Subject: Re: User namespaces and keys
Posted by ebiederm on Wed, 23 Feb 2011 20:55:01 GMT
View Forum Message <> Reply to Message

Casey Schaufler <casey@schaufler-ca.com> writes:

> I confess that I remain less well educated on namespaces than
> I probably should be, but with what I do know it seems that the
> relationships between user namespaces and LSMs are bound to be
> strained from the beginning. Some LSMs (SELinux and Smack) are
> providing similar sandbox capabilities to what you get from user
> namespaces, but from different directions and with different
> use cases.

Casey I won't argue about the possibility of things being strained, but I think if we focus on the semantics and not on the end goal of exactly how the pieces are to be used there can be some reasonable dialog.

>From 10,000 feet an user namespace represents one administrative domain. uids, gids and other external tokens are all controlled by a single group with a single security policy. In that single administrative domain things like nfs are expected to work without translating uids and gids. Before the implementation of user namespaces a single kernel could not even express the ability of dealing with multiple user namespaces simultaneously (nfs has usually punted and said despite being multiple machines you still have to be in the same administrative domain so the same user identifiers can be used throughout).

>From that principle we have the concept that use assigned/visible identifiers uid, gid, capabilities, security keys?, security labels? logically belong to the user namespace.

Which means in implementing there are two pieces of work in implementing the user namespace.

- Find all of the interesting comparisons and change them from
  "if (id == id)" to "if (userns == userns) && (id == id)".

- Potentially define and handle what happens when you mix user
  namespaces.

I think for the first round of this we simply want to define lsms and the user namespace as not mixing or the lsm interfaces only being visible if you are in the initial user namespace. Thus reserving the definition of what happens when you have lsms and multiple user namespaces as something we can define later. I expect the proper definition is something that would allow nested lsm policy.

Regardless. The namespaces are all about making the identifiers that processes deal with local to the namespace, while the underlying object manipulations should not care.

The big advantage of the user namespace is that it is the only way I can see to get us out of the bind we are in with suid root executables, where we can not enable new features to untrusted applications that can change the environment for a suid root executable, because it might confuse those suid root executables to misusing their power. Once inside a user namespace nothing has dangerous powers because even a root owned process with a full set of capabilities is less powerful than a guest user outside of the namespace. No process having dangerous powers allows us to enable unsharing of other namespaces and potentially other things that today are restricted with capabilities only because they can be used to confuse a privileged executable to do something malicious.

Eric

____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs