Subject: [RFC PATCH] namespaces: document unshare security implications
Posted by serue on Fri, 09 Nov 2007 22:21:45 GMT
View Forum Message <> Reply to Message

Ok, the following isn't meant so much as a patch as for discussion.
However, this may be a change we want to think about for awhile and
collect opinions and facts. So having this file sitting in the
kernel tree (updated with the results of any discussion we have in the
meantime) may be useful.

So what do people think? Are we ok using CAP_SYS_ADMIN? Do we
authorize unsharing of each resource using the capability required
to administrate the resource? Do we introduce CAP_NS_UNSHARE? Do
we add CAP_SYS_USHARE, CAP_NET_UNSHARE, and CAP_USER_UNSHARE? Or
do we allow unprivileged users to unshare, trusting that the actual
administration is properly authorized?

thanks,
-serge

>From 0a76e72a6900d1c47caa6aaeb5008e8408fd35e6 Mon Sep 17 00:00:00 2001
From: sergeh@us.ibm.com <hallyn@kernel.(none)>
Date: Fri, 9 Nov 2007 13:32:40 -0800
Subject: [PATCH 1/1] namespaces: document unshare security implications

Add a file under Documentation/namespaces to discuss security
implications of unsharing namespaces.

Signed-off-by: Serge Hallyn <serue@us.ibm.com>

 Documentation/namespaces/security.txt | 69 +++++++++++++++++++++++++++++++++
 1 files changed, 69 insertions(+), 0 deletions(-)
 create mode 100644 Documentation/namespaces/security.txt

diff --git a/Documentation/namespaces/security.txt b/Documentation/namespaces/security.txt
new file mode 100644
index 0000000..c68794b
--- /dev/null
+++ b/Documentation/namespaces/security.txt
@@ -0,0 +1,69 @@
+Currently, cloning/unsharing of namespaces requires CAP_SYS_ADMIN.
+This file addresses some ways to allow unprivileged users to
+unshare namespaces.
+
+First, of course, a program unsharing namespaces can be made setuid
+root. A slightly safer alternative a program unsharing namespaces
+can be given cap_sys_admin in its file permitted capabilities.
+

+Requiring CAP_SYS_ADMIN is a legacy behavior stemming from the
+fact that the original namespace, the mounts namespace, required
+CAP_SYS_ADMIN for cloning. Unfortunately CAP_SYS_ADMIN is used
+to authorize many other actions, so that giving away CAP_SYS_ADMIN
+to allow unsharing also allows the unsharing user many other
+privileges.
+
+Instead of using CAP_SYS_ADMIN, a new capability, CAP_NS_UNSHARE,
+could be introduced. This way a program or user wouldn't have
+to be granted full CAP_SYS_ADMIN rights to be able to clone/unshare
+namespaces, but a fully unprivileged user still could not
+clone/unshare.
+
+Or, unsharing namespaces could be turned into an entirely unprivileged
+operation. Unsharing a namespace does not give the user any new
+rights to modify the unshared resource in the new namespace. For
+instance, after doing
+ unshare(CLONE_NEWNS)
+the unprivileged task can't perform any mount actions he couldn't
+before the unshare.
+
+Is it safe to allow namespace unsharing by nonprivileged users?
+The following tries to answer that question per-namespace:
+
+UTS
+ Safe. Can't sethostname without cap_sys_admin.
+ But similarly, since you can't sethostname without
+ cap_sys_admin there may not be any point in
+ unsharing your utsns without cap_sys_admin.
+IPC
+ could be unsafe if a setuid root application expects
+ to talk to a privileged server in the init ipc ns.
+ The opencryptoki pkcsslotd might be an example.
+VFS
+ user might hide himself from root mount activity,
+ but in general vfs ns are safe and don't change the
+ safety of user mounts.
+ Since there is mount activity (and more to come) that
+ users can do without CAP_SYS_ADMIN, it may be useful
+ to allow unshare(CLONE_NEWNS) without CAP_SYS_ADMIN.
+PID
+ safe
+User
+ user can get around quotas. As user namespaces are
+ fleshed out, root in a user namespace will be confined,
+ and equivalent user ids between namespaces will be
+ isolated.
+Net

