

Re: compile+link Fujitsu Linux

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2008-02/msg00365.html>

- *From:* Robert <no@xxxxxx>
 - *Date:* Wed, 06 Feb 2008 21:06:55 -0600
-

On Wed, 6 Feb 2008 00:19:14 -0800 (PST), Richard <riplin@xxxxxxxxxxxxxx> wrote:

On Feb 6, 7:01 pm, Robert <n...@xxxxxx> wrote:

Well, no it doesn't do that. Not even close. But you still haven't explained how your mechanism is 'better'. Given equal slackness in permissions and such and users can access compilers then how is a library any different ? Tools can push and pull stuff out of libraries.

My security is better because it does not depend on file system permissions. It doesn't depend on application code, either.

The goal of security is to prevent unauthorized people from seeing or changing certain data. I do that by protecting the DATA; you do it by preventing them from running programs.

As an analogy, suppose we are parents who want to prevent our kids from watching sex movies on TV. You would do it by locking up the remote control, which is easily bypassed with a universal remote. I would do it by telling the cable box to play the movies only for designated users.

You claimed that your way was superior and the best that you have come up with is 'less ugly'.

Simpler deployment, can be checked by auditing tools, faster loading, more reliable because mid-run aborts due to missing program are impossible.

What makes you think that users have write access to the program directory ? I can drop in

Re: compile+link Fujitsu Linux

another program.

If they have read access, they can copy your whole directory structure to their home directory, change permissions all they want, and run their local copy to find out the boss' salary.

Well, actually, no they couldn't. I already told you that particular programs can be set to be owned by the application administrator with a group id of the trusted group and permissions of 640. It is only if you are a member of the trusted group that give read access to run (or copy) that program. Users can be members of several groups so access can be very granular. Users not a member of that group will get a 'program not found' if they try to select it from the menu.

I assumed the files had world read permission. You stopped that approach.

So, you would concede then that I have described a mechanism that would not be able to be used by your system the way that you have it and that this mechanism does protect the programs.

My system? I'd like to take credit for inventing Unix and ELF, but I believe Thompson and Ritchie did it first.

Thank you for starting to notice that some of us know what we are doing.

Yes, we do.

If the library was required at start up (as in your mechanism) then the application would fail to do anything for users not in _all_ the trusted groups. Also you gather several programs into one

Re: compile+link Fujitsu Linux

Re: compile+link Fujitsu Linux

library so
you have less, or no, usable granularity.

Granularity is in the called programs. It doesn't matter how they are packaged.

You did not follow that discussion then. If several programs are packaged together in a library then the group permission restrictions apply to all in the library.

Oh no, not file permissions again.

My
mechanism
is oriented
around data,
not
programs. If
I want to
protect
month-end
figures
in the
accounting
system, for
example, I
protect
those
numbers in
the
database.
Unauthorized
users cannot
change
them by
running the
monthend
program, by
writing their
own
program nor
by running
a database
utility. Your
notion of
security by
controlling

Re: compile+link Fujitsu Linux

access to
programs
was
obsolete
twenty
years ago.

Just change the subject then.
You were trying to say that
using
libraries was 'more secure'.
You tried to say that
'home-grown' was
not secure.

Now you just change the
subject to something
different because you
failed to establish your
unsupported and flawed
assertions.

YOU introduced application security, not I.

No, that is wrong. I was saying that "the CALL dataname
feature is
invaluable ..".

Then, out of the blue, you changed this to: "Now you're
advocating
home-grown security."

You said "invaluable as it means that the menu program can use a data file to
define what
is available in a particular application, or even to the particular user." That's
application security,

No. Wrong. You may have thought that security was the only reason for
this, but in my framework systems the 'user' identifies sets of data

Re: compile+link Fujitsu Linux

Re: compile+link Fujitsu Linux

files as well as applications because the systems are multi-company as well as multi-user.

I run multi-company, multi-user all day, every day. When I switch to another user id with su, it runs a script that sets environment variables to that 'user' and company's directories. The 'user' is typically a development version and stage such as dev, integration test, system test, UAT, prod. When I start an application, it runs another script that sets more variables specific to the application.

The important point here is that ALL applications are controlled the same way. If the environmental stuff were hard coded in the application, each application would be controlled differently.

One company may have bespoke programs that suit a particular need and the users in that company may have additional programs that are not available to the same application run for another company.

We put the menu (and error messages) in a database, keyed by company, same as you do. One off programs and scripts are run by us. If the user needs access to it, it's not a one timer, it gets added to the menu and library.

and is irrelevant to dynamic loading. You could do the same if all programs were in one library.

Well certainly your library could be used but would provide `_no_` advantage and would not provide the advantages that I have already given. For example if every program were in one library then that would include the bespoke programs and a reissue of those would require `_EVERY_PROGRAM_` to be retested everytime to meet adequate standards of reliability.

Nonsense. We have 30,000 developers with their fingers in the pie and we don't do that. It sounds like you don't have version control, or don't trust it.

We do run a regression test on the whole system for major releases, which are every two months. That's automated.

Security belongs in the called program, not the menu program. A user could write his own program that calls yours.

Which is not an advantage for your library because he could call that too.

Re: compile+link Fujitsu Linux

So? The called program issues an Unauthorized message and exits.

The moral is: don't assume users are too dumb to hack your system.

It seems that you allow them to break your [security] by merely overwriting one file.

My security isn't in files nor file system directories. It's not even in the passwd file.

If I DID
want to
limit access
to a
program
function,
the security
check
would be in
the
function,
not the file
it is
packaged in
nor the
calling
program.
Relying on
Unix file
system to
do
application
security is
weak. For
example, if
the
setgroupid
bit of a
program
file is set, it
executes
under the
owner's
group id
rather than
the user's.

Re: compile+link Fujitsu Linux

Re: compile+link Fujitsu Linux

Another straw-man. Why
would I have setgroupid set
?

Because a user set it.

Excuse me, but how did a user get to set this ?

He would have to trick someone in the group to set it. Are privileged users'
.profiles
world writable? Is root's? How about shell logins?

And this is different from your library system in what way ?

Instead of addressing your claimed superiority you merely ramble on
about something else entirely unrelated, and straw-men, in the hope
that it may bamboozle. Perhaps that is your normal 'arguing' skills,
you win if they give up so it doesn't matter what the content is.

It IS relevant to you, since your security depends on the Unix file system. If I can
change a group member's (hidden) profile or shell login, I can get his group permission,
even though I don't belong to the group. His script could copy the main menu (or ksh) to
my home, give it world execute and setgroupid. Next time I run it, I'm in the privileged
group.

.