August 1, 2017

RE: Strengthening the Cybersecurity of Federal Networks and Critical Infrastructure: Workforce Development

The Information Systems Security Association (ISSA)® is a not-for-profit, international organization of information security professionals and practitioners.

We offer the following comments in response to a Request for Information (RFI) by the National Institute of Science and Technology (NIST) in the Federal Register. (https://www.federalregister.gov/documents/2017/07/12/2017-14553/strengthening-the-cybersecurity-of-federal-networks-and-critical-infrastructure-workforce)

The focus of this inquiry appears to be "cyber enabled workforce," which surely will continue to be a need as we rush to IT-enable almost everything, including devices that fundamentally neither need nor (in many cases) benefit from adding software (e.g., refrigerators and washing machines). More specifically, by "IT enabling" many mechanical devices, we create a large and likely insecureable attack surface and shorten the lifecycle of these devices, because the software cannot be maintained as long as the normal useful life of the product. We can still drive cars built in the 1930s: this would be impossible if those autos had contained software since the operating systems, hardware, compilers and other required equipment would not exist 80-some years later.

It is therefore well worth noting that at least two of the problems that "enabling a larger cyber workforce" is attempting to solve require fundamentally *different* solutions than "adding more cyber people." The first of these problems is the potential for the creation of systemic risk which, by definition, cannot be mitigated:

"In finance, **systemic risk** is the risk of collapse of an entire financial system or entire market, as opposed to risk associated with any one individual entity, group or component of a system, that can be contained therein without harming the entire system.[1][2] It can be defined as "financial *system* instability, potentially catastrophic, caused or exacerbated by idiosyncratic events or conditions in financial intermediaries".[3] It refers to the risks imposed by *interlinkages* and *interdependencies* in a system or market, where the failure of a single entity or cluster of entities can cause a cascading failure, which could potentially bankrupt or bring down the entire system or market."[1]

The second issue is that security has to be "built in, not bolted on," and for the most part, the workforce that creates software and hardware is *not* educated on even the fundamentals of how security must be engineered *into* software and hardware systems. This is a core failure of much of the university

[1] https://en.wikipedia.org/wiki/Systemic_risk

educational system that has remained largely unaddressed though it has been raised by many "consumers" of this workforce – the vendors that build hardware and software – for at least ten years.

More in-depth discussion of these issues follows.

As we continue to IT-enable absolutely everything, we are creating in many respects "indefensible" attack surfaces and more specifically, *systemic* risk, which by definition cannot be mitigated because it is creating a *fragile system.* The solution to systemic risk is risk avoidance, not "risk mitigation," which is inapplicable to systemic risk. Therefore, part of the solution to the "cyber workforce" problem needs to be the general populace having a greater understanding of the limits of technology and understanding risks – and not merely rewards – of technology enablement. We've already seen one of the largest distributed denial of service (DDoS) attacks that occurred by co-opting Internet-accessible cameras (https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/). Nobody wants to worry about their neighbor's toaster being co-opted by hackers and used to attack their home security system, but that is increasingly the world we are building because of what can only be termed the hubris of technology-enabling almost everything.

Part of any cyber education push therefore needs to highlight the fact that "IT enablement" may create – and often does create – more problems than can be "solved" with a larger or more enabled cyber workforce. We need to create not a larger cyber workforce but a "*cyber-aware*" workforce, lest we continue to have IT enablement without an understanding of the consequences. (There's a reason nobody says, and with apologies to Apple Computer, "Launch nuclear weapons? There's an app for that!" Launching nuclear weapons from your phone is, for now, understood to be a ridiculous thing to do.) Technologists often think more technology is always better: what we need are people who understand both the benefits of technology and the limits - and unintended consequences – of it. A core element of a "cyber-aware workforce" is that people ask intelligent questions about risk and they understand technology well enough for analytical inquiry, but are not necessarily technical *experts.*

To provide one example, we could educate people on better gauging risks vs. rewards. Rewards of cyber enablement are often easy to spot: I "save" money by not having to send a technician out to the field if I can get a digital readout on something. That all sounds wonderful; however, a better metric may be "risk equivalence." Risk equivalence may be thought of as a comparison as to whether the addition of technology will lead to greater risk and, if so, whether the risk can be adequately mitigated —and at what cost.

