



**Pixee**

# Writing Python Codemods for Fun and Profit

Conf42 Python 2024




**Pixee**

**Hi**



I work at Pixee











I build security tools



 **codemodder-python** Public Edit Pins Watch 1 Fork 5 Starred 20

main 22 Branches 26 Tags  Add file Code

 **ryandens**  Upload sonar analysis to Pixee (#291) 9189ced · 8 hours ago 767 Commits

 .github	 Upload sonar analysis to Pixee (#291)	8 hours ago
 ci_tests	New codemod: add-requests-timeout	2 months ago
 integration_tests	Replace pylint with ruff (#277)	yesterday
 src	Replace pylint with ruff (#277)	yesterday
 tests	Replace pylint with ruff (#277)	yesterday
 .coveragerc	Implement new codemod API	last month
 .gitignore	Use setuptools_scm to implement tag-based versioning	4 months ago
 .pre-commit-config.yaml	Replace pylint with ruff (#277)	yesterday
 .semgrepignore	add basic semgrep run	8 months ago


**About**

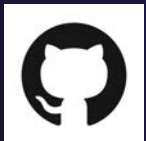
Python implementation of the Codemodder framework

- Readme
- AGPL-3.0 license
- Activity
- Custom properties
- 20 stars
- 1 watching
- 5 forks

Report repository

**Releases** 1

 **v0.80.0** Latest  
4 days ago



<https://github.com/pixee/codemodder-python>

# There is a problem

“Roughly 63% of applications have flaws **in first-party code**”

“We saw that 42% of all applications have flaws that [persist unremediated for longer than one year]”

# Security tools

Checkmarx

sonar

snyk

CodeQL

Semgrep

Contrast  
SECURITY

VERACODE

...security tools?

I pretend I do  
not see it



**We need to fix and harden code**  
***automatically***

**Enter:**



**An open-source codemod  
framework for fixing security  
issues**



# Code + Modification



Enable large-scale refactoring with some human intervention



Quickly apply updates, migrations to a large number of files (JavaScript + Typescript only)



Framework for parsing and transforming Python code

# Codemodder Philosophy

- Fix problems found by other (security) tools
- Tell a story and educate users
- Make changes that are simple to understand and approve

