Tölvutækni og forritun Verkefni 1

Ragnar Björn Ingvarsson, rbi3

12. október 2024

Hluti I

Verkefnislýsing

Í þessi verkefni eigið þið að bjarga heiminum frá hinum fúlmannlega doktor Vonda. Hann hefur útbúið stafrænar sprengjur sem þið eigið að gera óvirkar. Hver sprengja hefur nokkur þrep. Í hverju þrepi þurfið þið að slá inn tiltekinn streng (eða talnarunu) til að aftengja það þrep og þá færist þið yfir á næsta þrep. Ef þið sláið inn rangan streng þá springur sprengjan og prentar út "BOOM!!!".

Hlutverk ykkar í þessu verkefni er að finna út hvaða strengur aftengir hvert þrep og setja hann inn. Það eru 6 þrep í sprengjunni og þau verða erfiðari eftir því sem lengra er komið.

Mælt er með að nota kembiforrit eins og **gdb** og til að lesa smalamálskóðann að nota **objdump**.

Hluti II

Mín lausn

Phase 1

Ég byrjaði á að skoða phase_1 fallið í smalamálskóðanum og tók eftir í samræmi við bomb.c kóðann að strengurinn minn ætti að vera færður inn í fallið sem fyrsta inntak. Ég vissi að fyrsta inntak er alltaf geymt í %rdi svo ég notaði gdb til að setja break punkt á strings_not_equal fallið til að athuga svo gildi gista með **i r** í gdb og skoðaði svo inntakið mitt í %rdi með **x** skipuninni og formattaði gildið sem streng með **x/s** og vissulega kom mitt inntak.

Eftir það eyddi ég allt of miklum tíma í að skoða hvað strings_not_equal fallið gerir og til að vera hreinskilinn man ég bara ekkert hvað ég gerði til að finna út lausnina en ég hlýt að hafa fattað að fyrst eitthvað gildi var fært inn í %rsi fyrir kall á fallið, er mjög líklegt að fallið taki þá inn tvö inntök og athugi hvort þau séu eins. Þá hef ég orðið forvitinn um hvaða gildi væri sett inn í %rsi og athugað á því eins og með %rdi. Út kemur þá "Verbosity leads to unclear, inarticulate things."

Phase 2

Það fyrsta sem ég tók eftir hér var að fallið phase_2 kallar á fallið read_six_numbers svo það er nokkuð augljóst þaðan að lykilorðið er einhver samsetning af sex tölum. Eftir að grafa í gegn um read_six_numbers í of langan tíma fattaði ég að fallið tekur tvö inntök, fyrsta er strengur af sex tölum, semsagt bara strengurinn minn, og næsta er address í minni sem tölurnar munu verða geymdar á, þar sem ekki er pláss fyrir sex tölur bara í %rax. Til að tékka á hvort þetta sé satt gat ég sett break á explode_bomb fallið og athugað

gildið í minni á addressinu sem %rsp bendir á. Hér lærði ég líka að við hvert kall á fall lækkar %rsp um 8 svo ég þarf að bæta við 8 til að gera ráð fyrir því. Eftir það er auðvelt að sjá frá kóðanum að fyrsta talan er 1 og svo er hver næsta tala eftir það með formúluna: fyrri tala + index á núverandi tölu (0-indexað). Þá er næsta tala 2, næsta 4, svo 7, svo 11, svo 16. Og þar með er þetta leyst. 1 2 4 7 11 16.

Phase 3

Hér er fyrsta skrefið að athuga að fallið __isoc99_sscanf@plt tekur tvö inntök, strenginn til að lesa í %rdi, og streng sem segir hvernig á að formatta þann streng, í þessu tilfelli "%d %c %d". Þá gat ég vitað að lausnin er int char int. Síðan tekur fallið inntök fyrir staðsetningu í minni hvers gildi, svo þrjú auka í þessu tilfelli, fyrsta heiltalan í rsp + 16, characterinn í rsp + 15 og næsta heiltala í rsp + 20.

