

login:

AUGUST 2014

VOL. 39, NO. 4



Security

☞ **USB Insecurity Exposed**

Peter C. Johnson

☞ **Security at CERN**

Dr. Stefan Lüders

☞ **Build Web Applications with Encrypted Data**

Raluca Ada Popa

☞ **Hostbased SSH**

Abe Singer

Columns

Practical Perl Tools: Zero Plus One

David N. Blank-Edelman

Python: Command Line Option Parsing

David Beazley

iVoyeur: Seven Habits for Successful Monitoring

Dave Josephsen

/dev/random: Heartbleed and Other Failures

Robert G. Ferrell

Conference Reports

NSDI '14: 11th USENIX Symposium on Networked Systems Design and Implementation

23rd USENIX Security Symposium

August 20–22, 2014, San Diego, CA, USA
www.usenix.org/sec14

Workshops Co-located with USENIX Security '14

EVT/WOTE '14: 2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections

August 18–19, 2014
www.usenix.org/evtwote14

USENIX Journal of Election Technology and Systems (JETS)

Published in conjunction with EVT/WOTE
www.usenix.org/jets

CSET '14: 7th Workshop on Cyber Security Experimentation and Test

August 18, 2014
www.usenix.org/cset14

3GSE '14: 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education

August 18, 2014
www.usenix.org/3gse14

FOCI '14: 4th USENIX Workshop on Free and Open Communications on the Internet

August 18, 2014
www.usenix.org/foci14

HotSec '14: 2014 USENIX Summit on Hot Topics in Security

August 19, 2014
www.usenix.org/hotsec14

HealthTech '14: 2014 USENIX Summit on Health Information Technologies

Safety, Security, Privacy, and Interoperability of Health Information Technologies

August 19, 2014
www.usenix.org/healthtech14

WOOT '14: 8th USENIX Workshop on Offensive Technologies

August 19, 2014
www.usenix.org/woot14

OSDI '14: 11th USENIX Symposium on Operating Systems Design and Implementation

October 6–8, 2014, Broomfield, CO, USA
www.usenix.org/osdi14

Co-located with OSDI '14 and taking place October 5, 2014

Diversity '14: 2014 Workshop on Supporting Diversity in Systems Research

www.usenix.org/diversity14

HotDep '14: 10th Workshop on Hot Topics in Dependable Systems

www.usenix.org/hotdep14

HotPower '14: 6th Workshop on Power-Aware Computing and Systems

www.usenix.org/hotpower14

INFLOW '14: 2nd Workshop on Interactions of NVM/Flash with Operating Systems and Workloads

www.usenix.org/inflow14

TRIOS '14: 2014 Conference on Timely Results in Operating Systems

www.usenix.org/trios14

LISA14

November 9–14, 2014, Seattle, WA, USA
www.usenix.org/lisa14

Co-located with LISA14:

URES '14 West: 2014 USENIX Release Engineering Summit West

November 10, 2014

SESA '14: 2014 USENIX Summit for Educators in System Administration

November 11, 2014
www.usenix.org/sesa14

FAST '15: 13th USENIX Conference on File and Storage Technologies

February 16–19, 2015, Santa Clara, CA, USA
www.usenix.org/fast15
Submissions due: September 23, 2014

HotOS XV: 15th Workshop on Hot Topics in Operating Systems

May 18–20, 2015, Kartause Ittingen, Switzerland
www.usenix.org/hotos15
Submissions due: January 11, 2015

USENIX ATC '15: USENIX Annual Technical Conference

July 8–10, 2015, Santa Clara, CA, USA

Co-located with ATC '15 and taking place July 6–7, 2015

HotCloud '15: 7th USENIX Workshop on Hot Topics in Cloud Computing

HotStorage '15: 7th USENIX Workshop on Hot Topics in Storage and File Systems



;login:

AUGUST 2014 VOL. 39, NO. 4

EDITORIAL

- 2 Musings** *Rik Farrow*

OPINION

- 6 Why Offensive Security Needs Engineering Textbooks:
Or, How to Avoid a Replay of “Crypto Wars” in Security Research**
Sergey Bratus, Iván Arce, Michael E. Locasto, and Stefano Zanero

SECURITY

- 12 How USB Does (and Doesn't) Work: A Security Perspective**
Peter C. Johnson
- 16 Computer Security at CERN** *Dr. Stefan Lüders*
- 22 Building Web Applications on Top of Encrypted Data Using Mylar**
Raluca Ada Popa, Emily Stark, Jonas Helfer, Steven Valdez, Nikolai Zeldovich, M. Frans Kaashoek, and Hari Balakrishnan
- 28 cTPM: A Cloud TPM for Cross-Device Trusted Applications**
Chen Chen, Himanshu Raj, Stefan Saroiu, and Alec Wolman
- 35 Interview with Steve Bellovin** *Rik Farrow*

DIVERSITY

- 40 CRA-W: Taking Action to Achieve Diversity in Computing Research** *Dilma Da Silva*

SYSADMIN

- 42 Hostbased SSH: A Better Alternative** *Abe Singer*
- 47 Challenges in Event Management** *Jason Patee*
- 50 /var/log/manager: When Technology Isn't the Cause of a Technical Problem** *Andy Seely*

COLUMNS

- 54 Practical Perl Tools: Zero Plus One** *David N. Blank-Edelman*
- 58 Command Line Option Parsing** *David Beazley*
- 62 iVoyeur: 7 Habits of Highly Effective Monitoring Systems**
Dave Josephsen
- 66 Almost Too Big to Fail** *Dan Geer and Joshua Corman*
- 69 /dev/random** *Robert G. Ferrell*

BOOKS

- 71 Book Reviews** *Rik Farrow and Mark Lamourine*

CONFERENCE REPORTS

- 73 NSDI '14: 11th USENIX Symposium on Networked Systems Design and Implementation**



EDITOR
Rik Farrow
rik@usenix.org

COPY EDITORS
Steve Gilmartin
Amber Ankerholz

PRODUCTION MANAGER
Michele Nelson

PRODUCTION
Arnold Gatilao
Jasmine Murcia

TYPESETTER
Star Type
startype@comcast.net

USENIX ASSOCIATION
2560 Ninth Street, Suite 215
Berkeley, California 94710
Phone: (510) 528-8649
FAX: (510) 548-5738

www.usenix.org

;login: is the official magazine of the USENIX Association. *;login:* (ISSN 1044-6397) is published bi-monthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

\$90 of each member's annual dues is for a subscription to *;login:*. Subscriptions for non-members are \$90 per year. Periodicals postage paid at Berkeley, CA, and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

©2014 USENIX Association
USENIX is a registered trademark of the USENIX Association. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. USENIX acknowledges all trademarks herein. Where those designations appear in this publication and USENIX is aware of a trademark claim, the designations have been printed in caps or initial caps.



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

I've often dreamed of presenting security as a visualization: images that could clearly convey the dangers represented by different levels of access. My visualization would work so well that even non-technical people would easily understand the relative risks of different attacks. Alas, my skills are lacking when it comes to designing images. But I can write.

Several of the articles found in this issue inspired me in this direction. During my interview with Steve Bellovin, he spoke of walls and gates, nice solid visual metaphors. Sergey Bratus (and others) wrote of the lack of well-defined terms for describing offensive technology, and I certainly agree: the terms we have are often abused and misunderstood. Pete Johnson provided the allegory of a knight being challenged by a gatekeeper before being granted access. No wonder I am thinking in Technicolor.

Beige

Of course, then there's beige, the color of the first IBM PC. These early workstations shared something with their still extant bigger cousins, the mainframes, in terms of access. Rather than a PC, picture a 1970s era mainframe. Got it? Okay, I bet you are visualizing men with pocket protectors and a woman in high heels standing in front of tape drives. The tape drives were much more impressive than the actual mainframes, which were mostly featureless cabinets, often beige or gray. My favorites included lots of blinking lights, including ones attached to memory address lines.

Computer security was equally easy to visualize in that era: physical walls. The mainframe was secured within a special room, and you needed to gain access to that room if you wanted to steal or modify the data, a lot of which was stored on those magnetic tapes. The same was true for PCs for many years, as these were all standalone devices. Not that some mainframes didn't have terminal communication concentrators for remote access, but getting to the data still meant that someone in the secured room would need to heed your request to mount a tape.

The Network

By the end of the '80s, the real era of networking was just beginning. We have to see beyond the walls and locked doors and be able to visualize access to computers in a completely different way. In this case, I always wanted to see something right out of Gibson's *Neuromancer*, where corporate computers were protected by industrial grade "ice": defenses that could, and had, killed intruders. Somehow, Gibson's metaphoric ice was quite visual for me and, I presume, most others who read *Neuromancer*.

But translating ice into something that actually corresponds nicely with the real world of TCP/IP was much more difficult. In that world, what you can *see* from the network are open or closed ports, and the *ice* may or may not be visible as firewalls, and later, intrusion detection systems.

Still, one could have a nice visual representation, in textual form, by using Fyodor's Nmap (nmap.org). As Nmap grew in features and capabilities, you could learn not just which ports were open, but what version of server software was running attached to a port, as well as

what operating system supported the software. As real power goes, Nmap still is an incredible tool for visualizing a target textually, but it falls short of Gibson's ice.

We could use (or abuse) Bellovin's walls and gates: Each server is represented by a wall, penetrated by gates that are the open ports. The gates are labeled with the name and version of the service that appears there. I once tried to get a firewall company I was advising to color-code services, from "green" (fairly safe) to "red" (never safe), but they demurred. As this is my vision, I will take another tack at labeling the gates: Ports for insecure services appear as screen doors, while ports for much more secure services look like bank vaults. Too bad that OpenSSL's bank vault door turned out to have a backdoor in it, while appearing quite impressive.

Virtual Walls

Within anything we think of as a computer these days, including smartphones, tablets, desktops, mainframes, and servers within clusters, we also have gates and walls. Steve said, in his interview, that "strong walls are something we're pretty good at ... [but] components have to talk to each other, which implies gates." I've railed for years about the walls we've inherited, since the earliest multiprocessing system designs, and won't go there this time. I will point out that the walls are memory management, used to isolate processes from one another, and various rings of privilege accorded to the operating system by CPU hardware. The most prosaic of these gates are system calls, which allow an unprivileged process to ask the kernel to perform work on the process's behalf. And, as our hardware became more powerful, the number of walls and gates increased as we added virtualization to both hardware and the software that runs on it.

Even here, a bit of visualization might still prove useful. The kernel is like a castle, with a single gate: There is just one way in and one way out, via this gate. Or is there? I'll have more to say about that later, but for now, imagine a castle with an impressive gate. Processes only virtually enter this gate, as the kernel carries out activities vicariously, that is, the proper incantation made at the gate results in the kernel completing some activity and then sharing the results with the process waiting outside the gate. And, while all processes must use the gate, the processes can only interact via the kernel, via the gate of the system call interface.

If you've followed me so far, you are standing outside a castle, among a throng of other busy and eager processes, many clamoring for attention from the gatekeeper. Now that our kernels are multithreaded, it's as if there are many gatekeepers as well, all doing their best to respond to requests so that the processes are not held up. And even if the processes want to communicate with each other, they still must talk to the gatekeeper.

Inside the castle of the kernel, all access is allowed. It is as if the kernel is imbued with a magical quality that provides this level of access—because the kernel has total access. The side effect of this access is that any mistake in the hugely complex kernel can result in sharing this all-powerful access with any evil coder with the right spell: a kernel exploit.

Also, not all processes are treated equally: Even services have their 1%. In the realms of Linux and UNIX, root-owned processes have increased privileges within the castle. In the Windows world, root gets replaced with sets of privileges, mimicking the world of DEC's VMS with both finer control and much more complexity. And although not everyone can be one of the elite, even mere users have resources that exploits can use to abuse or abscond with the user's private data.

Fuzzy Picture

But the castle gate isn't the only way in. I've already mentioned the network, where each open port is like another open gate, each with a completely different set of guards, composed of policy and implementation. Lots can go wrong here, but the main point to keep in mind is that while it might be nice to imagine our castle having only a single entry gate, that's a false image.

And then there are other openings in the wall. In a wonderful presentation, Bill Cheswick described classic castle designs, based on visits he had made to real castles in Europe. But Ches went beyond these descriptions, to the story of the castle that fell because the invaders used a small back door, the one used for convenience by the castle's defenders to visit the town outside.

In my visualization, convenient backdoors look very much like USB ports. Even more than the system call interface, the USB interface is very complicated as it involves both parsing responses to a protocol and running the device driver of the USB device's choice. We all know this attack vector has been used successfully already (Stuxnet), and these convenient backdoors, available to any local attacker, or one that can trick a user into inserting a USB device, make our castle wall look more like Swiss cheese. So much for policy controlled gates.

Personally, I think we need more walls within our castles. At the very least, the gates themselves need to be run within isolated regions, because they too are complicated enough to be exploited.

The Lineup

We begin this issue with an opinion piece by Sergey Bratus, Iván Arce, Michael Locasto, and Stefano Zanero. These men were disturbed by the creation of new laws to regulate the creation, sharing and use of offensive software. Because we have yet to clearly define what exactly we mean by offensive software, new laws, and ones yet to be written, are vague and overreaching. The authors argue for the creation of clearly defined language that

will make writing and talking about offensive software, including exploits and vulnerabilities, much clearer and more precise.

I asked Pete Johnson, who had a paper published earlier this year on USB insecurity, to write and explain what's wrong with USB. Pete does a very nice job of explaining how the USB protocol works, as well as how it fails, both through allegory and diagrams.

I'd heard that Stefan Lüders had made presentations about how they handle security at CERN, and I asked him to tell us about that. CERN works with thousands of staff, visitors, and external researchers, which certainly makes security a daunting affair with almost everyone bringing their own device (BYOD). CERN works with people to secure their own devices, as well as educate their owners, but CERN also keeps a stick handy so warnings of failed security cannot be ignored.

Raluca Popa and her co-authors rewrote their NSDI paper on how to secure content on Web servers using encryption. Their solution, in a nutshell, is to handle encryption within the users' Web browsers, moving it away from a Web server that can be subverted or subpoenaed. They have also devised a method that allows searching of stored data on the Web server without sharing keys or using homomorphic encryption.

Chen Chen and his colleagues also rewrote their NSDI paper, and explain how TPM 2.0 can be extended to work through clouds and shared devices. Ordinary TPM can only perform tasks, such as signing a hash or encryption using a stored private key, on the device where TPM is installed. By using a small extension to TPM 2.0, Chen et al. explain how TPM can be leveraged to make sharing encrypted data between devices and clouds work securely.

I decided to interview Steve Bellovin for this issue. Steve has been a figure at USENIX meetings since the UNIX User Group changed its name to USENIX. Steve has also become well known in security through his research, his firewalls book, RFCs, and public speaking. I uncover some of the back story behind many of these accomplishments.

Dilma Da Silva has written an introduction to the Computer Research Association's Committee on the Status of Women in Computing Research (CRA-W) group. CRA-W has done much to help women and minorities succeed in getting into graduate school, publishing, and advancing in their careers. And as Dilma points out, papers with a diverse group of authors tend to get cited more often, implying that the level of creativity and quality is often higher than other paper-writing groups.

Abe Singer volunteered to write about hostbased SSH, a technique he has been using for many years. Although hostbased SSH is not new, it is also often ignored, or at least unknown. Abe

explains how hostbased SSH works, why it is better than other techniques, and where it is best used.

Jason Pree writes about event management, a nice term for "handling communications when things go wrong." Instead of the usual way of having too many open lines of communication, which often results in miscommunication and duplicated effort, Pree describes his own group's progress in centralizing communication, documenting, and managing events. For those of you interested in DevOps, event management is an important part of DevOps and getting your process under control.

Andy Seely writes from a manager's perspective about fixing a perception problem: that a part of IT is someone else's problem. Andy actually describes solving a DevOps issue, something I finally recognize after having read *The Phoenix Project* (see my book review). Like the fictional VP of IT in that book, Andy steps in to first understand the problem with one group, get other groups who actually support this group to buy in, and then reorganize to make the changes official.

David Blank-Edelman writes the second of a two-part column about ZeroMQ, a modern message queuing system that simplifies communication between processes, whether on the same system or across a network.

Dave Beazley tackles parsing command line options in Python. Dave begins with a confession, then demonstrates what some of the popular Python modules can do to make parsing options easier.

Dave Josephsen follows a tradition of successful authors who describe the seven habits of successful somethings. Dave, no surprise, explains the seven habits of successful monitoring, starting by telling us that it's about the data, not the tools.

Dan Geer and Joshua Corman take on the myth of the many eyes. The theory has been that open source software should be safer than closed source, but recent discoveries in security-critical open source projects provide fodder for Geer and Corman's investigation.

Robert Ferrell rants about the wonders of various Web tags, including the "Do not track" tag. Along the way, he casts a keen eye on other (current in late May) Internet memes, including Tara the cat, and what it really means when the US indicts five Chinese for stealing IP using the Internet.

Mark Lamourine has tackled a book about understanding the theory of computation. I finally read *The Phoenix Project* and really gained a better understanding of DevOps (and more) from it. I also review a beginner's book on penetration testing that is quite good.

We close out this issue with the summaries from NSDI '14.

I've often visualized computer security in a way not so different from the way I did in these musings. In this alternate scheme, certain programs were red and all the rest were green. If you could trick the red programs into running the code of your choice or accessing resources they were never intended to access, you could imbue your exploit with the color red. The red programs were root-owned processes, set-user-id root programs, and the kernel. Everything else was green by comparison to the power of root, or comparatively privileged parts of Windows.

While we continue to heap praise upon those who manage the feat of separation of privilege (Venema and Bernstein), we keep building monolithic applications with no such separation. Unless we can actually learn how to become designers and programmers who can build carefully limited modules with clear interfaces, we really won't have much use for walls and gates.



Become a USENIX Supporter and Reach Your Target Audience

The USENIX Association welcomes industrial sponsorship and offers custom packages to help you promote your organization, programs, and products to our membership and conference attendees.

Whether you are interested in sales, recruiting top talent, or branding to a highly targeted audience, we offer key outreach for our sponsors. To learn more about becoming a USENIX Supporter, as well as our multiple conference sponsorship packages, please contact sponsorship@usenix.org.

Your support of the USENIX Association furthers our goal of fostering technical excellence and innovation in neutral forums. Sponsorship of USENIX keeps our conferences affordable for all and supports scholarships for students, equal representation of women and minorities in the computing research community, and the development of open source technology.

Learn more at:
www.usenix.org/supporter

Why Offensive Security Needs Engineering Textbooks

Or, How to Avoid a Replay of “Crypto Wars” in Security Research

SERGEY BRATUS, IVÁN ARCE, MICHAEL E. LOCASTO, AND STEFANO ZANERO



Sergey Bratus is a research associate professor of computer science at Dartmouth College. He sees state-of-the-art hacking as a distinct research and engineering discipline that, although not yet recognized as such, harbors deep insights into the nature of computing. He has a PhD in mathematics from Northeastern University and worked at BBN Technologies on natural language processing research before coming to Dartmouth. sergey@cs.dartmouth.edu



Iván Arce is director of security in the Information and Communications Technology (ICT) R&D program at Fundación Dr. Manuel

Sadosky, a mixed (public-private) non-profit organization in Buenos Aires, Argentina, whose goal is to promote stronger and closer interaction between industry, government, and academia in all aspects related to ICT. Arce is also the co-founder and former CTO of Core Security Technologies. ivan.w.arce@gmail.com



Dr. Michael E. Locasto is an assistant professor in the Computer Science Department at the University of Calgary.

He seeks to understand why it seems difficult to build secure, trustworthy systems and how we can get better at it. He graduated magna cum laude from The College of New Jersey (TCNJ) with a BSc degree in computer science. Dr. Locasto also holds an MSc and PhD from Columbia University. locasto@ucalgary.ca

Offensive security—or, in plain English, the practice of exploitation—has greatly enhanced our understanding of what it means for computers to be trustworthy. Having grown from hacker conventions that fit into a single room into a distinct engineering discipline in all but the name, offensive computing has so far been content with a jargon and an informal “hacker curriculum.” Now that it is unmistakably an industry, and an engineering specialization, it faces the challenge of defining itself as one, in a language that is understood beyond its own confines—most importantly, by makers of law and policy.

Currently, lawmakers and policy-makers have no choice but to operate with pieces of our professional jargon that have been publicized by journalists. But writing laws based on professional jargon is dangerous: This jargon will be misunderstood by lawmakers and judges alike. It’s not the wisdom of the judge or the legislator that is in question, it’s their ability to guess the course of a discipline years in advance.

Consider the concept of *unauthorized access* at the heart of (and criminalized by) the Computer Fraud and Abuse Act (CFAA). The unanticipated, “unauthorized” uses of today will be primary uses or business models of tomorrow. When CFAA was written, connecting to a computer on which one had no account was pointless. Cold-calling a server could serve no legitimate purpose, as no servers were meant for random members of the public; each computer had its relatively small and well-defined set of authorized users. Then the World Wide Web happened, and connecting to computers without any kind of prior authorization became not just the norm but also the foundation of all related business. Yet the law stands as written then, and now produces conundrums such as whether port scans, screen-scraping, or URL crafting are illegal, or even whether telling journalists of a successful URL-crafting trick that revealed their email addresses could be a felony (as in the recent *US v. Auernheimer* case). Even accessing your own data on a Web portal in a manner unforeseen by the portal operator—as in the case of ApplyYourself users who could see their admission status prematurely—may similarly be a crime under CFAA (for discussion of these cases and different institutions’ reactions to them, see [14]).

Lawmaking with regard to offensive security artifacts has already started. Article 6 of the Budapest Convention on Cybercrime requires signatories to issue laws that criminalize “*production, sale, procurement for use, import, distribution or otherwise making available of...a device, including a computer program, designed or adapted primarily for the purpose of committing any of the offences*” it established as criminal; Germany and UK have since enacted laws targeting so-called “hacking tools.” Although, to the best of our knowledge, no prosecution of security researchers has yet taken place under these laws, they have had nontrivial chilling effects. More recently, *intrusion software* has been categorized by the December 2013 Wassenaar Arrangement as dual use technology subject to exports control; such software is defined as capable of “*extraction of data or information, from a computer or network capable device, or the modification of system or user data or modification of the standard execution path of a program or process in order to allow the execution of externally provided instructions.*” This is, of course, what debuggers and hypervisors do, not to mention

Why Offensive Security Needs Engineering Textbooks



Stefano Zanero received a PhD in computer engineering from Politecnico di Milano, where he is currently an assistant professor. His research focuses on mobile malware, malware analysis, and systems security. He's a senior member of the IEEE, the ACM, and the ISSA, and sits on the Board of Governors of the Computer Society. stefano.zanero@polimi.it

all varieties of JTAGs; although the document further stipulates that “*Intrusion software’ does not include any...hypervisors, debuggers, or Software Reverse Engineering (SRE) tools;*” the above functional description fits them perfectly.

Such language demonstrates the challenge we face. As native speakers of the jargon, we understand that an exploit, a rootkit, and a defensive module that inserts itself into a piece of software are all likely to use the same technique of reliably composing their own code with the target’s; however, lawmakers do not see their unity.

Will jailbreaking or composition beyond well-defined APIs such as DLL injection survive these challenges? Many sufficiently advanced techniques in both defense and exploitation perform some of a debugger’s or linker’s tasks without being either debuggers or linkers; new debugging and dynamic linking techniques are informed by exploitation. For example, BlackIce Defender, the first Windows firewall, linked itself into the kernel by “modifying the standard execution path” to defend the system, and even patented the technique that many rootkits have since rediscovered; Robert Graham tells the story in “The Debate over Evil Code” [2]. “Bring Your Own Linker” has long been a composition pattern for both offense and defense [1].

Proposals for stricter regulation of exploits are not hard to come by. A good example is provided by Stockton and Golabek-Goldman [3], which makes an aggressive and ill-informed call for regulation (and spells \emptyset day with a symbol for “empty set”). It defines “weaponized” on its first page to mean “disrupt, disable, or destroy computer networks and their components” and then on the next page claims that “*Criminals buy and use weaponized \emptyset day exploits to steal passwords, intellectual property, and other data,*” even though disabling or destroying a compromised computer in order to steal passwords or secrets is counterproductive; in fact, it would be just plain stupid, as it would alert the victim of the breach and likely eliminate the value of stolen passwords or data. Apparent lack of familiarity with the field, however, doesn’t stop the authors from calling for prosecution of security researchers under the CFAA—a law so broad and vague that prominent legal scholars argue it should be void for vagueness [15].

If anything, we can expect more laws and regulations on the basic artifacts of our profession. The only way for us to avoid overly broad formulations that would snare every technique we use is to develop a language that puts offensive computing in perspective with other computer engineering.

In short, we need textbooks and textbook definitions that describe offensive computing so that policy-makers need neither puzzle over jargon nor design their own language—both approaches being potentially disastrous to the future state of practical computer security.

Why Offensive Computing Matters for Security in General

If you shame attack research, you misjudge its contribution. Offense and defense aren’t peers. Defense is offense’s child. —John Lambert [4]

Exploitation is programming. It is the kind of programming that every programmer should, if not directly practice, at least understand in terms of its capabilities and limits, because it will be practiced on his code. Our security is only as good as our understanding of this kind of programming, because it’s the essential nature of general-purpose systems (or perhaps of all rich enough computing systems) to allow a myriad of other execution paths than merely the intended ones. Until all possible latent, unintended execution models are understood, they can neither be eliminated nor triaged.

Security and trustworthiness of code means attackers’ inability to program it. In computer science theory, we emphasize results that show what can and cannot be programmed; in

Why Offensive Security Needs Engineering Textbooks

fact, our very notions of computer architectures derive from these results. Programmers and designers of a trusted system must be equally focused on what can and cannot be programmed on (or against) their code, no less than a theorist is concerned with what can and cannot be computed by particular execution models, type systems, automatic theorem provers, verifiers, and the like.

The strongest kind of trust in systems security, just as in cryptography, derives from some programs provably not existing—or at least from their existence being highly unlikely. Ciphers are only trusted because no efficient algorithms to solve certain algebraic problems are believed to exist. Cryptographic protocols are only deemed trustworthy when no sequence of attacker manipulations of their messages can interfere with their transactions, and so on.

To stress the role of anticipating and precluding attackers' programs in the realm of cryptographic protocols, Anderson and Needham call the protocol designers' task *programming Satan's computer*:

In effect, [the protocol designer's] task is to program a computer which gives answers which are subtly and maliciously wrong at the most inconvenient possible moment... and we hope that the lessons learned from programming Satan's computer may be helpful in tackling the more common problem of programming Murphy's. [5]

For applied systems tasks, the primitives of adversarial programming may be different, but the essence of trustworthiness is the same: Such attacker programming must fail, preferably due to the provable impossibility of certain tasks.

We can trust any system only so far as we understand its unintended programming models (so-called “weird machines” [6], building on prior work by many others, such as Gerardo Richarte's *About Exploits Writing* [7]) and their limits. Exploits are merely artifacts and expressions of this understanding; the essence of the discipline is the skill to discover, validate, and generalize such models. Yet no research activity can develop without free exchange of its artifacts, and the discipline of systems security needs to develop a lot further before we can trust it even to the same extent as we trust analysis of cryptographic protocols.

Exploits are the primary tools in exploring the unexpected, latent models of programming that are inherent in the ways we currently build computing systems. Thus, we must be able to speak about them in all their unity and differences, and to be understood.

Exploits: Research or Development? Proof-of-Concept or “Weaponized”?

Compared with software engineering, arguably its most closely related field, security focuses much less on its engineering process. Unlike software engineering, which continually invents new processes and methodologies, and has an industry-wide shared vocabulary for the outcomes of different process stages (such as “design,” “architecture,” “prototype,” “alpha-,” “beta-,” “production quality,” etc.), the security industry does not appear concerned with defining its process or its product through the stages of its development and maturity.

Terms occasionally used to qualify important industry artifacts, such as *exploits*, do not appear to have consensus definitions. Perhaps the best example is the use of “weaponized” [8] to refer to a certain grade of readiness or effectiveness (or ease-of-use?) that must inspire awe in the prospective buyer (note also how such use in turn affects misuse in policy proposals, as quoted above).

