This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Looking Backwards (and Forwards): NSF Secure and Trustworthy Computing 20-Year Retrospective Panel Transcription

Carl Landwehr | University of Michigan
Michael K. Reiter | Duke University
Laurie Williams | North Carolina State University
Gene Tsudik | University of California, Irvine
Trent Jaeger | Pennsylvania State University
Tadayoshi Kohno | University of Washington
Apu Kapadia | Indiana University Bloomington

The U.S. National Science Foundation (NSF) celebrated the 20th anniversary of its research funding programs in cybersecurity, and more generally, secure and trustworthy computing, with a panel session at its conference held in June, 2022. The panel members, distinguished researchers in different research areas of trustworthy computing, were asked to comment on what has been learned, what perhaps should be "unlearned," what still needs to be learned, and the status of education and training in their respective areas of expertise. Laurie Williams covered enterprise security and measuring security, Gene Tsudik commented on cryptographic security, Trent Jaeger addressed computing infrastructure security, Tadayoshi Kohno reviewed security in cyberphysical systems, and Apu Kapadia provided insights on human-centered security. Michael K. Reiter chaired the panel and moderated questions from the audience. This report provides a brief summary of NSF's research programs in the area and an edited transcript of the panel discussion.

As *IEEE Security & Privacy Magazine* celebrates its 20th anniversary, the National Science Foundation (NSF) Secure and Trustworthy Computing (SaTC) program also celebrated the 20th anniversary of its funding for research in this area at a meeting in Alexandra, VA, in June 2022. As part of that celebration, I was asked to organize a panel to look back on 20 years of projects in this broad area to try to assess the progress we've made and the directions we should be heading. Because the space is vast, I broke it into a few reasonably distinct areas, with a panelist for each. I chose

Digital Object Identifier 10.1109/MSEC.2022.3208721
Date of current version: 25 January 2023

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

- enterprise security and measuring security, to be covered by Laurie Williams, Distinguished University Professor, North Carolina State University
- cryptographic security, to be covered by Gene Tsudik, Distinguished Professor of Computer Science, University of California, Irvine
- computing infrastructure security, to be covered by Trent Jaeger, professor of computer science and engineering, Pennsylvania State University
- cyberphysical systems (CPSs) security, to be covered by Tadayoshi Kohno, professor of computer science and engineering, University of Washington
- human-centered security, to be covered by Apu Kapadia, professor of computer science, Indiana University.

I was delighted that these distinguished academics accepted my invitation to participate.

For readers unfamiliar with the SaTC program and its history, in the fall of 2001, a few weeks after the 9/11 attacks, I arrived at the NSF as a newly minted program director to direct a new program called *Trusted Computing*. The program was organized and championed by Kamal Abdali and Helen Gill under Růžena Bajcsy, then the NSF assistant director for Computer and Information Science and Engineering (CISE). Although the NSF had funded some cryptographic research, this was its first direct foray into what now would be called *cybersecurity*. Helen and Kamal crafted and released the initial program announcement; I just had to encourage submissions and organize reviews for the proposals as they came in. The call yielded around 140 proposals, and we were able to fund about 37 of those with a budget just under US$5 million. Under Peter Freeman, the program expanded into "Cyber Trust" with a broader scope and a budget of US$25 million or more.

A few years later, it became Trustworthy Computing, and finally, SaTC as it is today, a multidisciplinary program that embraces activities in half a dozen of the NSF's research directorates, well beyond CISE. A recent search turned up more than 1,000 active awards with a total awarded value approaching $US400 million.

Although I was unable to chair the panel in person, Michael Reiter, the James B. Duke Distinguished Professor of Computer Science and Electrical and Computer Engineering, Duke University, and the principal organizer of the SaTC 20th anniversary principal investigator (PI) meeting, capably took that role. The transcription that follows has been lightly edited; audience questioners are left anonymous.

Mike Reiter: This is the 10th anniversary of the SaTC program and 20th anniversary of the programs that the NSF dedicated to cybersecurity funding. So, we thought it would be a good opportunity to take inventory of where we are and what we have accomplished as a field but also to kind of chart the course for the next 20 years. As part of this, we wanted to do a retrospective panel on the SaTC program. We laid out several questions for the panelists to address.

1. What have we learned? (List two or three milestone achievements.)
2. What we should unlearn (that is, where should we stop wasting time, as a field)?
3. What do we still need to learn (which problems are still worth pursuing)?
4. How have we done with education and training, both of our own students and of the public at large?

Enterprise Security and Measuring Security

Laurie Williams: My background is in software engineering plus security, so that's my perspective. Enterprise security for me is security done by the enterprise.

What Have We Learned, and What Are Some Milestones?

Exactly 20 years ago, Bill Gates's Trustworthy Computing memo went out to Microsoft. That marked a big change, a new emphasis on building security into a product, not just adding security features but building features securely. It led to Microsoft's contribution of the Secure Development Lifecycle, which lives today. A big lesson was to build security into a product rather than "penetrate and patch." Another milestone was the advent of cloud computing in about 2008. There were big changes (pros and cons). Enterprises no longer needed comprehensive security expertise but could employ a shared responsibility model, relying on the cloud provider for basic security expertise, functions, and certification. The enterprise still needed to secure its applications. On the downside, moving to the cloud could increase enterprises' vulnerability to Internet-wide attacks, including denial of service attacks.

Another milestone was the introduction of continuous integration/continuous delivery (CI/CD), DevOps, and later DevSecOps, which changed enterprise security. It's not so clear whether this was a gain or a loss. It's fast paced, with a focus on tooling. Overall, I saw benefits in getting vulnerabilities and defects fixed faster. It increased accountability to engineers. Earlier, it was common for engineers to throw a new release over the wall, expecting someone else to deal with its flaws. With CI/CD and rapid deployment, it was the engineer who produced the code who got the call in the middle of the night if something happened.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

That accountability drove better code review and unit testing. It also enabled continuous compliance, beneficial for government organizations, and provided a way to get security fixes out faster. In 2014, Tesla and Jeep takeovers made news; Tesla was able to update its vehicles over the air, while Jeep mailed USB sticks to its customers. In 2011, Amazon Web Services started using formal methods to check its customers' security policies and configurations, which was a good event. More recently, we've seen how many enterprises are vulnerable to attacks through the software supply chain.

What Do We Need To Unlearn? How Are Enterprises Wasting Time?

I see time wasted in the overreliance on penetration testing, which finds bugs very late in the development cycle, and in adversarial relationships between security and development teams. Also, I see security teams advocating measures such as "fix all static security alerts" that are not really risk-based. Not having risk-based guidance, but rather proclamations, can waste time.

What Do We Need To Learn?

We need to understand how to fit security into the developers' workflows so they can develop securely in the context of all their other challenges. We need to manage the social interactions that need to take place in this context.

Education

Education is getting better for enterprise security. From an academic standpoint, U.S. accreditation criteria require security to be throughout the computer science (CS) curriculum. That's helping out. We do need more just-in-time and practitioner education.

Switching to Measuring Security

First, why do we want to measure security? We need to decide when a product is secure enough to ship. How do we know if a voting system (or any system) is secure enough to ship? The question is always posed for a particular product, and the answer must consider the criticality of that product. There's an Executive Order (Executive Order 13691) that advocates a lot of security practices, but do we really know that they help? What areas of a product seem to have the most vulnerabilities? How can we measure that? Are those vulnerabilities likely to be exploited?

What Have We Learned About Measuring Security?

There are a number of industry-wide initiatives. Starting in 2005, the National Vulnerability Database (NVD) was established, where vulnerabilities are reported (a milestone), but we have to recognize that it captures only a small fraction of the vulnerabilities known. Two industry-wide taxonomies on security practices were developed, starting in 2008: the Building Security In Maturity Model (BSIMM) and the OWASP Software Assurance Maturity Model (SAMM). Very recently, the Linux Foundation has started the OpenSFF Security Metrics project to provide some guidance on whether a component is a good choice. Recently, there has been movement toward having machine-readable vulnerability databases, bringing in not only vulnerabilities from the NVD but other sources as well.

What Do We Need to Unlearn?

We need to stop treating all vulnerabilities as equal. We need to be able to identify those that are high risk and those that are exploitable.

What Do We Need to Learn?

The biggest hurdle to measuring security is: what is the dependent variable, and how do we measure it? What does it mean to be secure or not secure? We can't just use the number of vulnerabilities reported to the NVD because popular software may be more likely to be attacked and have its vulnerabilities revealed. Less popular software may have as many or more vulnerabilities, but they may not (yet) have been revealed. Overall, we need to make more risk-based decisions.

Teaching Security Measurement

I think we haven't done a very good job, and we have a long way to go to be able to teach people about measuring security.

Cryptography

Gene Tsudik: As part of my spiel, I was asked to criticize certain things. So, be ready for macro-aggressions, and note that this is now not a "safe space."

Advances/Milestones in Crypto

Starting in the early 2000s, ID-based encryption (IBE) and hierarchical IBE come to mind, and fully homomorphic encryption is definitely a real highlight of the last 20 years, followed, in no particular order, by oblivious random-access memory, private information retrieval, and differential privacy. Common applications for these advances, whether by design or not, include outsourced or cloud-based privacy-preserving computing of all types.

What to Give Up On

Let's start with anything to do with order-preserving encryption and its application; for example, CryptDB comes to mind. Provable obfuscation and watermarking

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

I would give up in a second. Also, achieving differential privacy in any practical fashion.

Just a pet peeve: let's stop sticking Merkle hash trees and Bloom filters everywhere they belong and don't belong! And finally, although this may seem like ancient history, let's forget about wireless sensor networks and multihop mobile ad hoc networks. These are mythical creatures, like unicorns, that no one has ever seen, but many of us wrote lots of articles about them.

But though I would like to forget about those things, I don't hold out much hope that they will disappear because, even though they were already insecure in the 1980s, privacy homomorphisms and knapsack crypto systems are still not gone and, believe it or not, people are still working on them nowadays. It's like hundreds of years after Galileo and Copernicus, we still have the Flat Earth Society!

What to Learn

We need to learn how to achieve arbitrary arithmetic or computation, something like what fully homomorphic encryption gives us, but with guaranteed or provable integrity and correctness. This is not out there yet, but I hope it will be. Also, I hope we will see more practical privacy models, for example, "indifferential privacy." Maybe one day we will learn how to build provably secure, including side channel-free, hardware. Maybe one day we will know what "provably secure" means!

Education

Not much time is needed to think about this: we just suck! Crypto and security should be introduced in middle school, just like they introduce health, sex ed, and basic hygiene. We should teach Internet hygiene (that is, security/privacy) early on because, in some ways, it's more important. Things like number theory, especially the kind of number theory used in crypto, are fairly elementary, much easier than trigonometry, and could be taught early. At the college level, most CS courses, except maybe for graphics, should have a module or two on the security of whatever they are teaching. Many schools already do that, but we are not quite there yet.

How can we teach better and simpler? I think that crypto can be taught through humor. We have a dearth of humor in our society, especially in the research community. There are things out there that I use in my elementary crypto course, like the well-known Zero-Knowledge Cave or Quisquater's article from the 1980s appropriately titled "How to Explain Zero Knowledge Protocols to Your Children." There is also the famous Dining Cryptographers Problem that you can explain to a child. Things like that.

Here is a random comment in conclusion. As the world reels from dual pandemics of COVID and mpox, we (the security research community) are also suffering from two viruses: machine learning and blockchains/cryptocurrencies!

Computing Infrastructure Security

Trent Jaeger: Those of you working with computers in 2002 were probably dealing with some kind of Internet worm event. In those days, all it took was one vulnerability for the adversary to gain control of your host. And if adversaries wanted to control other hosts, they could easily propagate the attack using the worm to gain control of a large fraction of the Internet. Researchers at the time were thinking about how fast could they infect all the (Windows) machines on the Internet—could it be done in an hour? And that's about where we were with infrastructure security in 2002.

We've made many improvements since then. Since then, a lot of things the research community worked on actually were adopted in the commercial sector.

Mandatory access control: There was little access control in commercial operating systems beyond discretionary Unix access control, but eventually, mandatory access controls (MACs), originally developed for military applications, were adopted by the commercial sector, and MACs became broadly available. For example, Android has several versions of MAC that it relies on.

Trusted computing base: In 2000, pretty much everything was in the trusted computing base (TCB). This meant that once your one network-facing daemon was compromised, your whole machine was owned by the bad guy. So, reducing a system's TCB became an important goal, both in the research and commercial communities, and led to the emergence of things like virtualization. In addition, new hardware to reduce the TCB was proposed (for example, the Trusted Platform Module). Some of that hardware has created other challenges, such as those introduced by side channels, but it was important that the research community both propose solutions and vet those solutions so that we can improve.

Program analysis techniques: One big change from 2002 has been the utilization of program analysis techniques of various kinds both to find vulnerabilities and to detect and analyze malware. This was quite a new field in 2002 and has grown tremendously in importance.

Software defenses: A variety of software defenses have been proposed in the past 20 years by the research community. A challenge we face is how to pull these together and utilize these defenses effectively and efficiently.

What We Should Unlearn/Learn

We've unlearned quite a few things already, so I want to address some things we should perhaps relearn. Something I thought was a little odd 20 years ago was that people would stand up at meetings and assert that if

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

we would only use the secure operating systems of the 1970s, like MULTICS, for example, we wouldn't have all these problems. I thought that was hyperbole, but today, I find while teaching software security classes to graduate students that they don't have the vaguest idea of any of that work. It's been buried in the annals of history. But when we build software today, whether operating systems or applications, we are facing the same kinds of threats that people faced in the 1970s and 1980s, and we don't have any good case studies for how to build a server application, a client application, or a cellphone app, in a secure way, from beginning to end.

The best documentation I've ever seen on how to try to build secure systems, the challenges faced, and the approaches to overcome those challenges was recorded by people attempting to build secure systems, especially secure operating systems, back in the 1970s and 1980s. If you talk to Peter Neumann, he will point you to the documentation for PSOS, which includes hundreds of pages documenting the full thought processes of what they were trying to do to build this artifact, the design for a secure operating system, PSOS. You may not be building a secure operating system now, but something we might want to fund are studies of how you would build software for various applications that aim to achieve concrete security goals, from beginning to end.

