New issue

Jump to bottom

Insecure string comparison (incomplete comparison) in _convert_from_str of descriptor.c #18993

Closed **Daybreak2019** opened this issue on May 11, 2021 · 32 comments

Labels 00 - Bug component: numpy.core

Milestone 1.22.0 release

Daybreak2019 commented on May 11, 2021

Reproducing code example:

Snippet:

```
/* Check for a deprecated Numeric-style typecode */
/* `Uint` has deliberately weird uppercasing */
char *dep_tps[] = {"Bytes", "Datetime64", "Str", "Uint"};
int ndep_tps = sizeof(dep_tps) / sizeof(dep_tps[0]);
for (int i = 0; i < ndep_tps; ++i) {
    char *dep_tp = dep_tps[i];
    if (strncmp(type, dep_tp, strlen(dep_tp)) == 0) {   ------> '\0' not considered here, should be strlen(dep_tp)+1. (value of "type" may come from external modules)
        /* Deprecated 2020-06-09, NumPy 1.20 */
        if (DEPRECATE("Numeric-style type codes are "
                      "deprecated and will result in "
                      "an error in the future.") < 0) {
            goto fail;
        }
    }
}
```

Error message:

When we run our analysis tool on NumPy, an incomplete comparison problem was reported, see details below:
File: numpy/core/src/multiarray/descriptor.c
Function: _convert_from_str (line 1727 : 1740)
Optional call-path: PyArray_DescrAlignConverter -> _convert_from_any -> _convert_from_str
Details in description

NumPy/Python version information:

the main branch of NumPy

seberg commented on May 12, 2021 Member

I doubt it matters (its not like this is relevant for security or anything). But adding the `+1` would be good, also to avoid the slightly wrong things being copied.

seberg added 00 - Bug component: numpy.core labels on May 12, 2021

Daybreak2019 changed the title ~~Unsecure string comparison (incomplete comparison) in _convert_from_str of descriptor.c~~ Insecure string comparison (incomplete comparison) in _convert_from_str of descriptor.c on May 12, 2021

NectDz commented on Jul 14, 2021 Contributor

Do you mind telling me the tool that was used to reproduce to get that error message?

NectDz commented on Jul 14, 2021 • edited by seberg — Contributor

1. Forked numpy and then cloned it
2. Ran python tests using "python3 runtests.py -v" and the build passed
3. The change I made was the following

```
diff --git a/numpy/core/src/multiarray/descriptor.c b/numpy/core/src/multiarray/descriptor.c
index 58aa608c3..f6202c7ea 100644
--- a/numpy/core/src/multiarray/descriptor.c
+++ b/numpy/core/src/multiarray/descriptor.c
@@ -1729,7 +1729,7 @@ _convert_from_str(PyObject *obj, int align)
         for (int i = 0; i < ndep_tps; ++i) {
             char *dep_tp = dep_tps[i];

-            if (strncmp(type, dep_tp, strlen(dep_tp)) == 0) {
+            if (strncmp(type, dep_tp, strlen(dep_tp)) + 1) {
                 /* Deprecated 2020-06-09, NumPy 1.20 */
                 if (DEPRECATE("Numeric-style type codes are "
                               "deprecated and will result in "
```

4. Ran the tests again and received the following error: **ERROR numpy/core/tests/test_regression.py - TypeError: data type 'int32' not understood**

Do you know where I might of gone wrong and why I am seeing this error?

eric-wieser commented on Jul 14, 2021 • edited — Member

That change is nonsense, you removed the `==` .

2

seberg commented on Jul 14, 2021 — Member

You removed the `== 0` , that would invert the logic.

1

NectDz mentioned this issue on Jul 21, 2021

DEP: Remove deprecated numeric style dtype strings #19539

Merged

seberg closed this as completed in `eeef9d4` on Aug 10, 2021

---

scratchmex pushed a commit to scratchmex/numpy that referenced this issue on Aug 13, 2021

`DEP: Remove deprecated numeric style dtype strings (numpy#19539)` ··· `bc3ffb2`

rgommers modified the milestones: **1.21.0 release**, **1.22.0 release** on Dec 22, 2021

anjaneya17 mentioned this issue on Dec 28, 2021

Exclude transitive dependency from requirements.txt linkedin/pygradle#355

Open

chrisfeltner commented on Jan 3

I realize the issue is closed, but would like to note that it has been assigned CVE-2021-34141

4

seberg commented on Jan 3 — Member

This code doesn't exist anymore anyway, but can someone explain what even the point of a CVE is here? At this point, the CVE feels just like useless alarming about completely harmless things.

rgommers commented on Jan 4 — Member

> At this point, the CVE feels just like useless alarming about completely harmless things.