Even terms purely technical in origin raise questions regarding their usefulness, for example, the use of “memory corruption” in advisories [9]. Even the typically used term *remote code execution* is somewhat ambiguous, because it obscures whether introduction of external code by a remote party is necessary or whether full control is achievable by manipulating the platform's existing code, with remotely crafted data inputs acting as the de facto exploit program.

It gets worse when we get to characterizing intentions of a particular research or engineering activity. Suppose some lawmakers would like to protect security research results while attempting to curb what they see as software developed with ill intent. Our industry's language, however, lacks the ability to clearly distinguish research results from engineering artifacts. An in-depth technical description of a software vulnerability may or may not be equivalent to an actual exploit program that leverages said vulnerability. How much detail and analysis do you need to consider the two equivalent? Is it possible to regulate one but not the other? And, if so, to regulate what exactly?

Even though there is a lot of architecting, programming, and testing involved in producing what could be called a “commercial grade exploit”—all activities that can be more closely associated with software engineering than with research as such—this nuance seems to be lost on much of the security industry, and certainly on the outside world, which speaks of “vulnerabilities,” “PoCs,” “triggers,” “payloads,” and “weaponized exploits” as if they were interchangeable. Given such usage, the difference between an open source research tool and a commercially backed software product that includes exploits is too nuanced to explain (see, e.g., Iván Arce's RSA 2005 presentation [10] on the subject).

All the more so, a “textbook” gradation of exploits with respect to their power and reliability is necessary. As a direct consequence

Why Offensive Security Needs Engineering Textbooks

of such a gradation, an evaluation of effort necessary to elevate privilege from any given exploit achievement becomes desirable. In other words, it is not enough for a customer of an engineering effort to know that a product or design is flawed; one might want to know how deeply the rabbit hole goes.

In plain English, what does it mean for software to withstand a particular kind of adversarial audit or testing? Once a vulnerability has been found, how general is its description as presented in an advisory or an exploit? Does the description need to capture an entire class of related vulnerabilities or merely a particular instance of an exploitable bug? How far should an exploitable bug be pursued by the researcher beyond the creation of code that exploits a particular platform or platforms? How resilient is the exploit against defenses such as address space randomization, non-executable memory, various canaries, and other memory integrity checks? How resilient can it become after a man-month of engineering effort by the exploit developer, and how qualified should this developer be to pull it off?

For all of these, there appear to be neither accepted answers nor a common language to provide them. Our industry still lacks a consensus vocabulary to describe the generality of knowledge about a flaw as encapsulated in an exploit or an advisory. For example, has the primary effort been spent on the discovery of the flaw or on constructing the exploit machine? How likely is the flaw to be present and/or exploitable in other instances of related codebases? Is the exploitability of the flaw an (un)happy accident, or does it reveal a general principle applicable even beyond related codebases?

Most of these answers become clear to experts after a careful study of the exploit, but no textbook or other authoritative publication captures them, which makes it hard to explain the insights and the impact. Not surprisingly, it is often a hard task to explain the impact of an “attack paper” to academics not versed in exploitation, as they, too, lack the terms for different degrees of impact and generality and have no referent in industry language.

In short, a “Rainbow Series” for offensive computing suddenly sounds like a good idea.

Common Criteria or FIPS for Offensive Computing?

Contrast the lack of terms to describe the generality, the resiliency, or the reliability of an exploit with the well-known criteria for government procurement of trusted computing systems, such as the Common Criteria or the FIPS certifications. Their different levels enumerate processes and methodologies applied in development of the software, with those at higher levels expected to provide relatively stronger assurance. A ranking, however imperfect, of software construction and testing methodologies is

implied with respect to their relative power to provide assurance and verification.

A similar ranking of attack and assessment methodologies may be possible, with respect to their power to reveal flaws. The similarity would, of course, extend to the cautions and provisos that apply to software construction methods, namely, that their ranking is relative rather than absolute, and provides evidence of effort invested rather than proof of security in any given sense.

However, no such ranking is enshrined to date in a form available to industry outsiders. Some policy-makers may understand that certain grades and levels of offensive skills, activities, and artifacts are indispensable to security education of every computer professional. They may understand that major advances in computer security have been made by the “Citizen Science” of hacking and only then adopted by industry or academia, and that curbing this citizen science by turning the respective activities into legal minefields will shrink the talent pool of “cyberdefenders.” Yet, even so, they lack the concepts and terms to clearly distinguish activities they want regulated from the basic tools of the discipline.

Moreover, perhaps their very ideas of what they want regulated will be changed once a proper language that shows the relative importance of offensive activities is available.

Have We Learned the Lesson of the “Crypto Wars”?

The 1990s were a formative decade for the commercial Internet in the United States. Unfortunately, during this same time the US government policy was to treat strong encryption as a threat and to control implementations of certain cryptographic algorithms as munitions, subject to vigorous enforcement of export regulations. In 1993, the author of the original PGP software, Phil Zimmerman, became the target of an FBI investigation for munitions export without a license, which lasted until 1996. At the same time, a series of failed technological “solutions” and mandates, such as the backdoored-by-design Clipper chip [11] and third-party key escrow were promoted as a legally safe way for the telecommunications industry to implement compliant encryption—which would have essentially amounted to pretend security.

Export restrictions on artifacts of cryptography have doubtlessly harmed its practical progress. It’s not only that Johnny Q. Public still can’t encrypt [12], but John the Special Agent can’t encrypt either! [13] No matter where one stands on whether and how much the latter should be allowed to wiretap the former, John certainly has things to hide and in fact a duty to hide them—in which he is conspicuously failing.

Could it be that *both* of these failures are due to the fact that deployment of strong crypto was stymied just when today’s

Why Offensive Security Needs Engineering Textbooks

dominant communication protocols and infrastructure were rapidly developing? The fact is, these technologies ended up leaving crypto behind and matured without incorporating cryptography at their core. Superiors of John the Special Agent may have had visions of him using separate, special technologies vastly stronger than Johnny Q. Public's and obtained from sources untainted by the weaknesses of public commodity communications; it appears their vision was wishful thinking.

If having to pretend that poor cryptography was secure because practically exploring stronger crypto was a legal minefield led us to this point, where would pretending that computers are secure because of a likely minefield arising in exploitation engineering lead us from here? It will likely be worse, because the field of cryptography by the 1990s already had mature mathematical theory not easily undercut by the drag created on its engineering practice. Systems security, on the other hand, is only building up its theoretical foundations and is in need of much more feedback and generalization of its practice and its failures.

If the practice of exploring the programming of programs' faults becomes subject to regulation as vigorous as the 1990s "Crypto Wars," will this practice develop enough to warn us before unsecurable designs come to dominate critical infrastructure, power management, medicine, or even household appliances beyond any hope of replacement? Will we be surrounded by an Internet of Untrustworthy Things just as we are surrounded today by an Internet of Things that Can't Keep a Secret (or at least are no help to an ordinary person for doing so)?

Conclusions

Offensive computing—by now a research and engineering discipline that cuts across many technologies and abstraction layers—is central to the security and trustworthiness of computer systems. However, the further one stands from security research, the less prominent the role of offensive computing appears. Even in the eyes of traditionally trained computer scientists and engineers this role looks somewhat peripheral; in the view of policy-makers, offensive computing is often completely marginalized and confused with the criminality and ill intent of surveillance and repression.

These diverging views of offensive computing are a clear and present danger to the development of the discipline, and thus to our hope for improving the trustworthiness of everyday computing. Without a concerted effort to claim its place, offensive computing will end up being further marginalized, nearly impossible to practice outside of costly legal protection, and completely impossible to practice as a citizens' science.

To protect our discipline, we need to make sure that good approachable textbooks, or at least comprehensive dictionaries, exist for it, that put it into proper perspective not only to experts but also to a much broader audience. Distracting as the task of writing these books may be, failure to communicate the importance of offensive research will be a lot more damaging in the long run, both to all of us and to the society that our research ultimately serves to protect.

References

- [1] Bratus et al., “Composition Patterns of Hacking,” in *Proceedings of the 1st International Workshop on Cyber Patterns*, Abingdon, Oxfordshire, UK, July 2012, pp. 80–85.
- [2] <http://blog.erratasec.com/2013/03/the-debate-over-evil-code.html>.
- [3] Paul N. Stockton and Michele Golabek-Goldman, “Curbing the Market for Cyber Weapons,” *Yale Law & Policy Review* (December 2013): <http://www.sonecon.com/docs/studies/SSRN-id2364658.pdf>.
- [4] <https://twitter.com/JohnLaTwC/status/44276049111178240>.
- [5] www.cl.cam.ac.uk/~rja14/Papers/satan.pdf.
- [6] Bratus et al., “Exploit Programming,” *login*, vol. 36, no. 6 (December 2011): <http://langsec.org/papers/Bratus.pdf>.
- [7] Gerardo Richarte, “About Exploits Writing,” Core Security Technologies presentation, 2002: http://corelabs.coresecurity.com/index.php?module=Wiki&action=view&type=publication&name>About_Exploits_Writing.
- [8] Core Security Technologies, “Speaking the Language of IT Security”: <http://blog.coresecurity.com/2009/11/05/speaking-the-language-of-it-security/>.
- [9] Risk-Based Security, “Memory Corruption... And Why We Dislike that Term”: <http://www.riskbasedsecurity.com/2013/08/memory-corruption-and-why-we-dislike-that-term/>.
- [10] “On the Quality of Exploit Code: An Evaluation of Publicly Available Exploit Code”: http://corelabs.coresecurity.com/index.php?module=Wiki&action=view&type=publication&name=rsa2005_quality_of_exploit_code.
- [11] M. Blaze, “Protocol Failure in the Escrowed Encryption Standard,” *Proceedings of Second ACM Conference on Computer and Communications Security*, Fairfax, VA, November 1994.
- [12] A. Whitten and J. D. Tygar, “Why Johnny Can’t Encrypt,” *Proceedings of the 8th USENIX Security Symposium*, Washington, DC, 1999: https://www.usenix.org/legacy/events/sec99/full_papers/whitten/whitten.pdf.
- [13] S. Clark et al., “Why (Special Agent) Johnny (Still) Can’t Encrypt,” *USENIX Security Symposium*, 2011: https://www.usenix.org/legacy/event/sec11/tech/full_papers/Clark.pdf.
- [14] S. W. Smith, “Pretending that Systems Are Secure,” *IEEE Security and Privacy*, vol. 3, no. 6 (November/December 2005), pp. 73–76.
- [15] Orin S. Kerr, “Vagueness Challenges to the Computer Fraud and Abuse Act,” *Minnesota Law Review* (2010): http://www.minnesotalawreview.org/wp-content/uploads/2012/03/Kerr_MLR.pdf.

How USB Does (and Doesn't) Work

A Security Perspective

PETER C. JOHNSON



Peter C. Johnson received his bachelor's degree from the University of California, San Diego and worked for a couple of computer systems companies

in the Bay Area before escaping back to academia. He is currently polishing up his PhD dissertation at Dartmouth (not coincidentally related to security of USB stacks) and will begin work as a visiting assistant professor of computer science at Middlebury College in fall 2014. pete@cs.dartmouth.edu

USB devices are easy to take for granted: They're innocuous by their nature (who's afraid of a keyboard?) and by their ubiquity. However, the architecture of the Universal Serial Bus ecosystem is surprisingly complex and deeply embedded in modern operating systems. Furthermore, having risen to awareness on the backs of traditionally "dumb" devices like keyboards and mice, the features of the USB protocol that very much resemble wide-area networking protocols can be underappreciated. This combination of complexity, embeddedness, and underappreciation is the unholy trinity of security "features." In the following paragraphs, I hope to sprinkle some holy water on this situation, so come along while I first share some fire and brimstone, then give reason for hope. To the Batmobile!

"It can't be that bad," you're saying to yourself, "a USB attack requires physical access." While absolutely true, this misses a crucial technicality: An attack over USB must indeed be delivered physically, *but the attacker herself need not be physically present*. How many people, upon finding a USB thumb drive lying on the ground, are able to resist the temptation to plug it into the first computer they find? Sufficient anecdotal evidence exists to suggest the number is "few enough for us to worry" (though I wish you the best of luck in getting IRB approval to verify this experimentally). I don't mean to imply that the physical nature of USB is impotent as a defense, but that it is not dependable.

Speaking of Stuxnet, once the USB drive prepared by [REDACTED] was plugged into a machine beyond the defensive air gap, the "vulnerability" it initially exploited was that Windows was configured to execute `autorun.inf` on any inserted devices. The realization that such critical systems were thus (mis)configured no doubt makes the sysadmin- and security-minded out there a bit light-headed, and the same people might be tempted to breathe a sigh of relief that the initial infection vector could be so easily shut off. Completely setting aside the raft of zero-day exploits also employed by Stuxnet, indulging in the aforementioned sigh of relief is a bit premature.

How Bad Is It Really?

In March 2013, Microsoft patched three similar vulnerabilities (CVE-2013-1285, CVE-2013-1286, CVE-2013-1287) in all extant versions of Windows that allowed "escalation of privilege" [4]. NIST's National Vulnerability Database puts it a bit more starkly [7-9]:

- ◆ Access Complexity: Low
- ◆ Authentication Required: None
- ◆ Confidentiality Impact: Complete
- ◆ Integrity Impact: Complete
- ◆ Availability Impact: Complete

These were *not* system configuration issues, like failing to disable execution of `autorun.inf`; these were bugs in the kernel's USB stack, ring-0 code that is run automatically *every* time a USB device is plugged in to the system. Running such code with such privileges is a

How USB Does (and Doesn't) Work: A Security Perspective

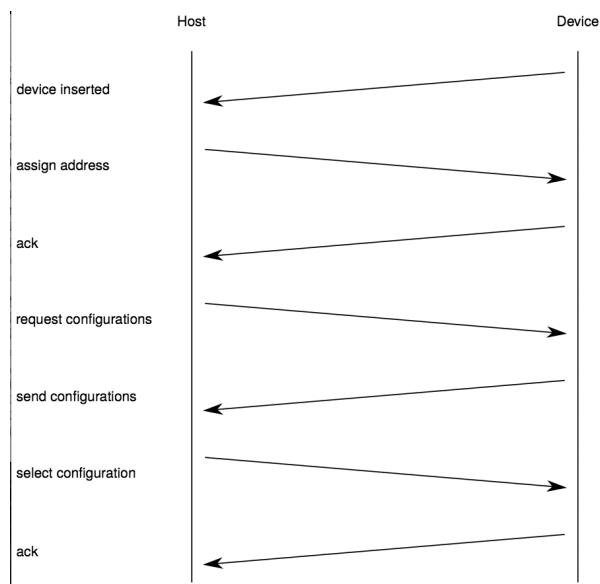


Figure 1: USB bus enumeration process as a ladder diagram

somewhat natural consequence of the “U” in USB: To support a broad array of devices, the kernel must get the device to identify itself so that the kernel can load the appropriate driver. This process is called “bus enumeration” (because the host is taking roll of devices on the bus) and looks something like this:

Kernel: Stop! What is your name?

Device: It is Arthur, King of the Britons.

Kernel: What is your quest?

Device: To seek the Holy Grail.

Kernel: What is the airspeed of an unladen swallow?

The device’s response at this point is key. If the answer is “I don’t know,” the device finds itself tossed from the Bridge of Death, never to return; if the answer is “What do you mean? An African or European swallow?” the kernel loses its mind and the Bridge of Death is no longer guarded. This example is surprisingly illustrative and not just the injection of a predictable computer nerd trope: A device can respond according to the USB protocol with an identification the kernel accepts, it can respond according to the protocol with an identification the kernel rejects (“I don’t know”), or it can deviate from the protocol entirely (“African or European?”).

If you squint only a bit, the bus enumeration process caricatured in Figure 1 bears more than a passing resemblance to the three-step TCP handshake or the request-response nature of an SMTP transaction. Figure 1 shows the enumeration process in the form of the ladder diagram we all know and love from the networking world. Indeed, the USB protocol sports a great number of fea-

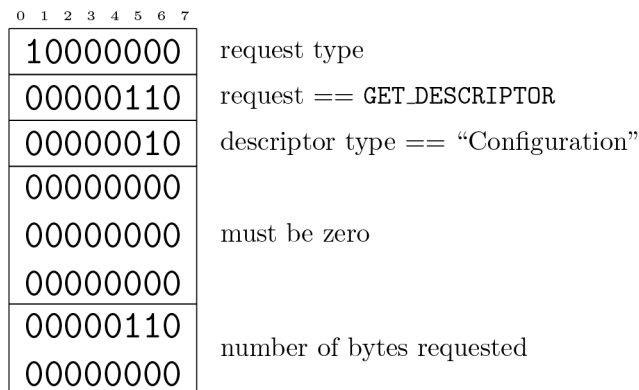


Figure 2: Bit-level description of message sent during bus enumeration from host to device requesting configuration descriptor

tures reminiscent of traditional network protocols: addressing, packetized data, sequence numbers, acknowledgments, and so on. (I’ll return to this comparison later on, I promise.)

The bugs Microsoft patched in 2013 were failures to correctly handle protocol deviations that allowed complete system compromise. Unfortunately, precise details on the patched vulnerabilities are difficult to come by, though we have good reason to believe the problems arose when parsing descriptors during enumeration. Parsing is one of those oft-underappreciated aspects of protocol implementation that should be relatively straightforward to get right, yet can lead to rather catastrophic failures. In the case of bus enumeration, the complexity of the messages involved can’t have helped. Figures 2 and 3 show a couple of packets sent during enumeration, specifically the host-to-device message that requests a configuration and the device-to-host response. (The specific semantics aren’t important, so don’t worry if “configurations,” “interfaces,” and “endpoints” mean nothing to you.)

The configuration request shown in Figure 2 is fairly simple, but the response (Figure 3) is anything but. After the standard

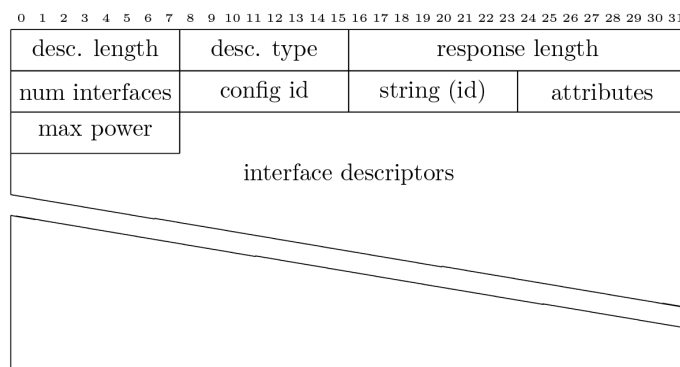


Figure 3: Bit-level description of message sent during bus enumeration from device to host containing configuration descriptors

header stuff (the details of which—barring the length fields—are secondary for this discussion), the device sends the descriptors of all interfaces contained within that particular configuration. Again, what a USB interface is doesn't really matter right now; the important point is that it's another message format that needs to be parsed. Not just that, but interfaces contain endpoints, the descriptors for which are also embedded in this single response. The result is a (relative) ton of data with all manner of length fields littered throughout, whose correct interpretation is vital to the correct interpretation of the message as a whole, and which are mutually dependent—that is, if the length field of one descriptor is messed up, the rest of the parse necessarily goes off the deep end.

This complexity makes implementing both host- and device-side logic dealing with descriptors a delicate task, but it gets better!

The Device *Is* the Application

Let's now move up the stack a bit and look at the application layer. In the world of networking, applications are, practically speaking, presented with their choice of either streams (TCP) or datagrams (UDP). Beyond that, they're more or less on their own, although standards like SMTP and DNS have been created to enforce some consistency (and, hopefully, quality). Proprietary protocols, on the other hand, are a completely different story: Vendors can and do design and implement protocols however they darn well please, which frequently results in the less-good kind of media attention. (Diebold, anyone?)

Fortunately for us innocent bystanders, some of the local effects of poorly designed or implemented application protocols can be mitigated by running server processes as an unprivileged user or in a chroot jail. Ideally, then, if a vulnerability in a protocol design or implementation is discovered, only resources owned by the unprivileged user or those within the chroot jail are susceptible to compromise. Other methods, including virtual machines and Linux containers, provide isolation sufficient to protect against whole-system compromise, although it isn't immediately clear how any of these map to the USB realm.

The USB protocol allows similar encapsulation (indeed, the USB Mass Storage Specification calls for stuffing raw SCSI commands inside USB packets much like iSCSI stuffs them inside IP packets). This freedom brings with it the same double-edged sword as in the networking world: Although developers can define their own protocols to create exciting new applications, they also run the risk of introducing vulnerabilities as they increase systems' attack surfaces. But wait! USB doesn't deal with applications, it deals with *devices*!

The implications of this are numerous and not altogether encouraging. First, it means that, once shipped, devices are often stuck with a specific version of a protocol implementation—one

that might be buggy (i.e., vulnerable) and difficult to upgrade. Second, the "server" implementation of the protocol frequently exists in the device driver—which *usually runs as kernel code*—so if the protocol is vulnerable, an exploit necessarily results in total system compromise. Third, although standards such as USB Mass Storage and USB Human Interface Device exist to bring some sanity to the land, many devices ship binary drivers. That's right: Devices can ship black-box code, implementing black-box protocols, that implicitly runs inside the kernel's address space.

That's okay, at least USB doesn't let the device pick which driver to talk to—nothing like `inetd` for networking services. Hmm? What's that you say? I already described how a device identifies itself to the kernel? And there's nothing to stop a device from identifying itself as a device with a known-vulnerable driver? And the kernel will happily load said driver and let the device talk to it, no questions asked? Well, that's potentially worrisome.

Unfortunately, it's true: In addition to the potentially unreliable nature of USB device driver protocols and implementations, a newly plugged device is in charge of choosing precisely which device driver to communicate with. To make matters worse, modern operating systems ship with support for a huge number of devices, many drivers for which haven't seen maintenance in years. To think there aren't exploitable vulnerabilities lurking among that crufty code would be naïve.

Okay, I'm Scared. Help?

In the preceding paragraphs, I've painted a pretty bleak picture. The (sort of) good news is that we don't know of any vulnerabilities in extant USB stacks. Of course, that doesn't mean there aren't any, nor does it mean that *other* people don't know about them or, if they do, that they aren't actively exploiting them. I said at the beginning that I'd give reason for hope, and here's where that comes in. I also said I'd return to the similarities between the USB architecture and the networking systems we all know and love. Two birds, one stone.

It's true that USB is an underappreciated attack vector; in contrast, networks are not. Because the two are so similar, we can take advantage of decades of tools, techniques, research, and lore in defending networks and apply it to the task of defending USB.

First and foremost, we know there's a problem and, as G.I. Joe would say, "Knowing is half the battle." My colleagues and I at Dartmouth have published work [3] that explores the attack surface presented by FreeBSD's USB stack, and the tools to mount such an attack. Andy Davis wrote an extensive whitepaper on USB driver vulnerabilities in 2013 [5]. As I mentioned earlier, Microsoft found a vulnerability, fixed it, and shipped the fix in its monthly Patch Tuesday event as opposed to waiting for a larger Service Pack update, evidence that Microsoft is convinced this area is worth defending as well.



Figure 4: The Facedancer board

Additionally, beyond the venerable microkernel model, a number of research projects have explored techniques to isolate device drivers in the name of system stability [2, 6, 10]. Microsoft has also started moving USB drivers to userspace. Though these measures won't eliminate vulnerabilities, they will help contain side effects of potentially buggy drivers.

We've also developed tools to help find vulnerabilities in USB implementations. Travis Goodspeed created the open source Facedancer (Figure 4) board [1] to facilitate exploration of both host- and device-side USB stacks, and I wrote the Python-based software stack to drive it.

The Facedancer board hosts an MSP430 microcontroller connected via SPI to a MAX3420 or MAX3421 USB controller. When connected to both a host machine and a target machine, the host can run Python code that causes the Facedancer to appear to the target as any USB device it wants. The Python framework handles as much or as little of the device enumeration process as you want it to: It allows you to write *in software* any USB device you can imagine, well-behaved or not. The latter is key: We want to emulate USB devices that deliberately misbehave so that we can probe the robustness of existing USB stacks that are not suitable for static analysis (either because they are too complex or because they are closed source).

We currently have code that emulates a USB keyboard, a USB thumb drive, and a USB FTDI serial connection. All of these have been successfully tested against real operating systems' USB stacks. The next step is to modify them to misbehave and see how the operating systems respond. If you're interested in playing around with a Facedancer but you'd prefer not to dig out your soldering iron, you can buy pre-assembled (and pre-flashed!) boards from <http://int3.cc>.

Mr. Samwise Gamgee holds that "it's the job that's never started as takes longest to finish." We've started. It is my hope that this article raises awareness among the operating system community that there may be exploitable vulnerabilities in this area of the code, and thus spur efforts to address them soon.

References

- [1] GoodFET: <http://goodfet.sourceforge.net>.
- [2] Silas Boyd-Wickizer and Nikolai Zeldovich, "Tolerating Malicious Device Drivers in Linux," *Proceedings of the USENIX Annual Technical Conference*, 2010.
- [3] Sergey Bratus, Travis Goodspeed, Peter C. Johnson, Sean W. Smith, and Ryan Speers, "Perimeter-Crossing Buses: A New Attack Surface for Embedded Systems," *Proceedings of the 7th Workshop on Embedded Systems Security (WESS 2012)*, 2012.
- [4] Microsoft Corporation, "Vulnerabilities in Kernel-Mode Drivers Could Allow Elevation of Privilege," Microsoft Security Bulletin MS13-027: <https://technet.microsoft.com/library/security/ms13-027>, 2013.
- [5] Andy Davis, "Lessons Learned from 50 Bugs: Common USB Driver Vulnerabilities," technical report, NCC Group, 2013.
- [6] Jorrit N. Herder, Herbert Bos, Ben Gras, Philip Homburg, and Andrew S. Tanenbaum, "Fault Isolation for Device Drivers," *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '09)*, 2009.
- [7] NIST, Vulnerability Summary for CVE-2013-1285: <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1285>, 2013.
- [8] NIST, Vulnerability Summary for CVE-2013-1286: <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1286>, 2013.
- [9] NIST, Vulnerability Summary for CVE-2013-1287: <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1287>, 2013.
- [10] Michael M. Swift, Brian N. Bershad, and Henry M. Levy, "Improving the Reliability of Commodity Operating Systems," *Proceedings of the 19th ACM Symposium on Operating System Principles (SOSP '03)*, 2003.

Computer Security at CERN

DR. STEFAN LÜDERS



Stefan Lüders, PhD, graduated from the Swiss Federal Institute of Technology in Zurich and joined CERN in 2002. Since 2009, he has headed the CERN

Computer Security Incident Response Team as CERN's computer security officer, coordinating CERN's office computing security, computer center security, GRID computing security, and control system security while taking into account CERN's academic environment and its operational needs. Dr. Lüders has spoken on computer security and control system cybersecurity topics to international bodies, governments, and companies on many different occasions, and has published several articles. Stefan.Lueders@cern.ch

Computer security is often seen as a technological problem: encryption, network anomaly detection, central (mobile) device management, firewalls, cloud-based SIEMs—each deemed to be the panacea. However, technical solutions fall short when dealing with a free and open academic environment like that of CERN, the European Organization for Nuclear Research. The CERN Computer Security Team faces the daily challenge of appropriately balancing CERN's operational and research needs with a reasonable level of computer security. At CERN, computer security is largely seen as a sociological problem. The first line of defense sits in front of the screen. Raising computer security awareness among CERN's 15,000 users is imperative to avert computer security incidents. Technological means, while still important, come second.

Introduction

CERN, the European Organization for Nuclear Research (or, according to its original French acronym, le Conseil Européen pour la Recherche Nucléaire; <http://cern.ch>), is one of the world's largest and most respected centers for scientific research. Its business is fundamental physics, finding out what the universe is made of and how it works.