Education

About education, I tend to agree with Gene, but I think education has improved slowly, albeit surely. Last term, I taught a software security class to seniors. They knew about security, about some of the attacks in the news, and the buzzwords, but they didn't really think at all about programming securely. We did a make-it/break-it/fix-it exercise, and they were all perfectly happy to write programs using many unsafe functions, such as the *strcpy* function, which should be a knuckle-rapping offense in the first class where you find it! A challenge we face, still, is: how do we get the students to think about security and functionality concurrently? How do we build tools that will enable them to think about those things at the same time?

CPSs

Tadayoshi Kohno: Twenty years ago, computer security and privacy research for CPSs was still a nascent field—not a field centered in mainstream computer security venues. That has changed.

CPS security and privacy issues are, for example, now listed as in the IEEE Symposium on Security and Privacy call for papers, along with other important new security and privacy subareas. In my list of achievements, this is item #1: that CPS security and privacy research is a focus of our field and receiving significant attention.

For someone not working in CPS security, it would be reasonable to ask: why is it important that CPS security research is now a focus? One answer is that there is the potential for significant harm with CPSs if security and privacy risks are not proactively mitigated. For example, there are risks to safety if a wireless computational implantable medical device; a self-driving car; the power grid; or a telerobotic surgery system is compromised.

I am not trying to scare people and say that we are at imminent risk of security attacks against medical devices, vehicles, or certain other classes of CPSs, like telerobotic surgery systems. While I can't say the same for systems like the electric grid, I do not recall any instance of anyone maliciously compromising a wireless medical device or an automobile.

But I also know the following to be true. It is because of research done by this community—and funded by the NSF—that millions of automobiles and countless medical devices and I'm sure other CPSs are more secure. If an adversary were to manifest today, they would have a much harder time compromising the security and privacy of medical devices and automobiles than they would if we—our research community—had not done the research that we did.

In my list of achievements, this is item #2: that millions of CPSs are safer and more secure because of this community. This community provides scientific foundations for industry and also keeps industry in check.

Let me now dive more deeply into the research that my colleagues and I did with respect to wireless medical devices and automobiles. There are, of course, many other important works to consider. But, as a coauthor on these projects, I am better positioned to talk about these works. Diving deep into these projects will allow me to cover a few key lessons.

For context, as a researcher, I often ask myself: what will be the next hot technology in five, 10, or 15 years? I try to predict what security and privacy threats and risks might manifest with those technologies. And I try to develop a foundation for mitigating harm. Sometimes, my process involves experimentally analyzing the security and privacy of a present-day version of that artifact and then extrapolating from those results into predictions for the future.

With that framing in mind, in 2006, Prof. Kevin Fu, now at the University of Michigan, and I began a research program focused on wireless medical device security. We obtained a short-range wireless implantable cardiac defibrillator. We experimentally discovered that an unauthorized party could change the setting on

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

that defibrillator and even wirelessly cause it to emit a large shock.

Similarly, in or around 2008, Prof. Stefan Savage, at the University of California San Diego, and I began a research program focused on the modern automobile. We obtained two 2009 sedans of the same make and model and experimentally analyzed them. Our car had a built-in cellular modem that allowed the car to effectively call 911 if it got into an accident. We found that we could call our car's phone number; play the appropriate modem tone to switch to an in-band modem; play more modem tones to bypass an authentication vulnerability; and then play more modem tones to exploit a buffer overflow vulnerability that resulted in our own code running on the car's telematics unit. From there, we could remotely and interactively control the car's engine, brakes, and more from anywhere in the world.

I want to step back and repeat that the security and safety risks of adversarial compromise to medical devices or cars were low. It took a significant amount of effort to do what we did. Any unauthorized party developing or implementing our methods might worry about leaving an incriminating forensic trail. And the technologies were still primitive compared to today.

Also, because the technologies were primitive compared to today, they were more agile. Not that they were agile, but they had fewer legacy artifacts to contend with than if vulnerabilities were discovered for the first time today. This leads me to my first lesson. Then was the right time to do our research—a time before the technologies became more sophisticated and harder to secure and when the capabilities, and hence the potential harms, of the technologies were fewer.

But even though the risks were low, government and industry mobilized. Full mobilization was not immediate—these were industries that had little to no prior experience with computer security. Today, these industries by and large consider security and privacy important priorities. Further, our research spawned numerous follow-on efforts to replicate our findings on other systems. For example, Charlie Miller and Chris Valasek later evaluated similar attacks against other automobiles. Such additional research further galvanized industry and government attention on securing CPSs.

And, over time, entire research and industry venues emerged around automotive security research and CPS security in general. Naturally, these venues did not arise just because of our work but because of the great work being done by so many people in this field. As a reminder, please check out the CPS breakout report from this PI meeting once the report is available.

This brings me to my second lesson. Before our research, there were prior studies that considered automotive security. But it took our work, which discovered real vulnerabilities with real cars, to catalyze the automotive industry into taking security and privacy seriously. The challenge is: how can we as a field learn to prioritize security and privacy research for technologies that have not yet been demonstrably hacked? It is comparatively "easy" to write an article that defends against a known attack. It is harder to write an article that both hypothesizes an attack scenario and then provides a defense against the hypothesized attack.

