RIK FARROW

musings

rik@usenix.org

AS TECHNOLOGISTS, WE ARE STUCK in a terrible double bind. We need to build upon the successes of the past—those who ignore history surely suffer from their ignorance. But there comes a time when past paradigms must be supplanted by new ideas, or we stagnate. This conundrum gets magnified by the effect of experience; that is, the more expert we become at a particular technology, the more value it has for us. To abandon what we know well, even if it no longer serves us, is like killing the goose that lays the golden eggs.

Strange thoughts indeed, but if you have read my columns before, I doubt that you are at all surprised. And it was new ideas, not retreads, that brought me to this point. I was fortunate enough to attend the WORLDS, OSDI, and HotDep workshops and symposium in Seattle (November 2006), where I got flooded with new ideas. Just as Washington State was being deluged with record rainfall, I felt like a privileged student at a fast-paced series of advanced seminars.

I discovered that OSDI and its ACM-sponsored companion, SOSP, are considered the most prestigious of operating systems conferences. One young attendee told me that getting a paper accepted at either conference could assure your future. I asked more seasoned veterans what they thought of this notion, and I got thoughtful responses that mostly agreed with this. I certainly came away impressed and uplifted (who cared if it was pouring rain outside?).

I would describe the view from ten thousand feet of OSDI '06 as papers presented on file system enhancements, performance improvements as well as measurement techniques, methods for improving reliability of operating systems, and security. You can read the summaries to get the complete picture, along with the papers themselves online. I want to constrain myself to those papers that most excited, entertained, or disturbed me.

Code Defense

Feng Zhou described SafeDrive as a method for guarding against failed device drivers through language extensions. By adding annotations to device drivers, processing and compiling those drivers, then using them with a modified Linux kernel, crashes in those drivers will be avoided, even safe-

ly recovered from. This is a pretty amazing idea, different from previous papers that included microkernel designs, separate hardware protection domains, and other techniques for software fault isolation (including SFI and XFI). I found myself wondering how good this solution was, as it still relied on the programmer doing the right thing (annotating the code properly in all the required places), with the programmer being the weak link already.

Still, SafeDrive does come closer to the solution of a real problem. Device drivers are notoriously difficult to write and debug. And the word in security circles is that wireless device drivers are looking like tempting targets, as a successful attack here bypasses all current defenses—except, of course, for solutions such as SafeDrive or XFI.

Úlfar Erlingsson of Microsoft Research presented the paper on XFI, a system that creates safe extensions by binary rewriting of Windows x86 Portable Executables. The inline guards provide runtime checks before calling other functions or making computed jumps, thus guarding against executing code at unexpected locations because of bugs or attacks. Úlfar's chosen example was one of my very favorites, a bug in a JPEG decoder that allows code of an attacker's choice to be executed. In a live demo, the unsafe version crashed the browser, while the XFI-protected version of the library returned, preventing a browser crash (and a potentially exploited system).

Charlie Reis presented BrowserShield, a different approach to protecting Internet Explorer. In research sponsored by Microsoft, Charlie explained that many browser vulnerabilities could be defended against by rewriting scripts before they could be interpreted by a browser. In the sample implementation, scripts would be partially rewritten using the Microsoft firewall (ISA) as a proxy, along with a bit of JavaScript running within the browser itself. When used in conjunction with patches and anti-virus software, this approach prevented successful attacks related to 19 critical vulnerabilities found during 2005 in IE.

I want you to consider the implications of these fine research papers. We can't write secure or even defect-free software. Having accepted this fact, our finest scientists are designing a patchwork of systems that may make it possible to run buggy and dangerous systems as safely as possible. I know personally how difficult it is to write software , and I decided early on not to expose myself to the embarrassments suffered, because people continue to find security holes in software that is open source, over 20 years old, and written by programmers who are much better at it than I ever was. There is a temptation to blame programming languages, but we have yet to develop a safe programming language. After all, that is just another programming project (or meta-programming project), with all the complexities that that brings. Perhaps there is yet another way . . .

Charlie Reis also presented a WiP about building a browser where each site gets encapsulated by running within its own process. The Konqueror browser actually facilitates this work, according to Charlie. Divide-and-conquer has been a strategy that has worked well for some secure servers, such as Postfix and DJBDNS, and I personally feel that isolation of code and the use of least privilege is critical in any future solution. In the three papers just described, the approaches are to notice that software has been either exploited or simply run amuck, or to filter out attacks after they are known but not yet patched (BrowserShield).

