Securing Python services using systemd

Zbigniew Jędrzejewski-Szmek

zbyszek@in.waw.pl

PyConPL, 5.11.2022

Agenda

- systemd — running services
- python — example service
- systemd — unit protections
- hardening the example service
- hardening the example service more
- passing zecure sekretz to the service
- (pystemd — starting transient services)

systemd units

traditional approach:
`/usr/lib/systemd/system/foobar.service`

alternative approach:
`systemd-run -u foobar.service …`

tonietak.py

github.com/keszybz/.../tonietak.py

Unit protections

- User=
- DynamicUser=yes
- StateDirectory=yes
- WorkingDirectory=yes

Unit protections

- ProtectHome=yes
- ProtectSystem=yes
- NoExecPaths=

Unit protections — more

- ProtectTmp=yes
- PrivateNetworking=yes
- SystemCallFilter=
  `systemd-analyze syscall-filter`

Encrypted Secrets

- `systemd-creds encrypt`
- `LoadCredentialEncrypted=`

Finding ideas for more protection directives

```
systemd-analyze security run-u4020.service
```

Putting the service in production

Running from the command-line is not a deployment strategy.
How to make the unit permanent?

Strategy I: save file

```
systemctl cat run-u3563.service

src=$(systemctl show -P FragmentPath run-u3563.service)
sudo cp "$src" \
    /etc/systemd/system/flask-app-tonietak.service
```

Strategy II: pystemd

```
import pystemd.run

cmd = 'flask --app tonietak run'.split()

properties = dict(
  DynamicUser=True,
  StateDirectory='files',
  WorkingDirectory='/var/lib',
  ProtectHome='yes',
  ProtectSystem='yes',
  # LoadCredentialEncrypted='password.bin',
)

pystemd.run(cmd, extra=properties)
```

Acknowledgments and links

Kushal Das — “Securing Services Using systemd”
`https://www.youtube.com/watch?v=UUW8RO4hkg0`, verybad.rs

`https://flask.palletsprojects.com/en/2.2.x/flask`
(with apologies to all flask developers ;))

systemd.exec(5), systemd.directives(5)

pystemd, pystemd.run

systemd-python-services.pdf (these slides), tonietak.py