My third lesson is the following. I would like us as a field to broaden our definition of what constitutes a CPS, though some people already share this broader definition. For example, consider a head-mounted mixed-reality display. With such a display, it is possible to display virtual content in the context of the physical world. Think about, for example, Pokémon GO.

Prof. Franzi Roesner and I have a 10-year research program at the University of Washington (UW) focused on security and privacy for mixed reality. Together with two neuroscientists, we wrote a think piece on how adversarial applications running inside a mixed-reality head-mounted display might manipulate a person's perception of the world, sometimes with long-term impacts. Given the intimate relationship between mixed-reality content and the physical world, I personally consider mixed-reality devices to be CPSs and hope our community considers them seriously over the next 20 years. DNA sequencing and synthesis should also be considered CPSs.

Education

On the education side, it is hard for me to know, globally, how much students learn about CPS security and privacy. However, I observe that CPS security and privacy issues have entered the popular culture and media, and hence, CPS security- and privacy-related concepts have become more accessible and intuitive to students. We can't always teach students how to solve all CPS security problems, but we can at least give them an awareness of the problems and the tools to think about them. In our undergraduate computer security class at UW, we always feature threat modeling exercises involving CPSs and other emerging technologies. I encourage other educators to do so as well if they aren't already.

Wrapping Up

In summary, while the battle isn't over, I am thrilled that research done over the past 20 years has made today's CPSs more secure. The challenge for us, over the next 20 years, is to proactively identify the next emerging technologies and to work toward proactively securing them, too, and I hope I am here to listen to what is discussed in SaTC's 40-year retrospective.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Usable/Human-Centered Security

Apu Kapadia: I'm representing the human-centered computing track and the usable security area. This is a large, multidisciplinary field, and I want to acknowledge that my opinions reflect a computer scientist's perspective. I want to start by thanking the NSF; SaTC wholeheartedly embraced human-centered security at its formation 10 years ago, and I thank them for that. I also want to thank Lorrie Cranor and Jean Camp for brainstorming ahead of this discussion.

What Have We Learned?

Since we are a younger community than some of the others here, I will speak to our milestones more generally. The first milestone, around 20 years ago, was to recognize that usable security was important. Even back in 1975, Saltzer and Schroeder had recognized psychological acceptability as one of their core security design principles. So, yes, the importance of usable security and the human-in-the-loop has been acknowledged for many years, but when did it actually gain traction? I can point to a couple of early studies. Mary Ellen Zurko recognized the concept of usable security or user-centered security in an article published in 1996. That may have been the first mention of the concept in our community. She and her coauthors argued that usable security, that is, considering the human, shouldn't be just any goal in designing a security technology, it should be a central goal. Unlike other security mechanisms, such mechanisms recognize the importance of the user's role in keeping systems secure.

Angela Sasse wrote in 1999 about how "users are not the enemy." Engineers might have claimed that they built awesomely secure systems and that it was the users just not doing the right thing. Today, attitudes may have changed, but this insight, that the engineers need to design better systems and not blame the user, was an important shift in 1999. So, the first milestone was acknowledging that we do have a problem and we need to think about usable security mechanisms instead of blaming the users of these systems.

If milestone one was about recognizing the problem, milestone two was about moving to do something about it, that is, moving to a more systematic study of usable security. Over the last 20 years, we've made a great deal of progress. I've been part of one particular community, the Symposium on Usable Privacy and Security (SOUPS) conference community, and I've noticed the progression of how research methodology has improved over the years. (I acknowledge I'm scared to reread some of my early articles!) We've become much better at using social science techniques, which have not been traditionally taught in the CS curriculum. Many of us coming from the CS community were not trained in these techniques and have had to learn a lot along the way, and I've seen dramatic improvements in the rigor of our methods over the years.

In terms of research areas, to pick a few broad examples, we have gotten better at security and privacy interfaces. But usable security is more than designing interfaces. It's also about understanding and shaping people's security behaviors, that is, understanding the psychology, the sociology, the economics—the behavioral perspective of people's security actions. For example, my colleague Jean Camp at Indiana University studies people's mental models—how do you communicate security risks to users effectively and improve their decision making? You need to understand how people think and perceive the world of computing if you want to influence their interactions with systems. There's been a fair deal of work on security warnings and how to improve them.

We've even discovered various paradoxical behaviors, highlighting that you shouldn't just build a system, thinking that humans will find your system important or useful. Many of your assumptions about what people want or how they will behave will be challenged. Results on paradoxes include Alessandro Acquisti's lab's work on the control paradox, for example. If you give people more control through privacy settings, you might think they will be better at managing their privacy, but it turns out they end up sharing more, leading to harm for themselves. In my own research, we have some recent work showing that a privacy warning in the context of photo sharing—"Are you sure you want to share this photo as it might infringe on someone's privacy?"—backfires; people receiving the warning are on average *more* likely to share the photo. This is why it's important for us not to make assumptions about how people will behave without studying their behaviors empirically.

