Securing and Sandboxing ConfD using Systemd

tail-f a Cisco company

Table of Contents

Securing and Sandboxing ConfD using Systemd

1. Overview

In this application note, we will explore how we can sandbox ConfD using security and sandboxing capabilities enabled by systemd.

Starting with version 7.3, the ConfD User Guide contains a section describing how to increase the security of ConfD deployments as well as a launch options to verify that confd.conf and other resources under ConfD's direct control don't contain any glaring security issues. This application note describes how to use the systemd init daemon and namespaces, both features available in most modern Linux distributions, to further enhance security and robustness of ConfD deployments.

Systemd provides a significant number of security features that can be used to isolate services and applications from each other as well as from the underlying operating system. In many cases, systemd provides easy access to the same mechanisms provided by the Linux kernel that are also used to create isolation for Linux containers. Having the ability to provide container-style isolation for traditional applications and services is powerful because it's now easy to improve the security and isolation of workloads without the operational impact that containers require. It's worth noting that the operational and organizational changes inspired by container adoption are indeed healthy and worthwhile. However, even in the most container-savvy enterprise, there are large numbers of traditional Linux deployments where security is a top priority. As we'll see, the workloads on these systems can benefit from just a few tweaks to the corresponding system service configuration files which are called unit files.

In this application note, we will show how to use these mechanisms to improve the security of ConfD deployments without any loss of functionality. If the ConfD process is ever compromised once these options are active, the potential for a breakout and ensuing damage to the rest of the system is drastically reduced.

We cannot blindly turn on everything. For example, there are settings to completely disallow writing anywhere in the filesystem or any network access which would certainly make for a very secure service but would also prevent ConfD from doing any of the tasks which are actually the whole point of running ConfD.

2. Running ConfD Under Systemd

For this application note, we rely on a ConfD target installation as described in the ConfD User Guide section "Installing ConfD on a Target System". ConfD is installed in the /opt/confd directory and owned by the confd user.

The ConfD basic service unit configuration is shown below.

```
[Unit]
Description=ConfD configuration daemon
After=network.target

[Service]
Environment=CONFD_DIR=/opt/confd
Environment=LD_LIBRARY_PATH=/opt/confd/lib
Environment=PATH=/opt/confd/bin:/usr/local/opt/python/
libexec/bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
Environment=PYTHONPATH=/opt/confd/src/confd/pyapi
ExecStart=/opt/confd/bin/confd -c $APP_NOTE_ROOT/confd.conf
--addloadpath /opt/confd/etc/confd
ExecStop=/opt/confd/bin/confd --stop
Restart=on-failure
Type=forking

[Install]
WantedBy=multi-user.target
```

We use the "Environment=" statement to define the environment variables typically defined by $CONFD_DIR/confdrc. "ExecStart=" and "ExecStop=" declare the commands systemd will use to start and stop the service. "Restart=" tells systemd to automatically restart ConfD if the service fails. Note that it makes sense to comment out this statement while developing the unit file because it is easy to make a mistake that results in the service cyclically restarting. Finally, the "Type=" statement tells systemd that the service calls fork() during startup.

In this application note, we will iteratively add statements to lock down the ConfD daemon.

Copy the service unit description to the /etc/systemd/system directory.

"Systemd-analyze" is a pretty cool systemd tool which can analyze and debug the service manager. When passed the security argument, it analyzes the security and sandbox settings of a service unit, reports the current settings, assigns an exposure score to each setting, and finishes with a verdict. We will make use of systemd-analyze throughout this app note to evaluate the effects of the changes we make.

Note though, that systemd-analyze doesn't catch everything. Many of the options we use

imply other options and these aren't caught when we run systemd-analyze. For example, if "SystemCallFilter=" is set, "NoNewPriviliges=" is also set but systemd-analyze doesn't know this and won't account for "NoNewPriviliges=" unless set explicitly.