+ user won't have cap_net_admin so won't be able to set
+ up networking in new ns. Biggest risk is due to
+ any root services which don't handle failure due to
+ no network right.
+ If netlink isn't handled right, user might be able to
+ get around the audit daemon! That's very bad.
+ Since network has it's own CAP_NET_ADMIN, it may make
+ sense to require that to unshare(CLONE_NEWNET). But
+ requiring differnet capabilities to unshare different
+ resources may be too confusing/annoying.
+
+All these share the threat of extra memory consumption, but
+this can be addressed using cgroups.
--
1.5.1

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [RFC PATCH] namespaces: document unshare security implications
Posted by ebiederm on Thu, 15 Nov 2007 13:55:53 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> Ok, the following isn't meant so much as a patch as for discussion.
> However, this may be a change we want to think about for awhile and
> collect opinions and facts. So having this file sitting in the
> kernel tree (updated with the results of any discussion we have in the
> meantime) may be useful.
>
> So what do people think? Are we ok using CAP_SYS_ADMIN? Do we
> authorize unsharing of each resource using the capability required
> to administrate the resource? Do we introduce CAP_NS_UNSHARE? Do
> we add CAP_SYS_USHARE, CAP_NET_UNSHARE, and CAP_USER_UNSHARE? Or
> do we allow unprivileged users to unshare, trusting that the actual
> administration is properly authorized?

Well if we ant to sit this in the kernel we need to remove mention
of CAP_NS_UNSHARE.

However even there the document below is only an ok first stab at
documenting things.

The big big big problem are suid executables. If we don't have suid

executables and the namespaces only apply to our children we can
unshare them all day long and no one cares. If we do have suid executables
any messing up of their context that they are not prepared to deal with
is a potential security violation.

So I think CAP_SYS_ADMIN is a good starting place. It is trivial verifiable
that it is safe. So starting there allows us to work on other aspects
of the problem for now.

I would like to remove the restrictions from creating new namespaces
however we will either have to have restrictions like the current
unprivileged mount patches, so we don't surprised root. Or we figure out
how to ensure we don't have suid applications.

Given my intuitive understanding of a complete uid namespace it
fundamentally prohibits suid executables from executing because those
users simply do not exist in the new namespace. So my hunch is we can
drop the requirement for CAP_SYS_ADMIN on namespace creation in
concert with a uid namespace creation.

So my feeling at the moment is that we need to flesh out and complete
the namespaces we have user, net, pid, user and then come back and
see what we can do.

I don't think it makes sense to document a snapshot in time of the
discussion with unresolved issues in the Documentation directory.
Documenting what is and how we got there is fine (that is timeless)
documenting what could be will likely be out of date before the patch
is merged into Linus's tree.

Eric

> thanks,
> -serge
>
>>From 0a76e72a6900d1c47caa6aaeb5008e8408fd35e6 Mon Sep 17 00:00:00 2001
> From: sergeh@us.ibm.com <hallyn@kernel.(none)>
> Date: Fri, 9 Nov 2007 13:32:40 -0800
> Subject: [PATCH 1/1] namespaces: document unshare security implications
>
> Add a file under Documentation/namespaces to discuss security
> implications of unsharing namespaces.
>
> Signed-off-by: Serge Hallyn <serue@us.ibm.com>
> ---

