

Re: How to check UID of process on the other side of local TCP/UDP connection

Source: <http://www.derkeiler.com/Mailing-Lists/securityfocus/focus-linux/2006-11/msg00032.html>

- *From:* "Greg Metcalfe" <metcalfe@xxxxxxxx>
 - *Date:* Mon, 27 Nov 2006 18:31:42 -0800
-

On Friday 24 November 2006 10:11, rainmailbox2001-ola@xxxxxxxx wrote:

Hello.

I have the following situation:

- Client communicates with server via TCP or UDP.
- Both client and server are on the same local host.
- Server runs with root privilege.

Now, client connects to server. Server has to check uid of the client. How it can be done? I need a solution that can be ported to all modern Unix and Linux systems.

See the `getuid(2)` and `geteuid(3p)` man pages. Whatever language you're writing in probably has the system call.

The most simple solution I came with is as follows:

1. Client connects to server.
2. Server asks client to create file with random name, for example `/tmp/check.6723`
3. Client generates the file.
4. Server checks the owner of the file.

That's just **nasty**.

The owner of the file is the UID under which client is running.

But

the problem is that it requires some additional communication between server and client. My programs can communicate hundreds times a second so creating, checking and removing the file is a big performance issue.

Yep, you need IPC.

Do you have any ideas how this local authentication can be achieved in some different way?

You want to use a UID for authentication???

I

was also thinking about using Unix sockets for communication, but it seems that they also lack any mechanism for authenticating the client. Anyways, I would prefer to stick with TCP/UDP, because this is what my programs use already, and I don't really want to change everything to Unix sockets (unless of course Unix sockets are the only good way to resolve my problems).

Unix sockets really would make more sense. This is what they're *for*. Why have another TCP/UDP service, running as root, if there's any way to avoid it? Even if it's only a localhost listener, it's possibly an easier elevation of privilege or DoS attack. Using the stack is also more expensive.

Plus, you have `socketpair(2)` to simplify your coding. A quick Google will give you the details on that. It should simplify your code, no matter the language. And you should have an implementation, as it's another system call.

As this is a security list, it's pretty much obligatory for me to always recommend simplification, if only to maintain the habit. ;)

You have to realize that sockets only provide a means of communication. What you *do* with your communications link, whether that be authentication or streaming the digits of pi, is up to you, and has to be coded.

Thanks,
Ola

Have fun,

Greg Metcalfe