RIK FARROW

editor's thoughts (a.k.a. musings)

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of UNIX System Security and System Administrator's Guide to System V.

rik@spirit.com

WELCOME TO THE SIXTH SECURITY edition of ;login:. Like past editions, this one contains articles about security topics that I consider to be among the most important current issues. I want to thank the authors who wrote for this edition. It is the authors who provide the content. All I do is find them and cajole them into writing.

This issue also includes the summaries of the 13th USENIX Security Symposium.

The Security Symposium started out with what some people considered a depressing keynote. Earl Boebert, of Sandia National Laboratories, and one of the authors of Multics, provided a rather pessimistic look at operating system security. I tend to agree with a lot of Boebert's assertions, and you can read them for yourself in the summaries.

You can also read Jonathan Shapiro's response to the keynote. Shapiro points out that there were some very good reasons why Multics did not succeed in the marketplace, outside of the "crap in a hurry" that Boebert mentioned in his keynote speech. Again, you can read Shapiro's thoughts on this topic for yourself.

I have often written about the failures of operating systems in my Musings columns. Largely based on the goals of Multics, operating systems were, at least in theory, supposed to protect a system against poorly written software—that is, logic or programming faults in software should never compromise the security of a system. The operating system should encapsulate the faulty process and prevent software flaws from changing the overall state of the system. As we all know, this is not how operating systems work.

Instead, what we see are systems that are exploited via a process that displays email for a user,or ones that permit a non-privileged user to become a privileged one, and totally compromise the security of a system.

I believe there really are two questions to ask about the future security of our operating systems. First, is it possible to build an operating system that is really secure and can be used by anyone? Second, do we have the will to end the current fiasco, and actually begin using secure operating systems?

Some people might argue that today's operating systems are secure. The Linux Security Module in the 2.6 series of kernels does provide hooks for adding much more comprehensive access controls than exist without the LSM. LSM does provide support for more control, but at the cost of complexity. These same hooks appear in FreeBSD 5, and come with the same level of

complexity. OpenBSD takes a different approach, based on profiling processes and systrace.

All BSD versions support the `jail()` system call for isolating processes in a changed root environment that includes process isolation and some control over the network environment as well. But a proper jail setup also means proper firewall configuration. The jail only becomes reliable at the network level in cooperation with configuration that is external to the jail. The jail also does not prevent against CPU DoS attacks, because it does nothing about process scheduling (read Kirk McKusick's article in the August 2004 issue of *;login:* for more about BSD jails). Memory can also be depleted from within a jail.

Sun has borrowed from the FreeBSD jail concept to create zones in Solaris 10. Each zone shares the same operating system, similar to the FreeBSD jail. But Sun's implementation goes a bit further, by allowing separate resource allocation for each zone. Using resource management, CPU scheduling and memory usage can be limited for each non-global zone (every zone except the default, first zone is non-global and unable to see other zones). Sun has apparently solved the resource depletion problem found in the jail approach. I've heard that AIX and HP-UX use logical partitions to do something similar.

All of these approaches represent attempts to retrofit security on existing operating systems so that they can transparently support existing software. All the sysadmin has to do is leap through a few hoops, carefully and without making any egregious errors, and things will work. Hopefully.

Microsoft's current security model is so broken it deserves mention. There are way too many privileged processes running. And putting IE and its related HTML-rendering engine (used when reading email) on every system means that compromise is just an email message away. If ever an application screamed out to be put in a really effective jail, Internet Explorer is that application. Take IE, add default Administrator privileges for the first user account on every XP system, and IE becomes a root compromise, even when patched during Microsoft's (mostly) monthly patch announcement.

Now I invite you to follow along as I imagine a different world, a world where the operating system actually did prevent software, and even bad configuration, from creating root compromises. Let's start out with the type of system that Bill Cheswick alluded to in his invited talk, "My Dad's Computer." Cheswick's dad's computer had become like a lot of other Windows systems—so loaded with viruses, adware, spyware, and plain old cruft that it was barely usable. I've heard of other people buying a new Windows system and not connecting it to the Internet just so they could use a wordprocessor or spreadsheet without being interrupted by obscene popups every minute or so.

Most people need a simple desktop system that can do three things securely: play games, do Internet stuff, and handle simple office tasks. I mention playing games first because games have been driving the PC industries' quest for ever greater graphics performance, and because most of the people I see using laptops on airplanes are playing Solitaire. Games must be important. The fantasy OS must allow users to play games without affecting the overall state of the system (other than the DoS and heat discharge caused by pegging the CPU and graphics systems while displaying millions of 3-D polygons per second as sub-woofers shake the room).

The Internet stuff is much more of a problem. This is in which the operating system must maintain a lockbox where Web browsers, mail readers, and various IM tools can wreak all the havoc they want to without impacting the rest of the system. Of course, if users can continue to install software and plugins, even the lockbox will become so infested as to become unusable. So the user will get a button labeled "Clean up Internet" which cleans his own Internet lockbox, restoring it to a usable condition. Remember that I am waving a magic wand here, so I don't have to concern myself about cleaning up the cookies file, and preserving the state that the user actually cares about.

Some vendors like to use Web browsers as the mechanism for installing patches. If one can trust the operating system, signed patches can still be installed—by some service running outside the lockbox, after verifying signatures on the patches. Ideally, no patches would be required. But reality sometimes intrudes into even the wildest fantasies.

The final lockbox contains the user, her files, desktop, and office applications. These applications are sadly lacking in the ability to execute macros that might be included in documents or spreadsheets. The Internet lockbox can leave email and files in a directory where the user can access them. But no file stored anywhere in the user's file space can ever be executed.

If desktop systems were actually designed to function as business machines, things would be a lot simpler in this fantasy. The desktops could be diskless workstations where the user could never install any software, including viruses, spyware, adware, or games (sorry about that). Imagine centralized backup, identity management, software updates, and no more touching desktop systems. Sun's SunRay comes close to this.

Servers are headless. No fancy GUI software, nothing but command-line interfaces run using SSH. For the weak in sysadmin skills, fancy GUIs can be installed on desktop systems to issue the command lines used to configure and maintain the servers. No users except

the administrator ever log in to the server. There are no accounts for them. The server does not include any software other than what is required for the operation and maintenance of installed services.

The server applications run in their own lockboxes, again preventing them from inadvertently damaging the systems they are hosted upon. The lockbox includes resource management controlling how the service behaves. For example, Web and SQL servers only accept connections; they never make outgoing connections. DNS servers send and receive packets on port 53. None of these services ever execs a shell. Services get allotted a reasonable fraction of total CPU, memory, and disk resources. While similar to a jail, this lockbox also includes resource allocation and limits on network activities. To make it easy to jump through configuration hoops, services come with configuration templates so that allocating resources is dead simple.

The fantasy OS would have to be tiny, so that it can be verified. We have seen more than enough kernel exploits in the last several years to convince people (at least me) that small is beautiful.

Frankly, the fantasy OS would permit us to get a lot more work done, as we would spend a lot less time dealing with broken systems.

I know I have left out a lot of necessary features, such as secure authentication that supports login and other services. You can read other people's ideas about single sign-on (Scher's article) and managing identification (Lear's article) in this issue. And just to balance things out, you can read about reverse engineering of code (Wysopal), slicker Windows rootkits (Butler and Sparks), defending against buffer overflows (Alexander), port knocking that unlocks SSH (Rash), and improvements to honeynets (Forte et al.). Jennifer Granick provides words of advice about the legality (or, rather, the lack thereof) of spyware. And Goel and Bush consider biological models for computer immune systems, because security will never be perfect.

Even in an imaginary world.