CERN hosts a large complex of so-called particle accelerators and colliders, all providing insight into the subatomic structure of those particles. Accelerators boost beams of particles to high energies before they are made to collide with each other or with stationary targets. Detectors observe and record the results of these collisions. These records allow physicists to study the properties of the particles and learn about the laws of nature. Very often, a single experiment involves a collaboration of several hundred if not thousands of people from all over the world. Currently, several dozen different experiments are in operation at CERN, detecting collisions from half a dozen different particle accelerators, including the world's most powerful one, the Large Hadron Collider (LHC).

Besides seeking and finding answers to questions about the universe, CERN is advancing the frontiers of technology, bringing nations together through science, and training the scientists of tomorrow. CERN's 2250 staff members welcome about 15,000 guest physicists and visiting collaborators (so-called "users") annually. Because the particle experiments are international, CERN's users come from hundreds of universities, organizations, and laboratories from all over the world. In fact, only a few countries have never sent citizens to CERN. And the turnover is high: Students join CERN during their summer vacation to follow additional lectures, contribute to dedicated projects, and initiate their careers; BSc, MSc, and PhD students come to CERN for a few months (or years!) to attend seminars, receive training, and conduct, prepare, and finally write their theses; post-doc physicists join ongoing collaborations to advance their careers and satisfy their interests; professors regularly visit CERN to stay in touch with their CERN-based teams and their colleagues, for workshops, or to attend or give lectures; engineers and technicians arrive to install their technical equipment in CERN-based detectors or accelerators; young people do internships at CERN, in administration or one of the technical sectors. In parallel, many of those users connect remotely to

CERN's computing facilities to conduct analyses, simulations, or solve engineering problems. Alternatively, they can use the Worldwide LHC Computing Grid (WLCG), a network of major computer centers around the globe with CERN being its head-node (the so-called "Tier-0"), to conduct large-scale analysis of physics data—we are talking about several petabytes of accumulated data sets produced by individual experiments.

Thus, CERN not only presents a working environment to users, it also provides private accommodations and hostels on CERN premises, restaurants, and recreational facilities. CERN hosts several dozen different clubs for after-work hours (e.g., micro-electronics club, car club, running club, yoga club, music club). In fact, professional and private life at CERN is pretty much entangled, giving users the necessary environment and freedom to pursue their research.

Academic Freedom versus Security

With such a vast academic user community and so many different cultures, nationalities, interests, and aims represented, standardization is unrealistic. From an IT perspective, users can bring their own laptops with their favorite operating systems in any flavor or language (the "BYOD" trend has existed at CERN for a while); developers can program in their favorite programming language; users can run any software tool or program and deploy whatever technological means they deem necessary to reach their goals. In that respect, CERN can be seen as an ISP and computing service provider for its users. Furthermore, users are accustomed to exchanging ideas, sharing information, and publishing results freely. Web sites can be created at the convenience of the users. In short, CERN hosts a vast academic environment that relies on freedom of choice and freedom of communication. It is not without reason that the World Wide Web was proposed 25 years ago by a CERN employee, Tim Berners-Lee.

CERN's Computer Security Team (<https://security.web.cern.ch/security/home/en/index.shtml>) has to find the right balance between CERN's academic environment, the safe operation of its accelerators and experiments, and its computer security. While this academic freedom makes for a very dynamic and innovative environment, with new systems and services being deployed all the time, it also increases security risks to our computing. CERN's Computer Security Team is mandated to minimize both the likelihood and impact of security events; to prevent and protect against digital attacks; and to maintain premium detection and response capabilities.

Although the environment is "free" at large, CERN users cannot act as if they are in a void. Use of CERN's computing facilities is governed by a set of lightweight policies (<https://security.web.cern.ch/security/rules/en/index.shtml>) that set rigid limits on what is allowed and what is not. Although "users can bring

their own laptops," for example, they are required to guarantee the laptops' security and apply prompt patching; whereas "Web sites can be created at the convenience of the users," the contents must not be offensive or illegal; although "users can run any software tool or program," they are still bound to obey copyrights and license conditions. In addition, any usage must neither be detrimental to the workings of the organization nor significantly affect computing resources (e.g., computing power, disk space, network bandwidth). For example, generating crypto-currencies on CERN-owned computing clusters or running Nmap scans without explicit authorization by the CERN Computer Security Team is prohibited. The CERN computing rules even provide the framework for the private use of CERN's computing facilities: While private and personal usage is generally tolerated, illegal, inappropriate, or offensive activities are banned, and violations lead to administrative measures.

However, the most important feature of CERN's security paradigm is delegation. While the CERN computer security officer is mandated by CERN's director general to coordinate all aspects of computer security at CERN through prevention, protection, detection, and response, he is not the person ultimately responsible for all computer security at CERN given the heterogeneous environment, and the academic freedom that comes with it, and given the consequently limited leverage of control. Instead, at CERN, staff and users individually assume primary responsibility for the security and protection of their computers, the operating systems they run, the applications they install, the software they program, the data they own, and the Web sites they maintain. Service and system managers are responsible for ensuring that their services and systems run securely, are maintained, and follow good security practices. Project managers are responsible for the security of their projects, and the line management for that of their constituency.

Basically, at CERN, "computer security" is dealt with in the same way as safety. Safety cannot easily be ignored: If there is a puddle of water on the floor, it is my personal responsibility to prevent people from slipping on it, and I cannot just relegate this to the building's safety officer. It's the same for security. Still, this does not mean that users are singly responsible for their own security. The CERN Computer Security Team (nominally four staff and a few students) provides assistance, consulting, and help in order to enable CERN's staff and users to fully, effectively, and efficiently assume that responsibility. CERN's IT department provides the necessary common tools and general services for the Computer Security Team and, more importantly, for CERN's user community: Instead of managing and patching their own PCs, users can obtain a centrally managed PC and antivirus software which is kept up-to-date by the IT department. The IT department provides Web servers, content management systems, databases, file storage systems, and

Computer Security at CERN

engineering applications that are properly managed, adequately secured, and maintained over the long run. In short, users can delegate their responsibility for security to the IT department and avoid the burden of managing “security” themselves. Instead, they can focus on their core work. Still, it is up to the users and each experiment to opt in. They are encouraged to do so, and usually do.

Security Training

With such a heterogeneous community, user awareness, education, and training are paramount. Users are often the weakest point in the security chain, are not necessarily aware of computer security issues, and do not always feel concerned. It is hard for them to really assume the responsibility imposed on them by the CERN Computing Rules. Thus, a trigger is needed to raise their computer security awareness or—even better—to educate them such that they understand security risks. Ideally, this is supposed to introduce a cultural change in the same way that young children can be taught to swim or to look left and right before crossing a street. Once certain practices become engrained, safety on the road or in the pool is automatically and subconsciously guaranteed. For “security,” we need the same automatism, (e.g., when receiving a “phishing” email or when prompted to install a new program).

Therefore, all new CERN users receive an introductory course on computer security matters when they arrive at CERN. This course is paralleled by an online course followed by a 10-question multiple-choice quiz to be successfully passed in order to obtain a computer account giving access to CERN’s computing facilities. This course and quiz must be renewed every three years and is aligned with similar courses on safety. In addition, various awareness campaigns are given periodically to all CERN units to reiterate the main security messages: “Protect your computers,” “Be careful with email and the Web,” “Protect your passwords,” “Protect your files and data,” “Respect copyrights,” and “Follow the CERN Computing Rules.” These six primary messages basically apply to everyone, everywhere, not only those at CERN, and the course encourages people to apply security principles at CERN as well as at home. A series of videos, posters, and handouts complement these campaigns and provide additional information. Overall, the Computer Security Team collaborates with the CERN Human Resources department to better integrate security knowledge, awareness, and behaviors into existing processes and situations.

The power of these awareness campaigns can be measured via the number of passwords lost to “phishing”: In 2008, 40 of about 1500 CERN recipients of a crude phishing mail divulged their password to the attackers. A subsequent analysis has shown that neither age, gender, attitude toward technology, salary, nor “intelligence” determines the likelihood of succumb-

ing to phishing. Instead, what counts (for the attacker at least) is the moment. Many affected recipients that we interviewed stated that they were busy with something, saw the mail from “Webmail IT service,” and answered it just to get rid of it. Only later did it occur to them that the “Webmail IT service” might not have been necessarily CERN’s. Today, after three years of awareness campaigns, CERN loses only about two to three accounts to such emails per month. Given the more than 20,000 active users and high turnover, this is deemed acceptably low.

Still, these awareness campaigns are just seeds. Once people understand that “security” is part of the overall IT phase-space containing “functionality,” “usability,” “availability,” and “maintainability,” they naturally ask for more. This is the moment when users ask the CERN Computer Security Team to consult with them before starting new IT projects, for penetration testing, and for assessing the security footprint of new systems and the auditing of existing deployments. It is also the moment for dedicated training: For software developers and system experts, the Computer Security Team, in collaboration with the CERN’s Technical Training team, provides optional in-depth training sessions on developing secure software, secure Web application development, as well as dedicated sessions on secure coding in C/C++, Java, Perl, Python, and for Web applications. In the past, these courses have been quite successful, with attendees from all different areas within CERN.

In addition, a series of static code analyzers were made available to all developers in order to further improve their code: Coverity and flawfinder for C/C++, FindBugs and CodePro’s Analytics for Java, RATS for Perl/Python/PHP, pychecker for Python, and Pixy for Perl. The configuration of those tools is simple, and, admittedly, these code analyzers will never find all flaws. However, even in their basic configuration they help developers to easily detect at least some potential security weaknesses (both functional bugs and security vulnerabilities). Once developers see these benefits, they are open to additional steps towards a “security” mind-set: enabling and checking on compiler warnings and error messages; employing more sophisticated code analyzers; doing full code reviews; embracing a full-blown secure software development life cycle; and, finally, employing sophisticated tools for software management and integration with nightly builds, regression testing, and permanent scanning for weaknesses, sub-optimal configurations, and flaws. Once developers are at this level, security worries are diminished.

Vulnerability Scanning

Preventive training is good, but verification is also necessary. Currently, CERN has registered on its public-IP networks about 180,000 devices (PCs, laptops, smartphones, tablets, etc.) by their MAC address, and it controls access through RADIUS/MAC-address-based authentication. About 80,000 have been

seen active use during the past months. While MAC-address spoofing rarely happens, this is usually quickly detected and followed up as a violation of the CERN Computing Rules. CERN's main computer centers alone host about 10,000 servers, 100,000 cores, and 75,000 hard disks, which currently store more than 100 PB of data (<http://information-technology.Web.cern.ch/about/computer-centre>). The CERN identity management system currently lists more than 36,000 CERN personal accounts plus about 8500 accounts for special purposes (e.g., database accounts, generic accounts for running automatic services). Its central Web service holds more than 12,000 Web sites (e.g., <https://security.web.cern.ch/security/home/en/index.shtml>) on more than 3 million Web pages using Sharepoint, Drupal, J2EE, CGI/ Python/Perl scripts, or plain HTML. A few hundred more Web servers are managed by individuals (users) for dedicated purposes that cannot be easily served by the central Web services (e.g., Web sites requiring proprietary software or database integration).

The Computer Security Team, therefore, actively and permanently scans major computing resources for vulnerabilities. All Web sites hosted at CERN are regularly scanned for vulnerabilities using Skipfish (a tool published by Google) as well as with Wapiti and w3af. Additional tools produce an inventory of all Web sites, Web applications, and Web technologies used on individual hosts, and compare this with a list of vulnerable or outdated versions. Similarly, the level of protections of all devices connected to CERN networks is regularly assessed using Nmap, which subsequently gives another valuable inventory of currently running services and their versions. A dedicated custom-scanning suite dubbed Prodder probes deeper into particular security issues (e.g., writable folders on Windows PCs, outdated myPHPadmin frameworks). CERN's centrally managed file systems and software repositories are regularly scanned for exposed (i.e., public) credentials like private SSH keys as well as for inconsistencies in their access configuration: A "private" folder should never permit access to everyone holding a CERN computing account. Finally, devices that require access from the Internet, like Web servers, have to undergo dedicated scans using Nessus and Skipfish. Usually, the results indicate the quality of the server setup and its security. Only servers that successfully passed the scans will be granted that access through CERN's outer perimeter firewall (the firewall hardware is maintained by the CERN Networking Group, but its configuration is maintained by the Computer Security Team).

Essential for such permanent scanning is a comprehensive and all-encompassing asset inventory: Devices (and their respective firewall openings), accounts, and Web sites must have a registered owner taking over the responsibility entrusted to him or her. Lots of effort has been made to ensure that this inventory is permanently up-to-date and accurate. Other computing services

are automatically assigned to an organic unit within the CERN hierarchy, which, thus, provides the necessary contact points in case of security issues. A recent project has been launched to further improve on this and have a life cycle for any computing resource used at CERN. Although declaring new resources (accounts, Web sites, devices, etc.) is always based on the incentive of the requestor, the resource life cycle will ensure that there is also an incentive once the registered owner leaves CERN. The resources are passed on to a new owner, assigned to the leaving person's supervisor, or destroyed.

Thanks to this proper asset inventory, all potential vulnerabilities can be communicated directly to the corresponding owner of the affected account, file space, Web site, or device, and must be mitigated. The Computer Security Team's Web-based event management system provides all necessary tips and tricks to allow users to mitigate these findings themselves. Alternatively, the IT department and the Computer Security Team once more provide assistance and help. Only in rare cases does the Computer Security Team need to take a harsher stance and block the Internet access of a certain Web site or disconnect a certain device from the network (not having permanent access to Facebook has been proven to be a good incentive to act quickly). On average, only three to six such blockings are executed per month. In all cases, the tight interaction with the users also opens up an opportunity to advertise the aforementioned training sessions.

For high-profile Web sites and vital computing services, the Computer Security Team offers in-depth reviews and security assessments. Dedicated so-called "Security Baselines" provide users with a short list of good practices for securing their computing servers, Web servers, or file servers.

Incident Response

Despite all of these preventive measures, incidents inevitably do happen. The Computer Security Team has deployed a series of sophisticated, intrusion detection means-monitoring activities on centralized computing facilities and on CERN networks in general.

These means include the centralized monitoring of the antivirus software installed on all centrally managed Windows PCs by colleagues from the Windows Support Group, the detection of malicious domains and IPs in DNS requests and in all network traffic, deep-packet inspection using "Snort," the statistical analysis of network flows ("netflows") indicating abnormal behavior, and the analysis of computer logs. All sensors run on standard Computer Center hardware managed by the IT department and configured through IT's "Agile Infrastructure" (i.e., Puppet, OpenStack, Git, etc.). The data analyses are fully automated, and any owner of an affected device is automatically notified of a malicious security event. Once more, the team's

Computer Security at CERN

Web-based event management system provides all necessary tips and tricks to allow users to mitigate these findings themselves. The mail system automatically suspends any mail activity if more than 3000 mails have been sent during one day. Alternatively, the user is assisted by the Computer Security Team's first line of support in solving the identified issues. More severe incidents are handled by the Computer Security Team's CSIRT (Computer Security Incident Response Team).

Summary

CERN's user community is vast and is used to the spirit of academic freedom and free communication. It is difficult (impossible?) to centralize or standardize the necessary computing environment without spoiling this freedom, and so a heterogeneous environment is prevailing at CERN. Given this special

challenge, the Computer Security Team had to tightly involve CERN's users: At CERN, users are primarily responsible for the security and protection of their assets. The Computer Security Team provides assistance and help, with a primary focus on education and culture change. Once "security" is part of the average user's mind-set, the overall level of security should further increase. Until then, sophisticated detection and protection means have spared CERN from too many too-visible security incidents. The Computer Security Team is working hard to maintain this status quo.



Do you know about the USENIX Open Access Policy?

USENIX is the first computing association to offer free and open access to all of our conferences proceedings and videos. We stand by our mission to foster excellence and innovation while supporting research with a practical bias. Your financial support plays a major role in making this endeavor successful.

Please help to us to sustain and grow our open access program. Donate to the USENIX Annual Fund, renew your membership, and ask your colleagues to join or renew today.

www.usenix.org/annual-fund



Buy the Box Set!

Whether you had to miss a conference or just didn't make it to all of the sessions, here's your chance to watch (and re-watch) the videos from your favorite USENIX events. Purchase the "Box Set," a USB drive containing the high-resolution videos from the technical sessions. This is perfect for folks on the go or those without consistent Internet access.

Box Sets are available for:

- » **URES '14:** 2014 USENIX Release Engineering Summit
- » **USENIX ATC '14:** 2014 USENIX Annual Technical Conference
- » **UCMS '14:** 2014 USENIX Configuration Management Summit
- » **HotStorage '14:** 6th USENIX Workshop on Hot Topics in Storage and File Systems
- » **HotCloud '14:** 6th USENIX Workshop on Hot Topics in Cloud Computing
- » **NSDI '14:** 11th USENIX Symposium on Networked Systems Design and Implementation
- » **FAST '14:** 12th USENIX Conference on File and Storage Technologies
- » **LISA '13:** 27th Large Installation System Administration Conference
- » **USENIX Security '13:** 22nd USENIX Security Symposium
- » **HealthTech '13:** 2013 USENIX Workshop on Health Information Technologies
- » **WOOT '13:** 7th USENIX Workshop on Offensive Technologies
- » **UCMS '13:** 2013 USENIX Configuration Management Summit
- » **HotStorage '13:** 5th USENIX Workshop on Hot Topics in Storage and File Systems
- » **HotCloud '13:** 5th USENIX Workshop on Hot Topics in Cloud Computing
- » **WiAC '13:** 2013 USENIX Women in Advanced Computing Summit
- » **NSDI '13:** 10th USENIX Symposium on Networked Systems Design and Implementation
- » **FAST '13:** 11th USENIX Conference on File and Storage Technologies
- » **LISA '12:** 26th Large Installation System Administration Conference

Learn more at: www.usenix.org/boxsets

Building Web Applications on Top of Encrypted Data Using Mylar

RALUCA ADA POPA, EMILY STARK, JONAS HELFER, STEVEN VALDEZ,
NICKOLAI ZELDOVICH, M. FRANS KAASHOEK, AND HARI BALAKRISHNAN



Raluca Ada Popa is a fourth-year PhD student at MIT working in security, systems, and applied cryptography. She is the recipient of a Google PhD fellowship for secure cloud computing and a CRA Outstanding Undergraduate Award. ralucap@mit.edu



Emily Stark is a core developer at Meteor Development Group. She holds an MS degree from MIT and a BS from Stanford University, both in computer science. emily@meteor.com



Jonas Helfer is a PhD student at MIT's Computer Science and Artificial Intelligence Lab. He holds a master's degree in computer science from EPFL (Switzerland). His many research interests include systems security, Web security and software project management. helfer@mit.edu

Using a Web application for confidential data requires the user to trust the server to protect the data from unauthorized disclosures. This trust is often misplaced, however, because there are many ways in which confidential data could leak from a server. For example, attackers could exploit a vulnerability in the server software to break in [9], a curious administrator could peek at the data on the server [1, 2], or the server operator may be compelled to disclose data by law [3]. How can one build Web applications that protect data confidentiality against attackers with *full access* to servers?

We developed Mylar for this purpose. Mylar is a new platform for building Web applications that stores sensitive data *encrypted* on the server. The keys that can decrypt the data are stored in some users' Web browsers, and the data only gets decrypted in these browsers. Even if an attacker fully compromises the server, the attacker gets access to only encrypted data and does not have the necessary decryption keys. Mylar achieves this organization through a new data sharing mechanism, practical ways of computing on encrypted data at the server, and a mechanism for verifying that the application code was not tampered with by a compromised server.

Crucially, Mylar enables many classes of applications to protect confidential data from compromised servers in a practical way. It leverages the recent shift in Web application frameworks towards implementing logic in client-side JavaScript code, and sending data, rather than HTML, over the network [5]; such a framework provides a clean foundation for security.

Mylar is open source and can be found at <http://css.csail.mit.edu/mylar/>. This article covers how Mylar works at a high level and how to use Mylar on an example application, a chat application. For more details on the research behind Mylar (e.g., detailed decryption of each component, detailed evaluation, etc.), we refer the reader to our NSDI '14 paper [7].

Mylar's Techniques

To understand Mylar's techniques, it is helpful to consider a simple attempt to solve the problem and to observe why this attempt does not suffice. A simple idea is to give each user her own encryption key, encrypt a user's data with that user's key in the Web browser, and store only encrypted data on the server. This model ensures that an adversary would not be able to read any confidential information on the server, because he would lack the necessary decryption keys. In fact, this model has been already adopted by some privacy-conscious Web applications [4, 8].

Unfortunately, this approach suffers from three significant security, functionality, and efficiency shortcomings. First, a compromised server could provide malicious client-side code to the browser and extract the user's key and data. Ensuring that the server did not tamper with the application code is difficult because a Web application consists of many files, such

Building Web Applications on Top of Encrypted Data Using Mylar



Steven Valdez is a graduate student pursuing a dual

bachelor's/master's degree in computer science at MIT, focusing on systems and security research. dvorak42@mit.edu



Nickolai Zeldovich is an associate professor at MIT.

His research interests are in building practical secure systems. nickolai@csail.mit.edu



M. Frans Kaashoek is a

professor at MIT, where he co-leads the Parallel and Distributed Operating Systems Group (<http://www.pdos.csail.mit.edu/>).

Frans is a member of the National Academy of Engineering and the American Academy of Arts and Sciences and is the recipient of the ACM SIGOPS Mark Weiser award and the 2010 ACM-Infosys Foundation award. He co-founded Sightpath, Inc. and Mazu Networks, Inc. kaashoek@mit.edu



Hari Balakrishnan's research interests are in networked

computer systems. He is a professor of computer science at MIT. hari@csail.mit.edu

as HTML pages, JavaScript code, and CSS style sheets, and the HTML pages are often dynamically generated.

Second, this approach does not provide data sharing between users, a crucial function of Web applications. To address this problem, one might consider encrypting shared documents with separate keys and distributing each key to all users sharing a document via the server. However, distributing keys via the server is challenging because a compromised server can supply arbitrary keys to users and thus trick a user into using incorrect keys.

Third, this approach requires all of the application logic to run in a user's Web browser, because it can decrypt the user's encrypted data. But this is often impractical: For instance, doing a keyword search would require downloading all the documents to the browser.

Mylar overcomes the challenges mentioned above with a combination of systems techniques and novel cryptographic primitives, as follows:

1. **Data sharing.** To enable sharing, each sensitive data item is encrypted with a key available to users who share the item. To prevent the server from cheating during key distribution, Mylar provides a mechanism for establishing the correctness of keys obtained from the server: Mylar forms certificate paths to attest to public keys and allows the application to specify which certificate paths can be trusted in each use context. In combination with a user interface that displays the appropriate certificate components to the user, this technique ensures that even a compromised server cannot trick the application into using the wrong key.
2. **Computing over encrypted data.** Keyword search is a common operation in Web applications, but it is often impractical to run on the client because it would require downloading large amounts of data to the user's machine. Although practical cryptographic schemes exist for keyword search, they require that data be encrypted with a single key. This restriction makes it difficult to apply these schemes to Web applications that have many users and hence have data encrypted with many different keys. Mylar provides the first cryptographic scheme that can perform keyword search efficiently over data encrypted with *different* keys. The client provides an encrypted word to the server, and the server can return all documents that contain this word without learning the word or the contents of the documents.
3. **Verifying application code.** With Mylar, code running in a Web browser has access to the user's decrypted data and keys, but the code itself comes from the untrusted server. To ensure that this code has not been tampered with, Mylar checks that the code is properly signed by the Web site owner. This checking is possible because application code and data are separate in Mylar, so the code is static. Mylar uses two origins to simplify code verification for a Web application. The primary origin hosts only the top-level HTML page of the application, whose signature is verified using a public key found in the server's X.509 certificate. All other files come from a secondary origin, so that if they are loaded as a top-level page, they do not have access to the primary origin. Mylar verifies the hash of these files against an expected hash contained in the top-level page.

Mylar's Architecture

There are three different parties in Mylar: the users, the Web site owner, and the server operator. Mylar's goal is to help the site owner protect the confidential data of users in the face of a malicious or compromised server operator.

System Overview

Mylar embraces the trend towards client-side Web applications; Mylar's design is suitable for platforms that:

Building Web Applications on Top of Encrypted Data Using Mylar

1. Enable client-side computation on data received from the server.
2. Allow the client to intercept data going to the server and data coming from the server.
3. Separate application code from data, so that the HTML pages supplied by the server are static.

AJAX Web applications with a unified interface for sending data over the network, such as Meteor [5], fit this model. Such frameworks provide a clean foundation for security, because they send data separately from the HTML page that presents the data. In contrast, traditional server-side frameworks incorporate dynamic data into the application's HTML page in arbitrary ways, making it difficult to encrypt and decrypt the dynamic data on each page while checking that the fixed parts of the page have not been tampered with.

Mylar's Components

The architecture of Mylar is shown in Figure 1. Mylar consists of the four following components:

Browser extension. It is responsible for verifying that the client-side code of a Web application that is loaded from the server has not been tampered with.

Client-side library. It intercepts data sent to and from the server and encrypts or decrypts that data. Each user has a private-public key pair. The client-side library stores the private key of the user at the server, encrypted with the user's password. (The private key of a user can also be stored at a trusted third-party server, to better protect it from offline password guessing attacks and to recover from forgotten passwords without regenerating keys.) When the user logs in, the client-side library fetches and decrypts the user's private key. For shared data, Mylar's client creates separate keys that are also stored at the server in encrypted form.

Server-side library. It performs computation over encrypted data at the server. Specifically, Mylar supports keyword search over encrypted data, because we have found that many applications use keyword search.

Identity provider (IDP). For some applications, Mylar needs a trusted identity provider service (IDP) to verify that a given public key belongs to a particular username. An application needs the IDP if the application has no trusted way of verifying the users who create accounts, and the application allows users to choose whom to share data with. For example, if Alice wants to share a sensitive document with Bob, Mylar's client needs the public key of Bob to encrypt the document. A compromised server could provide the public key of an attacker, so Mylar needs a way to verify the public key. The IDP helps Mylar perform this verification by signing the user's public key and username. An application does *not need* the IDP if the site owner wants to protect only against attackers that do not actively change a server's behavior (namely, attackers that only read the data at the server, and do not install software at the server), or if the application has a limited sharing pattern for which it can use a static root of trust (as described in our full paper [7]).

An IDP can be shared by many applications, similar to an OpenID provider [6]. The IDP does not store per-application state, and Mylar contacts the IDP only when a user first creates an account in an application; afterwards, the application server stores the certificate from the IDP.

Threat Model

Threats

Both the application and the database servers can be *fully* controlled by an adversary: The adversary may obtain all data from the server, cause the server to send arbitrary responses to Web browsers, etc. This model subsumes a wide range of real-world security problems, from bugs in server software to insider attacks.

Mylar also allows some user machines to be controlled by the adversary and to collude with the server. This may be either because the adversary is a user of the application or because the adversary broke into a user's machine.

Guarantees

Mylar protects a data item's confidentiality in the face of arbitrary server compromises, as long as none of the users with access to that data item use a compromised machine. Mylar does not hide data access patterns or communication and timing patterns in an application. Mylar provides data authentication guarantees but does not guarantee the freshness or correctness of results from the computation at the server.

Assumptions

To provide the above guarantees, Mylar assumes that the Web application as written by the developer will not send user data or keys to untrustworthy recipients and cannot be tricked into doing so by exploiting bugs (e.g., cross-site scripting). Our

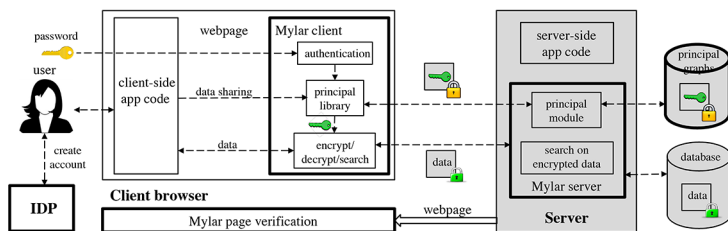


Figure 1: System overview. Shaded components have access only to encrypted data. Thick borders indicate components introduced by Mylar.

