計算機安全 HW5 Writeup

- Real name: 張瀚文
- Nickname on course website: Hwww
- Student ID: b07505027.

(#°д°)

解法

這題要送payload給 `eval` 執行， `eval` 接收一個字串，並將字串中的程式碼當作php的程式執行，指令要以 ; 結尾。而在php中， `("phpinfo")();` 會被解析成 `phpinfo();` 也就可以執行 `phpinfo()` 這個函數，也就是說我們可以執行任意的函數，如 `system();` ，如果要加參數也可以，比如 `("system")("ls");` 會被解析成 `system("ls");` ，就可以拿到shell了。

但payload必須要小於32個字，且不能包含數字及英文跟`，所以只要想怎麼繞過strlen長度限制跟preg_match就好。

先考慮preg_match，雖然我們不能送英文字過去，但因為可以在eval裡面執行程式，所以可以送編碼過的英文字過去，這時就可以通過preg_match，之後再讓php在eval中去解碼，執行程式。

至於要怎麼實作編碼跟解碼有各種方法，比如XOR、NOT，甚至還有對_做運算來湊出文字的方法...。最後使用了NOT的方法，才能夠符合長度限制。

在php中，可以對字串取 ~ ，比如 `~"system"` 並經過urlencode是 `"%8C%86%8C%8B%9A%92"` ，可以透過php的 `urlencode(~"system")` 得到，再 `~"%8C%86%8C%8B%9A%92"` 就可以變回 `"system"` 。所以我們只要傳
`'~"%8C%86%8C%8B%9A%92"();'` 給 `eval` ，他就會幫我們解碼並執行 `system();` ，參數也是一樣的方法，用程式可以得到 `~"ls /"` 是 `"%93%8C%DF%D0"` ，所以payload是：

```
# system("ls /");
payload = '(~%8C%86%8C%8B%9A%92)(~%93%8C%DF%D0);'
```

其中的url編碼可以不用加引號，因為php對url編碼會預設是字串型態。

利用NOT來做的話，長度小於32就很簡單，安全通過～如果用XOR來做，要傳n個字元就要送大約2*n個字元過去，長度很容易超過。

用python寫程式送好像會因為各種編碼解碼的關係造成eval parseError等等，但用瀏覽器送就沒問題。送出之後可以看到，flag的位置，確認完有flag之後，用上述方法再建構一個可以執行 `system("cat /f*")` 的payload，如下：

```
# system("cat /f*");
payload = '(~%8C%86%8C%8B%9A%92)(~%9C%9E%8B%DF%D0%99%D5);'
```

送出後就可以得到flag了～

Reference

- https://ironhackers.es/en/tutoriales/saltandose-waf-ejecucion-de-codigo-php-sin-letras (https://ironhackers.es/en/tutoriales/saltandose-waf-ejecucion-de-codigo-php-sin-letras)
- https://balsn.tw/ctf_writeup/20181108-defcampctffinal/ (https://balsn.tw/ctf_writeup/20181108-defcampctffinal/)
- https://zhuanlan.zhihu.com/p/80262143 (https://zhuanlan.zhihu.com/p/80262143)

心得

這題也真的滿難的，先試了XOR的方法，雖然可以 `ls` 但想要做其他事就會超過長度限制，一直在想辦法看能不能利用php神奇的特性壓縮，但最多也只能找到一個33個字元的（利用 `$_GET[_]` 達成可以執行任意函數及參數），中間雖然有看到跟想到用NOT，但不管怎麼送都會ParseError，本來以為是這個方法不能用了，但後來才知道用瀏覽器送是可以的。

嘗試的過程跟程式碼可以看 `code/(#°д°)/hack.py` 跟 `code/(#°д°)/index.php`

VISUAL BASIC 2077

解法

先不管要怎麼通過username跟password的檢查，先從比較簡單的開始，先看要送什麼東西才可以拿到flag，再想要怎麼通過檢查

因為通過檢查後，回傳的結果是對username組成的字串經過 `.format(flag=flag)` ，而因為format string的漏洞，我們可以在username裡面放入 `{flag.flag}` 就可以存取到flag物件裡面的flag字串，所以要想辦法讓username可以塞入任意的資料（如 `{flag.flag}` ）。另外這題看起來好像本來是要先取得 `secret_key` ，再去修改session的 `is_admin` ，但如果用 `{flag.flag}` 就可以直接拿到flag而不用管 `is_admin` 。

再來就要想比較困難的username跟password檢查。比較簡單的是在query中注入 `' UNION SELECT "res['username']" AS username, "res['password']" AS password--` 就可以偽造任意的 `res['username']` 跟 `res['password']` ，也就可以在 `username` 放入 `{flag.flag}` ， `password` 則放injection的部分。但困難的是，要怎麼通過檢查，讓「輸入的password跟sqlite抓出來的password相等」，也就是 `password == res['password']` ，因為要在 `password` 裡面做injection， `password` 的子字串 `res['password']` 勢必不會跟 `password` 相同，除非能做到動態更動 `password` 裡面的 `res['password']` ，讓 `res['password']` 在送出去之後才被替換成 `password` 。

這裡參考了hint的概念：利用格式化字串，先創造一個跟自己長得很像的格式化字串變數，再利用格式化字串把自己裡面的%r替換成自己，就可以讓這個字串在執行的時候把自己的子字串替換成自己本身。

有了這個概念，接下來要看sqlite中有什麼語法能夠達成，先試試看如何創造出類似變數的東西，再看看怎麼樣達到格式化字串。

字串變數可以利用 `AS` 來達成，比如 `SELECT s.q AS password FROM (SELECT 'a format string %s' AS q) AS s` ，但是此時的s.q是尚未經過替換後的字串，想要在執行的時候讓sqlite中替換掉就必須用sqlite的語法，類似的功能可能有 `REPLACE` ，但後來找到sqlite可以直接用 `printf` ，也就是說，如果寫 `SELECT printf(s.q, s.q) AS password FROM (SELECT '%s' AS q) AS s` ，就可以成功讓sqlite在執行的時候在s.q中插入s.q，這樣 `res['password']` 就可以跟 `password` 一樣了～

剩下就是微調一下payload，因為利用格式化字串會有一些引號的問題，比如雙引號跟裡面不能直接放雙引號，如果放兩個雙引號讓他escape的話也會有後面要REPLACE的麻煩，或是如果全部都用單引號sqlite會看不懂噴error...。最後用 `quote` 跟 `substr` 勉強解決了引號的問題。另外還有個小問題是換行會讓 `password` 跟 `res['password']` 有一些空格的差異，所以payload有些地方就沒有換行，降低了可讀性。總結一下payload是：

```
# username

{flag.flag}

# password

' UNION SELECT '{flag.flag}' AS username,
printf(s.q, substr(quote(s.q), 2, 130))
AS password FROM (SELECT ''' UNION SELECT ''{flag.flag}'' AS username,
printf(s.q, substr(quote(s.q), 2, 130))
AS password FROM (SELECT ''%s'' AS q) AS s--' AS q) AS s--
```

送出後就可以在username看到flag啦～

嘗試過的payload可以參考 `code/VISUAL BASIC 2077/hack.py` ，測試伺服器可以參考 `code/VISUAL BASIC 2077/app.py` 。

心得

這題也真的很難，完全沒有想過可以利用這樣通過檢查，感覺有點玄，而且網路上也沒有東西可以參考，全部都要自己摸出來的，雖然有給提示也大概看得懂用處，但剛開始還沒想到可以用sqlite的語法來達成，原本還以為是用python來達成，滿有挑戰性的～