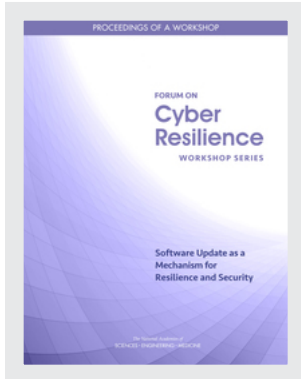


This PDF is available at <http://nap.edu/24833>

SHARE



Software Update as a Mechanism for Resilience and Security: Proceedings of a Workshop

DETAILS

92 pages | 8.5 x 11 | PAPERBACK

ISBN 978-0-309-46288-4 | DOI 10.17226/24833

CONTRIBUTORS

Anne Frances Johnson and Lynette I. Millett, Rapporteurs; Committee on Cyber Resilience Workshop Series; Forum on Cyber Resilience; Computer Science and Telecommunications Board; Division on Engineering and Physical Sciences; National Academies of Sciences, Engineering, and Medicine

GET THIS BOOK

FIND RELATED TITLES

Visit the National Academies Press at NAP.edu and login or register to get:

- Access to free PDF downloads of thousands of scientific reports
- 10% off the price of print titles
- Email or social media notifications of new titles related to your interests
- Special offers and discounts



Distribution, posting, or copying of this PDF is strictly prohibited without written permission of the National Academies Press. (Request Permission) Unless otherwise indicated, all materials in this PDF are copyrighted by the National Academy of Sciences.

Copyright © National Academy of Sciences. All rights reserved.

FORUM ON

Cyber Resilience

WORKSHOP SERIES

**Software Update as a
Mechanism for
Resilience and Security**

THE NATIONAL ACADEMIES PRESS

500 Fifth Street, NW

Washington, DC 20001

This project was supported by the National Science Foundation under award number CNS-14194917 and the National Institute of Standards and Technology under award number 60NANB16D311. Any opinions, findings, conclusions, or recommendations expressed in this publication do not necessarily reflect the views of any organization or agency that provided support for this project.

International Standard Book Number-13: 978-0-309-46288-4

International Standard Book Number-10: 0-309-46288-6

Digital Object Identifier: <https://doi.org/10.17226/24833>

Additional copies of this publication are available for sale from the National Academies Press, 500 Fifth Street, NW, Keck 360, Washington, DC 20001; (800) 624-6242 or (202) 334-3313; <http://www.nap.edu>.

Copyright 2017 by the National Academy of Sciences. All rights reserved.

Printed in the United States of America

Suggested citation: National Academies of Sciences, Engineering, and Medicine. 2017. *Software Update as a Mechanism for Resilience and Security: Proceedings of a Workshop*. Forum on Cyber Resilience Workshop Series. Washington, DC: The National Academies Press. doi: <https://doi.org/10.17226/24833>.

The National Academies of
SCIENCES • ENGINEERING • MEDICINE

Consensus Study Reports published by the National Academies of Sciences, Engineering, and Medicine document the evidence-based consensus on the study's statement of task by an authoring committee of experts. Reports typically include findings, conclusions, and recommendations based on information gathered by the committee and the committee's deliberations. Each report has been subjected to a rigorous and independent peer-review process and it represents the position of the National Academies on the statement of task.

Proceedings published by the National Academies of Sciences, Engineering, and Medicine chronicle the presentations and discussions at a workshop, symposium, or other event convened by the National Academies. The statements and opinions contained in proceedings are those of the participants and are not endorsed by other participants, the planning committee, or the National Academies.

For information about other products and activities of the National Academies, please visit www.nationalacademies.org/about/whatwedo.

FORUM ON
**Cyber
Resilience**
WORKSHOP SERIES

**Software Update as a Mechanism
for Resilience and Security**

PROCEEDINGS OF A WORKSHOP

Anne Frances Johnson and Lynette I. Millett, *Rapporteurs*

The National Academies of
SCIENCES • ENGINEERING • MEDICINE

THE NATIONAL ACADEMIES PRESS
Washington, DC
www.nap.edu

The National Academies of
SCIENCES • ENGINEERING • MEDICINE

The **National Academy of Sciences** was established in 1863 by an Act of Congress, signed by President Lincoln, as a private, nongovernmental institution to advise the nation on issues related to science and technology. Members are elected by their peers for outstanding contributions to research. Dr. Marcia McNutt is president.

The **National Academy of Engineering** was established in 1964 under the charter of the National Academy of Sciences to bring the practices of engineering to advising the nation. Members are elected by their peers for extraordinary contributions to engineering. Dr. C. D. Mote, Jr., is president.

The **National Academy of Medicine** (formerly the Institute of Medicine) was established in 1970 under the charter of the National Academy of Sciences to advise the nation on medical and health issues. Members are elected by their peers for distinguished contributions to medicine and health. Dr. Victor J. Dzau is president.

The three Academies work together as the **National Academies of Sciences, Engineering, and Medicine** to provide independent, objective analysis and advice to the nation and conduct other activities to solve complex problems and inform public policy decisions. The National Academies also encourage education and research, recognize outstanding contributions to knowledge, and increase public understanding in matters of science, engineering, and medicine.

Learn more about the National Academies of Sciences, Engineering, and Medicine at www.nationalacademies.org.

COMMITTEE ON CYBER RESILIENCE WORKSHOP SERIES

FRED B. SCHNEIDER, NAE,¹ Cornell University, *Chair*
ANITA ALLEN, NAM,² University of Pennsylvania
ERIC GROSSE, Independent Consultant
BUTLER W. LAMPSON, NAS³/NAE, Microsoft Corporation
SUSAN LANDAU, Worcester Polytechnic Institute

Staff

LYNETTE I. MILLETT, Director, Forum on Cyber Resilience
EMILY GRUMBLING, Program Officer
SHENAE BRADLEY, Senior Program Assistant

¹National Academy of Engineering.

²National Academy of Medicine.

³National Academy of Sciences.

FORUM ON CYBER RESILIENCE

FRED B. SCHNEIDER, NAE, Cornell University, *Chair*
ANITA ALLEN, NAM, University of Pennsylvania
BOB BLAKLEY, CitiGroup
FRED CATE, Indiana University
DAVID D. CLARK, NAE, Massachusetts Institute of Technology
RICHARD DANZIG, Johns Hopkins University Applied Physics Laboratory
ERIC GROSSE, Independent Consultant
DAVID HOFFMAN, Intel Corporation
PAUL KOCHER, Cryptography Research, Inc.
TADAYOSHI KOHNO, University of Washington
BUTLER W. LAMPSON, NAS/NAE, Microsoft Corporation
SUSAN LANDAU, Worcester Polytechnic Institute
STEVEN B. LIPNER, NAE, Independent Consultant
DEIRDRE K. MULLIGAN, University of California, Berkeley
TONY SAGER, Center for Internet Security
WILLIAM SANDERS, University of Illinois, Urbana-Champaign
PETER SWIRE, Georgia Institute of Technology
DAVID VLADECK, Georgetown University
MARY ELLEN ZURKO, Independent Cybersecurity Consultant

Ex Officio

DONNA F. DODSON, National Institute for Standards and Technology
JAMES KUROSE, National Science Foundation
WILLIAM B. MARTIN, National Security Agency

Staff

LYNETTE I. MILLETT, Director
EMILY GRUMBLING, Program Officer
KATIRIA ORTIZ, Research Associate
SHENAE BRADLEY, Administrative Assistant

For more information about the Forum, see its website at <http://www.cyber-forum.org>,
or e-mail the Forum at cyberforum@nas.edu.

COMPUTER SCIENCE AND TELECOMMUNICATIONS BOARD

FARNAM JAHANIAN, Carnegie Mellon University, *CSTB Chair*
LUIZ ANDRÉ BARROSO, Google, Inc.
STEVEN M. BELLOVIN, NAE, Columbia University
ROBERT F. BRAMMER, Brammer Technology, LLC
DAVID CULLER, NAE, University of California, Berkeley
EDWARD FRANK, Cloud Parity, Inc.
LAURA HAAS, NAE, IBM Corporation
MARK HOROWITZ, NAE, Stanford University
ERIC HORVITZ, NAE, Microsoft Corporation
VIJAY KUMAR, NAE, University of Pennsylvania
BETH MYNATT, Georgia Institute of Technology
CRAIG PARTRIDGE, Raytheon BBN Technologies
DANIELA RUS, NAE, Massachusetts Institute of Technology
FRED B. SCHNEIDER, NAE, Cornell University
MARGO SELTZER, Harvard University
JOHN STANKOVIC, University of Virginia
MOSHE VARDI, NAS/NAE, Rice
KATHERINE YELICK, NAE, University of California, Berkeley

Staff

JON EISENBERG, Director
LYNETTE I. MILLETT, Associate Director

VIRGINIA BACON TALATI, Program Officer
SHENAE BRADLEY, Administrative Assistant
JANEL DEAR, Senior Program Assistant
EMILY GRUMBLING, Program Officer
RENEE HAWKINS, Financial and Administrative Manager
KATIRIA ORTIZ, Research Associate

For more information on CSTB, see its website at <http://www.cstb.org>, write to CSTB, National Research Council, 500 Fifth Street, NW, Washington, DC 20001, call (202) 334-2605, or e-mail the CSTB at cstb@nas.edu.

ACKNOWLEDGMENT OF REVIEWERS

This Proceedings of a Workshop was reviewed in draft form by individuals chosen for their diverse perspectives and technical expertise. The purpose of this independent review is to provide candid and critical comments that will assist the National Academies of Sciences, Engineering, and Medicine in making each published proceedings as sound as possible and to ensure that it meets the institutional standards for quality, objectivity, evidence, and responsiveness to the charge. The review comments and draft manuscript remain confidential to protect the integrity of the process.

We thank the following individuals for their review of this proceedings:

Edward Felten, Princeton University,

Kevin Fu, University of Michigan,

William Sanders, University of Illinois, Urbana, Champaign, and

Mary Ellen Zurko, Independent Cybersecurity Consultant.

Although the reviewers listed above provided many constructive comments and suggestions, they were not asked to endorse the content of the proceedings nor did they see the final draft before its release. The review of this proceedings was overseen by Steven M. Bellovin, NAE,¹ Columbia University. He was responsible for making certain that an independent examination of this proceedings was carried out in accordance with standards of the National Academies and that all review comments were carefully considered. Responsibility for the final content rests entirely with the rapporteurs and the National Academies.

¹National Academy of Engineering.

Preface

The Forum on Cyber Resilience—a roundtable established in 2015 by the National Academies of Sciences, Engineering, and Medicine—facilitates and enhances the exchange of ideas among scientists, practitioners, and policy makers who are concerned with urgent and important issues related to the resilience of the nation’s computing and communications systems, including the Internet, other critical infrastructures, and commercial systems. Forum activities help inform and engage a broad range of stakeholders around issues involving technology and policy related to cyber resilience, cybersecurity, privacy, and related emerging issues. A key role for the Forum is to surface and explore topics that advance the national conversation.

In 2016, the Forum held a workshop exploring the topic of cryptographic agility and its implications for security and resilience. That workshop was summarized in *Cryptographic Agility and Interoperability: Proceedings of a Workshop*. Discussions during and subsequent to that workshop made clear that software update is an important mechanism by which security changes and improvements are made in software. Preliminary conversations among members revealed that this seemingly simple concept encompasses a wide variety of practices, mechanisms, policies, and technologies.

To explore the landscape further, the Forum decided to host a workshop. The workshop featured invited speakers from government, the private sector, and academia. This proceedings summarizes presentations made by invited speakers and other remarks by workshop participants. In keeping with the workshop’s exploratory

purpose, the proceedings does not contain findings or recommendations. Nor, in keeping with National Academies guidelines for workshop proceedings, does it necessarily report consensus views of the workshop participants or organizing committee. A planning group appointed to oversee all Forum workshops was limited to planning the workshop, and this workshop proceedings was prepared by the workshop rapporteurs and Forum staff as a factual summary of what occurred at the workshop. The document draws on prepared remarks of workshop speakers, comments made by workshop participants, and ensuing discussions.

The introductory section summarizes the introduction to the workshop and reproduces background material provided to all participants. Chapters 1 through 10 summarize speaker presentations. Chapter 11 describes the content of the final plenary discussion, highlighting some of the broader themes that emerged throughout the workshop. An Afterword offers additional commentary on the policy and technical aspects of the software update challenge. The agenda of the workshop is in Appendix A. Short biosketches of the planning group and speakers appear in Appendixes B and C, respectively.

My sincere thanks to the planning group, Forum members, and staff who helped organize the workshop, as well as to the invited speakers for their thoughtful remarks and enthusiastic participation in the discussions that ensued. Writing support was provided by Anne Frances Johnson and Kathleen Pierce, Creative Science Writing. Special thanks to Eric Grosse for his contributions to the Afterword. I also extend our appreciation to the National Science Foundation, the National Security Agency, the Special Cyber Operations Research and Engineering Working Group, and the National Institute of Standards and Technology for their support and encouragement of Forum activities.

Fred B. Schneider, *Chair*
Forum on Cyber Resilience

Contents

WORKSHOP INTRODUCTION 1

- 1 POLICY CONSIDERATIONS:
THE INTERSECTION OF PUBLIC VALUES AND PRIVATE INFRASTRUCTURE 7**
 - Policy Considerations for Creating an Update Infrastructure 8
 - Key Challenges Involved in Creating an Update Infrastructure 9
 - Guiding Principles for Cybersecurity 10
 - Key Players and the Need for Dialogue 10
 - The Challenge of Keeping Up With the Pace of Technological Advancement 11
 - Other Considerations 12
- 2 TECHNICAL CONSIDERATIONS FOR SECURE SOFTWARE UPDATES 14**
 - Vulnerabilities in Older Systems 14
 - Special Concerns for Embedded Medical Devices 15
 - Legal Mechanisms for Disclosure and Updates 17
 - Other Potential Solutions to Address Vulnerability 18
- 3 MICROSOFT’S APPROACH TO SOFTWARE UPDATES 19**
 - Today’s Threat Landscape 19
 - A New Model for Building and Delivering Updates 20
 - Quality Updates Versus Feature Updates 21
 - The Struggle with Longevity 22
 - Behind the Scenes on “Update Tuesday” 22
- 4 UPDATE ISSUES FOR OPEN-SOURCE SOFTWARE 24**
 - Inherent Challenges to Securing Open-Source Software 25
 - Dealing with the Fear of Destabilization 27
 - Open Source in the Internet of Things 28
- 5 CISCO’S APPROACH TO SOFTWARE UPDATES 30**
 - The Attack Environment 30
 - Cisco’s Approach to Addressing Vulnerabilities 31
 - Understanding User Perspectives 33

6	ENSURING ROBUST FIRMWARE UPDATES	36
	Unique Attributes of SEL’s Market Space	36
	SEL’s Approach to Security	37
	Managing Firmware and Software Updates	38
	Other Mechanisms for Maintaining Secure Systems	39
	The “Aurora” Vulnerability	40
	Fostering Development Expertise and Broader Lessons for the Software Industry	40
7	UPDATES IN THE CONSUMER ELECTRONICS INDUSTRY	42
	Challenges from Using Off-the-Shelf Components	42
	Business-Side Considerations	44
	Additional Technical Complexities	45
8	SOFTWARE UPDATES IN AUTOMOTIVE ELECTRONIC CONTROL UNITS	47
	History of Electronic Control Units	47
	The Challenge of Limited Networking Capabilities	49
	Other Challenges to Software Delivery	49
	Dealing with Legacy Products	50
	Security Concerns	51
9	THE NIST PERSPECTIVE ON SOFTWARE UPDATES	53
	Key Challenges	53
	Solutions Being Pursued	55
10	PROTECTING CONSUMERS FROM SOFTWARE UPDATE RISKS	59
	Working with Businesses	60
	Educating Consumers	61
	Toward a New Approach	62
11	DISCUSSION	63
	AFTERWORD	66
	APPENDIXES	71
	A Workshop Agenda and Participants List	73
	B Steering Committee Biographies	75
	C Speaker Biographies	78

Workshop Introduction

The Forum on Cyber Resilience of the National Academies of Sciences, Engineering, and Medicine hosted a Workshop on Software Update at its Winter 2017 meeting on February 6, 2017, in Washington, D.C.

The workshop featured experts representing various industries, research laboratories, and government agencies. Speakers discussed experiences and challenges related to a range of issues surrounding software updates, reflecting on the historical evolution, exploring today's tools and gaps, and considering future concerns and opportunities, especially as related to software update as a tool for improved cybersecurity and resilience. Participants identified key questions, suggested ideas, and closely examined uncertainties involved in improving software updates today and tomorrow.

The meeting was open to the public. This proceedings was created from the presenters' slides and a full transcript of the workshop; it is intended to serve as a public record of the workshop presentations and discussions.

OPENING REMARKS

Fred B. Schneider, Ph.D., the Samuel B. Eckert Professor of Computer Science at Cornell University, member of the National Academy of Engineering, and Forum chair, opened

the meeting with a brief overview of the National Academies' Forum on Cyber Resilience. He then introduced the workshop's topic with a metaphor: recycling. Soda used to come only in large glass bottles, but eventually cheaper, more convenient metal cans and plastic bottles were introduced. The new packagings had an unfortunate side effect: increased litter. Rebate and recycling programs, as well as laws criminalizing litter, evolved to address this problem, although, Schneider noted, "It took a while to put all the right mechanisms in place."

Like the landscape of soda bottles littering our streets before recycling, we now live in a world that is fast accumulating the remnants of "disposable" software, Schneider said. People constantly replace their devices, and software vulnerabilities are constantly discovered and fixed. But a great deal of vulnerable software remains in use. This software is desperately in need of updates that, for a variety of reasons, it's not getting. This software and these vulnerable systems are a new form of litter.

Software update methods allow us to "cope with the reality that there are going to be vulnerabilities," Schneider said, by either patching or replacing vulnerable software. As software continues to proliferate and our lives become ever more dependent on it, the consequences of vulnerabilities grow. Deploying updates securely is also an increasingly complex technical challenge. No longer are updates just one aspect of a developer's responsibilities, but they have become a central aspect of the industry.

Although not updating software is clearly problematic for security, deploying software updates also comes with risks. If an update goes wrong, the device could break or provide an opening for attackers. Any sort of centralized or automated distribution of updates becomes an attractive target for attackers. Even sending physical disks with updates through the mail, a system previously used for updating flight-control software on commercial airplanes, is not necessarily secure. Updates also, as a side effect, advertise that previous versions of the software are vulnerable to attack, even pointing attackers to specific vulnerabilities that can be exploited in non-updated devices.

Recertification of software raises other issues. After performing an update, a Naval warship might require 6 months in port to recertify all of its systems, Schneider noted. The lengthy recertification process also required for airplanes and medical devices means that automated updates (such as the regularly scheduled updates from Microsoft, known as "Patch Tuesday") would be untenable in those situations.

