strongSwan - Feature #519

selinux issues due to leaking file descriptors to xtables-multi

17.02.2014 12:08 - Pavel Šimerda

Status:	Closed	**Start date:**	17.02.2014
Priority:	High	**Due date:**	
Assignee:	Martin Willi	**Estimated time:**	0.00 hour
Category:	charon		
Target version:	5.2.0		
Resolution:	Fixed		

Description

Source: https://bugzilla.redhat.com/show_bug.cgi?id=903638

```
Jeremy Beker 2013-02-18 16:49:25 CET
```

Here you go. This is from a restart of strongswan, so might have messages from the shutdown as well.

```
----
type=SYSCALL msg=audit(02/18/2013 10:48:22.171:1126) : arch=x86_64 syscall=execve success=yes exit=0 a0=0x1d1b750 a1=0x1cfd3f0 a2=0x1d09db0 a3=0xc0 items=0 ppid=5762 pid=5775 auid=unset uid=root gid=root euid=root suid=root fsuid=root egid=root sgid=root fsgid=root ses=unset tty=(none) comm=iptables exe=/usr/sbin/xtables-multi subj=system_u:system_r:iptables_t:s0 key=(null)
type=AVC msg=audit(02/18/2013 10:48:22.171:1126) : avc:  denied  { read } for  pid=5775 comm=iptables path=/dev/random dev="devtmpfs" ino=1032 scontext=system_u:system_r:iptables_t:s0 tcontext=system_u:object_r:random_device_t:s0 tclass=chr_file
----
type=SYSCALL msg=audit(02/18/2013 10:48:22.178:1128) : arch=x86_64 syscall=execve success=yes exit=0 a0=0x1d1adf0 a1=0x1cfd3f0 a2=0x1d09db0 a3=0xc0 items=0 ppid=5762 pid=5776 auid=unset uid=root gid=root euid=root suid=root fsuid=root egid=root sgid=root fsgid=root ses=unset tty=(none) comm=iptables exe=/usr/sbin/xtables-multi subj=system_u:system_r:iptables_t:s0 key=(null)
type=AVC msg=audit(02/18/2013 10:48:22.178:1128) : avc:  denied  { read } for  pid=5776 comm=iptables path=/dev/random dev="devtmpfs" ino=1032 scontext=system_u:system_r:iptables_t:s0 tcontext=system_u:object_r:random_device_t:s0 tclass=chr_file
----
type=SYSCALL msg=audit(02/18/2013 10:48:22.180:1130) : arch=x86_64 syscall=execve success=yes exit=0 a0=0x1d1b410 a1=0x1cfd3f0 a2=0x1d09db0 a3=0xc0 items=0 ppid=5762 pid=5777 auid=unset uid=root gid=root euid=root suid=root fsuid=root egid=root sgid=root fsgid=root ses=unset tty=(none) comm=iptables exe=/usr/sbin/xtables-multi subj=system_u:system_r:iptables_t:s0 key=(null)
type=AVC msg=audit(02/18/2013 10:48:22.180:1130) : avc:  denied  { read } for  pid=5777 comm=iptables path=/dev/random dev="devtmpfs" ino=1032 scontext=system_u:system_r:iptables_t:s0 tcontext=system_u:object_r:random_device_t:s0 tclass=chr_file
----
type=SYSCALL msg=audit(02/18/2013 10:48:22.181:1132) : arch=x86_64 syscall=execve success=yes exit=0 a0=0x1d1b990 a1=0x1cfd3f0 a2=0x1d09db0 a3=0xc0 items=0 ppid=5762 pid=5778 auid=unset uid=root gid=root euid=root suid=root fsuid=root egid=root sgid=root fsgid=root ses=unset tty=(none) comm=iptables exe=/usr/sbin/xtables-multi subj=system_u:system_r:iptables_t:s0 key=(null)
type=AVC msg=audit(02/18/2013 10:48:22.181:1132) : avc:  denied  { read } for  pid=5778 comm=iptables path=/dev/random dev="devtmpfs" ino=1032 scontext=system_u:system_r:iptables_t:s0 tcontext=system_u:object_r:random_device_t:s0 tclass=chr_file
----
type=SYSCALL msg=audit(02/18/2013 10:48:22.572:1152) : arch=x86_64 syscall=execve success=yes exit=0 a0=0x24ab790 a1=0x248d3f0 a2=0x2499db0 a3=0xc8 items=0 ppid=5841 pid=5850 auid=unset uid=root gid=root euid=root suid=root fsuid=root egid=root sgid=root fsgid=root ses=unset tty=(none) comm=iptables exe=/usr/sbin/xtables-multi subj=system_u:system_r:iptables_t:s0 key=(null)