> Documentation/namespaces/security.txt | 69 ++++++++++++++++++++++++++++++++++
> 1 files changed, 69 insertions(+), 0 deletions(-)
> create mode 100644 Documentation/namespaces/security.txt
>
> diff --git a/Documentation/namespaces/security.txt
> b/Documentation/namespaces/security.txt
> new file mode 100644
> index 0000000..c68794b
> --- /dev/null
> +++ b/Documentation/namespaces/security.txt
> @@ -0,0 +1,69 @@
> +Currently, cloning/unsharing of namespaces requires CAP_SYS_ADMIN.
> +This file addresses some ways to allow unprivileged users to
> +unshare namespaces.
> +
> +First, of course, a program unsharing namespaces can be made setuid
> +root. A slightly safer alternative a program unsharing namespaces
> +can be given cap_sys_admin in its file permitted capabilities.
> +
> +Requiring CAP_SYS_ADMIN is a legacy behavior stemming from the
> +fact that the original namespace, the mounts namespace, required
> +CAP_SYS_ADMIN for cloning. Unfortunately CAP_SYS_ADMIN is used
> +to authorize many other actions, so that giving away CAP_SYS_ADMIN
> +to allow unsharing also allows the unsharing user many other
> +privileges.
> +
> +Instead of using CAP_SYS_ADMIN, a new capability, CAP_NS_UNSHARE,
> +could be introduced. This way a program or user wouldn't have
> +to be granted full CAP_SYS_ADMIN rights to be able to clone/unshare
> +namespaces, but a fully unprivileged user still could not
> +clone/unshare.
> +
> +Or, unsharing namespaces could be turned into an entirely unprivileged
> +operation. Unsharing a namespace does not give the user any new
> +rights to modify the unshared resource in the new namespace. For
> +instance, after doing
> + unshare(CLONE_NEWNS)
> +the unprivileged task can't perform any mount actions he couldn't
> +before the unshare.
> +
> +Is it safe to allow namespace unsharing by nonprivileged users?
> +The following tries to answer that question per-namespace:
> +
> +UTS
> + Safe. Can't sethostname without cap_sys_admin.
> + But similarly, since you can't sethostname without
> + cap_sys_admin there may not be any point in
> + unsharing your utsns without cap_sys_admin.

Yes. But you can hide from a future sethostname. So because
the your hostname won't change the effect is as if you had
set it.

> +IPC
> + could be unsafe if a setuid root application expects
> + to talk to a privileged server in the init ipc ns.
> + The opencryptoki pkcsslotd might be an example.
> +VFS
> + user might hide himself from root mount activity,
> + but in general vfs ns are safe and don't change the
> + safety of user mounts.
> + Since there is mount activity (and more to come) that
> + users can do without CAP_SYS_ADMIN, it may be useful
> + to allow unshare(CLONE_NEWNS) without CAP_SYS_ADMIN.

I assume you mean the mount namespace. There is still the
stale state problem and suid executables.

> +PID
> + safe

Nope. It can hide processes that a suid executable would
like to signal... It messes up the use of pid files and
the like.

> +User
> + user can get around quotas. As user namespaces are
> + fleshed out, root in a user namespace will be confined,
> + and equivalent user ids between namespaces will be
> + isolated.
> +Net
> + user won't have cap_net_admin so won't be able to set
> + up networking in new ns. Biggest risk is due to
> + any root services which don't handle failure due to
> + no network right.
> + If netlink isn't handled right, user might be able to
> + get around the audit daemon! That's very bad.
> + Since network has it's own CAP_NET_ADMIN, it may make
> + sense to require that to unshare(CLONE_NEWNET). But
> + requiring differnet capabilities to unshare different
> + resources may be too confusing/annoying.
> +
> +All these share the threat of extra memory consumption, but
> +this can be addressed using cgroups.
> --
> 1.5.1

>
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [RFC PATCH] namespaces: document unshare security implications
Posted by serue on Thu, 15 Nov 2007 16:40:35 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:

Thanks for responding, Eric. Good points.

> > Ok, the following isn't meant so much as a patch as for discussion.
> > However, this may be a change we want to think about for awhile and
> > collect opinions and facts. So having this file sitting in the
> > kernel tree (updated with the results of any discussion we have in the
> > meantime) may be useful.
> >
> > So what do people think? Are we ok using CAP_SYS_ADMIN? Do we
> > authorize unsharing of each resource using the capability required
> > to administrate the resource? Do we introduce CAP_NS_UNSHARE? Do
> > we add CAP_SYS_USHARE, CAP_NET_UNSHARE, and CAP_USER_UNSHARE? Or
> > do we allow unprivileged users to unshare, trusting that the actual
> > administration is properly authorized?
>
> Well if we ant to sit this in the kernel we need to remove mention
> of CAP_NS_UNSHARE.

Mostly for now I wanted it to sit in the mailing list :)

> However even there the document below is only an ok first stab at
> documenting things.
>
> The big big big problem are suid executables. If we don't have suid
> executables and the namespaces only apply to our children we can
> unshare them all day long and no one cares. If we do have suid executables
> any messing up of their context that they are not prepared to deal with
> is a potential security violation.

Ok let's look just at the mounts namespace since that one is complete.

