Kannel - Bug #771

Kannel should create its PID file before dropping privileges

08/29/2017 12:08 AM - Michael Orlitzky

Status:	Assigned	**Start date:**	08/28/2017
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
Affected version:			

Description

I tested this with bearerbox, but smsbox/wapbox might have the same problem.

Summary

The kannel daemons should create their PID files before dropping
privileges. This represents a minor security issue; additional factors
are needed to make it exploitable.

Details

The purpose of the PID file is to hold the PID of the running daemon,
so that later it can be stopped, restarted, or otherwise signalled
(many daemons reload their configurations in response to a SIGHUP).
To fulfil that purpose, the contents of the PID file need to be
trustworthy. If the PID file is writable by a non-root user, then he
can replace its contents with the PID of a root process. Afterwards,
any attempt to signal the PID contained in the PID file will instead
signal a root process chosen by the non-root user (a vulnerability).

This is commonly exploitable through init scripts that are run as root
and which blindly trust the contents of their PID files. Kannel itself
may ship such an init script; if Debian's start-stop-daemon behaves the
same way that the OpenRC one currently does, then the debian/*.init files
in the Kannel repository are vulnerable to this.

Exploitation

There is only a risk of exploitation when some other user relies on
the data in the PID file.

An example of a problematic scenario involving an init script would be,

1. I run "/etc/init.d/bearerbox start" to start the daemon.

2. bearerbox drops to the "kannel" user.

3. bearerbox writes its PID file, now owned by the "kannel" user.

4. Someone compromises the daemon.

5. The attacker is generally limited in what he can do because the
daemon doesn't run as root. However, he can write "1" into the
PID file, and he does.

6. I run "/etc/init.d/bearerbox stop" to stop the daemon while I investigate.

7. The machine reboots, because I killed PID 1 (this is normally restricted
to root).

History

#1 - 09/20/2017 10:18 PM - Michael Orlitzky

This has been assigned CVE-2017-14609.

#2 - 09/21/2017 10:55 AM - Alexander Malysh

- Status changed from New to Assigned

Hi Michael,

first of all thanks for the report.

I don't see this as a issue of Kannel itself but rather issue of the init script author.
Why init script killing some process based only on PID file? This is just wrong, because
Kannel may crash with segfault and process is not there anymore, etc. etc.
Therefore init script has to check if this is the process that it wants to kill.

As a second thing. If Kannel would write pid files as root , how Kannel should delete it while shutdown without root permissions?

Therefore I see this as not a bug.

#3 - 09/21/2017 01:46 PM - Michael Orlitzky

Just as a practical matter, if you look for init scripts that verify the contents of their PID files, you won't find any. The debian init scripts that ship with Kannel don't do it:

- https://github.com/markjeee/kannel/blob/master/debian/kannel-cvs.init
- https://github.com/markjeee/kannel/blob/master/debian/kannel-devel.init
- https://github.com/markjeee/kannel/blob/master/debian/kannel.init

The Gentoo init scripts don't do it:

- https://gitweb.gentoo.org/repo/gentoo.git/tree/app-mobilephone/kannel/files/kannel-bearerbox.initd
- https://gitweb.gentoo.org/repo/gentoo.git/tree/app-mobilephone/kannel/files/kannel-smsbox.initd
- https://gitweb.gentoo.org/repo/gentoo.git/tree/app-mobilephone/kannel/files/kannel-wapbox.initd

Even if you look outside of Kannel, I don't think you'll find any examples of init scripts that successfully verify the contents of their PID files.

Nevertheless, you might say that it's the responsibility of the init script writer to do so. Can it even be done? I've never seen an example of a POSIX shell function that does it. The best you can do is ensure that the user and process name are what you expect, but that still lets someone who controls the process prevent you from killing it by writing junk into the PID file. And to achieve that no-so-great result, every init script on every distro (three of them for Kannel!) needs to be modified to verify the PID data before signaling a process. Even if on principle this should be the job of the init script, everything would be cleaner overall if the init systems could trust the PID file and just kill its contents.

Not being able to remove the PID file isn't as much of a problem. That's easy for the init system to do if Kannel can't: kill $pid && rm -f $pidfile. Or, you can just leave it there. It's a much more pleasant problem for the init script author to have =)

Since the only purpose of the PID file is to make the init system happy, having it written as root better fulfils that purpose.

