EDITORIAL

Musings

RIK FARROW

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

I read Brian Kernighan's latest book recently, and many things in it struck chords with me. While the book was mostly about UNIX history, it was what Brian wrote about the influence that UNIX had on the development of computers, programming, and even printing that grabbed my attention.

For example, Brian pointed out that computers had been highly customizable machines that generally ran programs that were terribly inflexible. If you wrote files to disk using one program, you could only use that program, or a closely related one, to manipulate those files. UNIX, by comparison, uses byte streams for files, ones that can be opened by any program, even ones that can't do anything sensible with the bytes, but that flexibility is enormous. Think of the old version of `spell`; that was a pipeline that converted a text file to a list of words, sorted those words, ran `uniq` on them, and then compared the results to a dictionary. None of those tools is unique to a spell-checking program. Today, `spell` is a binary, not a shell script, but you can find a version of the original script in Brian's book.

Computer Architecture

Another idea that caught my attention appears in the book at the bottom of page 128: UNIX and C had a large impact on computing hardware in the 1980s and 1990s. Most successful instruction set architectures were well matched to C and UNIX.

I certainly never really thought about that back when I was working with UNIX workstations in the late 1980s. I know I worked on at least a dozen different workstations in that period, mostly various RISC architectures as that was the hotness of the day.

A key feature of all of those instruction set architectures (ISAs) was that instead of being word oriented, all were byte oriented. That may seem too simple, but consider the types that popular programming languages use: very few are tied to a memory-length word. Some types, like float and double, are closely related to actual hardware in the CPU, but things like arrays, strings, different flavors of integers, structures, are all byte, or multiples of bytes, oriented.

I am not certain that UNIX is responsible for this, but UNIX certainly was a huge influence. I was talking about this with Jon Callas, who worked for DEC in the '90s, and he pointed out that DEC's Alpha CPU worked equally well running Ulrix/64, DEC's VMS, and Windows NT (Windows these days). None of these operating systems and their underlying languages were word oriented, although I do wonder about VMS, which was still written in assembler in the 1990s.

CISC vs. RISC

Today, most servers run variants of Intel's ISA, while the world of the small is mainly RISC. That Intel is byte oriented is no mystery: the Intel 8080 CPU had eight-bit registers and a 16-bit address space. Most registers were paired, so could appear as 16 bits wide, but the only register capable of integer arithmetic was the `A` register, and that was eight bits in width.

In the '80s, we thought that RISC was the way of the future, as RISC allowed CPU designs to be simpler. What happened instead was that Intel kept making up for the weakness of CISC through hardware tricks, including converting their CISC ISA into an internal RISC-like ISA. These tricks require more transistors and more energy, meaning there is still a chance that ARM64 may become more popular in server farms and clouds—but I am not betting on it.

The Lineup

I started my search for authors by looking at the accepted papers at SOSP '19 and found two I particularly liked. Neither won best paper awards, but the author of the first article, Abutalib Aghayev, told me that his paper had been downloaded four times as often as the one that did win the best paper award. I think that's because his topic is more pragmatic.

I also liked the paper that this article is based on because it relates well to my Winter 2019 column about file systems. Aghayev and his colleagues at CMU and Red Hat created BlueStore for Ceph. Ceph is a distributed file system and had been relying on existing file systems for block storage. Aghayev et al. wrote BlueStore to work in raw partitions in just two years, while greatly improving performance and adding features unavailable when Ceph nodes ran over file systems like Btrfs and xfs.

The next article is based on a paper by Anish Athalye and his colleagues at MIT that uses a clever design to solve a security problem that had proved intractable. Hardware wallets, such as used to store and transact Bitcoin, have proven to be vulnerable to attacks, and Athalye fixed this by using two small processors and *reset-based switching*, so that multiple programs can be run on a hardware wallet but be unable to interfere or attack other programs and their data.

Next up we have two articles about AI/ML. Jessica Cussins Newman and Rajvardhan Oak write about ethical considerations for companies and researchers working with AI. The authors present a balanced and thoughtful look at the impacts AI will have on politics, justice, and human rights. L Jean Camp introduced me to the authors, and I am happy to extend our series about ethics with this article.

Nisha Talagala and Joel Young, the co-chairs of OpML '20, tell us about what they learned from the first OpML conference and explain what they expect will come out of the second conference in May 2020. The authors point out that ML differs from earlier computing paradigms, echoing Newman and Oak when it comes to ethical considerations, but also that AI/ML is different operationally.

Switching to SRE/Sysadmin, Luis Mineiro explains how Zalando, Europe's largest online fashion platform, has learned to deal with paging. In an age of distributed systems, when SLIs (service level indictors) show something has gone wrong, you only want to page the people responsible for the sub-system causing the slowdown or outage. And this can be trickier than it might seem.

Todd Palino explains a system for organizing work called "Getting Things Done." GTD is based upon a book, but Palino shares his own experience as well as tools that can be used to support the process. Just about everyone can benefit from learning about and, better, using GTD.

Jaime Woo and Emil Stolarsky examine how to choose the best SLIs. They use the analogy of a famous Florida theme park to explain what works best as indicators of customer satisfaction as opposed to choosing less potent indications of success.

I interviewed Mary Ann Horton. I met Mary Ann while at USENIX ATC '19 in Renton, Washington. She was there for the 50th anniversary of UNIX, celebrated at a gathering at the Living Computer Museum (https://livingcomputers.org) in Seattle. Mary Ann tells us a lot about the history of UNIX from a different perspective than Brian's, as she was part of the creation and spread of Netnews and UUCP mail. Mary Ann also has a story to tell about becoming a transgender programmer, beginning her transition while still at Lucent, the owner of Bell Labs.

Laura Nolan has more to say about SLIs and SLOs. Laura was very impressed with the work of MIT Professor Nancy Leveson, based on the keynote she presented at SREcon19 EMEA. Leveson has studied failures and accidents in complex systems, from waterworks to military air-traffic control, and come up with a better method for understanding complex systems. In the first of a two-part column, Laura examines the reference leg of this system, the input that controls the systems we use today, and ties in management's role to SRE.

Peter Norton discusses Python's memory management. Like other systems that must employ garbage collection, Python's design seeks to be as efficient as possible. But that system is generally opaque to programmers using Python, and Peter explains how to look beneath the covers, and he compares Python's GC to Java.

Mac McEniry expands on his column about handling `go` command lines with `cobra` (*;login:* Summer 2019) with `viper`. `viper` handles command-line defaults in a manner most of us should be familiar with, that is, that options used on the command line have priority, followed by the environment, then by defaults from configuration files.

EDITORIAL

Musings

Dave Josephsen found himself excited by a newish tool. eBPF has been around for some years now, and Dave has awakened to eBPF's promise of getting better and more specific insight into the workings of the Linux kernel. In this, the first of a two-part column, Dave compares an eBPF script to `iostat` for debugging problems with arrays of disks.

Focusing on cybersecurity, Dan Geer takes another look at job prospects under the growing impact of automation. Using data from the US Bureau of Labor Statistics, Dan helps us get real about where job and salary growth have been over the past decade, something that may be helpful if you are looking for a career or getting ready to jump ship to a new career.

Robert Ferrell ponders artificial ethics, the study of how inert electronics may appear, or not appear, to have any ethics at all.

Mark Lamourine has written three book reviews and managed by chance to parallel some of the topics that appear in this issue. Included in his reviews is one about Brendan Gregg's new book on eBPF. I review Brian Kernighan's *UNIX: A History and a Memoir*.

I've often mused about why CPU design appears conservative, meaning that certain aspects appear again and again in designs from many vendors. Sometimes, it's simply because other designs just don't work as well, such as Transmeta and Itanium, both very long instruction word (VLIW) designs. There are other designs, like Alpha and SPARC, that have hung on longer even though their performance isn't as good as what can be done with Intel-style processors.

I have tried to imagine what the ideal CPU design might look like, but the answer to that still lies in the unknowable future. For now, I am grateful for the CPUs that we have today, ones so powerful, and yet efficient, that we can carry them in our pockets. A very, very long road from the PDP 7, with 32 kilobytes of DRAM, that UNIX was written for.

29TH USENIX SECURITY SYMPOSIUM

BOSTON, MA, USA

Co-located Workshops

WOOT '20

14th USENIX Workshop on Offensive Technologies
August 10–11, 2020
Submissions due May 28, 2020
www.usenix.org/woot20

WOOT '20 aims to present a broad picture of offense and its contributions, bringing together researchers and practitioners in all areas of computer security. Offensive security has changed from a hobby to an industry. No longer an exercise for isolated enthusiasts, offensive security is today a large-scale operation managed by organized, capitalized actors. Meanwhile, the landscape has shifted: software used by millions is built by start-ups less than a year old, delivered on mobile phones and surveilled by national signals intelligence agencies. In the field's infancy, offensive security research was conducted separately by industry, independent hackers, or in academia. Collaboration between these groups could be difficult. Since 2007, the USENIX Workshop on Offensive Technologies (WOOT) has aimed to bring those communities together.

CSET '20

13th USENIX Workshop on Cyber Security Experimentation and Test
August 10, 2020
Submissions due May 19, 2020
www.usenix.org/cset20

CSET '20 invites submissions on cyber security evaluation, experimentation, measurement, metrics, data, simulations, and testbeds. The science of cyber security poses significant challenges. For example, experiments must recreate relevant, realistic features in order to be meaningful, yet identifying those features and modeling them is very difficult. Repeatability and measurement accuracy are essential in any scientific experiment, yet hard to achieve in practice. Few security-relevant datasets are publicly available for research use and little is understood about what "good datasets" look like. Finally, cyber security experiments and performance evaluations carry significant risks if not properly contained and controlled, yet often require some degree of interaction with the larger world in order to be useful.

ScAINet '20

2020 USENIX Security and AI Networking Summit
August 10, 2020
Talk proposals due March 27, 2020
www.usenix.org/scainet20

ScAINet '20 will be a single track summit of cutting edge and thought-inspiring talks covering a wide range of topics in ML/AI by and for security. The format will be similar to Enigma but with a focus on security and AI. Our goal is to clearly explain emerging challenges, threats, and defenses at the intersection of machine learning and cybersecurity, and to build a rich and vibrant community which brings academia and industry together under the same roof. We view diversity as a key enabler for this goal and actively work to ensure that the ScAINet community encourages and welcomes participation from all employment sectors, racial and ethnic backgrounds, nationalities, and genders.

FOCI '20

10th USENIX Workshop on Free and Open Communications on the Internet
August 11, 2020
www.usenix.org/foci20

FOCI '20 will bring together researchers and practitioners from technology, law, and policy who are working on means to study, detect, or circumvent practices that inhibit free and open communications on the Internet.

HotSec '20

2020 USENIX Summit on Hot Topics in Security
August 11, 2020
www.usenix.org/hotsec20

HotSec '20 aims to bring together researchers across computer security disciplines to discuss the state of the art, with emphasis on future directions and emerging areas. HotSec is not your traditional security workshop! The day will consist of sessions of lightning talks on emerging work and positions in security, followed by discussion among attendees. Lightning talks are 5 MINUTES in duration—time limit strictly enforced with a gong! The format provides a way for lots of individuals to share ideas with others in a quick and more informal way, which will inspire breakout discussion for the remainder of the day.

Registration will open in May 2020.