When run systemd-analyze with the service unit file listed above, we get an initial result:

```
$ systemd-analyze security confd.service
 NAME                                                    DESCRIPTION                                                     EXPOSURE
✗ PrivateNetwork=                                   Service has access to the host's network                      0.5
✗ User=/DynamicUser=                                Service runs as root user                                     0.4
✗ CapabilityBoundingSet=~CAP_SET(UID|GID|PCAP)      Service may change UID/GID identities/capabilities            0.3
✗ CapabilityBoundingSet=~CAP_SYS_ADMIN              Service has administrator privileges                          0.3
✗ CapabilityBoundingSet=~CAP_SYS_PTRACE             Service has ptrace() debugging abilities                      0.3
✗ RestrictAddressFamilies=~AF_(INET|INET6)          Service may allocate Internet sockets                         0.3
✗ RestrictNamespaces=~CLONE_NEWUSER                 Service may create user namespaces                            0.3
✗ RestrictAddressFamilies=~…                        Service may allocate exotic sockets                           0.3
✗ CapabilityBoundingSet=~CAP_(CHOWN|FSETID|SETFCAP)     Service may change file ownership/access mode/capabilities unrestricted     0.2
✗ CapabilityBoundingSet=~CAP_(DAC_*|FOWNER|IPC_OWNER)   Service may override UNIX file/IPC permission checks      0.2
✗ CapabilityBoundingSet=~CAP_NET_ADMIN              Service has network configuration privileges                  0.2
✗ CapabilityBoundingSet=~CAP_RAWIO                  Service has raw I/O access                                    0.2
✗ CapabilityBoundingSet=~CAP_SYS_MODULE             Service may load kernel modules                               0.2
✗ CapabilityBoundingSet=~CAP_SYS_TIME               Service processes may change the system clock                 0.2
✗ DeviceAllow=                                      Service has no device ACL                                     0.2
✗ IPAddressDeny=                                    Service does not define an IP address whitelist               0.2
✓ KeyringMode=                                      Service doesn't share key material with other services
✗ NoNewPrivileges=                                  Service processes may acquire new privileges                  0.2
✓ NotifyAccess=                                     Service child processes cannot alter service state
✗ PrivateDevices=                                   Service potentially has access to hardware devices            0.2
✗ PrivateMounts=                                    Service may install system mounts                             0.2
✗ PrivateTmp=                                       Service has access to other software's temporary files        0.2
✗ PrivateUsers=                                     Service has access to other users                             0.2
✗ ProtectClock=                                     Service may write to the hardware clock or system clock       0.2
✗ ProtectControlGroups=                             Service may modify the control group file system              0.2
✗ ProtectHome=                                      Service has full access to home directories                   0.2
✗ ProtectKernelLogs=                                Service may read from or write to the kernel log ring buffer  0.2
✗ ProtectKernelModules=                             Service may load or read kernel modules                       0.2
✗ ProtectKernelTunables=                            Service may alter kernel tunables                             0.2
✗ ProtectSystem=                                    Service has full access to the OS file hierarchy              0.2
✗ RestrictAddressFamilies=~AF_PACKET                Service may allocate packet sockets                           0.2
✗ RestrictSUIDSGID=                                 Service may create SUID/SGID files                            0.2
✗ SystemCallArchitectures=                          Service may execute system calls with all ABIs                0.2
✗ SystemCallFilter=~@clock                          Service does not filter system calls                          0.2
✗ SystemCallFilter=~@debug                          Service does not filter system calls                          0.2
✗ SystemCallFilter=~@module                         Service does not filter system calls                          0.2
✗ SystemCallFilter=~@mount                          Service does not filter system calls                          0.2