# Leverage Open-Source Tools

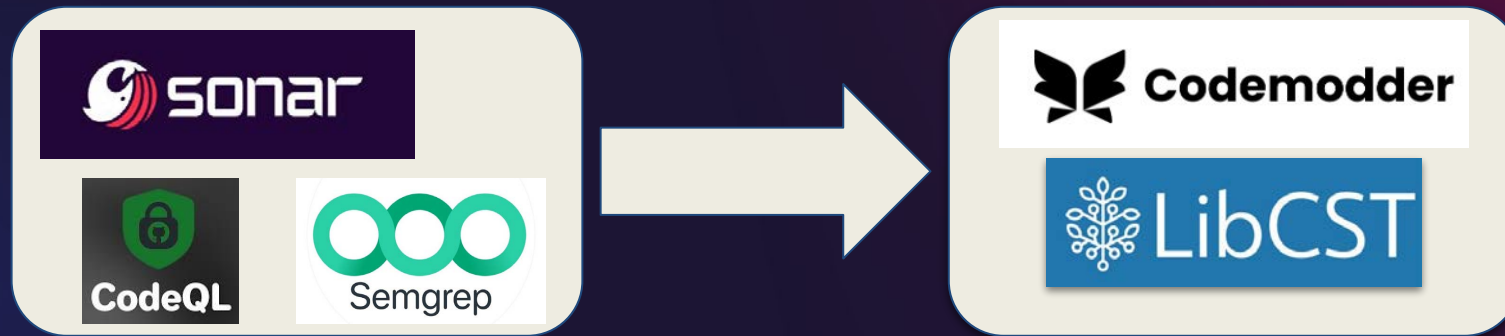


Good at finding problems

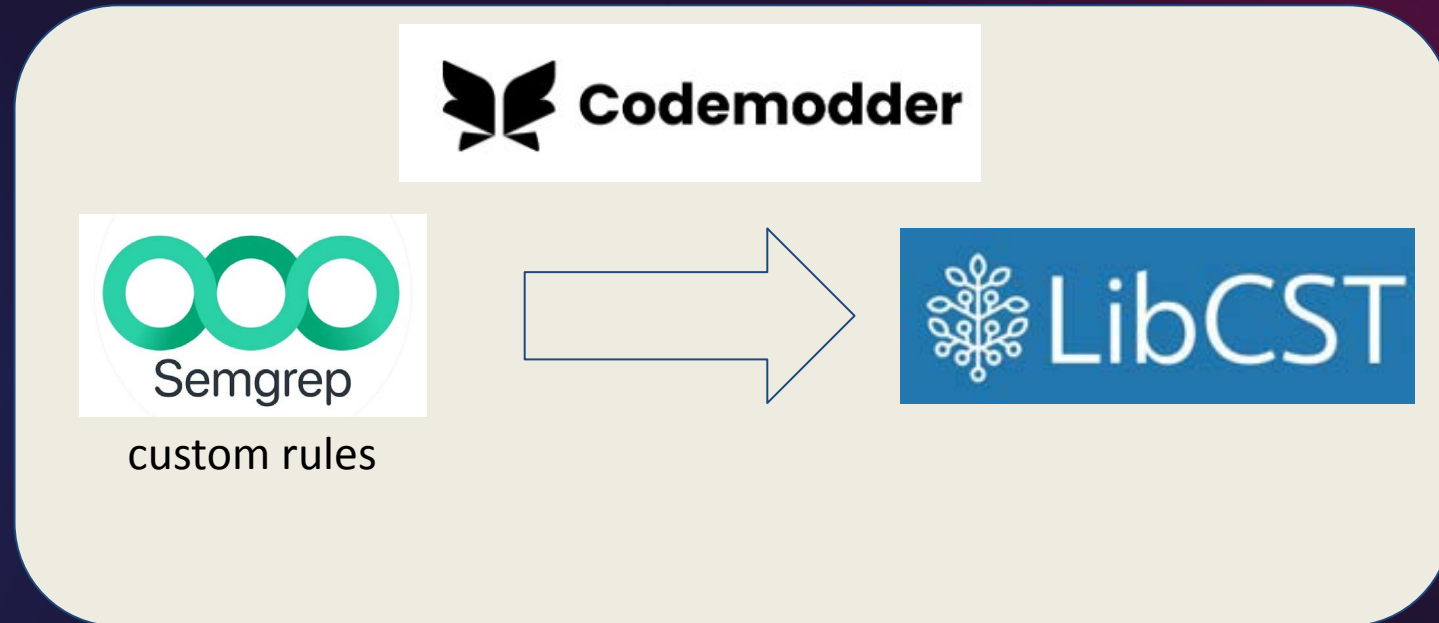


Good at transforming code


# Process results from other tools



# Invoke open-source tools



# Codemods Tell a Story

- Every fix is an opportunity to educate
- Fixes should be comprehensible and compelling
- Fixes should be easy to merge 

# How can I use it?

```
$ pip install codemodder
```

```
$ codemodder -h
```

```
usage: codemodder [-h] [--output OUTPUT] [--codemod-exclude CODEMOD_EXCLUDE | --codemod-include CODEMOD_INCLUDE] [--version] [--list] [--describe]
                  [--output-format {codetf,diff}] [--dry-run | --no-dry-run] [--verbose | --no-verbose] [--log-format {human,json}]
                  [--project-name PROJECT_NAME] [--path-exclude PATH_EXCLUDE] [--path-include PATH_INCLUDE] [--max-workers MAX_WORKERS] [--sarif SARIF]
                  directory
```

Run codemods and change code.