Building Web Applications on Top of Encrypted Data Using Mylar

Function	Semantics
<code>idp_config(url, pubkey)</code>	Declares the <i>url</i> and <i>pubkey</i> of the IDP and returns the principal corresponding to the IDP.
<code>create_user(uname, password, auth_princ)</code>	Creates an account for user <i>uname</i> , which is certified by principal <i>auth_princ</i> .
<code>login(uname, password)</code>	Logs in user <i>uname</i> .
<code>logout()</code>	Logs out the currently logged-in user.
<code>collection.encrypted({field: princ_field}, ...)</code>	Specify that <i>field</i> in <i>collection</i> should be encrypted for the principal in <i>princ_field</i> .
<code>collection.auth_set([princ_field, fields], ...)</code>	Authenticate the set of <i>fields</i> with principal in <i>princ_field</i> .
<code>collection.searchable(field)</code>	Mark <i>field</i> in <i>collection</i> as searchable.
<code>collection.search(word, field, princ, filter, proj)</code>	Search for <i>word</i> in <i>field</i> of <i>collection</i> , filter results by <i>filter</i> , and project only the fields in <i>proj</i> from the results. Use <i>princ</i> 's key to generate the search token.
<code>princ_create(name, creator_princ)</code>	Create principal named <i>name</i> , sign the principal with <i>creator_princ</i> , and give <i>creator_princ</i> access to it.
<code>princ_create_static(name, password)</code>	Create a static principal called <i>name</i> , hardcode it in the application, and wrap its secret keys with <i>password</i> .
<code>princ_static(name, password)</code>	Return the static principal <i>name</i> ; if a correct password is specified, also load the secret keys for this principal.
<code>princ_current()</code>	Return the principal of currently logged in user.
<code>princ_lookup(name₁, ..., name_n, root)</code>	Look up principal named <i>name₁</i> as certified by a chain of principals named <i>name₁</i> , rooted in <i>root</i> (e.g., the IDP).
<code>granter.add_access(grantee)</code>	Give the <i>grantee</i> principal access to the <i>granter</i> principal.
<code>grantee.allow_search(granter)</code>	Allow matching keywords from <i>grantee</i> on <i>granter</i> 's data.

Figure 2: Mylar API for application developers split in three sections: authentication, encryption/integrity annotations, and access control. All of the functions except `princ_create_static` and `searchable` run in the client browser. This API assumes a MongoDB storage model where data is organized as collections of documents, and each document consists of fieldname-and-value pairs. Mylar also preserves the generic functionality for unencrypted data of the underlying Web framework.

prototype of Mylar is built on top of Meteor, a framework that helps programmers avoid many common classes of bugs in practice.

Mylar also assumes that the IDP correctly verifies each user's identity (e.g., email address) when signing certificates. To simplify the job of building a trustworthy IDP, Mylar does not store any application state at the IDP, contacts the IDP only when a user first registers, and allows the IDP to be shared across applications.

Finally, Mylar assumes that the user checks the Web browser's security indicator (e.g., the https shield icon) and the URL of the Web application they are using before entering any sensitive data. This assumption is identical to what users must already do to safely interact with a trusted server. If the user falls for a phishing attack, neither Mylar nor a trusted server can prevent the user from entering confidential data into the adversary's Web application.

Security Overview

At a high level, Mylar achieves its goal as follows. First, it verifies the application code running in the browser, so that it is safe to give client-side code access to keys and plaintext data. Then, the client code encrypts the data marked sensitive before sending it to the server. Because users need to share data, Mylar provides a mechanism to securely share and look up keys among users. Finally, to perform server-side processing, Mylar introduces a new cryptographic scheme that can perform keyword search over documents encrypted with many different keys, without revealing the content of the encrypted documents or the word being searched for.

Implementation and Evaluation

To evaluate Mylar's design, we built a prototype on top of the Meteor Web application framework [5]. We ported six applications to protect confidential data using Mylar: a medical application for endometriosis patients, a Web site for managing

Building Web Applications on Top of Encrypted Data Using Mylar

homework and grades, a chat application called kChat, a forum, a calendar, and a photo-sharing application. The endometriosis application is used to collect data from patients with that medical condition and was designed under the aegis of the MIT Center for Gynepathology Research by surgeons at the Newton-Wellesley hospital (affiliated with Harvard Medical School) in collaboration with biological engineers at MIT; the Mylar-secured version is currently being tested by patients and is undergoing IRB approval before deployment.

Our results show that Mylar requires little developer effort: We had to modify an average of just 36 lines of code per application. We also evaluated the performance of Mylar on three of the applications above. For example, for kChat, our results show that Mylar incurs modest overheads: a 17% throughput reduction and a 50-ms latency increase for the most common operation (sending a message). These results suggest that Mylar is a good fit for multi-user Web applications with data sharing.

Using Mylar

Mylar for Developers

The developer starts with a regular (non-encrypted) Web application implemented in Mylar's underlying Web platform (Meteor in our prototype). To secure this application with Mylar, a developer uses Mylar's API (Figure 2), which we show how to use on a chat example. First, the developer uses Mylar's authentication library for user login and account creation. If the application allows a user to choose which other users to share data with, the developer should also specify the URL and public key of a trusted IDP.

Second, the developer specifies which data in the application should be encrypted and who should have access to it. Mylar uses principals for access control; a principal corresponds to a public/private key pair and represents an application-level access control entity, such as a user, a group, or a shared document. In our prototype, all data is stored in MongoDB collections, and the developer annotates each collection with the set of fields that contain confidential data and the name of the principal that should have access to that data (i.e., whose key should be used).

Third, the developer specifies which principals in the application have access to which other principals. For example, if Alice wants to invite Bob to a confidential chat, the application must invoke the Mylar client to grant Bob's principal access to the chat room principal.

Fourth, the developer changes their server-side code to invoke the Mylar server-side library when performing keyword search. Our prototype's client-side library provides functions for common operations such as keyword search over a specific field in a MongoDB collection.

Finally, as part of installing the Web application, the site owner generates a public/private key pair and signs the application's files with the private key using Mylar's bundling tool. The Web application must be hosted using https, and the site owner's public key must be stored in the Web server's X.509 certificate. This ensures that even if the server is compromised, Mylar's browser extension will know the site owner's public key and will refuse to load client-side code if it has been tampered with.

Chat Application Example

To demonstrate how a developer can build a Mylar application, we show the changes that we made to the kChat application to encrypt messages. In kChat, users can create chat rooms, and existing members of a chat room can invite new users to join. Only invited users have access to the messages from the room. A user can search over data from the rooms she has access to. Figure 3 shows the changes we made to kChat, using Mylar's API (Figure 2).

```
// On both the client and the server:
idp = idp_config(url, pubkey);
Messages.encrypted({“message”: “roomprinc”});
Messages.auth_set([“roomprinc”, [“id”, “message”,
    “room”, “date”]]);
Messages.searchable(“message”);

// On the client:
function create_user(uname, password):
    create_user(uname, password, idp);

function create_room(roomtitle):
    princ_create(roomtitle, princ_current());

function invite_user(username):
    global room_princ;
    room_princ.add_access(princ_lookup(username, idp));

function join_room(room):
    global cur_room, room_princ;
    cur_room = room;
    room_princ = princ_lookup(room.name,
        room.creator, idp);

function send_message(msg):
    global cur_room, room_princ;
    Messages.insert({message: msg, room: cur_room.id,
        date: new Date().toString(),
        roomprinc: room_princ});

function search(word):
    return Messages.search(word, “message”,
        princ_current(), all, all);
```

Figure 3: Pseudo-code for changes to the kChat application to encrypt messages. Not shown is unchanged code for managing rooms, receiving and displaying messages, and login/logout (Mylar provides wrappers for Meteor's user accounts API).

The call to `Messages.encrypted` specifies that data in the “message” field of that collection should be encrypted. This data will be encrypted with the public key of the principal specified in the “roomprinc” field. All future accesses to the `Messages` collection will be transparently encrypted and decrypted by Mylar from this point. The call to `Messages.searchable` specifies that clients will need to search over the “message” field; consequently, Mylar will store a searchable encryption of each message in addition to a standard ciphertext.

When a user creates a new room (`create_room`), the application in turn creates a new principal, named after the room title and signed by the creator’s principal. To invite a user to a room, the application needs to give the new user access to the room principal, which it does by invoking `add_access` in `invite_user`.

When joining a room (`join_room`), the application must look up the room’s public key, so that it can encrypt messages sent to that room. The application specifies both the expected room title as well as the room creator as arguments to `princ_lookup`, to distinguish between rooms with the same title.

To send a message to a chat room, `kChat` needs to specify a principal in the `roomprinc` field of the newly inserted document. In this case, the application keeps the current room’s principal in the `room_princ` global variable. Similarly, when searching for messages containing a word, the application supplies the principal whose key should be used to generate the search token. In this case, `kChat` uses the current user principal, `princ_current()`.

Mylar for Users

To obtain the full security guarantees of Mylar, a user must install the Mylar browser extension, which detects tampered code. However, if a site owner wants to protect against attackers who only read server data (as opposed to actively modifying data or installing software at the server), users don’t have to install the extension and their browsing experience is entirely unchanged.

Conclusion

Mylar is a novel Web application framework that enables developers to protect confidential data in the face of arbitrary server compromises. Experimental results show that using Mylar requires few changes to an application, and that the performance overheads of Mylar are modest.

Acknowledgments

This research was supported by NSF award IIS-1065219, by DARPA CRASH under contracts #N66001-10-2-4088 and #N66001-10-2-4089, by Quanta, and by Google.

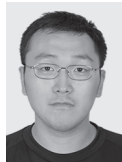
References

- [1] D. Borelli, “The Name Edward Snowden Should Be Sending Shivers Up CEO Spines,” *Forbes*, Sept. 2013: <http://www.forbes.com/sites/realspin/2013/09/03/the-name-edward-snowden-should-be-sending-shivers-up-ceo-spines/>.
- [2] A. Chen, “GCreep: Google Engineer Stalked Teens, Spied on Chats,” *Gawker*, Sept. 2010. <http://gawker.com/5637234/>.
- [3] Google, Inc. User Data Requests—Google Transparency Report: <http://www.google.com/transparencyreport/userdatarequests/>, accessed Sept. 2013.
- [4] MEGA: The Privacy Company: <https://mega.co.nz/#privacycompany>, accessed Sept. 2013.
- [5] Meteor, Inc., Meteor: A Better Way to Build Apps: <http://www.meteor.com>, accessed Sept. 2013.
- [6] OpenID Foundation, OpenID: <http://openid.net>, accessed Sept. 2013.
- [7] R. A. Popa, E. Stark, J. Helfer, S. Valdez, N. Zeldovich, M. F. Kaashoek, and H. Balakrishnan, “Building Web Applications on Top of Encrypted Data Using Mylar,” *Proceedings of the 11th Symposium on Networked Systems Design and Implementation (NSDI ’14)*, Seattle, WA, Apr. 2014.
- [8] Cryptocat: <https://crypto.cat/>, accessed Sept. 2013.
- [9] J. Tudor, “Web Application Vulnerability Statistics,” June 2013: <http://www.contextis.com/services/research/white-papers/web-application-vulnerability-statistics-2013/>.

cTPM

A Cloud TPM for Cross-Device Trusted Applications

CHEN CHEN, HIMANSHU RAJ, STEFAN SAROIU, AND ALEC WOLMAN



Chen Chen is a PhD student in the Department of Electrical and Computer Engineering at Carnegie Mellon University. He is advised by Professor Adrian Perrig. He holds BS degrees in applied math and automation from Tsinghua University. His research interests include network security, secure network architecture, and trusted computing chenche1@andrew.cmu.edu



Himanshu Raj is a principal software engineer in the Azure Cloud Networking group at Microsoft, Silicon Valley. He is interested in systems, networking, and security. Before joining Azure, Himanshu worked in the XCG Lab at Microsoft Research, where he focused on systems solutions for trusted computing. Himanshu holds a PhD from Georgia Institute of Technology, Atlanta, and a Bachelor of Technology from Indian Institute of Technology, Guwahati. rhim@microsoft.com



Stefan Saroiu is a senior researcher in the Mobility and Networking Research Group at Microsoft Research in Redmond, WA. Stefan's research interests span mobile systems, distributed systems, and computer security. Before joining MSR in 2008, Stefan spent three years as an assistant professor at the University of Toronto and four months at Amazon.com, where he worked on the early designs of their new shopping cart system (aka Dynamo). Stefan finished his PhD at the University of Washington where he was co-advised by Steve Gribble and Hank Levy. ssaroiu@microsoft.com

Current Trusted Platform Modules (TPMs) are ill-suited for use in mobile services because they hinder sharing data across multiple devices seamlessly, they lack access to a trusted real-time clock, and their non-volatile storage performs poorly. We present cloud TPM (cTPM), an extension of the TPM's design, to address these problems. cTPM includes two features: a cloud seed shared between the TPM and the cloud, and remote storage in addition to the on-chip storage. cTPM allows the cloud to create and share TPM-protected keys across multiple devices, to manage a portion of a mobile device's TPM storage, and to provide each TPM with a trusted real-time clock and with high-performance non-volatile storage.

Introduction

People are increasingly relying on more than one mobile device. Recent news reports estimate that the average US consumer owns 1.57 mobile devices; Singapore has 7.8 million mobile devices, which translates to 150% mobile penetration; and the average Australian will own five mobile devices by 2040. Given this trend, mobile platforms are recognizing the need for "cross-device" functionality that automatically synchronizes photos, videos, apps, data, and even games across all devices owned by a single user.

Mobile platforms, such as laptops, smartphones, and tablets, are increasingly incorporating trusted computing hardware. For example, Google's Chromebooks use TPM to prevent firmware rollbacks and to store and attest a user's data encryption keys. Windows 8 (on tablets and phones) offers BitLocker full-disk encryption and virtual smart cards using TPMs. Recent research leverages TPMs to build new trusted mobile services [3, 7], trusted cloud services [8], and operating systems [9].

Unfortunately, these two trends may be at odds: Trusted hardware, such as the TPM, does not provide good support for cross-device functionality. Specifically, we have identified three limitations in the TPM design that hamper building cross-device trusted applications.

Limitation 1: Cross-Device Data Sharing. Current TPM abstractions offer guarantees about one single computer, and TPM's hardware protection mechanisms do not extend across devices. For example, TPM's owner domain provides an isolation mechanism for only a single TPM. A new owner of the TPM cannot access the previous owner's TPM-protected secrets. When the same user owns two different TPMs (on two different devices), the owner domains of each TPM remain isolated and cannot jointly offer hardware-based protection of the user's keys and data. Thus, mobile services cannot rely on TPMs alone to enable secure data sharing across devices. While, in theory, migrating a TPM-protected key from one TPM to another is possible, in practice, it requires using secure execution mode (SEM), such as Intel's TXT and AMD's SEM, and trusting a third-party PKI. Such requirements are very challenging. Our NSDI paper [2] describes in more depth the nature of these challenges.

Limitation 2: Trusted Clock. Today's TPMs do not offer a trusted real-time clock. Instead, the TPM combines a trusted timer with a secure, volatile counter, which is periodically persisted to the TPM's NV storage. However, this mechanism can keep track of time

cTPM: A Cloud TPM for Cross-Device Trusted Applications



Alec Wolman is a principal researcher in the Mobility and Networking Research Group at Microsoft Research in Redmond, WA. His interests include mobile and wireless systems, distributed systems, and computer security. He received a PhD in computer science from the University of Washington in 2002. Before graduate school, he worked for DEC at the Cambridge Research Lab. alecw@microsoft.com

only when the TPM is running (and not when the platform is powered off). Moreover, after an unclean reboot, the timer is rolled back to the last persisted counter value violating monotonicity. The TPM's timer mechanism solely guarantees that as long as the platform does not reboot, the timer will move forward. As such, it can provide an approximate time-since-boot.

This mechanism is inadequate for offering real-time guarantees that would be useful for offline content access. For example, movie studios already charge a premium to make a movie available on home theaters on the day of release. Although TPMs can provide offline access securely, they cannot offer making the new movie available for watching next Friday at midnight.

Limitation 3: Non-Volatile (NV) Storage. The TPM's NV storage is inadequate for applications that require frequent writes or require large amounts of trusted storage. For example, previous work [3] has shown that a trusted module offering a monotonic counter and a key solves several problems in distributed systems that stem from participants' ability to equivocate. Unfortunately, even though TPMs offer this functionality, their implementation of NV storage cannot meet the write frequency requirements of distributed systems protocols. The TPM specification dictates the inclusion of monotonic counters, but the spec requires only the ability to increment these counters at a very slow pace (e.g., once every five seconds), which is insufficient for high-event applications such as networked games [3]. Similarly, although the TPM specification mandates access-controlled, non-volatile storage, most implementations provide only 1,280 bytes of NVRAM [7]. These limitations have led researchers to seek alternative designs for trusted devices [3].

Overcoming these limitations requires altering the TPM design, which raises the following question: *Can a small-scale TPM design change overcome these limitations?* Although a clean-slate TPM redesign could provide a variety of additional security properties, there are two pragmatic reasons why a smaller change is preferable. First, TPMs have undergone a decade of API and implementation revisions to reduce the likelihood of vulnerabilities. A clean-slate redesign would demand considerable time and effort to provide a mature code base. Second, TPM manufacturers would more willingly adopt smaller and simpler changes.

To address these limitations, we propose a single, simple modification to the TPM design, called cTPM: equipping the TPM with one primary seed that is shared with the cloud. Sharing the seed with the cloud allows both cTPM and the cloud to generate the same cloud root key. Combining the cloud root key with remote storage lets cTPM: (1) share data via the cloud, (2) have access to a trusted real-time clock, and (3) have access to remote NV storage that supports a large quantity of storage and high frequency writes.

cTPM's design facilitates data sharing. The pre-shared primary seed lets the cloud effectively act as a PKI. The cloud and the device's TPM can use this shared secret to encrypt and authenticate their messages to each other. The identity problem has now been "pushed" to ensuring that the cloud primary seed is shared securely between cTPM and the cloud. This initial sharing step should be done at cTPM manufacturing time when the cTPM's three other primary seeds are provisioned.

The pre-shared primary seed also equips cTPM with a trusted clock using a protocol similar to the Time Protocol described in RFC 868. Once the clock value is obtained from the cloud, cTPM uses its local timer to advance the clock. It has a global variable that dictates how often it should resynchronize the clock; the TPM owner sets this variable whose default value is one day.

Finally, cTPM uses the cloud for additional NV storage to overcome TPM NV storage limitations. There are no limits on how much additional NV storage the cloud can provide to a single cTPM. A portion of the physical cTPM chip's RAM is thus allocated as a local cache

cTPM: A Cloud TPM for Cross-Device Trusted Applications

for the cloud-backed NV storage. The performance of cTPM cloud-backed NV storage exceeds that of the TPM because TPM NV accesses are no longer needed.

Background

TPM Primer. At manufacturing time, TPM chips are provisioned with a couple of public/private key-pairs for cryptography (i.e., digital signatures and asymmetric encryption). The TPM design guarantees that the private keys of these root key-pairs never leave the TPM, thereby reducing the possibility of compromise. TPMs can also generate public/private key-pairs with private keys stored in the TPM's NV storage. However, TPMs have limited NV storage and thus cannot store many such key-pairs.

The TPM specification also mentions that a certificate demonstrating the authenticity of the TPM's embedded key pairs may be provided by the TPM's hardware manufacturer. In our experience, many TPMs (although not all) lack this certificate. The absence of this certificate makes it impossible for a third-party to determine whether a signed statement (e.g., a software attestation) is produced by a valid TPM or by an impersonating entity.

TPMs are equipped with a set of "extend-only" platform configuration registers (PCRs) that are guaranteed to be reset upon a computer reboot. PCRs are primarily used to store fingerprints of a portion of the booting software (e.g., the BIOS, firmware, and OS bootloader); Chromebooks and BitLocker use PCRs in this way.

TPMs can perform cryptographic algorithms for encrypting, authenticating, and attesting data. Implementing functionality beyond that offered by TPMs in a trustworthy manner can be done using secure execution mode, a form of hardware protection offered by x86 CPUs. Intel's secure execution architecture, called Trusted Execution Technology (TXT), offers a runtime environment strongly isolated from other software running on the computer. When invoked, the CPU disables interrupts (to ensure no other software is running), and a small bootloader starts executing. The bootloader then jumps to an address specified by the caller to execute any additional code. Flicker is an earlier project that demonstrated the use of secure execution mode [5].

The TPM spec does not provide minimum performance requirements, and, as a result, today's commodity TPMs are slow and inefficient. TPM vendors have little incentive to use faster but more expensive internal parts when building their TPM chips. This performance handicap has limited the use of TPMs to scenarios that do not require fast or frequent operations. However, no technological constraints prevent a hardware vendor from building a high-performance TPM.

TPM 2.0. The Trusted Computing Group (TCG) is currently defining the specification for TPM version 2.0, the next version of the TPM. TPM 2.0 offers several improvements, including

cryptographic algorithm agility. For example, SHA-2 and elliptic curve cryptography (ECC), in addition to SHA-1 and RSA, are offered by TPM 2.0. TPM 2.0 also provides more PCRs and supports more flexible authorization policies that control access to TPM-protected data. Finally, TPM 2.0 provides a reference implementation, while TPM 1.2 provides only an open-source implementation developed by a third party.

In TPM 2.0, three entities can control the TPM's resources: the platform manufacturer, the owner, and the privacy administrator. The TPM 2.0 spec *control domain* refers to the specific resources that each entity controls. The platform firmware control domain overseen by the platform manufacturer updates the TPM firmware as needed. The owner control domain protects keys and data on behalf of users and applications. The privacy administrator control domain safeguards privacy-sensitive TPM data. Each TPM 2.0 control domain has a primary seed, which is a large, random value permanently stored in the TPM. Primary seeds are used to generate symmetric/asymmetric keys and proofs for each control domain.

Trust Assumptions and Threat Model

Trusting the Cloud

All the new cTPM functionality associated with the cloud domain assumes the cloud is trustworthy and not compromised by malware. Although everyone may not agree with this assumption, cloud providers have more incentives and resources to monitor and eliminate malware than average users. Security-conscious cloud providers could use secure hypervisors with a small TCB [4], narrow interfaces [6], or increased protection against cloud administrators [10].

Whether using a TPM or not, a cloud compromise would already affect the security of a mobile service relying on the cloud for its functionality. However, even if the cloud were compromised, all secrets protected by the TPM-specific control domains, other than the cloud domain, would remain secure. For example, all device-specific secrets protected in the owner's control domain (i.e., using TPM's SRK) would remain uncompromised.

Threat Model

Our threat model resembles that of traditional TPMs: All software attacks are in scope (including side-channel attacks) because cTPM is isolated from the host platform and can therefore provide its security guarantees even if the host were compromised (e.g., infected with malware). However, physical attacks and DoS attacks in which the (untrusted) operating system or applications deny access to the cTPM or to the cloud are out of scope.

Another class of attacks specific to the cTPM stems from our use of remote cloud storage. The (untrusted) OS could drop,

cTPM: A Cloud TPM for Cross-Device Trusted Applications

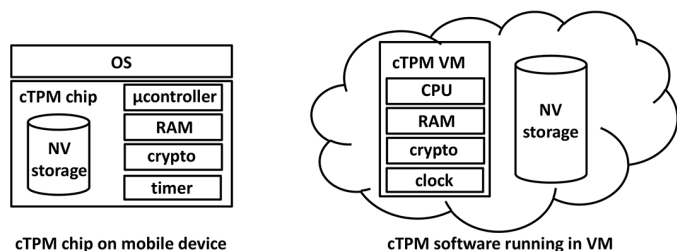


Figure 1: cTPM high-level architecture

corrupt, or reorder messages from the cloud. Even worse, it could delay messages from the cloud in an effort to serve stale data to the TPM. All such attacks are in scope and addressed by cTPM; for example, to ensure freshness, cTPM uses a local timer to time out any pending requests not yet serviced.

cTPM High-Level Design

The cTPM design extends the TPM 2.0 by its ability to share a primary seed with the cloud and to access cloud-hosted non-volatile storage. This section describes the high-level design and the challenges we encountered when implementing these features. While our description is TPM 2.0-specific, our changes could be equally applied to TPM 1.2.

Cross-Device Usage Model

Each device has a unique cTPM with a unique primary seed shared with the cloud and used to derive additional keys. All devices registered with the same owner have their keys tied to the owner's credentials. The cloud could then offer cTPM services that create a shared key across all devices owned by the same user. For example, when "bob@hotmail.com" calls this service, a shared key is automatically provisioned to the cTPM on each of Bob's devices. This shared key can bootstrap the data-sharing scenarios described by this paper.

Architecture

cTPM consists of two different components: one running on the device and the other in the cloud. Both components implement the full TPM 2.0 software stack with the additional cTPM features. This ensures that all cloud operations made to the cTPM strictly follow TPM semantics, and thus we do not need to re-verify their security properties. On the device-side, the cTPM software stack runs in the TPM chip, whereas the cloud runs the cTPM software inside a VM. On the cloud-side, the NV storage is regular cloud storage, and the timer offers a real-time clock function. The cloud-side cTPM software reads the local time upon every initialization and uses NTP to synchronize with a reference clock. When running in the cloud, cTPM resources (e.g., storage, clock) need not be encapsulated in hardware, because the OS running in the VM is assumed to be trusted. In

contrast, the device's OS is untrusted, and thus the cTPM chip itself must be able to offer these resources in isolation from the OS. Figure 1 illustrates the high-level architecture of the cTPM.

Shared Cloud Primary Seed

Upon starting, the local cTPM checks whether a shared cloud primary seed is present. If not, it disables its new cTPM functionality and all commands associated with it. A cTPM is provisioned with a cloud primary seed via a proprietary interface available only to the device manufacturer.

The cTPM uses the cloud primary seed to generate an asymmetric storage root key, called the *cloud root key* (CRK), and a symmetric communication key, called the *cloud communication key* (CCK). Both keys are derived from the cloud primary seed based on use of an approved key derivation function. These key derivations occur twice: once on the device-side and once on the cloud-side of the cTPM. Because the key derivations are deterministic, both the device and the cloud end up with identical key copies. The CRK's semantics are identical to those of the *storage root key* (SRK) controlled by the TPM's owner domain. The CRK encrypts all objects protected within the cloud control domain (similar to how SRK encrypts all objects within the owner domain). The CCK is specific to the cloud domain, and it protects all data exchanged with the cloud.

Secure Asynchronous Communication

cTPM cannot directly communicate with the cloud. Instead, it must rely on the OS for all its communication needs. Because the OS is untrusted, cTPM must protect the integrity and confidentiality of all data exchanged between the cTPM and the cloud-backed storage, as well as protect against rollback attacks. The OS is regarded merely as an insecure channel that forwards information to and from the cloud.

In addition to ensuring security, cTPM must support asynchronous communication between the local cTPM and the cloud. Today, the TPM is single-threaded, and all TPM commands are synchronous. When a command arrives, the caller blocks and the TPM cannot process any other commands until the command terminates. Making cTPM cloud communication synchronous would lead to unacceptable performance. For example, consider issuing a cTPM command that increments a counter in cloud-backed NV storage. This command would make the TPM unresponsive and block until the increment update propagates all the way to the cloud and the response returns to the local device.

Instead, we chose to make cloud communication asynchronous. Whenever a command that needs access to remote NV is received, cTPM returns to the caller an encrypted blob that needs to be sent remotely. The caller must send this blob to the cloud; if the cloud accepts the blob, it returns another encrypted

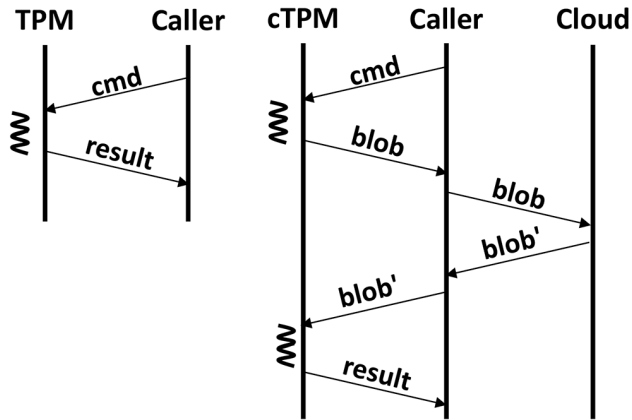


Figure 2: The sequence of steps for issuing a synchronous command (left) versus an asynchronous command (right). The cTPM remains responsive to other commands while the caller relays the blob to the cloud.

blob reply to the caller. The caller then passes this reply to the cTPM, at which point the command completes. cTPM remains responsive to all other commands during this asynchronous communication with the cloud. Figure 2 illustrates these steps and contrasts them with a traditional simple TPM command. All cTPM commands that do not require access to remote NV storage remain synchronous, similar to TPMs today.