Fyrsta talan þarf svo að vera minni eða jöfn 7 svo ég ákvað að setja 7 sem fyrsta inntak. Nokkrir reikningar eru svo gerðir og svo kemur forritið að þessari línu.

```
16f2:   3e ff e0                notrack jmp *%rax
```

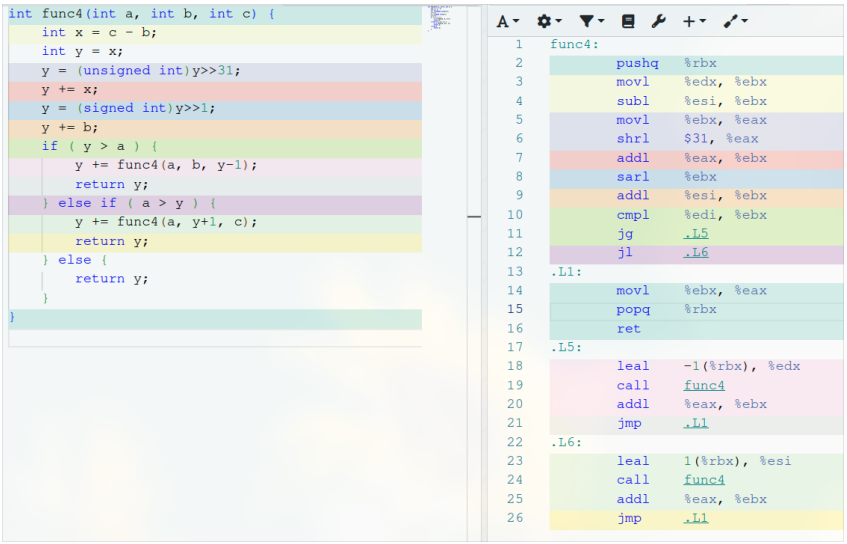
Ég reyndi mikið að leysa hvert forritið væri að hoppa hér með því að reyna að reikna þetta sjálfur en það gekk eitthvað illa, og ég var ekki búinn að læra hvernig á að breaka á sérstökum línum í gdb svo þetta var smá mál. Þangað til að ég fattaði að hægt er að fara áfram þangað til sprengjan springur, stoppa þar, setja "where" inn í gdb og sjá hvert explode_bomb á að fara eftir að það er búið að keyra, og svo athuga hvaðan ég gæti hafa komist þangað. Með þessu fattaði ég að ég hafði hoppað hingað

```
17cb:   b8 50 00 00 00          mov    $0x50,%eax
17d0:   81 7c 24 14 3e 03 00    cmpl   $0x33e,0x14(%rsp)
17d7:   00
17d8:   74 16                   je     17f0 <phase_3+0x153>
17da:   e8 00 05 00 00          call   1cdf <explode_bomb>
```

Og þá er einfalt að breyta hex í ascii og decimal og finna út að strengur sem virkar er þá "7 P 830". (kóðinn sem athugar á char-inum er á 17f0)

Phase 4

Hér notaði ég sömu aðferð með sscanf fallið til að sjá að inntakið ætti að vera tvær heiltölur og sá svo að fyrsta inntakið mátti ekki vera stærra en 14 og var svo tekið sem inntak í func4 fallið ásamt 0 og 14. Búist er svo við gildinu 0x25 frá fallinu svo ég endurskrifaði func4 í c og keyrði for loopu í gegn um það frá 0 - 14.



```
int func4(int a, int b, int c) {
    int x = c - b;
    int y = x;
    y = (unsigned int)y>>31;
    y += x;
    y = (signed int)y>>1;
    y += b;
    if ( y > a ) {
        y += func4(a, b, y-1);
        return y;
    } else if ( a > y ) {
        y += func4(a, y+1, c);
        return y;
    } else {
        return y;
    }
}
```

```
1  func4:
2          pushq   %rbx
3          movl    %edx, %ebx
4          subl    %esi, %ebx
5          movl    %ebx, %eax
6          shrl    $31, %eax
7          addl    %eax, %ebx
8          sarl    %ebx
9          addl    %esi, %ebx
10         cmpl    %edi, %ebx
11         jg      .L5
12         jl      .L6
13 .L1:
14         movl    %ebx, %eax
15         popq    %rbx
16         ret
17 .L5:
18         leal    -1(%rbx), %edx
19         call    func4
20         addl    %eax, %ebx
21         jmp     .L1
22 .L6:
23         leal    1(%rbx), %esi
24         call    func4
25         addl    %eax, %ebx
26         jmp     .L1
```

Ég lenti í miklum vandræðum hér þar sem hvað sem ég reyndi gaf fallið ekki niðurstöðuna 25 þangað til ég fattaði að ég var að leita að 0x25 ekki bara 25, sem er þá í decimal 37 og þá gefur inntakið 10 þá útkomu. Svo sést augljóslega að seinni talan á að vera 37 og þá er lausnin komin, "10 37".