Economics is also a concern, underscored by the proliferation of ever-cheaper mobile devices and the applications people buy for them. At these low price points, or "hit-and-run sales," manufacturers may assume they are not entering into a long-term relationship with consumers. Such manufacturers might not feel obligated to provide

software updates to provide the level of security that manufacturers of other types of products might provide. Users' rights are also a factor: Deploying software updates can give manufacturers broad access to a user's device, which raises potential privacy issues and creates antitrust minefields, Schneider said.

In short, cleaning up the "discarded soda bottles" of our Information Technology Ecosystem not only involves complex technical challenges, but economic, political, and social consequences, as well. The Forum provided a venue to dive into these issues, tease out nuances, and expose hidden assumptions surrounding the software updates.

Background and Context

The following information was provided to workshop speakers and attendees to offer context on software update issues and to provide a structure for the workshop and its intended purposes.

In October 2016, botnet malware known as Mirai was used to launch a disruptive denial of service attack on a Domain Name System (DNS) service provider. That disruption resulted in hundreds of popular websites and services being inaccessible for several hours. Mirai exploited the fact that numerous Internet of Things (IoT) devices, such as cameras and digital video recorders, still had vendor default usernames and passwords (and in some cases had unchangeable firmware passwords accessible via telnet and Secure Shell [SSH]).¹ This attack spotlighted several security challenges related to the mass proliferation of IoT devices; perhaps the most significant is that the software on many IoT devices is difficult or impossible to update.

Software updates have long been a mechanism through which security improvements in systems and devices are made. It is timely, then, to consider the value and prospects for software update as a security mechanism in the future. In what sorts of systems are software updates difficult? What makes it difficult? What incentives can be put in place to ensure that needed updates can be done?

Software-based systems undergird every type of critical infrastructure. Software enables business, manufacturing, transportation, and health care. Software is embedded in homes and offices, in safety-critical contexts, in entertainment contexts, and, increasingly, in wearable devices on human bodies. As bugs and vulnerabilities are discovered, as circumstances and requirements change, and as new features are developed, software updates are made available.

Software updates encompass a range of activities. Consumers are familiar with updating operating systems and individual applications. Enterprise information technology departments typically develop processes for how software updates will be implemented for their users. Providers of cloud services and Web-based applications update their products and systems regularly.

The question of how, when, and why to update software is not new. Before personal computers could be easily networked, updates had to be applied using physical media (“sneakerware”). Today, updates to some mobile phones, IoT devices, and even cars can arrive automatically over wireless connections directly from the software manufacturer. Many enterprises push automatic updates to users’ machines and devices. Consumers are sometimes given the option of having their personal devices update themselves automatically, whenever new software is available. In some cases, manufacturers will design systems so that updates are done automatically without allowing the consumer the choice.

Software updates are used to improve security and other important attributes as well as to provide upgrades or changes to software features. As security has become a priority over the last two decades, software update has become a key mechanism by which improvements in security in systems that are already deployed can be made. The development of mass automatic updates allows for fast application of patches when security problems in a given system or deployment need to be addressed.

Software updates as a security mechanism comes with other challenges and risks. As most users of information systems know, an update that goes wrong can have downsides ranging from minor incompatibilities with existing applications to loss of time spent reconfiguring systems or even destruction of data. Moreover, not all systems can be readily updated. Some IoT devices offer no obvious update capability—they may lack regular Internet connectivity, or the device may not have the computing resources to validate and install updates, or updates may no longer be available from the company that sold them. Some systems, such as medical devices, might require significant verification and validation after updating; and some, such as those on some Naval ships, cannot be updated while the ship is actively deployed. In such a case, what other options are available? When software update is possible, how should the update itself be verified and authenticated? When and how do we believe the update channel is secure enough that the benefits outweigh the risks? How should devices reject obsolete revisions? What security procedures should be employed when updates are issued?

Another challenge is that software updates for security reasons tend to happen under time pressure. At the same time, it may be important not to release an update until there's a way to be sure to get it to everyone who needs it. This is because anyone who sees the update could potentially reverse engineer it to find and exploit the vulnerability that is being corrected. Ensuring that everyone who needs the update gets it can be a particular challenge in scenarios where the software developer does not or cannot themselves ensure that the update is sent to those who need it. Consider the following scenarios:

- A software supplier releases the update to the device vendor; the device vendor has to “qualify” it before releasing it to customers. If device vendors are not diligent about getting the update out, some customers are left vulnerable, and reverse engineering by adversaries is possible based on customers (or device vendors) who do get the update.
- Suppose a widely used open-source component is found to need a security update. If many organizations use the component, how can an update process be managed so that the update is developed and qualified for all of the organizations that rely on it without the knowledge of the vulnerability leaking and being exploited before the update is distributed?
- Similar to the above: A developer ships software that depends on certain components, and those components require a software update for security reasons. How can the developer find everyone who needs to install this update?
- Or, consider the challenge of updating products when the original developer is no longer available, or the product is out of support.

This workshop will bring together software experts, security experts, practitioners, and researchers to explore how software updates are accomplished and implications for security and resilience of systems that depend on software. Questions to explore include

General: How have “software updates” changed over time and what are prospects for the future? How do different technical sectors manage software updates? For what sorts of systems are software updates likely to be an appropriate method of deploying improvements in resilience and security? What are the advantages and disadvantages

of distinguishing and separating feature updates from security updates in software updates? What policy options are there to improve the practice and outcomes of software updates (e.g., what are the pros and cons of enforcing automatic updates)?

Security Risks: Do current practices related to software encourage vendors to ship buggy implementations on the assumption that bugs will be fixed later? If insecure-but-updatable products are the only ones available in the market, will they be used in high-value environments that will then fall to 0-day attacks?

End-of-Life: Should vendors be required to provide security updates? If so, for how long? When the support period ends, should source code and signing tools be made open-source to allow third-party updates? Who is liable when products fail (perhaps in unsafe ways) after their support period ends? What are the economic and environmental impacts when products are discarded because essential updates are no longer available?

User Rights: What rights and obligations should users have? For example, are disclosure of limitations and/or acceptance by end users required (or even practical given the difficulty of understanding legalese, lack of alternatives, and number of devices that users own)? Should users be allowed to reject safety-critical updates (e.g., that may facilitate denial of service attacks against others on the network)? Should patches to address security or safety vulnerabilities be treated differently from updates that modify functionality?

Privacy, Conflicting Interests: Should vendors be permitted to leverage security update mechanisms as a way to achieve other business objectives, such as obtaining data from users (e.g., to train vendor artificial intelligence systems), obtaining users' consent to vendor-chosen contractual obligations, modifying device capabilities in ways that may be undesirable to the user, or pressing users to install software products unrelated to the one being updated? What challenges arise when multiple parties (e.g., telecommunications carriers and mobile operating system developers) are involved in software updates and what options are there for dealing with them?

Costs and Accounting: What are the long-term costs of maintaining software updates for products, and how should these costs be funded? When companies sell a product, do they need to take a charge for the future costs of producing updates for the product?

¹ *Krebs on Security*, 2016, "Hacked Cameras, DVRs Powered Today's Massive Internet Outage," October 21, blog, <https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage>.

Policy Considerations

The Intersection of Public Values and Private Infrastructure

Deirdre Mulligan, University of California, Berkeley

Deirdre Mulligan, associate professor at the School of Information at the University of California, Berkeley, framed the challenge of software updates with a broad overview of the tangle of issues at the intersection of public values and private infrastructure.

Mulligan opened with her proposal, first posited along with co-author Fred Schneider in 2011, for a new doctrine of cybersecurity—an approach that considers cybersecurity as a public good and stresses greater attention to security issues during software development and managing vulnerabilities after deployment, as opposed to chasing hackers or making software invulnerable.¹ No matter how attentive software designers are to security issues, she said, new threats and insecurities will inevitably emerge. Figuring out how to deal with that ongoing insecurity deserves some portion of our attention from both a technical and policy standpoint.

¹D.K. Mulligan and F.B. Schneider, 2011, Doctrine for cybersecurity, *Daedalus* 140(4):70-92, doi:10.1162/daed_a_00116.

POLICY CONSIDERATIONS FOR CREATING AN UPDATE INFRASTRUCTURE

While current policies that recognize and reduce vulnerabilities are beneficial, Mulligan suggested that there should be incentives, or even coercive measures, to induce companies to produce and maintain more secure software in order to better defend the rights of individuals to be protected against threats.

Mulligan identified three main aspects of a potential policy infrastructure for software updates. The first is a need for a shared definition of the term “cybersecurity.” For example, whose “security,” exactly, does it refer to? Also, what would justify the

use of a security update channel? Second, she noted that cybersecurity is embedded within an ecosystem of related issues such as individual privacy, consumer protection, industry competition, and intellectual property. Finally, software updates encompass the tension between society’s collective interests, on one hand, and the rights and interests of individuals and corporations on the other.

In Mulligan’s view, situations like the Mirai attack (described briefly on page 4) demonstrate why policy intervention is warranted. “When we have externalities—when my choices are impacting you in ways that are negative and in ways that are really difficult for you to address—that is a justifiable reason for looking to public policy to intervene, to create

incentives, or to punish people for failure,” she asserted. A public policy intervention, in this case, could potentially create incentives, for effective software updates or even punishments for failure, while still being mindful of a balance between the rights of individuals and the needs of a society.

An update infrastructure that allows the monitoring—and addressing—of threats and vulnerabilities would require extensive information, Mulligan noted. Monitoring and surveillance would be necessary to identify specific threats and contribute to a shared knowledge of evolving risks or attacks. This requires knowledge about devices, such as their properties, the applications used on them, and their network connections. An update infrastructure, she noted, would also require the ability to update a device “in the wild,” and channels used to convey such updates must remain secure themselves.



Channels used
to convey
updates must
remain secure
themselves.

KEY CHALLENGES INVOLVED IN CREATING AN UPDATE INFRASTRUCTURE

Developing an infrastructure for updates raises several key challenges from the perspective of both businesses and consumers, and these challenges are especially acute as we move into the Internet of Things (IoT) space, said Mulligan.

Making accurate vulnerability assessments of software and fixing associated vulnerabilities could potentially require the collection of private data from consumers' devices, where there is an expectation of privacy and control. As a result, it is important to consider consumers' expectations and the privacy implications of a software update infrastructure, Mulligan said.

An update infrastructure policy would also, of course, affect companies. There remain many unanswered questions. For example, should the initial seller have exclusive power to update device security or functionality, or should other companies offering similar, potentially more secure, updates be allowed to compete in this space? Stability is another concern, because security updates could destabilize a device or service, frustrating the consumer or the manufacturer.

For the IoT space, there is a need to carefully consider how to use traditional update channels, which have long been used to update other kinds of computers, in the context of IoT devices. Should a new channel exclusively for security updates be created, and could it truly be secured? As an example, Mulligan pointed to Tesla, which sends updates to its cars via an over-the-air update channel. One recent update included a new feature, Summon, which enables owners within a certain distance to press a button and "summon" the car to them. The initial software had safety issues that the company later corrected, but that update channel could have potentially created a huge risk to others near the car when it is "summoned,"² Mulligan noted.

It is also possible that update channels could be abused to *downgrade* product functionality, which could make consumers unhappy and mistrustful of the update channel altogether. Update channels highlight the tensions between the market, which may want to use the channel as an avenue to increase sales, and consumer advocates, who want to make sure it is used to deliver necessary security improvements.

Defining what, exactly, is a security upgrade is another nettlesome question, as is the question of *who* gets to define cybersecurity. Mulligan cited the example of Sony's use of DRM-protected compact discs (CDs) to prevent theft of its intellectual property; while the CDs may have advanced security in the eyes of the company, they also provided

²J. Fisher, 2016, "Tesla to Fix Self-Parking Feature After Consumer Reports Raises Safety Concern," *Consumer Reports*, February 10, <http://www.consumerreports.org/car-safety/tesla-fixes-self-parking-feature-after-consumer-reports-raises-safety-concern>.

the company with backdoor access to the consumer's machine, arguably not supporting security in the eyes of the consumer. Gatekeeping is also an issue: Would individual companies control update channels exclusively, or could other companies use them as well?

Consumers would have their own questions about update capabilities, such as whether they are mandatory or optional. If they are mandatory, consumers might have to reveal private data to receive the security benefit. Yet if the security channel isn't properly restricted, consumers could be exposed to identity theft in addition to unwanted downgrades or modifications.

IoT products in particular have poorly defined security requirements and support timelines, said Mulligan. What, if any, obligations do manufacturers have for maintaining or updating their security? Given that many of these devices are so low cost as to be considered almost disposable, for how long can a consumer expect them to remain secure? If IoT devices aren't upgraded but are still in use, what economic costs does that impose and on whom?

GUIDING PRINCIPLES FOR CYBERSECURITY

All these questions and concerns led Mulligan and Schneider to their guiding principles for cybersecurity. First, they posit that cybersecurity is a public good, and thus private or individual choices could have negative consequences for the public as a whole. Second, cybersecurity is a political construct whose goals and means must be clearly defined and generally agreed on through a series of conversations. The variety of domains where cybersecurity is important and software updates take place means that solutions may need to be adjusted at times. Finally, if cybersecurity is defined as a public good to be protected, it follows that it has to be in the forefront of the design stage of software and update channels instead of an afterthought for others to deal with.

KEY PLAYERS AND THE NEED FOR DIALOGUE

Conversations about the definition and governance of software updates could involve a range of players, Mulligan said. The Federal Trade Commission has a long history of protecting consumers. The National Telecommunications and Information Administration is beginning to discuss these issues as they relate to IoT devices. The National Highway Traffic Safety Administration has also started discussing these questions. Mulligan expressed her hope that agencies increase their focus on individual privacy, especially in

the context of automotive over-the-air updates. These and other existing institutions can be leveraged to take on this task.

The bigger obstacle, to Mulligan, will be convincing industry to more effectively integrate cybersecurity protections into their product design objectives. The technical community has certainly embraced security as a design issue, and there have also been privacy improvements, but Mulligan noted that engineers by and large do not feel they have the expertise to make privacy decisions when creating software.

Building software with security updates in mind requires genuine dialogue between technical experts and policy experts. Both groups might at times be out of their depth, but effectively addressing issues around consumer protection, industry competition, and software engineering demands a high level of coordination and commitment on both sides: Bringing both technical and policy expertise to the design process is an important piece of the puzzle to support true cybersecurity.

Such a strategy also requires an educational commitment from universities—a commitment to discussing the legal and ethical consequences of software updates when teaching software design, Mulligan argued. Workplaces also need to prioritize such thinking so that engineers know that policies are in place to ensure that cybersecurity is addressed at every stage of design. Such policies should rely on an interdisciplinary approach to solving complex problems like software updates, Mulligan concluded.

THE CHALLENGE OF KEEPING UP WITH THE PACE OF TECHNOLOGICAL ADVANCEMENT

Richard Danzig, Johns Hopkins University Applied Physics Laboratory, asked Mulligan to address the speed at which technology changes and proliferates. Given that this is a dynamic and not a static system, big problems are generated quickly. He likened the situation to the evolution of cars: As cars and their use changed over the decades and the challenges increased in scale, policy responses (licenses, roads, policing, and so forth) evolved in response. He wondered if public policy is destined to lag behind technological innovation, and, further, the degree to which public policy regulations might actually stifle innovation. He also raised the additional concern that policy makers may inevitably lag behind the technical reality in terms of their understanding of the issues.

Building on the example of cars, Mulligan noted how the automotive industry deals with recalls and other safety risks. Historically, car makers have relied on owners to have their vehicles serviced, and the turnout is usually below what car makers hope. In this respect, a technical breakthrough like Tesla's over-the-air channel theoretically improves vehicle safety by sending updates in a timely way. Getting the policy part

right, along with the technical part, could lead to faster, safer patches, a “potential win-win,” she said.

But while most automotive product recalls have to do with reducing the risk to passengers, there is another security risk: Software vulnerabilities that could turn a car into a weapon, such as the vulnerabilities discovered in Jeep Cherokees that allowed remote hacking of the entire system, engine included, which makes these issues all the more urgent.³ As the potential dangers increase, Mulligan said, cybersecurity becomes a necessary public good. This means requiring manufacturers to update car software automatically, instead of putting the onus on consumers. The risk now is not one of mere malfunction but of actual malfeasance—cars could be leveraged in real time to create an attack.

Regulations may not always be able to keep up with technological innovations, Mulligan acknowledged, but she said that right now, the marketplace is not keeping up either. The development of standards and definitions will make it easier to design, develop, and deploy secure products and patches. Regulation, in this case, can happen alongside the speed of technological change as long as both camps are committed to learning from the past, fixing current problems, and addressing future challenges. The policy and technology communities can collaborate on the values and issues—public, private, individual, corporate, and legal—and on finding the right balance among them. Furthermore, while these conversations might be easier to have in private settings, they need to happen in public, Mulligan said, because cybersecurity is a public good.

OTHER CONSIDERATIONS

Several participants raised nuances of the business environment, particularly for IoT devices, that warrant consideration in the policy space.

Tadayoshi Kohno, University of Washington, noted the proliferation of companies launching products with funding from Kickstarter.com or similar services. After producing an IoT device, for example, and attracting perhaps hundreds of thousands of users, such companies can sometimes go out of business quickly. Where does that leave customers, who are using devices with software that is never going to be maintained or updated, he asked.

Mulligan shared two ideas: Devices could come with a kill switch that is activated once the software for the device is no longer being maintained (although depending on

³A. Greenberg, 2015, “Hackers Remotely Kill a Jeep on the Highway—With Me in It,” *Wired*, July 21, doi:<https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway>.

the device, this might pose safety concerns), or, the software could become open source after the manufacturer ceases supporting it, so that others could take on the security support themselves.

Drilling deeper into the example of the hacked Jeep Mulligan raised in her presentation, one participant asked what policies could protect against this type of



New capabilities
of IoT devices
may require new
certification and
testing methods.

situation, without infringing on other areas. Mulligan noted an important distinction between the Jeep case, which she believes was a vulnerability that was not intentional on the part of the manufacturer—a mistake, and cases such as the Volkswagen test mode code, which appears to have been a deliberate decision to deceive regulators.

A policy framework would need to account for both types of cases, Mulligan said. One lesson that applies across the board is that the agencies responsible for standards and compliance need increased technical expertise, she suggested. Mulligan noted that there has been progress in this area. The Federal Automated Vehicle Policy focuses on increasing technical expertise within government and also creates an external advisory

board made up of industry and academic experts to enable the agency to better understand new trends, research, and potential threats.⁴

These examples also raise the prospect that the new capabilities of cars and other IoT devices may require new certification and testing methods, Mulligan said. This might look similar to the level of regulation used by the U.S. Food and Drug Administration. While she noted that this might not be appropriate for all areas of IoT, “When we’re talking about large pieces of metal hurtling around at really fast speeds, it’s something we should at least be considering,” she said.

The discussion wrapped up with a question from another participant regarding the potential policy implications of the distinction between hardware and software products that reside on a customer’s device, in which the user often has some control over software updates, versus cloud-based services, for which the user typically has no choice as to what version they are using. Mulligan noted that consumer protection agencies have historically focused on products that are sold directly to consumers—for example, stepping in if the company’s behavior seems deceptive or egregious—but the issue of cloud-based software or services will demand closer scrutiny in the future.