✗ SystemCallFilter=~@raw-io                         Service does not filter system calls                          0.2
✗ SystemCallFilter=~@reboot                         Service does not filter system calls                          0.2
✗ SystemCallFilter=~@swap                           Service does not filter system calls                          0.2
✗ SystemCallFilter=~@privileged                     Service does not filter system calls                          0.2
✗ SystemCallFilter=~@resources                      Service does not filter system calls                          0.2
✓ AmbientCapabilities=                              Service process does not receive ambient capabilities
✗ CapabilityBoundingSet=~CAP_AUDIT_*                Service has audit subsystem access                            0.1
✗ CapabilityBoundingSet=~CAP_KILL                   Service may send UNIX signals to arbitrary processes          0.1
✗ CapabilityBoundingSet=~CAP_MKNOD                  Service may create device nodes                               0.1
✗ CapabilityBoundingSet=~CAP_NET_(BIND_SERVICE|BROADCAST|RAW) Service has elevated networking privileges          0.1
✗ CapabilityBoundingSet=~CAP_SYSLOG                 Service has access to kernel logging                          0.1
✗ CapabilityBoundingSet=~CAP_SYS_(NICE|RESOURCE)    Service has privileges to change resource use parameters      0.1
✗ RestrictNamespaces=~CLONE_NEWCGROUP               Service may create cgroup namespaces                          0.1
✗ RestrictNamespaces=~CLONE_NEWIPC                  Service may create IPC namespaces                             0.1
✗ RestrictNamespaces=~CLONE_NEWNET              Service may create network namespaces                      0.1
✗ RestrictNamespaces=~CLONE_NEWNS               Service may create file system namespaces                  0.1
✗ RestrictNamespaces=~CLONE_NEWPID              Service may create process namespaces                      0.1
✗ RestrictRealtime=                         Service may acquire realtime scheduling                    0.1
✗ SystemCallFilter=~@cpu-emulation              Service does not filter system calls                   0.1
✗ SystemCallFilter=~@obsolete                 Service does not filter system calls                   0.1
✗ RestrictAddressFamilies=~AF_NETLINK           Service may allocate netlink sockets                       0.1
✗ RootDirectory=/RootImage=                   Service runs within the host's root directory               0.1
✗ SupplementaryGroups=                        Service runs as root, option does not matter
✗ CapabilityBoundingSet=~CAP_MAC_*              Service may adjust SMACK MAC                               0.1
✗ CapabilityBoundingSet=~CAP_SYS_BOOT           Service may issue reboot()                                 0.1
✓ Delegate=                                 Service does not maintain its own delegated control group subtree
✗ LockPersonality=                          Service may change ABI personality                     0.1
✗ MemoryDenyWriteExecute=                     Service may create writable executable memory mappings           0.1
✗ RemoveIPC=                                Service runs as root, option does not apply
✗ RestrictNamespaces=~CLONE_NEWUTS              Service may create hostname namespaces                      0.1
✗ UMask=                                    Files created by service are world-readable by default     0.1
✗ CapabilityBoundingSet=~CAP_LINUX_IMMUTABLE        Service may mark files immutable                           0.1
✗ CapabilityBoundingSet=~CAP_IPC_LOCK             Service may lock memory into RAM                           0.1
✗ CapabilityBoundingSet=~CAP_SYS_CHROOT           Service may issue chroot()                                 0.1
✗ ProtectHostname=                          Service may change system host/domainname                  0.1
✗ CapabilityBoundingSet=~CAP_BLOCK_SUSPEND        Service may establish wake locks                           0.1
✗ CapabilityBoundingSet=~CAP_LEASE              Service may create file leases                             0.1
✗ CapabilityBoundingSet=~CAP_SYS_PACCT            Service may use acct()                                     0.1
✗ CapabilityBoundingSet=~CAP_SYS_TTY_CONFIG        Service may issue vhangup()                                0.1
✗ CapabilityBoundingSet=~CAP_WAKE_ALARM           Service may program timers that wake up the system                 0.1
✗ RestrictAddressFamilies=~AF_UNIX              Service may allocate local sockets                         0.1

→ Overall exposure level for confd.service: 9.6 UNSAFE 😳
```

