New issue

Jump to bottom

`torch.jit.annotations.parse_type_line` is not safe (command injection) even it seems already deprecated. #88868

Closed **Lyutoon** opened this issue 22 days ago · 7 comments

Assignees

Labels high priority **oncall: jit** topic: security triaged

Milestone 1.13.1

Lyutoon commented 22 days ago • edited by pytorch-bot bot

Describe the bug

In `torch.jit.annotations`, it looks like there are some functions that are deprecated, but still retain code, which may lead to some backdoors, especially since some of these functions still use eval while implementing.
But now I'm not sure if there are some features (jit decorator) in some version of pytorch are still using this function `parse_type_line` or `get_signature` which calls `parse_type_line`, if so, it can cause RCE, if not, maybe someone can also leave a backdoor by calling this function while writing code and share it to the people.

```
import torch

torch.jit.annotations.parse_type_line('# type: __import__("os").system("ls") -> 234', None, 1)
```

Versions

master

cc **@ezyang @gchanan @zou3519 @EikanWang @jgong5 @wenzhe-nrv @sanchitintel**

malfet added triage review topic: security labels 21 days ago

malfet commented 19 days ago • edited Contributor

Marking for triage review(and not assigning `oncall: jit` yet otherwise it will disappear to the void) to discuss what to do with those kinds of security issues, which, in my opinion, is pretty minor: if one have an access to local Python runtime they can do anything they want.

Lyutoon commented 19 days ago • edited Author

> Marking for triage review(and not assigning `oncall: jit` yet otherwise it will disappear to the void) to discuss what to do with those kinds of security issues, which is in my opinion is pretty minor: if one have an access to local Python runtime they can do anything they want.

Yes, this seems not a very urgent bug, but we still need to pay attention to these dangerouse functions such as `eval`. And in `CVE-2022-0845`, this bug is also caused by using `eval` to parse the args so it leads to code injection, and it seems also must have an access to local python. To be honest, I don't know how people think about these kind of problems, but we need to pay more attention :p. So we need a discuss about it whether it can be considered as a big security problem.

malfet added the high priority label 19 days ago

malfet self-assigned this 19 days ago

malfet commented 19 days ago Contributor

We should have a doc marking unsafe function and safe versions of the same.
And also, probably should not use eval, if possible.

albanD added **oncall: jit** and removed triage review labels 19 days ago

malfet added triage review and removed triage review labels 19 days ago

Lyutoon commented 19 days ago Author

> We should have a doc marking unsafe function and safe versions of the same. And also, probably should not use eval, if possible.

That's right, there was also a cve (https://github.com/tensorflow/tensorflow/blob/master/tensorflow/security/advisory/tfsa-2022-060.md) in tensorflow that used `eval` in `saved_model_cli` and caused code injection. Also I've found that PaddlePaddle has also `eval` problems (https://github.com/PaddlePaddle/Paddle/blob/develop/security/advisory/pdsa-2022-002.md). But sometimes, if we do not use `eval`, the code will become much more complex. Maybe developers can just add a critical check about the parameters of the function to avoid this problem easily (if possible).

malfet added the triaged label 16 days ago

malfet commented 16 days ago — Contributor

It seems reasonable, that valid type annotations should not have any funciton/method calls, so splitting it into `compile` (which is safe)->error on function calls-> `eval` would secure the deprecated codebase without overcomplicating deprecated code too much

1

malfet mentioned this issue 16 days ago

[JIT][Security] Do not blindly eval input string #89189

Closed

pytorchmergebot closed this as completed in `767f6aa` 15 days ago

---

roywei commented 3 days ago

Hi guys, is there any plan to release this patch to torch 1.12.x and 1.13.x? any ETAs? thanks!

1

roywei mentioned this issue 3 days ago

[v.1.13.1] Release Tracker #89855

Open

seemethere commented 3 days ago — Member

> Hi guys, is there any plan to release this patch to torch 1.12.x and 1.13.x? any ETAs? thanks!

We don't currently have any plans to do continued releases for 1.12.x but given the security implications of this this patch *will* be included in the next patch release for 1.13.x

1

seemethere added this to the **1.13.1** milestone 3 days ago

atalman pushed a commit to atalman/pytorch that referenced this issue 3 days ago

`[JIT][Security] Do not blindly eval input string (pytorch#89189)` ··· `78cad99`

atalman mentioned this issue 3 days ago

[JIT][Security] Do not blindly eval input string (#89189) #89925

Merged

malfet added a commit that referenced this issue 3 days ago

`[JIT][Security] Do not blindly eval input string (#89189) (#89925)` ··· ✕ `74a9ca9`

Vatshank mentioned this issue 11 hours ago

[pytorch] [trcomp] Updated binaries with CVE fix for PT.12 SM Training Compiler aws/deep-learning-containers#2478

Merged

18 tasks

Assignees

malfet

Labels

high priority · **oncall: jit** · topic: security · triaged

Projects

None yet

Milestone

1.13.1

Development

Successfully merging a pull request may close this issue.

[JIT][Security] Do not blindly eval input string
pytorch/pytorch

5 participants