Milestone three was that once we settled into a systematic study of usable security, we embraced interdisciplinarity. I like that the SaTC Dear Colleague letters have been pushing us to pair up with social scientists and catalyzing collaborations. These are important structural improvements. As a community, we've been publishing more with social, behavioral, and economics researchers, which in the end improves not only our methods but also our broader understanding of the problem and the potential for multifaceted solutions.

Milestone four was to embrace the concept of inclusive security and privacy—security and privacy for all, for marginalized and vulnerable populations: for example, the visually impaired, victims of intimate partner violence, undocumented immigrants, and incarcerated people. We are just starting to study and understand the security and privacy needs of various subpopulations.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Where Do We Go From Here?

In our next milestone, we need to move from point solutions to broader frameworks and systematization of knowledge. As a community, we've already started to do so, and I'm glad to see this as our next major focus of research.

We also need to engage more with the ethical choices of our designs. The ethics panel (earlier in the meeting) focused on how to be ethical in conducting research, particularly in relation to human subjects and ethics review boards. But we should also consider the ethical implications of our choice of research topics. For example, should we be designing machine learning algorithms that aim to determine an individual's gender?

In my lab's work with visually impaired people who might wear augmented reality glasses as an assistive device, we wanted to know what kind of information they would want about the people in front of them. And you might think they would want to know the person's gender, height, and other visual characteristics. Our visually impaired participants felt this would be crossing the line—they don't fully trust artificial intelligence and worry such systems might misrepresent people. And as a society, we are recognizing that gender is not a binary concept, and so can you really develop an algorithm to visually gender people? More broadly, we need to consider the potential for harm when building solutions that might seem to benefit one population but then might harm others.

What Should We Unlearn?

First, and maybe our subcommunity already knows this, but usability and security are *not* a tradeoff! It's not the case that if you make a system more usable, it will become less secure. I also see some people with very strong opinions about how we should not have security warnings, saying that we should just design these systems to be secure in the first place. But more generally, we must recognize that you cannot eliminate all the risks. We must help people manage the risks and present them with the information (or warnings) they need to make informed choices.

Now thinking more broadly about the community, coming from the perspective of reviewing articles, I think we must unlearn this obsession with technical novelty. I see a lot of articles getting rejected for not being technically novel enough, as if this is some kind of competition for technical superiority. What I have learned in the human-centered community is that our heart has mostly been in the right place, in focusing on the problems people face and trying to solve these problems scientifically, as opposed to devising a really amazing algorithm as being the central goal. When reviewing articles, I hope the more systems-oriented reviewers will think of the bigger picture and the overall impact of the work.

Audience Q&A

Reiter: I'm starting us off with a question for Laurie. You made a comment about deciding when a product is "secure enough to ship," a favorite topic of mine. I've marveled at our collective inability to reach a consensus on that. Other fields, I think, do a much better job of deciding what is safe enough to ship. When a plane flies, the builders put together an assurance case for it to convince the U.S. Federal Aviation Administration that it's safe enough to fly. But we seem to leave this to each and every vendor of products. What makes this so hard in our case? And is there any prospect for improvement?

Williams: It is hard. I have a background in software reliability; there is a process—what's the acceptable arrival rate of failures during testing? There are mathematical and statistical models, and I do have some work where we are trying to translate that into security, but it's difficult in two respects. One is that testing has to be with malicious intent as opposed to just general functionality testing. But really, the hardest thing is asking what's the "right" number of vulnerabilities. We don't have a good way to answer that question; we don't have that dependent variable. I'm hopeful that in 20 years, even in five years, we'll have the analogy to reliability where we have statistical models, and we understand what the right arrival rate is and what the right vulnerability density is, to make that informed decision.

Audience: If you had a magic wand, and you could create any capability, what would that capability be, from a security point of view?

Kohno: I'll jump in. "Capability"—how to define it? Maybe helping make sure that everyone in industry creating technology and all policymakers creating technology policy understand the presence of an adversary and how hard it is to design for and to think about systems in the presence of an adversary. So, the magic wand is around making that happen. I think that connects to education, not just in the computing field, but the education of everyone who makes decisions around computing technologies and the intersection between technologies and society.

Tsudik: I like Yoshi's interpretation. Being labeled a crypto person, I have to think about that domain. If I could wave a magic wand, I would have guaranteed "time-lock" security: something that guarantees me integrity and/or secrecy for a predefined amount of time so I can just sit back and relax. That would be my wish.

Kapadia: I might have an impossible request, but in thinking about moving to solutions that are not just point solutions, but that affect multiple populations, I've thought about how we create solutions that are great for one community but horrible for others. For example, you might build better parental controls, but then, these can be used for surveillance, such as intimate

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

partner surveillance. So, the magic wand would be: how can we create technologies that can't be misused? Now, I know that's not possible, but as a research community, we need to work toward preventing or minimizing, as much as we can, these possibilities and abuses of technology.