For example, in the "old" model, few people were able to engage the controls of a nuclear reactor. They were likely very well vetted, needed physical access not only to the plant but to specific areas of the plant, and had to pass through guns, gates and guards to wield these controls. That is a quantifiable, measurable risk. In the "new model," where everything can be done "remotely," anyone with knowledge (of exploiting a security weaknesses), and an Internet connected device could *theoretically* access these controls. Noting that all software has bugs, some of those are security related and may also be significant, what are the odds that these two scenarios will ever be risk equivalent? Do we even have people thinking about their ability to foresee, much less measure, downsides of technology deployment? We've already seen software-enabled cars hacked remotely (which was completely, utterly predictable). It's ironic that the mantra of many of those pushing "self driving cars" is that "flawed humans cause traffic accidents and self-driving cars is the remedy for that problem." And yet, these same flawed humans will, the mantra repeaters believe, design and build perfect self-driving cars

with no coding errors and no design flaws that could lead to a remote takeover of these cars, much less a remote takeover at scale (one flaw enabling massive hacks of affected vehicles).

The second problem involves the nature of what we have to defend. We keep recreating the same security issues as we deploy new technologies or find more places to use technology. For example, many developers fail to understand that they need to validate input – not just that it is "good" or "valid" input, but specifically that it is *not* "malicious" or "evil" input. Yet, many developers still write code that accepts input without properly vetting it, which leads to the potential to take over a system, or inject "bad" data, or other nefarious consequences. It's not merely a failure of programmatic norms ("validate all input") but for many individuals, a failure to understand that bad guys have unlimited patience to attempt to break something: they can and will find those coding errors, and exploit them. This problem has existed in client-server computing, in three tier systems, and will continue to exist so long as computing models evolve and developers fail to grasp "security 101."

Furthermore, many users of systems continue to run out-of-support software or hardware, don't patch promptly, and leave their systems vulnerable long after exploit code is available that "leverages" the ability to break fragile systems. A particularly egregious case of this was a large company, in a regulated industry, running on long out-of-support product, had never applied any announced security fixes from their vendor, and insisted that their core system (which was subject to regulation) was "too critical to patch." Had that system been compromised, it would have damaged their customers, their brand, and doubtless their earnings and share price. "Run on supported systems and patch regularly" is a *basic* security principle that many organizations still do not follow consistently.

To describe another, more recent example of "failure to learn from the past," it's not a surprise that the Internet of Things (IoT) has turned out to be, in many respects, the Internet of "0wn3d" Things, as people delivering these devices incorporate software in these devices (with no mechanisms to update it), or software that won't be maintained for the useful life of what it is embedded in (meaning that there will be old devices that can't be "fixed" because the software cannot be updated), or that was not designed for the threat environment (because the designer never thought about bad guys attacking a thermostat), or uses hardcoded passwords (that cannot be changed), or uses very old/buggy third party components. These problems are very well understood in the larger (mature) IT industry even if that industry is not yet "perfect" at addressing all these issues all the time. Manufacturers rushing to build an app to text you when your laundry is done seem to have to relearn these lessons. So does everyone rushing to make everything an IoT device.

Note: in particular, the "embedded third party libraries" issue is a microcosm of the perils of embedding software in devices when the software cannot be maintained for the life of the product. Specifically, much software is built by using third party components, many of them "open source." It's certainly both attractive and understandable that many people building software use *components* to do so, for the same reason that people building a house purchase lumber (rather than growing, cutting down and planing their own trees), nails (rather than mining iron and fashioning it into nails, rebar and so on), that they buy pre-made appliances instead of building their own, and so forth. Similarly, the use of third party components enables software development organizations to use their resources on innovation, and adding value on top of "pre-fab" components. However, many of these third party libraries are targeted by hackers (who can "break once, hack many times") if they are successful. Furthermore, if an embedded third party component is not maintained for the life of the product in which it is embedded,

the device becomes vulnerable if there are security vulnerabilities discovered in the components and they are no longer being maintained by the developers.

