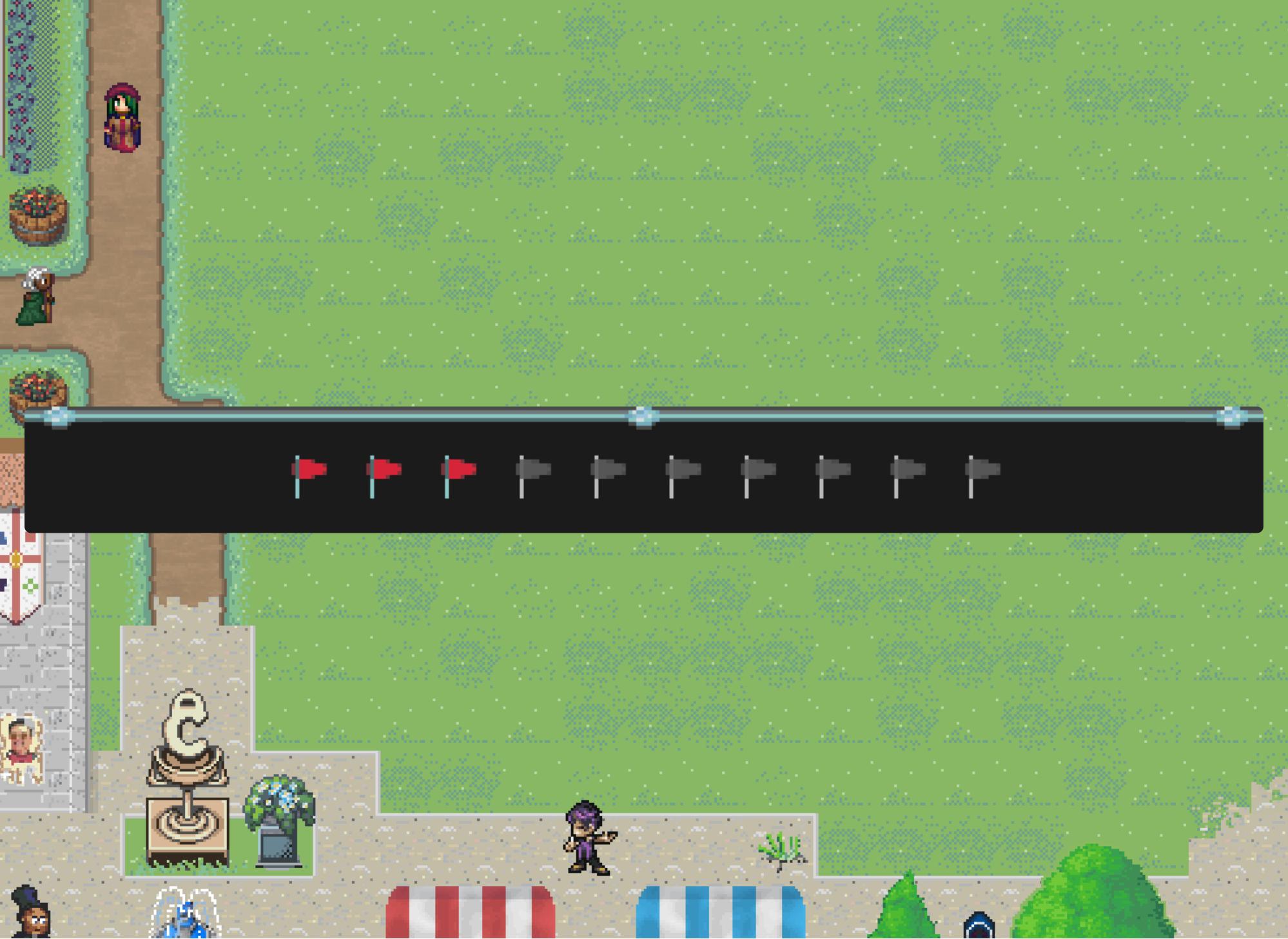
Le Lac du Quack

```
File System  
quackquackquickquack  
^[[26;1H^[[K^[[25;1H^[[K^[[24;1H^[[K^[[23;1H^  
^[[26;1H^[[K^[[25;1H^[[K^[[24;1H^[[K^[[23;1H^  
<1H^[[Kquackquickquackquick  
^[[26;1H^[[K^[[25;1H^[[K^[[24;1H^[[K^[[23;1H^  
^[[26;1H^[[K^[[25;1H^[[K^[[24;1H^[[K^[[23;1H^  
^[[26;1H^[[K^[[25;1H^[[K^[[24;1H^[[K^[[23;1H^  
^[[26;1H^[[K^[[25;1H^[[K^[[24;1H^[[K^[[23;1H^
```