Dealing with Connectivity Loss. Loss of connectivity is transparent to the cTPM because all network signaling and communication is done by the operating system. However, the two-step nature of asynchronous commands requires the cTPM to maintain in-memory state between the steps. This introduces another potential resource allocation denial-of-service attack: A malicious OS could issue many asynchronous commands that cause the cTPM to fill up its RAM. Also, as mentioned in our threat model, an attacker could launch a staleness attack whereby artificial delays are introduced in the communication with the cloud.

To protect against these attacks, cTPM maintains a *global read timeout* (GRT) value. Whenever an asynchronous request is issued, cTPM starts a timer set to the GRT. Additionally, to free up RAM, cTPM scans all outstanding asynchronous commands and discards those whose timers have expired. The GRT can be set by the cTPM's owner and has a default value of five minutes.

Cloud-Backed NV Storage

At a high level, the cloud-backed NV storage is just a key-value store whose keys are NV indices. Accessing the remote NV index entries requires the OS to assist with the communication between the cTPM and the cloud. These operations are thus asynchronous and follow the same two-step model described in Figure 2. However, the remote nature of these NV indices raises additional design challenges.

Local NV Storage Cache. Remote NV entries can be cached locally in the cTPM's RAM. To do so, we add a time-to-live (TTL) to locally cached NV entries. The TTL specifies how long (in seconds) the cTPM can cache an NV entry in its local RAM. Once the TTL expires, the NV index is deleted from RAM and must be reloaded from the remote cloud NV storage with a fresh, up-to-date copy. The TTL controls the tradeoff between performance and staleness for each NV index entry. Furthermore, the local storage cache is *not persistent*—it is fully erased each time the computer reboots.

For writes, the local cache's policy is *write back*, and it relies on the caller to propagate the write to the cloud NV storage. A cTPM NV write command updates the cache first and returns an error code that indicates the write back to the NV storage is pending. The caller must initiate a write protocol to the cloud NV. If the caller fails to complete the write back, the write remains volatile, and the cTPM makes no guarantees about its persistence.

Trusted Clock. In cTPM, the trusted clock is an NV entry (with a pre-assigned NV index) that only the cloud can update. The local device can read the trusted clock simply by issuing an NV read command for this remote entry. Reading the entry is subject to a timeout much stricter than the regular GRT, called the *global clock timeout* (GCT). The trusted clock NV entry is cached in the on-chip RAM. In this way, the cTPM always has access to the current time by adding the current timer tick count to the synchronization timestamp (ST) of the clock NV entry.

Detailed Design and Implementation

This section provides more detail on the cTPM's design and implementation. We describe how the cTPM shares TPM-protected keys between the cloud and the device, and we present the changes made to support NV reads and writes. We also describe the cloud/device synchronization protocol and the new TPM commands we added to implement synchronization.

Sharing TPM-Protected Keys

The TPM 2.0 API facilitates the sharing of TPM-protected keys by decoupling key creation from key usage. TPM2_Create(), a TPM 2.0 command, creates a symmetric key or asymmetric key-pair. The TPM creates the key internally and encrypts any private (or symmetric) keys with its storage key before returning them to the caller. To use the key, the caller must issue a TPM2_Load() command, which passes in the public storage key and the encrypted private (or symmetric) key. The TPM decrypts the private key, loads it in RAM, and can begin to encrypt or decrypt using the key.

This separation lets cTPM use cloud-created keys on the local device to gain two benefits. First, key sharing between devices becomes trivial. The cloud can perform the key sharing protocol

cTPM: A Cloud TPM for Cross-Device Trusted Applications

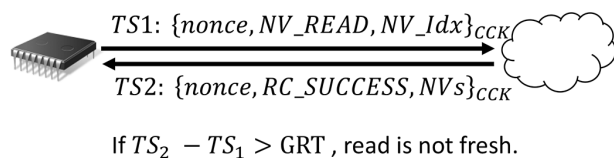


Figure 3: Synchronization protocol: pull NV entry from cloud-backed NV storage

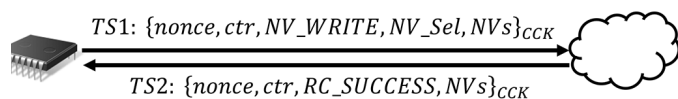


Figure 4: Synchronization protocol: push NV entry to cloud-backed NV storage

between two cTPM VMs. Unlike TPM 2.0, this protocol does not need to use a PKI, nor does it need to run in SEM. Once a shared key is created, both mobile devices can load the key in their chips separately by issuing TPM2_Load() commands. Second, key creation can be performed even when the mobile device is offline, greatly simplifying creating a shared key.

Accessing Cloud NV Storage

The cTPM maintains a local cache of all reads and writes made to the cloud NV storage. A read returns a cache entry, and a write updates a cache entry only. The cTPM does not itself update remote cloud NV storage; instead, the caller must synchronize the on-chip RAM cache with the cloud NV storage. This is done using a synchronization protocol.

Read Cloud NV. Upon an NV read command, the corresponding NV entry is returned from the local cache. If not found, cTPM returns an error code. The caller must now check the remote NV; to do so, it needs to initiate a pull synchronization operation (described in the next section) to update the local cache. After synchronization completes, the caller must reissue the read TPM command, which will now be answered successfully from the cache.

Write Cloud NV. An NV write command first updates the cache and returns an error code that indicates the write back to the remote NV storage is pending. The caller must initiate a push synchronization operation to the cloud NV (see the next section). If the caller fails to complete the write back, the write remains volatile, and cTPM makes no guarantees about its persistence.

Synchronization Protocol

The synchronization protocol serves to: (1) update the local cache with entries from the cloud-backed NV storage (for NV reads) and (2) write updated cache entries back to the cloud-backed NV storage (for NV writes). On the device side, the

caller performs the protocol using two new commands, TPM2_Sync_Begin() and TPM2_Sync_End(). These commands take a parameter called direction, which can be set to either a pull or push to distinguish between reads and writes. All messages are encrypted with the cloud communication key (CCK), a symmetric key.

Pull from Cloud-Backed NV Storage. The cTPM first records the value of its internal timer and sends a message that includes the requested NV index and a nonce. The nonce checks for freshness of the response and protects against replay attacks. Upon receipt, the cloud decrypts the message and checks its integrity. In response, the cloud sends back the nonce together with the value corresponding to the NV index requested. The cTPM decrypts the message, checks its integrity, and verifies the nonce. If these checks are successful, cTPM performs one last check to verify that the response's delay did not exceed its global read timeout (GRT) value. If all checks pass, cTPM processes the read successfully. Figure 3 shows the precise messages exchanged between the cTPM and the cloud to read the remote NV.

Push to Cloud-Backed NV Storage. The protocol for writing back an NV entry is more complex because it must also handle the possibility that an attacker may try to reorder write operations. For example, a malicious OS or application can save an older write and attempt to reapply it later, effectively overwriting the up-to-date value. To overcome this, the protocol relies on a secure monotonic counter maintained by the cloud. Each write operation must present the current value of the counter to be applied; thus, stale writes cannot be replayed. cTPM can read the current value of the secure counter using the previously described pull protocol. Figure 4 shows the precise messages exchanged between the cTPM and the cloud to write a remote NV entry. Note that reading the secure counter need not be done on each write because the local cTPM caches the up-to-date value in RAM.

Protocol Verification. We verified our protocols' correctness using an automated theorem prover, ProVerif [1], which supports the specification of security protocols for distributed systems in concurrent process calculus (pi-calculus). We specified our synchronization protocol—both pull and push—in 98 lines of pi-calculus code. ProVerif verified the security of our protocols in the presence of an attacker with unrestricted access to the OS, applications, or network. The attacker could intercept, modify, replay, and inject new messages into the network (similar to the Dolev-Yao model).

Conclusion

The traditional TPM design fails to meet the requirement of today's cross-device trusted applications. This paper introduces cTPM, a cloud-enhanced design change to the traditional TPM design that enables: (1) cryptographic keys and data to be shared

cTPM: A Cloud TPM for Cross-Device Trusted Applications

across a user's many devices, (2) a trusted clock synced with the cloud, and (3) high-performance NV storage of unlimited size. cTPM accomplishes these goals by only adding a cloud seed shared between the device and the cloud. Together with the asynchronous communication channel, the seed allows cTPM to interact with the cloud to provide better support for cross-device trusted applications.

References

- [1] B. Blanchet, "An Efficient Cryptographic Protocol Verifier Based on Prolog Rules," in *Proceedings of the Computer Security Foundations Workshop*, 2001.
- [2] C. Chen, H. Raj, S. Saroiu, and A. Wolman, "cTPM: A Cloud TPM for Cross-Device Trusted Applications," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation* (USENIX Association, 2014), pp. 187–201.
- [3] D. Levin, J. R. Douceur, J. R. Lorch, and T. Moscibroda, "TrInc: Small Trusted Hardware for Large Distributed Systems," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation* (USENIX Association, 2009), pp. 1–14.
- [4] J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, and A. Perrig, "TrustVisor: Efficient TCB Reduction and Attestation," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2010.
- [5] J. M. McCune, B. Parno, A. Perrig, M. K. Reiter, and H. Isozaki, "Flicker: An Execution Infrastructure for TCB Minimization," in *Proceedings of EuroSys*, Glasgow, UK, 2008.
- [6] A. Nguyen, H. Raj, S. Rayanchu, S. Saroiu, and A. Wolman, "Delusional Boot: Securing Cloud Hypervisors without Massive Re-engineering," in *Proceedings of EuroSys*, Bern, Switzerland, April 2012.
- [7] B. Parno, J. R. Lorch, J. R. Douceur, J. Mickens, and J. M. McCune, "Memoir: Practical State Continuity for Protected Modules," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, 2011.
- [8] N. Santos, R. Rodrigues, K. P. Gummadi, and S. Saroiu, "Policy-Sealed Data: A New Abstraction for Building Trusted Cloud Services," in *Proceedings of the 21st USENIX Security Symposium*, Bellevue, WA, 2012.
- [9] E. G. Sirer, W. de Bruijn, P. Reynolds, A. Shieh, K. Walsh, D. Williams, and F. B. Schneider, "Logical Attestation: An Authorization Architecture for Trustworthy Computing," in *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP)*, Cascais, Portugal, 2011.
- [10] F. Zhang, J. Chen, H. Chen, and B. Zang, "CloudVisor: Retrofitting Protection of Virtual Machines in Multi-Tenant Cloud with Nested Virtualization," in *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP)*, Cascais, Portugal, 2011.

Interview with Steve Bellovin

RIK FARROW



Steven M. Bellovin is a professor of computer science at Columbia University, where he does research on networks, security, and, especially, why the two don't get along, as well as related public policy issues. In his spare professional time, he does some work on the history of cryptography. He joined the faculty in 2005 after many years at Bell Labs and AT&T Labs Research, where he was an AT&T Fellow. He received a BA degree from Columbia University, and an MS and PhD in computer science from the University of North Carolina at Chapel Hill. As a graduate student, he helped create Netnews; for this, he and the other perpetrators were given the 1995 USENIX Lifetime Achievement Award (The Flame). Bellovin has served as chief technologist of the Federal Trade Commission. He is a member of the National Academy of Engineering and is serving on the Computer Science and Telecommunications Board of the National Academies, the Department of Homeland Security's Science and Technology Advisory Committee, and the Technical Guidelines Development Committee of the Election Assistance Commission. He has also received the 2007 NIST/NSA National Computer Systems Security Award.

bellovin@acm.org



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

I first met Steve at an early USENIX Security symposium, and over the years, that's where we would often meet. I mostly knew Steve through the research he had published, as well as the book he co-authored with Bill Cheswick [1]. But I really didn't know much more than that Steve and Bill had met while they were both working at Bell Labs in Murray Hill, New Jersey. When we met at conferences, we'd mostly talk about what was happening at the time, not the past.

I decided that it was time to ask Steve a few questions about his past, as well as get a better understanding of the path that has led him from being a student to being a professor, with many interesting adventures along the way.

Rik: What I am curious about is how you became involved with Internet security. I know you worked on EKE and helped write Netnews while a graduate student, but that doesn't really tell me how you got to writing a popular and important book on firewalls plus multiple security-related RFCs over the next 10 years.

Steve: My background, ultimately, is as a sysadmin. I learned programming in my sophomore year of high school, when that was very rare. The students who knew how to program—about half a dozen of us—ran the machine, an IBM 1130, at Stuyvesant High School in NYC, without interference from the teachers. We knew more about it than they did, and they didn't resent us for it. I was curious how a “kernel” (to use today's terminology) worked, so I wrote a disassembler to study the OS.

Rik: How is it that your high school had a computer, back when computers were truly rare?

Steve: Stuyvesant is an examination-entrance NYC public high school that's for students interested in math and science. Someone there talked someone into believing Stuyvesant needed a computer...after all, the “competition,” Bronx High School of Science, already had one.

When I got to college, my part-time jobs were all systems programming. My last two years, I worked at the City College of New York Computer Center at the time when it was the central site for all of the City University of New York; we had a smallish IBM mainframe that was used for academic and administrative computing. My college years are also when I learned networking—IBM Bisync in those days. At CCNY, I caught my first two hackers; they were poking around at the administrative side of things. After studying their card decks(!), I hired one and referred the other to the dean.

Rik: I remember card decks all too well. What did you focus on at grad school?

Steve: I was mostly interested in programming languages. For breadth, I did a theory dissertation on proving the output of compilers correct; although I wasn't doing security then, the dissertation actually turned out to be security-relevant. I learned UNIX and kept up my systems programming and kernel-hacking skills (writing device drivers, studying how things worked, etc.).

Netnews was a separate story, but the original impetus was the need for a convenient mechanism for administrative announcements. It's also when I first got involved with USENIX—I

Interview with Steve Bellovin

was at the meeting where, for trademark reasons, they had to change the name from the “Unix User’s Group.” I also started studying TCP/IP around 1980 or 1981, when I got a copy of the “Internet Protocol Transition Handbook.”

Rik: And when you started working at Bell Labs, you also worked on networking?

Steve: When I got to Bell Labs, I became responsible for 1.5 of the first 3 lengths of thick Ethernet cable in the entire company: my lab, another lab, and the “backbone” cable connecting us. That got me more heavily involved with TCP/IP, which I helped bring to the rest of Bell Labs.

There were occasional network outages due to misconfigurations. In those days, there were no dedicated routers; you simply stuck another Ethernet board into your VAX and used it as a router. Multihomed 4.2bsd hosts would forward packets by default—a misfeature, but few people realized that at the time. Many of the outages were routing-related, and I realized that what could happen by accident could happen intentionally. One cannot be a good sysadmin without worrying about security. I started worrying about routing security and address-based authentication. Adding to that, Robert T. Morris interned at the Labs and invented sequence number attacks, so I had more to worry about.

In addition, there were ongoing attacks from the outside against Bell Labs [1]. I was one of several people who detected the attacks—I’d added a cron job that scanned the UUCP log files for attempts to snarf/etc/passwd. I had nothing to do with the subsequent investigation.

Rik: Perhaps you could expand on the problems with multihomed 4.2bsd, since these also occurred with SunOS. I believe this is one of the reasons the Morris Worm was so successful: People used Suns and VAXen as routers, and they shared directory structure and commands like Sendmail.

Steve: No, I don’t think it was a factor in the worm’s spread; that was more a case of monocultures and no filtering. The issue was more subtle: It was a confusion of the difference between a multihomed host and a router, which meant that topologies were richer than intended. The folks at Berkeley who wrote 4.2bsd were very good UNIX and kernel hackers, but arguably didn’t have a good grasp of some of the more subtle points of TCP/IP. It took RFCs 1122 and 1123 to sort that out.

Rik: I recall just how unpleasantly mysterious TCP/IP was. I had been a UUCP expert, but when it came to assigning IP addresses on a private network, no guidance existed in the late ’80s. I, like many others, wound up using an address that appeared in Sun documentation, so we could use the thin Ethernet cables and connectors that came with Sun 3 workstations.

Steve: Right, that and a related issue were some of the things that caused trouble. Sun implemented something that was what today we’d call “zeroconf”—if you didn’t set an IP address, the software would pick one via an algorithm and protocol known only to other Suns. When this happened on the backbone, it meant that someone would suddenly grab .1 on that net, and .1 was really the router to other locations in the Labs. Then I asked myself, “What would happen if someone did that maliciously?” and my career took a sharp turn.

Worrying about the TCP/IP issues led to my first major paper, on TCP/IP protocol-level security [2]. The authentication and routing issues led me to think more about crypto; in addition, sometime in the 1980s my wife gave me a copy of the hardcover of Kahn’s *The Codebreakers*. Looking at Kerberos and thinking about password guessing led me to worry about guessing attacks on the initial sequence used to get a ticket-granting ticket. Mike Merritt and I talked about it, and I worried about it for several months. Finally, my mind wandered while I was sitting in a really boring talk and I had an inspiration; the result was EKE. Today, I use this story today to motivate students to come to class, no matter how boring it is; they might invent something while I’m droning on.

By this point, I was doing network security full time. Ches and I dealt with Berferd in 1991. A chance meeting with him on a train ride to Baltimore for the USENIX Security conference led to us agreeing to write a book. John Wait, the eventual editor of the book, happened to pay his routine annual visit to me shortly afterwards. He always wanted to know whether I was interested in writing a book; I always declined, because I didn’t have anything to say. This time, I did have something to say.

Firewalls were a pretty obvious path for network security then, given both the Presotto/Cheswick design of the AT&T gateway to the Internet and my statement about topological defenses in the TCP/IP protocol insecurity paper. It was easy around then to be one of the top network security people because there were so few, but that meant I was noticed. I was invited to be on the IPng directorate; that got me involved with the IETF, so I wrote RFCs, etc.

Rik: With your early interest in IPv4 routing, and your participation in the IETF, did that lead to any advances in improving the security of routing protocols? Or have any influence on IPng?

Steve: I was one of the people responsible for IPv6 requiring IPSec in all implementations; this is precisely because of all the risks from address-based authentication. Routing security is still an open issue, although I was one of the people who did the work leading to the IETF’s BGPSEC working group.

Rik: You once mentioned to me that one reason for the lack of routing security is the convenience of the current state of affairs

for nation states who might wish to route traffic past points where they can intercept it. I am of course paraphrasing, as you said this quickly in passing. But BGP is still just tables of information exchanged between routers with no signatures to verify routing assertions.

Steve: I don't think I said that that's one reason, but it's certainly something that many countries like and exploit now. There have been incidents, such as when Pakistan decided to block YouTube internally and affected global routing [3].

Rik: I would appreciate it if you would say more about the lack of robust security for the routing infrastructure. I've often assumed this is what the L0pht members were referring to when they claimed they could "bring down the Internet in 30 minutes," given just how much trouble we've experienced from accidents (routing of a lot of the Internet to a single small ISP in Florida as an example [4]).

Steve: Routing and the DNS [5]. A 1999 National Academies study committee that I was a member of called routing and DNS the two main trouble spots on the Internet.

There have been two main reasons why BGPSEC hasn't happened yet. First, it's expensive: Lots of routers will have to be replaced by ones with a lot more RAM and a lot more CPU power to do signing and signature verification. Second—and this is the interesting one—it creates new failure modes, and some of these failure modes have political components.

For BGPSEC to work, you MUST have a PKI for IP addresses. A failure at any node in the path from you to the root means that you won't have a good certificate, which in turn means that you'll be off the air. Worse yet, this PKI is inherently a tree structure, i.e., every node is a monopoly, and monopolies don't have particularly much market pressure to make them behave efficiently or to provide good customer service. Also, any node is susceptible to pressure or compulsion by its government: "Revoke this address space certificate under penalty of law." Today, ISPs work by trusting each other on such issues; BGPSEC will require correct technical functioning at all levels of the PKI.

Rik: In your 2003 statement before the DHS subcommittee [6], you wrote that today's operating systems are far more reliable than those used a generation ago. They are also far more complex. What do you think about research toward building partitioned kernels, such as the seL4 microkernel, or the work being done by Robert Watson and others to build a system with efficient hardware segment registers for enforced separation both within applications and at the OS layer?

Steve: Well, strong walls are something we're pretty good at. The problem is that the components have to talk to each other, which implies gates, and these gates have policies attached. That's what we're lousy at: specifying and implementing the gates and their

policies. More walls can lead to higher assurance, which is good, but it's not really the solution. My overarching research goal is to understand how to divide a system into walls-separated components in the proper way [7].

Rik: While reading an article you co-authored [8], I learned that call detail records (CDRs), which are described as call-metadata, also cover information that's included in email headers and can be collected without requiring a search warrant. It seems that CDRs for email provide a lot more information than just the caller and the callee's number, date, and length of call.

Steve: CDRs for mobile devices give approximate location to a pretty fine granularity. CDRs for wireline devices give the phone number, which for ordinary PSTN is pretty closely tied to an address. All CDRs have call length; most have caller and callee. Email headers have more or less that; in particular, the first "Received:" line generally gives the sender's IP address, which is a decent clue to location for non-mobile devices. There are two exceptions: if you use cryptographic tunnels (including, but not limited to, VPNs) or if you use Gmail. Gmail strips all that off—Google knows, but the recipient doesn't.

Rik: As members of research or IT communities, what should we be doing to encourage greater privacy?

Steve: First and foremost, don't collect data you don't need. If you do need it for immediate operational purposes (e.g., mail logs), discard it when you don't need it, or perhaps hash some of the fields.

Second, consider what privacy-preserving options might exist in the systems we design. Take the "Message-ID:" header as defined in RFC 5322:

The message identifier (msg-id) itself MUST be a globally unique identifier for a message. The generator of the message identifier MUST guarantee that the msg-id is unique. There are several algorithms that can be used to accomplish this. Since the msg-id has a similar syntax to addr-spec (identical except that quoted strings, comments, and folding white space are not allowed), a good method is to put the domain name (or a domain literal IP address) of the host on which the message identifier was created on the right-hand side of the "@" (since domain names and IP addresses are normally unique), and put a combination of the current absolute date and time along with some other currently unique (perhaps sequential) identifier available on the system (for example, a process id number) on the left-hand side. Though other algorithms will work, it is RECOMMENDED that the right-hand side contain some domain identifier (either of the host itself or otherwise) such that the generator of the message identifier can guarantee the uniqueness of the left-hand side within the scope of that domain.

Interview with Steve Bellovin

How about making the part to the right of the @ the SHA-256 hash of the domain name? It's just as unique and doesn't leak information. Yes, you can brute force it—if you know the name of all original sending hosts. Is that example too contrived? The Canadian Privacy Commissioner's report on the TJX hack slapped them down for storing driver's license numbers instead of a hash thereof.

As a researcher, the problem is easier to state: Where is privacy lost, and what technically can be done?

Rik: In your 2010 undergraduate “Computers and Society” course, you point out that voluntary surrender of data can lead to secondary use of that data. Many people are happy to share information about their everyday lives, as well as their likes (and dislikes), with social media, but even using a public email account like Gmail results in the sharing of personal information.

Our choices there do not appear to be good: Participate or don't participate. Encrypting email doesn't work without a simple way of securely sharing keys. What do you suggest?

Steve: There are three things. One, of course, is education. Second is research on things like key distribution and easier-to-use encryption. I think this can be done decently well; in fact, I have some projects going on now on that topic. Third is law or regulation, concepts that I, at least, am not allergic to. I do think that we're better off with use restrictions rather than collection restrictions, but in the privacy community I think I'm in the minority on that.

Why do I prefer use restrictions? Some data has to be collected for operational reasons—I've been Postmaster; I know how important mail logs are—and some data, such as health records, can be used for exceedingly important purposes that don't violate anyone's privacy. The risk is that today's rules will be ignored or will be changed for a compelling-enough—or currently compelling-enough—reason (e.g., the misuse of the 1940 census records to aid in interning the Japanese).

Rik: In your presentation about your year as the chief technologist at the FTC [9], you explain that the FTC is reactive in how they can act. For example, if a company promises to keep data secure, but has inadequate technical controls, the FTC has seen this as “deceptive and/or unfair” practices, and can take the perpetrator to court. Most companies sign consent orders, and companies that fail to improve can then be fined. But one company, Wyndham, which had lost data three times in two years, has decided to fight back. What's happening with Wyndham? You said that Wyndham wants a ruling that would limit the FTC's ability to regulate in this area.

Steve: So that's a very interesting question. Wyndham's basic position is that since the FTC has never issued any rules, they

don't know what standard they should meet, so the enforcement is unfair. About a month or so ago, the judge finally handed down her ruling, completely rejecting Wyndham's arguments. Naturally, they're going to appeal to the Second Circuit.

Another company, LabMD, filed similar objections. They didn't survive the process; they went bankrupt. A week or two ago, an administrative law judge—part of the FTC, but organizationally independent—was very, very critical of the FTC, but just this week ruled against LabMD despite that. I haven't had a chance to look at the opinion, so I don't know the grounds; it might have been hyper-technical rather than substantive.

Rik: How did you get involved in the legal realm?

Steve: The short answer is that I've always been interested in law and policy. I did (minor) campaign work as a teenager; in college, I took a constitutional law class because it was interesting. I was one of very few people in that class who wasn't intending to go to law school.

About 20 years ago, I started working with legal issues professionally. These were the days of the Clipper Chip, the Crypto Wars, and the bills that would become the DMCA and CALEA. Matt Blaze and I were able to work with the AT&T policy people and persuade them that the things we wanted for privacy reasons were in the company's interest—and, of course, any company wants to avoid government regulation, so that wasn't that hard.

In the Berferd incident, the lawyers made us kick him off the machine, so I started wondering about liability. When we did the *Firewalls* book, I threw in a chapter on the legal aspects. Unlike most of the book, which was joint work, that chapter was all mine. I got an attorney (who later went on to become second in charge of DoJ's computer crime section) to teach me the basics; I was also assisted by one of the AT&T patent attorneys. Things grew from there.

Around 1995 or so, a Fordham law professor spent his sabbatical in my department at Murray Hill. I still work with him on tech and law. Basically, the need was there and I was interested. There was never any danger, way back when, of me going to law school; I liked computers too much. But I was always interested, and over the years, I've done more and more of it professionally.

Rik: After working for many years at the Labs, you moved to Columbia. Can you tell us why you decided to become a professor?

Steve: I left AT&T Labs Research for a number of reasons. One was simply that it was time to do something different. I'd been there for more than 20 years, I had a great time, and I had management that supported me. But I'd always wanted to teach, and I decided that it was time. I really enjoy teaching, and a lot of

what I've done—talks, writing papers, the *Firewalls* book—is just another form of teaching. The other factor, of course, was that I was not sanguine about the future of research there, and when a good opportunity arose I decided to take it. I'd received other offers from other universities in the past, but it wasn't time to leave.

Sadly, I was right about AT&T Labs. It's not the place it was; from what I hear there's not nearly as much freedom to do research and to publish, and many of the very best people have left or been laid off.

If I wanted to teach, why didn't I do that straight out of grad school? I did—and do—dislike everything to do with getting grants. In fact, it's been worse than I had expected. On the other hand, some of the benefits—the ability to work and speak freely on public policy issues, the freedom to do things like write law review and history of cryptography articles, and, above all, access to a wonderful research library—have been greater than I had anticipated. AT&T was a wonderful place, but I don't regret moving on—it was time.

Rik: Finally, why did you start writing about technical history?