⁴U.S. Department of Transportation, 2017, “Federal Automated Vehicles Policy,” <https://www.transportation.gov/AV>, updated April 13, 2017.

2

Technical Considerations for Secure Software Updates

Kevin Fu, University of Michigan

Kevin Fu is an associate professor of computer science at the University of Michigan and co-founder of Virta Labs, a health-care cybersecurity company serving health-care providers and medical device manufacturers. He explored the technical aspects of software updates, discussing the evolution of updates over time before delving into contemporary problems, with a particular focus on the area of embedded medical devices.

VULNERABILITIES IN OLDER SYSTEMS

As a graduate student 17 years ago, Fu focused on creating secure, scalable, high-performance software updates. At that time, software updates were made via RPM Package Managers (RPMs)—software packages downloaded from websites. Each package was individually signed, which meant that the updates had been authenticated and, thus, could be assumed to be trustworthy. “The beauty was you could download it and not have to worry about the package being secure, because it was signed,” Fu reflected.

However, although each package may have been signed, the overall system was not signed, which could lead to problems. One such problem was what Fu referred to

as the “freshness factor,” in which users could be tricked into installing old updates with known vulnerabilities.

Turning to antivirus updates as an example, Fu noted that today, it is common for companies and universities to require that users regularly update antivirus DAT files to detect new viruses. In an experiment 11 years ago, Fu’s students demonstrated how easy it is to establish a root shell on a computer using an antivirus software update; the experiment revealed that McAfee, the popular antivirus software company, was not checking the cryptographic signatures on its updates. In essence, McAfee’s update channel, which was meant to increase software security, actually created new risks.

Unfortunately, many software products still do not use proper authentication, leaving this channel open to exploitation, Fu said. Antivirus software can cause other problems as well; Fu described an instance when the system at a Rhode Island hospital accidentally misclassified a critical Windows DLL as malicious, and the hospital’s admission systems ground to a halt, forcing the hospital to stop admitting patients except for those with gunshot wounds.

SPECIAL CONCERNS FOR EMBEDDED MEDICAL DEVICES


Embedded medical devices raise particular—and often overlooked—concerns, Fu explained. Before wireless updates to pacemakers, changing a device’s settings required inserting a needle into the pacemaker through a wearer’s armpit and manually adjusting a dial. Now, software is updated over the air, which is far easier for wearers, although this creates new types of vulnerabilities.

When medical device makers expressed their belief, at a Food and Drug Administration (FDA) workshop, that malware was not a concern for an automated external defibrillator being developed, Fu recruited his colleagues Dawn Song and Steve Hanna to prove them wrong. The team created a custom firmware update that included malware that could spread from a hospital computer, onto the defibrillator, and back onto more hospital computers, revealing that there appeared to be no authentication at all on the firmware going into the defibrillator.

Fu also related the story of a ventilator company that, after an FDA recall, instructed customers to download a software update from its website. Although initially encouraged that the company was providing an update, when he tried to download it himself, he was presented with a malware warning by his Web browser. Digging deeper, Fu discovered that hackers had laced the update with drive-by downloads thanks to an opening left by an outdated version of Microsoft IIS and unmaintained server scripts, leading the Google Safe Web browsing service to flag the software

download as suspicious. Fu was alarmed to learn that he was the first to report the problem, despite the update, in theory, having been downloaded by many actual users of the software.

Despite these examples, manufacturers are making progress, Fu said. For medical devices, there are now detailed regulatory guidelines that dictate the obligations that manufacturers and consumers (in this case, the hospitals) each have, for example. Philips, a maker of medical devices and other electronics, is considering providing consumers with a list of the software components used on its devices, so that they have a better



Focus on the
end users who
are not trained
IT security
professionals.

understanding of the risks they take when using each product, Fu noted. That may not completely solve the problem, but it would help to know exactly what, including third-party software, is inside these complex systems, said Fu. A fully linked vulnerability database, he suggested, could also help consumers figure out where their risk is and who is responsible for managing it.

Fu noted that the Association for the Advancement of Medical Instruments, the standards body for medical device safety, is also working on this problem. Last summer the group released the first FDA-recognized standards for premarket security on embedded devices, and it is now tackling standards

for software updates. However, he said, there are still crucial questions to be answered, including the need to define the responsibilities of the producer versus the purchaser.

Although cryptography can help address key security properties, such as integrity, authenticity, and freshness, in practice it is difficult to ensure all of these solutions are implemented flawlessly, Fu said. In the scheme of current vulnerabilities, what is most crucial, he argued, is safety—particularly for “cyber-physical” devices with moving parts, such as cars, satellites, and medical devices. He also stressed the importance of human factors. Engineers must focus on the end users, he argued, who, more often than not, are not trained information technology security professionals. “Putting the onus on the user is a pretty big ask for the average American,” he said.

Other questions concern who is responsible for applying the software update—the manufacturer, or the user—and whether the update is optional or required. The degree of risk can be a key factor in these decisions, Fu observed. Several years ago, he said, Medtronic discovered a faulty electrode in its defibrillators. A hardware recall, requiring removal of the device, carried a small but real risk of death. Faced with this conundrum, Medtronic came up with a clever solution: They developed a software update, delivered wirelessly, that would measure the fitness of the electrode and made

a risk-based decision as to whether the patient required the extra risk of removal and reimplantation, or whether the risk was small enough to live with.

The timing of update deployment is also important to consider, Fu noted, and if safety is paramount, then additional verification, validation, and other engineering would be required. Rather than asking, Can we be hacked?, Fu believes the more important questions are the following: How well can we survive these attacks? What kind of tolerance do we have? and How can we fail gracefully?

LEGAL MECHANISMS FOR DISCLOSURE AND UPDATES

In the discussion, Eric Grosse, an independent consultant, wondered if it were possible, legally, to attach a copyright notice to open-source tools in such a way that it would be possible to extract a list of all the software components operating in the system and their respective version numbers, thus allowing a user to more accurately assess whether updates are needed.

While recognizing that the legal questions involved in such a solution are better answered by a lawyer, from a technical perspective, Fu suggested such a solution would find some support in the National Institute of Standards and Technology (NIST) Cybersecurity Framework. He pointed to three pillars for industrial control systems within the framework: First, it's important to know exactly what assets are involved and what the risk is. Second, the proper controls should be deployed depending on the risk. And finally, it is important to continuously measure the effectiveness of those controls. Making all software components transparent, as Grosse suggested, would answer the first question. This is an important step, Fu emphasized, because often even product managers themselves aren't aware of all the software involved in a complex system.

Deirdre Mulligan, University of California, Berkeley, weighed in with a legal perspective. She suggested that Grosse's goal would perhaps be better fulfilled using open-source licensing or the Creative Commons approach, which grants the user a few more rights than copyright as a legal mechanism. That could be helpful, she said, as could creating a set of standardized software disclosures. The National Telecommunications and Information Administration project that Mulligan mentioned in her presentation is researching similar ideas. Whatever path that group takes, Mulligan stressed that detailed disclosures about software updates, maintenance, and other issues would be essential to consumers.

Nicko van Someren, Linux Foundation, mentioned that a Linux project, the Software Package Data Exchange, could also provide a mechanism for making what is inside these products more visible.

OTHER POTENTIAL SOLUTIONS TO ADDRESS VULNERABILITY

Tadayoshi Kohno, University of Washington, asked the group to consider whether software updates could be one of a broader set of solutions to vulnerabilities, as opposed to depending on it as the whole solution. As an example, he pointed to Microsoft's Shield, a product designed to detect attack signatures that are exploiting known vulnerabilities while engineers are working on a patch. He noted that other kinds of instrumentation or mitigations are important in improving cybersecurity, as well.

Recognizing the reality that software updates are sometimes an afterthought in the software community, Fu noted that concepts of "controlled risk" versus "uncontrolled risk" might be useful for framing the challenges after real-world deployment. Software updates should be balanced between those kinds of risks, he said, bearing in mind that there is always the risk that an update could do further harm. This risk balance might also shift depending on whether you are considering harm to a few individuals or harm to a larger population.

Another potential remedy to consider is turning off the software, which could halt intrusions but have other negative consequences. The question of providing software updates is probably not a yes-no question, Fu said, but one where different risks must be assessed.

Fred Schneider added that it may sometimes be feasible to update an associated part of the software system but not necessarily the device itself. In the recent Mirai attack, for example, a firewall from the Internet service providers could have been used to address the large amount of traffic coming from the devices, thus compensating for the fact that the device manufacturers had not built sufficient protections into the affected devices themselves.

3

Microsoft's Approach to Software Updates

Carlos Picoto, Microsoft Corporation

Carlos Picoto is the general manager for the Microsoft Corporation team that produces software updates for all versions of Windows. He opened by sharing Microsoft's stated mission, "To empower every person and every organization on the planet to achieve more," a mission that he allowed can be sometimes challenged by the need to continue supporting legacy software. He offered insights on today's threat landscape before describing Microsoft's approach to addressing vulnerabilities and deploying updates.

TODAY'S THREAT LANDSCAPE

The latest version of Microsoft's operating system, Windows 10, is far more than just software for personal computers, Picoto said. The operating system also runs on diverse products such as Raspberry Pis and augmented reality devices. Reiterating other workshop participants' characterizations of security breaches as "only a matter of time"—a risk to be managed but perhaps never eliminated—Picoto noted that the inherent vulnerabilities of software are nonetheless often underappreciated by users and organizations.

As operating systems have evolved, Picoto said, one crucial lesson is that new threats demand new defense techniques; older solutions are no longer effective at keeping information secure. Picoto told the story of a recent flight in which Picoto watched the passenger in front of him type in his password to log in even though he could have used Touch ID. What's worse, the device had such a long timeout between keys that Picoto was able to see every letter of the user's password.

Consumer inattention to security protections can sometimes provoke changes in vendor behavior and in how security tools are designed. As one example, Picoto explained that consumer behavior is a large part of why Microsoft is moving away from passwords and toward biometric identification. In Windows 10, the company has also experimented with hardware virtualization, which gives separate software components separate trust verification protocols in order to increase security. In addition, he noted, Microsoft features such as Defender Advanced Threat Protection and the Enhanced Mitigation Experience Toolkit enable engineers to detect attacks and begin designing security updates to address them before they become public.

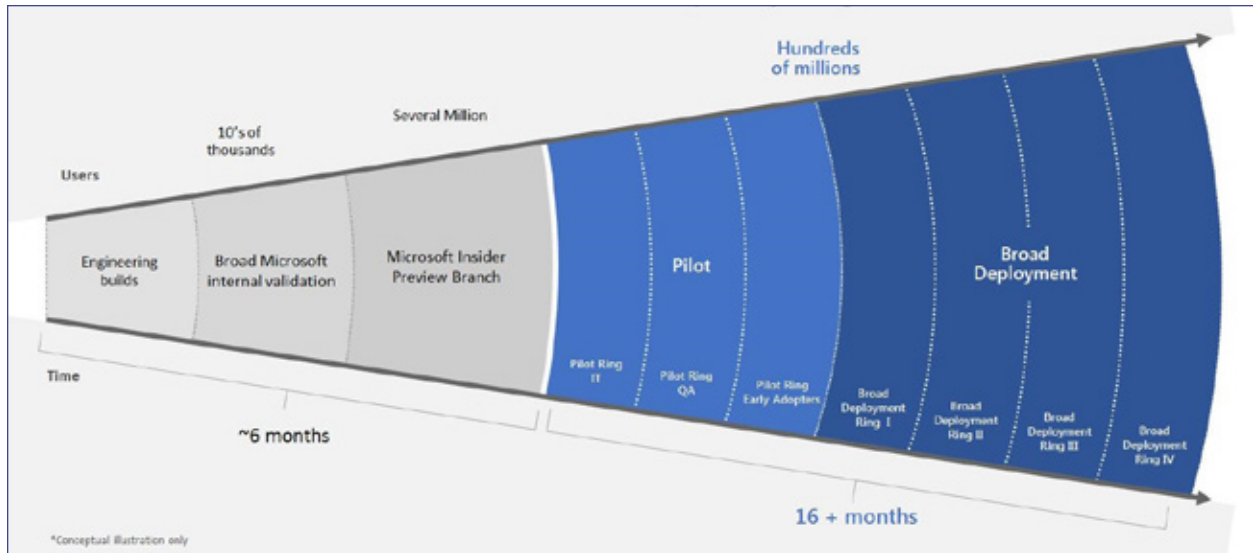
Picking up on a theme that emerged throughout the workshop, Picoto emphasized that the growing landscape of Internet of Things (IoT) devices presents new challenges. Another tension, he noted, is the risk inherent to implementing a security update, which can require system downtime and cost customers productivity.

A NEW MODEL FOR BUILDING AND DELIVERING UPDATES

To address these challenges, Microsoft has created a new product offering: "Windows as a service." In addition to a different sales model from the old product-based approach, this shift has been reflected in "radical changes" to every aspect of deployment and maintenance, including how the software is built, how it is updated, and how those updates are delivered, improving the company's ability to keep customers both secure and up to date, Picoto said.

Picoto's teams are constantly making updates to Windows. Initially, those updates are sent to Microsoft employees, creating a large testing base that helps bring broad validation to the updates. After honing them based on employee feedback, Picoto's team pilots the updates with the Insider Preview Program, a group of almost 10 million enterprise users. Their feedback is then incorporated into the larger Windows updates. This multistep testing and validation process thus improves the overall quality of software updates made across the board.

This process represents an enormous change for Microsoft. Previously, new Windows versions were released in alpha and beta versions before a major release,



Windows as a service: deploying windows.

which occurred every few years. Today's extended pilot period and more frequent deployment of updates enables Microsoft to ensure that each new operating system version is ready for major release while providing new features to customers much faster. However, the fast pace of these releases has been a difficult adjustment for some of Microsoft's enterprise users, who must accommodate far more frequent updates than in the past.

Microsoft's new model also addresses the challenge of what Picoto called selective patching, a previous approach that allowed users to pick and choose the different security patches they wanted to install. He said that software updates are developed with the assumption that all the currently recommended patches are in place, and when this isn't the case, it can cause quality issues. To address this problem, the release of Windows 10 eliminated the option for users to select which patches to install, and instead combined all the available patches into one. All operating system (OS) versions are now structured this way, Picoto noted. Eliminating fragmentation improves the quality and reliability of the OSs for Microsoft, its users, and third-party companies who use the OS as a platform to design applications, said Picoto.

QUALITY UPDATES VERSUS FEATURE UPDATES

Microsoft's new approach involves two separate types of software updates: quality updates and feature updates. Quality updates, released on "Update Tuesdays" (also known as "Patch Tuesdays"), are minor updates that fix security bugs or other quality

issues. Feature updates occur twice a year and deliver new OS features to customers. Customers can choose whether they want to receive feature updates each time they become available or remain with the same features for 10 years, which is how long Microsoft commits to supporting a given OS version. After experimenting with this update model in Windows 10, Microsoft decided to make it available to customers using both Windows 7 (which most of their enterprise customers are currently running) and Windows 8.

Feature updates are reviewed for 4 to 8 months before being made available to customers, at which point Microsoft releases the software publicly but doesn't push it to everyone right away. The first customers to use the update are "determined seekers" who are up on the latest tech news and enjoy being the first to use new software, Picoto said. Users with the newest computers and fastest connection speeds are early adopters, and rolling out software in these contexts first allows Microsoft to learn more about potential quality issues that are not related to platform constraints. Eventually, the update is rolled out to all users.

THE STRUGGLE WITH LONGEVITY

Of course, at the same time that these new OS updates are being released, Picoto's team must also keep up with security updates to the previous Windows versions. It wasn't until July 2016 that his team stopped producing updates to Windows 2000, for example—which is, remarkably, still in use despite being 17 years old. Microsoft's default life-span for software is 10 years, although customers often ask for extensions, Picoto said.

Longevity of systems is always a struggle, because customers want consistency and the cost of change can be considerable, particularly in machines built to function for decades. Windows 2003 was a particular challenge, Picoto recalled; when the scheduled end to updates arrived, there were still unpatched security vulnerabilities, but fixing them was too complicated given how old the software was in the first place.

BEHIND THE SCENES ON "UPDATE TUESDAY"

Picoto sketched a typical Update Tuesday workflow, which starts well before the update is released. Researchers from around the globe are constantly discovering and disclosing vulnerabilities, sometimes giving Microsoft a deadline for making a fix. Microsoft developers then create and deploy security updates, sometimes releasing these fixes first

to qualifying organizations as part of the Security Update Validation Program, which provides additional validation.

A common criticism is that fixing a bug takes too long. The reason, Picoto noted, is that the team is not updating one OS but all of them, in order to minimize the impact to all of Microsoft's customers, regardless of the OS version they might be using. Synchronizing the fixes for all supported versions to arrive at the same time can create challenges.

Of course, releasing updates is not the end of the story for Picoto's team. The next critical task is to monitor delivery and adoption of the updates. Despite their best efforts at running the most complete tests, problems can still occur, for example, if a user configuration runs both Microsoft and another operating system. If there is a problem, Picoto said it usually comes to Microsoft's attention on a Wednesday or Thursday, often through Twitter. This cycle continues at a relentless pace; while preparing the next batch of Update Tuesday deployments, Picoto's team must be prepared to face emergent security issues arising from a previous update that would have to be addressed quickly as well. "You can imagine that with the matrix of browsers and other different operating systems, it gets complex," he said.

A bright spot among these problems is that as customers move to the most recent software updates, Microsoft has seen a drop in help desk phone calls reporting malware, Picoto said. Drastically simplifying user choice (e.g., by eliminating the ability to choose which pieces of the software to update) has also helped keep customers secure and up to date. Picoto closed with data indicating that the latest Windows software update, released in July 2016, has been installed on 79 percent of personal computers running Windows 10 in the United States. Of those devices, 84 percent are also on the latest Update Tuesday patch.

4

Update Issues for Open-Source Software

Nicko van Someren, Linux Foundation

Nicko van Someren is the chief technology officer of the Linux Foundation and a fellow of the Royal Academy of Engineering in the United Kingdom. His talk focused on his role as head of Linux's Core Infrastructure Initiative (CII), a nonprofit organization funded by the information technology industry that provides training, testing, and financial support for software security projects. Its mission is to improve the security of open-source software, an essential part of the Internet ecosystem.

CII was created after the 2014 Heartbleed bug found in OpenSSL, which left approximately 70 percent of the HTTPS services on the Internet vulnerable to security breaches of sensitive information. The Heartbleed bug was eventually patched, but 7 percent of Internet servers today are *still* vulnerable to Heartbleed, despite the fact that the patch was released 3 years ago and was considered a critical security vulnerability. The experience sparked the creation of CII and underscores the fundamental challenge of disseminating software updates in the open-source environment, said van Someren.

