SPQR

SPQR

The
Roman
Xpl0it

How to reverse a firmware WITHOUT PAIN

by cristian-richie and chq-matteo

About us

Cristian Assaiante
[@cristianrichi3, cristianrichie.github.io]

22 years old, Msc. student of Engineering in Computer Science.

Interested in reverse engineering
and doing stuff with hypervisors (maybe stuff for another talk!!)

Capturing flags with TRX and with mhackeroni.

Qian Matteo Chen
[@chqmatteo]

Capturing flags with TRX and with mhackeroni.

CSAW ESC 19

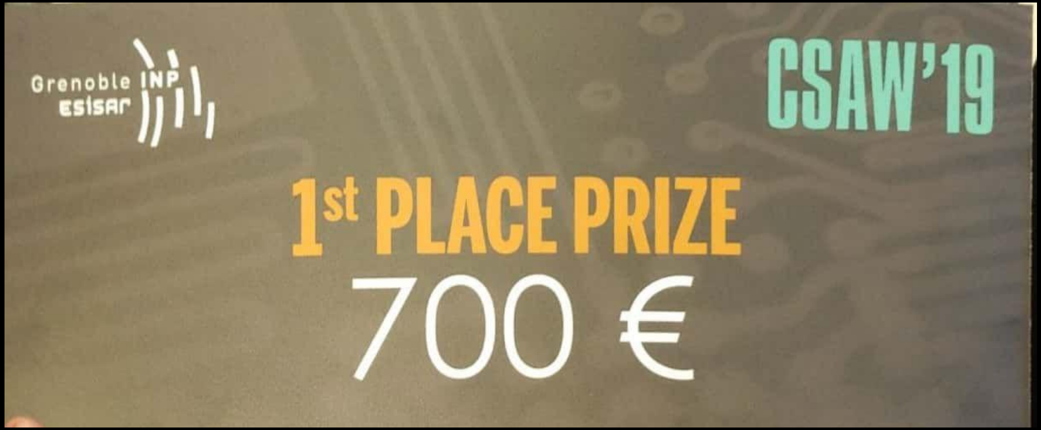
On November 8 we played the CSAW Embedded Security Challenge Finals.

We were given a board with an RFID r/w.

We had to program the RFID card in order to “open the 18 doors”.

!!! SPOILER !!!

THANK YOU
TO OUR SPONSORS
ANK YOU
R SPONSORS
YES WE H4CK
THAL
700 €
BUG


Grenoble INP
Esisar
CSAW'19
1st PLACE PRIZE
700 €

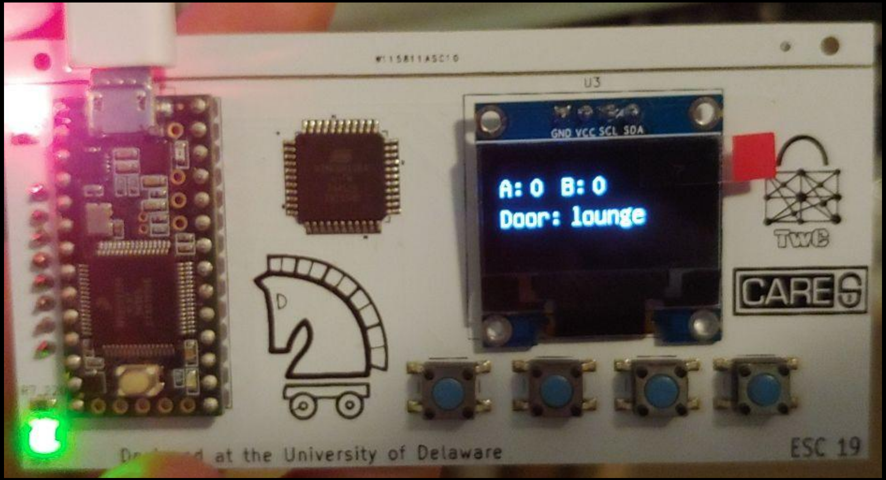
The lost art of Static Analysis

The binaries and the board

- An AVR binary
- An ARM binary

The AVR binary contained the RFID Authentication logic and the communication between the board and the checker.

The ARM binary contained the challenges checker logic.



Ghidra and R2

IDA WAS FORBIDDEN!!

We used the NSA Tool GHIDRA for the ARM binary and radare2 with the r2ghidra-dec plugin for the AVR binary.

The AVR binary was useful only for one challenge, so in this talk we are going to see how the ARM binary works.

Calling Conventions

We knew from reversing that every challenge function was like this: challenge_n(packet p)

The goal is to find what to write in the RFID but also the offset where we have to start writing!

Understanding the ghidra decompiler output can be tricky without knowing the ARM calling conventions!

Calling Conventions

Example:

```
void foo(int arg1, int arg2, int arg3, int arg4,
    int arg5, ...);
```

arg1 ➜ r0

arg2 ➜ r1

arg3 ➜ r2

arg4 ➜ r3

arg5 ➜ stack

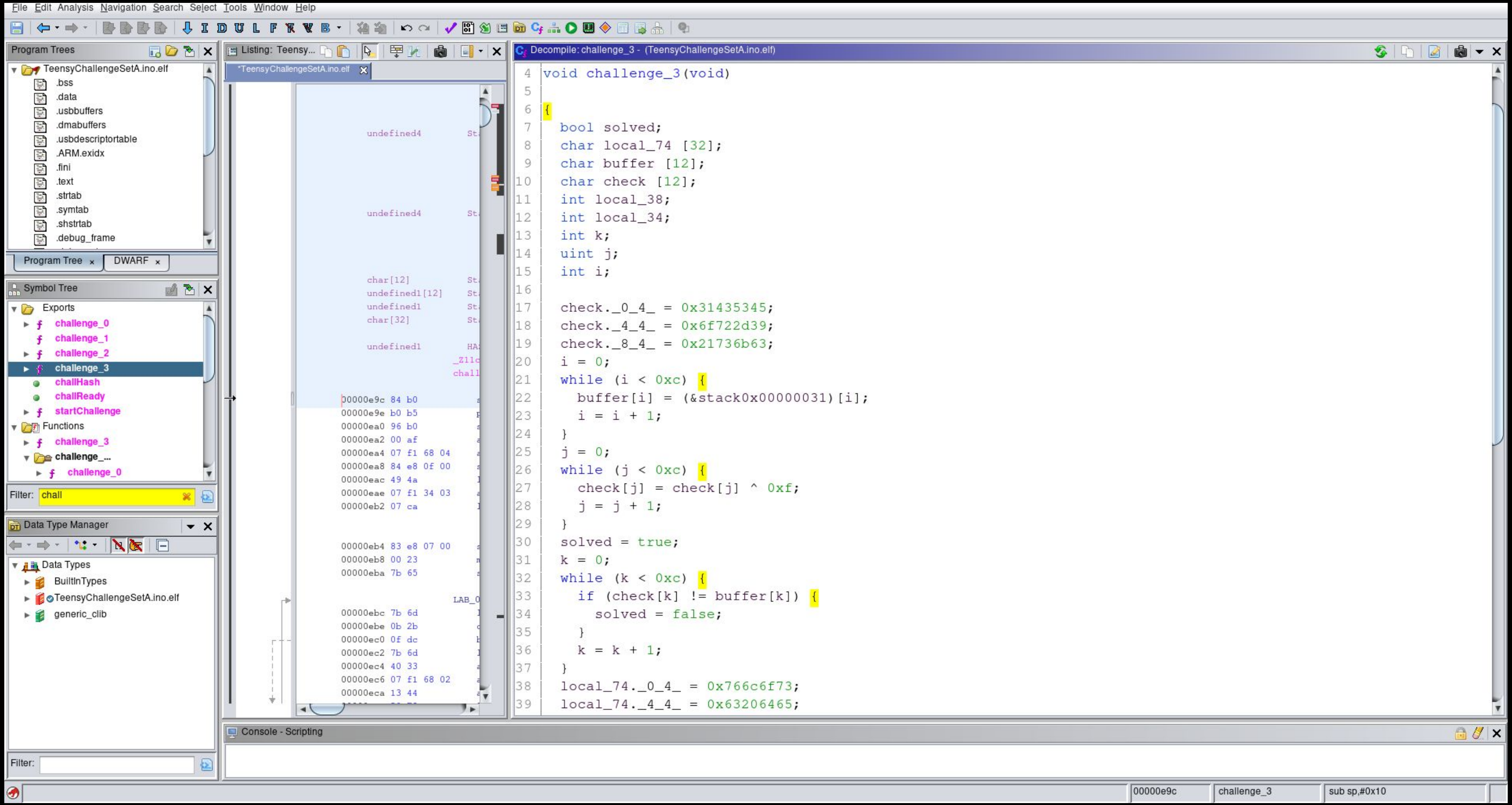
and so on...