Williams: I'll take off on that prevention theme. One of the times I was speaking with someone from industry, they had a quote, which I repeat a lot: "We can no longer do hand-to-hand combat with individual vulnerabilities." We can't handle them one on one; we really need to prevent them. So that's a focus on, whether it be formal methods, safe languages, or compilers that won't compile if there's a vulnerability, just getting it so the vulnerabilities aren't there in the first place, much more than they are today.

Jaeger: Going along this thread a bit, I'm less sanguine about removing risky or unsafe operations from our systems, but if we have unsafe operations, we need to identify how to deal with them, accounting for the overhead they incur. So, I guess for my one wish, if we could identify all unsafe operations in a particular system automatically, and then, for that system, produce the best defenses that we could within some cost budget automatically, that would be a cool thing.

Audience: We've heard a lot of great positive things about progress and advances that have come. I think we would learn more from failures. I would like to hear more from the panel about the failures. Could panelists share things that, if we went back 10 or 20 years, this is a problem we should have solved, or made a lot of progress on, but have failed to make the progress we should have?

Kapadia: In the usable security space, we have a string of articles with playful titles, like "Why Johnny can't encrypt" and the like. It's been a while since that first article was written. It seems that Johnny still can't encrypt. It still takes the average computer scientist a good deal of effort to set up secure e-mail. For instant messages as well, like Signal, presumably we are supposed to verify the key fingerprints, and I don't know how many of us do that, even those of us in this room. So, I think we are still not there yet with secure communication. At the same time, there has been some neat research in this area, so my conclusion is that this is just a very hard problem that we have underestimated as a community.

Williams: I wish we had come a lot further with security metrics. I've been with the Science of Security program since 2011, and one of the hard problems is security metrics. Being able to measure security reliably is something that I feel bad we haven't gotten better at. As I mentioned before, the NVD captures probably 5% of the vulnerabilities. We just can't operate with that. And so how do we get a good picture of the vulnerabilities in a product so that we can do more with that?

Jaeger: Twenty years ago or so, there was an article called "CCured" that reported that around 90% of the pointers in each program are not used in pointer arithmetic nor in typecasting operations, which are operations that may cause spatial and type memory errors. As a result, these pointers are safe with respect to those types of memory errors. So, I was optimistic at that time, somewhat overly so, that we would have tooling to help programmers avoid these kinds of errors and deal with remaining unsafe pointers systematically to remove a critical threat. This also relates to usable security because more usable interactive development environments that are security aware could help the programmers address these challenges. I guess I expected there would be more tools for programmers to remove and/or protect code that may be prone to memory errors than fuzz testing by this time.

Tsudik: I thought one shade of the question was, "What have you done that you wished you hadn't." If I were confessing, I would say that I wasted a lot of time on this evil thing called *group key management*. I wish I hadn't! If you find yourself confronted with this topic, stay away! If the question comes to, "What could we have done better?", I'll come to it from the educational perspective. I fail to teach people the importance of *not* designing their own security protocols from scratch. No matter how much I teach them all the subtle, abstruse, weird, bizarre errors people make in designing protocols, and the literature is full of them, they still try. A few months ago, I received an e-mail from a student I taught about eight years ago. And it was a weird e-mail because the student started to say, "Well, you know during the part of the class where you were teaching Diffie–Hellman that nobody ever uses and all these strange things that can happen, all these weird bugs you can have, for three weeks, I pretty much slept through it." Now the student is working for a company that requires him to integrate some security protocols and authentication protocols. Guess what? He wrote one from scratch! A month later, a bug was found. And then he had an epiphany. He said he went back to the notes, and now he wishes he hadn't slept through the class. So I failed. He still made the mistake. I still don't know how to teach people *not* to design things from scratch.

Kohno: Great question. I think our community, as a scientific discipline, is constantly putting pressure on itself to not be happy with the progress that we've made. Building on Apu's comments about usable security, if we rewind 20 years, we see that as a field, we were largely working on the technical elements of security and not thinking about users. If we continued to do that,

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

that would have been a failure. But we then started to think about users. Then, if we kept on thinking that all users are the same, that would have been a failure. But we're not doing that. We're realizing that we should not design for a so-called "default persona." We need to be thinking about different users and circumstances. So my perspective is that at individual times, there have been directions that, in retrospect, our community realized require more nuance and attention. Are those failures, or are those contributing to our greater growth? It's a thought-provoking question.

Reiter: I would say our inability to enforce least privilege in almost any aspect of systems is a categorical failure of our efforts.

Reiter: Final question?

