WeGet letters

by Michael W Lucas

Dear Journal Letters Column,

I have to integrate this new hardware doohickey into all our authentication systems on all our hosts, no matter which operating system they're using. It's harder than I thought. The differences between OpenSolaris and FreeBSD and Linux and AIX and HP/UX and all the other Unixes are all tiny — but taken together they seem huge. Is there an easier way to do this?

—Perplexed

Perplexed,

I recently had the chance to go to my first concert in three years—Nine Inch Nails, Nitzer Ebb, and Ministry. I kept myself safe, with my stick-on mask and ear plugs and eye goggles and full-body bunny suit, not to mention the barbed wire halo, but at least I was able to attend this glorious outpouring of incendiary rage and righteous betrayal and the kind of defiant bitterness that gives me reason to crawl out of my cage and scrape the bile off my teeth every morning.

That approaches how I feel about PAM.

"But you have to have PAM" people shout. "It's a necessary evil, it's a standard!" Nope. It isn't. It looks like a standard so long as you don't look at it. Sun Microsystem, the well-spring of NFSv2 and Java and many other seductive immortal nightmares, offered it up to the public in the hope it would be adopted. It was. Sun did not organize an Interop as they did for NFS or maintain Java-style control. Instead, they left everybody free to implement it in their own preferred, slightly different manner. Yes, yes, the Common Desktop Environment became a standard back in the 1990s and mentioned PAM integration, but any standards that coexisted with Saturday morning cartoons and the ankylosaurus should not be considered relevant today. The closest thing we have to a PAM standard is in the document *X/Open Single Sign-on Service (XSSO) — Pluggable Authentication Modules* from an attempt to nail it into POSIX, but folks followed about ninety percent of that, and we all know that ninety percent compatible equals zero percent interoperable.

We don't even have a standard language. Is it a PAM policy or a chain? Rules or modules? Types or rules? Even if you read the documentation, you can only follow it through intuition and good karma.

The Journal's editors saw fit to have some PAM apologist write a piece for this issue. I won't glorify it by calling it an "article," because he probably cut-and-pasted snippets of his book for it and shamelessly ended it with a plug for same. I'm not saying that he'd do

anything for a buck, but if I was unlucky enough to be near him, I would absolutely mention in the sort of voice I normally reserve for screaming back at Al Jourgensen that "ethics" are a thing even in information technology and that he's doing all this in public where anybody who exerts a morsel of effort can figure out his little scam. Fortunately for him, nobody cares enough about his feeble antics to bother.

Forget standardization. Not everything *has* to be a standard — otherwise, we couldn't make the mistake of inventing new things. Look at how PAM works. You grab these shared libraries, never mind where they came from or how carefully they've been audited, chain them together, and force them to collectively decide how your authentication is going to work? We all know how access control lists work. *This* is allowed. *That* is not. You carefully define the characteristics of permitted activity and block everything else. What you do not do is implement a wishy-washy system where rules can say things like "yes, but only if everybody else agrees" or "I'm gonna veto it, but y'all go ahead and vote."

Voting? Security is not a democracy! It's not even a republic.

Not everything has to be a standard — otherwise, we couldn't make the mistake of inventing new things.

Not that PAM holds a proper vote. It's more like a bunch of drunk programmers deciding what to order for dinner. You go around the table, sure, but finally the one with the deepest understanding of compiler internals picks whatever will give everyone the worst hangover possible. The others get to pick a couple of side dishes and maybe ask for a pack of fortune cookies, even though the cashier keeps reminding everyone that they do fortune saganaki because they're a Greek joint and *your fate is always delicious*.

How is that access control, especially without wonton soup?

Fine. Fine. Here we are.

But another thing—debugging. I fully understand that all debugging boils down to scattering print statements throughout the code and watching it go wildly astray, but PAM doesn't even have a standard way to do that. Maybe debug statements will work. Perhaps you can use PAM's "echo" module and spit stuff back at the user, which will absolutely never terrify that guy from Shipping & Receiving who needs three tries and divine intervention to successfully log onto the menu-based inventory system. He'll be fine. Pinky swear.

So you use pam_exec and write a little script that dumps information to a log file, or maybe even into logger(1) and straight into the system log. Using a shell script in your authentication system doesn't guarantee you'll get broken into, especially if they're extremely simple, but shell scripts have this horrid tendency to grow and every line of code is a vulnerability. You might as well write a little Perl script that checks authentication credentials against a Microsoft Excel spreadsheet over the network.

Wait—the PAM apologist already suggested doing exactly that?

Time to lower my standards. Again.

But, again, here we are. PAM is the standard that isn't. We're stuck with it.

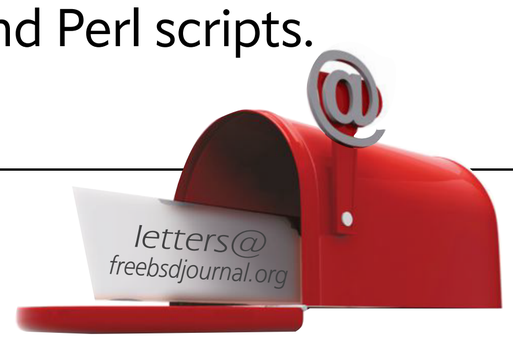
The only consolation I can offer is that your impressions are valid. Nothing is compatible. Everything uses its own language. I'm told that the Pope declared that time spent

configuring PAM counts as time served in Purgatory, however, so be sure to fill out your time sheet correctly.

Hope? Yes, I have hope. I hope is that systemd swallows Linux-PAM and OpenPAM becomes the Last Stack Standing. Perhaps then we can have an authentication system designed by sober people who know how to order fortune cookies.

With our luck, though, we'll get one involving spreadsheets and Perl scripts.

Have a question for Michael?
Send it to letters@freebsdjournal.org



MICHAEL W LUCAS is the author of *Networking for System Administrators*, *$ git commit murder*, and many others. His new books include *OpenBSD Mastery: Filesystems* and *Prohibition Orcs*. Get the entire interminable list from his SNMP OID or at https://mwl.io.