Subject: Re: [RFC] [PATCH 4/8] user namespace: enforce CAP_NS_OVERRIDE for cross-namespace kill
Posted by serue on Wed, 12 Dec 2007 19:22:49 GMT
View Forum Message <> Reply to Message

Quoting Serge E. Hallyn (serue@us.ibm.com):
> >From 62e6efe435a24f430e28f2398f374cef197b4964 Mon Sep 17 00:00:00 2001
> From: sergeh@us.ibm.com <sergeh@us.ibm.com>
> Date: Thu, 29 Nov 2007 08:18:16 -0800
> Subject: [RFC] [PATCH 4/8] user namespace: enforce CAP_NS_OVERRIDE for
cross-namespace kill
>
> Require CAP_NS_OVERRIDE to 'kill' across user namespaces.
>
> Signed-off-by: Serge Hallyn <serue@us.ibm.com>
> ---
> kernel/signal.c | 5 +++++
> 1 files changed, 5 insertions(+), 0 deletions(-)
>
> diff --git a/kernel/signal.c b/kernel/signal.c
> index 787521e..a06dcc2 100644
> --- a/kernel/signal.c
> +++ b/kernel/signal.c
> @@ -534,6 +534,11 @@ static int check_kill_permission(int sig, struct siginfo *info,
>   error = audit_signal_info(sig, t); /* Let audit system see the signal */
>   if (error)
>   return error;
> +
> + if (current->nsproxy->user_ns != t->nsproxy->user_ns
> +   && !(capable(CAP_KILL) && capable(CAP_NS_OVERRIDE)))
> +   return -EPERM;
> +
>   error = -EPERM;
>   if (((sig != SIGCONT) ||
>   (task_session_nr(current) != task_session_nr(t)))
> --
> 1.5.1

This patch has a sadly obvious, but (just as sadly) hard for me to
reproduce pair of bugs which Cedric found (thanks).

The following patch *should* be a correct fix, but as it's the fourth
version I'm going to keep testing for awhile.

In the meantime, the last three patches in this set are really
independent of the rest, so I plan to send them individually later today
with cc:s to the appropriate maintainers for their review.

thanks,
-serge

>From 558e37c30d2047e040e4e2d40ed87c96cd78183f Mon Sep 17 00:00:00 2001
From: sergeh@us.ibm.com <sergeh@us.ibm.com>
Date: Thu, 29 Nov 2007 08:18:16 -0800
Subject: [PATCH 1/1] user namespace: enforce CAP_NS_OVERRIDE for cross-namespace kill (v4)

Require CAP_NS_OVERRIDE to 'kill' across user namespaces.

If the signaling task is exiting, then current->nsproxy is
NULL. Since we are only notifying a parent of our death,
we permit the signal.

If the target task is exiting, our signal doesn't matter
anyway.

Signed-off-by: Serge Hallyn <serue@us.ibm.com>
---
 kernel/signal.c | 23 ++++++++++++++++++++++

diff --git a/kernel/signal.c b/kernel/signal.c
index 06e663d..75c4d00 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -531,9 +531,32 @@ static int check_kill_permission(int sig, struct siginfo *info,
   return error;

  if (info == SEND_SIG_NOINFO || (!is_si_special(info) && SI_FROMUSER(info))) {
+ struct nsproxy *nsproxy;
+ struct user_namespace *my_user_ns = NULL, *t_user_ns = NULL;
+
   error = audit_signal_info(sig, t); /* Let audit system see the signal */
   if (error)
    return error;
+
+ /*
+   * if current->nsproxy is NULL, then we are exiting and
+   * are just sending an exit signal to our parent.
+   * Uid may be wrong under certain circumstances, but
+   * global init shouldn't care, and a container creation
+   * program should know what it is doing.
+   * If target is exiting then it doesn't matter anyway.
+   */
+ rcu_read_lock();
+ nsproxy = task_nsproxy(t);
+ if (nsproxy)

```
+   t_user_ns = nsproxy->user_ns;
+  rcu_read_unlock();
+  if (current->nsproxy)
+   my_user_ns = current->nsproxy->user_ns;
+  if (my_user_ns && t_user_ns && my_user_ns != t_user_ns
+   && !(capable(CAP_KILL) && capable(CAP_NS_OVERRIDE)))
+   return -EPERM;
+
   error = -EPERM;
   if (((sig != SIGCONT) ||
    (task_session_nr(current) != task_session_nr(t)))
--
1.5.1
```

_____

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