positional arguments:

directory path to find files

options:

```
-h, --help show this help message and exit
--output OUTPUT name of output file to produce
--codemod-exclude CODEMOD_EXCLUDE
                Comma-separated set of codemod ID(s) to exclude
--codemod-include CODEMOD_INCLUDE
                Comma-separated set of codemod ID(s) to include
--version show program's version number and exit
--list Print codemod names to stdout and exit
--describe Print detailed codemod metadata to stdout exit
--output-format {codetf,diff}
                the format for the data output file
--dry-run, --no-dry-run
                do everything except make changes to files
--verbose, --no-verbose
                print more to stdout
--log-format {human,json}
                the format for the log output
--project-name PROJECT_NAME
                optional descriptive name for the project used in log output
--path-exclude PATH_EXCLUDE
                Comma-separated set of UNIX glob patterns to exclude
--path-include PATH_INCLUDE
                Comma-separated set of UNIX glob patterns to include
--max-workers MAX_WORKERS
                maximum number of workers (threads) to use for processing files in parallel
--sarif SARIF
                Comma-separated set of path(s) to SARIF file(s) to feed to the codemods
--sonar-issues-json SONAR_ISSUES_JSON
                Comma-separated set of path(s) to Sonar issues JSON file(s) to feed to the codemods
```

# What does it do?

```
$ codemodder /path/to/my-project
```

- Apply changes to files on disk
- Generate **CodeTF** output files



**What can we fix?**

# Replace unsafe pyyaml loader

```
introduction/views.py
@@ -548,7 +548,7 @@ def a9_lab(request):
    try :
        file=request.FILES["file"]
        try :
-           data = yaml.load(file,yaml.Loader)
+           data = yaml.load(file,yaml.SafeLoader)
        return render(request,"Lab/A9/a9_lab.html",{"data":data})
    except:
```

# Use defusedxml for parsing XML

```
introduction/views.py
@@ -17,9 +17,8 @@
17 17
18 18     from .models import FAANG,info,login,comments,otp
19 19     from random import randint
20 - from xml.dom.pulldom import parseString, START_ELEMENT
20 + from xml.dom.pulldom import START_ELEMENT
21 21     from xml.sax.handler import feature_external_ges
22 - from xml.sax import make_parser
23 22     from django.views.decorators.csrf import csrf_exempt
24 23     from django.template import loader
25 24     from django.template.loader import render_to_string
@@ -39,6 +38,9 @@
39 38     import logging
40 39     import requests
41 40     import re
41 + import defusedxml.pulldom
42 + import defusedxml.sax
43 +
44 44     #*****Login and Registration*****#
45 45
46 46     def register(request):
@@ -247,9 +249,9 @@ def xxe_parse(request):
247 249     @csrf_exempt
248 250     def xxe_parse(request):
249 251
250 -         parser = make_parser()
252 +         parser = defusedxml.sax.make_parser()
251 253     parser.setFeature(feature_external_ges, True)
252 -         doc = parseString(request.body.decode('utf-8'), parser=parser)
254 +         doc = defusedxml.pulldom.parseString(request.body.decode('utf-8'), parser=parser)
253 255     for event, node in doc:
254 256         if event == START_ELEMENT and node.tagName == 'text':
255 257             doc.expandNode(node)
```

# Automatically close resources

```
import tempfile
path = tempfile.NamedTemporaryFile().name
-file = open(path, 'w', encoding='utf-8')
-file.write('Hello World')
+with open(path, 'w', encoding='utf-8') as file:
+    file.write('Hello World')
```

# Parameterize SQL queries

144	144		<code>self.connection = self.connect()</code>
145	145		<code>self.cursor = self.connection.cursor()</code>
146	146		
147		-	<code>query = f"SELECT COUNT(*) FROM users WHERE token = '{token}'"</code>
148		-	<code>self.cursor.execute(query)</code>
	147	+	<code>query = f"SELECT COUNT(*) FROM users WHERE token = ?"</code>
	148	+	<code>self.cursor.execute(query, (token, ))</code>