Hmm, this does not look too great. It's a little bit hard to see but systemd-analyze tests on the order of 80 different settings and we only get three green ticks for a final sum of all exposure scores of 9.6 (the exposure score falls in the 0.0 to 10.0 range) and an "UNSAFE" verdict. Our goal is to improve this score and achieve an "OK" verdict.

It's important to realize that systemd-analyze only examines the per-service security features implemented by systemd and that any security mechanisms implemented by the service itself are not accounted for. A high exposure score means that systemd applies fewer security measures, not that the service necessarily is vulnerable to remote or local exploits.

3. Run ConfD as a Non-Root User

Let’s tell systemd to run ConfD as a regular user rather than root. Add these two lines to the service unit file:

```
User=confd
Group=confd
```

Save and tell systemd to reload our service unit file "sudo systemctl daemon-reload".

If we run "systemd-analyze security confd.service" again we still get an "UNSAFE" rating although the score has improved to 9.2. It’s better, but not by a lot. Furthermore, ConfD doesn’t start anymore, we get the following error:

```
Sep 03 14:26:01 lab confd[41526]: - Starting to listen for
NETCONF SSH on 127.0.0.1:830
Sep 03 14:26:01 lab confd[41526]: - Cannot bind to NETCONF
socket 127.0.0.1:830 : permission denied
Sep 03 14:26:01 lab confd[41522]: Cannot bind to NETCONF
socket 127.0.0.1:830 : permission denied
Sep 03 14:26:01 lab confd[41522]: Daemon died status=20
Sep 03 14:26:01 lab systemd[1]: confd.service: Control
process exited, code=exited, status=20/n/a
Sep 03 14:26:01 lab systemd[1]: confd.service: Failed with
result 'exit-code'.
Sep 03 14:26:01 lab systemd[1]: Failed to start ConfD
configuration daemon.
```

This makes sense. Since we’re not running as root, we can’t open privileged ports.

Fortunately, Linux provides a way to assign capabilities they normally won’t to processes. A process that has the CAP_NET_BIND_SERVICE capability set can open privileged ports.

```
AmbientCapabilities=CAP_NET_BIND_SERVICE
CapabilityBoundingSet=CAP_NET_BIND_SERVICE
```

The "AmbientCapabilities=" option enables capabilities and the "CapabilityBoundingSet=" option limits the allowed set that may be assigned to the process. In this example, we add CAP_NET_BIND_SERVICE.

Analyzing the security settings again gives the score 6.8 and the verdict "MEDIUM"; we are heading in the right direction. Even better, this time ConfD starts without any issues.

```
$ sudo systemctl start confd.service
$ systemctl is-active confd.service
active
$
```

4. Protecting the Filesystem

The next thing we want to do is to protect the file system, i.e. lock down the file system so that even if someone manages to compromise the running ConfD process they won't be able to do anything because everything except for a few directories needed for CDB, log files, etc. is mounted as read-only.

```
ProtectSystem=strict
ReadOnlyPaths=/opt/confd/etc/confd
ReadWritePaths=/opt/confd/var/confd/candidate /opt/confd/
var/confd/cdb /opt/confd/var/confd/log /opt/confd/var/confd/
rollback /opt/confd/var/confd/state
```

With "ProtectSystem=strict", the entire file system hierarchy is mounted read-only, except for the API file system subtrees /dev, /proc, and /sys. ConfD has no business touching any of these directories. So, we set "PrivateDevices=", "ProtectControlGroups=" and "ProtectKernelTunables=" too.

```
PrivateDevices=yes
ProtectControlGroups=yes
ProtectKernelTunables=yes
```

Related to these settings are "ProtectKernelModules=" and "ProtectKernelLogs=" which, when set, prevent the service from messing with kernel modules and kernel logs. (regular user space syslog-ing is still allowed).

```
ProtectKernelLogs=yes
ProtectKernelModules=yes
```

With all these settings enabled our score is down to 5.4, still with a "MEDUIM" rating but a huge improvement.