Calling Conventions

```
struct packet {
        char comm;           ➜ r0[0]
        char RFID[1024];     ➜ r0[1:], r1, r2, r3 + stack
        char keys[48];       ➜ stack
        char buttons;        ➜ stack
        char challengeNum;   ➜ stack
};

sizeof(packet) = 1075 !!!
```

DEMO TIME
(1st)

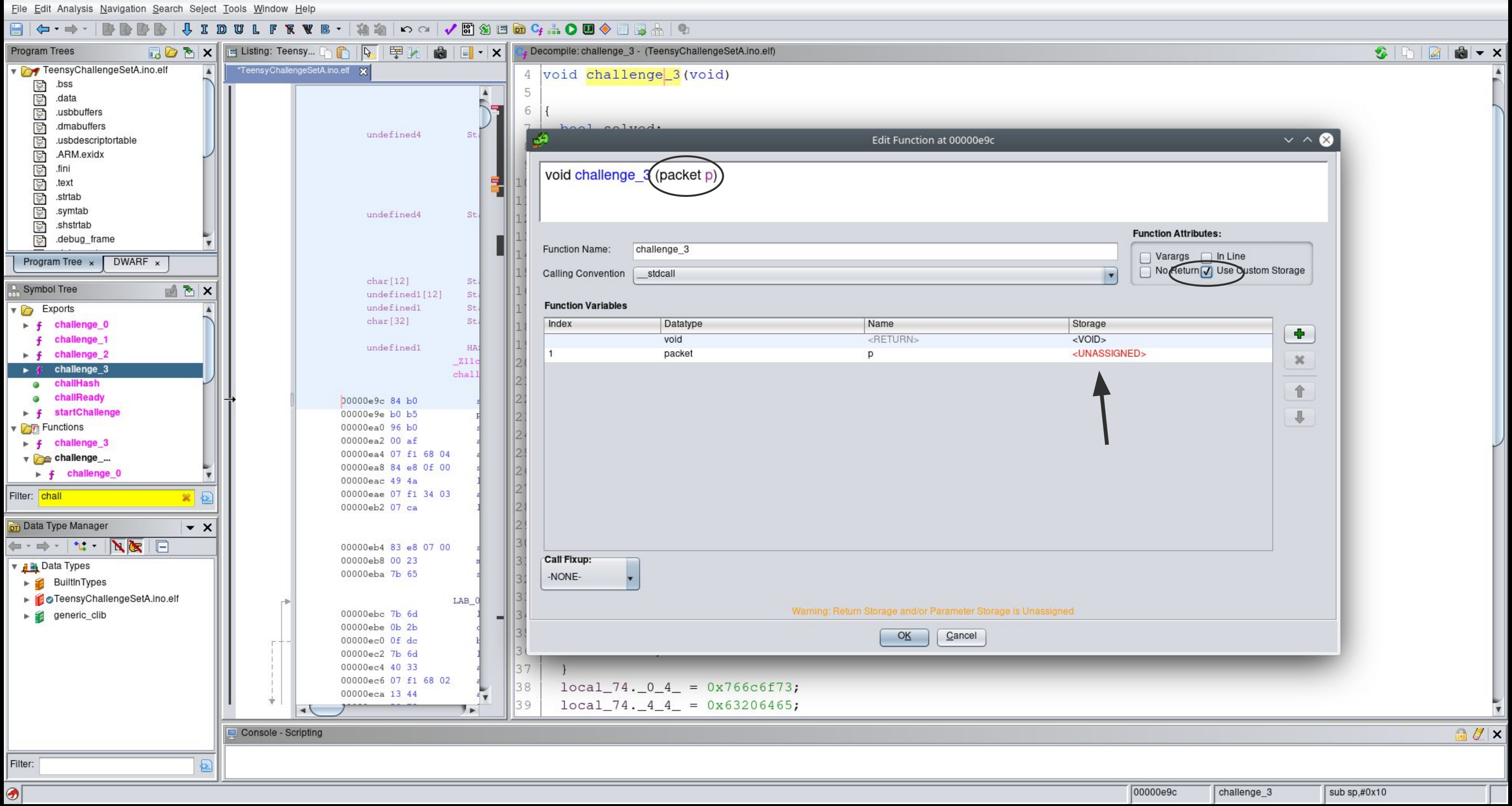
Calling Conventions (DEMO)



```
                    undefined4         St

                    undefined4         St

                    char[12]           St
                    undefined1[12]     St
                    undefined1         St
                    char[32]           St

                    undefined1         HA
                                     _Z11c
                                     chall
          00000e9c 84 b0
          00000e9e b0 b5
          00000ea0 96 b0
          00000ea2 00 af
          00000ea4 07 f1 68 04
          00000ea8 84 e8 0f 00
          00000eac 49 4a
          00000eae 07 f1 34 03
          00000eb2 07 ca

          00000eb4 83 e8 07 00
          00000eb8 00 23
          00000eba 7b 65

                                     LAB_0
          00000ebc 7b 6d
          00000ebe 0b 2b
          00000ec0 0f dc
          00000ec2 7b 6d
          00000ec4 40 33
          00000ec6 07 f1 68 02
          00000eca 13 44
```

```c
void challenge_3(void)

{
  bool solved;
  char local_74 [32];
  char buffer [12];
  char check [12];
  int local_38;
  int local_34;
  int k;
  uint j;
  int i;

  check._0_4_ = 0x31435345;
  check._4_4_ = 0x6f722d39;
  check._8_4_ = 0x21736b63;
  i = 0;
  while (i < 0xc) {
    buffer[i] = (&stack0x00000031)[i];
    i = i + 1;
  }
  j = 0;
  while (j < 0xc) {
    check[j] = check[j] ^ 0xf;
    j = j + 1;
  }
  solved = true;
  k = 0;
  while (k < 0xc) {
    if (check[k] != buffer[k]) {
      solved = false;
    }
    k = k + 1;
  }
  local_74._0_4_ = 0x766c6f73;
  local_74._4_4_ = 0x63206465;
```