# Use generator expressions

12

12

13

```
- total = sum([x for x in range(1, 101)])
```

13

```
+ total = sum(x for x in range(1, 101))
```

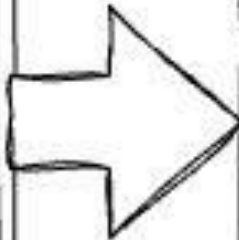
# Codemodder Architecture

- Detector Finds problems
- Transformer Changes code
- Metadata Tells a story

# BaseCodemod

**Detector**

- sonar
- CodeQL
- Semgrep



### Metadata

Name	Summary	Description
		References

### TransformerPipeline

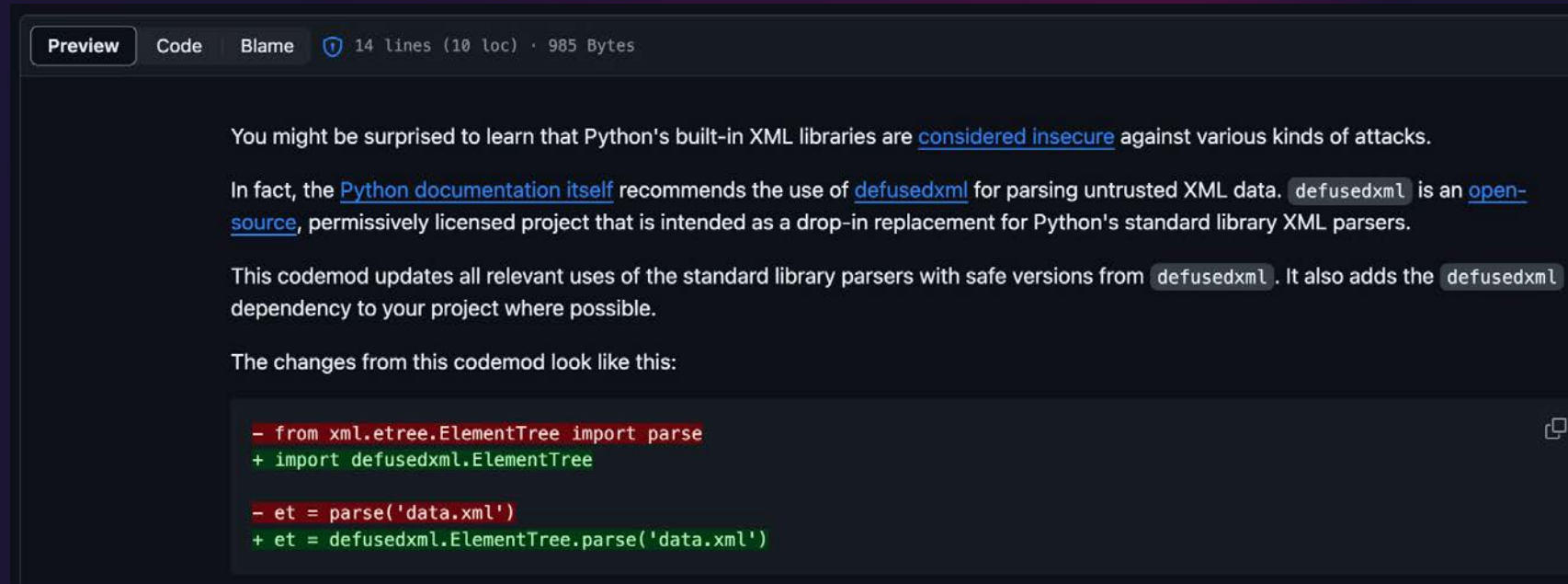
LibCST  
Transformer

LibCST  
Transformer  
...



# Codemod Metadata

- Name **pixee:python/use-defusedxml**
- Summary **Use defusedxml for parsing XML**
- Description



The screenshot shows a GitHub Codemod preview interface. At the top, there are tabs for 'Preview', 'Code', and 'Blame', followed by a lock icon and the text '14 lines (10 loc) · 985 Bytes'. The main content area contains the following text:

You might be surprised to learn that Python's built-in XML libraries are [considered insecure](#) against various kinds of attacks.