<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 quackquickquack " "quackquackquack quackquickquackquick quackquickquack quack "  
"quackquackquack quackquackquick quick quack quackquickquackquick "  
"quackquackquick quick quickquack quackquickquick quickquickquack "  
"quackquackquick quick quack quickquickquackquick quickquick " "quackquickquickquick quickquickquackquickquackquick quickquack  
"quackquackquick quackquickquickquick quackquick quickquickquackquack "  
"quackquackquickquick quackquickquack quickquack quickquackquack " "quackquack quickquickquick quackquickquackquick quickquick";
```



Hacking session!



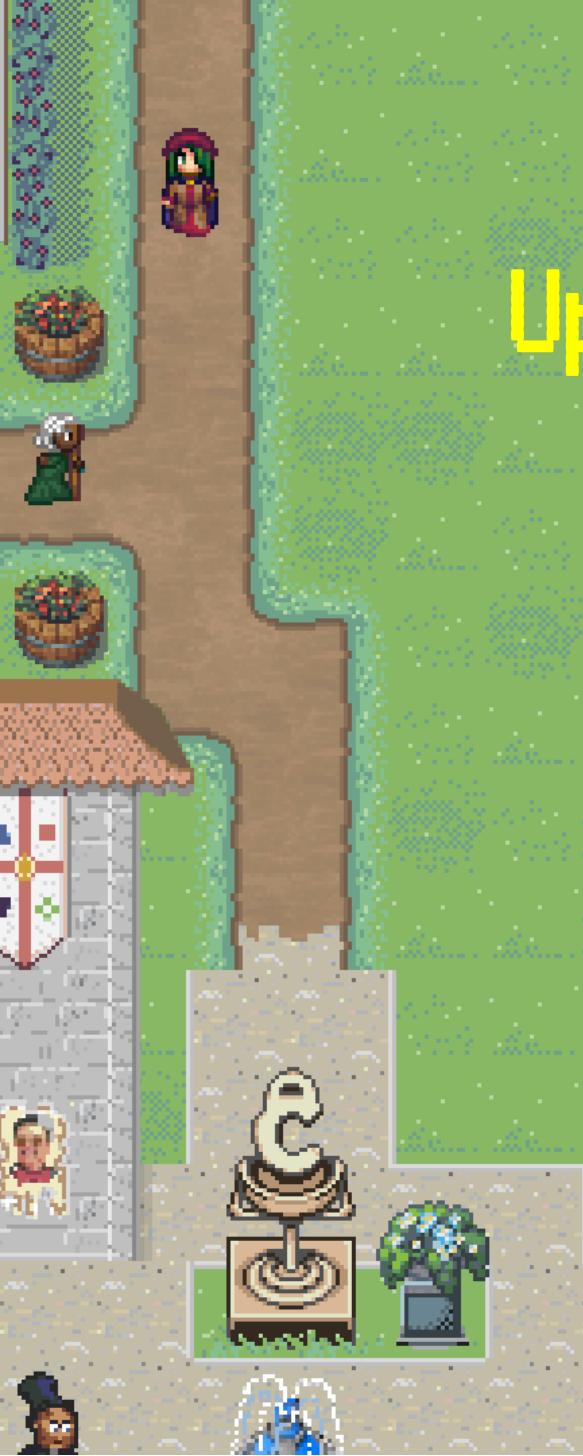
Up-down-left-right



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 . . . . . . . . . . . .
```


How do I get past the gate?

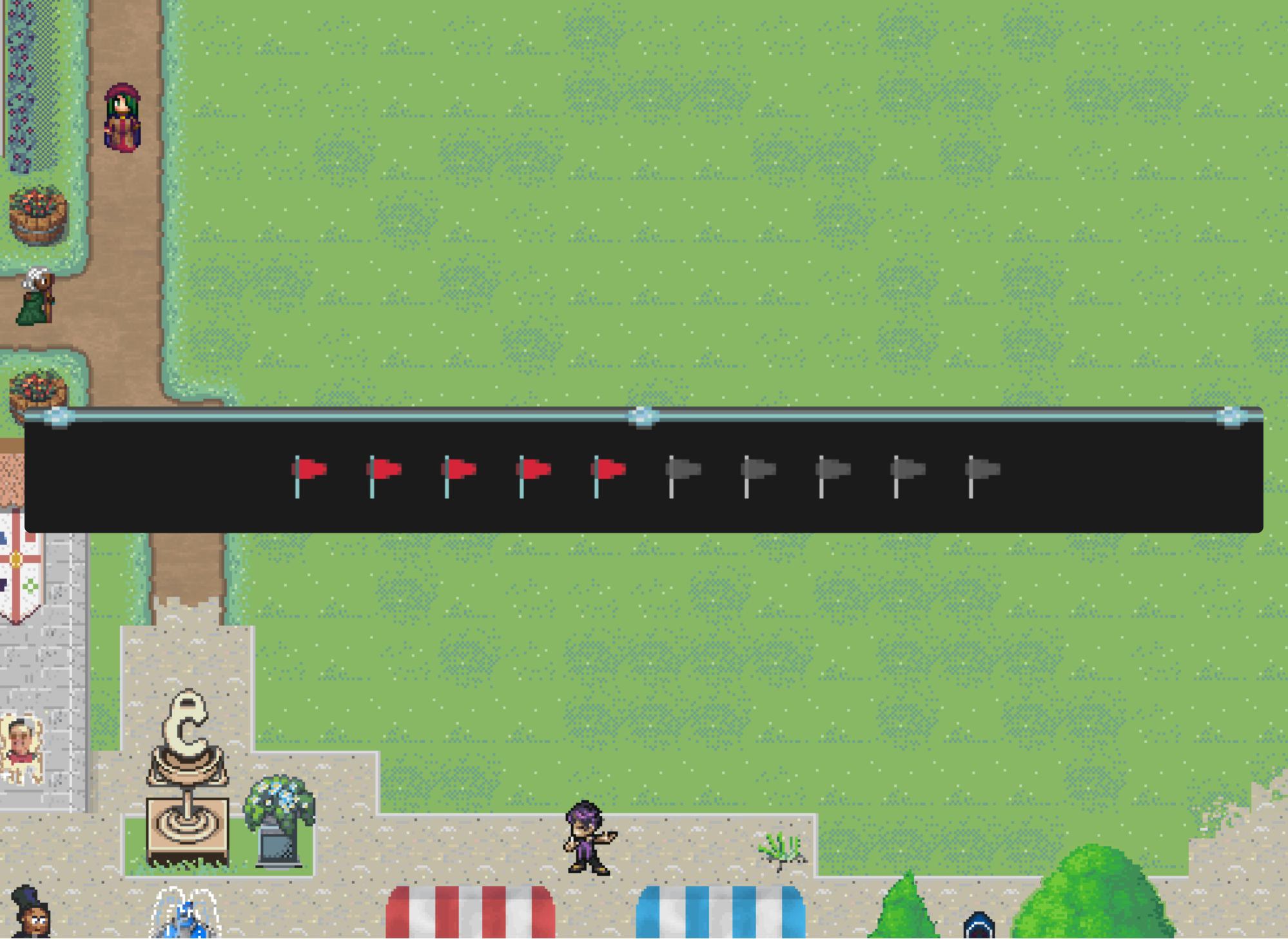


How do I get past the gate?



How do I get past the gate?





Hacking session!



Are you really CYBER!?



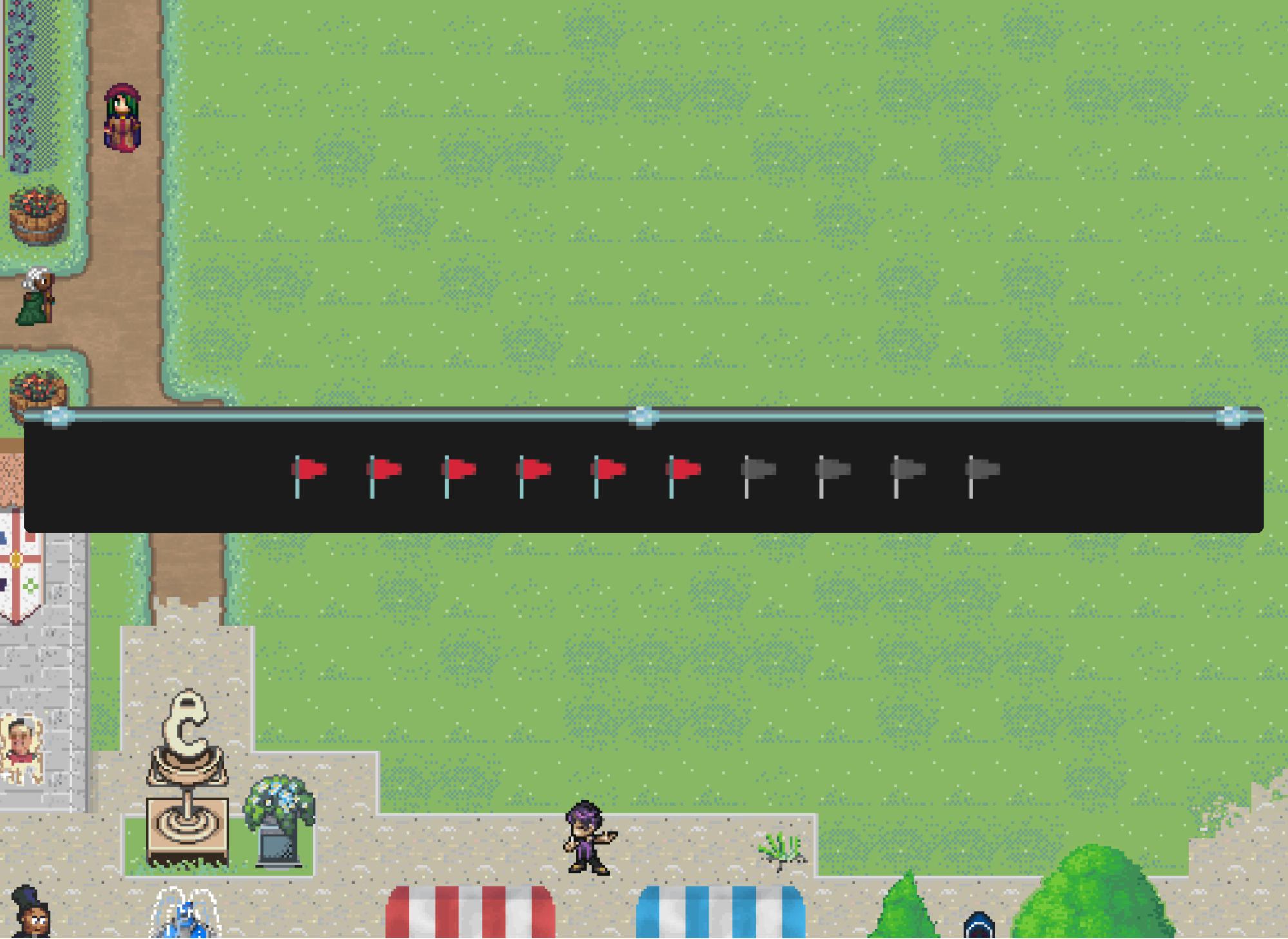
Are you really CYBER!?



Are you really CYBER!?

```
PTR_s_Are_you_really_3ffb11cc          XREF[1]: 400d0c24(*)  
3ffb11cc f4 70 40 3f      addr    s_Are_you_really_3f4070f4  
3ffb11d0 04 71 40 3f      addr    s_CYBER!_?_PROVE_IT_3f407104  
3ffb11d4 4c a5 40 3f      addr    s_3f40a500+76  
3ffb11d8 4c a5 40 3f      addr    s_3f40a500+76  
3ffb11dc 4c a5 40 3f      addr    s_3f40a500+76  
3ffb11e0 4c a5 40 3f      addr    s_3f40a500+76  
3ffb11e4 4c a5 40 3f      addr    s_3f40a500+76  
3ffb11e8 18 71 40 3f      addr    DAT_3f407118  
3ffb11ec 44 71 40 3f      addr    DAT_3f407144  
3ffb11f0 90 5c 40 3f      addr    DAT_3f405c90  
  
PTR_s_Go_sit_in_a_3ffb11f4  
3ffb11f4 70 72 40 3f      addr    s_Go_sit_in_a_3f4072  
3ffb11f8 80 72 40 3f      addr    s_detonation_box._3f  
3ffb11fc 90 5c 40 3f      addr    DAT_3f405c90  
  
PTR_s_I_am_watching_that_3ffb1200  
3ffb1200 90 72 40 3f      addr    s_I_am_watching_that  
3ffb1204 a4 72 40 3f      addr    s_guy_over_there._I.  
3ffb1208 b8 72 40 3f      addr    s_think_he_has_a_for  
3ffb120c d0 72 40 3f      addr    s_bomb._3f4072d0  
3ffb1210 90 5c 40 3f      addr    DAT_3f405c90  
  
PTR_s_I_hear_a_lot_that_3ffb1214  
3ffb1214 d8 72 40 3f      addr    s_I_hear_a_lot_that  
3ffb1218 ec 72 40 3f      addr    s_they._bring_their.  
3ffb121c 00 73 40 3f      addr    s_own_land." How wil  
  
XREF[1]: 400d0c24(*)  
= "Are you really\n"  
= "CYBER! ? PROVE IT\n"  
= "\n"  
= "\n"  
= "\n"  
= "\n"  
= "\n"  
= 46h F  
= F1h  
  
3f4070e8 3c ec 00 40      addr    DAT_400dec3c  
3f4070ec 4c ec 0d 40      addr    DAT_400dec4c  
3f4070f0 a8 b6 0d 40      addr    DAT_400db6a8  
  
s_Are_you_really_3f4070f4  
3f4070f4 41 72 65      ds     "Are you really\n"  
20 79 6f  
75 20 72 ...  
  
s_CYBER! ?_PROVE_IT_3f407104  
3f407104 43 59 42      ds     "CYBER! ? PROVE IT\n"  
45 52 21  
3f 20 50 ...  
3f407116 00      ??      00h  
3f407117 00      ??      00h  
  
DAT_3f407118  
3f407118 46      ??      46h F  
3f407119 ff      ??      FFh  
3f40711a 4c      ??      4Ch L  
3f40711b ff      ??      FFh  
3f40711c 41      ??      41h A  
3f40711d ff      ??      FFh  
3f40711e 47      ??      47h G  
3f40711f ff      ??      FFh  
3f407120 2d      ??      20h -  
3f407121 ff      ??      FFh  
3f407122 4b      ??      48h K  
3f407123 ff      ??      FFh  
3f407124 4c      ??      4Ch L  
3f407125 ff      ??      FFh  
3f407126 4a      ??      4Ah I
```