CII focuses on a variety of methods to secure open-source software. One method is to devote resources to projects focused on open-source security, such as OpenSSL, OpenSSH, and GnuPG. CII also invests in infrastructure like the Network Time Protocol daemon (NTPD), timing software used until recently by many of the world's major stock exchanges, which, van Someren mentioned, happens to be maintained part-time by

a single developer. In response to a question from Bob Blakley, CitiGroup, Inc., who wondered if there had been a risk analysis of the consequences if the NTPD stopped working or, worse, if it came under the control of a hacker, van Someren noted that CII had conducted a security audit on the NTPD code and that maintainers of NTPD have established the Network Time Foundation to support its continued maintenance. In addition, CII established a project known as Census that attempts to identify at-risk components of NTPD and similar infrastructure to determine the factors that contribute to risks, gauge vulnerabilities, and identify the community of people who could step in to maintain it if circumstances should change.

Elaborating on the NTPD story later in the discussion, van Someren described CII's approach to risk assessment for maintenance of open-source software. His team looks at several characteristics, such as how many people maintain it, how vibrant a community it has, and how long it takes for bugs to be fixed, to calculate a risk score for each component. For products that are found to lack consistent and robust maintenance, CII is creating a "software orphanage" where engineers can monitor this untended software, respond to bug reports, and create and deploy updates.

In addition to supporting outside projects and providing training, CII also builds its own programs to develop more secure software and to improve security practices across the Internet, both for software they consider to be core infrastructure and for open-source projects generally, because, as van Someren noted, "We don't necessarily know what will be core infrastructure in a few years' time." As an example, he pointed to the fact that Node.js (a platform for building network applications) is the most active open-source code used right now, although 5 years ago no one would have predicted that JavaScript would be so widely used in this context.

INHERENT CHALLENGES TO SECURING OPEN-SOURCE SOFTWARE

Open-source software is not inherently any more or less secure than other software, but it does have a few different characteristics that affect the security landscape. First, its development process is fragmented, not streamlined, and often spread out across more people than in a closed-source development team. Also, instead of having top-down management pushing specifications, open source operates with a more collaborative spirit and an emphasis on running code.

These differences add to the complexities of software updates. Eric Grosse, an independent consultant, commented during Kevin Fu's presentation that it might be helpful to see all the components within a software program in order to assess the need for updates on each of them. In open-source software, van Someren said, there is

often such a deep, layered stack of components, highly specialized to the user's unique needs, that such transparency may not be feasible. These deeply layered setups also make sending updates difficult because of all the testing that has to be done for each component, in order, from the bottom of the stack to the top. While in theory open-source users can refigure their stacks at any time, in practice they are more likely to wait until another user has tested the piece one level down, which adds delays to any software update deployment as updates filter up the stack.

This collaborative, distributed model also means that it can take longer for the open-source community to identify and respond to vulnerabilities. Once a patch is built and made available, it still has to go through several layers of distribution, such as



GitHub, Red Hat, and others, before reaching users. "We have got a bunch of extra layers, I think, that make the handling of software updates a bit more complicated and a bit different," van Someren said.

Open-source software is not one monolithic entity, and it is deeply embedded across commercial software, numerous operating systems, and other crucial products, making the challenges of open-source software updates both diverse and pervasive. There is no one-size-fits-all solution, van Someren said, "because pretty much every project has its own unique way of doing things."

Each project even has its own way of describing software updates, he observed. For example, when OpenSSL releases an update, developers call it a

"security update" and detail all the changes it involves. With Linux kernels, on the other hand, there are no specific "security bugs," because the philosophy in that community is that *all* bugs could affect security. A user could reasonably wonder, however, whether it is necessary to install all the new kernels, which are released weekly, even if they are not specified as improving security, since there is a risk that changing kernels frequently could destabilize their system. By not indicating how severe the vulnerability is, Linux kernels could be inadvertently turning users away from deploying essential updates.

Another major difference with open-source software is the matter of liability. Commercial software often involves legal obligations for service and updates; with open-source software, there is no such agreement between developers and users. While some businesses offer service-level agreements to support open-source software, that is not feasible for every case or every organization. Those responsible for the Linux kernel, for example, do not want to take on any level of liability. "There are

a whole host of problems and opportunities that crop up from that abrogation of liability,” van Someren concluded.

“Free software,” a subset of open-source software that is free to use *and* alter, also brings a host of unique challenges. The most common example of this is the GNU Public License (GPL), which is how the Linux kernel is licensed. TiVo at one point used code released under GPL, but its devices restricted users’ ability to update the software themselves by requiring their devices to only run code signed by TiVo. When the next version of GPL was drafted, designers added “anti-TiVo-ization” code to prevent this behavior. While this was considered a victory for free software, it raises new software update challenges, such as the question of who is authorized to sign an update before it runs on a device.

Van Someren suggested that those security liabilities are a good reason to avoid GPL on open-source products, despite some technical solutions that can improve security. In general, he noted, it is said that software can be secure or software can be free. “You can have one or the other—but you can’t have both,” van Someren summarized.

Another CII project is working to create a global platform of trusted execution engines for mobile devices. Some options being pursued would allow users a certain amount of control, and van Someren expressed optimism about their potential value.

DEALING WITH THE FEAR OF DESTABILIZATION

Right now, the biggest impediment van Someren sees to software updates for open source is users’ common concern about whether a software update will cause instability across their systems. This concern partly stems from that fact that users aren’t sure exactly who made the patches or how they were tested. He told the story of a Wall Street chief information officer (CIO) who “nearly had a heart attack” when van Someren pointed out that many of the bank’s front-end Web services came from GitHub, which means that an attacker could commit code to a project used by the bank and thus insert vulnerabilities into its live online banking system. The CIO’s first instinct, to stop applying security patches, is not necessarily the best move. The story illustrates a key question that everyone is wrestling with in open source, said van Someren: Who has authority to determine what code is “good” and what code is not?

In the discussion, Forum Chair Fred Schneider reiterated the sense that fear of system destabilization is a main reason users may ignore updates. Van Someren noted that several Linux distributions do offer rollbacks if an update causes destabilization. Vendors of open-source distributions (as opposed to the community using free software)

also offer restabilization, because their business model depends on keeping customers satisfied.

Open-source creators should, in theory, retest any existing components that they add to new software, but that takes a long time to percolate up through all levels of software, especially when components are stacked so deeply. Open-source distributors like Red Hat have done this extra testing, which should increase stability, van Someren noted, but he emphasized that there really isn't a single solution that would work in all cases. Because the open-source world is so varied, many projects have come up with their own systems for handling the destabilization issue.

Carlos Picoto, Microsoft Corporation, noted that using a public or well-known interface to deploy security updates does not always prevent destabilization. For example, he pointed to software that is either not using the common Microsoft API or is using it incorrectly; he noted that Linux kernels can be particularly problematic for Microsoft updates, for example, as can vendors who do not use properly compatible software.

OPEN SOURCE IN THE INTERNET OF THINGS

Van Someren wrapped up with a discussion of the role of open source in Internet of Things (IoT) devices. As noted at several points in the workshop, startup IoT manufacturers often assume their products have a short life span and are, essentially, disposable. Van Someren countered that while that may be true for more esoteric products like a Wi-Fi-connected hairbrush, it is not for household necessities like water heaters and lighting systems, and it is certainly not for cars or industrial machines. "We need to think about what to do when, not if, the [IoT] vendor goes bust," van Someren said.

One idea CII is considering for these situations is "code escrow," in which vendors put the last version of their code, along with dedicated funding, into a trust that can then be tapped to continue to support the software when the company goes under. How exactly this would work—and whether companies would sign up for it—are open questions, van Someren said. In the discussion, Bob Blakley, CitiGroup, Inc., delved deeper into how companies could potentially be convinced to get on board with the idea of code escrow. Noting that the Dodd-Frank bill required financial institutions to set aside capital in case of emergencies, he speculated that regulations requiring software firms to escrow code on the basis of public interest might be worth considering.

Peter Swire, Georgia Institute of Technology, questioned how such an escrow system would work in the context of bankruptcy law, which prevents companies from

holding on to assets amassed within 6 months preceding a declaration of bankruptcy. Van Someren clarified that, ideally, an escrow would be established at the outset of launching a product, or, failing that, when the reality of bankruptcy is at least 6 months away.

In response to a follow-up question from Blakley, who suggested that such a system would require regulation to enforce it, van Someren clarified that he would prefer a mechanism such as a checklist or badge-earning system where open-source IoT projects could self-assess their ability to create a secure development life cycle, similar to Linux's existing Best Practice Badge Program for open-source software. Such criteria would improve security outcomes by allowing companies to self-certify, but then make that certification public, where it could be scrutinized and verified. The best outcome, in van Someren's view, would be to get the industry's buy-in on such a program before approaching the Federal Trade Commission and asking it to "put some teeth behind it."

One other possible mechanism for protecting consumers from the dangers of outdated IoT devices, van Someren suggested, is to take advantage of what's known as a watchdog, a system configured to detect critical failures and trigger a complete reset of the system. In this vein, software updates could be seen as "food to feed the watchdog," allowing a device to maintain its Internet connectivity. For this to work, the devices themselves—such as garage door openers, lights, and hot water heaters—would need to be able to perform their functions without Internet connectivity, although they would stop communicating with their mobile and Internet-based controls. This limits damage when devices stop receiving software updates. "We need to make sure that the mode of failure is disconnection, rather than the product stopping working," van Someren said.

5

Cisco's Approach to Software Updates

Ed Paradise, Cisco Systems, Inc.

Ed Paradise is vice president of engineering for the Security and Trust Organization at Cisco Systems, Inc. He addressed today's attack environment, Cisco's response to security vulnerabilities, and key considerations for the network infrastructure across the Internet.

Paradise is responsible for making Cisco products secure and trustworthy. His teams define and maintain Cisco's Secure Development Lifecycle (CSDL), including its engineering, design, and security update procedures, plus product testing, evaluation, and incorporation of feedback. As a center for security innovation, his teams also develop embedded security technologies, which are deployed broadly across Cisco's full portfolio. Examples of these technologies, which are mandated by CSDL, include CiscoSSL, trust anchor modules, secure boot, and image signing, as well as many other hardware- and software-based security technologies.

THE ATTACK ENVIRONMENT

Paradise shared projections indicating that by 2020 global Internet Protocol (IP) traffic is expected to double, to 2.3 zettabytes, and that broadband speeds are also expected

to double. Two-thirds of that traffic will be mobile and wireless, and 82 percent of all consumer Internet traffic is projected to be video streaming. This large number of devices and vast amount of bandwidth raises the stakes considerably, Paradise suggested, when considering the prospect of another attack like Mirai.

Paradise described an experiment demonstrating how rapidly and frequently software is contacted and probed. He connected a brand new device to the Internet and tracked what happened. Within 5 seconds, another device made contact. Within 5 minutes, a device from China made contact, and another actor checked if the device could be turned into a botnet and used in a distributed denial-of-service (DDoS) attack. Within 30 minutes, an actor had run a full vulnerability scan on the device. By the end of its first 24 hours connected to the Internet, about 4,000 different machines tried to contact the device.

As his experiment demonstrates, devices and software can be compromised in a matter of seconds. According to Paradise's research, about 70 percent of contacts are attempts to use the device as a botnet in DDoS attacks. Other key risks include fraud and data theft. "There are a lot of bad things happening, by a lot of bad actors, pretty quickly," he summarized.

CISCO'S APPROACH TO ADDRESSING VULNERABILITIES

Paradise detailed how Cisco handles security incidents. Vulnerabilities can be identified through a variety of mechanisms, including internal testing and validation, customer notification, or open-source notification. Depending on the vulnerability, it can take minutes or months for Paradise's team to learn about it. Activating Cisco's Product Security Incident Response Team Process (PSIRT), the team assesses the scope and scale of the situation and then comes up with a plan for fixing the problem and alerting users. Ultimately, he explained, an update goes out simultaneously to all the affected customers, and the team monitors and incorporates any feedback they receive to improve the process, repeating fixes as necessary until the crisis is over. Prompted by a question from Bob Blakley, CitiGroup, in the discussion, Paradise reiterated Cisco's policy of providing simultaneous notification to all users, rather than identifying priority users for advance notification.

Eric Grosse, an independent consultant, recalled that in his experience, it was difficult to know which Cisco updates to install because the names were so confusing. Paradise recognized that that was indeed an issue, mostly because of the large number of products that Cisco offers, and he noted that the company is working to beef up

customer service to help customers navigate the product portfolio in order to find needed updates.¹

Steven Lipner, an independent consultant, asked how Cisco handles updates to products that include open-source components. Paradise noted that Cisco does a significant amount of testing on these products, and it also has a robust certification program. For example, Cisco has an automated registration and verification process for certain software that incorporates CiscoSSL (based on OpenSSL with different features and functionalities). The two groups have a strong working relationship for when joint

updates are necessary, and Cisco helps OpenSSL by doing daily automated testing.

That partnership is an example of a good relationship with open-source software. However, Paradise noted, there can be problems—for example, when the Heartbleed bug came out. Tens of thousands of products were affected, and because of that large number, it took a long time to complete all the necessary verifications.

Forum Chair Fred Schneider asked Paradise to describe the extent to which Cisco controls its software and how it is used, and the extent to which Cisco is therefore able to mitigate the potential for destabilization. Paradise said that Cisco maintains tight control over the majority of its software—although there

are some customers who want to add code to Cisco products—and he underscored that updates are thoroughly tested to prevent destabilization in the context of use that Cisco can control (e.g., not necessarily accounting for any code added on top of a Cisco product).

Paul Kocher, Cryptography Research Division, Rambus, Inc., asked what Cisco might be doing to help provide a safe platform for vulnerable or unmaintained Internet of Things (IoT) devices to continue to operate, as discussed during Nicko van Someren's presentation (see Chapter 4). Paradise suggested that there may be opportunities to prevent future DDoS attacks by properly securing networks for IoT devices and embedding a “kill switch” that detects when a device is performing a function that it was not built to perform.

Building on this discussion, David Hoffman, Intel Corporation, agreed that IoT devices are vulnerable because they don't have regular security updates the way that

There are opportunities for network infrastructure companies to create relationships with IoT providers to improve device security.

¹See, for example, O. Santos, 2017, “Keeping up with Security Vulnerability Updates,” *openVuln API*, Cisco Blogs, January 24, <http://blogs.cisco.com/security/openvuln-update>.

computer operating systems do. But, he said, companies may be prevented from doing security scans on these devices by the Computer Fraud and Abuse Act. He suggested that companies might overcome this limitation by working together to scan devices for security issues or breaches as an added service to companies and individuals, thus increasing transparency and alerting Internet service providers to potentially harmful traffic.

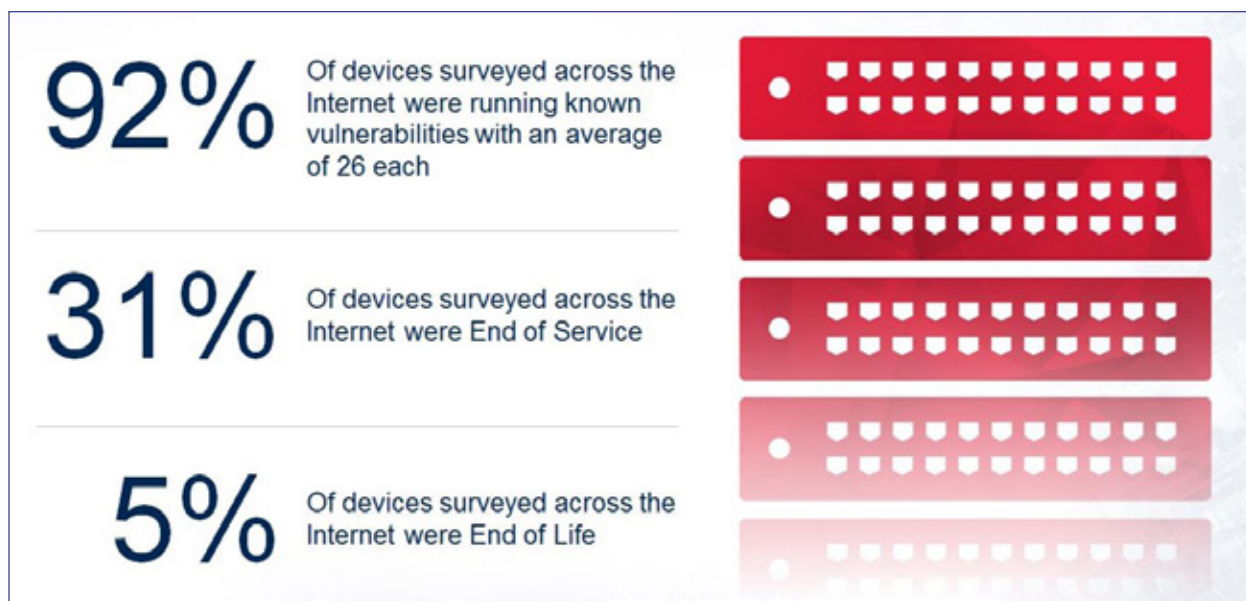
Paradise agreed that these issues present an opportunity for Cisco and other network infrastructure companies to create relationships with IoT providers to improve device security, including a multi-tier “IoT ready” certification program. Using cameras as an example, Paradise said such an arrangement could allow Cisco to offer consumers a list of camera models that work best with a Cisco router and would be supported against future security risks. Choosing to buy a camera *not* on that list would mean that a customer is taking the risk, knowingly, that the camera may not be protected and could be used to transmit sensitive data or propagate harm through the network.

UNDERSTANDING USER PERSPECTIVES

Zooming out to the technology-using population as a whole, Paradise reiterated the notion, raised elsewhere during the workshop, that most users live “in a world of false confidence” about the true level of their security. A recent Cisco survey revealed that a majority of users have “strong confidence” that, among other things, they can detect security vulnerabilities in advance (51 percent), that they can defend themselves against



Attack awareness fades confidence. SOURCE: Cisco, 2016, *2016 Cisco Annual Cybersecurity Report*, http://www.cisco.com/c/m/en_us/offers/sc04/2016-annual-security-report/index.html?KeyCode=001031952&_ga=1.57614639.400663907.1438105234.



Reliability breeds complacency.

such attacks (54 percent), and that they regularly review security policies (56 percent). He noted, however, that these numbers are slowly dropping, as users become less confident with every well-publicized security breach.

Another Cisco research report showed that 92 percent of devices currently connected to the Internet have an average of 26 security vulnerabilities each. This isn't just old iPods, he reminded the participants: "This is more about devices that are on the Internet as part of the infrastructure." A whopping 31 percent of connected devices are classified as "end of service," meaning that manufacturers will no longer support security updates, and the manufacturers are not patching vulnerabilities. "Even if we wanted to," Paradise explained, "we probably don't have the capability to service that device any longer."

After devices enter "end-of-life" status, they are no longer serviced or sold at all. Five percent of devices currently connected are in this category. The main reason people hang on to devices, particularly between end-of-life and end-of-service, is financial: If the device is still working, consumers don't feel the need to replace it. Convincing these customers of the security risks they are taking can feel like a losing battle.