Calling Conventions (DEMO)



```
undefined4
undefined4
char[12]
undefined1[12]
undefined1
char[32]
undefined1

00000e9c 84 b0
00000e9e b0 b5
00000ea0 96 b0
00000ea2 00 af
00000ea4 07 f1 68 04
00000ea8 84 e8 0f 00
00000eac 49 4a
00000eae 07 f1 34 03
00000eb2 07 ca

00000eb4 83 e8 07 00
00000eb8 00 23
00000eba 7b 65

00000ebc 7b 6d
00000ebe 0b 2b
00000ec0 0f dc
00000ec2 7b 6d
00000ec4 40 33
00000ec6 07 f1 68 02
00000eca 13 44
```

```
void challenge_3(void)

{
  bool solved;
  ...
  }
  local_74._0_4_ = 0x766c6f73;
  local_74._4_4_ = 0x63206465;
```

Edit Function at 00000e9c

void challenge_3 (packet p)

Function Name: challenge_3

Calling Convention: __stdcall

Function Attributes:
□ Varargs □ In Line
□ No Return ☑ Use Custom Storage

Function Variables

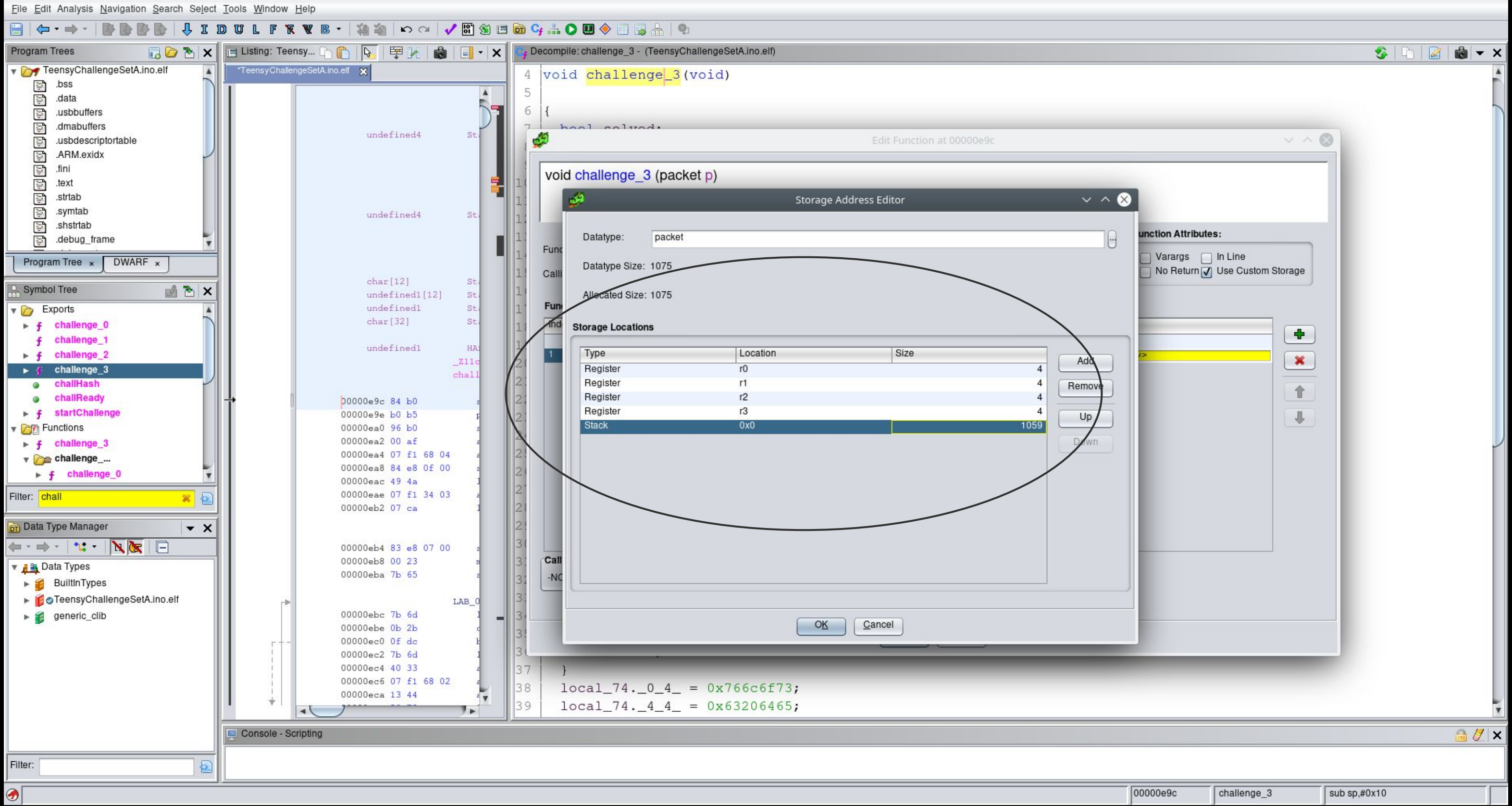
Index	Datatype	Name	Storage
	void	<RETURN>	<VOID>
1	packet	p	<UNASSIGNED>

Call Fixup: -NONE-

Warning: Return Storage and/or Parameter Storage is Unassigned

OK Cancel

Calling Conventions (DEMO)



```
                    undefined4
                    undefined4
                    char[12]
                    undefined1[12]
                    undefined1
                    char[32]
                    undefined1
00000e9c 84 b0
00000e9e b0 b5
00000ea0 96 b0
00000ea2 00 af
00000ea4 07 f1 68 04
00000ea8 84 e8 0f 00
00000eac 49 4a
00000eae 07 f1 34 03