Unsharing the mounts namespace makes no change in mounts context, and
does not provide the user any additional privilege to make such change.
So suid executable are safe.

> So I think CAP_SYS_ADMIN is a good starting place. It is trivial verifiable
> that it is safe. So starting there allows us to work on other aspects
> of the problem for now.

It was a good starting place, but at this point I have two concerns with
sticking with CAP_SYS_ADMIN:

1. now that file capabilities are upstream, people may want to
add just the requisite capability in fP for an unsharing helper
program. Cedric had mentioned wanting to do that.
If we are going to switch to unprivileged unshares, then doing
so later is ok. But if we're going to switch to a custom
capability later, then that could be seen as an API change
since users will have to switch the capability on all the
unsharing programs.

2. As I pointed out a few times, we can cleanly separate
unsharing namespace and actually manipulating the resources.
By requiring CAP_SYS_ADMIN for both unsharing a mounts namespace
and for performing privileged mounts, any program given the
authority to unshare is automatically given the authority to
also completely manipulate the mounts, both in the new private
namespace and the original namespace (by just not unsharing).

It's even worse with the net namespace, since the privilege
needed to unshare the namespace authorizes you to update
other namespaces in the system, but *not* network devices!
But like you say let's stick with established namespaces.

So while I started the original email just wanting some discussion, now
I'm actually thinking that we should consider the appended patch soon.

> I would like to remove the restrictions from creating new namespaces
> however we will either have to have restrictions like the current
> unprivileged mount patches, so we don't surprised root.

Yes, exactly, we need to understand exactly how the resources being
unshared can be updated and how separate the unsharing and updating
really are semantically (per namespace). That's why I wanted to start
floating the document now. I don't think it's something one person can
sit down and write out in one sitting, bc something will be overlooked.

> Or we figure out

> how to ensure we don't have suid applications.

... "how to ensure we don't have suid applications" seems the wrong
level to think at. Rather, for each namespace, what do the tools which
will be privileged to perform updates depend upon?

An obvious example is depending on a file location, i.e. /etc/fstab.
The proper implementation of user mounts solves that. /etc/resolv.conf
is another example, except in this case we have the updating of the
network namespace (kinda) depending on proper updates of the mounts
namespace (and user namespace).

> Given my intuitive understanding of a complete uid namespace it
> fundamentally prohibits suid executables from executing because those
> users simply do not exist in the new namespace. So my hunch is we can
> drop the requirement for CAP_SYS_ADMIN on namespace creation in
> concert with a uid namespace creation.
>
> So my feeling at the moment is that we need to flesh out and complete
> the namespaces we have user, net, pid, user and then come back and
> see what we can do.

I think you're saying "what we're doing is at least safe, so let's wait
to change things."

My assertion is that the current approach is not the safest bc we have
to give unneeded extra authority to a mounts unsharing helper.

> I don't think it makes sense to document a snapshot in time of the
> discussion with unresolved issues in the Documentation directory.

1. The basics aren't going to change, updating your network namespace
will require CAP_NET_ADMIN.

2. These things should really be considered now, bc resulting
implications may have effects on the namespace design. As an example,
the interaction of network namespaces, pid namespaces, netlink sockets,
and audit daemons makes me uneasy.

> Documenting what is and how we got there is fine (that is timeless)
> documenting what could be will likely be out of date before the patch
> is merged into Linus's tree.

Yes you're right. I'm keeping my own list, just turned it into a patch
for getting comment :) But keeping the doc in sync with the code
in several trees would be (near) impossible.

>

> Eric

>From 0e04048d0a22cfd9507487a09ca8d7aa500be1c2 Mon Sep 17 00:00:00 2001
From: Serge E. Hallyn <serue@us.ibm.com>
Date: Thu, 15 Nov 2007 10:47:32 -0500
Subject: [PATCH 1/1] namespaces: introduce CAP_NS_UNSHARE for some namespaces

(purely for comment at this point)

Unsharing and manipulating a namespace can be - depending upon the
namespace and the implications of unsharing - two different things.
But we are using the same capability to authorize unsharing and
manipulating the mounts and uts namespaces.

Using CAP_SYS_ADMIN to authorize namespace unshares means that a program
given just the CAP_SYS_ADMIN file permitted capability also authorizes
the program to manipulate the namespaces - before and after unshare.