type=AVC msg=audit(02/18/2013 10:48:22.572:1152) : avc:  denied  { write } for  pid=5850 comm=iptables path=/run/charon.pid dev="tmpfs" ino=3168043 scontext=system_u:system_r:iptables_t:s0 tcontext=system_u:object_r:initrc_var_run_t:s0 tclass=file
type=AVC msg=audit(02/18/2013 10:48:22.572:1152) : avc:  denied  { read } for  pid=5850 comm=iptables path=/dev/random dev="devtmpfs" ino=1032 scontext=system_u:system_r:iptables_t:s0 tcontext=system_u:object_r:random_device_t:s0 tclass=chr_file
----
type=SYSCALL msg=audit(02/18/2013 10:48:22.574:1154) : arch=x86_64 syscall=execve success=yes exit=0 a0=0x24aadf0 a1=0x248d3f0 a2=0x2499db0 a3=0xc8 items=0 ppid=5841 pid=5851 auid=unset uid=root gid=root euid=root suid=root fsuid=root egid=root sgid=root fsgid=root ses=unset tty=(none) comm=iptables exe=/usr/sbin/xtables-multi subj=system_u:system_r:iptables_t:s0 key=(null)
type=AVC msg=audit(02/18/2013 10:48:22.574:1154) : avc:  denied  { write } for  pid=5851 comm=iptables path=/run/charon.pid dev="tmpfs" ino=3168043 scontext=system_u:system_r:iptables_t:s0 tcontext=system_u:object_r:initrc_var_run_t:s0 tclass=file
type=AVC msg=audit(02/18/2013 10:48:22.574:1154) : avc:  denied  { read } for  pid=5851 comm=iptables path=/dev/random
```

```
dev="devtmpfs" ino=1032 scontext=system_u:system_r:iptables_t:s0 tcontext=system_u:object_r:random_device_t:s0 tclass=chr_file
----
type=SYSCALL msg=audit(02/18/2013 10:48:22.576:1156) : arch=x86_64 syscall=execve success=yes exit=0 a0=0x24ab450
a1=0x248d3f0 a2=0x2499db0 a3=0xc8 items=0 ppid=5841 pid=5852 auid=unset uid=root gid=root euid=root suid=root fsuid=root
egid=root sgid=root fsgid=root ses=unset tty=(none) comm=iptables exe=/usr/sbin/xtables-multi
subj=system_u:system_r:iptables_t:s0 key=(null)
type=AVC msg=audit(02/18/2013 10:48:22.576:1156) : avc:  denied  { write } for  pid=5852 comm=iptables path=/run/charon.pid
dev="tmpfs" ino=3168043 scontext=system_u:system_r:iptables_t:s0 tcontext=system_u:object_r:initrc_var_run_t:s0 tclass=file
type=AVC msg=audit(02/18/2013 10:48:22.576:1156) : avc:  denied  { read } for  pid=5852 comm=iptables path=/dev/random
dev="devtmpfs" ino=1032 scontext=system_u:system_r:iptables_t:s0 tcontext=system_u:object_r:random_device_t:s0 tclass=chr_file
----
type=SYSCALL msg=audit(02/18/2013 10:48:22.578:1158) : arch=x86_64 syscall=execve success=yes exit=0 a0=0x24aba30
a1=0x248d3f0 a2=0x2499db0 a3=0xc8 items=0 ppid=5841 pid=5853 auid=unset uid=root gid=root euid=root suid=root fsuid=root
egid=root sgid=root fsgid=root ses=unset tty=(none) comm=iptables exe=/usr/sbin/xtables-multi
subj=system_u:system_r:iptables_t:s0 key=(null)
type=AVC msg=audit(02/18/2013 10:48:22.578:1158) : avc:  denied  { write } for  pid=5853 comm=iptables path=/run/charon.pid
dev="tmpfs" ino=3168043 scontext=system_u:system_r:iptables_t:s0 tcontext=system_u:object_r:initrc_var_run_t:s0 tclass=file
type=AVC msg=audit(02/18/2013 10:48:22.578:1158) : avc:  denied  { read } for  pid=5853 comm=iptables path=/dev/random
dev="devtmpfs" ino=1032 scontext=system_u:system_r:iptables_t:s0 tcontext=system_u:object_r:random_device_t:s0 tclass=chr_file
```

`Daniel Walsh 2013-02-18 16:56:22 CET`

As I expected these are leaks from stongswan.