Steve: During a 1993 conference, Matt Blaze and I heard an ex-NSA cryptologist say that the needs of Permissive Action Links (PALs)—the cryptographic combination locks on nuclear weapons—led the NSA to invent public key cryptography in the 1960s. Now, PALs are supposedly impossible to bypass, and a security mechanism that can't be broken is of course of great interest to security people. Matt and I wondered about both parts of this: How do PALs work, and is that historical statement accurate? I did a lot of digging, including a Freedom of Information Act request, and eventually generated a lengthy Web page and a talk,

which I gave at USENIX Security in 2004. I also honed my historian skills doing a non-computer research project.

Coincidence then took a hand. I have an odd hobby: I collect old telegraph codebooks. (I gave a talk on them at USENIX Security in 2009.) A few years ago, I had a free day in Washington—what should I do? In the morning, I went to the Supreme Court to hear oral arguments in a case—for all of my interest in legal matters, I'd never done that before. In the afternoon, I decided to go to the Library of Congress to look at some of their codebooks. They have hundreds, though; which should I examine? I spotted one from 1882 whose title spoke of “privacy” and “secrecy”—it sounded better than most for a security guy, even though I had low hopes. However, when I read its preface, I realized that it described the one-time pad 35 years before the textbooks say it was invented. I dropped a note to David Kahn asking if he knew anything about it. He didn't and suggested that I write a paper—which I needed no prompting to do; I'm an academic and academics write papers. Before I went to sleep that night, I'd tentatively identified the author. By the time I was done, I had a 20-page paper with 78 references, ranging from the society pages of the *San Francisco Chronicle* from 1907 to a biography of the founder of theosophism to an 1829 history of Freemasonry.

Well, that paper led to another (which has led to two accidental spin-offs), and I have several more planned. None of these will change the way we do things, but it's always good to learn where we've come from. Fun fact: that 1882 codebook shows the use of someone's mother's maiden name to authenticate certain financial transactions. That's one mistake we haven't corrected yet!

Resources

[1] *Firewalls and Internet Security: Repelling the Wily Hacker* (Addison-Wesley Professional, 2003): https://encrypted.google.com/books/about/Firewalls_and_Internet_Security.html?id=_ZqIh0IbcrgC.

[2] S. Bellovin, “A Look Back at ‘Security Problems in the TCP/IP Protocol Suite’”: <https://www.cs.columbia.edu/~smb/papers/ipext.pdf>.

[3] Ryan Singel, “Pakistan's Accidental YouTube Re-Routing Exposes Trust Flaw in Net,” *Wired*, Feb. 25, 2008: <http://www.wired.com/2008/02/pakistans-accid/>.

[4] AS 7007 Incident: http://en.wikipedia.org/wiki/AS_7007_incident.

[5] S. Bellovin, “Using the Domain Name System for System Break-ins”: <https://www.cs.columbia.edu/~smb/papers/dnshack.pdf>.

[6] Statement before the House Select Committee on Homeland Security Subcommittee on Cybersecurity, Science and Research & Development: <https://www.cs.columbia.edu/~smb/papers/Statement.pdf>.

[7] K. Dent, S. Bellovin, “Newspeak: A Secure Approach for Designing Web Applications”: <https://mice.cs.columbia.edu/getTechreport.php?techreportID=506>.

[8] S. Bellovin, M. Blaze, W. Diffie, S. Landau, P. Neumann, and J. Rexford, “Risking Communications Security: Potential Hazards of the Protect America Act”: <https://www.cs.columbia.edu/~smb/papers/j1lanFIN.pdf>.

[9] S. Bellovin, “Life Amidst the Lawyers: A Technologist's Year at the FTC”: https://www.cs.columbia.edu/~smb/talks/Life_FTC.pdf.

CRA-W

Taking Action to Achieve Diversity in Computing Research

DILMA DA SILVA



Dilma Da Silva is a principal engineer and manager at Qualcomm Research in Santa Clara, California, where she leads in the area of mobile cloud computing. She previously worked at IBM Research in New York and at University of Sao Paulo in Brazil. She received her PhD from Georgia Tech in 1997. She has published more than 80 technical papers and filed 14 patents. Dilma is an ACM Distinguished Scientist, an ACM Distinguished Speaker, a member of the board of CRA-W and CDC, co-founder of Latinas in Computing, and treasurer for ACM SIGOPS. More information is available at www.dilmamds.com.

Presented by



I've been a member of the Computing Research Association's Committee on the Status of Women in Computing Research (CRA-W) for six years. Over that time, I've seen the impact that the CRA-W programs have had on the women and minorities who have participated. In this short article, I'll introduce CRA-W's goals, share some evidence of CRA-W's impact, and describe upcoming opportunities for participation.

One aspect that has kept me very active at CRA-W is the action-oriented approach that the organization takes to pursue the mission of increasing the success and number of women participating in computer science and engineering research and education at all levels. The board operates with very few meetings or open-ended discussions; instead, we pursue our goals by having each board member implement a program. Our programs aim to develop research, communication, and career strategy skills, as well as create a sense of community for women in computing research. It is the experience of many of us that such skills help women to earn advanced degrees in computing and achieve success in research careers in academia or industry. The committee is made up of prominent and dedicated women who have progressed well in their careers and are now willing to devote time and energy to design, implement, and secure the necessary funding for initiatives that advance diversity in computing research.

CRA-W programs expose women to knowledge that illustrates the rewards of a research career and the various paths people take to get where they want to be. The programs showcase the value of a PhD and provide guidance on getting to graduate school and making the best out of the experience. Our programs help participants develop self-efficacy through increased skills, knowledge, and confidence. The programs have been designed to connect participants to the research community and to each other to build a network for success. From 1992 to 2013, CRA-W programs have directly impacted more than 7,700 women and minorities.

Many members of the USENIX community have been involved in CRA-W programs, and I believe there are many opportunities to join forces to continue to advance our fields. All USENIX communities can gain a lot by including participation from all groups in society so that a diverse group of people continues to pursue innovation and contribute to advancing our society. But much needs to be done before we get there: Women and minority groups (e.g., Native Americans, African Americans, and Hispanics) are severely underrepresented in all areas of computing. There is increasing demand for talent in all areas of computer science: the Occupational Outlook Handbook (2012-2013) forecasts 22% growth in computer occupations between 2012 and 2022, including 15% growth in computer and information research scientists [1]. In the areas covered by USENIX members, the situation is even more promising, but we still lack the diversity in our professional communities that one would expect based on the general population makeup.

There is strong evidence that diversity can impact business in significant ways:

- ◆ A 2007 NCWIT study shows that IT patents issued to mixed gender teams are cited 26% to 42% more than similar IT patents by all men or all women teams [6].
- ◆ Herring found that companies with reported highest levels of racial diversity had 15 times more sales revenues than those with lower diversity [7].

CRA-W: Taking Action to Achieve Diversity in Computing Research

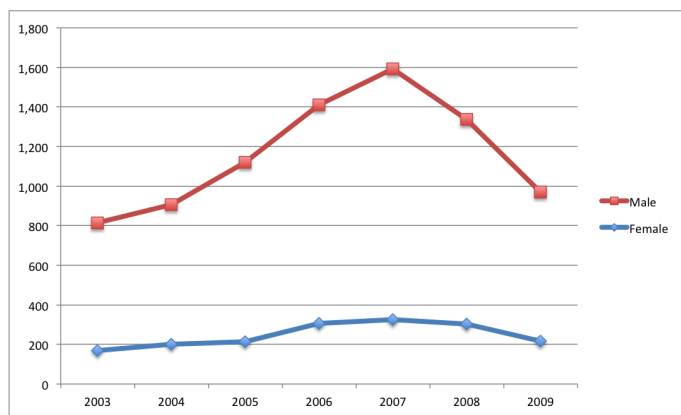


Figure 1: Computer science doctoral degrees granted

- ◆ In the mid-1990s, IBM expanded its minority markets by promoting diversity in its own workforce [8].
- ◆ A study from 2008 finds that having multi-cultural experience enhances creativity [9].

In short, there is evidence showing that a diverse group of contributors can lead to better results. My work in the CRA-W is motivated by much more than any “let’s do the right thing” attitude; like many people I know, I want to help build a future for computing in which the IT industry continues to improve society through significant technological contributions and economic impact.

In research careers requiring graduate degrees, women and minorities continue to be a small fraction of computing PhD recipients (see Figure 1). We are failing to capitalize on the creativity of a large part of our society, and CRA-W works to change this picture so that we achieve a diverse technical community. Studies indicate that a diverse leadership drives students and workforce diversity [2–5], so we have many CRA-W programs that focus on exposing students to a diverse group of role models and mentors from academia and industry. We also have programs to guide junior women researchers towards successful careers.

In CRA-W, we value systematic evaluation of our programs. We founded CERP (CRA Center for Evaluating the Research Pipeline) and perform quantitative comparisons of a nationwide sample of undergraduates, graduate students, and faculty to our program participants. Our evaluation results show that our undergraduate participants are almost four times more likely than nonparticipants to enroll in a PhD program, and our graduate participants are more likely to publish, be first author, and collaborate—all indicators of success in the research community [10].

In upcoming issues of *login*, I will be presenting detailed information on several CRA-W programs and discussing opportunities for CRA-W and USENIX members to collaborate towards a stronger research community. USENIX itself began its Women in Advanced Computing (WiAC) initiative in 2012 and is partnering with CRA-W to bring this content to the USENIX community via *login*, as well as exploring other paths of partnership.

For the list of CRA-W programs and events, please visit www.cra-w.org. An upcoming event of particular interest to *login* readers is USENIX’s Diversity ’14, a discipline-specific mentoring workshop for the system software community co-located with OSDI ’14.

References

- [1] Bureau of Labor Statistics, Occupational Outlook Handbook, 2012-13 Edition, Computer and Information Research Scientists, 2012: <http://www.bls.gov/ooh/computer-and-information-technology/computer-and-information-research-scientists.htm>, accessed May 1st, 2014.
- [2] H. Astin and L. Sax, “Developing Scientific Talent in Undergraduate Women,” in C. S. Davis, A. Ginorio, C. Hollenshead, B. Lazarus, and P. Rayman, eds, *The Equity Equation: Fostering the Advancement of Women in the Sciences, Mathematics, and Engineering* (Jossey-Bass, 1996).
- [3] G. Hackett, D. Esposito, and M. S. O’Halloran, “The Relationship of Role Model Influences to the Career Salience and Educational and Career Plans of College Women,” *Journal of Vocational Behavior*, vol. 35, no.2 (October 1989), pp. 164–180.
- [4] M. A. Mason, M. Goulden, and K. Frasch, “Why Graduate Students Reject the Fast Track,” *Academe*, vol. 95, no. 1 (January-February 2009): https://www.law.berkeley.edu/files/Grad_Students_Fast_Track_Article.mamason.pdf, accessed May 9, 2012.
- [5] B. W. Packard and E. D. Wong, “Future Images and Women’s Career Decisions in Science,” in *Proceedings of the Annual Meeting of the American Educational Research Association*, Montreal, Canada, April 1999.
- [6] C. Ashcraft and A. Breitzman, “Who Invents IT? An Analysis of Women’s Participation in Information Technology Patenting,” technical report, National Center for Women and Information Technology, March 2007.
- [7] C. Herring, “Does Diversity Pay? Race, Gender and the Business Case for Diversity,” *American Sociological Review*, vol. 74, no. 2 (2009), pp. 208–224.
- [8] D. A. Thomas, “Diversity as Strategy,” *Harvard Business Review* (September 2004).
- [9] A. K. Leung, W. Maddux, A. D. Galinsky, and C-Y. Chiu, “Multicultural Experience Enhances Creativity: The When and How,” *American Psychologist*, vol. 63, no. 3 (2008), pp. 169–181.
- [10] E. Bizot, K. Hines, I. Pufahl, T. McKlin, and S. Engelman, “The Data Buddies Project: CRA-W/CDC Project on Measuring Outcomes for Students in Computing: Report on Spring 2011 Surveys,” technical report, Computing Research Association, Washington, DC, 2012: <http://cra.org/cerp/wp-content/uploads/2011/11/Data-Buddies-Report-CRA-Report-Final.pdf>, accessed May 1, 2014.

Hostbased SSH

A Better Alternative

ABE SINGER



Abe Singer is the chief security officer for the Laser Interferometer Gravitational Wave Observatory at the California Institute of Technology. He has been a programmer, system administrator, security geek, occasional consultant, and expert witness. His areas of interest are in security that actually works. abe@ligo.caltech.edu

Almost all SSH users are familiar with two modes of authentication over SSH: passwords and SSH keys. SSH supports another method that seems to be less well known: hostbased, which allows for users to ssh securely between cooperating hosts without providing a credential. It's called hostbased because the client (source) host authenticates itself to the remote host, and the remote host then trusts the client to identify the user. The term "hostbased" is often employed to describe use of hostname or IP-address access control lists. That's not what I'm talking about, so please keep reading.

Sounds scary? Hostbased SSH can be at least as secure as SSH, or more so, and can be simpler to manage.

Hostbased SSH isn't the answer to everything, but I think it's the right answer in particular common scenarios. In the first scenario, you manage a network of computers where each user has the same account, with the same credential, across multiple machines. Once users have authenticated to one of the hosts, there's no added value in requiring them to authenticate again to other hosts on the network. The users may also want to run unattended jobs that execute commands between hosts. Clusters are a particular variant, where users log in to a head node to, in turn, access compute nodes.

In the second common scenario, you want to automate root access to multiple machines in order to control who has remote access to the root account, and from where, and to minimize having to type the root password on the remote host. The latter is especially a good idea if you are investigating a host that might be compromised. As a bonus, you might want to give a user root access to particular hosts without having to divulge the root password.

Hostbased authentication solves these problems because it doesn't require the user to have, know, or enter any credentials.

How does that work? First, a little bit of history.

The R-Commands

Some of you may remember the Good Old Days™ when we had the "r-commands": rsh, rlogin, and rcp. Rlogin worked like ssh: You would "rlogin" to an account at a remote host and, by default, be prompted for the password for that account. If you didn't want to have to type a password every time, you could create a file in your home directory on the remote host called ".rhosts" and, in the file, put a line with the local host and local username.

For example, if `alice@foo.example.com` wanted to log in to `aliceb@bar.example.com`, she'd create the file `~aliceb/.rhosts` on bar, that looked like this:

```
foo.example.com alice
```

Then, when she was on foo and typed "rlogin `aliceb@bar.example.com`," the rlogin server on bar would read the entry in the rhosts file and log Alice into the account `aliceb` on bar, without prompting for a password.

rlogin used a pretty weak security model. Unlike ssh, rlogin provided no encryption, which was a serious problem. The whole protocol was done over cleartext, with no integrity checking, relied on privileged port numbers for validation, and was vulnerable to network address spoofing.

Hostbased SSH

Hostbased SSH implements rlogin style access control without the insecurity of rlogin. The SSH server uses “.shosts” files (or a global “shosts.equiv” file) that use the same format as the old .rhosts file. The SSH client host authenticates itself to the remote SSH server. It then asserts to the server its own hostname and the username originating the session, which the server then checks against the shosts.equiv file or the .shosts file for that user.

So, what prevents a rogue user from asserting any hostname and/or username? This is the awesome part: The client signs the assertion with its host key. In detail, it works like this:

- ◆ Alice, on host “client,” runs “ssh alice@server.”
- ◆ The client makes an ssh connection to the server, negotiates hostbased login and a session ID for alice@server.
- ◆ If hostbased negotiation was successful, the client creates an assertion consisting of the session ID, Alice’s username, and the client’s hostname, signs the assertion with its host key, and sends it to the server.
- ◆ The server checks that the signature on the assertion matches the client’s public key in /etc/ssh/ssh_known_hosts (or ~alice/.ssh/known_hosts if enabled), and that alice@client matches an entry in ~alice/.shosts or /etc/ssh/shosts.equiv.
- ◆ If all the checks pass, the server proceeds with the login [1].

The hostbased dance works because the client’s host SSH private key is (supposed to be) only readable by root, which is why the remote host trusts the key for authentication—the fact that the assertion was signed with the host key is proof that the assertion was signed by root, not the user.

ssh doesn’t normally (and shouldn’t) run as root. Rather, it uses the helper application /usr/lib/openssh/ssh-keysign, which runs setuid root. ssh-keysign gets invoked automatically by ssh, reads the host SSH key, and does the required signing.

This is an elegant little design. The user cannot subvert the contents of the assertion, and only the code that handles the signing has to run as root, minimizing the potentially exploitable setuid codebase.

Of course, you still have to have user accounts on the remote machine. But, because the user doesn’t need to have a password, you have the option of creating accounts with no usable password, so that the only way the user can log in is by using host-

based authentication from hosts that you authorize. Definitely useful in a cluster scenario.

Hostbased SSH uses the same technology as SSH keys, so cryptographically speaking, hostbased SSH authentication is just as strong as SSH key authentication.

Obviously, you can only do hostbased SSH with a client that has a host key. Normally, that would be a host running an SSH server. Hostbased wasn’t really designed to be run from a client-only setup such as a laptop, although you could in theory just generate a host key manually without running the ssh daemon. I’ll leave how to do that as an exercise for the reader.

Howto

Here’s how to make hostbased SSH work on the client and on the server.

On the Client

You need to have the following entries in /etc/ssh/ssh_config:

```
EnableSSHKeysign yes
HostbasedAuthentication yes
```

/usr/lib/ssh-keysign has to be setuid root (which it is by default) so that it can read the host key.

On the Server

You need the following in /etc/ssh/sshd_config:

```
HostbasedAuthentication yes
IgnoreRhosts no
```

Put the client host’s SSH keys in /etc/ssh/ssh_known_hosts. The ssh_known_hosts file functions for hostbased SSH similar to how the user’s authorized_keys file functions for SSH key authentication: The server will only accept hostbased authentication from clients whose host public keys are in the file. You can get public keys from clients using ssh-keyscan:

```
ssh-keyscan -t dsa client.example.com
```

Verify that the results are indeed the public key of the client by using key fingerprints:

```
ssh-keygen -l -f <public key file>
```

Remember to tell the ssh daemon to reload the configuration:

```
service ssh reload
```

Of course, you can create a script or use a configuration management tool to push the configuration and ssh_known_hosts file to several machines at once.

Note that adding keys to ssh_known_hosts does not require restarting sshd.

Hostbased SSH: A Better Alternative

Now you just have to configure which users get access.

Controlling User Access

Hostbased SSH provides two flavors of controlling which users can log into accounts: `~/.shosts` or `/etc/ssh/shosts.equiv`.

`shosts.equiv` allows users on clients to log in to accounts on the server with identical usernames. In other words, `alice@client` can log in to `alice@server`, but not to `aliceb@server`. You can enable access on a per-user basis or allow all users on the client to log in to the corresponding account on the server. The file should be editable only by root, so regular users cannot make any changes to it.

The syntax for entries in the `shosts.equiv` file is:

```
<remote host> [-][<remote username>]
```

`sshd` reads the file from the beginning and stops at the first matching line. A hostname without a username allows all users (except root) from the client to log in to the matching username account on the server. A hostname and a username allows a specific user to log in. A “-” in front of a username excludes that user—useful only if followed by a line with just a hostname—to allow all other users.

Note that you cannot enable root access with `shosts.equiv`. The hostname-only format excludes root, and the server will ignore explicit entries for root.

If you want to allow root login, or let users log in to accounts that have different usernames, you have to use the `~/.shosts` file. The syntax of the file is identical to `shosts.equiv`, and multiple entries are allowed. Specifying a hostname works identically to `shosts.equiv`, but specifying a username allows a non-matching username login to the account.

Thus, if on the server, `~alice/.shosts` contains:

```
ws1.example.com alice
ws1.example.com bob
ws2.example.com alice
```

then `alice@ws1`, `alice@ws2`, and `bob@ws1` can ssh to `alice@server`.

`~/.shosts` gives you more flexibility, but `shosts.equiv` file gives you more control over who gets authorized, at the expense of your having to maintain it.

Yeah, You Could Use SSH Keys

Everything I’ve talked about could be implemented using SSH keys, but with worse failure modes. For starters, the default access mode for SSH keys is to allow access from anywhere; any restrictions applied are vulnerable to spoofing, and there is no way to say “only `bob@foo` can log in to `carol@bar`.”

An unfortunately common solution to passwordless login is the use of SSH keys with no passphrase on the secret key (“passwordless keys”), which is effectively storing an unencrypted password in a text file (which users also often do), a basic security no-no.

And many scenarios require putting the user’s SSH key on every host, which defeats the design of SSH keys, where the key only lives on the client.

`ssh-agent` makes things a little better, but it has to be manually restarted when a host reboots—a pain when you had it running on a thousand nodes that just rebooted due to a kernel patch.

Auditing access and de-authorization are more difficult with SSH keys.

Key management is always difficult. SSH key solutions require more key management than hostbased. In fact, SSH key management is difficult enough that `ssh.com` even sells a product focusing on just that.

Hostbased SSH is just simpler.

What, Me Worry?

Usually at some point when I’m explaining hostbased SSH, someone says “But what if the client machine is rooted? Anyone with root could log in to any user’s remote account!” Yes, that is true. But then any scheme—be it hostbased, password, or SSH key—fails if the client is rooted. You have to trust root on the remote host in all cases. Hostbased SSH is just a bit easier to manage.

Let’s go over the risks of using hostbased, and compare to SSH keys.

Hostbased SSH uses the same technology as SSH keys; so, cryptographically speaking, hostbased SSH authentication is just as strong as SSH key authentication.

With SSH keys and hostbased, you have to trust root (users) on the remote host and trust that the root hasn’t been compromised. With hostbased, you don’t have to trust that user keys are being managed properly and whether they have strong passphrases.

Hostbased does true authentication of the client, whereas SSH keys can only validate the client’s hostname or IP address, which can be spoofed.

In short, hostbased SSH has fewer trust requirements than SSH keys and is harder for the users to circumvent.

The user’s `.shosts` file can, of course, be modified by the user to allow access to other users. That can be good or bad, depending on your policies. However, with hostbased the user can only do so for remote hosts that you have explicitly authorized to do host-based login by adding their host keys to `/etc/ssh/ssh_known_hosts`. With SSH keys, the user could allow someone access from any remote host.

If a user is going to allow other users access to their account, I'd like to have a log showing which other user logged in. And I really prefer an access mode that doesn't require users to share passwords.

Additional Tricks

Here are some other options that you can use with hostbased auth.

You can get the hosts to log which user logged in using host-based access by setting the log level to "VERBOSE" in `/etc/ssh/sshd_config`:

```
LogLevel VERBOSE
```

You get a log message that looks like this:

```
sshd[8180]: Accepted DSA public key 7c:3d:bc:84:c5:87:71:06:93:
56:ff:d6:8c:c4:ae:66 from alice@client.example.com
```

You can let users configure hostbased SSH access to just their account from some external host by putting that remote host's public key in their `~/.ssh/known_hosts` file, if you include the following directive in `sshd_config`:

```
IgnoreUserKnownHosts no
```

You can let clients assert their hostname with just their host key. This can be used to allow hostbased SSH from a roaming laptop whose IP address changes:

```
HostbasedUsesNameFromPacketOnly yes
```

You can force `sshd` to use `shosts.equiv` only:

```
IgnoreRhosts yes
```

You can force the server to only accept hostbased:

```
AuthenticationMethods Hostbased
```

or just do hostbased first:

```
PreferredAuthentications HostBased,PublicKey,Password
```

Restricting what commands a user can run is a little complicated. You can do it by creating a script in `/etc/ssh/sshrd`. The details are left as an exercise for the reader.

Hostbased SSH can be a bit painful to debug when it doesn't work right. In the interests of space, I'll refer you to [2] and [3] for some debugging help.

Secure Remote Root Access

I want to be able to remotely log in to my hosts as root for a variety of reasons, but I want to be able to do it in a secure manner.

The Problem with Remote Root Login

It used to be a "best practice" that one did not remotely log in to a host as root; rather, one would log in as a regular user and then `su` to root. In short, the problems are: no accounting of who has had root at a given time; an attacker with the root password gets immediate access; and trojaned SSH clients get login passwords much more easily than a password entered in a shell.

Back in the Telnet days, most distros had root login via Telnet disabled by default; you had to explicitly enable it. Telnet was also unencrypted, an additional problem.

Unfortunately, OpenSSH comes with root login turned on by default. As a result, we have a new generation of sysadmins who blithely `ssh` as root from anywhere, with password or SSH key or whatever. It still makes me cringe when I see someone do it.

Particularly when doing incident response, the last thing you want to do is type a password on a host that might be compromised. Attackers usually root systems without having the root password (they wouldn't need an exploit if they had it). If the root password on the compromised host is the same as on other hosts, and they allow root login, the attacker might just get root access to all your hosts, without even needing another exploit.

My Remote Root Solution

Clearly, there are a number of situations where you need remote root access. I want to run the same commands across hundreds of hosts automatically. I want to be able to run remote commands non-interactively, and I don't want to type a password on a potentially compromised host. I want to give other users selective root access to hosts without giving them the root password, and easily disable their access. And I want a log of who had access.

Hostbased SSH makes this easy.

Here's my solution:

I have a dedicated bastion host whose sole purpose is to provide root access to other hosts. The root account on the bastion host is authorized for hostbased access to my other hosts (the "target hosts") and is used to manage hostbased access on the target hosts.

Each user of the bastion host has her own account and has to use a type of two-factor authentication to get to that account.

Hostbased configuration on the bastion host is done exactly as described above. But on the target hosts, I add some more restrictions. I want to allow root `ssh` only from the management host. I do this using the "Match" statement in `/etc/ssh/sshd_config`:

Hostbased SSH: A Better Alternative

```
IgnoreRhosts no
PermitRootLogin no
Match Host bastion.example.com
    HostbasedAuthentication yes
    PasswordAuthentication no
    PermitRootLogin without-password
    RhostsRSAAuthentication no
    PubkeyAuthentication no
    GSSAPIAuthentication no
```

The “IgnoreRhosts” line is required so that I can use `~root/.shosts` for controlling who gets access. `sshd` doesn’t allow that directive to be inside a match statement, so it has to be global. However, because the `.shosts` file only works for hosts authorized for hostbased access, the net result is the same.

Inside the match statement, I’ve disabled all modes of authentication except for hostbased. There’s no reason to allow them. Similarly, I’ve enabled root login using the “without-password” option, which means that root login over SSH cannot be done with a password in any circumstances (a bit belt-and-suspenders since password authentication is separately disabled, but better to be cautious).

Then in `~root/.shosts`, I put in an entry for root on the bastion host, and then for each user who gets access:

```
bastion.example.com root
bastion.example.com alice
bastion.example.com bob
bastion.example.com carol
```

Remember, Alice, Bob, and Carol don’t get the root password to the bastion host nor to the target hosts; they just have credentials for their account. I can give them each access to only the hosts that they need access to, and I can quickly disable their root access to all hosts by disabling their account on the bastion server.

And Bob’s your uncle.

Resources

- [1] T. Ylonen, RFC 4252, The Secure Shell Protocol, Section 9 “Host-Based Authentication,” January 2006.
- [2] Wikibooks OpenSSH/Cookbook/Host-based Authentication: http://en.wikibooks.org/wiki/OpenSSH/Cookbook/Host-based_Authentication.
- [3] Daniel J. Barrett and Richard E. Silverman, *SSH: The Secure Shell: The Definitive Guide* (O’Reilly and Associates, February 2001).

Challenges in Event Management

JASON PAREE



Jason works for CACI Inc. as the release and deployment process owner for US CENTCOM J6.

His position requires managing change as well as responding

to and controlling outages and issues on live, operational networks. He received his BS in criminal justice administration in 2011 but switched career goals and is currently an MBA student. jasonparee@gmail.com.

Inherent to providing and managing IT services is having to deal with occasional outages, issues, and security concerns. These types of events, while inevitable even on the most reliable networks, can wreak havoc on a service provider's reputation if not handled and communicated properly. Customers of IT typically do not understand the guts of providing service. They know they turn on their computer, open Word, surf the Internet, or communicate using email and chat. In general, they do not understand nor do they usually care what makes it work. This changes, however, when those services suddenly stop working. At this point, a process by which to manage these occasional events and communicate about them effectively with the customer comes in handy.