Yes indeed. I've recently reviewed six CVEs on NumPy for the Tidelift database, and IIRC the only one that was not completely baseless I think was the one about `np.load` defaulting to `allow_pickle=True` (and even that one already had a warning and was quite harmless).

2

b-abderrahmane commented on Jan 4

> At this point, the CVE feels just like useless alarming about completely harmless things.
>
> Yes indeed. I've recently reviewed six CVEs on NumPy for the Tidelift database, and IIRC the only one that was not completely baseless I think was the one about `np.load` defaulting to `allow_pickle=True` (and even that one already had a warning and was quite harmless).

Indeed, It seems pretty odd to make a CVE for such a problem with a CVSS scoring of 9.8, It would be interesting to see where this is coming from and to try and stop this trend.

One more thing, If I understand correctly, this code along with user provided dtypes have been dropped only starting 1.22.0
What I don't understand is why for instance here the affected versions are those between 1.9.0 and 1.9.3?

westonsteimel commented on Jan 4

> > At this point, the CVE feels just like useless alarming about completely harmless things.
>
> Yes indeed. I've recently reviewed six CVEs on NumPy for the Tidelift database, and IIRC the only one that was not completely baseless I think was the one about `np.load` defaulting to `allow_pickle=True` (and even that one already had a warning and was quite harmless).
>
> Indeed, It seems pretty odd to make a CVE for such a problem with a CVSS scoring of 9.8, It would be interesting to see where this is coming from and to try and stop this trend.
>
> One more thing, If I understand correctly, this code along with user provided dtypes have been dropped only starting 1.22.0
> What I don't understand is why for instance here the affected versions are those between 1.9.0 and 1.9.3?

It's very possible the NVD team have just made a mistake with it. All of these entries are still reviewed manually as far as I am aware, and they do make mistakes. You can email them suggesting a correction or further review and usually get a reply back in a couple of days. I was also wondering about these last few as they auto-fed into the PyPI Advisory Database and it looked as though the affected version ranges weren't quite right on the underlying NVD entries.

b-abderrahmane commented on Jan 4

> > > At this point, the CVE feels just like useless alarming about completely harmless things.
> >
> > Yes indeed. I've recently reviewed six CVEs on NumPy for the Tidelift database, and IIRC the only one that was not completely baseless I think was the one about `np.load` defaulting to `allow_pickle=True` (and even that one already had a warning and was quite harmless).
> >
> > Indeed, It seems pretty odd to make a CVE for such a problem with a CVSS scoring of 9.8, It would be interesting to see where this is coming from and to try and stop this trend.
> > One more thing, If I understand correctly, this code along with user provided dtypes have been dropped only starting 1.22.0
> > What I don't understand is why for instance here the affected versions are those between 1.9.0 and 1.9.3?
>
> It's very possible the NVD team have just made a mistake with it. All of these entries are still reviewed manually as far as I am aware, and they do make mistakes. You can email them suggesting a correction or further review and usually get a reply back in a couple of days. I was also wondering about these last few as they auto-fed into the PyPI Advisory Database and it looked as though the affected version ranges weren't quite right on the underlying NVD entries.

It could very well be the case. I just send them an e-mail about it.

👍 2

rgommers commented on Jan 4 Member

> It could very well be the case. I just send them an e-mail about it.

Please do not ask them to update to newer versions without objecting to this being a critical CVE. That's just going to give a lot of people extra work, for a nonsense CVE (which also wasn't disclosed correctly by the way).

b-abderrahmane commented on Jan 4

> Please do not ask them to update to newer versions without objecting to this being a critical CVE. That's just going to give a lot of people extra work, for a nonsense CVE (which also wasn't disclosed correctly by the way).

Could you explain which part of the disclosure was not done correctly?

westonsteimel commented on Jan 4

@rgommers, do you think there is any chance they'd withdraw it completely (along with any of the others you'd consider nonsense)? Also, I don't know if there's any good way of finding where they're coming from, but I can do some asking around

rgommers commented on Jan 4 Member

> Could you explain which part of the disclosure was not done correctly?