Richard Danzig, Johns Hopkins University Applied Physics Laboratory, asked about the role of psychology in determining how people perceive security. He pointed to the cognitive psychologists Amos Tversky and Daniel Kahneman (whose collaboration is the subject of the recent Michael Lewis book *The Undoing Project*²), who study the psychological factors that color people's perceptions of different types of risky

²M. Lewis, 2017, *The Undoing Project: A Friendship that Changed the World*, Norton & Co., New York.

situations. Paradise agreed that psychology is a factor when customers are deciding whether the risks they face are large enough to invest in new equipment. He suggested that the industry could do more in the way of sales, marketing, or consumer education to highlight the security risks customers invite when they continue to rely on end-of-service or end-of-sale equipment. “We haven’t presented the arguments to our customers,” he said.

Paradise suggested that part of the solution is that customers need to be taught better questions to ask about security features when they are buying new equipment. While software companies are motivated to incorporate new features into their software in order to woo customers and grow revenue, unfortunately, “we don’t have a large base of our customers asking for those security features,” he said. Cisco is pursuing research on “the psychology of security” in order to better educate their customers about the security risks they might be taking when they choose not to update or replace their software.

6

Ensuring Robust Firmware Updates

Dave Whitehead and Edmund Schweitzer, Schweitzer Engineering Laboratories

Dave Whitehead, vice president of research and development for Schweitzer Engineering Laboratories (SEL), presented a detailed look at SEL's industry space, products, and update challenges. Edmund Schweitzer, president and board chairman of SEL, was on hand to answer questions.

SEL is a software and firmware company that specializes in inventing, designing, building, and supporting electric power systems for industrial companies and global utilities. With a mission to make electric power safer, more reliable, and more economical, Whitehead described SEL as "the brains of the electric power systems." While all of the design, manufacturing, and support is done within the United States, SEL serves clients in about 160 countries, making it a global company.

UNIQUE ATTRIBUTES OF SEL'S MARKET SPACE

SEL makes a suite of products that are meant to work in concert to support electrical power systems. The company's protection and control devices measure electric current and voltage, determining at every millisecond whether a circuit breaker or a power line needs to be tripped. SEL also makes automation equipment, such as special-purpose

computers that interpret data and make higher-level decisions about power use, passing those decisions up to an information center or down to a protection device. In order to keep these systems secure, SEL also builds switches, firewalls, and gateways. Finally, SEL makes operations software that enables client technicians to make decisions, such as determining when to send control commands to shut a power line down for maintenance. “It’s a very layered approach to how we go about controlling and operating the power systems,” Whitehead said.

Electric power systems are managing electric power and transmission concepts that are more than 100 years old and were created well before the Internet and firmware. Whitehead noted that this is actually good news for security: If power systems lose communication or other computerized controls, they can still remain functional. On the other hand, our strong reliance on electricity today means the stakes are extremely high. “When we mess up, letters get written to the President . . . so we take this stuff very seriously.” Whitehead said.

Whitehead noted that SEL has several advantages that more general software companies like Microsoft or Cisco do not. For example, it has the luxury of building its systems to do one thing only, which is to protect power systems: “We don’t have to be all things to everyone,” he said. SEL benefits from the ability to tailor its systems to their intended use and only include those necessary features. In addition, SEL systems are created to be static. Client utilities expect to use their designs for 20 years, with minimum upgrades, because once the systems are built, they are considered “done.” Therefore, in general, SEL does not anticipate frequent firmware updates, although he noted that one of their devices does use a Microsoft product, so they must be aware of any patches and notify affected customers accordingly.

SEL’S APPROACH TO SECURITY

Whitehead described security as integral to the SEL product life cycle and said this ethos is reflected in well-defined, robust engineering principles implemented all the way to Capability Maturity Model level 5. Founder Ed Schweitzer drew on his previous experience at the Department of Defense to place security at the forefront of the company’s products from the get-go. For example, from its first product 30 years ago to today, SEL has had two levels of access to its devices: one for a technician to view data, and another that allows a higher-level user to set and configure a device.

In addition, SEL engineers own every piece of code in its devices and have revision control for every device in its factories. They also use automated test code analysis and human teams to review every piece of firmware top to bottom. They create single binary

files that match a firmware product to its version number, allowing them to track and fix any security problems. The final step in product development is negative testing, in which engineers deliberately try to break every aspect of the firmware. “We really want to beat up the product before we ever release it to any of our customers, because . . . once I release a product out into the electric sector, we’ve got one chance to get it right,” Whitehead said.

SEL also provides a mechanism for customers to verify that all firmware is from SEL before installation, which further enhances security. SEL also now includes signatures in its software, preventing it from being loaded onto SEL firmware until it passes cryptography checks, Whitehead said.

MANAGING FIRMWARE AND SOFTWARE UPDATES

SEL takes a somewhat unique approach to managing firmware upgrades: The company itself keeps track of which customers are using every piece of firmware, so it can alert those customers when updates are needed. SEL also provides education and training for customers to implement SEL products securely, even providing cybersecurity services tailored to each client’s needs.

Firmware management has evolved since SEL began. In the 1980s, firmware was manually upgraded—an engineer had to physically remove an old memory chip from a device and add a new one. In the 1990s, flash memory enabled firmware updates through communication ports. In the 2000s, substations began using Ethernet, which made upgrades easier but also presented cybersecurity challenges, and so the Internet was not often used for firmware upgrades. While SEL *could* push out firmware upgrades today online, because of the security risk, it avoids this approach, particularly for upgrades to sensitive functionality like protection devices. A technician will instead visit a client site, take the whole device out of service to mitigate any security risks or errors from the upgrade process, and then manually install new firmware.

If a defect is found, SEL will resolve it, push out updates, and notify customers through a security bulletin. Whitehead noted that the company issues a security bulletin each month whether there have been any updates issued or not; this helps customers prove to regulators that their systems are up to date. If there is a security defect, the client has 30 days to evaluate whether or not it affects its equipment, and if so, whether the equipment needs to be taken offline for servicing, which can be a major undertaking involving a planned outage. The firmware is then upgraded, tested, and verified before being put back into service.

Prompted by a question from Richard Danzig, Johns Hopkins University Applied Physics Laboratory, Whitehead elaborated on this 30-day window during the discussion. The window starts when SEL notifies a client about an available fix for a discovered vulnerability. The client then has 30 days to decide whether the vulnerability applies to them, what steps they will take, and how they will mitigate any risk. While the actual fix does not have to be implemented within 30 days, the operator must be able to demonstrate a plan by the end of that period and potentially justify any delays in an audit. While clients may prioritize certain updates over others, by and large they are very responsive to these notifications, Whitehead said.

SEL also creates updates that provide new features, usually in two to four releases per year. These deployments are handled separately from security updates so that clients can respond to the two types of updates accordingly. Whitehead noted that policy-driven software updates can be problematic, because policies can never truly fit every possible situation and sometimes can end up driving out other creative solutions to problems. In his view, the U.S. government should inform power system operators of security threats, but leave specific decisions about features or security updates to those who own the system's assets.

OTHER MECHANISMS FOR MAINTAINING SECURE SYSTEMS

While power systems operate on networks, Whitehead stressed that there are critical steps to ensuring these networks cannot be penetrated by attackers via the Internet. First and foremost, he said, operators should never connect their systems to the Internet. They should also conduct audits to ensure there are no such connections. SEL clients maintain private, secured networks for their control networks that are completely separate from any other corporate information technology functions, said Whitehead.

In addition to layering and separating different parts of a system and properly educating clients on security defenses, Whitehead noted that SEL also scrutinizes and verifies every aspect of its supply chain in order to track parts, enhance security, and build trust. Other SEL methods for increasing security include maintaining private security plans, compartmentalizing systems, and monitoring and investigating any unusual events.

Because they involve physical systems, one advantage of SEL products, Whitehead noted, is that it is easier to identify problems compared to, for example, online banking, where it can be hard to determine what exactly has happened when there is a breach. Because all of the systems in use are constantly measuring the power system, information about any unusual occurrence can be validated among the systems.

THE “AURORA” VULNERABILITY

At Danzig’s request, Whitehead and Schweitzer described an incident known as Aurora and discussed its security implications. In a power system, a circuit breaker has two parts. When a breaker is opened, the frequency and phase must be the same on both sides; if they aren’t, it can cause excessive torques in the rotating machinery and cause damage. At Con Edison, Schweitzer’s grandfather created a way to record every instance of this occurrence after observing that operators would sometimes intentionally cause it to happen in order to enjoy the noise and drama of watching the breaker synch up again.

An intentional demonstration of this phenomenon, called the Aurora vulnerability, in 2007 raised fears that it could be exploited to break electric power systems. In the incident, operators set up a 4-megawatt diesel generator incorrectly on purpose and then physically manipulated it to trigger the Aurora phenomenon. The incident was not caused by a firmware vulnerability, Schweitzer said, and using this approach to attack a power system would require such a long chain of events that the company does not view it as a significant vulnerability.

FOSTERING DEVELOPMENT EXPERTISE AND BROADER LESSONS FOR THE SOFTWARE INDUSTRY

Bob Blakley, CitiGroup, expressed his admiration for SEL’s robust development process and asked Whitehead how the company manages to recruit qualified developers. Whitehead explained that a close association with Washington State University allows SEL to tap engineering and computer science talent early and train young developers on the company’s methodical engineering design process. While the universities teach the fundamentals of mathematics and engineering principles, Whitehead emphasized that industry is where design and practice are taught, so SEL expects to need to train recruits on its process.

Tadayoshi Kohno, University of Washington, asked if SEL represented the industry as a whole, or is ahead of its competitors, and whether there are any specific lessons the SEL experience could offer for the software industry more broadly. Schweitzer responded that SEL evaluates its products in comparison to those of competitors and remains confident in its work. In terms of broader lessons, Schweitzer noted that the intelligence community could be more involved, and that industry-wide sharing of information about threats could also improve security. For example, SEL writes many technical papers every

year that become resources for the broader community. One, for example, focused on how the company would know if power systems had been hacked, concluding that because it's easy to measure use in power systems, a hack would be fairly obvious. Other resources include SEL's in-depth training and its Modern Solutions to Power Systems Conference. Schweitzer noted that this year's conference theme, "The Roots of Cyber Insecurity," could be particularly relevant to this discussion.

7

Updates in the Consumer Electronics Industry

Will Drewry, Google, Inc.

Will Drewry is a principal software engineer at Google, Inc., where he is responsible for ensuring privacy and security protections on consumer products. He described the challenges involved in building and securing consumer electronics, including the use of off-the-shelf components, business-side considerations at play, and a variety of technical factors.

Drewry explained that the products he has worked on typically include an operating system and an application layer on a physical device such as a phone or laptop, although these products have a variety of update strategies. While one product line receives feature and security updates every 6 weeks, another has monthly security updates and less frequent feature releases, and still other products follow an “update-on-demand” schedule.

CHALLENGES FROM USING OFF-THE-SHELF COMPONENTS

While the makers of operating systems and Internet of Things (IoT) devices intend to make secure products from the beginning, it can be difficult to do when working with commercial, off-the-shelf parts and trying to keep up with the speed of production

required to survive in the technology industry, Drewry said. In this environment, the potential for updates can be seen as a “crutch,” because security can always, in theory, be improved after deployment.

Drewry offered a hypothetical example of a new device with an application platform that can add more and more features over time. A designer could choose to start layering in security in the browser, the application layer, or somewhere else. The security process might be rushed if the manufacturer wants to bring this device to market quickly and knows updates can be made later. Most builders use best practices such as compartmentalization, tracking information flow, and balancing when assets are available, but no code is invulnerable.

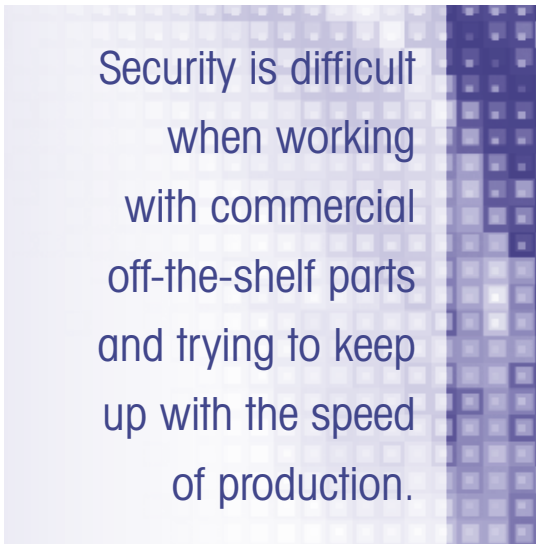
Many products, including all of the ones Drewry has worked on, use open-source components like Linux kernels, he said. The kernel has to be inspected to determine if it’s trustworthy, and kernels often lack a full annotation of security bugs; because it’s unclear

which updates include security fixes and which do not, Drewry emphasized his belief that it is important to update the Linux kernel frequently.

Drewry also oversees the hardware components of these devices, where the main processor has to interact with the software kernel being used. This is complicated by the fact that these processors are built by third parties and may not be compatible with a certain kernel, or the company may not offer direct customer service. Processors are often manufactured for short-term use; for example, a camera component on sale now will likely be replaced with a newer one within a couple of years. In that instance, if Drewry is using

that older camera in his hardware, he will have to update the kernel, which becomes an increasingly complex process, or ignore the updates.

The Linux kernel isn’t the deepest layer to be concerned about, either. Mobile or device processor vendors layer software on top of a chip—that may itself be modified or adapted, creating an opening for problems. A few years ago, for example, Intel released microcode updates that could inadvertently alter instruction codes, causing some alarm. The full-featured operating systems sold today in “Trusted Execution Environments” may not always be trustworthy, Drewry noted. Computer chip makers feel pressure to include software on their products as a way to differentiate themselves from others, although Drewry would prefer that they come without it. “You can’t even pay them to take it off,” Drewry said. “I keep trying.”



Security is difficult
when working
with commercial
off-the-shelf parts
and trying to keep
up with the speed
of production.

All of these extra layers from the chip up through the software pose added risks. “I think there are lots of ways you can ameliorate that risk, but the farther up you go, it gets harder,” Drewry said.

BUSINESS-SIDE CONSIDERATIONS

Drewry faces these challenges from the perspective of an industry giant; he noted that the many smaller companies and individuals, such as those launching products with funding from Kickstarter.com, for example, rarely have the resources needed to invest heavily in software security. Instead, they are typically entirely dependent on the protections that are already built into off-the-shelf components available from third parties. Google has sufficient resources to spend time compartmentalizing software that it purchases from code that it writes in-house. The company also isolates the Linux kernel from other layers, and invests in projects like the Kernel Self-Protection Project (supported by Linux’s Core Infrastructure Initiative, described by van Someren) that increase kernel security. But this process can be messy even within a company like Google, and navigating such protections could be much harder for smaller companies.

The mechanisms to support the sort of secure development life cycle discussed throughout the workshop could soak up a lot of a company’s bandwidth, Drewry emphasized. Creating updates is a business expense. Key management is also necessary to keep updates secure, but that brings in more people, more technology, and enormous expenses, “just to get a modicum of assurance,” Drewry noted.

Because companies want their customers to use the updates, it is also important to carefully manage how and when they are deployed; updates can’t be so frequent that users ignore them, and they can’t cause frustrating instabilities. In addition, as mentioned previously in the workshop, even the most secure updates in the world still need a secure delivery route to the client, who then has to deploy them correctly. A company’s “update agent” might need to build a secure connection to send the update, creating another layer that has to be implemented, protected, and given privileged access to the relevant devices.

Drewry shared an experience he had shipping an update for Chromebooks: To avoid the situation where users cannot update without logging in (and thus being immediately vulnerable to an attack), Google devised a process by which users could get an update on the log-in screen. This also avoided customers having to be walked through a tedious workaround to install the necessary updates and finally log in.

Experiences like that demonstrate that in addition to thinking about how to provide security, developers must consider how that security can be maintained within

a system that comprises multiple interconnected layers, he said. In a world in which even the chips have software, nothing can be taken for granted. Windows 10 recently added the “Windows Trusted Boot.” Older TiVo devices used to rely on the kernel to check that certain key files had not been altered, and Google is constantly checking all the software signatures and configurations every time a device is restarted, not just when software is first loaded.

ADDITIONAL TECHNICAL COMPLEXITIES

Rollback protection, a mechanism to prevent the system from reverting to an older version of the software, raises additional challenges, Drewry said. The record of having converted to the new version must be stored in a way that will prevent tampering, or at least show evidence of having been tampered with, and this protection must be stored in an appropriate place. One of the most common choices is to use Replay Protected Memory Block (RPMB), built on software and firmware, in the main storage device. That is still problematic, he noted, due to the increased reliance on the storage device vendor’s software development and key management and update practices. Because processors run on a process size that is too small to include flash storage, this functionality cannot be put on the chip itself. As a result, “We lack this essentially critical secure storage for making decisions,” Drewry said, and developers have become reliant on third parties that are part of their trusted network.

Another complex issue is the “single signer” problem. Although users might be protected from unauthorized attacks, what if the signer is the one compromised? “We haven’t actually solved the problem of protecting them against us,” Drewry said, emphasizing that even 1-bit code errors can become vulnerabilities in the right context.

Drewry discussed the importance, and challenges, of preventing targeted attacks. Although Google focuses on making it difficult for hackers to target specific users, it’s still fairly easy to identify people with just a few pieces of information. Removing unique identifiers from the system can help make it harder for attackers to target a specific victim.

Broadly speaking, while software transparency may be an excellent reactive mechanism and make attacks more complicated, Drewry emphasized that it is not proactive and cannot protect users from having their identity or information compromised in the first place. It is important for companies to seek a more proactive solution, he said. One avenue that could improve security updates is finding the right incentives so that designers *want* to build more secure systems, he suggested. Drewry believes incentives for chip makers, operating system designers, and other key players

will be a strong motivation. “There is room in the industry to provide these tools,” he said, adding that Linux and big companies alike could help move the process along.

Eric Grosse, an independent consultant, asked Drewry if there were best practices the community could use to prevent a new device, fresh out of the box, from being immediately attacked (as Ed Paradise had described in his presentation), while still in need of a software update. Is there a way, he wondered, to build in special protections to keep a device from being penetrated on its very first startup?

Drewry expressed agreement that initial startup could present added vulnerabilities, offering as an example the story of Chromebooks that were manufactured in large numbers and then ultimately sold by resellers up to 2 years later. Seeing the risk that these machines would be significantly more vulnerable to attack if the on-launch software update took longer than 30 minutes, the company focused on creating an update that could be complete within that time window. Another solution being used increasingly is to develop systems that update the software on devices while they are waiting in a factory or warehouse so that the software isn’t too outdated by the time the device reaches the customer, he said.

Software Updates in Automotive Electronic Control Units

John Vangelov, Ford Motor Company

John Vangelov manages the embedded modem features group at Ford Motor Company. He shared a history of automotive electronics before discussing the update challenges faced in the automotive sector today, including the limitations to network capabilities and the challenges of software delivery methods.

HISTORY OF ELECTRONIC CONTROL UNITS

When they were first developed in the 1980s, electronic control units (ECUs) were housed on chips that were either erasable-programmable read-only memory (EPROM) or masked read-only memory (MROM), Vangelov said. EPROMs were slow to program but could be erased and reused, although the process was tedious. MROMs were a better value, because they could be programmed on a large scale, but they couldn't be updated or rewritten. There was also EEPROM (electronic EPROMs), which was cost-prohibitive for wider use but suitable for storing essential automotive configurations or calibrations.

