So Where Are We? A Guest Opinion Editorial

RELIABILITY and reliability engineering have been long-standing, well understood, well defined, and practiced by a small community when one looks at the entirety of engineering. Of more recent, the hardware, component, and system principles of reliability and reliability engineering have gradually moved into the software arena. Today, terms like software reliability, software reliability engineering, software fault tolerance, and software resilience are common vernacular and main stream. And we are now able to achieve levels of ultra-reliability in software systems of uncompromising criticality, but unfortunately at extreme costs, and slowly.

Today, software is ubiquitous, pervasive, and ethereal. An interesting and important question is whether software, to most consumers and users, is now a *service* or *product*. Why? Because the answer to that question affects how we can attempt to *trust* it. Thirty years ago, the issue was easier to resolve: software then was a product that came with upgrade and maintenance fees, and was licensed to specific machine IDs. But with access to commercial cloud services, that offer software-as-a-service and massive data storage, software has become more of a service than a product to most consumers and users. This condition highlights the issue of *service reliability*, which in the case of software is more closely related to *availability* and *performance*, not semantic correctness.

However, when we step aside from purely focusing on reliability and look at security, a sibling or cousin attribute (depending on perspective) to reliability, we see that software security is far from being as mature of a discipline as software reliability. The reason for this immaturity is many-fold, and far too expansive and debatable for this editorial; however, a few principles are irrefutable. First, software security deals with *malicious intent*, a principle somewhat foreign to reliability. Secondly, software security has little notion of *operational profiles*, and instead relies on passwords, debugging code for known vulnerabilities, or scanning code for known virus signatures to only name a few. This second principle suggests that security is still a reactive art-form to the *knowns*, and does poorly with the *unknowns*. Both are true.

The importance of always being mindful of the inevitable conflict between developing trust from knowns versus unknowns cannot be overstated. A core reason that software reliability theory and practice evolved to the main stream position that it is today is due to its roots in the telecommunication industry. That field had access to all phone call records, and for decades. What a beautiful and rich set of data to explain how consumers made calls, e.g., where, when, length of calls, etc. Does security have anything similar, except for the reuse of known attacks or attack patterns on systems that were never properly fortified when the time to do so was ripe? And is the use of simulated, artificial attacks believable enough to fortify systems appropriately without over-fortifying or under-fortifying them?

Readers at this point might also ask "but what about firewalls, and what about encryption?" Admittedly, yes, those are additional approaches that the computer and software communities rely on. But let's look briefly at just one, then ask why, and what that might mean going forward.

Encryption and cryptography have been around in varying forms for thousands of years. The "why" has always been simple: to keep a piece of valuable information secret. Today, we live as much in a security and privacy-dependent world as we do a reliability-dependent world. We hear and read far more about cyberterrorism, cyber-warfare, hacking, and cybersecurity than we do about reliability. And the reason for this is that in many ways it is no longer the software that matters, but it is the data, that is, we now live in a data-dependent world. And we want our data "secret-ized." In computing, we've moved from main-frames to desktops to laptops to smartphones (wireless and with software apps), and to now a data-dependent and controlled world. Data are facts, and most successful decisions are based on facts. While it is true that software controls data, it is the data that contains the value. Data is now a "secret-sauce." For example, bankcard data describing personal details about millions of customers is likely worth more than the software's malicious attack signature and implementation that allowed the customer data to be compromised. Why? Because data has a far longer shelf-life than malware. Malware has a periodic genealogical pattern that suggests ancestry, and with each generation being replaced more quickly than current detection schemes can keep up with.

Don't be surprised if the grandest future engineering challenge is going to be in secure and private data engineering. No one knows how to do it well, yet. But the race is on. It's a mine, holding valuable new engineering principles for engineering non-physical artifacts (such as data) that affect everything. And while there are principles such as data integrity from yesteryears that apply, few if any of them address the scalability, heterogeneity, and temporal characteristics of the massive-scale big data that is on the horizon. Hence the road for enhanced cybersecurity must eventually pivot towards data as being both the castle and the keys to the castle, or cyber-insecurity will be the norm.

And finally, be on the lookout for two other secret-sauces: "things" on the Internet, and our old friend, the algorithm. The confluence of data, algorithms, and things are likely to upend the cybersecurity landscape for decades.

Just a thought.

Digital Object Identifier 10.1109/TR.2015.2408231

Jeffrey Voas