Securing your daemons using systemd

Zbigniew Jędrzejewski-Szmek

zbyszek@in.waw.pl

Brno, 2020.1.26

Before we begin...
Why use systemd for this at all?

- centralization
- abstraction of hardware architecture / kernel version
- unprivileged operation

Before we begin...
Why use systemd for this at all?

- centralization
- abstraction of hardware architecture / kernel version
- unprivileged operation

Numbers:

```
$ dnf repoquery --releasever=32 -l --whatprovides \
    '/usr/lib/systemd/system/*' \
    rg -i '^/usr/lib/systemd/system/[a-z0-9_@.\\-]+$' | \
    sort -u | wc -l
1740!!!
```

Before we begin...
Unit files

Before we begin...
Unit files

```
# /etc/systemd/system/mydaemon.service
[Service]
ExecStart=/usr/local/bin/mydaemon
```

```
$ sudo systemctl start mydaemon.service
```

Before we begin...
Unit files

```
# /etc/systemd/system/mydaemon.service
[Service]
ExecStart=/usr/local/bin/mydaemon
```

```
$ sudo systemctl start mydaemon.service
```

```
$ sudo /usr/local/bin/mydameon
```

Before we begin...
Unit files

```
# /etc/systemd/system/mydaemon.service
[Service]
ExecStart=/usr/local/bin/mydaemon
```

```
$ sudo systemctl start mydaemon.service

$ sudo /usr/local/bin/mydameon

$ sudo systemd-run /usr/local/bin/mydameon
$ sudo systemd-run -t /usr/local/bin/mydameon
```

Basics

```
User=
```

Basics

User=

```
$ systemd-run whoami
root

$ systemd-run -p User=zbyszek whoami
zbyszek
$ systemd-run --uid=zbyszek whoami
zbyszek
```

Limiting access to the file system

- `ProtectHome=yes|read-only`
- `ProtectSystem=yes|full|strict`

Limiting access to the file system

- `ProtectHome=yes|read-only`
- `ProtectSystem=yes|full|strict`
- `InaccessiblePaths=`
- `ReadOnlyPaths=`
- `ReadWritePaths=`

Limiting access to the file system

- `ProtectHome=yes|read-only`
- `ProtectSystem=yes|full|strict`
- `InaccessiblePaths=`
- `ReadOnlyPaths=`
- `ReadWritePaths=`
- `BindPaths=`
- `ReadOnlyBindPaths=`

Limiting access to the file system

- `PrivateTmp=yes`

Limiting access to the file system a better way

Limiting access to the file system
a better way

- `RuntimeDirectory=`*foo* — /run/*foo*/
- `StateDirectory=`*foo* — /var/lib/*foo*/
- `CacheDirectory=`*foo* — /var/cache/*foo*/
- `LogsDirectory=`*foo* — /var/log/*foo*/
- `ConfigurationDirectory=`*foo* — /etc/*foo*/

```
$ sudo systemd-run -t -p User=zbyszek \
                      -p RuntimeDirectory=foo \
                      ls -ld /run/foo
```

- automatic *creation* and *ownership*
- automatic *removal*

User creation on demand?

User creation on demand?

- `DynamicUser=yes`

User creation on demand?

- `DynamicUser=yes`

```
$ systemd-run -p DynamicUser=1 -t whoami
```

User creation on demand?

- `DynamicUser=yes`

```
$ systemd-run -p DynamicUser=1 -t whoami
$ echo -e 'asdf\nasdf' | \
```

User creation on demand?

- `DynamicUser=yes`

```
$ systemd-run -p DynamicUser=1 -t whoami
$ echo -e 'asdf\nasdf' | \
  systemd-run --pipe -p DynamicUser=1 \
    bash -c 'grep .; whoami' | \
```

User creation on demand?

- `DynamicUser=yes`

```
$ systemd-run -p DynamicUser=1 -t whoami
$ echo -e 'asdf\nasdf' | \
  systemd-run --pipe -p DynamicUser=1 \
    bash -c 'grep .; whoami' | \
  systemd-run --pipe -p DynamicUser=1 \
    bash -c 'grep .; whoami' | \
```

User creation on demand?

- `DynamicUser=yes`

```
$ systemd-run -p DynamicUser=1 -t whoami
$ echo -e 'asdf\nasdf' | \
  systemd-run --pipe -p DynamicUser=1 \
    bash -c 'grep .; whoami' | \
  systemd-run --pipe -p DynamicUser=1 \
    bash -c 'grep .; whoami' | \
  systemd-run --pipe -p DynamicUser=1 \
    bash -c 'grep .; whoami'
```

What about the network?

What about the network?

- `PrivateNetwork=yes`

What about the network?

- `PrivateNetwork=yes`

"`PrivateNetwork=yes` is the recommeded way to run network services"

Socket Activation

A daemon does not open a socket itself, it receives a socket from the manager

Socket Activation

A daemon does not open a socket itself, it receives a socket from the manager

Two types of socket activation:

`Accept=yes`
→ a single instance of the service is started for each connection
→ "wait" under inetd/xinetd

`Accept=no`
→ a single instance of the service is started for each connection
→ "nowait" under inetd/xinetd

Low level stuff

- MemoryDenyWriteExecute=yes
- PrivateDevices=yes
- NoNewPrivileges=yes
- RestrictAddressFamilies=AF_UNIX|AF_INET|AF_INET6 |AF_CAN|AF_APPLETALK|...
- RestrictSUIDSGID=yes
- ProtectKernelTunables=yes
- ProtectHostname=yes
- ProtectKernelLogs=yes
- SystemCallArchitectures=native|x86_64|i386|...
- LockPersonality=yes

Capability limits

- `CapabilityBoundingSet=`
- `Capability=`
- `DropCapability=`
- `AmbientCapabilities=`

System call filtering

“seccomp mode 2”

System call filtering

“seccomp mode 2”

- `SyscallFilter=...`
  implemented using `libseccomp`
- `syscall1 | syscall2 | @group`
- `@basic-io`
- `@obsolete`

System call filtering

“seccomp mode 2”

- `SyscallFilter=...`
  implemented using `libseccomp`
- `syscall1 | syscall2 | @group`
- `@basic-io`
- `@obsolete`

```
$ systemd-analyze syscall-filter @obsolete
```

Per-service network firewall

- `IPAddressAllow=10.20.30.0/24 1.2.3.4`
- `IPAddressDeny=*`
- `IP{Ingress,Egress}FilterPath=`

systemd-analyze security

```
$ systemd-analyze security systemd-resolved.service
```

The End

https://github.com/systemd/systemd
docs: https://systemd.io/
https://www.freedesktop.org/wiki/Software/systemd/

this:
https://github.com/keszybz/systemd-security-talk/blob/master/systemd-security.pdf