Wherever software was housed in the early days, this software was very difficult to alter once it was in vehicle production, Vangelov said. Most issues were addressed via

direct part replacement. Customers were notified of required changes and brought their vehicles into the dealer, where the ECUs were physically replaced.

As new automotive features were introduced, new software systems were developed. First came the engine controller, which optimizes car performance by reading sensors and sending controls to the car engine; the controller was invented to help cars meet Clean Air Act requirements. “The engine controller was really the inflection point where we started bringing in a lot of electronics,” Vangelov reflected.

Next, centralized body control modules appeared, which included new electronic features such as interior lights, air conditioning, and one-touch power windows, mirrors, and locks. These control modules replaced separate mechanical controllers for each accessory. ECUs were also central to anti-lock brakes, which monitor wheel speed and control the brake valves to make driving safer.

In the late 1990s and early 2000s, ECU processors started using flash memory, said Vangelov, and engineers developed several paths for updates to the operating system (OS) and applications, such as on-board diagnostics (OBD) and BDM or JTAG connection ports. Some ECUs had programmable pins on the BDM and JTAG ports, which raised security concerns and were eventually pulled from production, Vangelov noted.

During this time, designers were integrating more and more ECUs into car features and rapidly increasing their capabilities, requiring far more memory and processing power than ever before. A modern seat controller, for example, uses pulse counts and a memory switch to store a driver’s preferred seat position; steering wheels and interior temperature can similarly be controlled with ECUs.

As the applications and the OSs became more sophisticated, software code did, too. ECUs are now written in higher-level languages (such as C or C++), Vangelov observed, and require more memory and processing power. At the same time, ECUs decentralized some functions by leveraging the increasing network communication and processing from other computerized modules in a vehicle. This increased communication, however, meant that cars needed more internal bandwidth.

To meet the continually increasing demand for new features and the software to support them, the automotive industry created guidelines for three components: common library components, a common OS, and communication and diagnostic standards.

In the late 1990s and 2000s, Vangelov said, car makers began to allow mechanics to use diagnostic testing tools to install software updates. Small updates took a few minutes, but depending on how much software was in a car, some updates could take more than an hour. In order to shorten that process, car makers then introduced ECUs that could be updated wirelessly, a significant shift that opened many new opportunities.

THE CHALLENGE OF LIMITED NETWORKING CAPABILITIES

Today, limited networking capabilities are a significant challenge for Ford, Vangelov said. While FlexRay, automotive Ethernet, and MOST150 connections boast data speeds of 10, 100, and 140 megabits per second, respectively, those high-speed networks are not available for the OBD connector, so they can't be used to deliver a software update to a car's ECU via a diagnostic tool. Controller Area Network (CAN), the one most commonly used by Ford, Vangelov said, provides automotive software updates at about 500 kilobits per second.

In 1999, diagnostic capabilities were standardized through the International Organization for Standardization (ISO) 14230-Keyword Protocol 2000. Before that, diagnostics were disjointed and often proprietary, Vangelov noted. In 2013, Unified Diagnostic Services and ISO 15765 developed and defined diagnostic services for software update deliveries. Diagnostic tools are the primary method for delivering the majority of the ECU software updates, Vangelov said, although there are several other methods, including CAN, USB (mostly for updating the "infotainment" systems), Wi-Fi, cellular, and Ford's SYNC 3 system. Vangelov noted that Ford has been using Wi-Fi to synchronize system updates with its assembly plants since 2010, which helps reduce the need for immediate or complex updates after a car has been purchased.

OTHER CHALLENGES TO SOFTWARE DELIVERY

Size is a growing challenge for delivering software updates in the automotive industry, Vangelov said. A car today can have more than 10 million lines of code, compared to about 50,000 or so lines in the 1980s. Updates now often require multiple decentralized ECUs to be updated at the same time, which requires coordination and increases the size of the software delivery. This, in turn, means more time and bandwidth are required.

That growth shows no signs of stopping. Software size and complexity will continue to increase as advanced driver assistance systems, autonomous vehicle mapping data, connectivity applications, and other current capabilities continue to evolve.

Another challenge is software integration. The operating systems, their services, and the application logic are often not optimized for software deliveries. As a result, Vangelov said, "If I have to deliver an update to a module, I'm updating everything." One opportunity for the industry to improve update deliveries would be to structure the architectures in a way that allows updating of single ECUs, he said. This would

drastically reduce the time it takes to deliver updates, without losing any functionality of other components.

Steven Lipner, an independent consultant, asked Vangelov to delve deeper into this problem in the discussion. Vangelov said that some OS components are better partitioned, but other systems have a conglomerated binary that requires coordination for updates. Also, software updates themselves are often structured in a way that only allows delivery of the entire package at once, which takes a long time. Ideally, each specific component's software could be partitioned, Vangelov said, but unfortunately ECUs are not written that way, and even if they were, that could create a different problem because multiple parts and subparts would all need to be individually managed and updated across a distributed system of ECUs.

A car's power supply presents additional challenges. As computerized as cars have become, the vast majority are powered by a lead acid battery and internal combustion engine, a power supply system that is simply not optimized for long software updates without additional charging. Very large updates could risk draining the battery, which could erase the module being updated and even deprogram some of the car's ECUs, making them inoperable, Vangelov said.

DEALING WITH LEGACY PRODUCTS

As in many industries, legacy products pose another software challenge for automotive ECUs. They are not typically included in verification and validation cycles done for new product releases and so must be pursued in a different development track, Vangelov noted. A further complication is that any update issued to ECUs in older vehicles must be verified to meet requirements for cars on the road today, which could be different from the requirements for which the systems were initially designed years ago. In addition to the development challenges this raises, software updates extensive enough to meet new requirements can sometimes exceed the capabilities or the memory available on the original system, making it essentially impossible to implement the fix, he said.

The technology supporting software update delivery is also constantly changing. Companies take away old products or services in order to free up space for new product offerings. For example, he noted, AT&T recently retired its 2G network, and customers with 2G-connected phones can no longer receive updates. It may be only a matter of time before 3G or 4G, which some cars use for software updates, meet a similar fate. Even before technologies become obsolete, regional differences can undermine their utility for delivering software updates reliably. Globally and even domestically, cellular

service can be inconsistent or intermittent, which can affect update speed, time, or even whether Ford can connect to a vehicle at all.

User experience is another important factor, and the challenges for the automotive industry are in some ways reflective of those seen in the technology industry more broadly. While “tech guys” like Vangelov are interested in the latest and greatest software capabilities, many average users just want their devices to work, without having to know how it works or navigate cumbersome solutions, he said.

One important need is to better communicate the need for software updates. “We really need to focus on how we drive the expectations for software updates to our customers,” Vangelov said. Other changes could help make the update process more palatable for drivers. For example, many devices have to be offline while they are updated, which means a car can’t be driven at all. Longer update times also mean that certain accessory functions, like the radio, could be unavailable during drives. Minimizing the update time would improve the update experience for customers.

In the discussion, Will Drewry, Google, Inc., followed up on this point, asking whether doubling the amount of storage would help to improve the user experience of updates by allowing the car to operate the old system while the new one is being installed. Vangelov said that Ford’s new SYNC 3 system uses this approach to run updates in the background while all of the car’s functions remain in use.

SECURITY CONCERNS

Forum Chair Fred Schneider asked Vangelov to speak to the issue of security, inquiring specifically about whether it is possible to “brick” a car. Vangelov explained that when a car gets an update, it’s not an update for “an entire computer” but rather “a conglomeration of multiple ECUs.” In order to brick a vehicle, everything would have to be updated at the same time and the lead acid battery would be drained in the process. If, in the middle of updating certain critical components, such as the engine controller itself, the update were interrupted and it couldn’t resume operations, that would leave an opening, he said. Ford is aware of this possibility and is looking for ways to strengthen security—not just for safety-critical components, but ensuring that the update mechanism is robust against all potential failure modes for all ECU modules.

Vangelov emphasized that update security is a primary aspect of his work at Ford. For example, Ford does verify keys on ECUs and signs binaries that are delivered over the air. Before a software update can be downloaded, a user must first go through several steps that determine both an update’s validity and the validity of the vehicle and user.

Tadayoshi Kohno, University of Washington, asked if Ford was concerned about the potential impacts of aftermarket components such as stereos or replacement transmissions in terms of software compatibility or security. Vangelov agreed that this phenomenon poses a problem, irrespective of how software updates are delivered. Before any updates are delivered, the system is designed to conduct an interrogation of all a vehicle's parts, which should help determine whether a new component could compromise an update, or if it will react safely and as expected. "If the interrogation results in something that seems foreign for that particular vehicle, it's not a build that we recognize, we will actually not deliver the update because we don't have a deterministic understanding of what the expectation would be if you were to deliver it downstream," said Vangelov.

Peter Swire, Georgia Institute of Technology, asked Vangelov to explain whether the connections among different ECUs in a car could allow a hacker to get further into the car's controls after breaching one component. Vangelov explained that while the ECUs do interact, that interaction is at the networking signal communication level. For example, an anti-lock braking system (ABS) module can read certain speed information from the CAN bus. However, that doesn't mean the ABS module can *alter* what is on the CAN bus, and so if the ABS module were being updated, the update couldn't affect the CAN bus, because it's not involved in the update. If, however, the update was meant to fix a communication problem with the ABS module reading the CAN bus data, then the two systems would have to be fixed together.

Swire asked if there were one main component of a car, central to its operation and connected to all the other accessory components, whose software, if hacked, could allow the car to be turned into a weapon. Vangelov replied that there are some systems, particularly those related to powertrain systems or the chassis, that are obviously essential to a car's operation. Some models have these systems segregated and some don't, and Vangelov noted that he could see a situation "potentially go awry if there were a dependency between the two and they weren't both updated at the same time."

The NIST Perspective on Software Updates

Paul E. Black and Lee Badger, National Institute of Standards and Technology

Paul E. Black and Lee Badger serve at the National Institute of Standards and Technology (NIST), the non-regulatory laboratory and agency of the U.S. Department of Commerce that advances standards in the information technology sector. Black is in NIST's Software Quality Group and Badger is in the Security Components and Mechanisms Group. Black spoke about software updates from the agency's perspective, and both fielded questions in the discussion period.

KEY CHALLENGES

Black opened with a discussion of the multifaceted challenges to creating software update infrastructure.

Processing power is one important factor, Black said, because speed and capabilities vary greatly among chips. Some, such as RFID chips, have very little power for updates, while others have massive computing power that larger updates require. In addition, some chips, such as read-only memory chips, have to be physically removed in order to be updated, while Internet-connected chips can be updated wirelessly. For those updates that involve connectivity, network availability can also become an issue. For

phones, cellular connectivity can be assumed, while other devices might be infrequently connected or might need to be updated in areas without reliable cellular service.

As noted in other workshop sessions, the user can be a factor, too, Black noted. Most users do not have the expertise to understand what exactly a security update will do, what to install, and what to avoid.

Software incompatibility is a key concern that can have security implications, inadvertently leading to bugs, crashes, or even system failure. For example, there could be a downstream, isolated component that relies on a format that wasn't included in a new update. A software update could even introduce new vulnerabilities to a system that had previously been secure, such as by creating vulnerabilities when two previously secure modules interact in a new way. Program shepherding, which monitors control-

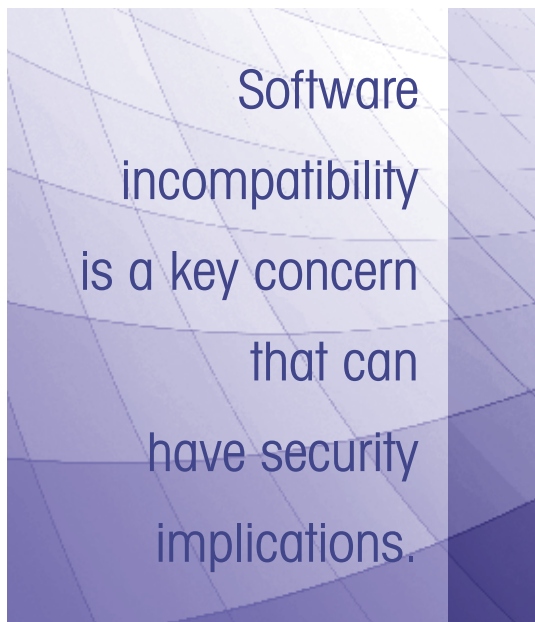
flow transfers during program execution, can become an important mechanism to monitor for potential intrusions during an update, as well, he said.

As noted throughout the workshop, software updates can also alert attackers to vulnerabilities. For example, hackers might be able to compare the old software to the updated version, identify the vulnerability being patched, and attack private information on systems that haven't been updated yet.

Black suggested that there may also be some overlooked challenges surrounding updates to newer systems like virtual machines, containers, or microservers. Although these are software packages, they are handled differently than typical software. Virtual machines, for example, can execute multiple operating

system configurations, depending on their use, and would need all the corresponding software updates to remain secure.

Building on a theme raised earlier in the workshop, Richard Danzig, Johns Hopkins University Applied Physics Laboratory, asked how the growth of cloud computing and machine learning might be affecting this landscape, in NIST's view. Black responded that the cloud, in one sense, is just another environment for operations, and in fact, the isolation of cloud systems make them easier to deal with in some respects. Another nuance is that the cloud business model is founded on offering up-to-date, secure, well-configured computing. This model turns operations from a business expense into a profit-making center, which could potentially work in users' favor because there is perhaps more of an incentive for companies to put resources behind frequent, high-quality updates, Black suggested.



Machine learning (or artificial intelligence) could be used to support security in a number of ways, Black said. For example, these technologies could be used to create more intelligent mechanisms for identifying vulnerabilities, monitoring intrusions, and responding to breaches. Black said his group is experimenting with Google's open-source software library, TensorFlow, to recognize software vulnerabilities. To be successful, we must learn from both good and bad experiences with applying machine learning in this context, he said.

Another challenge in the current landscape that is sometimes overlooked, Black noted, is ephemeral code. A lot of software is written in JavaScript, which is itself constantly being updated, and it's difficult to know which version of JavaScript certain code is relying on. NIST doesn't have an answer to this problem yet, Black said, but "we're raising the problem so that smarter people can think about it."

Finally, the relationship between software vendors and users can also be problematic for appropriate use and updating of software. Most software is licensed to a user instead of sold outright, in order to protect the maker's intellectual property rights. However, the licensing agreement doesn't set forth security requirements for the user. This licensing arrangement thus gives software makers "the best of both worlds," with fewer protections for the consumer, Black said.

SOLUTIONS BEING PURSUED

Black presented some possible solutions to address some of these key challenges.

One idea NIST is exploring to keep hackers from reverse-engineering patches is a "one-way function," he said. He introduced a hypothetical scenario with a software component, "S," that has an input vulnerability, "V." Typically a patch would be designed to detect a V input and filter it out but otherwise run S as usual. But this approach reveals to hackers that V is the input that causes the vulnerability. An alternative idea is to create a patch that uses a one-way function on the input, so instead of filtering out V inputs, it would be detected with a hash signature and compared to a given value (or range of values). "Because it's one-way, it's infeasible to reverse-engineer that and figure out what the vulnerability is from that patch," Black explained. If such one-way functions were feasible, this type of patch would allow a system to recognize V and reject it without broadcasting its vulnerabilities.

Because one-way functions tend to require a lot of computation, Black said it would be ideal to issue a code fix at some point, but that could be done in a way that separates the patch information from the code change and delivers them at different

times. This could help address the challenge of sending an update quickly enough to prevent hackers from reverse-engineering an exploit.

Regarding the idea of a software inventory, suggested earlier in the workshop, Black suggested that such a mechanism would indeed be helpful in shedding light on the building blocks inside complicated software modules, and he noted that NIST is working on enabling a software identification tag (SWID) that could be complementary to this approach. SWID is an International Organization for Standardization (ISO) standard that gives every version of software a unique ID. “It doesn’t solve problems,” Black said, “but at least this way there’s an ISO standard to communicate the information.”

SWID tags could create an inventory of all the software on a system, which would help users know, for example, how many different versions of a piece of software are running or which software needs to be updated first. Facilitating automated updates like this would also help control the costs of software updates, Black suggested.

Circling back to this notion of software transparency in the discussion, Bob Blakley, CitiGroup, noted that containers, complete systems that contain everything software needs to run and can be easily installed on servers, could help advance transparency because they are discrete parts with known materials, making it easier to see and address any vulnerabilities discovered within them. He suggested that more of these container-based models could meet the need for a “bill of materials.”

Black also addressed the issue of configuration updates, which are more complicated than software updates. Even if the system is properly configured for the old software, an update can introduce new parameters that essentially wipe the old configuration or create a potential mismatch between the old configuration and the new parameters.

In the discussion, Tony Sager, Center for Internet Security, noted that in his experience with the Department of Defense (DOD), configuration was a major consideration when evaluating the need for updates. If a system’s configuration eliminated the bug that the software update was trying to address, then there might be no need for the update in a particular situation. “Getting that right is hard to do,” Sager said, “but it really is, I think, a key part of the decision.”

Configuration updates typically come with specific instructions from the vendors, who use can NIST’s checklist program to guide what to update, how to communicate updates to users, and how to make sure the settings are correct. In some cases, NIST reviews the checklists, but in others, such as with OpenSSL, the process is centralized and users receive one update message. Checklists can sometimes be out of date, or they might be for updating software only, as opposed to the entire configuration, “but at least the information may get out there,” Black noted.

Another relevant NIST project is the security content automation protocol (SCAP), which is also meant to automate updates. SCAP verifies system details (such as which versions, privileges, or daemons are running) in advance, so that updates run more smoothly and only change what each particular configuration requires.

Prompted by a question from Sager, Black elaborated on SCAP in the discussion. Sager wondered whether NIST faces a chicken-and-egg problem in trying to get industry adoption for this idea. Standard naming and enumerations could simplify the problem, while trying to update an unknown number of variations on a theme could make it a much more difficult problem.



NIST is seeking ways to encourage high-quality, rapid, and secure software development.

Black agreed that industry can be hesitant to adopt new tools, pointing to SWID tags, which have also created a chicken-and-egg problem, as one example. Very few software packages provide SWID tags, because there aren't a lot of SWID tools out there for them to use in the first place. To remedy this problem, NIST is producing tens of thousands of SWID tags for existing packages to get them into wider circulation.

To the question of industry adoption challenges for SCAP, Badger said that the antivirus sector has relatively good adoption rates, although it's been difficult to get wider adoption in the industry as a whole. One barrier is that SCAP is complex and has a considerable learning curve. Sager suggested this might

be a case in which buyers need to be educated to know what protections to demand, but the standard itself must be sufficiently mature to drive this demand. Buyers like DOD and the National Security Agency were able to ratchet up the demands in their contracts, which led to the creation of the Common Vulnerabilities and Exposures and Open Vulnerability and Assessment Language, for example, noted Sager. But when buyers and vendors lose momentum, the demand can wane.