It needs to close the file descriptors on exec.

fcntl(fd, F_SETFD, FD_CLOEXEC)

For /dev/random and /run/chron.pid

You might be able to do this in the open call also.

Associated revisions

Revision cdfc2706 - 24.06.2014 14:49 - Martin Willi

charon: Set CLOEXEC flag on daemon PID file and /dev/(u)random source FDs

On Fedora, SELinux complains about these open file descriptors when the
updown script invokes iptables. While it seems difficult to set the flag
on all file descriptors, this at least fixes those covered by the SELinux
policy.

As these two cases are in code executed while the daemon is still single
threaded, we avoid the use of atomic but not fully portable fdopen("e") or
open(O_CLOEXEC) calls.

Fixes #519.

Revision 866514c7 - 24.06.2014 15:26 - Martin Willi

charon: Set CLOEXEC flag on daemon PID file and /dev/(u)random source FDs

On Fedora, SELinux complains about these open file descriptors when the
updown script invokes iptables. While it seems difficult to set the flag
on all file descriptors, this at least fixes those covered by the SELinux
policy.

As these two cases are in code executed while the daemon is still single
threaded, we avoid the use of atomic but not fully portable fdopen("e") or
open(O_CLOEXEC) calls.

Fixes #519.

History

#1 - 19.02.2014 10:12 - Pavel Šimerda

- File strongswan-selinux.patch added

#2 - 14.04.2014 13:47 - Pavel Šimerda

Hi,

the patch is rather trivial and shouldn't cause any problems and would let me remove one more patch from the Fedora package.

Cheers,

Pavel

#3 - 14.04.2014 14:00 - Martin Willi

Hi Pavel,

> the patch is rather trivial and shouldn't cause any problems and would let me remove one more patch from the Fedora package.

While I agree to that, the problem is that it fixes the issue for two single file descriptors only.

Depending on the plugin configuration, there is a large number of additional file descriptors not covered. Even worse, over some file descriptors we don't even have direct control, for example those opened by libraries. So setting CLOEXEC on all file descriptors is not only cumbersome, but also can't work for all of them.

> avc: denied { read } for pid=5777 comm=iptables path=/dev/random dev="devtmpfs" ino=1032 scontext=system_u:system_r:iptables_t:s0 tcontext=system_u:object_r:random_device_t:s0 tclass=chr_file

I'm no SELinux expert, but why does it complain about a read() access on /dev/random just for an open file descriptor? No actual read() is done, so I don't see why a non-CLOEXECed file descriptor is a policy violation. I think enforcing a policy for that is just a little too strict?

I fully agree that CLOEXECing file descriptors before calling xtables would be preferable, but there is no global way of doing so. How should we handle file descriptors outside of our control?

Regards
Martin

#4 - 14.04.2014 14:07 - Tobias Brunner

>> the patch is rather trivial and shouldn't cause any problems and would let me remove one more patch from the Fedora package.
>
> While I agree to that, the problem is that it fixes the issue for two single file descriptors only.

The fix is also not very portable. The "e" option for fopen(3) is a GNU extension. And O_CLOEXEC is pretty new (POSIX.1-2008) and not yet supported on all platforms.

#5 - 15.04.2014 10:01 - Pavel Šimerda

Tobias Brunner wrote:

> The fix is also not very portable. The "e" option for fopen(3) is a GNU extension. And O_CLOEXEC is pretty new (POSIX.1-2008) and not yet supported on all platforms.

So, what next?

1) Do we keep the patch in Fedora (and other linux distributions) and keep the upstream leaking the descriptors?

2) Am I expected to come up with a patch using conditional builds or similar technique?

3) Is upstream going to prepare the patch themselves?

More resources:

- http://danwalsh.livejournal.com/53603.html
- http://danwalsh.livejournal.com/6117.html?thread=23525
- https://docs.fedoraproject.org/en-US/Fedora_Security_Team//html/Defensive_Coding/sect-Defensive_Coding-Tasks-Descriptors-Child_Processes.html

#6 - 15.04.2014 10:02 - Pavel Šimerda

The good old way would be to close the file descriptors explicitly just before calling exec.

#7 - 20.06.2014 12:01 - Pavel Šimerda

- File 0001-use-O_CLOEXEC-to-avoid-file-descriptor-leaks-on-plat.patch added

A new patch that provides fallbacks for platforms not supporting O_CLOEXEC in open() or fopen().

#8 - 20.06.2014 15:05 - Martin Willi

Pavel,

Thanks for your patch. However, I still think that setting CLOEXEC on these specific file descriptors is not the real fix. There are many other file descriptors that are not covered. Can you or an SELinux expert address the questions I raised at #519#note-3?