Introduce CAP_NS_UNSHARE and use it to authorize unsharing of the
uts and mounts namespaces.

CAP_SYS_ADMIN continues to authorize the unsharing of other namespaces
until the implications are better understood.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
---
 include/linux/capability.h | 4 ++++
 kernel/nsproxy.c | 29 ++++++++++++++++++++++++-----
 2 files changed, 28 insertions(+), 5 deletions(-)

diff --git a/include/linux/capability.h b/include/linux/capability.h
index a1d93da..2660f8a 100644
--- a/include/linux/capability.h
+++ b/include/linux/capability.h
@@ -314,6 +314,10 @@ typedef struct kernel_cap_struct {

 #define CAP_SETFCAP   31

+/* Unshare mounts namespace */
+/* Unshare UTS namespace */
+#define CAP_NS_UNSHARE   32
+
 /*
  * Bit location of each capability (used by user-space library and kernel)
  */
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index 79f871b..a4972fb 100644
--- a/kernel/nsproxy.c

```
+++ b/kernel/nsproxy.c
@@ -114,6 +114,25 @@ out_ns:
  return ERR_PTR(err);
 }

+#define NEED_CAPSYSADMIN_FLAGS \
+  (CLONE_NEWIPC| \
+  CLONE_NEWUSER| \
+  CLONE_NEWPID| \
+  CLONE_NEWNET)
+
+#define NEED_CAPNSUNSHARE_FLAGS \
+  (CLONE_NEWNS | \
+  CLONE_NEWUTS)
+
+static int authorize_unshare(unsigned long flags)
+{
+ if ((flags & NEED_CAPSYSADMIN_FLAGS) && !capable(CAP_SYS_ADMIN))
+  return -EPERM;
+ if ((flags & NEED_CAPNSUNSHARE_FLAGS) && !capable(CAP_NS_UNSHARE))
+  return -EPERM;
+ return 0;
+}
+
 /*
  * called from clone.  This now handles copy for nsproxy and all
  * namespaces therein.
@@ -133,10 +152,9 @@ int copy_namespaces(unsigned long flags, struct task_struct *tsk)
     CLONE_NEWUSER | CLONE_NEWPID | CLONE_NEWNET)))
   return 0;

- if (!capable(CAP_SYS_ADMIN)) {
-  err = -EPERM;
+ err = authorize_unshare(flags);
+ if (err)
   goto out;
- }

  new_ns = create_new_namespaces(flags, tsk, tsk->fs);
  if (IS_ERR(new_ns)) {
@@ -186,8 +204,9 @@ int unshare_nsproxy_namespaces(unsigned long unshare_flags,
         CLONE_NEWUSER | CLONE_NEWNET)))
   return 0;

- if (!capable(CAP_SYS_ADMIN))
-  return -EPERM;
+ err = authorize_unshare(unshare_flags);
+ if (err)
```

```
+  goto out;

  *new_nsp = create_new_namespaces(unshare_flags, current,
    new_fs ? new_fs : current->fs);
--
1.5.1.1.GIT
```

Subject: Re: [RFC PATCH] namespaces: document unshare security implications
Posted by serue on Thu, 15 Nov 2007 17:01:17 GMT
View Forum Message <> Reply to Message

Quoting Serge E. Hallyn (serue@us.ibm.com):
> Quoting Eric W. Biederman (ebiederm@xmission.com):
> > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> > So I think CAP_SYS_ADMIN is a good starting place. It is trivial verifiable
> > that it is safe. So starting there allows us to work on other aspects
> > of the problem for now.
>
> It was a good starting place, but at this point I have two concerns with
> sticking with CAP_SYS_ADMIN:
>
> 1. now that file capabilities are upstream, people may want to
> add just the requisite capability in fP for an unsharing helper
> program. Cedric had mentioned wanting to do that.
> If we are going to switch to unprivileged unshares, then doing
> so later is ok. But if we're going to switch to a custom
> capability later, then that could be seen as an API change
> since users will have to switch the capability on all the
> unsharing programs.
>
> 2. As I pointed out a few times, we can cleanly separate
> unsharing namespace and actually manipulating the resources.
> By requiring CAP_SYS_ADMIN for both unsharing a mounts namespace
> and for performing privileged mounts, any program given the
> authority to unshare is automatically given the authority to
> also completely manipulate the mounts, both in the new private
> namespace and the original namespace (by just not unsharing).
>
> It's even worse with the net namespace, since the privilege
> needed to unshare the namespace authorizes you to update
> *other* namespaces in the system, but *not* network devices!
> But like you say let's stick with established namespaces.