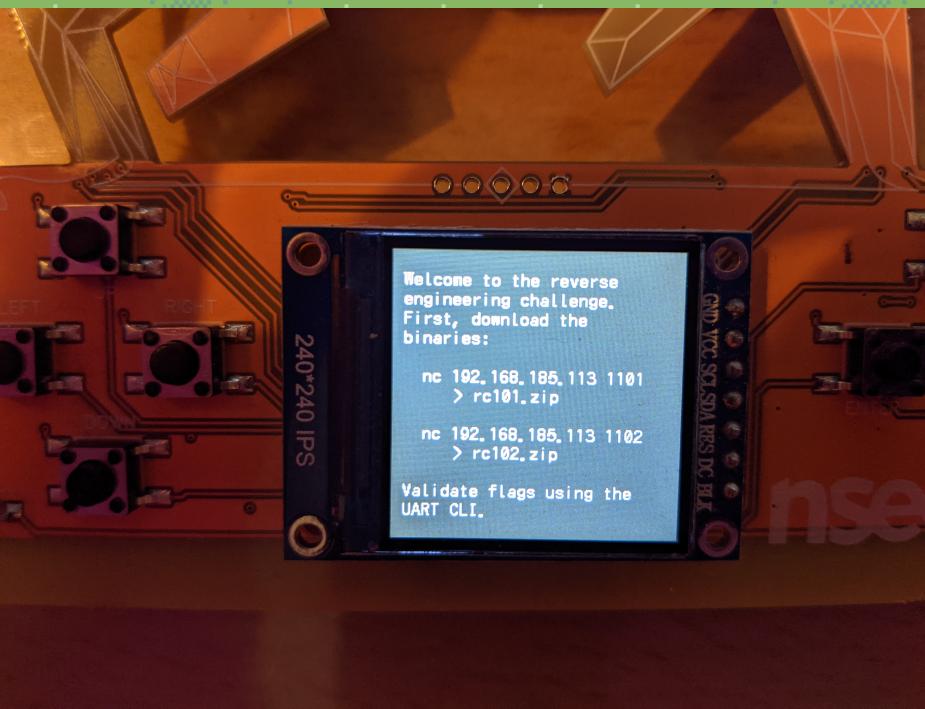
<https://blog.quantumlyconfused.com/ctf/2021/05/30/nsec2021-badgelife-pt1/>



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: ???

re101.elf

Cutter – re101.elf

File 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  
call18 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  
movi.n a11, 0
```

Dashboard Strings Imports Search Disassembly Graph (entry0) Hexdump

Cutter – re101.elf

File Windows Debug Help

Type flag name or address here

Functions

c_init
c_release
c_restorecs
c_savecs
exception
t4
t5
t2
t3

_exc
ivisor_init
exc
exit

pt_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

o_pointer_default

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
call18 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
call18 verify ; sym.verify
beqz.n a10, 0x400d2390

```

d0394

; sym.printf ; int printf(const char *format)

132r a10, 0x400d0398
call18 puts ; sym.puts

retw.n

Type flag name or address here

Functions

c_init
c_release
c_restorecs
c_savecs
exception
t4
t5
t2
t3

_exc
ivisor_init
exc
exit

opt_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

o_pointer_default

Graph (sym.verify)

```
; 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
```


Windows Debug Help

Type flag

View

Start debug F9

Start emulation

Attach to process

Connect to a remote debugger

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

Graph (sym.verify)

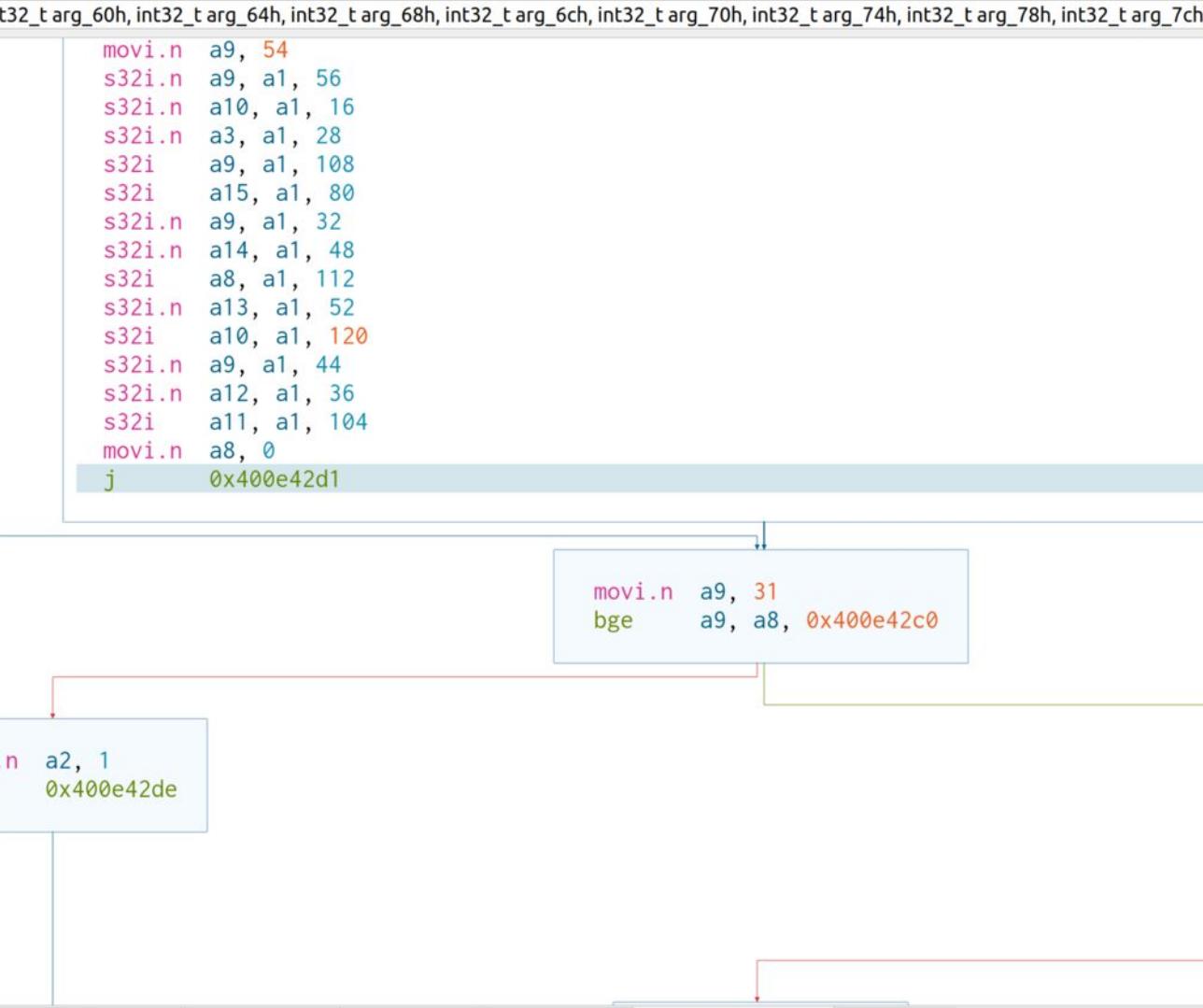
Dashboard Strings Imports Search Disassembly Graph (sym.verify) Hexdump

Cutter – re101.elf

Windows Debug Help

Type flag name or address here

Graph (sym.verify)



		Stack	
	Offset	Value	Reference
	0x00178000	0x00000031	-> (.debug_frame) ascii
	0x00178004	0x00000031	-> (.debug_frame) ascii
	0x00178008	0x00000061	-> (.debug_frame) ascii
	0x0017800c	0x00000039	-> @a13 (.debug_frame) ascii
	0x00178010	0x00000032	-> @a10 (.debug_frame) ascii
	0x00178014	0x00000032	-> @a10 (.debug_frame) ascii
	0x00178018	0x00000031	-> (.debug_frame) ascii
	0x0017801c	0x00000038	-> @a3 (.debug_frame) ascii
	0x00178020	0x00000036	-> @a9 (.debug_frame) ascii
	0x00178024	0x00000063	-> @a12 (.debug_frame) ascii
	0x00178028	0x00000034	-> @a14 (.debug_frame) ascii
	0x0017802c	0x00000036	-> @a9 (.debug_frame) ascii
	0x00178030	0x00000034	-> @a14 (.debug_frame) ascii
	0x00178034	0x00000039	-> @a13 (.debug_frame) ascii
	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) ascii
	0x0017804c	0x00000031	-> (.debug_frame) ascii
	0x00178050	0x00000037	-> @a15 (.debug_frame) ascii
	0x00178054	0x00000063	-> @a12 (.debug_frame) ascii
	0x00178058	0x00000031	-> (.debug_frame) ascii
	0x0017805c	0x00000032	-> @a10 (.debug_frame) ascii
	0x00178060	0x00000038	-> @a3 (.debug_frame) ascii
	0x00178064	0x00000062	-> (.debug_frame) ascii
	0x00178068	0x00000033	-> @a11 (.debug_frame) ascii
	0x0017806c	0x00000026	-> @a9 (.debug_frame) ascii

Stack Backtrace Threads

Registers

a0	0000000	a9	0000036
a1	0178000	a10	0000032
a2	0000000	a11	0000033

Strings Search Memory Map Breakpoints Disassembly Graph (sym.verify) Hexdump

Cutter – re101.elf

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

Offset	Value	Stack	Reference
0x00178000	0x00000021	Stack position	frame) ascii
0x00178004	Show in		Decompiler
0x00178008	Copy address	Ctrl+Shift+C	Disassembly
0x0017800c	Show X-refs	X	Graph (sym.verify)
0x00178010	Add comment	;	
0x00178014	Edit stack value...		
0x00178018	0x00000031	> (.debug_f	New disassembly
0x0017801c	0x00000038	> @a3 (.deb	New graph
0x00178020	0x00000036	> @a9 (.debug_f	New hexdump
0x00178024	0x00000063	> @a12 (.debug_f	
0x00178028	0x00000034	> @a14 (.debug_f	
0x0017802c	0x00000036	> @a9 (.debug_f	
0x00178030	0x00000034	> @a14 (.debug_f	
0x00178034	0x00000039	> @a13 (.debug_f	
0x00178038	0x00000036	> @a9 (.debug_f	
0x0017803c	0x00000065	> (.debug_f	
0x00178040	0x00000062	> (.debug_f	
0x00178044	0x00000031	> (.debug_f	
0x00178048	0x00000033	> @a11 (.debug_f	
0x0017804c	0x00000031	> (.debug_f	
0x00178050	0x00000037	> @a15 (.debug_f	
0x00178054	0x00000063	> @a12 (.debug_f	
0x00178058	0x00000031	> (.debug_f	
0x0017805c	0x00000032	> @a10 (.debug_f	
0x00178060	0x00000038	> @a3 (.debug_f	
0x00178064	0x00000062	> (.debug_f	
0x00178068	0x00000033	> @a11 (.debug_f	
0x0017806c	0x00000026	> @a9 (.debug_f	

Stack Backtrace Threads

Registers

a0	00000000	a9	0000036
a1	0178000	a10	0000032
a2	00000000	a11	0000033

Strings Search Memory Map Breakpoints Disassembly Graph (sym.verify) Hexdump

Cutter – re101.elf

Windows Debug Help

Type flag name or address here

Hexdump

	0	1	2	3	4	5	6	7	01234567
0x000000000000178000	b1	00	00	00	31	00	00	00	1.....
0x000000000000178008	61	00	00	00	39	00	00	00	a.....
0x000000000000178010	32	00	00	00	32	00	00	00	2.....
0x000000000000178018	31	00	00	00	38	00	00	00	1.....
0x000000000000178020	36	00	00	00	63	00	00	00	6....c..
0x000000000000178028	34	00	00	00	36	00	00	00	4....6..
0x000000000000178030	34	00	00	00	39	00	00	00	4....9..
0x000000000000178038	36	00	00	00	65	00	00	00	6....e..
0x000000000000178040	62	00	00	00	31	00	00	00	b....1..
0x000000000000178048	33	00	00	00	31	00	00	00	3....1..
0x000000000000178050	37	00	00	00	63	00	00	00	7....c..
0x000000000000178058	31	00	00	00	32	00	00	00	1....2..
0x000000000000178060	38	00	00	00	62	00	00	00	8....b..
0x000000000000178068	33	00	00	00	36	00	00	00	3....6..
0x000000000000178070	31	00	00	00	37	00	00	00	1....7..
0x000000000000178078	32	00	00	00	30	00	00	00	2....0..
0x000000000000178080	00	00	00	00	00	00	00	00
0x000000000000178088	00	00	00	00	00	00	00	00
0x000000000000178090	00	00	00	00	00	00	00	00
0x000000000000178098	00	00	00	00	00	00	00	00
0x0000000000001780a0	00	00	00	00	00	00	00	00
0x0000000000001780a8	00	00	00	00	00	00	00	00
0x0000000000001780b0	00	00	00	00	00	00	00	00
0x0000000000001780b8	00	00	00	00	00	00	00	00
0x0000000000001780c0	00	00	00	00	00	00	00	00
0x0000000000001780c8	00	00	00	00	00	00	00	00
0x0000000000001780d0	00	00	00	00	00	00	00	00

Parsing **Information**

MDS: 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	Value
0x0001780000000000	0x00000000
0x0001780000000008	0x00000000
0x0001780000000010	0x00000000
0x0001780000000018	0x00000000
0x0001780000000020	0x00000000
0x0001780000000028	0x00000000
0x0001780000000030	0x00000000
0x0001780000000038	0x00000000
0x0001780000000040	0x00000000
0x0001780000000048	0x00000000
0x0001780000000050	0x00000000
0x0001780000000058	0x00000000
0x0001780000000060	0x00000000
0x0001780000000068	0x00000000
0x0001780000000070	0x00000000
0x0001780000000078	0x00000000
0x0001780000000080	0x00000000
0x0001780000000088	0x00000000
0x0001780000000090	0x00000000
0x0001780000000098	0x00000000
0x00017800000000a0	0x00000000
0x00017800000000a8	0x00000000
0x00017800000000b0	0x00000000
0x00017800000000b8	0x00000000
0x00017800000000c0	0x00000000
0x00017800000000c8	0x00000000
0x00017800000000d0	0x00000000

Registers

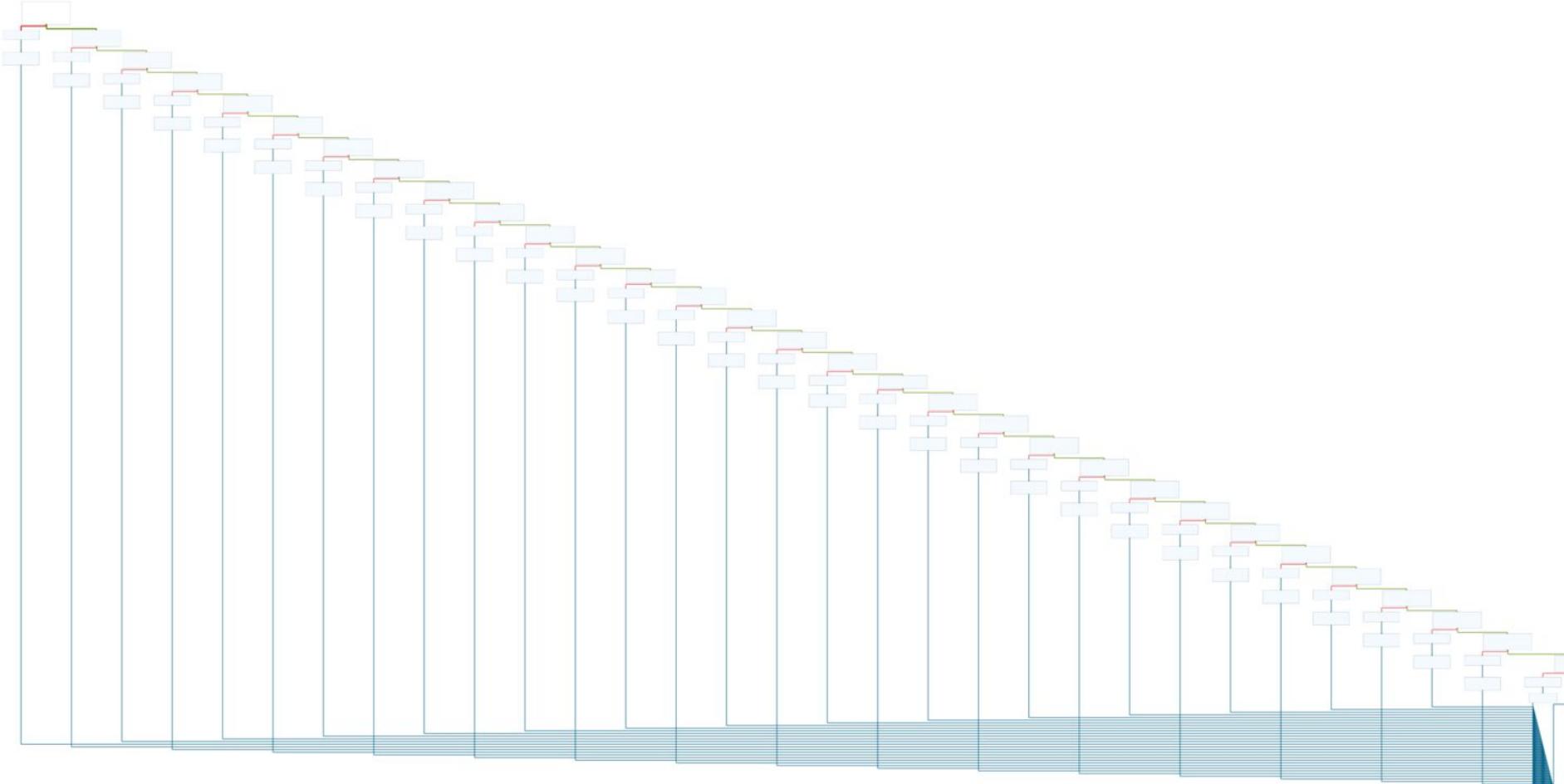
a0	00000000	a9	0
a1	01780000	a10	0
a2	00000000	a11	0

re102.elf

Type flag name or address here

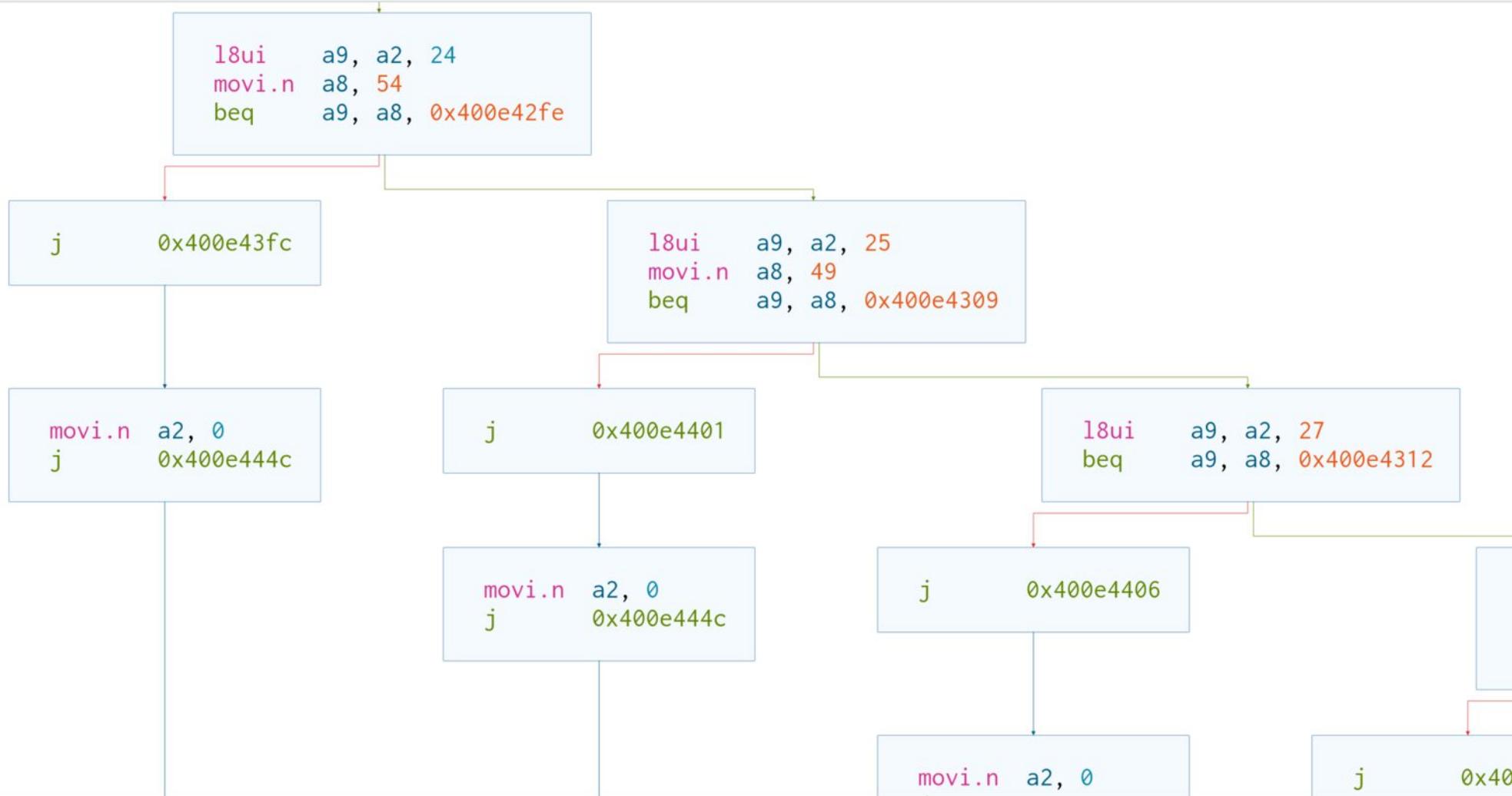
Graph (sym.verify)

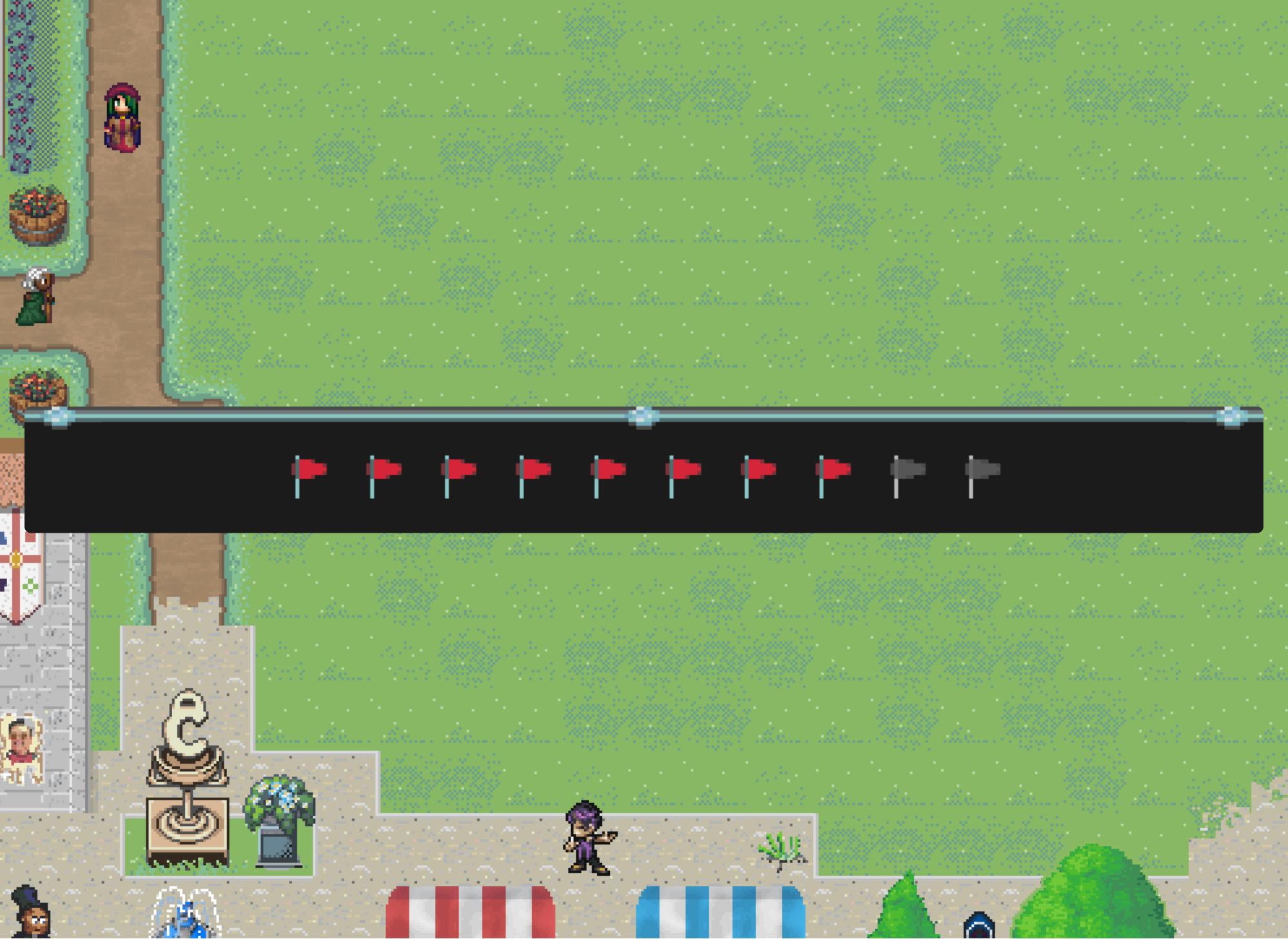
n.verify();