Phase 5

Um leið sést að inntaksstrengurinn þarf að vera af lengd 6. Síðan er lykkja sem fer í gegn um öll bæti strengsins og AND-ar þau við 0xf, sem tekur í raun bara hægri hex stafinn í bætinu. Svo er það gildi notað til að fá eitthvað gildi í minni sem er falið og það gildi bætt við lokaniðurstöðu sem á að vera jöfn 0x25 eða 37 í endann. Ég byrjaði þá á því að athuga öll mögulegu óþekktu gildin í minninu með því að nota gdb til að athuga alltaf á minni í $\%rsi + 4i$ eftir þessa skipun (ég lærði loksins hvernig á að breaka á sérstökum skipunum)

```
    18e4:   48 8d 35 f5 18 00 00    lea    0x18f5(%rip),%rsi        # 31e0 <
array.0>
```

Og þá kemur út: 0x07, 0x03, 0x0a, 0x04, 0x10, 0x09, 0x0b, 0x08, 0x0c, 0x0f, 0x0d, 0x06.

Ég nýti mér þá þetta til að ákveða að ég vil 0x03 þrisvar, 0x06 einu sinni, 0x0a einu sinni og loks 0x0c. Til þess nota ég strenginn "111;28".

Phase 6

Hér er almennt þema á hvernig ég leysti þetta. Ég fattaði að magn kóða í þessu falli er svolítið mikið svo ég ákvað að sleppa því að reyna að afkóða nákvæmlega hvað fallið gerir og í staðinn finna hvernig hvert inntak sem ég setti inn breytti stöðu fallsins og nota það til að komast framhjá öllum köllum á explode_bomb. Svo ég sé náttúrulega fyrst að þetta eru 6 tölur og aðeins seinna í kóðanum er tölunum rennt í gegn um lykkju sem athugar hvort þær séu allar á bilinu 1-6 og hvort þær séu allar ólíkar.

```
1961:   48 83 c3 01             add    $0x1,%rbx
1965:   83 fb 05                cmp    $0x5,%ebx
1968:   0f 8f a7 00 00 00       jg     1a15 <phase_6+0x103>
196e:   41 8b 44 9d 00          mov    0x0(%r13,%rbx,4),%eax
1973:   39 45 00                cmp    %eax,0x0(%rbp)
1976:   75 e9                   jne    1961 <phase_6+0x4f>
```

```
1a1d:   4c 89 f5                mov    %r14,%rbp
1a20:   41 8b 06                mov    (%r14),%eax
1a23:   83 e8 01                sub    $0x1,%eax
1a26:   83 f8 05                cmp    $0x5,%eax
1a29:   0f 87 28 ff ff ff       ja     1957 <phase_6+0x45>
1a2f:   41 83 ff 05             cmp    $0x5,%r15d
1a33:   0f 8f 46 ff ff ff       jg     197f <phase_6+0x6d>
1a39:   4c 89 fb                mov    %r15,%rbx
1a3c:   e9 2d ff ff ff          jmp    196e <phase_6+0x5c>
```

Síðan er rennt í gegn um heilmikið magn reikninga og ég gat séð nokkurnveginn útfrá þessu að við tökum inntakstölurnar og drögum þær allar frá 7 svo þær snúast í raun við. Síðan er rennt í gegn um kóða sem mér sýnist setja einhver gildi falin í minni inn í stað gildanna sem voru lesin með read_six_numbers og hér nennti ég eiginlega ekki að lesa úr öllum þessum reikningum þannig að ég athugaði hvar var næst kallað á explode_bomb.

```
1a50:   39 03                   cmp    %eax,(%rbx)
1a52:   7d ed                   jge    1a41 <phase_6+0x12f>
1a54:   e8 86 02 00 00          call   1cdf <explode_bomb>
```

Frá þessu gat ég athugað með því að breyta inntakinu mínu á mismunandi hátt að verið var að bera saman gildin í minni sem fyrstu tvær tölurnar leiddu til. Með þessu gat ég athugað á gistunum og tékkað hvaða gildi þetta voru. Ég gerði þetta þá fyrst með 1 2 sem fyrstu tölurnar, síðan 3 4 og loks 5 6. Ég var þá kominn með lista yfir hvert gildi sem hver tala leiddi til.

$$1 = 0x33e$$

$$2 = 0x34f$$

$$3 = 0x2c0$$

$$4 = 0x352$$

$$5 = 0x3cb$$

$$6 = 0x299$$

Eftir það er augljóst að sjá að forritið ber saman tölu 1 og tölu 2 og heldur áfram ef tala 1 er stærri en tala 2, svo fer það í 2 og 3 o.s.frv. svo gildin eiga að vera í lækkandi stærðarröð. Auðvelt er þá að raða þeim í svoleiðis röð og fæst út talnarunan 5 4 2 1 3 6.