EDITORIAL

Musings

RIK FARROW

Rik is the editor of *;login:*.
rik@usenix.org

I have often mused about how the architecture of the CPUs we use influences the way our operating systems and applications are designed. A book I reviewed in this issue on the history of computing managed to cement those ideas in my head. Basically, we've been reprising time-sharing systems since the mid-60s, whereas most systems serve very different purposes today.

Along the way, I also encountered a couple of interesting data points, via pointers from friends who have been influencing me. One was Halvar Flake's CYCON 2018 talk [1], and another was about a new OS project at Google. The opinion piece by Jan Mühlberg and Jo Van Bulck that appears in this issue influenced me as well, as it describes a CPU feature that, among other things, could block the use of gadgets in return-oriented programming (ROP).

Flake explained that much of our current problems with security have to do with cheap complexity. Even though a device, like a microwave oven or intravenous drip-rate controller, only requires a PIC (Programmable Interrupt Controller), it may instead have a full-blown CPU running Windows or Linux inside it. CPUs are, by design, flexible enough to model any set of states, making them much more complex than what is needed inside a fairly simple device. Designers instead choose to use a full-blown CPU, usually with an OS not designed to be embedded, to model the states required. Vendors do this because many more people understand Windows or Linux programming than know how to program a PIC.

This isn't just a problem for ovens or routers. Let's not even discuss home routers, although the arguments for using a real OS are at least stronger in the case of routers. Dan Farmer published research, funded by DARPA, in 2013 about IPMI and BMC [2], the controllers found on most server-class motherboards. These controllers provide an over-the-network method of managing servers—e.g., rebooting them or installing updates. But the controllers are full-blown Linux systems, burned into ROM, using very old versions of Linux and exploitable software. The controllers can read or write any system memory, as well as use either a dedicated network interface or any network interface on the server, making them the obvious point for an undetectable attack using an embedded system that does no logging and cannot be patched. Ouch.

One of Flake's concluding slides had this bullet point, one I particularly liked:

- CPU-architecture and programming models are in flux for the first time since the 1980s.

I'd argue that the date is wrong, as we didn't get heavily into threaded programming until the noughts; other than that, the CPU architecture has remained very similar in rough outline to late '60s mainframes. But ignore the date and ponder Flake's implied suggestion: now is a good time for some serious changes in architecture and programming models.

Another data point is currently much more obscure. Google has a project, an OS named Fuchsia powered by the Zircon microkernel [3], based on another Google project, a microkernel named LK. Both appear to be focused for use in IoT and embedded systems. But Zircon has been designed to work on modern devices with more powerful CPUs and lots more memory than LK [4].

Zircon has a small kernel that manages processor time, memory, I/O, interrupts, and waiting/signaling. Everything else gets done in userspace, as processes. The enabling technologies that make this work well are IOMMUs and ARM SMMUs. Both the IOMMU and SMMU were designed to support virtual machines, allowing a VM to have access to, for example, a network interface queue. But these subsystems also mean that userspace programs can gain access to device memory and be able to copy data between the devices and other memory, something that has been a barrier to running system services in other microkernels.

While the Fuchsia project appears targeted at embedded systems, likely including support for Android where message passing is already used in the API, having a very small kernel reduces the immense attack space provided by modern operating systems. I've skimmed the source code for Zircon enough to see that it is a message passing system that does so by passing ownership of memory between processes and has support for both IOMMUs and SMMUs. Tinkering with the design of CPU paging systems, so that context switches don't require flushing page caches, would make this an even faster system.

I believe that Fuchsia is still in such an early phase that not a lot can be said about it, but I'm certainly excited by the concept. There are other microkernels in very wide use, such as seL4 [5], used on the radio side of hundreds of millions of cell phones. But with the potential to support Android, I think that Zircon may turn out to be something much more visible, and make devices much more secure than the usual OS used in devices like tablets and smartphones.

The Lineup

Jan Tobias Mühlberg and Jo Van Bulck sent me an opinion piece about the trouble with closed and complex CPUs. They have been working on hardware that will have an open design facilitating public verification as well as security features that will cut-off many exploit techniques. Bulck also had a paper about extracting keys from Intel SGX at ScAINet '18, part of the fallout from the exploits known as Meltdown.

While there was lots of interesting research at Annual Tech last summer, I asked two groups to write about their research since I thought both projects might have some interesting future impact, and both groups published their code.

Hu et al. write about their extension to ext4 that adds transactions, TxFS. By building upon journaling, a feature of other file systems types as well as ext4, they have added the ability to start, commit, or cancel transactions with the addition of kernel code (that is published) and a handful of function calls. I think that TxFS stands for Transaction File System, but might also be Texas File System, as the authors are at UT Austin.

Oakes et al. write about a lightweight container for use with Lambdas. AWS introduced Lambdas for serverless computing, but the problem with using these is the startup cost for loading a container complete with the necessary libraries for the servlet code. SOCK builds on previous work [6] and provides a much lighter-weight container than Docker, for example, and this article explains how and why that is done.

You can expect more articles about security in the Winter issue, as the security papers deadline came too late for me to ask authors to write for this issue. But we do have the sixth BeyondCorp article. Google's BeyondCorp focuses on securing the clients that access resources within Google, and this article reveals more about how the BC team has done this for their fleet of clients. While not everyone can expect to be able to do what Google has done, there are many useful pointers in the work they have made public in this article and the ones that have come before. For example, BC can whitelist software, something that anyone can do using policy in Windows or Macs or with a commercial product like Carbon Black (Bit9).

John Kristoff has written about his own project, a sensor net. Kristoff explains how he has set up instances that listen for probes and exploits on a handful of services, provides data via his website on the attacks he sees, and describes how you can set up your own sensor net. I found John's approach practical and interesting, and the cost is reasonable enough to be supported by small grants.

Aleatha Parker-Wood has written an excellent summary of the first Security and AI workshop, ScAINet. Applying AI techniques to security data, such as logs, is a growing area but one that is also fraught with issues that can make AI fail. The workshop examines both the benefits and the issues with using machine learning (ML).

I asked Rick Boone, an SRE at Uber, about the talk he gave during SREcon18 Americas. Boone explains how Uber does capacity prediction instead of capacity planning, using ML techniques that I recognized after my foray into ML in the Summer 2018 issue.

We have a new Python columnist. Peter Norton, co-author of several books and current SRE, takes us through his issues with how poorly Python packages that rely on shared objects work and how he'd like them to work. Norton crafts a new method for using packages that allows loading of shared objects without leaving a mess of files to clean up, relying on a relatively new Linux system call, `memfd_create()`.

David Blank-Edelman demonstrates GraphQL, an API query tool created by Facebook. GraphQL has nothing to do with graph databases, the topic of his Summer 2018 column, but instead provides an interface that is a step deeper than REST, and can return more results than REST with a single query.

Chris "Mac" McEniry demonstrates using GoLang with LDAP. Mac points out that while there are commandline tools for LDAP, having a GoLang app allows encapsulation of site-specific information.

Dave Josephsen waxes enthusiastic about data lakes. Data lakes imply large amounts of unstructured data that you don't want to spend money adding indices to, but do want to be able to query. Dave explains how this can be done using tools like Apache Parquet.

Dan Geer examines the numbers found in Mary Meeker's (of Kleiner Perkins) "Internet Trends 2018" presentation. Dan drills down and exposes the portions of the slides he found particularly interesting as representative of the types of data useful for security metrics, as well as pointing out the use of AI in content platforms.

Robert Ferrell explains that AI and bots are already in control of our online lives, so we might as well get used to it.

We have three book reviews: Mark Lamourine covers the fifth edition of the Nemeth classic and a book with proof that Agile techniques work, while I review a wonderful illustrated book covering the history of computing.

Changing CPU architecture is very hard, as companies have spent many billions of dollars tweaking their designs to produce the best performance. Flake points out that the same vendors are quite willing to trade off reliability for performance, as seen in Meltdown, an intersection between a trusted subsystem and branch prediction. We do have problems with security, ones that need to be dealt with, not only with changes to software toolchains but also to the underlying hardware. Let's hope that this is a direction that may prove fruitful soon, even if it's unlikely to prevent attacks on our critical infrastructure in the near term [7].

References

[1] T. Dullien, aka Halvar Flake, "Security, Moore's Law, and the Anomaly of Cheap Complexity," CYCON 2018: https://goo.gl/3HzQ1y.

[2] D. Farmer, "IPMI: Freight Train to Hell": http://fish2.com/ipmi/itrain.pdf.

[3] Zircon: https://fuchsia.googlesource.com/zircon/.

[4] S. De Simone, "An Early Look at Zircon, Google Fuchsia New Microkernel," *InfoQ*, April 15, 2018: https://www.infoq.com/news/2018/04/google-fuchsia-zircon-early-look; M. Bergan and M. Gurman, "Project 'Fuchsia': Google Is Quietly Working on a Successor to Android," *Bloomberg,* July 19, 2018: https://www.bloomberg.com/news/articles/2018-07-19/google-team-is-said-to-plot-android-successor-draw-skepticism.

[5] The seL4 Microkernel: https://sel4.systems/.

[6] S. Hendrickson, S. Sturdevant, E. Oakes, T. Harter, V. Venkataramani, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Serverless Computation with OpenLambda," *;login:,* vol. 41, no. 4 (Winter 2016): https://www.usenix.org/publications/login/winter2016/hendrickson.

[7] D. Sanger, "Pentagon Puts Cyberwarriers on the Offensive, Increasing the Risk of Conflict," *New York Times,* June 17, 2018: https://www.nytimes.com/2018/06/17/us/politics/cyber-command-trump.html.

ENIGMA®

A USENIX CONFERENCE

SECURITY AND PRIVACY IDEAS THAT MATTER

Enigma centers on a single track of engaging talks covering a wide range of topics in security and privacy. Our goal is to clearly explain emerging threats and defenses in the growing intersection of society and technology, and to foster an intelligent and informed conversation within the community and the world. We view diversity as a key enabler for this goal and actively work to ensure that the Enigma community encourages and welcomes participation from all employment sectors, racial and ethnic backgrounds, nationalities, and genders.

Enigma is committed to fostering an open, collaborative, and respectful environment. Enigma and USENIX are also dedicated to open science and open conversations, and all talk media is available to the public after the conference.

PROGRAM CO-CHAIRS

Ben Adida
Clever

Franziska Roesner,
University of Washington

The full program and registration will be available in November.

enigma.usenix.org

JAN 28–30, 2019

BURLINGAME, CA, USA

usenix® THE ADVANCED COMPUTING SYSTEMS ASSOCIATION