#4 - 09/21/2017 01:59 PM - Alexander Malysh

Hi,

I think you are wrong.

1. Official Kannel repo:

https://redmine.kannel.org/projects/kannel/repository/entry/trunk/debian/kannel.init

2. For debian we are using start-stop-daemon and use name option. With name option AFAIK start-stop-daemon checks the name of process
```
echo -n "Stopping WAP gateway:"
test ! -z $START_SMSBOX && (
echo -n " smsbox"
start-stop-daemon --stop --retry 5 --quiet \
--pidfile $PIDFILES/kannel_smsbox.pid \
--name smsbox
```

3. PID file that Kannel writes is not *only* for init scripts. Just check sources if you like, Kannel will not start if PID file exists, that prevents from running multiple instances.

4. Because of (3) and because Kannel runs not only on Linux (but Windows, OSX etc) , Kannel has to cleanup his PID file by itself.

#5 - 09/21/2017 02:26 PM - Michael Orlitzky

> For debian we are using start-stop-daemon and use name option. With name option AFAIK start-stop-daemon checks the name of process

If you pass the --pidfile flag, then the PID file is all that gets used. The name of the executable is ignored. The same thing is true in OpenRC.

> PID file that Kannel writes is not only for init scripts. Just check sources if you like, Kannel will not start if PID file exists, that prevents from running multiple instances.

Ah, so you're also using it for a lock file. The same situation arose with MIMEDefang. The solution we came up with there was to split the PID file and lock file into two separate paths (specified by two different command-line flags). The PID file and lock files serve two fundamentally different purposes: the PID file contains sensitive data, the lock file doesn't; the lock file needs to be cleaned up, the PID file doesn't. Trying to use one file for two things is the problem there.

#6 - 09/12/2018 03:30 PM - Guillem Jover

Michael Orlitzky wrote:

> > For debian we are using start-stop-daemon and use name option. With name option AFAIK start-stop-daemon checks the name of process
>
> If you pass the --pidfile flag, then the PID file is all that gets used. The name of the executable is ignored. The same thing is true in OpenRC.

For Debian's or dpkg's start-stop-daemon, that's certainly not correct. Any of the match options specified will apply at the same time. If this does not work for you and can provide a reproducer I'd be interested, because that'd be definitely a bug in start-stop-daemon that'd I'd like to fix. If this actually happens only with the OpenRC version of start-stop-daemon, then that's a bug for them to fix (and a pretty serious one at that).

In this case though, I don't see much of an issue, either --exec or --name should be enough of a protection against random killings. It would be nice to also use the --user match option on both start and stop to make the init script even more robust, and possibly add the --exec option in addition to the --name one.

Otherwise, I think this report should simply be closed.

#7 - 09/12/2018 06:25 PM - Michael Orlitzky

To be clear, the Debian issue is a red herring.

If Debian's start-stop-daemon works like you say, and if we add --user parameters to every Debian init script, and if we change the way OpenRC works (affecting thousands of existing init scripts...), then **it's still impossible to write a POSIX SysV init script that does the right thing**.

Debian and Gentoo don't even need the PID file; s-s-d has the ability to supervise foreground processes. The PID within the PID file is only needed for SysV init, where it can't be used because the data is unreliable.

#8 - 09/14/2018 04:11 PM - Guillem Jover

Michael Orlitzky wrote:

> If Debian's start-stop-daemon works like you say, and if we add --user parameters to every Debian init script,

As I've said above, this is not strictly necessary, and should not turn those invocations from insecure to secure, it just adds additional robustness. And in any case it only makes sense when the user is not root, so definitely not for every init script.

> and if we change the way OpenRC works (affecting thousands of existing init scripts...),

That's for the OpenRC developers to consider, if their implementation does not work like the original one in dpkg, then I'd consider that a bug than just needs fixing.

> then **it's still impossible to write a POSIX SysV init script that does the right thing**.

I'm still failing to see why. The daemon either runs as root, which means that if it gets compromised everything is lost anyway, or as an unprivileged user. If the latter, and assuming it does not perform privilege escalation, then it can only affect processes and files owned by that user, which means not the init script itself. So it cannot start making s-s-d match on a different process name or executable pathname (or possibly if --user is used on a different user), even if the pid has been modified by the compromised daemon.

