LCA: The security panel

By **Jonathan Corbet**
January 21, 2009

The linux.conf.au security miniconf hosted a number of talks on specific security technologies, many of which have been covered here in the past. The final event of the day, though, was a panel discussion covering a wide variety of security issues. Panellists Casey Schaufler (who also doubled as moderator), Russell Coker, James Morris, Z. Cliffe Schreuders, and Kentaro Takeda discussed module stacking, increasing the use of security technology, authoritative hooks, and more.

Module stacking was the first topic of interest. "Stacking" refers to the practice of loading more than one security module, allowing each of them to participate in security decisions. The technique has its appeal; it would allow more tightly-focused modules to be written and mixed together in interesting ways. But stacking of security modules is not currently supported in the Linux kernel - a situation which does not appear to be set to change anytime soon.

Casey, who had raised the issue, answered his own question by saying that he would like to see module stacking supported; it would add to the flexibility of the system. His preferred solution would involve the creation of a special security module which would arbitrate between all the others, deciding which modules get to make decisions in each specific situation. As far as your editor knows, this stacker module does not actually exist yet.

Russell's response was simpler: he would like to see a reasonable number of users actually running with a single security module first. Once that problem has been solved, one can move on to more complicated tasks.

James raised the issue of the "composability problem": the combination of security technologies in ways not anticipated by their designers can lead to unpredictable results. People working on security technologies *hate* unpredictable results. The SELinux developers tried to deal with this problem by turning SELinux into the one true security module (your editor's term, not James's), so that any security situation could be addressed within a single module. This aspect of SELinux is not really being used, though.

Cliffe's response was that stacking should be allowed, if only to discourage developers from adding parallel sets of hooks to support their own security technologies. James responded, though, that many security-related modules (integrity management, say, or malware scanning) really should have their own API. Kentaro noted that the TOMOYO Linux developers really want their work to be able to coexist with other modules, so he would like to see stacking supported as well.

From there, your editor asked the panelists to follow up on Russell's point: what is it going to take to get people to actually use the security technologies that we have now? A security module does little good if frustrated system administrators simply turn it off as soon as it gets in their way. Casey responded that it was unfortunate that the first security module made available for widespread use (SELinux) was such a complex one. A lot of people really don't need all of the capability which is provided by SELinux. A set of smaller, more understandable security modules would have gained acceptance much more easily. SELinux is far too monolithic; there is no easy way into it.

Russell, instead, suggested that we should look at the history of security. Once it was accepted that all important processes would run as root. Over time, it has been made clear that this is a bad idea, and various system daemons have been moved to other user IDs. Ill-advised practices like running IRC clients as root have been banned. It has taken a long education process to get to this point; this process will have to continue for technologies like SELinux. James agreed that time was required, and noted that, over time, use of SELinux is increasing. Some simple things, like getting administrators to shift SELinux to permissive mode when they run into problems rather than turning it off altogether, can also help in this regard.

In the longer term, though, there is still a need for higher-level tools. The current SELinux policy interface is really the assembly language of (SELinux-based) security; most users should not have to deal with the system at that level. Cliffe agreed, saying that blaming users for turning off security is the wrong thing to do. It is the fault of the security developers, who have not made their tools sufficiently easy to use. Security must be built using higher-level abstractions which users can understand; the technology he is working on (FBAC-LSM), is designed with this goal in mind. Kentaro added that most users don't want to have to think about security; it needs to be implemented so that they don't have to.

From there the panellists went into a rather cloudy discussion of cloud computing. James, after asking what that term really meant, noted that there are useful things to be done in this area, and that Linux offers a number of useful technologies, such as namespaces, which can help. There is, though, a lot of work to be done. Cliffe added that the infrastructure is there for people who want to work on secure cloud computing, but that module stacking would make it easier. Kentaro stated that cloud computing is, in fact, one of the core targets for his work; there is a lot of space for Linux here. We do, however, need to be sure to avoid creating single points of failure, which can bring the whole thing down. Casey's take on this topic was that cloud computing is likely to bring cryptography back to the forefront of security research; when all of your data is on other people's servers, that data needs to be well protected.

Russell took a different tack, noting that the security of a number of current cloud offerings is substandard. They often provide distributions which no longer receive security support, and they provide lots of unpatched software. They are insecure by default and "ripe for harvesting," but it is not easy, in such environments, for even a relatively high-clue user to figure out how to secure things. The real problem, he says, is that there is no business model for better security, so "cloud" providers are not investing in that area.

A member of the audience asked about authoritative hooks. These hooks were a contentious issue early in the development of the Linux security module architecture. LSM is current designed to allow restrictive hooks only: a security module can only make policy tighter than basic Linux discretionary access control would allow. The thinking is that, with restrictive

hooks only, a buggy security module cannot make things worse than they were before. Authoritative hooks would, instead, let a security module empower a process to do things which would not otherwise be allowed.

Casey reiterated the history behind the current "no authoritative hooks" policy, adding that the kernel developers also feared that authoritative hooks would make the LSM API more suitable for abuse by binary-only modules. Indeed, he says that was the primary reason for disallowing those hooks. But this policy has not slowed down proprietary security modules, and, at this point, a model allowing authoritative hooks would be better. Making that change would be "a really big deal," though. Russell agreed that the "irrational fear" of authoritative hooks remains widespread, but the reassurance provided by their absence may be worth it in the end. Both Cliffe and Kentaro thought that interesting things could be done with authoritative hooks, and that it would be a good time to review just how Linux security modules work.

This policy has not slowed down proprietary security modules, and, at this point, a model allowing authoritative hooks would be better. Making that change would be "a really big deal," though.

There was a brief discussion on the feeling that the LSM API is too heavily oriented toward the needs of SELinux. James agreed that it was "SELinux-shaped," but noted that this was a natural result of the fact that SELinux has been the only user of the API for most of its history. Casey noted that things have recently been changed to support the needs of his SMACK module. There have also been some new hooks added to support pathname-based modules like TOMOYO Linux and AppArmor.

Going back to another point raised by Russell, a member of the audience asked what distributions should do once they go past their end of life. Should a system with an unsupported kernel refuse to boot, or, at least, refuse to bring up network interfaces? Russell came back with the obvious response: how would one then update such a system? Casey pointed out that there are an awful lot of routers out there running old, unsupported software. The Internet, he says, is made of expired systems. Russell suggested that ISPs should, perhaps, enforce the use of supported software, and that, maybe, governments could compel such behavior. Cliffe noted that all of this really poses another usability problem; what we really need to do is to make it easy to run a current system. Quite a bit of progress has already been made in this direction.

The final topic had to do with "security mythology," things that "everybody knows" improve security but which really don't. Forced password rotation was one such idea. Casey said that, for some 20 years, everybody "knew" that security meant strong cryptography. There's no real way to address such things except as a people problem. Russell added that there's often no way to know what the consequences of security rules are. James said that there is a real need for technical people to push back against silly security rules. He likened the problem to the early adoption of Linux, where people with clue simply deployed it, then asked for forgiveness later. Cliffe's point of view is that users do not really know when they are being asked to make security decisions, so they don't really know when their actions may be putting their security in peril. And Kentaro agreed, noting that we need to find ways to provide more information to users about what their security technology is really for.

Thereafter the panel broke up, and the PGP key signing party (done, no doubt, in a highly secure manner) began.