A secondary "externality" of putting software in hardware devices (like refrigerators, scales, toasters and the like" is that the software's useful life is most likely not as long as the device. This means more devices – including relatively expensive ones like appliances and automobiles – may end up in landfill once they cannot be maintained. That "shortened lifecycle and greater volume of devices in landfill" is potentially a significant "cost" to software-enabling these devices. *The lifecycle of software, firmware and even computer hardware is a limiting factor in the lifetime of anything incorporating or dependent upon such software, firmware and hardware.*

As noted, the lack of understanding of security principles and fundamentals is a far more pressing problem than "finding more cyber defenders." And yet, collectively, and despite pleas from many in the IT industry about the lack of basic security education in computer science and related disciplines - universities *still* do not teach principles of secure design and development. If they teach "security" at all, they teach "Kerberos" or "encryption" or "access control" instead of inculcating students with the foundations of secure engineering, which *must be a core element of every single university program related to computer science, or coding – the breadth of what we use computers for.*

In the past, one large IT vendor wrote a letter to the top 10 or so educational institutions from which the vendor recruited, indicating that this vendor – and many other large vendors – had to educate "new hire" computer science graduates in *basic* secure development methodology and *basic* security principles. The vendor also pointed out that industry spends millions and millions of dollars fixing avoidable and preventable security issues, and that customers generally don't beg for, much less want to apply more patches: they prefer nice, stable, well-designed and secure systems. The vendor asked that the curriculum at these institutions be modified to include security across the spectrum of computer science and related classes. Only one university replied, asking for money to change their curricula (noting they were a well endowed institution and could have well afforded to make this change). Another university's representative asserted that knowledge of security was "training," and insisted that the university's quest was "education." Clearly universities do not, for the most part, understand that security is both a core cultural value for those building IT-reliant systems and that there must be educational *underpinnings* to develop and reinforce a security culture.

Putting it differently, civil engineers cannot graduate without taking a structures class, and building upon those principles of structural engineering as they take upper level classes. Why, then, do so many universities think "security" or "secure development practice" is somehow *not* a core part of computer science? (The same argument can be made for privacy and will become increasingly important as the European Union's General Data Protection Regulation (GDPR) requirements come into play, as "privacy by design" becomes a required element of development practice if one wants to do business in the European Union.)

To reiterate: without a fundamental transformation of the educational inculcation of "coding" or "computer science," we will continue to fail at securing our systems, and we will never —ever —have enough "cyber personnel" to defend the indefensible, any more than you can add rebar to a bridge after it's been erected and hope to avoid catastrophic structural failure. The key way to force this change upon universities – who, at best, have tried to institute a "cyber security major" instead of fixing the underlying problem—is to engage with (if not force) accreditation bodies for these educational

programs (e.g., ACM) to change their accreditation requirements. The curricula for computer science (and related majors) must require not only security education as a foundational course, but must incorporate security into *each* class *so that it is reinforced throughout the relevant curricula.*

One way to do this easily is to add material to the textbooks that are already in use. A small amount of grant money could be used to pay knowledgeable security professionals to add security-related examples to these textbooks, which means "teachable material" would be incorporated into the totality of what students are exposed to. A large IT vendor noted a few years ago that many of their "coding samples" in documentation were *not* written to their own secure coding standards (and were then amended to be "secure" code samples); in addition, documentation writers were required to take secure coding training. Why it matters: often, people will "cut and paste" sample code and use it in their applications. Thus "bad /insecure code" – even if it is sample code - propagates into deployed applications. Similarly, showing examples of *good* security and security principles throughout relevant curricula will reinforce the importance of (and the techniques of) good security practice. There are many large IT vendors who already have secure coding training as a part of their secure development lifecycle and could be engaged as resources (some have banded together via Safecode (https://www.safecode.org/) to propagate secure coding/secure development practice). Note that the principles of understanding code security – and insecurity – will increasingly need to be part of other disciplines: e.g., mechanical engineering, since machinery is increasingly software-enabled.

"Fixing the computer-related educational system" absolutely must be an underpinning of any success we wish to have in securing infrastructure. If systems are not designed, built, and maintained to *be* secure, we cannot "add cyber-bodies" to fix this problem . Yes, we will need more cyber professionals, because the IT-ization of almost everything is not likely to cease or even slow down. But we should need fewer "defenders" and have more defensible digital terrain. Moreover, general knowledge of cyber security will need to permeate other disciplines. As an example, people who are licensed to use firearms are not generally experts on ballistics or gun manufacturing. Yet, through gun safety classes, licensing regimes and other mechanisms, gun owners know how to operate guns *safely.* Gun safety has also become a cultural norm in the military (who wields weapons more broadly and more often than any other demographic), as anybody who has ever been on a military gun range knows.

Ultimately, unless and until "security" becomes a cultural norm, we will fail at it, no matter how many cyber personnel we train, hire and promote.

Thank you for the opportunity to share our comments.