What am I missing?

> Debian and Gentoo don't even need the PID file; s-s-d has the ability to supervise foreground processes. The PID within the PID file is only needed for SysV init, where it can't be used because the data is unreliable.

Not really, s-s-d does not supervise processes, it can take a process that only supports foreground running and force it into backgrounding, even

creating the pidfile on its behalf. The pidfile is required to avoid killing unintended instances of the same daemon running for example inside a chroot, or for daemons that support running multiple instances on the same system, etc. For unprivileged daemons that have created the pidfile themselves as that user, the pidfile might be untrustworthy, but the security is not provided by it, but by the other match options. I will clarify the security implications of this specific case in dpkg's s-s-d man page.

#9 - 09/14/2018 04:43 PM - Michael Orlitzky

Guillem Jover wrote:

> Michael Orlitzky wrote:
>
> > If Debian's start-stop-daemon works like you say, and if we add --user parameters to every Debian init script,
>
> As I've said above, this is not strictly necessary, and should not turn those invocations from insecure to secure, it just adds additional robustness.

I guess it depends on how loose you want to be with your definition of secure =)

Without the --user flag, the process-name matching will still allow an unprivileged user to kill processes owned by root, so long as they have the same name. That's not a huge vulnerability or anything, but it's still wrong.

> then **it's still impossible to write a POSIX SysV init script that does the right thing**.
>
> I'm still failing to see why. The daemon either runs as root, which means that if it gets compromised everything is lost anyway, or as an unprivileged user. If the latter, and assuming it does not perform privilege escalation, then it can only affect processes and files owned by that user, which means not the init script itself. So it cannot start making s-s-d match on a different process name or executable pathname (or possibly if --user is used on a different user), even if the pid has been modified by the compromised daemon.
>
> What am I missing?

With respect to the point I'm trying to make: that s-s-d is not required to exist by POSIX. If s-s-d is there, you're fine; but Kannel can't ship a "SysV init script" that uses s-s-d and expect it to work everywhere.

For sure, you can fix this issue one way on Debian. And you can fix it another way on Gentoo, and a third on Fedora, and so on. Everyone can hack around the problem in their own distro-specific way, but you can't write a single init script using only POSIX that will work everywhere. And when I say "work," I mean "guarantee that the unprivileged user can only do things that the operating system would otherwise allow him to do." So he should never be able to kill a process that he doesn't own, for example.

To elaborate, what you would really want to do in this case is drop privileges before sending the SIGTERM. Then the unprivileged user can't kill any processes he doesn't own, and everyone's happy. The problem is that there's no cross-platform (guaranteed to be there by POSIX) way to drop privileges. Creating the PID file as root avoids the whole issue, since you no longer need to drop privileges. You can just send the SIGTERM as root, which in practice is what everyone does anyway. And at that point, you can write an init script that will work on every POSIX system.

Is that one of Kannel's goals? Maybe, maybe not. But the people who aren't using bare SysV init all have better solutions (like s-s-d) to track the daemon process. So if you don't want to support SysV init, you can get rid of the PID file altogether -- problem solved. But if you are going to support SysV init, then it should be possible to use the PID file securely on one of those systems, without having everyone invent their own platform-specific hacks to do so.

> Debian and Gentoo don't even need the PID file; s-s-d has the ability to supervise foreground processes. The PID within the PID file is only needed for SysV init, where it can't be used because the data is unreliable.
>
> Not really, s-s-d does not supervise processes, it can take a process that only supports foreground running and force it into backgrounding, even creating the pidfile on its behalf. The pidfile is required to avoid killing unintended instances of the same daemon running for example inside a chroot, or for daemons that support running multiple instances on the same system, etc. For unprivileged daemons that have created the pidfile themselves as that user, the pidfile might be untrustworthy, but the security is not provided by it, but by the other match options. I will clarify the security implications of this specific case in dpkg's s-s-d man page.

But notably, when s-s-d forces a process into the background, the PID file created by s-s-d is owned by root, regardless of whether or not the daemon drops privileges. I only mentioned this because it lets you work around the security issue: if the daemon writes a PID file after dropping privileges, then you can run it in the foreground and tell s-s-d to create a PID file (owned by root -- good). You might wind up with a stray user-owned PID file somewhere, but so long as it's not used by the init system (i.e. root), you're fine.