00000eb2 07 ca
00000eb4 83 e8 07 00
00000eb8 00 23
00000eba 7b 65
                              LAB_0
00000ebc 7b 6d
00000ebe 0b 2b
00000ec0 0f dc
00000ec2 7b 6d
00000ec4 40 33
00000ec6 07 f1 68 02
00000eca 13 44
```

```
void challenge_3(void)

{
  bool solved;
  ...
  }
  local_74._0_4_ = 0x766c6f73;
  local_74._4_4_ = 0x63206465;
```

void challenge_3 (packet p)

Function Attributes:

□ Varargs □ In Line
□ No Return ☑ Use Custom Storage

Datatype: packet

Datatype Size: 1075

Allocated Size: 1075

Storage Locations

Type	Location	Size
Register	r0	4
Register	r1	4
Register	r2	4
Register	r3	4
Stack	0x0	1059

00000e9c | challenge_3 | sub sp,#0x10

The broken board

After reversing some challenges we were ready to test our solutions!

But of course, our board was not working….

How can we check our solutions?!

The rise of Dynamic Analysis

A Different Perspective on Emulation

Traditional emulation

The good

➔ Accurate

➔ Fast

The bad

➔ Ad hoc

➔ Low level

Our take

The good

➔ A new lower bound

➔ High level

The bad

➔ Less accurate

➔ Slower

A Different Perspective on Emulation

- The tool is published on GitHub
  https://github.com/TheRomanXpl0it/ghidra-emu-fun
- In this talk
  https://www.megabeets.net/reverse-engineering-a-gameboy-rom-with-radare2/
- As homework
  http://blog.pkh.me/p/11-secball.html

DEMO TIME
(2nd)

Thank you!

Any Question?