NIST also is seeking ways to encourage and reward high-quality, rapid, and secure software development, Black said. For example, the agency is running a Secure Toolchain competition, in which it presents a problem and gives teams 1 day to develop a fix. Through many rounds of these challenges, more and better secure tools are created, which raises the security bar higher and higher.

Steven Lipner, an independent consultant, noted that such competitions can have downsides and lead to false conclusions if the lessons from successfully handling "toy problems" are extrapolated too far. Black agreed that this is a risk and said it is a factor

they have tried to account for. While these contests are not likely to draw out solutions that would apply to the development of a new OS, he noted, they are potentially very useful for creating better security solutions for smaller apps, which are often “slapped together” rapidly and often have significant security vulnerabilities.

A recent NIST report, *Dramatically Reducing Software Vulnerabilities*,¹ identifies specific technical methods such as proof-carrying code, well-analyzed frameworks, and potential update mismatches to improve software security, noted Black. Well-analyzed frameworks, for example, enable updates to insert small bits of code around a framework, instead of updating an entire piece, thus increasing the security of updates.

¹National Institute of Standards and Technology, 2016, *Dramatically Reducing Software Vulnerabilities*, November, <http://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8151.pdf>.

10

Protecting Consumers from Software Update Risks

Ruth Yodaiken, Federal Trade Commission

Ruth Yodaiken is a senior attorney in the Bureau of Consumer Protection at the Federal Trade Commission (FTC), the agency that protects consumers from unfair or deceptive business practices across all industries. In her talk, Yodaiken described the FTC's approach to consumer protections in the Internet of Things (IoT) sector, including software update challenges. She specified that she was speaking at the workshop for herself and not on behalf of the FTC.

The FTC has two primary methods of ensuring that consumers are treated fairly: law enforcement and policy. The agency takes a comprehensive view of unfair or deceptive business practices, such as misleading claims or outright fraud, that could open up consumers to risk, before considering crafting new policies or taking legal action.

The FTC is active in the technology industry in general, including the IoT landscape, said Yodaiken. But, the agency has not issued any blanket rules or requirements for security or update protocols in IoT devices. Rather, it has focused on the context that a particular business is operating in, recognizing that different devices and different ecosystems require very different update approaches.

Yodaiken said the main questions driving the FTC's work are the following: What are the benefits to the consumer? What is the risk of harm to the consumer? Can that risk be lessened? and Is a business taking reasonable measures to do that? In the IoT industry,

the main challenge, she said, is that “a consumer very often has no idea what is going on with their devices, [or] if their devices are compromised.”

Yodaiken noted that companies are responsible for taking reasonable measures to protect consumers from security risks, and if they don’t, the FTC can step in. In 2013, for example, the FTC alleged that TRENDnet’s baby monitors and video cameras lacked adequate security protections, leaving consumers vulnerable to privacy breaches. The FTC alleged that the company failed to encrypt credentials, provide adequate security training to its employees, or conduct vulnerability testing. TRENDnet settled the highly publicized case, but unfortunately, Yodaiken said, the FTC continues to see many of the same issues in today’s consumer products.

WORKING WITH BUSINESSES

In addition to its legal work, a big part of the FTC’s work is to educate businesses and consumers about security issues. Taking a close look at IoT devices, for example, the agency found that different businesses have different definitions of “basic security.”



The FTC encourages developers to design software with security in mind and plan for updates.

Some companies were extremely experienced and engaged in security issues, yet they still struggled to address vulnerabilities and deploy software updates in IoT devices, often for many of the reasons being discussed at the workshop. Other companies might have a less experienced security team or focus on product features at the expense of security. Other companies might not even consider security a priority. “All of these pieces become parts of the ecosystem and obviously create problems,” Yodaiken said.

When those problems occur, software updates are, of course, a key remedy. By initiating

conversations about security and providing educational materials, the FTC encourages developers to design software with security in mind and plan for updates.

In the discussion, Richard Danzig, Johns Hopkins University Applied Physics Laboratory, asked whether the FTC was considering using insurance companies to compel businesses to better secure their software products. Yodaiken replied that the FTC has had some internal discussions about incentives, but noted that the National Telecommunications and Information Administration’s Multi-Stakeholder Process on

Software Updates has a subgroup focusing on economic incentives that might be another appropriate forum for further discussing such an approach.

EDUCATING CONSUMERS

Educating consumers is also a key piece of the puzzle. As an example, to combat phishing and malware hacking, the FTC has tried to educate people not to download unfamiliar e-mail attachments, Yodaiken said. The IoT space poses some unique types of challenges. While the FTC does not want to shift the responsibility for security to the consumers, the agency is grappling with how to help consumers better understand the risks inherent to IoT devices. “There are a lot of consumers who don’t understand that their devices are connected to anything, they don’t really understand how the magic works once they’ve set it up,” Yodaiken said. Without this basic understanding, it’s difficult to convey the importance of keeping up with software updates.

Some companies prioritize security and deploy updates seamlessly behind the scenes, but unfortunately this is not always the case. A witness at a recent House Subcommittee on Energy cybersecurity hearing described some IoT devices as having “consumer-grade” security, by which he meant poor security. “That’s not how it should be,” Yodaiken said.

Asking consumers to keep up with security updates while also warning them of phishing and malware attacks—clearly both important factors for enhancing security—can sow confusion. Some companies are not in frequent contact with their customers, and so a user might receive a message about an update from an unfamiliar source. How could we expect the user to evaluate and trust that message in order to determine which notifications are valid and which could be dangerous?

Another problem is the “update fatigue” that can result when a user is constantly bombarded by update notifications, leading them to take notifications less seriously or avoid spending the time needed to install them and perhaps the need to reboot. The risk of dissuading users from installing updates rises when updates wind up changing device settings. “You like things the way they are, and the device seems to work anyway, so why would you do the update?” Yodaiken summarized.

In the discussion, Eric Grosse, an independent consultant, suggested that one opportunity for better educating consumers is for the FTC to counter common security myths or unhelpful advice that is often shared, such as “never click on a link in an e-mail.” Yodaiken agreed that dispelling myths could be a useful approach to complement consumer guidance.

TOWARD A NEW APPROACH

Yodaiken described a competition the FTC is holding as part of its efforts to empower users to access needed updates and protect their security, the IoT Home Inspector Challenge. The competition is designed to spur the creation of a centralized tool for consumers that enables them to better understand the workings—and vulnerabilities—of IoT devices in their homes. The exact form such a tool would take is unknown, and the FTC is open to all ideas, but the crux is that the tool would free the consumer from complicated detective work and help people protect themselves against IoT-based vulnerabilities and breaches.

Of course, to avoid creating “a gift for hackers,” security is a foremost concern of the tool’s structure. “We don’t want to create a conduit or a list of all the devices you have with vulnerabilities,” Yodaiken emphasized.

There are many questions that such a tool would need to address, such as the following: What devices are in a home? What are their software components? What *should* be their software components (e.g., are they running the latest versions)?, and How are updates facilitated?

The basic goal is to find a way to empower consumers to identify the vulnerabilities they have and take the steps necessary to mitigate them. While the contest is focused on software updates to IoT devices, she said, submissions may also address other security challenges, such as the use of default passwords, privacy issues, or updating of separate components.

In the discussion, Danzig asked about whether the Underwriters Laboratories (UL) project might offer another mechanism for this type of consumer empowerment. Yodaiken replied that the FTC is aware of UL and its effort to review and rate consumer software products. That is promising for the future, Yodaiken agreed, but the FTC is focused more on consumers who have already purchased devices and are struggling with them right now.

11

Discussion

The workshop concluded with an overarching reflection on the workshop by Deirdre Mulligan, University of California, Berkeley, followed by a short open discussion session.

Despite the clear need for an update infrastructure to support security throughout the software life cycle, the workshop underscored just how complex that proposition is, Mulligan said. The presentations demonstrated a great deal of variation in the approaches being used as well as the perceptions of what's possible, from both a technical and a managerial perspective. There also was a great deal of variation in how different people and organizations perceive what they do and do not have control over, as well as how the external landscape influences their decision-making.

Mulligan reflected that she was surprised to learn how little overlap there seems to be in the types of software update challenges faced by different sectors. While some challenges are universal, the list of challenges seemed to grow longer and longer with each presentation.

The workshop surfaced some important differences in terms of the technical factors and the business factors at play in various types of situations, she noted, highlighting, as an example, the differences between Linux and Microsoft in terms of how updates are created and deployed. While open-source development allows for a lot of innovation and iteration, staging and rolling out updates in a closed system, such as Microsoft, has its own distinct advantages. Also, working in a business environment that can be

tightly controlled and producing single-purpose products can also bring advantages, as exemplified by the experience of Schweitzer Engineering Laboratories, Inc. (SEL). Mulligan suggested that makers of other single-purpose devices, such as perhaps those making medical devices, could learn from the SEL approach.

Mulligan highlighted the idea that if actual software updates to a particular device aren't feasible, then perhaps signatures, firewalls, patches, or other mechanisms that affect other parts of the system could be used to help limit attacks or the damage they can cause. Alternatively, at the other end of the spectrum, if a device or software can't be updated, it is useful to explore ways for it to be disabled or at least disconnected

If a device can't be updated, explore ways to limit the damage of a potential breach while retaining core functionality.

from the network to limit the damage of a potential breach while retaining the device's core functionality. For example, a "gatekeeper" could contain vulnerable software and compel a user to patch it, giving people both an education and an incentive to attend to updates while adding a layer of protection. While "cutting off" a device could frustrate users, it might be necessary to craft a cybersecurity policy that defines the situations in which doing so is justified in order to protect the greater society as whole, Mulligan suggested.

The appropriate timeline for supporting a product once it is sold, an issue that arose on multiple occasions, is also an important factor to consider, Mulligan reflected. She noted, for example, that it is useful to hear that Microsoft generally supports updates for 10

years on its operating systems, and Cisco supports updates for 5 years on its routers. It would be helpful to know how companies come to those decisions and how that could be applied in other markets, she suggested. While it's not reasonable to expect everything to be maintained for 20 years, like SEL's products are, some kind of guidance or transparency around these support timelines for products could be very helpful for consumers, consumer advocates, and decision makers.

Mulligan also highlighted the discussion around software inventories and other technical solutions that would allow users to assess the state of their software stack and determine whether updates are needed, a vein that the National Institute of Standards and Technology (NIST) is supporting with its Software Identification (SWID) project. This idea, she suggested, could be very helpful for both businesses and customers. In addition, another important factor that arose multiple times is that updates are often released in a way that can alert hackers to the vulnerabilities they are fixing, underscoring the need to think about ways to mitigate that.

One aspect not covered very extensively in the workshop was the process by which companies discover software vulnerabilities and what influences their update process. Mulligan suggested this topic could perhaps be the basis for a future workshop. Other important related issues that were only touched on briefly but that could warrant further discussion include privacy implications, future business models, new software trends such as ephemeral code, and the need for a mechanism to involve the public in advancing conversations about what the appropriate limits to monitoring might be, Mulligan concluded.

Launching the discussion session, Butler Lampson, Microsoft Corporation, suggested that the workshop focused too much on how software operated in the past, when it was downloaded and expected to perform one function. Today, he said, software is embedded into complex, cloud-based systems that run across multiple devices. “I think most of what was said at the very least would need to be rethought in a fairly serious way in order to meet this new reality,” he said. While he acknowledged that in the future, many devices will indeed operate on the model described in today’s presentations, “they’re just going to be an increasingly small fraction of everything that’s going on,” he emphasized.

Bob Blakley, CitiGroup, said he was struck by the lack of a “baseline threat model” on the part of manufacturers and consumers alike. If such a model existed, he suggested, “It might be more obvious why [software] should be designed properly for updates, and it also might be more obvious why you want to update them.” For example, it’s not obvious to a consumer why a network-connected light bulb might need a software update, but there are nonetheless potential threats. For example, a hacker could gain control of an individual light bulb and program it to flash in such a way to induce seizures. Or, large number of light bulbs could be controlled and used to originate a distributed denial-of-service (DDoS) attack. “But,” he cautioned, “in the absence of at least some notion of what each of these devices do, it’s hard [for consumers] to become psychologically fond of the idea that updating them is important.”

Afterword

Discussions at this workshop and among Forum on Cyber Resilience members during its planning and after the workshop make clear that software update is more complicated than it seems at first glance. This Afterword is offered to distill and share some of the insights shared during those discussions. Software update raises different sets of concerns, depending on who needs to address it. In particular, policymakers face different sorts of questions than engineering managers. Below are checklists of concerns that might arise for the two groups, based on points raised during the workshop. Additional detail can be found in the previous chapters.

First is the question of market mechanisms. This topic was not discussed at length during the workshop, but it merits careful consideration. Developers and engineers ought to plan for supporting updates to artifacts they create and respond to software update for building blocks they incorporate, because situations change and customers and/or their data may be put at risk. Questions enterprise purchasers could raise as they negotiate with vendors, especially with regard to Internet of Things (IoT) devices include the following:

- How would I know this device has failed or been compromised? What is the surprising worst that can happen if this device is successfully attacked?
- Can I afford to replace this device every few years?

- If not, what penalties ensure continued support? Is there escrow to a “software orphanage” in case support is abandoned, as discussed in Chapter 4? This device is just one piece of a network of many connected systems; how do I avoid endless finger pointing if it fails?
- Can I buy insurance, and what sorts of failures or breaches would it cover?

The rise of externalities from software that is not readily updateable, such as seems plausible now that IoT devices are surreptitiously used in denial of service attacks, makes software update not just something to leave to the market, but a public policy issue. The security of IoT is a public good.¹ What options are there to disincentivize lax maintenance by vendors, developers, and purchasers that could equitably reduce harm to innocent third parties? Below is a partial list of policy options that could lead to improvements:

- A vendor could clearly state how many years of security support come after the last sale, ensure that there is a viable economic model behind those promises, and ensure the appropriate entity is balancing risks of enabling software update versus not. Vendors could make clear to customers what, if any, updates are mandated.
- Information about installed software could help a purchaser or user have more information for assessing risk (e.g., scanning one’s own systems for instances of vulnerable libraries). Software identification (SWID) tags are one proposed approach for providing such information.
- The privacy implications of the update channel should be considered along with the security implications.
- Recognize what is reasonable to expect of a small start-up company and avoid blanket requirements. For instance, one could set a minimum number of devices or revenue before certain requirements or regulations become applicable.
- A well-managed cloud commonly makes updates easier because such an installation would likely have a large professional security staff, uniform hardware, and software control procedures already in place. The IoT and cyber-physical systems can lack these properties, and thus cannot take advantage of the security benefits that a well-managed cloud environment provides.

In addition to policy and regulatory challenges, there are numerous technical issues that engineering managers and developers need to consider with regard to software

¹D. Mulligan and F. Schneider, 2011, Doctrine for cybersecurity, *Dædalus, the Journal of the American Academy of Arts and Sciences* 140(4).

update. This list is intended to serve only as a starting point for those grappling with how best to implement and deploy software update capabilities.

- Software update is more complicated than it might seem at first. Managing it appropriately during the development process and life cycle of the device or system will take resources, so it will need a budget early.
- Updates should almost always be safe or trust could be destroyed, making user acceptance much harder to obtain. Secure development life cycle, test via tooling, daily builds, and employee use all help to improve safety. However, managing update dissemination is much more difficult if the customer or end user can pick and choose which updates to deploy. Synchronized global update is a best practice that helps keep the vulnerabilities, which are inevitably disclosed by updates, from becoming live exploits used against users who have not yet received the update. Significant bandwidth would be needed in most cases.
- The client update software “agent” that accepts and performs the update needs to be exceptionally robust and metaphorically paranoid. Ideally it would be linked to the boot process so an adversary cannot bypass updates undetected by just writing over disk image. The update agent best practices include the following:
 - Ensure that a power failure or other interruption of the updating process does not brick the device or system.
 - Design with the understanding that there are often only minutes, or even seconds, from initial network connection of a device to the first probing attacks.
 - Design to expect advanced attackers, and do not assume a firewall will provide sufficient protection.
- Consider designing a fail-safe mode for the product—for example, a mode that disables all communications except for update or perhaps is triggered automatically (after a warning period) if the product has not been updated.
- Observe trial users managing the updates, and design the process in ways that would make their lives easier, including the following:
 - Involve the product marketing team in persuading customers to accept updates and ensure that the process to accept updates is straightforward.

- Consider that bad actors will mimic your update screens to trick users into installing malware; ideally avoid the need to “pop-up” any screen at all, and instead rely on signed automatic updates.
 - Schedule updates to happen at convenient times.
 - Maintain user configuration choices and make intelligent extensions and do what you can to prevent updates from resulting in loss of user data—something that occurs occasionally today and motivates end users to disable automatic updates.
- Be sure that your own systems are getting updates from all your own suppliers, including open source. Avoid suppliers whose devices have proprietary drivers for which only limited or no future support is provided.
 - Updates themselves need to be done in a trustworthy manner. The best technical practices at the moment include the following:
 - Test thoroughly but swiftly, accept feedback, and do postmortems after deployment;
 - Provide quick-rollback mechanisms to mitigate the impact of failed updates;
 - Protect against subversion of the update channel;
 - Avoid using “security update” to advance other business interests; and
 - Carefully consider whether to separate security updates from feature updates. Such separation can be helpful to enterprise and other large users who are cautious about system reliability and perceive a need to verify that security updates will not have unexpected impacts; however, some suppliers, notably the Linux kernel, do not distinguish.
 - Very large proprietary firmware (such as the Unified Extensible Firmware Interface) is often problematic, even with support, because of the large attack surface unmitigated by compartmentalization.
 - Updates (and all boot-time executables) need to be digitally signed.² As demonstrated by Stuxnet and other malware, those signing keys are high-value targets and deserve extraordinary protective measures. It will also be important

²See the Forum on Cyber Resilience’s recent exploration of cryptographic agility and interoperability challenges in National Academies of Sciences, Engineering, and Medicine, 2017, *Cryptographic Agility and Interoperability: Proceedings of a Workshop*, The National Academies Press, Washington, D.C., <https://doi.org/10.17226/24636>.

to ensure rollback protection against active attackers, as discussed in Chapter 7. Design verifiably identical updates made to everyone to prevent targeted malware distribution.

We offer these thoughts as an additional personal perspective inspired by conversations with workshop participants and Forum members. Our thanks to them for their insightful discussions and comments on this important topic.

Eric Grosse, *Member*, Forum on Cyber Resilience

Fred B. Schneider, *Chair*, Forum on Cyber Resilience

Appendixes



Workshop Agenda and Participants List

WORKSHOP ON SOFTWARE UPDATE

February 6, 2017

Keck Center of the National Academies of Sciences, Engineering, and Medicine
Washington, D.C.

AGENDA

- | | |
|------------|---|
| 11:00 a.m. | Welcome and Overview
Fred B. Schneider, Forum Chair |
| 11:05 | Deirdre Mulligan, University of California, Berkeley |
| 11:35 | Kevin Fu, University of Michigan and Virta Labs |
| 12:05 p.m. | Working Lunch |
| 12:45 | Carlos Picoto, Microsoft Corporation |
| 1:15 | Nicko van Someren, Linux Foundation |
| 1:45 | Break |
| 2:00 | Ed Paradise, Cisco Systems, Inc. |
| 2:30 | Dave Whitehead and Edward Schweitzer,
Schweitzer Engineering Laboratories |
| 3:00 | Break |
| 3:15 | Will Drewry, Google, Inc. |
| 3:45 | John Vangelov, Ford Motor Company |
| 4:15 | Break |
| 4:30 | Paul E. Black and Lee Badger,
National Institute of Standards and Technology |
| 5:00 | Ruth Yodaiken, Federal Trade Commission |
| 5:15 | Wrap-up Discussion and Q&A
Moderator: Deirdre Mulligan |

PARTICIPANTS LIST

Lee Badger, National Institute of Standards and Technology
Nat Beuse, National Highway Traffic Safety Administration
Paul E. Black, National Institute of Standards and Technology
Bob Blakley, CitiGroup, Inc.
Arthur Carter, National Highway Traffic Safety Administration
Fred Cate, Indiana University
David Clark, Massachusetts Institute of Technology
Richard J. Danzing, Center for a New American Security
Donna Dodson, National Institute of Standards and Technology
Will Drewry, Google, Inc.
Kevin Fu, University of Michigan and Virta Labs
Eric Grosse, Independent Consultant
Cem Hatipoglu, National Highway Traffic Safety Administration
Robert Heilman, Department of Transportation/National Highway Traffic Safety Administration
David Hoffman, Intel Corporation
Nicole Hughes, Department of Defense
Paul Kocher, Cryptography Research, Inc.
Tadyoshi Kohno, University of Washington
Jim Kurose, National Science Foundation
Butler Lampson, Microsoft Corporation
Susan Landau, Worcester Polytechnic Institute
Steve Lipner, Independent Consultant
Brad Martin, National Security Agency
Deirdre Mulligan, University of California, Berkeley
Edward Paradise, Cisco Systems, Inc.
Carlos Picoto, Microsoft Corporation
Tony Sager, Center for Internet Security
Fred Schneider, Cornell University
Edmund Schweitzer, Schweitzer Engineering
Peter Swire, Georgia Institute of Technology
Nicko van Someren, Linux Foundation
Frank Taylor, Department of Defense
John Vangelov, Ford Motor Company
Kelly Vangelov, Ford Motor Company
Eric Wenger, Cisco Systems, Inc.
David Whitehead, Schweitzer Engineering
Ruth Yodaiken, Federal Trade Commission
Mary Ellen Zurko, Independent Cybersecurity Consultant

B

Steering Committee Biographies

FRED B. SCHNEIDER is the Samuel B. Eckert Professor of Computer Science at Cornell University and chair of the department. He joined Cornell's faculty in fall 1978, having completed a Ph.D. at Stony Brook University and a B.S. in engineering at Cornell in 1975. Schneider's research has always concerned various aspects of trustworthy systems—systems that will perform as expected, despite failures and attacks. Most recently, his interests have focused on system security. His work characterizing what policies can be enforced with various classes of defenses is widely cited, and it is seen as advancing the nascent science base for security. He is also engaged in research concerning legal and economic measures for improving system trustworthiness. Schneider was elected a fellow of the American Association for the Advancement of Science (AAAS, 1992), the Association of Computing Machinery (ACM, 1995), and the Institute of Electrical and Electronics Engineers (IEEE, 2008). He was named professor-at-large at the University of Tromsø (Norway) in 1996 and was awarded a doctor of science honoris causa by the University of Newcastle-upon-Tyne in 2003 for his work in computer dependability and security. He received the 2012 IEEE Emanuel R. Piore Award for "contributions to trustworthy computing through novel approaches to security, fault-tolerance, and formal methods for concurrent and distributed systems." The National Academy of Engineering (NAE) elected Schneider to membership in 2011, and the Norges Tekniske Vitenskapsakademi (Norwegian Academy of Technological Sciences) named him a foreign member in 2010. He is currently a member of the Naval Studies Board and the Computer Science and Telecommunications Board of the National Academies of Sciences, Engineering, and Medicine, and is founding chair of the National Academies' Forum on Cyber Resilience.

ANITA L. ALLEN is the Henry R. Silverman Professor of Law and Professor of Philosophy at the University of Pennsylvania Law School, where she is also the university's vice provost for faculty. She is an expert on privacy law, bioethics, and contemporary values, and is recognized for her scholarship about legal philosophy, women's rights, and race relations. In 2010 she was appointed by President Obama to the Presidential Commission for the Study of Bioethical Issues. Her books include *Unpopular Privacy: What Must We Hide* (2011); *Privacy Law and Society* (2011); *The New Ethics: A Guided Tour of the 21st Century Moral Landscape* (2004); *Why Privacy Isn't Everything: Feminist Reflections on*

Personal Accountability (2003); and *Uneasy Access: Privacy for Women in a Free Society* (1988). She co-edited (with Milton Regan) *Debating Democracy's Discontent* (1998). Allen, who has written more than a 100 scholarly articles, has also contributed to popular magazines and blogs and has frequently appeared on nationally broadcast television and radio programs. She has served on numerous editorial and advisory boards and on the boards of a number of local and national nonprofits and professional associations, including the Hastings Center, the Electronic Information Privacy Center, and the Bazelon Center for Mental Health Law.

ERIC GROSSE is a senior member of the Google security team and was previously vice president, Security & Privacy Engineering, at Google in Mountain View, California, leading a team of 512 who ensure systems and data stay safe and users' privacy remains secure. Improved and wider use of SSL, stronger consumer authentication technology, detection and blocking of espionage, transparency on legal requests for data, sophisticated malware analysis, tools and frameworks for safer building of Web applications are among the achievements of the Google Security Team. Before Google, Grosse was a research director and fellow at Lucent Bell Labs where he worked on security, networking, algorithms for approximation and visualization, software distribution, and scientific computing. He has a Ph.D. in computer science from Stanford University.

BUTLER W. LAMPSON is a technical fellow at Microsoft Corporation and an adjunct professor at the Massachusetts Institute of Technology (MIT). He has worked on computer architecture, local area networks, raster printers, page description languages, operating systems, remote procedure call, programming languages and their semantics, programming in the large, fault-tolerant computing, transaction processing, computer security, WYSIWYG editors, and tablet computers. He was one of the designers of the SDS 940 time-sharing system, the Alto personal distributed computing system, the Xerox 9700 laser printer, two-phase commit protocols, the Autonet LAN, the SPKI system for network security, the Microsoft Tablet PC software, the Microsoft Palladium high-assurance stack, and several programming languages. He received the ACM Software Systems Award in 1984 for his work on the Alto, the IEEE Computer Pioneer award in 1996, the von Neumann Medal in 2001, the Turing Award in 1992, and the NAE's Draper Prize in 2004. He is a member of the National Academy of Sciences and the NAE and a fellow of the ACM and the American Academy of Arts and Sciences.

SUSAN LANDAU is a professor of cybersecurity policy in the Department of Social Science and Policy Studies at Worcester Polytechnic Institute. Landau has been a senior

staff privacy analyst at Google, a Distinguished Engineer at Sun Microsystems, a faculty member at the University of Massachusetts, Amherst, and at Wesleyan University. She has held visiting positions at Harvard, Cornell, Yale, and the Mathematical Sciences Research Institute. Landau is the author of *Surveillance or Security? The Risks Posed by New Wiretapping Technologies* (2011) and co-author, with Whitfield Diffie, of *Privacy on the Line: The Politics of Wiretapping and Encryption* (1998, rev. ed. 2007). She has written numerous computer science and public policy papers and op-eds on cybersecurity and encryption policy and testified to Congress on the security risks of wiretapping and on cybersecurity activities at National Institute of Standards and Technology's Information Technology Laboratory. Landau currently serves on the Computer Science Telecommunications Board of the National Academies. A 2012 Guggenheim fellow, Landau was a 2010–2011 fellow at the Radcliffe Institute for Advanced Study, the recipient of the 2008 Women of Vision Social Impact Award, and a fellow of the AAAS and the ACM. She received her B.A. from Princeton University, her M.S. from Cornell University, and her Ph.D. from MIT.



Speaker Biographies

LEE BADGER is a computer scientist at the National Institute of Standards and Technology (NIST) and manages the Security Components and Mechanisms group in the Computer Security Division of NIST's Information Technology Laboratory. Badger has more than 20 years of experience with computer security research, with a focus on operating systems and access control. Prior to joining NIST in 2008, Badger served as a Defense Advanced Research Projects Agency (DARPA) program manager for 6 years, where he funded and managed a variety of programs focusing on self-regenerating systems, intrusion tolerance, self-defending applications, software security analysis, and software producibility. Prior to joining DARPA, Badger led development efforts culminating in implementations of Domain and Type Enforcement (DTE) for UNIX, a DTE-enforcing firewall, a Generic Software Wrappers system for UNIX, and application of software wrappers for intrusion detection. He holds an M.S. in computer science from the University of Maryland, College Park, awarded in 1987.

PAUL E. BLACK is a computer scientist in the Software Quality Group, Systems and Software Division, Information Technology Laboratory at NIST. Black has nearly 20 years of industrial experience in areas such as developing software for IC design and verification, assuring software quality, and managing business data processing. The website he began and edits at NIST, the online *Dictionary of Algorithms and Data Structures* (<http://www.nist.gov/dads>), was accessed almost 20,000 times a day from all over the world. Black began his Ph.D. at University of California, Berkeley, then transferred to Brigham Young University where he graduated in 1998. He has taught classes at Brigham Young University and Johns Hopkins University. Black has published in the areas of static analysis, software testing, software configuration control, networks and queuing analysis, formal methods, software verification, quantum computing, and computer forensics. He has been awarded the Department of Commerce Bronze Medal for leadership in the development of software assurance test methods and reference data and the ITL Outstanding Authorship Award in recognition of his publication, "Quantum Computer and Communications." He is a senior member of the Institute of Electrical and Electronics Engineers (IEEE) and a member of IEEE Computer Society and the Association for Computing Machinery (ACM).

WILL DREWRY is a principal software engineer at Google leading security efforts for Chrome OS and Pixel. Joining in 2003, his contributions have spanned initiatives ranging from large-scale infrastructure assurance to defining and driving security and privacy for consumer products. Starting in 2009, Drewry built and led security and privacy engineering for Chrome OS ensuring a holistic and coherent product design. Along the way, he designed and implemented key enabling technology (dm-verity, seccomp-bpf) now in use across the industry. Drewry received his B.A. in computer science from Boston University and holds more than 20 patents.

KEVIN FU is an associate professor of computer science and engineering at the University of Michigan, where he conducts research on computer security and health care as part of the National Science Foundation's (NSF's) Trustworthy Health and Wellness (THAW.org) Frontiers project. He also directs the Archimedes Center for Medical Device Security, whose mission is to improve medical device security through research and education, and he co-founded Virta Labs, a health-care cybersecurity company based in Ann Arbor, Michigan. Over the last decade, Fu has given nearly 100 invited talks on medical device security to industry, government, and academia—including U.S. Senate and U.S. House hearings, the Institute of Medicine, and National Academy of Engineering (NAE) events. Beginning with his 2006 security seminar at Food and Drug Administration CDRH, Fu's medical device security efforts were recognized with a Fed100 Award, Sloan Research Fellowship, NSF CAREER Award, the Massachusetts Institute of Technology (MIT) TR35 Innovator of the Year Award, and best paper awards on medical device security by organizations such as IEEE and ACM. Fu earned a Ph.D., master's degree, and bachelor's degree from MIT.

ED PARADISE is the vice president of engineering for the Security and Trust Organization. He is also the site executive for Cisco's Research Triangle Park Site. As vice president of engineering, Paradise is responsible for the security and trustworthiness of Cisco's product portfolio. He leads the engineering team that focuses on developing trustworthy systems by developing new tools, processes, and technologies that further enhance the security of Cisco product portfolios. Paradise joined Cisco in 1993. He has held various leadership positions in Cisco's engineering organization, including general manager of the Mobile Wireless Group and general manager of the IP Communication Business Unit. He also became the RTP site executive in 2002. In this role, he leads the second-largest U.S. Cisco site, also being the liaison between Cisco and the local community, including local and state governments, and its philanthropic interests. Paradise holds an M.S. in electrical engineering from Syracuse University and a B.S. in electrical engineering from the University of Hartford.

CARLOS PICOTO is an engineering general manager at Microsoft. He started his career with Microsoft in 1997. He received his M.S. in computer science from the Instituto Superior Técnico and his B.S. in computer science from the Universidade de Lisboa.

EDMUND O. SCHWEITZER III is recognized as a pioneer in digital protection and holds the grade of fellow in the IEEE, a title bestowed on less than 1 percent of IEEE members. In 2002, he was elected as a member of the NAE. Schweitzer received the 2012 Medal in Power Engineering, the highest award given by IEEE, for his leadership in revolutionizing the performance of electrical power systems with computer-based protection and control equipment. Schweitzer is the recipient of the Regents' Distinguished Alumnus Award and Graduate Alumni Achievement Award from Washington State University and the Purdue University Outstanding Electrical and Computer Engineer Award. He has also been awarded honorary doctorates from both the Universidad Autónoma de Nuevo León, in Monterrey, Mexico, and the Universidad Autónoma de San Luis Potosí, in San Luis Potosí, Mexico, for his contributions to the development of electric power systems worldwide. He has written dozens of technical papers in the areas of digital relay design and reliability and holds nearly 180 patents worldwide pertaining to electric power system protection, metering, monitoring, and control. Schweitzer received his bachelor's and master's degrees in electrical engineering from Purdue University and his doctorate from Washington State University. He served on the electrical engineering faculties of Ohio University and Washington State University, and in 1982, he founded Schweitzer Engineering Laboratories, Inc. (SEL), to develop and manufacture digital protective relays and related products and services.

JOHN VANGELOV is the embedded modem features manager at Ford Motor Company. He began his career at Ford in 2002, previously holding positions at Siemens, Lear Corporation, and AirTouch Cellular. Vangelov's experience includes leading the production delivery of vehicle ECU software; the development of embedded operating systems, diagnostic kernels, and software bootloaders; vehicle electrical system and architecture design; RF engineering; and the development and delivery of cloud-based services for software update and consumer features for connected vehicles. He has 16 patents issued or in process in the domain of software updates. He received his M.S. in software engineering from the University of Michigan and his B.S. in electrical engineering from Lawrence Technological University.

NICKO VAN SOMEREN is the Linux Foundation's chief technology officer (CTO), focused on the Core Infrastructure Initiative and other security-focused efforts at the organization.

He has extensive experience across the security and networking industries. Most recently, he was the CTO of Good Technology, and prior to this he served as chief security architect at Juniper Networks, where he was responsible for the company's Network Security products. Before joining Juniper, he was founder and CTO of the security technology company nCipher Plc. Van Someren holds a doctorate and first-class degree in computer science from Cambridge University in the United Kingdom. He is a fellow of both the Royal Academy of Engineering and the British Computer Society.

DAVID E. WHITEHEAD, P.E., is the vice president of Research and Development at SEL. He also oversees the company's Government Services Division and serves on the board of directors. After joining SEL in 1994, Whitehead worked as a hardware engineer, research engineer, and chief engineer of the Government Services Division before taking on the leadership of SEL R&D in 2006. Whitehead has been a passionate driver of product and talent development at SEL, and has had a hand in the steady stream of inventions and innovations to come out of U.S. based-technology company. He currently holds 55 patents worldwide. Whitehead is a leader in utility and industrial control system cybersecurity. He has presented at conferences, testified before FERC, chairs the IEEE Power and Energy Society Substations C6 group that addresses serial cryptographic protocols, and has authored numerous papers on the topic. Whitehead received his B.S.E.E. from Washington State University in 1989 and his MSEE from Rensselaer Polytechnic Institute in 1994. He is a registered Professional Engineer in Michigan, New York, North Carolina, and Washington.

RUTH YODAIKEN is a data protection attorney in the Division of Privacy and Identity Protection at the Federal Trade Commission. Her work revolves around how entities comply with existing laws and protect information about individuals from misuse.

OTHER RECENT REPORTS OF THE COMPUTER SCIENCE AND TELECOMMUNICATIONS BOARD

Cryptographic Agility and Interoperability: Proceedings of a Workshop (2017)
Foundational Cybersecurity Research: Improving Science, Engineering, and Institutions (2017)
Information Technology and the U.S. Workforce: Where Are We and Where Do We Go from Here? (2017)

A 21st Century Cyber-Physical Systems Education (2016)
Continuing Innovation in Information Technology: Workshop Report (2016)
Data Breach Aftermath and Recovery for Individuals and Institutions: Proceedings of a Workshop (2016)
Exploring Encryption and Potential Mechanisms for Authorized Government Access to Plaintext:
Proceedings of a Workshop (2016)
Future Directions for NSF Advanced Computing Infrastructure to Support U.S. Science and Engineering in
2017-2020 (2016)
Privacy Research and Best Practices: Summary of a Workshop for the Intelligence Community (2016)

Bulk Collection of Signals Intelligence: Technical Options (2015)
Cybersecurity Dilemmas: Technology, Policy, and Incentives: Summary of Discussions at the 2014
Raymond and Beverly Sackler U.S.-U.K. Scientific Forum (2015)
Interim Report on 21st Century Cyber-Physical Systems Education (2015)
A Review of the Next Generation Air Transportation System: Implications and Importance of System
Architecture (2015)
Telecommunications Research and Engineering at the Communications Technology Laboratory of the
Department of Commerce: Meeting the Nation's Telecommunications Needs (2015)
Telecommunications Research and Engineering at the Institute for Telecommunication Sciences of the
Department of Commerce: Meeting the Nation's Telecommunications Needs (2015)

At the Nexus of Cybersecurity and Public Policy: Some Basic Concepts and Issues (2014)
Future Directions for NSF Advanced Computing Infrastructure to Support U.S. Science and Engineering in
2017-2020: An Interim Report (2014)

Geotargeted Alerts and Warnings: Report of a Workshop on Current Knowledge and Research Gaps (2013)
Professionalizing the Nation's Cybersecurity Workforce? Criteria for Future Decision-Making (2013)
Public Response to Alerts and Warnings Using Social Media: Summary of a Workshop on Current
Knowledge and Research Gaps (2013)

Computing Research for Sustainability (2012)
Continuing Innovation in Information Technology (2012)
The Safety Challenge and Promise of Automotive Electronics: Insights from Unintended Acceleration
(2012, with the Board on Energy and Environmental Systems and the Transportation Research Board)

Limited copies of CSTB reports are available free of charge from:
Computer Science and Telecommunications Board
Keck Center of the National Academies of Sciences, Engineering, and Medicine
500 Fifth Street, NW, Washington, DC 20001
(202) 334-2605/cstb@nas.edu
www.cstb.org

The National Academies of
SCIENCES • ENGINEERING • MEDICINE

The nation turns to the National Academies of Sciences, Engineering, and Medicine for independent, objective advice on issues that affect people's lives worldwide.

www.national-academies.org

ISBN-13: 978-0-309-46288-4
ISBN-10: 0-309-46288-6