Also, do you classify any file descriptor without CLOEXEC in all strongSwan code and any potentially linked library as a bug?

Best regards
Martin

#9 - 20.06.2014 15:14 - Martin Willi

Or what about adding something like:

```
for fd in $(ls /proc/$$/fd); do
  case "$fd" in
    0|1|2)
      ;;
    *)
      eval "exec $fd>&-"
      ;;
  esac
done
```

to the default _updown script as a work-around?

#10 - 23.06.2014 15:30 - Pavel Šimerda

Martin Willi wrote:

> I'm no SELinux expert, but why does it complain about a read() access on /dev/random just for an open file descriptor?

In my opinion, all linux access control is done on **open** or **exec**, not **read**, as the latter syscall just requests more data from an already opened and authorized stream. The read() function doesn't even have EACCESS among its error values.

> No actual read() is done, so I don't see why a non-CLOEXECed file descriptor is a policy violation. I think enforcing a policy for that is just a little too strict?

Probably answered by the above.

Martin Willi wrote:

> Pavel,
>
> Thanks for your patch. However, I still think that setting CLOEXEC on these specific file descriptors is not the real fix.

At least it gets us over the security policy used in Fedora, which is better than nothing. That may be because Fedora protects those specific resources in a specific way, or that other protected resources are not accessed by default. I believe that the Strongswan selinux policy is somehow derived from the Openswan one but I don't have more information.

> There are many other file descriptors that are not covered.

In my opinion, all other file descriptors could be covered the same way. While many library calls support O_CLOEXEC in one way or another, there's still fcntl() that can be called on an existing file descriptor if you receive one without O_CLOEXEC from another library or whatever.

But I'm curious whether there are any other file descriptors at all and why wouldn't selinux complain about them. In my the policy causes all file descriptors to be checked for access.

> Can you or an SELinux expert address the questions I raised at #519#note-3?

Done.

> Also, do you classify any file descriptor without CLOEXEC in all strongSwan code and any potentially linked library as a bug?

I would say yes. From the security point of view, all software should only open resources that it actually needs, and if you want to use a policy engine like selinux, it feels natural to extend it to indirect opening which happens when exec() is called and the new process receives a bunch of already open file descriptors.

- http://danwalsh.livejournal.com/53603.html

I don't see any valid reason to pass file descriptors to programs that won't use them anyway as they don't know about them. The three standard file descriptors should be good enough.

As for the script, if that makes selinux stop complaining about the leaks and helps me to stop carrying a patch, why not. But I guess it would be best to just open all file descriptors with O_CLOEXEC except in situations where you actualy want to pass file descriptors through exec. This is how it's done in most packages used in Fedora.

#11 - 24.06.2014 15:31 - Martin Willi

- Tracker changed from Issue to Feature

- Category set to charon

- Assignee set to Martin Willi

- Target version set to 5.2.0

- Resolution set to Fixed

Pavel,

> While many library calls support O_CLOEXEC in one way or another, there's still fcntl() that can be called on an existing file descriptor if you receive one without O_CLOEXEC from another library or whatever.

Most likely you won't even have access to the fd; think of a socket or something. If your SELinux policy says iptables is not allowed to have open TCP sockets, it gets difficult to use a networking library. Unlikely that all of them set SOCK_CLOEXEC, which is rather new.

> From the security point of view, all software should only open resources that it actually needs

As said, I agree to that. But as we have no way to globally CLOEXEC, it is just not really practical for all FDs.

> I don't see any valid reason to pass file descriptors to programs that won't use them anyway

Me neither, but unfortunately that is the default behavior, which is difficult to fix properly and in a portable way.

Anyway: I've push the associated commit for 5.2.0. It sets CLOEXEC on these two FD, but only using the portable fcntl(). That code does not run while the daemon is multi-threaded, and therefore no race conditions are to expect.

Fixing all FDs in our code base is another story, though. We'd need atomic operations in many cases. That works on newer Linux systems, but could be difficult to implement on others.

Regards
Martin

#12 - 26.06.2014 15:21 - Pavel Šimerda

Thanks for the fix for the two specific file descriptors. For me the issue is fixed, as I can drop the patch from the package with the next release. I'll let you know if I have more information for the all FDs case.

#13 - 08.07.2014 09:44 - Tobias Brunner

- Status changed from New to Closed

Files

strongswan-selinux.patch	876 Bytes	19.02.2014	Pavel Šimerda
0001-use-O_CLOEXEC-to-avoid-file-descriptor-leaks-on-plat.patch	1.56 KB	20.06.2014	Pavel Šimerda