When you think you have discovered a vulnerability, you get in touch with the project (in private, there are best practices for this). Our way of doing this is clearly documented in the main README and a few other places:

- **Bug reports:** https://github.com/numpy/numpy/issues
- **Report a security vulnerability:** https://tidelift.com/docs/security

In this case, I don't think anyone got in touch. Someone just opened a CVE.

> **@rgommers**, do you think there is any chance they'd withdraw it completely (along with any of the others you'd consider nonsense)?

I don't know, I've seen the status change to "disputed" sometimes, but I've never seen the actual process directly with people who can assign CVE numbers.

> Also, I don't know if there's any good way of finding where they're coming from, but I can do some asking around

I don't know of a way of finding out, but that may just be my lack of knowledge. If you could look into that, that'd be great.

👍 4

rgommers commented on Jan 4 Member

> Also, I don't know if there's any good way of finding where they're coming from

After yet another CVE came in, I looked closer and these are all issues by **@Daybreak2019**. So **@Daybreak2019** are you requesting these, or someone who works with you?

achraf-mer commented on Jan 11

Any updates on the status of CVE-2021-34141 with regards to current issue?
As mentioned in previous comment, is there any chance the severity gets adjusted, or CVE totally withdrawn if does not make sense?
It just adds unnecessary complexity for numpy users with upgrades.

seberg commented on Jan 11 Member

These CVEs are nonsense, there is no way this is exploitable (in this case even meaningful if it was exploited) by a user that is not already privileged. I do not think we have tried to get the CVE marked as disputed, but it sounds like the CVE process is too weird to expect more than it getting tagging on a "disputed" with a reason...

rgommers commented on Jan 12 Member

Also, since the fix is to remove deprecated code, we can't backport that fix as is to `1.21.6` (if we did plan to make such a release, which we currently don't).

So for people who must avoid CVEs due to policies in their organization, they should unfortunately upgrade to `1.22.0`. Everyone else can just ignore this non-issue.

👍 1

westonsteimel commented on Jan 12

Looking at the history on it it looks like they briefly adjusted it to DISPUTED and then reverted it, but looks like they also adjusted the score down from Critical at least?

10 hidden items
Load more...

sbrunner added a commit to camptocamp/c2cgeoportal that referenced this issue on Feb 3

 `Fix CVE` ··· ✓ `8f1cc41`

sbrunner added a commit to camptocamp/c2cgeoportal that referenced this issue on Feb 3

 `Fix CVE` ··· ✓ `0bcd811`

sbrunner added a commit to camptocamp/c2cgeoportal that referenced this issue on Feb 4

 `Fix CVE` ··· ✗ `7b0ebe6`

sbrunner added a commit to camptocamp/c2cgeoportal that referenced this issue on Feb 4

 `Fix CVE` ··· ✗ `30454e8`

sbrunner added a commit to camptocamp/c2cgeoportal that referenced this issue on Feb 4

 `Fix CVE` ··· ✗ `c5af071`

sbrunner added a commit to camptocamp/c2cgeoportal that referenced this issue on Feb 4

 `Fix CVE` ··· ✗ `76cc0b7`

sbrunner added a commit to camptocamp/c2cgeoportal that referenced this issue on Feb 4

 `Fix CVE` ··· ✗ `a92fb7a`

sbrunner added a commit to camptocamp/c2cgeoportal that referenced this issue on Feb 4

 `Fix CVE` ··· ✓ `67abe59`

sbrunner added a commit to camptocamp/c2cgeoportal that referenced this issue on Feb 4

 `Ignore CVE` ··· ✓ `52c40f6`

Ren0216 commented on Feb 16

Hi，can someone helps? Is it properly to update "strlen(dep_tp)" to "strlen(dep_tp)+1" in the line "if (strncmp(type, dep_tp, strlen(dep_tp)) == 0) {" within numpy-1.16.5 to fix cve-2021-34141? We do not want to upgrade for some reasons. Thanks.
@seberg @rgommers

rgommers commented on Feb 16 Member

@Ren0216 I'd recommend re-applying the fix from gh-19539, and possible previous fixes to the same code. Or just leave it alone, since **@seberg** explained in the comment above that this CVE isn't valid. Trying to come up with a new fix isn't a good idea, it's much more likely to cause problems than solve them.

VachaShah mentioned this issue on Mar 8

CVE-2021-34141 (Medium) detected in numpy-1.21.5-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl opensearch-project/opensearch-py#127

Closed

dbalchev pushed a commit to dbalchev/numpy that referenced this issue on Apr 19

`DEP: Remove deprecated numeric style dtype strings (numpy#19539)` ··· `78c6978`

blortuga mentioned this issue on Apr 29

NumPy 1.22 support numba/numba#7754

Closed

5 tasks

taladrane mentioned this issue on May 25

[GHSA-fpfv-jqm9-f5jm] Incomplete string comparison in the numpy.core component... github/advisory-database#287

Closed

rgommers mentioned this issue on Jun 10

ENH: Extend/enhance NEP-29 to cover security fix back-ports. #21713

Open

connortann mentioned this issue on Jul 1

Drop python 3.7 bp/resqpy#526

Merged

kanderso-nrel mentioned this issue on Jul 5

numpy 1.9.X security issue pvlib/pvanalytics#137

Open

bmerry commented on Jul 14 • edited — Contributor

From what I can see the original code was actually *correct*: it's checking whether `dep_tp` is a prefix of `type`, so that (for example) `Uint64` will be flagged as deprecated because it has `Uint` as a prefix:

```
>>> np.dtype("Uint64")
__main__:1: DeprecationWarning: Numeric-style type codes are deprecated and will result in an error in the future.
dtype('uint64')
```

So the proposals to add `+ 1` to the length would actually make the code wrong as it would prevent that DeprecationWarning from appearing. Not that it matters for future versions since the code has been removed, but that might be of interest for any OS distributions planning to "fix" older versions.

Ghostkeeper mentioned this issue on Jul 26

Bump numpy from 1.21.5 to 1.22.0 Ultimaker/Uranium#838

Open

chtompki commented on Aug 16

I'm a tad confused by the lack of support for fixing this CVE? You've kinda left it hanging despite saying that `1.21` is supported until June 23, 2023. https://numpy.org/neps/nep-0029-deprecation_policy.html#drop-schedule

👍 1

mattip commented on Aug 16 — Member

@chtompki could you point to which part of the discussion above is confusing? I think the position of the NumPy team is quite clear. The process for disputing bogus CVEs is not transparent, we do not know why they have not withdrawn it.

👀 1

chtompki commented on Aug 16 • edited

Curious, I've always found the CVE process workable when I've written them. Who is your CVE Numbering Authority (CNA)? Is it the 501(c)(3) NumFOCUS or did you guys go through the request directly from MITRE? If you went directly through MITRE, then you may have to file an appeal: https://www.cve.org/ResourcesSupport/AllResources/CNARules#section_9_appeals_process

Also, let me know if there's anything I can do to help. Would love to, if possible.

seberg commented on Aug 16 — Member

> Curious, I've always found the CVE process workable when I've written them. Who is your CVE Numbering Authority (CNA)?

Well, nobody here has done this professional or even more than once. And maybe a point is that you are going through some CNA where the process surprisingly works easier?

We tried to tell MITR that it this is bogus, explaining why. They just asked for evidence without guidance how that would look like. So it is disputed, but thats it until someone explains how to "proof" that it is not a CVE.

chtompki mentioned this issue on Aug 19

numpy-1.21.6-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl: 1 vulnerabilities (highest severity is: 5.3) capitalone/ablation#5

Closed

chtompki commented on Aug 19

Hm....I would ask who applied for the CVE Number? Can we not go back to them and ask them to un-apply? If not then we need to appeal to MITRE through their process. I'd be happy to help with that if possible

rgommers commented on Aug 19 Member

> Hm....I would ask who applied for the CVE Number? Can we not go back to them and ask them to un-apply? If not then we need to appeal to MITRE through their process. I'd be happy to help with that if possible

Probably **@Daybreak2019** who opened this issue. But they seem like a known bad actor, lots of bogus CVEs and no response after that anymore (see #18993 (comment) above).

This is the problem with the whole security circus - no accountability from anyone (neither the CVE submitter nor MITRE), and we get left with this mess.

2

falkoschindler mentioned this issue on Sep 7

Security alert for NumPy<=1.21.6 zauberzeug/nicegui#67

Closed

swryan mentioned this issue on Oct 13

make python 3.7 the oldest in workflow, add ignore-vuln tadkollar/OpenMDAO#159

Merged

jpn-- mentioned this issue 4 days ago

Fix numpy ActivitySim/activitysim#584

Closed

Assignees

No one assigned

Labels

00 - Bug component: numpy.core

Projects

None yet

Milestone

1.22.0 release

Development

No branches or pull requests

13 participants