At HotOS in 2005, I learned of an operating system called Asbestos. In Asbestos, all data gets tainted with labels as it flows through the system, and tainted data cannot escape via the network once it is mixed with data that has a different taint. During OSDI '06 Nickolai Zeldovich described HiStar, the successor to Asbestos. Like Asbestos, information flows get

tainted. But HiStar goes beyond Asbestos in that everything gets labeled with categories, and these categories control how information can flow through the system. Nickolai used the example of running ClamAV, an open-source anti-virus system that must have read permission on all of an owner's files. HiStar can safely read all files because it prevents ClamAV from leaking any information read in any file.

HiStar approaches, and may have reached, what I'd like to see in a new, secure operating system. In HiStar, there is no root, nor is there a complex policy definition (as in SELinux); it is a system designed from the ground up to provide robust isolation. Combined with programming techniques, such as having a browser thread for each site visited, HiStar just might be the OS I have been dreaming of. It will take time to tell, plus additional time for me and others to understand this completely different OS with a Linux API.

Reduction and Configuration Management

Although log compression and configuration management might not seem related, there was an amazing paper by Chad Verbowski (also of Microsoft Research) and others that does unite these two disparate topics. Flight Data Recorder (FDR) (say, haven't I heard of another similarly named software project?) has the goal of capturing configuration and file changes from Microsoft systems and will be shipped with Windows Vista. Using a time window of only 6 ms, FDR captures all changes to system configuration–related registry entries and files, saves the log locally, then cleverly compresses it, without losing any interesting data, before uploading the compressed logs to a server. The goal was to capture data from thousands of servers while using less than 1% of network bandwidth, with a less than 20 MB/day logfile per system that can be analyzed in 3 seconds.

Sounds unbelievable, but FDR manages to compress each event into an average of 0.7 of a byte. The motivation for this clever work was the discovery that 33% of system outages were related to configuration changes, so tracking those changes was key to system reliability.

Speaking of system reliability, the final talk had to do with an interesting sensor network, one you might have heard about if you read science news. Geoff Werner-Allen explained some of the problems he and his co-authors had in monitoring Reventador, an active volcano located in an Ecudorian jungle. The current monitoring scheme relies on a barely luggable device powered by multiple car batteries. The team built small sensors powered by flashlight batteries, complete with a small seismometer and a microphone that captures subsonics typical of the rumblings of Reventador. The sensor communicated via a mesh to a single wireless uplink and then to a base station located in a hotel several kilometers away.

The sensors worked well. But in the hotel, the electric generators only ran about three hours a day, not enough to charge the batteries on the laptop used as the base station. Even with the occasional loss of the base station, the researchers were able to collect useful data. Geoff explained that time synchronization of the sensors had worked great in the lab but not so well in the field, but they were able to correct for time differences using some clever analysis.

If you think this is a cool project, well, so do I. There were some real downsides to the onsite research, though. One of the sensors shut down unexpectedly, and the cause turned out to be a chunk of rock that had smashed its antenna. Keep in mind that these researchers had to hike out and place, and later recover, the sensors on a very active volcano. Another issue had

to do with insurgent groups that would block the only road leading back to "civilization." When this occurred, the supply truck would be blocked as well, and there would be no beer. Ah, the pains of doing field research.

After this talk, attendees for the most part stayed in the conference room. There were clusters of people gathered around the final three speakers. After fifteen minutes, USENIX staff had to urge people to move elsewhere so that the room could be reconfigured for HotDep '06.

The Lineup

We begin this issue with an article by Simson Garfinkel. Simson has been studying recycled hard drives and needed a way to store massive amounts of data and then analyze that data. While searching for the perfect hardware solution, he decided to try Amazon's Simple Storage Service (S3) and Elastic Computing Cloud (EC2). His article reports his experiences using these systems, their security, availability, and suitability for various uses. I really liked learning about EC2 and S3, as these services are a real hint of the distributed services we will see much more of in the near future. Simson also compares the Amazon services to other grid computing services. Like Simson, you might find that these services provide an alternative to buying and building your own grid computing cluster for a research project, but they are not quite ready for commercial use yet.

In the next article, Jorrit Herder and other project members provide an update to MINIX 3 that focuses on failure resilience. Herder writes a perfect companion article to the OSDI summaries, in that this paper looks at some of the same issues, such as device driver reliability, using a microkernel designed to run everything except a few core services in isolated tasks in userspace. I asked Jorrit why their work hadn't appeared at OSDI, and he said that this particular research wasn't far enough along at the OSDI paper submissions deadline.

Steven Hand and members of Xensource explain the issues involved in virtualization. I had become curious about what Intel and AMD were doing in the newer CPUs to provide hardware support for virtualization. I knew that a VM ideally sees an environment that appears exactly the same as a native, bare-metal environment, but at the same time the VM monitor must capture all direct accesses to the underlying hardware. There is also the messiness that occurs when an OS within a VM believes it controls the page maps, but in reality the VM page maps are just another level of abstraction on top of the monitor's page maps. Also, the Intel IA-32 architecture was not designed with VMs in mind, so there are instructions that behave differently when executed outside of ring 0 (in nonprivileged mode), causing more problems for designers of VMs. I hope this article will instruct you in what hardware manufacturers have done to help support the growing use and improve the performance of VMs.

In the Sysadmin section, Mark Burgess completes his cycle of articles about configuration management. Mark continues his exploration of how configuration management should be done by moving on from the representation of configuration information to talking about style of management. Should there be centralized, authoritative control, or something much more adaptive modeled on the economics of trade? As always, Mark provides a deeply thoughtful and very well written article.

Next, Leigh Griffin and John Ronan have written a guide to getting Xen servers up and running. Hewing to the operating system theme, which veered off into the world of VMs without much assistance, these two men

decided that the world needed an easy-to-follow beginners' guide, and they set about writing it, then sharing it with *;login:* readers.

Robert Marmorstein and Phil Kearns have written about the tool they have been building that analyzes firewall policies, based on Linux netfilter. They start by building a series of tests to determine whether a firewall configuration actually worked as expected. From this experience they discovered that it made more sense to analyze firewall rulesets and then convert these rules into classes of systems that have similar policies. This technique can reveal unpleasant surprises in your firewall configurations, but ones that you will want to discover yourself, instead of having someone else do it for you.

David Blank-Edelman plays along with the operating system theme by explaining fork and how to work with multiple-thread Perl scripts. As David explains, this is a much deeper topic than can be handled in a column, but with his usual aplomb he provides us with working examples and pointers to cool modules that make using multiple threads a bit easier.

Robert Haskins decided to take a look at the various projects and products that support DHCP. As always, Robert writes from the viewpoint of an ISP, giving us a different perspective on a service that we generally configure and forget (until there is a change or a problem). Heison Chak joins in the VM subtheme, discussing how he runs different versions of Asterix within Xen virtual machines, and Robert Ferrell entertains us with his own views on operating systems.

In the Book Reviews, Elizabeth Zwicky, our official book reviewer, leads off with a long look at *Mastering Regular Expressions*. As Elizabeth writes, this is a topic that we all should learn more about, and she tells you just why this is important and what this book can do for you. Elizabeth next tackles a couple of management-level books, then has strong words about the final book on her list this issue. Next, Paul Armstrong discusses a good book he has read about IPv6. Sam Stover provides us with another in-depth look at a security book, and I follow on with two reviews of my own, including the newest in the Sysadmin Handbook series.

In Standards, some members of the C standard committee invite you to comment on changes related to support for threads. Finally, we have summaries for WORLDS, OSDI, and HotDep. We also received summaries of the Grace Hopper conference and workshop focusing on women in computing, and we end on that needed note.

Stuckness

I began my column by alluding to the tendency to stick with what is well known. Voyaging out beyond the frontier is risky, upsetting, and disturbing, because it suggests that perhaps what we have spent years learning is not the best solution. Pioneers have always had it rough. When we consider that Galileo was ordered to stand trial for heresy for his book suggesting that the earth revolves around the sun, our own trials pale. We do not face a literal burning at the stake for suggesting new ideas. We might be roasted for going up against the status quo, but that is only ego bruising, and at worst harmful to one's career.

I don't want to suggest (quoting Firesign Theater) that "Everything you know is wrong." Hardly. I do want to encourage you to keep on the lookout for new ideas, software, operating systems, and techniques that might very well solve problems in security, system administration, programming, and configuration management that so plague us today.