joblib / joblib Public

Code | Issues 319 | Pull requests 55 | Actions | Projects 1 | Wiki | ...

New issue

Jump to bottom

The potential security vulnerability for the flag pre_dispatch in Parallel() class due to the eval() statement. #1128

Closed jimlinntu opened this issue on Nov 11, 2020 · 10 comments · Fixed by #1321

---

jimlinntu commented on Nov 11, 2020 • edited

As the title shows, if you try to enter a statement in the flag `pre_dispatch`, it will run whatever you want to run.
This should present a potential security vulnerability.

```
def f():
    return 1
p = Parallel(n_jobs=3, pre_dispatch="sys.exit(0)")
p(delayed(f)() for i in range(10)) # this will cause the system to exit
```

joblib/joblib/parallel.py
Line 1020 in 53a8173

```
1020        pre_dispatch = eval(pre_dispatch)
```

jimlinntu changed the title ~~The potential security issue for the flag pre_dispatch in Parallel() class due to the eval() statement.~~ **The potential security vulnerability for the flag pre_dispatch in Parallel() class due to the eval() statement.** on Nov 11, 2020

ogrisel commented on Dec 15, 2020 Contributor

Thanks for the report. We should indeed improve this by making sure that the `pre_dispatch` expression only has expression to a list of symbols and cannot import modules.

ogrisel commented on Dec 15, 2020 Contributor

Something like the following should work:

```
# Set builtins to empty dict to make it impossible to import arbitrary modules
# and other unsafe operation.
pre_dispatch = eval(pre_dispatch, {"n_jobs": n_jobs, "__builtins__": {}})
```

ogrisel commented on Dec 15, 2020 Contributor

We might want to add a whitelist of allowed built-in that can be useful for arithmetic operations (e.g. round, ceil, int, abs, float...), though.

👍 1

jimlinntu commented on Jan 5, 2021 Author

> Something like the following should work:
>
> ```
> # Set builtins to empty dict to make it impossible to import arbitrary modules
> # and other unsafe operation.
> pre_dispatch = eval(pre_dispatch, {"n_jobs": n_jobs, "__builtins__": {}})
> ```

Cool!
Thanks for your suggestion.
I think I can make a pull request for it later!

adrinjalali mentioned this issue on Sep 5

FIX make sure pre_dispatch cannot do arbitrary code execution #1321

Merged

adrinjalali commented on Sep 5 Contributor

Opened #1321

ogrisel closed this as completed in #1321 on Sep 5

---

adrinjalali mentioned this issue on Sep 12

FIX parse pre-dispatch with AST instead of calling eval #1327

Merged

miraculixx commented on Sep 26 • edited

@jimlinntu @ogrisel What is the rationale of marking this a *security* issue? Presumably if someone uses `Parallel` they can already submit *whatever* code they want.

Unless I'm missing something, restricting `pre_dispatch` expressions will not change the security level of joblib (and this issue is not a potential security vulnerability to begin with, afaict). Even with the rather complex fix in #1327 the following will reproduce the exact behavior. To be sure, this is intended behavior, and there is no need to restrict `f()`, as that would render `Parallel` unusable.

```
def f():
    sys.exit(0)
p = Parallel(n_jobs=3, pre_dispatch="5*n_jobs")
p(delayed(f)() for i in range(10)) # this will cause the system to exit
```

The rationale for using `eval(pre_dispatch)` in the code is for syntactic reasons, such that the developer can write code like `Parallel(..., pre_dispatch='5*n_jobs')`, as noted in the docstring. If we think use of `eval()` should be avoided (in case of what use case?), I would suggest the better path would be to deprecate this feature and remove it all together, or perhaps replace it with a callable. With a callable, the code would then be e.g. `Parallel(..., pre_dispatch=lambda : 5 * n_jobs)` which is only slightly less readable, albeit a lot more annoying to write .

ogrisel commented on Sep 26 Contributor

If someone exposes the pre_dispatch parameter in a user facing web ui or config file for instance they might not expect that this can lead to arbitrary python code injection.

ogrisel commented on Sep 26 Contributor

I would be ok to accept a callable if needed.

👍 1

miraculixx commented on Sep 26 • edited

@ogrisel Ok, I see how this might be a potential issue, though I would think this comes down to secure programming practices, i.e. never trust user input. Anyhow, I would prefer if pre_dispatch were changed into accepting expressions as a callable only, or perhaps to document the use of eval().

adrinjalali commented on Sep 27 Contributor

It's not just a user facing web UI. This argument is passed to `Parallel` by downstream libraries and that means if those downstream libraries somehow load some pre-existing persisted object, the user can easily exploit the fact that whatever's passed to pre_dispatch runs in `eval`.

We certainly shouldn't allow callable as an expression, since that itself again opens the door to easy exploitation.

Accepting a callable, however, might be okay.

thomasjpfan mentioned this issue on Sep 30

RFC bump up dependencies for 1.2 scikit-learn/scikit-learn#24401

Closed

abdel91 mentioned this issue on Oct 7

The potential security vulnerability on the joblib library tensorflow/data-validation#226

Closed

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging a pull request may close this issue.

FIX make sure pre_dispatch cannot do arbitrary code execution
adrinjalali/joblib

4 participants

4 participants