Graph (sym.verify)

n.verify();





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-separated list of 5 I/O numbers to use for SPI flash (QI K Q D1 D2 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

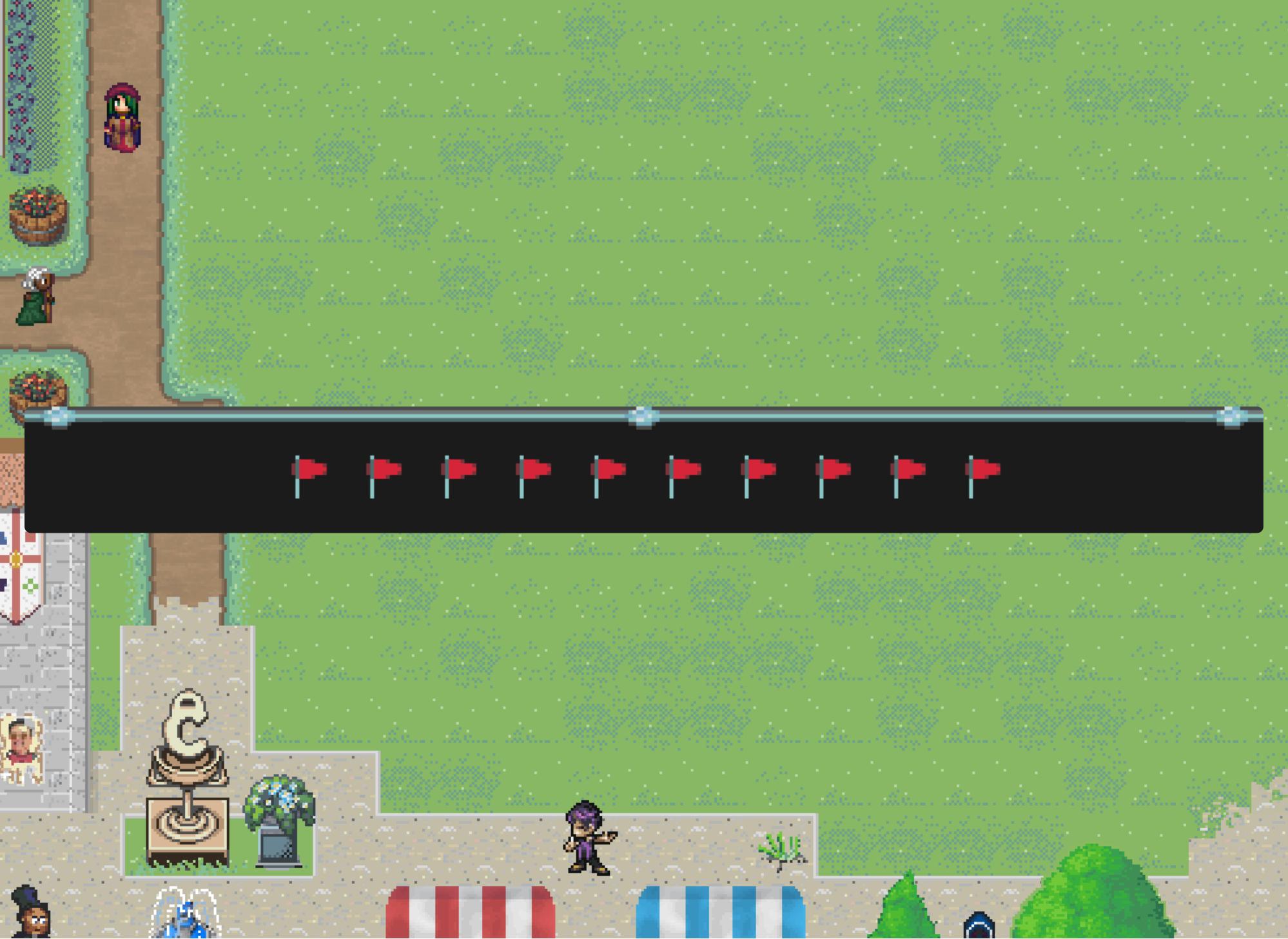
```
1 $ esp32_image_parser.py show_partitions nsec2021.bin
2
3 reading partition table...
4 entry 0:
5   label      : nvs
6   offset     : 0x9000
7   length    : 24576
8   type      : 1 [DATA]
9   sub type   : 2 [WIFI]
10
11 ...
```

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!



Thank you for listening!

<https://nsec.io/>

<https://montrehack.ca/>