Audience: One of my pandemic hobbies has been reading research articles in really far-away fields from ours—from the usual epidemiology to exercise science to cancer research to a variety of things. One of my favorite things to do there is to read meta-analyses. I was thinking about Apu's comment about novelty, and we've finally gotten to the point in security where we are accepting and publishing systematic reviews. I found myself reading these useful nice thorough meta-analyses in other fields, but I can't really picture this in computer security because we don't allow people to publish on repeated studies, with different cohorts, across too many different, but similar enough, topics to actually build up a body of work that would support these kinds of articles. Do you think that would be valuable, do you think we will get there, and is there a way to make that happen?

Kapadia: I completely agree with you; I think we need to get to the point when we accept replication and meta-analyses as important for the community, but I do worry that the average reviewer in the Big Four security conference is overly focused on technical novelty. Do we need hard problems to work on? No, we need *important* problems to work on. Replication and meta-analyses are important aspects of scientific study.

Williams: I can just make a comparison. I publish in software security, but also software engineering, and I see the same kind of discussion, although I think in software engineering, we are a little bit further along. As reviewers and program committees, we are encouraged to accept replications; replications are called for. We even get badges now, artifact badges, and there's a replication badge. You can retroactively get a replication badge if someone ever replicates your work. I know, particularly for conferences, that everyone wants big exciting presentations. If there were a track or some acknowledgment that the replications that enable meta-analyses in the future were valuable, that would be a good contribution.

Tsudik: I'd just like to concur with that. I would even go further: replications are an important confirmation of previous research results, but I still think that constructive novel research is super important. We are dead in the water without it. We will move nowhere without novel constructive research. But it should be judged against results of the same caliber. We should not be reviewing constructive novel research in the same way as we review replicated research or in the way we review attack-oriented research. These three have their place under the sun, and they should, in all these top venues. But they should be judged using different criteria.

Kohno: I haven't read as broadly as you, but the question that comes to my mind is: how do these replication studies fit into the career trajectories of the people involved? I think it would be super great if our community had that type of study. But I'm trying to figure out how such studies fit into the ecosystem of students' careers. So I would love to know more about how that works in those other communities.

Kapadia: Reacting broadly to comments made earlier today, I've heard many times, in discussions like this, that we should make it a requirement of the tenure track, and I really feel strongly against this because it's like saying, "Oh yes, the assistant professors, we'll make them do all this, and we get away scot-free." I think we need to decide what's important and make the community do it together. So, I really want to push back on the idea of requiring such work from a few specific groups.

Jaeger: I just want to say that replication is good, along the lines of what Gene said. Laurie mentioned that software engineering conferences give replication badges, as do some operating systems conferences. For example, we just got a replication badge for our OSDI paper, so we should consider doing that in security as well.

Reiter: Thanks to all! ■

Carl Landwehr is a visiting professor of electrical engineering at the University of Michigan, Ann Arbor, MI 48109-2122 USA. His research interests include cybersecurity, software engineering, and formal methods. Landwehr received a Ph.D. in computer and communications sciences from the University of Michigan. He is a Fellow of IEEE. Contact him at carl.landwehr@gmail.com.

Michael K. Reiter is a James B. Duke distinguished professor of computer science and electrical and computer engineering at Duke University, Durham, NC 27708 USA. His research interests include computer security, distributed computing, and applied cryptography. Reiter received a Ph.D. in computer science

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

from Cornell University. He is a Fellow of IEEE. Contact him at michael.reiter@duke.edu.

Laurie Williams is a distinguished university professor in the Computer Science Department of the College of Engineering at North Carolina State University, Raleigh, NC 27513 USA. Her research interests include software security, software supply chain security, and software development process. Williams received a Ph.D. in computer science from the University of Utah. She is a Fellow of IEEE. Contact her at lawilli3@ncsu.edu.

Gene Tsudik is a distinguished professor of computer science at the University of California, Irvine. His research interests do not include machine learning, blockchains/cryptocurrencies, or differential privacy. Tsudik received a Ph.D. in computer science from the University of Southern California. He's a Fellow of IEEE. Contact him at gts@ics.uci.edu.

Trent Jaeger is a professor of computer science and engineering at The Pennsylvania State University, University Park, PA 16802 USA. His research interests include operating systems security, software security, and distributed systems security. Jaeger received a Ph.D. in computer science and engineering from the University of Michigan. He is a Senior Member of IEEE. Contact him at trj1@psu.edu.

Tadayoshi Kohno is a professor in the Paul G. Allen School of Computer Science and Engineering at the University of Washington, Seattle, WA 98195 USA. His research interests include helping protect the security, privacy, and safety of users of current- and future-generation technologies. Kohno received a Ph.D. in computer science from the University of California, San Diego. He is a Fellow of IEEE. Contact him at yoshi@cs.washington.edu.

Apu Kapadia is a professor in the Department of Computer Science at Indiana University Bloomington, Bloomington, IN, 47408 USA. His research interests include privacy, usable security, photo sharing, Internet of Things and wearables, and mixed and virtual reality. Kapadia received a Ph.D. in computer science from the University of Illinois at Urbana-Champaign. He is a Member of IEEE. Contact him at kapadia@indiana.edu.