In fact, the [Python documentation itself](#) recommends the use of [defusedxml](#) for parsing untrusted XML data. `defusedxml` is an [open-source](#), permissively licensed project that is intended as a drop-in replacement for Python's standard library XML parsers.

This codemod updates all relevant uses of the standard library parsers with safe versions from `defusedxml`. It also adds the `defusedxml` dependency to your project where possible.

The changes from this codemod look like this:

```
- from xml.etree.ElementTree import parse
+ import defusedxml.ElementTree

- et = parse('data.xml')
+ et = defusedxml.ElementTree.parse('data.xml')
```

# Dependency Management

- Sometimes codemods add dependencies
- Doing this in Python is not really easy

```
1 requirements.txt
... @@ -1,2 +1,3 @@
1 requests~=2.30.0
2 flask~=3.0.0
3 + defusedxml~=0.7.1
```

# Dependency Management

## pyproject.toml

```
[project]
# ...
dependencies = [
    "docutils",
    "BazSpam == 1.1",
]
# ...
```

## setup.cfg

```
[options]
#...
install_requires =
    docutils
    BazSpam ==1.1
```

## setup.py

```
setup(
    ...,
    install_requires=[
        'docutils',
        'BazSpam ==1.1',
    ],
)
```

**Let's write a codemod**

# Codemod Plugin Template

The screenshot shows a GitHub repository page for 'cookiecutter-codemodder-plugin'. The repository is public and has 10 stars, 0 forks, and 0 watches. The main branch is 'main'. The repository contains several files and folders, including 'hooks', '{{ cookiecutter.project\_slug }}', '.gitignore', 'LICENSE', 'README.md', and 'cookiecutter.json'. The README file is selected, showing the repository's name and a description: 'A Cookiecutter template for generating custom Python Codemodder plugins'. The 'Setup' section provides instructions on how to install Cookiecutter and generate a new plugin project, with a code block showing the commands: '\$ pip install cookiecutter' and '\$ cookiecutter https://github.com/pytest-dev/cookiecutter-codemodder-plugin'. The right sidebar contains information about the repository, including the number of stars, forks, and watches, and a list of files like 'Readme', 'MIT license', and 'Activity'.

cookiecutter-codemodder-plugin Public

Edit Pins Watch 0 Fork 0 Starred 10

main 1 Branch 0 Tags

Go to file Add file Code

drdavella Add license section to README 9d6e8c7 · 5 hours ago 9 Commits

hooks	Use cookiecutter (#4)	5 hours ago
{{ cookiecutter.project_slug }}	Use cookiecutter (#4)	5 hours ago
.gitignore	More updates. Squash later	2 days ago
LICENSE	Update LICENSE to use MIT	12 hours ago
README.md	Add license section to README	5 hours ago
cookiecutter.json	Use cookiecutter (#4)	5 hours ago

README MIT license

## cookiecutter-codemodder-plugin

A [Cookiecutter](#) template for generating custom [Python Codemodder](#) plugins

### Setup

Install [Cookiecutter](#) and then generate a new plugin project:

```
$ pip install cookiecutter
$ cookiecutter https://github.com/pytest-dev/cookiecutter-codemodder-plugin
```

### About

Template repository for creating custom Python codemod plugins

- Readme
- MIT license
- Activity
- Custom properties

10 stars  
0 watching  
0 forks

Report repository

### Releases

No releases published  
[Create a new release](#)

### Packages

No packages published  
[Publish your first package](#)

### Languages

- Python 100.0%

```
from core_codemods.api import CoreCodemod
```

```
SecureRandom = CoreCodemod(  
    metadata=metadata,  
    detector=detector,  
    transformer=transformer,  
)
```

```
from codemodder.codemods.api import Metadata
```

```
metadata = Metadata(  
    name="secure-random",  
    review_guidance=ReviewGuidance.MERGE_AFTER_CURSORY_REVIEW,  
    summary="Secure Source of Randomness",  
)
```

```
1. from codemodder.codemods.semgrep import SemgrepRuleDetector
2.
3.
4. detector = SemgrepRuleDetector(
5.     """
6.     - patterns:
7.       - pattern: random.$F(...)
8.       - pattern-not: random.SystemRandom()
9.       - pattern-inside: |
10.         import random
11.         ...
12.     """
13. )
```



```
1. from codemodder.codemods.api import LibcstResultTransformer, LibcstTransformerPipeline
2.
3.
4. class TransformSecureRandom(LibcstResultTransformer):
5.     change_description = (
6.         "Replace random.{func} with more secure secrets library functions."
7.     )
8.
9.     def on_result_found(self, original_node, updated_node):
10.         self.remove_unused_import(original_node)
11.         self.add_needed_import("secrets")
12.         return self.update_call_target(updated_node, "secrets.SystemRandom()")
13.
14.
15. transformer = LibcstTransformerPipeline(TransformSecureRandom)
```

```
from core_codemods.api import CoreCodemod
```

```
SecureRandom = CoreCodemod(
```

```
    metadata=metadata,
```

```
    detector=detector,
```

```
    transformer=transformer,
```

```
)
```

```

8 introduction/views.py
6 6 from requests.structures import CaseInsensitiveDict
7 7 from django.contrib.auth import login,authenticate
8 8 from django.contrib.auth.forms import UserCreationForm
9 - import random
10 9 import string
11 10 import os
12 11 from hashlib import md5
@@ -16,7 +15,6 @@
16 15 #*****Lab Requirements*****#
17 16
18 17 from .models import FAANG,info,login,comments,otp
19 - from random import randint
20 18 from xml.dom.pulldom import parseString, START_ELEMENT
21 19 from xml.sax.handler import feature_external_ges
22 20 from xml.sax import make_parser
@@ -39,6 +37,8 @@
39 37 import logging
40 38 import requests
41 39 import re
40 + import secrets
41 +
42 42 #*****Login and Registration*****#
43 43
44 44 def register(request):
@@ -484,7 +484,7 @@ def login_otp(request):
484 484 def Otp(request):
485 485     if request.method=="GET":
486 486         email=request.GET.get('email')
487 -         otpN=randint(100,999)
487 +         otpN=secrets.SystemRandom().randint(100,999)
488 488         if email and otpN:
489 489             if email=="admin@pygoat.com":
490 490                 otp.objects.filter(id=2).update(otp=otpN)
@@ -668,7 +668,7 @@ def a10_lab2(request):
668 668 #*****A11*****#
669 669
670 670 def gentckt():
671 -     return (''.join(random.choices(string.ascii_uppercase + string.ascii_lowercase, k=10)))
671 +     return (''.join(secrets.SystemRandom().choices(string.ascii_uppercase + string.ascii_lowercase, k=10)))

```

**Make the easy things easy**

```
class SecureRandom(SimpleCodemod):
```

```
    metadata = Metadata(  
        name="secure-random",  
        review_guidance=ReviewGuidance.MERGE_AFTER_CURSORY_REVIEW,  
        summary="Secure Source of Randomness",  
    )
```

```
    detector_pattern = """  
    - patterns:  
      - pattern: random.$F(...)  
      - pattern-not: random.SystemRandom()  
      - pattern-inside: |  
          import random  
          ...  
    ""
```

```
    change_description = (  
        "Replace random.{func} with more secure secrets library functions."  
    )
```

```
    def on_result_found(self, original_node, updated_node):  
        self.remove_unused_import(original_node)  
        self.add_needed_import("secrets")  
        return self.update_call_target(updated_node, "secrets.SystemRandom()")
```

**Make the hard things possible**

```
class SubprocessShellFalse(SimpleCodemod, NameResolutionMixin):
    metadata = Metadata(...)
    change_description = "Set `shell` keyword argument to `False`"
    SUBPROCESS_FUNCS = [
        f"subprocess.{func}"
        for func in {"run", "call", "check_output", "check_call", "Popen"}
    ]

    METADATA_DEPENDENCIES = SimpleCodemod.METADATA_DEPENDENCIES + (ParentNodeProvider,)
    IGNORE_ANNOTATIONS = ["S603"]
```

```
def leave_Call(self, original_node: cst.Call, updated_node: cst.Call):
```

```
    if not self.filter_by_path_includes_or_excludes(
        self.node_position(original_node)
    ):
        return updated_node
```

```
    if self.find_base_name(original_node.func) in self.SUBPROCESS_FUNCS:
        for arg in original_node.args:
            if matchers.matches(
                arg,
                matchers.Arg(
                    keyword=matchers.Name("shell"), value=matchers.Name("True")
                ),
            ) and not is_disabled_by_annotations(
                original_node,
                self.metadata, # type: ignore
                messages=self.IGNORE_ANNOTATIONS,
            ):
                self.report_change(original_node)
                new_args = self.replace_args(
                    original_node,
                    [NewArg(name="shell", value="False", add_if_missing=False)],
                )
                return self.update_arg_target(updated_node, new_args)
        return original_node
```

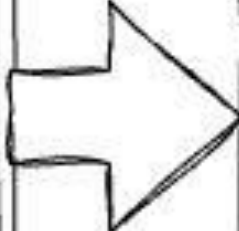


**Looking ahead**



# BaseCodemod

A vertical rectangular box labeled "Detector" containing three logos: Sonar (top), CodeQL (middle), and Semgrep (bottom).



### Metadata

Name	Summary	Description
		References

### TransformerPipeline

A diagram showing a sequence of components within a "TransformerPipeline" box. It starts with an "OpenAI" logo and "Transformer" text, followed by a "LibCST" logo and "Transformer" text. An arrow with three dots points to a second, identical "OpenAI" and "LibCST" Transformer block, with three dots below it, indicating a pipeline of multiple transformers.

# Basemod



Detector



## Metadata

Name

Summary

Description

References

## Transformer Pipeline



Transformer



AI



ner



# We want your feedback!

- GitHub Issues
- Clone/fork
- Stars
- Contribute codemods and/or ideas

# Pixeebot App →

The screenshot shows a GitHub pull request interface for the repository 'pixee / pygoat'. The pull request title is 'Use defusedxml for Parsing XML #2'. The pull request is from 'pixeebot/drip-2024-01-13-pixee-python/use-defusedxml' and is being merged into the 'master' branch. The pull request is currently open. The pull request description includes a comment from 'pixeebot' (bot) that explains the security implications of Python's built-in XML libraries and the benefits of using 'defusedxml'. The comment also includes a diff showing the changes made to the code.

Use `defusedxml` for Parsing XML #2

Open pixeebot wants to merge 1 commit into master from pixeebot/drip-2024-01-13-pixee-python/use-defusedxml

Conversation 1 Commits 1 Checks 0 Files changed 1

pixeebot bot commented 2 weeks ago

You might be surprised to learn that Python's built-in XML libraries are [considered insecure](#) against various kinds of attacks.

In fact, the [Python documentation itself](#) recommends the use of `defusedxml` for parsing untrusted XML data. `defusedxml` is an [open-source](#), permissively licensed project that is intended as a drop-in replacement for Python's standard library XML parsers.

This codemod updates all relevant uses of the standard library parsers with safe versions from `defusedxml`. It also adds the `defusedxml` dependency to your project where possible.

The changes from this codemod look like this:

```
- from xml.etree.ElementTree import parse
+ import defusedxml.ElementTree

- et = parse('data.xml')
+ et = defusedxml.ElementTree.parse('data.xml')
```

The screenshot shows a terminal window titled 'Terminalizer'. The prompt is 'Daniels-MBP:~ danieldavello\$'. The terminal is currently empty, showing only the prompt and a cursor.

```
Daniels-MBP:~ danieldavello$
```

# ← Pixee CLI



# Pixee



@drdavella

dan.davella@pixee.ai

<https://pixee.ai>



pixee / [codemodder-python](#)