Ok I'm being inconsistent (waffling between talking about not needing capabilities and using a separate capability), imprecise, and overly verbose.

Point 2 above is my key motivating factor.

So to attempt to state a clear, precise goal:

If limited unprivileged updates to a namespace are possible,
then the privilege needed to unshare the namespace should
be as isolated from other privileges as possible.
If limited unprivileged updates are not possible, or if
unsharing implicitly equals updating (*1) then the
privilege needed to unshare should equal that to update
the namespace, not another namespace (*2).

*1: as may be the case with NETNS since the new network namespace
is created empty
*2: i.e. not CAP_SYS_ADMIN to unshare(NETNS) and CAP_NET_ADMIN to update.

-serge

Subject: Re: [RFC PATCH] namespaces: document unshare security implications
Posted by ebiederm on Thu, 15 Nov 2007 18:45:15 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> Quoting Eric W. Biederman (ebiederm@xmission.com):
>> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> Thanks for responding, Eric. Good points.
>
>> > Ok, the following isn't meant so much as a patch as for discussion.
>> > However, this may be a change we want to think about for awhile and
>> > collect opinions and facts. So having this file sitting in the
>> > kernel tree (updated with the results of any discussion we have in the
>> > meantime) may be useful.
>> >
>> > So what do people think? Are we ok using CAP_SYS_ADMIN? Do we
>> > authorize unsharing of each resource using the capability required
>> > to administrate the resource? Do we introduce CAP_NS_UNSHARE? Do

>> > we add CAP_SYS_USHARE, CAP_NET_UNSHARE, and CAP_USER_UNSHARE? Or
>> > do we allow unprivileged users to unshare, trusting that the actual
>> > administration is properly authorized?
>>
>> Well if we ant to sit this in the kernel we need to remove mention
>> of CAP_NS_UNSHARE.
>
> Mostly for now I wanted it to sit in the mailing list :)
>
>> However even there the document below is only an ok first stab at
>> documenting things.
>>
>> The big big big problem are suid executables. If we don't have suid
>> executables and the namespaces only apply to our children we can
>> unshare them all day long and no one cares. If we do have suid executables
>> any messing up of their context that they are not prepared to deal with
>> is a potential security violation.
>
> Ok let's look just at the mounts namespace since that one is complete.
>
> Unsharing the mounts namespace makes no change in mounts context, and
> does not provide the user any additional privilege to make such change.
> So suid executable are safe.

Generally. However we get a stale view, of the mount namespace (which
isn't a big deal because changing mounts is rare) and a stale view
is the equivalent of making a specific change to a namespace from
a suid executables perspective.

I guess I could change /etc/mtab to not reflect the mount namespace
of the initial mount namespace (by running mount), possibly I could
run with an older copy of /dev.

Further you pin filesystems making unmounts difficult and untrackable
with fuser.

I'm not good at coming up with exploits but stale data and state manipulations
that don't manipulate what you think you are manipulating feels like something
a creative person could use to generate an exploit.

A general problem with stale namespaces is that if someone ever does
something stupid you can open the window of vulnerability from just a
moment to years.

>> So I think CAP_SYS_ADMIN is a good starting place. It is trivial verifiable
>> that it is safe. So starting there allows us to work on other aspects
>> of the problem for now.

>
> It was a good starting place, but at this point I have two concerns with
> sticking with CAP_SYS_ADMIN:
>
> 1. now that file capabilities are upstream, people may want to
> add just the requisite capability in fP for an unsharing helper
> program. Cedric had mentioned wanting to do that.
> If we are going to switch to unprivileged unshares, then doing
> so later is ok. But if we're going to switch to a custom
> capability later, then that could be seen as an API change
> since users will have to switch the capability on all the
> unsharing programs.