I work for an IT contracting company that provides services internally to a specific command within the Department of Defense. Our environment is a complex hybrid of several autonomous, secure networks supported by a litany of technologies, from Cisco Nexus 7K routers to virtualized desktops spread out over several domestic and international locations servicing more than 5000 active users. In addition, the customer's requirements usually come fast and with little time for planning. It is a difficult environment in which to manage changes or outages while remaining flexible to the customer's needs.

During the past two years, we worked very hard to develop, enforce, and maintain a streamlined change deployment process, which has paid huge dividends in providing a reliable and stable IT environment [1]. With that process firmly in place, it was time for us to turn our attention to the management of major outages, issues, or events. This need to manage events as they occurred was born out of frustration felt by both our company's and our customer's leadership. Too often, our leadership didn't get the information they needed when they needed it. In addition, our troubleshooting was often poorly communicated and coordinated, resulting in significant inefficiencies. Technicians would work on things other groups of technicians were working on, wasting time and effort and lengthening the downtime to the customer and users. To make things worse, communications were not being centrally managed, which led to confused messaging and inaccurate data. Regular updates to the customer were not reliable or required from any one person, and so the customer would receive conflicting information. Needless to say, all of these problems created the perception (sometimes rightly) that our technicians were not working cohesively, reliably, or efficiently. In short, it made us look unprofessional and uncoordinated.

A few months ago, my operations manager approached me about setting up a new process to rein in our efforts when responding to "events." We decided on the name "Event Management" for the process by which we would make this happen. Once we began planning, we quickly realized how much more difficult this would be to implement than we had previously thought. It would require:

Challenges in Event Management

- ◆ Participation by the customer and all the technology leads
- ◆ A new tool to track the details of the events
- ◆ New personnel to provide support 24x7
- ◆ A documented process detailing the procedures to follow

In addition to these challenges, we needed to make cultural changes to prevent the customers from going directly to the technology leads for answers (as was previously the case) and convince them of the value and need for the Event Management team.

The Beginning

My first task was to market the idea to the customer and ensure their buy-in; otherwise the whole idea was pointless. I spoke with their watch officer, the position closest to the concept of event management and the person who would usually go out and collect information about outages and issues. The watch officer is a customer-owned position, which creates unique challenges for dividing roles and responsibilities between us (the contractors) and the customer. However, after explaining how this approach would provide a one-stop-shop for them, allowing them to simply rely on our team for information rather than run around asking people, they immediately bought into the idea. And why not? We just made their job much easier by providing a single, central point of contact for all ongoing events.

After we got cooperation from the customer, we set out to build a small team of people to provide coverage. Because this process wasn't built in to the original contract, the customer had no obligation to support us in funding the extra personnel. As a result, I had to work with the operations manager to find extra positions within the division that could be repurposed for this new role. This was a difficult task because, as you would expect, these contracted positions do not hang from trees, and money is tightly budgeted. In any case, we were able to find open positions that we could shuffle around to make a team. Unfortunately, we could only muster three positions, which limited our ability to provide complete coverage. Instead of the originally intended 24x7 coverage, we settled for 24x5. Once we had a team, we turned our attention to scratching out some rough procedures on how the process would work and how responsibilities would be divided.

Writing out the procedures involved a lot of consideration since there would have to be requirements built from several areas, including the service desk, the technology leads, the operations manager, and the event manager. At the request of our operations manager, we held an off-site meeting with several key technology leads to rough out some basic procedures. This was enough to get us off the launch pad. The procedures included requirements for how and when to notify Event Management of an issue, when updates were expected, and how Event Management would communicate and escalate issues. The next day, I emailed all the key leads and gave them a simple flow of how the process would work, and I continued to draft the official procedures.

Once we had a small team with decent coverage and some semblance of procedures, I worked to develop tools by which to track and communicate outages and issues as they occurred. I ended up going simple and using a glorified Excel spreadsheet posted on the main page of our SharePoint site. This made tracking easy, simple, and available to everyone. In addition, we developed specifically formatted emails to communicate the details of outages. These emails have specific information requirements, formatting requirements, and a defined distribution list. The intent is to provide a consistent and reliable product that all key stakeholders can use and understand, including our government customer, the company leadership, and relevant technology leads. At this point, we had addressed many of the practical and tangible problems. The hard part was and continues to be the cultural shift in implementing the process.

Cultural Norms

The most glaring problem of all, and one that will most likely continue for the long term, is the change in cultural norms. The technology leads are accustomed to a certain way of doing things and to not having to explain themselves to external groups. The watch officer is used to reaching out to technology areas directly rather than going through an intermediary. The operations manager is used to hearing directly from individual leads and reaching into their teams for answers and to give directions. All of this has to change and be retuned to utilize the Event Management team to coordinate efforts, understand the problem, and provide accurate outage reporting and a general sense of organization and leadership during an outage. The Event Management team must assert itself as the "belly button" of information. In addition, we must prove ourselves capable of managing the efforts of several teams, communicating status and maintaining awareness of the problem at hand. The operations manager has to support this team and avoid reaching into the teams directly. This kind of management support is critical to the team's ability to succeed and reach its objective. The technology leads will have to get used to providing more detail about outages and allowing the Event Management team to have more visibility in their areas. This is another cultural change that will require the support of and enforcement by the operations manager.

Currently, we have a formal process document outlining all of the requirements and buy-in from all the key stakeholders. Thus far, we have successfully built positive relationships with many of the technology areas and with the customer watch officer. This has allowed us to foster a viable environment by which the process can take root. As previously noted, 24x7 coverage has been a little trickier than originally assumed. We have settled on 24-hour operations Monday through Friday with a stand-by schedule for weekend coverage. Generally, we have made a lot of progress toward developing the process and creating an environment in which the process can take hold.

Conclusion

If we can make this process work, it has the potential to pay huge dividends over time. The company will have centralized, dedicated management and communications during an outage, issue, or security event. In addition, the customer will have only one place to look for answers instead of getting several different answers from several different sources. This approach has the potential to provide political benefits as well. The customer will have a positive perception of the company's ability to provide singular communications and reliable reporting during an outage. In addition, the presence of the team provides a certain level of customer confidence in the company's contract team, which has an intangible value.

This type of management function and process is in fact very valuable for any IT service-providing entity. Providing a single point of contact to collect information, facilitate and coordinate efforts, and provide a conduit for management oversight of troubleshooting efforts allows for more efficient operations and better customer service. Organizations can gain much by utilizing a central presence for which all key stakeholders can find reliable, consistent, and authoritative information.

Resources

[1] Jason Patee and Andy Seely, "The Evolution of Managed Change in a Complex IT Enterprise," *login.*, vol. 39, no. 1, February 2014: <https://www.usenix.org/publications/login/feb14/evolution-of-managed-change>.



Do you have a USENIX Representative on your university or college campus? If not, USENIX is interested in having one!

The USENIX Campus Rep Program is a network of representatives at campuses around the world who provide Association information to students, and encourage student involvement in USENIX. This is a volunteer program, for which USENIX is always looking for academics to participate. The program is designed for faculty who directly interact with students. We fund one representative from a campus at a time. In return for service as a campus representative, we offer a complimentary membership and other benefits.

A campus rep's responsibilities include:

- Maintaining a library (online and in print) of USENIX publications at your university for student use
- Providing students who wish to join USENIX with information and applications
- Distributing calls for papers and upcoming event brochures, and re-distributing informational emails from USENIX
- Helping students to submit research papers to relevant USENIX conferences
- Encouraging students to apply for travel grants to conferences
- Providing USENIX with feedback and suggestions on how the organization can better serve students

In return for being our "eyes and ears" on campus, the Campus Representative receives access to the members-only areas of the USENIX Web site, free conference registration once a year (after one full year of service as a Campus Representative), and electronic conference proceedings for downloading onto your campus server so that all students, staff, and faculty have access.

To qualify as a campus representative, you must:

- Be full-time faculty or staff at a four year accredited university
- Have been a dues-paying member of USENIX for at least one full year in the past

For more information about our Student Programs, contact Julie Miller, Marketing Communications Manager, julie@usenix.org

www.usenix.org/students

/var/log/manager

When Technology Isn't the Cause of a Technical Problem

ANDY SEELY



Andy Seely is the manager of an IT engineering division, customer-site chief engineer, and a computer science instructor for the University of Maryland University College. His wife Heather is his init process and his sons Marek and Ivo are always on the run queue. andy@yankeetown.com

If the only tool you have is a hammer, every problem looks like a nail. In the technology space, we tend to approach all problems as technology problems. That's how we're wired. We're systems people. If something's not performing correctly, maybe we can adjust the system settings or the resource provisioning. Maybe we can buy a new software tool to compensate. It's a different kind of management challenge to see a technical problem's organizational roots and to make an adjustment, far removed from the actual technology, that can relieve tension in the organization and result in better system performance.

It's Just a Slow "Picard vs. Saruman" Sort of Day

Everything's fine. The enterprise is running within parameters. The technical team seems happy, at least in that they're not dealing with things any more important than Pickard vs. Kirk (see Sidebar). The management team's biggest problem is worrying about which Web sites the employees are surfing to when they're on the clock. Customer calls to the service desk are normal noise: password resets and unreasonable demands for magical computers that don't exist. It's a good day.

We use an internal service to process requests for a business intelligence (BI) product. It's a pretty sizable data warehouse with a number-cruncher front end. There's a small team of operators and a couple of sysadmins, all of whom keep mostly to themselves. It runs; no one worries. Once the Picard vs. Kirk got some Saruman and Gandalf thrown into the fight (without Picard and Kirk exiting, which is in itself interesting), I wandered down the hall to ask how those keep-to-themselves BI folks were doing.

Everything's fine. Customer queries were being answered. They had no problems. Except that performance wasn't really what they liked. OK, performance tuning is something we do, so I asked them to describe the performance problem. Well, they say, the front end has been broken for months, and the sysadmin can't keep up with the operator requests that he's been answering directly from the database using the command line SQL interface.

What?

Finding Common Ground in a "Picard vs. Saruman" Sort of Situation

I'm not a fan of meetings for the sake of meetings, but if ever there was a need to get everyone around the same table, this was it. We called a meeting of the BI team, the engineering team, the storage team, the operations team, the database team, the monitoring team, the service desk, and the management team.

From the start, we didn't have consensus. Each area of our overall team felt that they either already knew the whole story or didn't have any responsibility for this system at all. I like to joke that I'm "classically trained" in the art of holding meetings, but this was a tough one to navigate. I'm the head of engineering and have a finger on the pulse of almost all we do, but this Business Intelligence system pre-dates me, and almost no engineering project or sup-

`/var/log/manager`: When Technology Isn't the Cause of a Technical Problem

port work had come up during my tenure. The meeting was as much about discovery for me as anything.

And what did I discover? Operations had no sense of ownership because the BI team had their own internal sysadmins. The BI sysadmins felt like they had been abandoned, because they had submitted dozens of service requests to operations over the years, and they couldn't understand why no one realized they had a systemic problem. The engineers acknowledged the presence of the BI system but only so much as they occasionally got asked for very specific help doing very specific tasks. The storage team responded to storage requests like they do to everyone, with an initial "we don't have any more storage capacity" followed by a grant of storage after they discovered a way to free more space. If we'd all had pistols, we'd have looked like a Spaghetti Western, with all of us pointing our guns at each other.

Leadership with a Lowercase "l"

I'm not the manager of the BI system or of the operations team, but I am a senior manager in the technical staff. If I see something broken, it's my responsibility to ensure it gets fixed. There are many schools of thought on leadership. I chose to employ my own "big-L/little-l method." This meeting cried out for some "little-l," or lowercase leadership: It didn't need some big boss to make big decisions, just someone to get his hands dirty and help clear the path so that everyone could have a say and get all the facts on the table.

I guided the discussion and turned it over and around until everyone at the table had the same basic understanding of the BI system architecture and dependencies. Then we drew it on a whiteboard and walked through it again, refining the diagram until it reflected both the system and our common understanding.

After we all agreed on architecture, we walked through data flow. Request comes in, gets received here, gets processed here, traverses this subsystem and that subsystem; we followed the flow from query to answer. We talked about system failures and how they're reported and recovered. We talked about resource provisioning and network link speeds. We asked the functional expert to talk about the BI system's internal limitations for complex queries and how the vendor's tuning recommendations were being applied.

Leadership with an Uppercase "L"

We discovered some non-obvious but fundamental flaws in the system, but not the system one would think. Our technical flaws were coming from the organization itself.

1. The BI team gave the appearance of running their own show. The operations team didn't track metrics or report BI outages on their balance sheet, which meant that operations management never put pressure on the BI system to be tuned or improved from a systems perspective.
2. By having its own sysadmins, the BI team built an unintentional wall between themselves and the rest of the sysadmin team. The operations sysadmins never added up all the little requests for support to make a bigger-picture approach because they figured the BI sysadmins knew what they were doing. By reporting issues through business rather than technical management chains, the BI sysadmins' complaints up their management chain fell on deaf ears.
3. By not having storage engineers involved with a holistic perspective, requests were fulfilled as requested rather than as needed, and they weren't requested in such a way as to put database indexes on the fast storage.

Fixing this required "big-L," or uppercase Leadership: The boss needs to make changes in how we do business.

True Story: The DNS Subdomain Generation and Genre Problem

We were troubleshooting a DNS problem with a delegated subdomain. When we started looking into the architecture of the subordinate organization, we found that they had four redundant DNS hosts with host names "picard," "kirk," "gandalf," and "saruman." I was leading the technical team researching the problem. We found the root cause was bad glue records, but in my final analysis I pointed out that there were at least two major system incompatibilities in the subdomain. First, there's a generational gap; Picard and Kirk are not going to cheerfully serve up the same answer as peers. Kirk will overpower Picard whenever he can, serving DNS answers that are the best for Kirk's own position. Second, there's a genre gap; you can't have Kirk and Gandalf working in the same DNS namespace. You'll get DNS query responses in Elvish one time and in Klingon the next, obviously resulting in protocol errors. Our recommendation was to rebuild the whole DNS environment and rename with more of a modern meme. The DNS servers should be: "neo," "morpheus," "trinity," and "tank." This way, they're all on the same team, serving the same mission. Performance will be improved through the virtualization of three DNS hosts, but it's a good idea to keep one DNS server physical to remove the common dependency on the virtual environment.

Starship Captains to the Bridge, Wizards to the Tower: Small Personnel Adjustments Can Make a Big Difference

We drastically improved the BI system with some small organizational changes. The BI sysadmins were reassigned to the application support group in operations. The enterprise monitoring team was given the green light to dig deeper and monitor more aspects of the system and to treat problems with more vigor than just sending an email to the BI team. The storage engineers were given greater purview over the “why” as well as the “what” when it came to decisions on storage provisioning for the BI system.

These organizational changes allowed real system improvements to flow:

1. Network links were upgraded and made standard between all BI systems, removing inter-system bottlenecks.
2. Database indexes were moved off SATA and onto solid-state drives, removing the BI query bottleneck.

3. Benchmarks were established for BI queries, creating a measuring stick of how to interpret BI system performance.
4. New monitoring hooks were established and alert playbooks created, improving overall awareness and problem response times.

These system improvements allowed the real benefit to happen: The BI query backlog was eliminated, and the BI functional operators were able to do their own jobs effectively. Getting there wasn't obvious, and it took a combination of the little-l leadership of guiding people to talk to other people and the big-L leadership of making immediate organizational changes in multiple areas to get work flowing and the system back to its core function of making money for the company. A reorganization of the team wasn't the most obvious approach, but ultimately it was the correct one. I'm the manager. That's my job.

USENIX Member Benefits

Members of the USENIX Association receive the following benefits:

Free subscription to *login*, the Association's magazine, published six times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on such topics as security, Perl, networks, and operating systems, book reviews, and reports of sessions at USENIX conferences.

Access to *login*: online from December 1997 to this month: www.usenix.org/publications/login/

Access to videos from USENIX events in the first six months after the event: www.usenix.org/publications/multimedia/

Discounts on registration fees for all USENIX conferences.

Special discounts on a variety of products, books, software, and periodicals: www.usenix.org/membership/specialdisc.html.

The right to vote on matters affecting the Association, its bylaws, and election of its directors and officers.

For more information regarding membership or benefits, please see www.usenix.org/membership/ or contact office@usenix.org. Phone: 510-528-8649

USENIX Board of Directors

Communicate directly with the USENIX Board of Directors by writing to board@usenix.org.

PRESIDENT

Brian Noble, *University of Michigan*
noble@usenix.org

VICE PRESIDENT

John Arrasjid, *VMware*
johna@usenix.org

SECRETARY

Carolyn Rowland, *National Institute of Standards and Technology (NIST)*
carolyn@usenix.org

TREASURER

Kurt Opsahl, *Electronic Frontier Foundation*
kurt@usenix.org

DIRECTORS

David Blank-Edelman, *Northeastern University*
dnb@usenix.org

Cat Allman, *Google*
cat@usenix.org

Daniel V. Klein, *Google*
dan.klein@usenix.org

Hakim Weatherspoon, *Cornell University*
hakim@usenix.org

EXECUTIVE DIRECTOR

Casey Henderson
casey@usenix.org

SAVE THE DATE!



11th USENIX Symposium
on Operating Systems Design
and Implementation

October 6–8, 2014
Broomfield, CO

Join us in Broomfield, CO, October 6–8, 2014, for the **11th USENIX Symposium on Operating Systems Design and Implementation (OSDI '14)**. The Symposium brings together professionals from academic and industrial backgrounds in what has become a premier forum for discussing the design, implementation, and implications of systems software.

Don't miss the co-located workshops on Sunday, October 5

Diversity '14: 2014 Workshop on Supporting Diversity in Systems Research

HotDep '14: 10th Workshop on Hot Topics in Dependable Systems

HotPower '14: 6th Workshop on Power-Aware Computing and Systems

INFLOW '14: 2nd Workshop on Interactions of NVM/Flash with Operating Systems and Workloads

TRIOS '14: 2014 Conference on Timely Results in Operating Systems

All events will take place at the Omni Interlocken Resort



www.usenix.org/osdi14

Practical Perl Tools

Zero Plus One

DAVID N. BLANK-EDELMAN



David N. Blank-Edelman is the director of technology at the Northeastern University College of Computer and Information

Science and the author of the O'Reilly book *Automating System Administration with Perl* (the second edition of the Otter book), available at purveyors of fine dead trees everywhere.

He has spent the past 24+ years as a system/network administrator in large multi-platform environments, including Brandeis University, Cambridge Technology Group, and the MIT Media Laboratory. He was the program chair of the LISA '05 conference and one of the LISA '06 Invited Talks co-chairs. David is honored to have been the recipient of the 2009 SAGE Outstanding Achievement Award and to serve on the USENIX Board of Directors beginning in June of 2010. dnb@ccs.neu.edu

In our last time together, we spent the column exploring ZeroMQ (sometimes written OMQ). We looked at the basics of what it is, why it is cooler than I will ever be, and began to look at some sample Perl code that uses it. With this column, I hope to take the subject just a little further by exploring a few slightly more complex OMQ topics. I'm going to (mostly) avoid rehashing the basics from last time, so I recommend you check out that column first.

Begin Again, Again

Early in the last column, I mentioned that there were two (well, two and a half) main strains of Perl modules that would let you work with ZeroMQ from Perl. The first was the ZMQ::LibZMQx series (ZMQ::LibZMQ2 and ZMQ::LibZMQ3 for versions 2 and 3 of the ZeroMQ libraries). There was also a thin wrapper around these (hence the "half" comment) called ZMQ that would call one of the two. The second main strain I mentioned was ZMQ::FFI, which used the libffi library as a bridge to the ZeroMQ libraries. In the last column, I made the decision to show code using the first kind of module (ZMQ::LibZMQ3). I still think this is a fine and dandy sort of thing to do; but, since that column was written, it has become more apparent to me that the ZMQ::FFI may be the future of ZeroMQ for Perl. For example, although there is not yet a ZMQ::LibZMQ4 to use version 4 of ZeroMQ (and I'm not sure there will be), ZMQ::FFI works with it out of the box. This may be important to you if, for example, you were interested in using some of the brand-new encrypted transport hotness that just arrived in version 4 of ZeroMQ. The version 3 libraries work perfectly fine, so if you are already happily using ZMQ::LibZMQ3 there's no real need to rush out right this minute and rewrite all of your code.

If indeed ZMQ::FFI will ascend as the preferred module, I think it would be a service to you, dear reader, that I demonstrate how to code using it. So, in this column, we'll be switching horses midstream and will start using it. To help ease the transition a little bit, I'll take a look at the example client-server (or REQ-REP) code that I ended the previous column with—this time coded using ZMQ::FFI. Here's the server version:

```
use ZMQFFI;
use ZMQFFIConstants qw(ZMQ_REP);

my $ctxt = ZMQ::FFI->new();
my $socket = $ctxt->socket(ZMQ_REP);

my $rv = $socket->bind('tcp://127.0.0.1:8888');

while (1) {
    my $msg = $socket->recv();
    print "Server received: $msg\n";

    $socket->send($msg);
}
```


The differences between this version and the `ZMQ::LibZMQ3` are pretty small but worth pointing out. First, this version is more object-oriented. Everything except the messages being received and sent are objects; whereas before we would just “`zmq_recvmsg($socket)`,” here you can see that our `recv()` is a method of the socket object. The second significant difference is that we no longer have to do any special processing to prepare or receive messages. This simplifies the code a wee bit.

The client code has almost identical changes from the previous version:

```
use ZMQ::FFI;
use ZMQ::FFI::Constants qw(ZMQ_REQ);

my $ctx = ZMQ::FFI->new();
my $socket = $ctx->socket(ZMQ_REQ);

$socket->connect('tcp://127.0.0.1:8888');

my $counter = 1;

print "I am $$\n";

while (1) {
    $socket->send("$counter:$$");
    print "Client sent message " . $counter++ . "\n";

    my $msg = $socket->recv();
    print "Client received ack:$msg\n";
    sleep 1;
}
```

I know I mentioned this last time, but as I look over this code now I feel compelled once again to offer my appreciation to the OMQ folks and the author of this module for providing a framework that lets us write code that looks this simple but does so much great stuff behind the scenes. Let’s take this code a little further.

Ah, to Be Young and Asynchronous

I’m not sure whether you recall the output of the previous examples, but a key quality of the code it demonstrated was the synchronous nature of the REQ-REP (request-reply) socket types. They are constructed in such a way as to mandate a strict “you request - I reply, you request - I reply” pattern. It doesn’t work to fire off a bunch of requests without `recv()`-ing the replies back for each one. If we want to do that sort of thing, we need to explore some more complicated socket types. There are two sets of socket types that can help with this. The first is an extension of the REQ-REP types we’ve been using. The second introduces a different paradigm, so we’ll hold off on that until the end of this column when you’ll see a second way to program with ZeroMQ.

The two new socket types I’d like to introduce you to now are DEALER and ROUTER (`ZMQ_DEALER` and `ZMQ_ROUTER`). I find it easiest to remember them by matching them up to the

previous types we’ve been using. A DEALER is like a REQ socket in that it gets used to connect in a “client-like” way to other socket types (and, indeed, those types could be the REP type we’ve seen before). If you can imagine a card dealer sitting in the middle of a table, dealing out messages to eager card players who receive them, that’s the basic idea. I call this “client-like” because the basic direction of packets is out from the DEALER to a receiving (server-esque) socket. The flip side to a DEALER socket is a ROUTER socket. ROUTER sockets are expected to receive messages from a number of sources (which could be REQ sockets or DEALERS). If you think of a network router that sits on a network waiting to receive packets that it passes along, you’ve got the right idea again.

I’ll start off checking out what happens if I combine a REQ and a ROUTER socket. I’ll use almost exactly the same REQ code as before with a few small twists:

```
use ZMQ::FFI;
use ZMQ::FFI::Constants qw(ZMQ_REQ);

my $ctx = ZMQ::FFI->new();
my $socket = $ctx->socket(ZMQ_REQ);

$socket->connect('tcp://127.0.0.1:8888');

my $counter = 1;

print "I am $$\n";

while (1) {
    $socket->send("$counter:$$");
    print "Client sent message " . $counter++ . "\n";

    my @msgs = $socket->recv();
    print "Client received ack:$msgs[0]\n";
    sleep 1;
}
```

The only change I want to bring to your attention is the retrieval of an array of values from `recv()`. I can best explain the `recv()` change in the context of the second piece of code—the one with a ROUTER socket:

```
use ZMQ::FFI;
use ZMQ::FFI::Constants qw(ZMQ_ROUTER);

my $ctx = ZMQ::FFI->new();
my $socket = $ctx->socket(ZMQ_ROUTER);

my $rv = $socket->bind('tcp://127.0.0.1:8888');

while (1) {

    my ( $id, $spacer, $msg ) = $socket->recv_multipart();
    print "Server received: $msg\n";

    $socket->send_multipart( [ $id, '', $msg ] );
}
```

Practical Perl Tools: Zero Plus One

Now we see an interesting change from the REP code we looked at before. In this code, I've replaced `recv()` with `recv_multipart` and `send()` with `send_multipart()`. The distinction is the `_multipart` forms know how to handle the sending and receipt of multiple message frames at the same time.

Let me step back a moment to explain why this is necessary. Behind the scenes, messages actually consist of multiple frames. With the REQ->REP code, we didn't have to care because there was a one-to-one relationship between a message and its reply that ZeroMQ handled automatically. Even with multiple REQ clients talking to the same REP "server," this relationship was always true (i.e., the expectation is a REP would reply to the client before moving on). With a REQ-REP combination, ZeroMQ handled details like "Where do I send this reply back to?" for us.

With a ROUTER socket, we don't have the promise of a synchronous traffic pattern. We could get a message and choose to reply to it or perhaps forward that message along to another socket. To make this flexibility possible, ZeroMQ has to have a way for the application using a ROUTER socket to know where the message is coming from so that it can either reply to the source or pass the info on so that someone else can. This is implemented in a super simple fashion: Messages consist of multiple frames that have a source identifier frame, message contents frames, and a blank frame in between these two things. In the code above, `recv_multipart` returns these three parts.

Now, the fact that the REQ socket is talking to a ROUTER doesn't change the REQ socket's synchronous nature. A REQ socket always needs to get a response back to a message it sends. We accommodate this need by having our ROUTER code echo back the message it received. By sending a message using the same three frames (via `send_multipart()`), the message is destined for the client that made the request and everybody is happy.

I'll run this code and see what happens. First, I'll start the ROUTER-based server and then spin up three REQ-based clients simultaneously. Here's some of the output from the server:

```
Server received: 1:72551
Server received: 1:72550
Server received: 1:72549
Server received: 2:72551
Server received: 2:72549
Server received: 2:72550
Server received: 3:72549
Server received: 4:72549
Server received: 3:72551
Server received: 4:72551
Server received: 3:72550
```

```
Server received: 4:72550
Server received: 5:72549
Server received: 5:72551
Server received: 6:72551
Server received: 5:72550
Server received: 6:72550
Server received: 6:72549
Server received: 7:72549
```

This output shows each message the server receives. The first number is the request number the client sends, the second is what each client is calling itself (it's that client's PID). It is not exactly easy to tell that the ROUTER socket is behaving asynchronously, but it is.

If we wanted to go in the other direction and have a DEALER talk to a REP socket, that approach works swimmingly as well. As the ZeroMQ handbook says (referring to a previous REQ-REP example): "This gives us an asynchronous client that can talk to multiple REP servers. If we rewrote the 'Hello World' client using DEALER, we'd be able to send off any number of 'Hello' requests without waiting for replies." As in the previous example, we need to think a bit harder about actual message frames. The message our DEALER sends out should mimic the same format a REQ socket might send. This is spelled out more explicitly in the doc than I want to get into here; but, in short, that's sending out a spacer frame followed by the message contents in a separate frame. The tools are exactly the same as in the previous example (`send_multipart()`), so I won't repeat all of that code.

I'd like to show another architecture before this column is done, but I suspect you might be curious about the combination we haven't discussed, namely DEALER-ROUTER combinations. Not to be too glib but, yup, that works great, too. If you pair two asynchronous socket types, you can make spiffy things like a component that can sit in between front-end and back-end pieces, asynchronously receiving messages and fanning them out as desired.