5. Limit Network Access

Restrict Access to only the address families needed. In this example, we only allow IPv4. In actual practice, you may also want to support IPv6. More importantly is that setting this exclude access to "exotic" address families such as AF_PACKET.

```
RestrictAddressFamilies=AF_INET
```

Additionally, if possible, limit access to only well-known IP-addresses or IP-address ranges. Remember that systemd shuts down ConfD over the loopback interface so at the very least we have to leave 127.0.0.1 in the set of allowed interfaces. The backend instrumentation code usually talks to ConfD over loopback as well. So, it has to be open anyways. In this example we know that northbound clients only connect from the 172.28.128.0/24 network and that all southbound clients are local. Hence, we only allow these addresses and block everything else.

```
IPAddressAllow=127.0.0.1 172.28.128.0/24
IPAddressDeny=any
```

Things are getting more interesting now. let's make sure we still can connect from a remote node using NETCONF.

```
$ ifconfig vboxnet1
vboxnet1: flags=8842<BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500
  ether 0a:00:27:00:00:01
$ netconf-console --port 830 --host 172.28.128.56 --hello
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
    …
    <capability>urn:ietf:params:xml:ns:yang:ietf-yang-
metadata?module=ietf-yang-metadata&amp;revision=2016-08-05</
capability>
    <capability>urn:ietf:params:xml:ns:yang:ietf-yang-
types?module=ietf-yang-types&amp;revision=2013-07-15</
capability>
  </capabilities>
  <session-id>18</session-id>
</hello>
$
```

Running "systemd-analyze --no-pager security confd.service" one more time brings the score to 4.8 and the verdict to "OK". We are getting somewhere.

6. Restrict System Calls

The "SystemCallFilter=" option gives us the ability to block or allow individual system calls or groups of system calls. In general, explicitly allowing the system calls used by a service is preferred over blocking the ones not used. As the number of possible system calls is large, predefined sets of system calls are provided. We will set "SystemCallFilter=" to "@ system-service", a reasonable set of system calls used by common system services.

If the system supports multiple ABIs such as 32 and 64-bit x86, it is also recommended to turn off alternative ABIs for services so they cannot be used to circumvent restrictions enforced by the system call filter. Specifically, it is recommended to combine this option with "SystemCallArchitectures=native" or similar.

```
SystemCallArchitectures=native
SystemCallFilter=@system-service
SystemCallErrorNumber=EPERM
```

The final setting specifies what error code should be returned when a system call is filtered.

At this point "systemd-analyze --no-pager security confd.service" returns a score of 3.3 while the verdict is still "OK".

7. Odds and Ends

"NoNewPrivileges=" ensures that the service process and all its children can never gain new privileges through execve() (e.g. via setuid or setgid bits, or filesystem capabilities). This is the simplest and most effective way to ensure that a process and its children can never elevate privileges again. It should be noted that "NoNewPrivileges=yes" is implied by among other settings "PrivateDevices=", "ProtectKernelTunables=", and a number of other options enabled in this application note.

```
NoNewPrivileges=yes
```

When "ProtectClock=" is true, writes to the hardware clock or system clock will be denied. It is recommended to turn this on for most services that do not need modify the clock. When "ProtectHome=" is set, the directories /home, /root, and /run/user are made inaccessible and empty for processes invoked by this unit. It is recommended to enable this setting for all long-running services (in particular network-facing ones) to ensure they cannot get access to private user data, unless the services actually requires access to the user's private data. When "ProtectHostname=" is set, it sets up a new UTS namespace for the executed processes. In addition, changing hostname or domainname is prevented.

```
ProtectClock=yes
ProtectHome=yes
ProtectHostname=yes
```