Ugh. The only capabilities I see that make sense to switch to are
the capabilities needed to make deep changes to a namespace say
(CAP_NET_ADMIN and CAP_NET_RAW be required for the network namespace).

It is a good point that we should work to get the capabilities correct.

> 2. As I pointed out a few times, we can cleanly separate
> unsharing namespace and actually manipulating the resources.
> By requiring CAP_SYS_ADMIN for both unsharing a mounts namespace
> and for performing privileged mounts, any program given the
> authority to unshare is automatically given the authority to
> also completely manipulate the mounts, both in the new private
> namespace and the original namespace (by just not unsharing).
>
> It's even worse with the net namespace, since the privilege
> needed to unshare the namespace authorizes you to update
> *other* namespaces in the system, but *not* network devices!
> But like you say let's stick with established namespaces.
Yes.

> So while I started the original email just wanting some discussion, now
> I'm actually thinking that we should consider the appended patch soon.
>
>> I would like to remove the restrictions from creating new namespaces
>> however we will either have to have restrictions like the current
>> unprivileged mount patches, so we don't surprised root.
>
> Yes, exactly, we need to understand exactly how the resources being
> unshared can be updated and how separate the unsharing and updating
> really are semantically (per namespace). That's why I wanted to start
> floating the document now. I don't think it's something one person can
> sit down and write out in one sitting, bc something will be overlooked.

Definitely. Especially the nasties with suid executables.

>> Or we figure out
>> how to ensure we don't have suid applications.
>
> ... "how to ensure we don't have suid applications" seems the wrong
> level to think at. Rather, for each namespace, what do the tools which
> will be privileged to perform updates depend upon?

Totally. It isn't the unix way to kill suid applications.
suid applications are the Achilles heel in a lot of ways for
applications that want to enhance the unix API as they make it much
much harder. Plan9 gained tremendous support when it dropped the
ability. I'm not at all certain what I think of fine grained
filesystem capabilities. I honestly think that is moving in the wrong
direction. Generally we can solve the same problem in user space with
servers on a system that start out with privileges have a narrow
channel for communicating with the outside world (reducing their
amount of vulnerability), and that drop privileges as they go.

If we try to limit ourselves to what the privileged tools actually
depend on (instead of what they can depend on) we are setting
ourselves up for a world of hurt when we miss something.

> An obvious example is depending on a file location, i.e. /etc/fstab.
> The proper implementation of user mounts solves that. /etc/resolv.conf
> is another example, except in this case we have the updating of the
> network namespace (kinda) depending on proper updates of the mounts
> namespace (and user namespace).
>
>> Given my intuitive understanding of a complete uid namespace it
>> fundamentally prohibits suid executables from executing because those
>> users simply do not exist in the new namespace. So my hunch is we can
>> drop the requirement for CAP_SYS_ADMIN on namespace creation in
>> concert with a uid namespace creation.
>>
>> So my feeling at the moment is that we need to flesh out and complete
>> the namespaces we have user, net, pid, user and then come back and
>> see what we can do.
>
> I think you're saying "what we're doing is at least safe, so let's wait
> to change things."

Largely. Safe at least from the well understood perspective.

> My assertion is that the current approach is not the safest bc we have
> to give unneeded extra authority to a mounts unsharing helper.

True. Safest and most flexible is to figure out how to remove the
need for capabilities at all.

>> I don't think it makes sense to document a snapshot in time of the
>> discussion with unresolved issues in the Documentation directory.
>
> 1. The basics aren't going to change, updating your network namespace
> will require CAP_NET_ADMIN.

Agreed.

> 2. These things should really be considered now, bc resulting
> implications may have effects on the namespace design. As an example,
> the interaction of network namespaces, pid namespaces, netlink sockets,
> and audit daemons makes me uneasy.

Yes. I think that sounds like a fun one to sort through.

>>From 0e04048d0a22cfd9507487a09ca8d7aa500be1c2 Mon Sep 17 00:00:00 2001
> From: Serge E. Hallyn <serue@us.ibm.com>
> Date: Thu, 15 Nov 2007 10:47:32 -0500
> Subject: [PATCH 1/1] namespaces: introduce CAP_NS_UNSHARE for some namespaces
>
> (purely for comment at this point)

I'm not at all ready to consider the implications of a new capability
at this point.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers