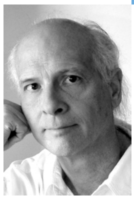
RIK FARROW

musings



rik@usenix.org

IT'S A BEAUTIFUL DAY OUTSIDE, BUT the nights are tending toward cold. And by the time you read this column, fall will be changing into real winter. I love this time of year, when overheated summer days have passed and milder weather rules. The change into really cooler weather has always felt stimulating to me.

These changes got me thinking about the changes we've seen in the computer industry. Some obvious ones have to do with the price versus performance of desktop computers. Software has changed too, as desktop CPUs became fast enough to include in supercomputers. Even graphics processors are evolving into array processors useful for calculations having nothing to do with games. But even as there are changes, there are also those things that seem almost changeless.

Peek from the Past

I started reading an article referenced by Jason Dusek (October 2008 *;login:*) by Alan Kay. Kay is one of the creators of SmallTalk, but what really struck me was how he was thinking about computers in 1967 [1]. Kaye envisioned under-five-pound, wirelessly connected tablet computers used specifically for helping children learn. He mused about this in the day when graphical displays were rare, and a computer that could run one was the size of a desk with a mainframe for a backend. Today, SmallTalk runs on the XO as Squeak [2], and it grew directly out of Kay's ideas surrounding the DynaBook, that tablet computer he envisioned in the 1960s.

Kay's vision included both software and the hardware that would support it. And it is really great to learn that this distant vision has actually become reality in some form. But I would like to see more.

If you read Kay's article, you might be struck by how much of the computer architectures of the 1960s remains familiar. Kay worked his way through college as a programmer for the Air Force, using Burroughs mainframes. His second system, a B5000, had segmented storage, high-level language compilers, byte-code execution, automated mechanisms for subroutine calling and multiprocess switching, pure code for sharing, and memory protection mechanisms. This was in 1961 and should all sound familiar except for the name of the mainframe company.

As I've pointed out before, our computers, even if we can carry them within our cell phones, still share a strong architectural heritage with these 1960s mainframes.

Changing tack a bit, I continue to be influenced by a position paper and talk given by Timothy Roscoe during HotOS XI [3]. During this talk, Mothy suggested that the move into virtual machines is an incremental one, and not a particularly good one for OS research. Although his focus was research, I found a lot of what he had to say mind-opening.

Consider Microsoft's continuing supremacy as a desktop operating system. Microsoft has two strengths: a wealth of popular applications and a huge stable of device drivers. Roscoe suggested that neither is really tied to a Microsoft OS, as both device drivers and applications can already be run within VMs. Now, I am guessing you are thinking that you have to run Windows within the VM to do that, but that's almost not true. You can run Windows device drivers on top of other interfaces, and run some applications within WINE.

Roscoe said that what was important in each case is that, whether you want to run a Windows device driver or a Windows application, what you need to do this is a software stack that supports it—not all of Windows, just the portions required to run the software. Consider Device Domains within VMs, where the purpose of the domain is to support a particular device and make it available to applications running within other VMs. Today, that requires an entire OS, but should it?

Today we need to run Windows to gain access to particular devices or applications. But I can imagine a day when this will not be so—and, apparently, so can Microsoft.

Clouds

Another change appears in cloud computing and SAAS. Although cloud computing is not particularly successful so far, Steve Ballmer pre-announced Windows Cloud, a development environment for cloud computing based on Windows. You should know a lot more about this than I do, as all I have are hints coming from Ballmer. But he says it means moving .NET into the browser, as Microsoft has already done with Silverlight.

Silverlight has not been catching on like gangbusters, and it is hard to imagine people stepping back into any environment where they are controlled by one vendor. And this includes Google as well.

This brings up another change. Ten years ago, it was hard to imagine storing all of your data in the cloud—or, rather, a particular vendor's cloud. Yet today, many people choose to use Google, Apple, or Microsoft to hold not just their email, but also backup data. The privacy issues alone are chilling, but so is a change where people give up control of their data and rely on the ability of others to keep it safe and secure. When you consider that these offers actually promise little privacy and often no actual guarantee that the data will remain available, this is a scary change indeed.

Google has released Chrome for Windows. While speaking at Google two years ago on security, I was asked by an employee what my talk on OS and browser security had to do with Google. I replied that Google's entire business model relies on people being able to use browsers securely, and if people give up Web browsers as hopelessly insecure, then Google's business model will die. In other words, Google really must care about browser security. And Google does care.

Not that Chrome has been that exciting so far [4], with some killer bugs in the early versions. I did study the Chrome sandbox [5] some. I thought it was nothing but a system and library call wrapper, but it turns out that Chrome relies on a Windows XP and Vista security feature called integrity levels. You split an application into parts, with the ones running software that process untrusted input (anything you get from the Internet) with reduced privileges that cannot make most system calls. If some JavaScript code wants to write to a file, it must communicate with its parent application, one that runs outside of the sandbox. This strongly reminded me of the OP Browser [6], written by Sam King and his students.

Chrome, as well as the OP browser, still has a problem present in other browsers, and that is plug-ins. While attending the USENIX Security Symposium this summer, I learned, to my dismay, that plug-ins run outside the security models used to isolate one Web site from another and the rest of your system [5]. Simply stated, when you run Flash or PDF or media players, they run as ordinary user applications, not in any sandbox. They can do whatever you can do, and, get this, most include JavaScript interpreters. So just running a Flash movie can result in your system being owned.

The Dark Economy

I am writing during what I hope are the darkest days of the financial collapse. But the dark economy represents another change, this time in computer security. You might have wondered why we rarely experience worms anymore. You might even have thought that security has improved to the point where these worms can no longer rampage across the Internet, taking down millions of systems in minutes.

It is not security that has done away with worms, but motivation. Go back seven years, when people released worms to prove a point or impress their hacker friends. Today, the same type of exploits that worms relied upon are hot items on the dark market. Using an unknown exploit in a flashy worm would be a waste of a resource that might be worth tens or even over a hundred thousand euros.

So we don't see worms anymore. But we still see lots of viruses. During NSDI '08, I sat next to a gentleman who wants to remain anonymous. He told me he collects spam for research, lots of spam. As part of this effort, he analyzes email viruses. He told me that hundreds of new viruses, mostly modifications in packing and encryption, get released every day and that no anti-virus software does better than 80% at detecting these. Most anti-virus software will release protection within a week, but by then new versions are being used. Niels Provos said, during both of his two talks during that USENIX Security Symposium, that only heuristic-based AV has any chance of reaching even that 80% detection rate.

Web browsers have been the main way of exploiting desktop systems for many years now, so some things haven't changed.

The Lineup

Our Security issue begins with an article about a very cool bit of research. Sam Small, Joshua Mason, Ryan MacArthur, and Fabian Monrose provide more details about their research into writing a honeypot that uses natural-language learning techniques to represent hundreds of Web applications simultaneously. Their article explains what they did and why they did it, and it reports the astonishing number of attacks against their simulated server in a short time.

Tim Yardley has written an article about SCADA. Yardley researches SCADA security issues, so I asked him if he could go beyond the headlines and tell us more about real problems with SCADA. Yardley does this well, and although he ends on an upbeat note, I find myself not feeling comforted by the progress so far.

Zhang, Porras, and Ullrich reprise their award-winning paper from Security '08 by explaining their new blacklisting technology in more detail. Blacklists have been around for many years, but Zhang reports on a new type of blacklist that is proving especially efficient at blocking attacks.

Berg, Teran, and Stover share their experiences with another honeypot, Argos. Argos runs inside an emulator, QEMU, that allows it to detect new Windows exploits by tainting registers and noticing whether tainted registers result in changes in execution flow. When a possible exploit is detected, Argos halts the program and saves a lot of state, making it possible to understand what happened, as well as to recover the bytecode that formed the heart of the exploit.

Calvin Ardi and Daniel Chen, two UC Berkeley undergraduates, share their analysis of AirBear, the UC Berkeley wireless infrastructure. The two students discuss real and potential vulnerabilities along with possible solutions.

David Blank-Edelman sticks to our security theme, writing about tools for analyzing the quality of passwords using Perl. Dave Josephsen sticks to his own theme, extending his October 2008 column by updating his example code to run on the new version of Nagios. Dave makes a strong case for writing and using your own event brokers, and his demonstrations make it look possible.

Robert Ferrell examines the implications of the software patch cycle. Writing perfect software remains an impossibility, but Robert reminds us that we willingly accept quality levels that we wouldn't dream of accepting in other products.

In Book Reviews, Elizabeth Zwicky starts with a book on making slide presentations, then digs into another book (which does turn out to be different) on security visualization. She concludes with a review of the new book by Whitfield Diffie and Susan Landau on privacy. Sam Stover treats us to *Hacking Exposed, Linux*, which turns out to be less a book about hacking then it is about the Open Source Security Testing Methodology. In some ways this book, Sam tells us, teaches us "more about 'hacking' than other books." Brandon Ching reviews *Rails for PHP Developers*: not only does he like the book, he might even start to like Ruby.

I finish out the year by reviewing a noncomputer book. Neal Stephenson released another thoughtfully written book, and one I feel fits a geek culture like a pair of gloves. On the one hand, Stephenson exposes us to the philosophy of science and the tension between the everyday world and that of the researcher. On the other hand, he takes us on a sometimes thrilling ride through a richly imagined other world that seems very much like our own in many ways. In the end, the reason for the parallels between worlds does become apparent, even as the polycosm collapses into one reality.

We are graced with many summaries this issue. We start off with the '08 Security Symposium and continue with summaries of many of the workshops that occurred around the symposium: WOOT, USENIX/Accurate Electronic Voting Technology, HotSec, and Metricon.

We live in a changing time. Always. Nothing stays the same except our own hidebound traditions, and even those do change. Staying the same benefits the established hierarchies, whereas change benefits the outsiders to the system. Sometimes change brings unpleasant upheavals as well. In the end, our culture is the result of many revolutionary changes, and I don't expect that to end.

REFERENCES

[1] Alan C. Kay, "The Early History of Smalltalk": http://gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html.

[2] SmallTalk for the XO: http://wiki.laptop.org/go/Smalltalk.

[3] Timothy Roscoe, Kevin Elphinstone, and Gernot Heiser, "Hype and Virtue," HotOS XI: http://www.usenix.org/events/hotos07/tech/full_papers/roscoe/roscoe_html/.

[4] Google Chrome releases (bug fixes): http://googlechromereleases.blogspot.com/2008/09/beta-release-0214929.html.

[5] Adam Barth, Collin Jackson, and John C. Mitchell, "Securing Frame Communication in Browsers": http://www.usenix.org/events/sec08/tech/barth.html.

[6] OP browser: Chris Grier, Shuo Tang, and Samuel T. King, "Building a More Secure Web Browser," *;login:* (August 2008), pp. 14–21.