"RestrictNamespaces=" restricts access to Linux namespace functionality for the processes of this unit. If set, "RestrictRealtime=" refuses any attempts to enable realtime scheduling in a process of the unit. This restricts access to realtime task scheduling policies such as SCHED_FIFO, SCHED_RR, or SCHED_DEADLINE. Realtime scheduling

policies may be used to monopolize CPU time for longer periods of time and may hence be used to lock up or otherwise trigger Denial-of-Service situations on the system. It is hence recommended to restrict access to realtime scheduling to the few programs that actually require them.

```
RestrictNamespaces=yes
RestrictRealtime=yes
RestrictSUIDSGID=yes
```

Saving confd.service and running "systemd-analyze --no-pager security confd.service" one last time brings the score down to 2.0

8. Conclusion

In this application note, it was shown how to protect ConfD and the rest of the system without any loss of functionality other than limiting network access to a set of approved address ranges. We've picked most of the low hanging fruit but there are a few more options that can be applied. We'll leave this as an exercise for the reader.

9. For More Information

For more information about ConfD, visit https://www.tail-f.com

To learn more about systemd in general and sandboxing in particular, see the relevant man-pages systemd(1), systemctl(1), systemd-analyze(1), journalctl(1) and, in particular, systemd.service(5), systemd.exec(5), systemd.resource-control(5).

Appendix A:

Summary of Sandbox and Security Options

The table below describes the systemd sandbox/security options used in this application note.

Option	Description	Comment
AmbientCapabilities=	Controls which capabilities to include in the ambient capability set for the executed process. Takes a whitespace-separated list of capability names	Ambient capability sets are useful if you want to execute a process as a non-privileged user but still want to give it some capabilities. Specifically, if we run ConfD as a non-root user and use the standard NETCONF port we much add CAP_NET_BIND_SERVICE to the AmbientCapabilities set.
CapabilityBoundingSet=	Allowed and forbidden privileged capabilities for the unit. Capabilities prefixed with ~ are disallowed, without are allowed.	
IPAddressAllow= IPAddressDeny=	Turn on address range network traffic filtering for IP packets sent and received over AF_INET and AF_INET6 sockets.	It makes sense to turn on address filtering if the address ranges used by clients are known ahead of time.
NoNewPriveleges=	If true, ensures that the service process and all its children can never gain new privileges through execve() (e.g. via setuid or setgid bits, or filesystem capabilities).	This is the simplest and most effective way to ensure that a process and its children can never elevate privileges again.
PrivateDevices=	If true, sets up a new /dev mount for the executed processes and only adds API pseudo devices such as /dev/null, /dev/zero or /dev/random (as well as the pseudo TTY subsystem) to it, but no physical devices such as /dev/sda, system memory /dev/mem, system ports /dev/port and others. (PrivateDevices=).	This is useful to securely turn off physical device access by the executed process.

PrivateNetwork=	Takes a boolean argument. If true, sets up a new network namespace for the executed processes and configures only the loopback network device "lo" inside it. No other network devices will be available to the executed process.	In general, not possible ConfD since NB clients (and sometimes SB too) must be able to reach us over the network. However, if ConfD is configured to use external ssh/http(s)/... servers we can configure ConfD itself to only see the loopback interface.
PrivateTmp=	If true, sets up a new file system namespace for the executed processes and mounts private /tmp and /var/tmp directories inside it that is not shared by processes outside of the namespace.	Redundant. ConfD doesn't write to /tmp or /var/tmp. On the other hand, it doesn't hurt.
ProtectClock=	Writes to the hardware clock or system clock will be denied.	It is recommended to turn this on for most services that do not need modify the clock.
ProtectHome=	If true, the directories /home, /root, and /run/user are made inaccessible and empty for processes invoked by this unit. If set to "read-only", the three directories are made read-only instead. If set to "tmpfs", temporary file systems are mounted on the three directories in read-only mode.	It is recommended to enable this setting for all long-running services (in particular network-facing ones), to ensure they cannot get access to private user data, unless the services actually require access to the user's private data. This setting is implied if DynamicUser= is set.
ProtectSystem=	Mounts the /usr and /boot directories read-only for processes invoked by this unit. If set to "full", the /etc directory is mounted read-only, too. If set to "strict" the entire file system hierarchy is mounted read-only, except for the API file system subtrees /dev, /proc and /sys (protect these directories using PrivateDevices=, ProtectKernelTunables=, ProtectControlGroups=).	This setting ensures that any modification of the vendor-supplied operating system (and optionally its configuration, and local mounts) is prohibited for the service. It is recommended to enable this setting for all long-running services, unless they are involved with system updates or need to modify the operating system in other ways. If this option is used, ReadWritePaths= may be used to exclude specific directories from being made read-only. This setting is implied if DynamicUser= is set. This setting cannot ensure protection in all cases. We should use ProtectSystem=strict with ConfD.

