Malicious scripts in untrusted directories are executed #122

Closed · saleemrashid opened this issue on Jan 7, 2020 · 8 comments

Labels: enhancement

saleemrashid commented on Jan 7, 2020

```
git clone https://github.com/saleemrashid/evil-zsh-autoswitch-virtualenv.git
cd evil-zsh-autoswitch-virtualenv
cat evil.txt
```

The script `evil_virtualenv/bin/activate` will be sourced **without any user interaction**, which will:

1. Erase the "Switching virtualenv" message from the terminal (so the user isn't even aware that anything has happened)
2. Write the output of `id; ls ~` to a file called `evil.txt`. Obviously this would be more malicious in practice

👀 1

MichaelAquilina commented on Jan 7, 2020 · Owner

Hi **@saleemrashid** this is true. But if I understand this correctly, apart from the automation it is no more or less secure than any virutalenv activation script (Please feel free to correct me if I am wrong here). If there was no automation the user would have activated the corresponding virtualenv manually.

A couple of checks are made to make sure the correct virtualenv target is not tampered with https://github.com/MichaelAquilina/zsh-autoswitch-virtualenv/blob/master/autoswitch_virtualenv.plugin.zsh#L140-L147

We could probably look into making sure the virtualenv files are not writable except by the user too (using chown)

MichaelAquilina changed the title ~~This is incredibly insecure!~~ **Malicious virtualenv activation scripts can be executed** on Jan 7, 2020

MichaelAquilina added the **enhancement** label on Jan 7, 2020

saleemrashid commented on Jan 7, 2020 • edited · Author

@MichaelAquilina The point here is that the user isn't expecting there to be a virtualenv, so they would not have activated it themselves. They are cloning a random Git repository, that they don't necessarily trust, and by simply running `cd evil-zsh-autoswitch-virtualenv`, they have executed malicious code.

This is a **huge** issue, unless you believe that you will never `cd` into a directory that you don't completely trust - have you never cloned a random Git repository or downloaded a random archive that you don't necessarily trust enough to let loose on your system?

This is why, for example, direnv has a security mechanism that requires you to whitelist directories (e.g. run `direnv allow .`) before the `.envrc` is used.

saleemrashid commented on Jan 7, 2020 · Author

I think your threat model is incredibly unusual:

- You check that the `.venv` cannot be modified by other users. I think this is incredibly low risk: it would require the `.venv` to have insecure permissions, as well as all the parent directories.
- You don't check that the `.venv` is not from a random folder you downloaded off the Internet! The threat is very real here - the user might be oblivious that the folder contains a `.venv` and by the time they have changed into the folder to examine it, the malicious script has already executed - this is Game Over.

saleemrashid changed the title ~~Malicious virtualenv activation scripts can be executed~~ **Malicious scripts in untrusted directories are executed** on Jan 7, 2020

MichaelAquilina commented on Jan 7, 2020 • edited · Owner

> **@MichaelAquilina** The point here is that the user isn't expecting there to be a virtualenv, so they would not have activated it themselves. They are cloning a random Git repository, that they don't necessarily trust, and by simply running cd evil-zsh-autoswitch-virtualenv, they have executed malicious code.

Yep that makes it clearer to me. I now realise how the contents of `.venv` are also abused with `/self/proc` to get to the current working directory (which I was not aware you could do). It does require the correct number of parent traversals to be specified (e.g. I was not affected. UPDATE: actually I was), but I agree its an issue that needs fixing asap.

> This is why, for example, direnv has a security mechanism that requires you to whitelist directories (e.g. run direnv allow .) before the .envrc is used.

That's an interesting approach which we can probably streamline without the user. Preventing unallowed chars is probably a huge improvement too (`.` and `/`).

> I think your threat model is incredibly unusual:

I appreciate you are trying to be helpful, but there's no need to take a dramatic tone here. Help me understand the issue instead :)

> You check that the .venv cannot be modified by other users. I think this is incredibly low risk: it would require the .venv to have insecure permissions, as well as all the parent directories.

If you are sharing a machine, then this check prevents anyone from making you execute a malicious virtualenv by allowing them to over write the target (which could be made malicious).

> You don't check that the .venv is not from a random folder you downloaded off the Internet!

I'm curious to see how you would even detect this? A file you downloaded on your system is just a file in the end.

Next time, I would appreciate you consider responsible disclosure and email me directly if you think there is a threat though.

MichaelAquilina added a commit that referenced this issue on Jan 7, 2020

`fix: insecure activation of virtualenvs` … d475692

MichaelAquilina mentioned this issue on Jan 7, 2020

fix: insecure activation of virtualenvs #123

Merged

MichaelAquilina commented on Jan 7, 2020 Owner

@saleemrashid please take a look at the latest release and see what you think. I have a few other ideas I will implement but the issue you specifically posted should be resolved.

MichaelAquilina closed this as completed on Jan 7, 2020

saleemrashid commented on Jan 7, 2020 Author

> Yep that makes it clearer to me. I now realise how the contents of `.venv` are also abused with `/self/proc` to get to the current working directory (which I was not aware you could do). It does require the correct number of parent traversals to be specified (e.g. I was not affected. UPDATE: actually I was), but I agree its an issue that needs fixing asap.

Sorry for the lack of clarity.

For completeness, it is not necessary to use the exact number of path traversals because `/..` is a hard link to `/`, so extra path traversals will have no effect - therefore, using a copious number of them would be sufficient to attack any user.

> > You check that the .venv cannot be modified by other users. I think this is incredibly low risk: it would require the .venv to have insecure permissions, as well as all the parent directories.
>
> If you are sharing a machine, then this check prevents anyone from making you execute a malicious virtualenv by allowing them to over write the target (which could be made malicious).

I understand the concept, but let us imagine that your `.venv` file is at `/home/user/projects/python/.venv`. This would require all of `/home`, `/home/user`, `/home/user/projects`, and `/home/user/projects/python` to have execute permissions, and then for `/home/user/projects/python/.venv` to have writable permissions. You're just as likely to have `/home/user/.zshrc` with writable permissions, and then it's game over anyway.

> Next time, I would appreciate you consider responsible disclosure and email me directly if you think there is a threat though.

This kind of vulnerability is common to all utilities of this nature, so I am confident that anyone who wants to exploit it would spot it without my help. I don't use this plugin, but someone I follow on GitHub happened to star it so I figured I would take a look and see if it had this vulnerability.

Had the situation been different, e.g. you had implemented a security mechanism and I discovered a bypass for it, then I would have reached out to you privately.

saleemrashid commented on Jan 7, 2020 Author

I still think you should consider implementing white-listing. Even if your validation for `.venv` proves to be secure, you're still assuming that `pipenv` is safe when executed with an untrustworthy `Pipfile`, but it definitely doesn't make any such guarantees.

MichaelAquilina commented on Jan 7, 2020 Owner

> I still think you should consider implementing white-listing. Even if your validation for .venv proves to be secure, you're still assuming that pipenv is safe when executed with an untrustworthy Pipfile, but it definitely doesn't make any such guarantees.

Agreed there is more work to be done. Just wanted to fix the main issue for now in order to get it out asap.

Thanks for posting this issue!

Assignees

No one assigned

Labels

enhancement

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants