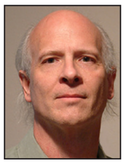
EDITORIAL

Musings

RIK FARROW



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

Change is hard. Once you have things working, however shakily, you realize that making changes could result in a slippery slide back to where you were. Yet change is critical if you are going to advance.

The SRE culture is built upon supporting rapid change and automating away every routine activity possible. But for people outside of that culture, change remains scary, as failure is always an option. And an unpleasant option at that if things were already basically working.

The first article in this issue, by Ben Purgason, got me thinking about the culture of change. There's also a lot about security in this issue, and the combination of the two reminded me of a security audit I helped with many years ago.

Changeless

I was sitting in a cubicle in a state-level IT department. The other part of the audit team had the easier job. They were running password sniffers, and on a late '90s mostly Windows network; they were laughing a lot as the sniffer scarfed up passwords. I, on the other hand, had to comprehend the Cisco router's access control lists, about 15 pages of them.

After a few hours, I could see I was wasting my time. The network guys weren't using most of the ACLs they had given me. They were a smoke screen. They were using exactly one ACL.

And that ACL referenced a system they hadn't told me about and were pretending wasn't there. I ran an early vulnerability scanner (SATAN) on the demilitarized zone (DMZ), looking both for systems and for vulnerabilities, as presented by network services, on those systems. And not only was the mystery system there, it was unpatched and vulnerable.

The IT group had a replacement system in place and ready to go. But they hadn't enabled it, postponing the change, one that would affect any user who logged in remotely in this large state. Not changing over appeared safer then leaving the old, dangerously vulnerable system in place. Fear of what might happen if they changed over was greater than the fear of penetration of their internal network. This was in the early days of intrusion detection, and ID was beyond the scope of the audit. I still find what was likely happening scary to think about.

I can look at myself, and see how often in my life I've been forced into making changes I should have made earlier. Systems that died long after I should have migrated them to newer hardware, not hardening security because of fears I would break things I had fixed long before, and changes forced upon me by advances in technology.

In many ways, the rapid pace of changes in the big Internet-facing companies of today is both daunting and frightening. They have learned how, not just to live with change, but to thrive. For the rest of us, these are important lessons that we need to learn as well.

The Lineup

Ben Purgason starts off this issue with the five stages of SRE, based on his experience working with the SRE teams at LinkedIn. Purgason begins this article, based on his keynote at SRECon18 Asia, by describing the three founding principles of SRE at LinkedIn: Site Up, Empower Developer Ownership, and Operations Is an Engineering Problem. Purgason then enumerates the five stages, as he experienced them, working in the trenches.

Gruss, Hansen, and Gregg examine the patches made to three major OSes in response to the Meltdown exploit found in recent Intel processors. The initial work was done using Linux on a single hardware platform more as an academic project, but the solutions tried there later became the basis for patches to Linux, macOS, and Windows. Gruss also discusses how, and why, this patch impacts performance. For the most common workloads, the impact on performance is surprising.

O'Neill, Seamons, and Zappala discuss their USENIX Security '18 paper [1], where they examine the failures in the usage of TLS in OpenSSL and how to avoid them. Instead of allowing programmers to set possibly insecure defaults, their solution moves TLS into the sockets interface of the Linux kernel, and uses a system-wide set of minimal defaults. The programming model is simple enough that adding support for TLS in a tool like `netcat` required only five lines of code.

Erik Poll explains the intersection of forwarding and parsing. Most service applications today consist of front and back ends, with the front end accepting requests, then distributing the work to be done to back-end services. The problem, as Poll points out, is where and how should forwarded input, or potential injection faults, be processed, as these may conceal forward attacks. Poll describes anti-patterns, techniques that have been shown to be flawed, and then suggests remedies that are safer and follow LangSec best practices, as found in his paper [2].

Peter Peterson wrote a summary of CSET '18, a one-day workshop about cybertesting, measurement, and experimental testbeds. The talks were wider-ranging than I would have thought, and Peterson provides succinct summaries of each talk and one panel.

George Amvrosiadis et al. have also taken a crack at experimental verification. In their USENIX ATC '18 paper [3], the group compares Google cluster-behavior traces to traces they have collected from two US defense/scientific and two financial analytics clusters. The Google cluster traces have been the standard for measuring the value of techniques in cluster research, but this group shows that Google traces might actually be outliers when compared to how many others use cluster computing.

Amandeep Khurana and Julien Le Dem demonstrate how database storage has changed over the decades. From IBM mainframes specifically designed for storing records, to SQL, and finally to the much more loosely structured data-lakes we see today, this team explains how and why data storage has evolved.

Peter Norton takes his column in an unusual direction. Peter focuses on the changes taking place in the Python community with the departure of its Benevolent Dictator for Life, Guido van Rossum. The community seems to be cautiously and carefully moving forward, perhaps to avoid disturbing the ecosystem that is Python.

David Blank-Edelman is taking this issue off.

Mac McEniry chose to write about how to access AWS S3 storage using Golang. His particular challenge was how to provide some configuration information to a widely distributed set of servers, and while his example does rely on S3, the ideas in his column can be used as the basis for other ways of sharing some small amount of data.

Dave Josephsen continues the discussion of flow that he started in his Fall '18 column. Dave explains the problems the team at Sparkpost ran into when they attempted to use `syslogd` to collect and distribute log messages from their front-end Nginx servers. Think "firehose" and you are getting the idea of the volume of log messages.

Dan Geer and Paul Vixie work at enumerating the number of systems attached to the Internet. Starting with named systems, they examine other means of at least estimating just how many devices are out there, a daunting task. The ballooning growth of IoT devices, along with generally poor security (by negligence) and inability ever to be updated bodes poorly for the future of Internet security.

Robert Ferrell considers the notion that we exist only as a great simulation. Other great minds, like Elon Musk, have proposed this recently, but, as always, Robert has his own take on our digital future.

Mark Lamourine has three book reviews, and I have written one.

Announcements

The year 2019 marks the 50th anniversary of the UNIX system (™), and Clem Cole, past USENIX Board President, has written an article that helps to explain just why UNIX has been so successful. I've read his article several times, and attempted to either create a "digest" version or split his article into smaller parts. In the end, I felt you are better off reading the original.

Clem makes many good points about UNIX, such as, unlike other operating systems of the time, UNIX was a system written by programmers for programmers. If you consider IBM's OS/360 using the perspective presented by Khurana and Le Dem in their article, you can see that Clem's point is valid: UNIX had a very different purpose right from the start.

Clem's article originally appeared in CNAM, a French publication that examines the history of technology sciences [4], and we present an English version of his article here [5]. If you plan on disrupting the usual course of events when it comes to new systems, I recommend that you take the time to read his entire article.

There was another event that occurred recently, although I suspect that it passed well below most people's radar. Early issues of *;login:*, starting with volume 8 in February 1983, are now online. Thanks to the efforts of USENIX staff, particularly Olivia Vernetti and Arnold Gatilao, you can now find these issues of *;login:*, up to the year 2000 at the time of this writing, on archive.org [6]. When USENIX made the transition to a modern web server design, some older issues were lost, and these are now also being hosted online again.

I can sincerely say that great changes are afoot. Not because I have a crystal ball or can pull predictions out of my naval lint, but simply because change constantly occurs. There is no stopping change, not even with death.

References

[1] M. O'Neill, S. Heidbrink, J. Whitehead, T. Perdue, L. Dickinson, T. Collett, N. Bonner, K. Seamons, and D. Zappala, "The Secure Socket API: TLS as an Operating System Service," in *Proceedings of the 27th USENIX Security Symposium (Security '18), pp. 799–816: https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-o_neill.pdf.*

[2] E. Poll, "LangSec Revisited: Input Security Flaws of the Second Kind," in *Proceedings of the IEEE Symposium on Security and Privacy Workshops*, 2018, pp. 329–334: http://spw18.langsec.org/papers/Poll-Flaws-of-second-kind.pdf.

[3] G. Amvrosiadis, J. W. Park, G. R. Ganger, G. A. Gibson, E. Baseman, and N. DeBardeleben, "On the Diversity of Cluster Workloads and Its Impact on Research Results," in *Proceedings of the 2018 USENIX Annual Technical Conference (ATC '18)*, pp. 533–546: https://www.usenix.org/system/files/conference/atc18/atc18-amvrosiadis.pdf.

[4] Conservatoire national des arts et métiers (CNAM), Cahiers d'histoire du Cnam, vol. 7–8, La recherche sur les systèmes : des pivots dans l'histoire de l'informatique, vol. 7–8: http://technique-societe.cnam.fr/la-recherche-sur-les-systemes-des-pivots-dans-l-histoire-de-l-informatique-ii-ii-988170.kjsp?RH=cdhte.

[5] C. T. Cole, "Unix: A View from the Field as We Played the Game," Cahiers d'histoire du Cnam, vol. 7–8, La recherche sur les systèmes : des pivots dans l'histoire de l'informatique, January 2 to July 1, 2018, pp. 111–127: www.usenix.org/login_winter18_cole_unix_cnam.pdf. This article is licensed under CC BY 2.0: https://creativecommons.org/licenses/by/2.0/.

[6] USENIX Archives: *;login:* magazine: https://archive.org/details/usenix-login.