ProtectControlGroups=	If true, the Linux Control Groups (cgroups(7)) hierarchies accessible through /sys/fs/cgroup will be made read-only to all processes of the unit.	Except for container managers no services should require write access to the control groups hierarchies; it is hence recommended to turn this on for most services.
ProtectKernelLogs=	Prevent access to the kernel log ring buffer.	Not to be confused with the userspace logging, which is obviously still allowed.
ProtectKernelModules=	If true, explicit module loading will be denied. This allows module load and unload operations to be turned off on modular kernels.	It is recommended to turn this on for most services that do not need special file systems or extra kernel modules to work.
ProtectKernelTunables=	If true, kernel variables accessible through /proc/sys, /sys, /proc/sysrq-trigger, /proc/latency_stats, /proc/acpi, /proc/timer_stats, /proc/fs and /proc/irq will be made read-only to all processes of the unit.	Usually, tunable kernel variables should be initialized only at boot-time, for example with the sysctl.d(5) mechanism. Few services need to write to these at runtime; hence, it is recommended to turn this on for most services. (ProtectKernelTunables=)
ReadWritePaths=, ReadOnlyPaths=, InaccessiblePaths=	Sets up a new file system namespace for executed processes. These options may be used to limit access a process might have to the file system hierarchy. Each setting takes a space-separated list of paths relative to the host's root directory (i.e. the system running the service manager).	Note that the effect of these settings may be undone by privileged processes. In order to set up an effective sandboxed environment for a unit it is thus recommended to combine these settings with either CapabilityBoundingSet=~CAP_SYS_ADMN or SystemCallFilter=~@mount.
RestrictAddressFamilies=	Restricts the set of socket address families accessible to the processes of this unit. Takes a space-separated list of address family names to allowlist, such as AF_UNIX, AF_INET or AF_INET6.	Use this option to limit exposure of processes to remote access, in particular via exotic and sensitive network protocols, such as AF_PACKET.

RestrictRealtime=	If set, any attempts to enable realtime scheduling in a process of the unit are refused.	Realtime scheduling policies may be used to monopolize CPU time for longer periods of time and may be used to lock up or otherwise trigger Denial-of-Service situations on the system. It is hence recommended to restrict access to realtime scheduling to the few programs that actually require them.
RestrictSUIDSGID=	If set, any attempts to set the set-user-ID (SUID) or set-group-ID (SGID) bits on files or directories will be denied	As the SUID/SGID bits are mechanisms to elevate privileges and allows users to acquire the identity of other users, it is recommended to restrict creation of SUID/SGID files to the few programs that actually require them.
SystemCallFilter=	Allow-lists and block-lists of individual system calls or groups of system calls. Provides the same functionality as seccomp filtering with containers.	@system-service service is a good starting point.

a Cisco company

www.tail-f.com
info@tail-f.com

Corporate Headquarters

Sveavagen 25
111 34 Stockholm
Sweden
+46 8 21 37 40