PUB and SUB

The final socket types we'll look at provide a slightly different paradigm than the ones we've seen so far. This paradigm came up when I talked about Redis a few columns ago and is becoming more common these days. With the publisher-subscriber (pub-sub) paradigm, there is a component that sends out messages (the publisher) that get "tagged" as being associated with specific topics. Some number of clients can then connect to the publisher, indicate interest in one or more of those topics, and then receive just the messages tagged with those topics.

I'll let some code do the introduction. Here's the publisher (the server that clients will connect to in order to receive their content):

```

use ZMQ::FFI;
use ZMQ::FFI::Constants qw(ZMQ_PUB);

my $ctx = ZMQ::FFI->new();
my $socket = $ctx->socket(ZMQ_PUB);

my $rv = $socket->bind('tcp://127.0.0.1:8888');

# bogus way to sync
sleep 5;

while (1) {

    $socket->send('bunnies furry');
    $socket->send('doggies barky');
    $socket->send('fishies swimmy');

    sleep 1;
}

```

This should look remarkably like the other examples, because, well, it is. The only difference is that we've changed the socket type. I consider the lack of difference to be a plus because it demonstrates how it is easy to build on prior knowledge when working with ZeroMQ. Before turning to the slightly more interesting "client," I do want to briefly discuss the comment above about the `sleep()` call being a bogus way to sync.

When dealing with a publisher-subscriber model, one of the common problems a first-time coder encounters is a synchronization problem. If the publisher publishes information before all of the subscribers have connected, the latecomers will miss messages. This isn't always a problem (e.g., the above code which repeats until interrupted), but in most real-world cases, it's highly suboptimal. There are a number of ways to fix this; the one above, where we just wait a bit, is probably the worst.

Far preferable would be for the publisher to have some way to know when all of the subscribers are present and listening. If you know how many subscribers constitutes "all," the easiest way is to dedicate a second socket pair in your code to just this sort of out-of-band signaling. You could imagine using a REQ-REP socket pair where the subscriber checks in to the publisher by sending something over the REQ, and the publisher acks the notice on the REP (basically using the code we've seen before). Once the publisher is satisfied everyone is tuned in, it can then begin sending actual content.

Now let's look at a simple subscriber:

```

use ZMQ::FFI;
use ZMQ::FFI::Constants qw(ZMQ_SUB);

my $ctx = ZMQ::FFI->new();
my $socket = $ctx->socket(ZMQ_SUB);

```

```

$socket->connect('tcp://127.0.0.1:8888');

my @topics = qw(bunnies doggies fishies);

# subscribe to a random topic
my $interest = $topics[ int( rand(3) ) ];
$socket->subscribe($interest);

while (1) {

    my ( $topic, $message ) = split / /, $socket->recv();
    print "Today I learned that $topic are $message!\n";

}

```

Again, super simple. The code picks a random topic and registers interest in this topic via a `subscribe()` call. From that point on, it will only "hear" messages on that topic when it calls `recv()`, like so:

```

Today I learned that bunnies are furry!
Today I learned that bunnies are furry!
Today I learned that bunnies are furry!
Today I learned that bunnies are furry!

```

If I were to move the topic subscription code into the loop so it picks a random topic on the fly:

```

while (1){

    # subscribe to a random topic
    my $interest = $topics[ int( rand(3) ) ];
    $socket->subscribe($interest);

    my ( $topic, $message ) = split / /,$socket->recv();
    ...
}

```

we get the expected results:

```

Today I learned that fishies are swimmy!
Today I learned that doggies are barky!
Today I learned that fishies are swimmy!
Today I learned that bunnies are furry!
Today I learned that doggies are barky!

```

I won't show you the server output (because you are probably fully up on your bunnies vs. doggies distinction), but we can easily run a metric ton of subscribers at the same time against our publisher. And, as before, no code changes are necessary to support going from one-to-one connections to multi-to-one.

Pretty cool, so with that, let's wrap. Take care, and I'll see you next time.

Command Line Option Parsing

DAVID BEAZLEY



David Beazley is an open source developer and author of the *Python Essential Reference* (4th Edition, Addison-Wesley, 2009). He is also known as the creator of Swig (<http://www.swig.org>) and Python Lex-Yacc (<http://www.dabeaz.com/ply.html>). Beazley is based in Chicago, where he also teaches a variety of Python courses. dave@dabeaz.com

If I look back, an overwhelming number of the Python programs I have written have been simple scripts and tools meant to be executed from the command line. Sure, I've created the occasional Web application, but the command line has always been where the real action is. However, I also have a confession—despite my reliance on the command line, I just can't bring myself to use any of Python's built-in libraries for processing command line options. Should I use the `getopt` module? Nope. Not for me. What about `optparse` or `argparse`? Bah! Get out! No, most of my programs look something like this:

```
#!/usr/bin/env python
# program.py
...
... Something or another
...
if __name__ == '__main__':
    import sys
    if len(sys.argv) != 3:
        raise SystemExit('Usage: %s infile outfile' % sys.argv[0])
    infile = sys.argv[1]
    outfile = sys.argv[2]
    main(infile, outfile)
```

Sure, the exact details of the options themselves might change from program to program, but, generally speaking, the programs all look about like that. Should things start to get more complicated, I'll ponder the situation a bit before usually concluding that I should probably just keep it simple. Again, I'm not proud of this, but it's a fairly accurate description of my day-to-day coding. In this article, I'm going to visit the topic of command line option parsing. First, I'll quickly review Python's built-in modules and then look at some newer third-party libraries that aim to simplify the problem in a more sane manner.

Command Line Parsing in the Standard Library

As background, let's consider the options for command line option parsing in the standard library. First, suppose your program had a main function like this:

```
def main(infiles, outfile=None, debug=False):
    # Imagine real code here. We'll just print the args for an example
    print(infiles)
    print(outfile)
    print(debug)
```

In its most basic use, suppose you wanted the program to simply take a list of input files to be provided as the `infiles` argument. For example:

```
% python prog.py infile1 ... infileN
```

In addition, suppose you wanted the command line interface to have an optional outfile argument provided by an `-o` or `--outfile=` option like this:

```
% python prog.py -o outfile infile1 ... infileN
% python prog.py --outfile=outfile infile1 ... infileN
%
```

Finally, suppose the debug argument is provided by an optional `-d` or `--debug` option. For example:

```
% python prog.py --debug -o outfile infile1 ... infileN
%
```

At the lowest level, the `getopt` module provides C-style command line parsing. Here is an example of how it is used:

```
# prog.py
...

if __name__ == '__main__':
    import getopt
    usage = '''\
Usage: prog.py [Options]

Options:
-h, --help      show this help message and exit
-o OUTFILE
--output=OUTFILE
-d
--debug
'''
    try:
        optlist, args = getopt.getopt(sys.argv[1:], 'dho:',
['output=', 'debug', 'help'])
    except getopt.GetoptError as err:
        print(err, file=sys.stderr)
        print(usage)
        raise SystemExit(1)

    debug = False
    outfile = None
    for opt, value in optlist:
        if opt in ['-d', '--debug']:
            debug = True
        elif opt in ['-o', '--output']:
            outfile = value
        elif opt in ['-h', '--help']:
            print(usage)
            raise SystemExit(0)

    main(args, outfile, debug)
```

If that's a bit too low-level for your tastes, you can move up to the `optparse` module instead. For example:

```
# prog.py
...

if __name__ == '__main__':
    import optparse
    parser = optparse.OptionParser()

    parser.add_option('-o', action='store', dest='outfile')
    parser.add_option('--output', action='store', dest='outfile')
    parser.add_option('-d', action='store_true', dest='debug')
    parser.add_option('--debug', action='store_true',
dest='debug')
    parser.set_defaults(debug=False)

    opts, args = parser.parse_args()
    main(args, opts.outfile, opts.debug)
```

Or, if you prefer, you can use the more recent `argparse` module. For example:

```
# prog.py
...

if __name__ == '__main__':
    import argparse
    parser = argparse.ArgumentParser()

    parser.add_argument('infiles', metavar='INFILE', nargs='*')
    parser.add_argument('-o', action='store', dest='outfile')
    parser.add_argument('--output', action='store',
dest='outfile')
    parser.add_argument('-d', action='store_true', dest='debug',
default=False)
    parser.add_argument('--debug', action='store_true',
dest='debug', default=False)

    args = parser.parse_args()
    main(args.infiles, args.outfile, args.debug)
```

Both the `optparse` and `argparse` modules hide the details of command line parsing through extra layers of abstraction. They also provide error checking and the ability to create nice help messages. For example:

```
$ python prog.py --help
usage: prog.py [-h] [-o OUTFILE] [--output OUTFILE] [-d]
[--debug]
           [INFILE [INFILE ...]]

positional arguments:
  INFILE
```

Command Line Option Parsing

optional arguments:

```
-h, --help      show this help message and exit
-o OUTFILE
--output OUTFILE
-d
--debug
```

Interlude

Looking at the above examples, it might seem that either `optparse` or `argparse` should work well enough for parsing a command line. This is true. However, they are also modules that are difficult to remember—in fact, I always have to look at the manual (or at my own book). Moreover, if you look at the documentation, you'll find that both modules are actually massive frameworks that aim to solve every possible problem with command line options that might ever arise. It's often overkill for more simple projects—in fact, it usually just makes my overworked pea-brain throb. Thus, it's worth looking at some alternatives that might serve as a kind of middle ground.

docopt

One alternative to the standard libraries is to use `docopt` (<http://docopt.org>). The idea with `docopt` is that you simply write the help string that describes the usage. An option parser is then automatically generated from it. Here is an example:

```
# prog.py
'''
My program.

Usage:
  prog.py [-o OUTFILE] [-d] [INFILES ... ]
  prog.py [--outfile=OUTFILE] [--debug] [INFILES ...]
  prog.py (-h | --help)

Options:
  -h, --help          Show this screen.
  -o OUTFILE, --outfile=OUTFILE Set output file
  -d, --debug         Enable debugging
'''

if __name__ == '__main__':
    import docopt
    args = docopt.docopt(__doc__)
    main(args['INFILES'], args['--outfile'], args['--debug'])
```

In this example, the documentation string for the module describes the usage and command line options. The `docopt.docopt(__doc__)` statement then automatically parses the options directly from that. The result is simply a dictionary

where values for the various options are found. It's a neat idea that flips option parsing on its head—instead of specifying the options through a complicated API, you simply write the usage string that you want and it figures it out.

Click

A newer entry to the command line argument game is `Click` (<http://click.pocoo.org/>). `Click` uses decorators to annotate program entry points with a command line interface. Here is an example:

```
import click

@click.command()
@click.argument('infile', required=False, nargs=-1)
@click.option('-o', '--outfile')
@click.option('-d', '--debug', is_flag=True)
def main(infile, outfile=None, debug=False):
    print(infile)
    print(outfile)
    print(debug)

if __name__ == '__main__':
    main()
```

In this example, the `@click.command()` decorator declares a new command. The `@click.argument('infile', required=False, nargs=-1)` decorator is declaring the `infile` argument to be an optional argument that can take any number of values. The `@click.option()` decorators are declaring additional options that are tied to arguments on the decorated function.

Once decorated, the original function operates in a slightly different manner. If you call `main()` without arguments, `sys.argv` is parsed and used to supply the arguments. You can also call `main()` and provide the argument list yourself, which might be useful for testing. For example:

```
main(['--outfile=out.txt', 'foo', 'bar'])
```

One of the more interesting features of `Click` is that it allows different functions and parts of an application to be composed separately. Here is a more advanced example that defines two separate commands with different options:

```
# prog.py
import click

@click.group()
@click.option('-d', '--debug', is_flag=True)
def cli(debug=False):
    if debug:
        print('Debugging enabled')
```

```

@cli.command()
@click.argument('infile', required=False, nargs=-1)
@click.option('-o', '--outfile')
def spam(infile, outfile=None):
    print('spam', infile, outfile)

@cli.command()
@click.argument('url')
@click.option('-t', '--timeout')
def grok(url, timeout=None):
    print('grok', url, timeout)

if __name__ == '__main__':
    cli()

```

In this example, two commands (`spam` and `grok`) are defined. Here is an interactive example showing their use and output:

```

% python prog.py spam -o out.txt foo bar
spam (u'foo', u'bar') out.txt
% python prog.py grok http://localhost:8080
grok http://localhost:8080 None
%

```

The debugging option (`-d`), being defined on the enclosing group, applies to both commands:

```

% python prog.py -d spam -o out.txt foo bar
Debugging enabled
spam (u'foo', u'bar') out.txt
%

```

Corresponding help screens are tailored to each command.

```

% python prog.py --help
Usage: prog.py [OPTIONS] COMMAND [ARGS]...

Options:
  -d, --debug
  --help          Show this message and exit.

Commands:
  grok
  spam

% python prog.py spam --help
Usage: prog.py spam [OPTIONS] [INFILES]...

Options:
  -o, --outfile TEXT
  --help          Show this message and exit.

%

```

The ability to easily compose commands and options is a powerful feature of Click. In many projects, you can easily have a large number of independent scripts, and it can be difficult to keep track of all of those scripts and their invocation options. As an alternative, Click might allow all of those scripts to be unified under a common command line interface that provides nice help functionality and simplified use for end users.

Final Words

If you write a lot of simple command line tools, looking at third-party alternatives such as `docopt` and Click might be worth your time. This article has only provided the most basic introduction, but both tools have a variety of more advanced functionality. One might ask if there is a clear winner. That, I don't know. However, for my own projects, the ability to compose command line interfaces with Click could be a big win. So I intend to give it a whirl.

Resources

<http://docs.python.org/dev/library/getopt.html>
(`getopt` module documentation).

<http://docs.python.org/dev/library/optparse.html>
(`optparse` module documentation).

<http://docs.python.org/dev/library/argparse.html>
(`argparse` module documentation).

<http://docopt.org> (`docopt` homepage).

<http://click.pocoo.org> (Click homepage).

iVoyeur

7 Habits of Highly Effective Monitoring Systems

DAVE JOSEPHSEN



Dave Josephsen is the sometime book-authoring developer evangelist at Librato.com. His continuing

mission: to help engineers worldwide close the feedback loop. dave-usenix@skeptech.org

Having recently returned from Monitorama [1], I can attest that it is exactly what it sounds like: a collection of people so enamored of the technical discipline that has been my obsession for the better part of the last decade that they literally fly from the far corners of the world in order to shut themselves up in a single room and geek out about it for days. There are drunken diatribes about RabbitMQ in the context of metric transmission, hallway arguments about whether CPU percentage or load average is the superior metric of computational stress, and diabolical plots to compress time series data by converting it to frequency space. My point is, this conference could not be more custom-tailored to please me were we gathering in a fellowship quest to craft the ultimate bacon, lettuce, and tomato sandwich.

If there was a theme that permeated the event, I think it was to be found in the contrast between two very specific kinds of talk. The first type is the kind given by someone attempting to apply mathematical (usually statistical, but sometimes signal processing) techniques to detect aberrant behavior in time series data. These are always technical and, with a few notable exceptions [2], do not attempt to practically apply their findings via a tool the rest of us can experiment with. They customarily provide an overview of relevant mathematical techniques, usually beginning with simple thresholds, moving through standard deviation and various types of exponential moving averages like Holt-Winters, and winding up somewhere in the vicinity of forward decaying priority sampling. At this point, they usually throw up their hands, mutter something about domain-specific knowledge and monitoring data being a non-Gaussian distribution, and ask for questions.

The second type of talk is the kind given by an engineer who has implemented a monitoring system that seems to be working for them at the moment. It is often a tenuously wired together Frankenstein's monster that will almost certainly look different the next time we see it (which is fine if it's solving their problems). To be clear, I greatly enjoy both of these kinds of talks. If there were a cable channel that brought me only this content, I would never leave the house.

Automated fault detection is absolutely worth pursuing, I'm excited about it, and I have no doubt there will be breakthroughs as we get more eyes on it. Further, it's always fascinating to hear about the real-life trials and tribulations of my fellow plumbers who are holding things together in their respective corner of the Internet. Their every success is a ray of glorious hope that brightens my day.

Being repeatedly subjected to these two types of talks back to back, however, was, I have to admit, a little disheartening. The contrast between the cold mathematical certainty promised by the former type compared to the banal reality of the latter really got me thinking about the current state of monitoring as I've personally witnessed it. Aren't there real-life monitoring systems out there that are purposefully designed, elegantly engineered, and that meet 100% of the needs of every engineering team in their respective organization? Yes, as a matter of fact, I happen to know that there are well-engineered monitoring systems that world-class IT shops are happy with: Systems that sure would benefit from automatic fault

detection, but wouldn't be defined by it. Systems that are worlds away from the cobbled together collections of tools from the second kind of talk—the kind, I might add, that the preponderant quantity of attendees I spoke to are running. And yet, the monitoring systems I'm talking about are often composed of those same pieces, but somehow manage to become more than the sum of their parts.

I like to think I've done a good job of resisting the urge to pontificate about the state of monitoring in general in this column, focusing instead on interesting tools and techniques. It just seems presumptuous of me to tell you what to install and how, but Monitorama has left me with both a burning desire to spout off at the mouth about monitoring theory and the feeling that I may have been remiss in avoiding it in the past. So, I give you my take on the current state of how to monitor well, organized into seven habits that summarize what the good systems are doing right today.

Habit 1: It's About the Data

Were I in a darker mood, I might have titled this “Stop Looking for an Ubertool.” Awesome monitoring systems value data over tools—they understand that a monitoring tool is merely a means to obtain data. They treat metrics and telemetry data as first-class citizens and rarely leave it to rot within the tool that collected it. Rather, they send the data “up” to be processed, stored, and analyzed together with all of the other data collected by all the other tools, on all the other systems, organization-wide.

When you make the data a first-class citizen, you wind up with data-centric tools that enable you to correlate measurements taken from any layer of the stack. You can, for example, quantify the effect of JVM garbage collection on service latency, or if the number of calls to the `foo()` function in your application across three different nodes correlates to the odd behavior in the byte counter that resides on the switch they are all connected to. You know you are doing it right when you can “tee” off a subset of your monitoring data at will and send it as input to any new tool you might decide to use in whatever format that tool expects.

Now that I've made a big deal about it not being about tools, let's talk about the kinds of tools that let data thrive, beginning with an example of what not to do. I'll go ahead and pick on Nagios for this, since that's so in-vogue these days. Nagios was designed for a very specific job, namely, to collect availability data on services and hosts on the order of minutes (usually about every five minutes).

This is useful data to collect, and Nagios is, in my opinion, the best tool for accomplishing this task. It also makes some annoying assumptions about how you want to process the data it collects, and those assumptions make it more difficult than it should be to get data out of Nagios and into other tools. This

is evidenced by the plethora of single-purpose tools that have sprung into being for no other purpose than to take data from Nagios and place it in X, where X is some other monitoring tool from which it is usually even more difficult to extract the monitoring data.

And so it is that we devolve into this anti-pattern of implementing the tool we think we need, and then more tools to connect our tool to yet other tools in an attempt to make up for some deficiency in the one we thought we wanted. The complexity of our monitoring efforts grows quadratically as our chosen tool bogs down with every new tool we bolt onto it. God help us if we ever want to connect a tool to the tool that's connected to the original tool, because our data just gets more and more specific, ever-increasingly locked-in to the toolchain we've painted ourselves into.

If, however, we recognize that Nagios is merely one of many data collectors and place a transmission layer above Nagios that is designed to accept metrics data from any sort of data collector so that it can be processed and persisted in a common data format, our tools no longer depend on each other, and we have a single source of telemetry data that we can wire to any tools that make sense. Obviously, I think this is a fantastic idea, and I even began to implement it myself [3] before Riemann [4] and Heka [5] did a much better job of it.

Habit 2: Use Monitoring for Feedback

Who is choosing your metrics? Are you using a turnkey agent that collects umpteen hundred metrics from every node that you install it on? How many of those metrics do you track? How many do you alert on? Great monitoring systems are driven by purpose. They are designed to provide operational feedback about production systems to people who understand how those systems work—people who have chosen what to monitor about those systems based on that knowledge.

Monitoring isn't a “thing”; it does not stand on its own. It is not a backup system or a disaster recovery plan, or any other sort of expensive and annoying burden heaped on Ops to satisfy the checklist requirements of a regulatory body or an arbitrary quarterly goal. It is not a ritual that grown-ups tell us to follow—like keeping our hands and arms inside the vehicle at all times—a habit we all must perform to stave off some nameless danger that no one can quite articulate.

Monitoring is an engineering tool. It exists to provide closed-loop feedback from engineering systems. It is the pressure meter on your propane tank. Through monitoring, we gain visibility into places we cannot go, and we prevent explosions from happening in those places. The engineers in your organization should understand the metrics you monitor, because each should have been configured by an engineer to answer a specific question or provide a concrete insight about the operational characteristics of your service.

Habit 3: Alert on What You Draw

When an engineer in your organization receives an alert from a monitoring system, and moves to examine a graph of monitoring data to analyze and isolate the problem, it's critically important that the same data was used to generate both the alert and the graph. If, for example, you're using Nagios to check and alert, and Ganglia to draw the graphs, you're raising the likelihood of uncertainty, stress, and human error during the critically important time of incident response.

One monitoring system or the other could be generating false positives or negatives; they could each be monitoring subtly different things under the guise of the same name, or they could be measuring the same thing in subtly different ways. It actually doesn't matter, because there is likely no way to objectively tell which system is correct without a substantial effort, and even if you do figure out which is lying, it's unlikely you will be able to take a meaningful corrective action to synchronize the behavior of the systems.

Ultimately, what you've done is shifted the problem from "improve an unreliable monitoring system" to "make two unreliable monitoring systems agree with each other in every case." The inevitable result is simply that your engineers will begin to ignore both monitoring systems because neither can be trusted.

Great monitoring systems require a single source of truth. In the current example, the most expedient way to achieve this is to configure Nagios to monitor thresholds in Ganglia's data [6] (because Ganglia has the best resolution). The concept of a single source of truth is a fundamental requirement to good systems monitoring. It's also another great argument in favor of focusing on data rather than tools.

Habit 4: Standardize Processing, but Emancipate Collection

I've run into business consultants who were convinced that the proper way to implement monitoring solutions was to first create a plan that lists every possible service that you could ever want to monitor and then choose a tool that meets your data collection list. In my experience, great monitoring systems do the opposite. They plan and build a substrate—a common, organization-wide service for processing telemetry data from monitoring systems—like the ones I described above in Habit #1. Then they enable and encourage every engineer, regardless of team affiliation or title, to send monitoring data to it by whatever means necessary.

Awesome monitoring systems standardize the metrics processing, storage, analysis, and visualization tools, but they declare open season on data collectors. One shop whose engineers I've

spoken with (apologies, I've forgotten which) has the motto "new metrics in minutes." Every engineer should be free to implement whatever means she deems appropriate to monitor the services she's responsible for. Monitoring new stuff should be hassle-free.

Habit 5: Let the Consumers Curate

Another popular notion about monitoring systems in the corporate world is that they should provide a "single pane of glass," by which I assume they mean the monitoring system should have a single, primary dashboard that shows a high-level overview of the entire system state.

That's great and I'm not necessarily arguing against it, but the best monitoring tools I've seen focus instead on enabling engineers to create and manage their own dashboards, thresholds, and notifications. If you're doing it right, you should have a dashboard for every service that your team supports or contributes to, curated by your team members. Effective monitoring systems don't just allow non-ops engineers to interact with the system, they demand it.

Great monitoring systems are timely, open, and precise. They represent a single source of truth that is so compelling and easy to interact with that the engineers naturally rely on them to understand what's going on in production. When they want to track how long a function takes to execute in production, they should naturally choose to instrument their code and observe feedback using the monitoring system. When they have an outage, their first thought should be to turn to the dashboard for that service before they attempt to ssh to one of the hosts they suspect is involved.

A monitoring system that requires coercion for adoption isn't solving the right problems. So, if your engineers are avoiding the monitoring system, or ignoring it, or rolling their own tools to work around it, then you have an impedance problem, and you should ask yourself why they prefer the tools they do over yours, and focus in on making it easier for the consumers of the system to use it to solve their problems.

Habit 6: Evolve by Tiny Iterations

Healthy monitoring systems don't need a semi-monthly maintenance procedure. They stay relevant because they're constantly being iterated by the engineers who rely on them to solve everyday problems. New metrics are added by engineers who are instrumenting a new service or trying to understand the behavior of some misanthropic piece of infrastructure or code. Measurements are removed when they're no longer needed by the team that put them there—because they're superfluous and cluttering up the dashboards.

By focusing on the data, relying on the accuracy of the results, and enabling everyone to iteratively fix the pieces they rely on, your monitoring system will evolve into exactly what your organization needs it to be, rather than a complicated ball of cherished tools tenuously strung together that everyone ignores except the dude holding the string.

Habit 7: Instrumentation != Debugging

Monitoring is unit testing for operations. For all distributed applications—and, I'd argue, for a great deal of traditional services—it is the best if not the only way to verify that your design and engineering assumptions bear out in production.

Further, instrumentation is the only way to gather in-process metrics that directly correspond to the well-being and performance of your production applications.

Therefore, instrumentation is code. It is a legitimate part of your application—not extraneous debugging rubbish that can be slovenly implemented with the implicit assumption that it will be removed later. Your engineers should have libraries at their disposal that enable them to thoughtfully and easily instrument their application in a way that is commonly understood and repeatable. Libraries like Coda Hale Metrics [7] are a fantastic choice if you don't want to roll your own. In the same way your feature isn't complete until you provide a test for it, your application is not complete until it is instrumented so that its inner workings can be verified by the monitoring data stream.

As always, I hope you found something helpful in this diatribe. As the DevOps revolution continues to utterly confound and mystify the IT managementosphere, I think we have a golden

opportunity to reinvent monitoring. My hope is that we can expand it from a thing that operations does because: computers, and replace it with a commons—supported by Ops—that welcomes measurements from every type of engineer and encourages them to define their own interactions, no matter how convoluted their title. To the extent we achieve this, I believe we will improve the transparency of both our services and infrastructure, increase our understanding of the systems we support, and carry with us quieter paggers.

Good luck!

References

- [1] The Monitorama conference: <http://monitorama.com>.
- [2] Abe Stanway, Jon Cowie: "Bring the Noise," Velocity Santa Clara 2013: <https://www.youtube.com/watch?v=3nF426i0cBc>.
- [3] Hearsay: <https://github.com/djosephsen/Hearsay>.
- [4] Riemann: <http://riemann.io/>.
- [5] Mozilla, Introducing Heka: <http://blog.mozilla.org/services/2013/04/30/introducing-heka/>.
- [6] Monitoring Ganglia data from Nagios: <https://github.com/ganglia/monitor-core/wiki/Integrating-Ganglia-with-Nagios>.
- [7] Coda Hale Metrics: <http://metrics.codahale.com/>.

Almost Too Big to Fail

DAN GEER AND JOSHUA CORMAN



Dan Geer is the CISO for In-Q-Tel and a security researcher with a quantitative bent. He has a long history with the USENIX Association, including officer positions, program committees, etc.

dan@geer.org



Joshua Corman is the chief technology officer for Sonatype. Previously, Corman served as a security researcher and strategist at Akamai

Technologies, The 451 Group, and IBM Internet Security Systems. A respected innovator, he co-founded Rugged Software and I Am the Cavalry to encourage new security approaches in response to the world's increasing dependence on digital infrastructure. He is also an adjunct faculty for Carnegie Mellon's Heinze College, IANS Research, and a Fellow at the Ponemon Institute. Josh received his bachelor's degree in philosophy, graduating summa cum laude, from the University of New Hampshire. joshcorman@gmail.com

Both dependence on open source and adversary activity around open source are widespread and growing, but the dynamic pattern of use requires new means to estimate if not bound the security implications. In April and May 2014, every security writer has talked about whether it is indeed true that with enough eyeballs, all bugs are shallow. We won't revisit that topic because there may be no minds left to change. Unarguably:

- ◆ Dependence on open source is growing in volume and variety.
- ◆ Adversary interest tracks installed base.
- ◆ Multiple levels of abstraction add noise to remediation needs.

We begin with two open source examples.

Apache Struts CVE-2013-2251, July 6, 2013 - CVSS v2 9.3

Apache Struts is one of the most popular and widely depended upon open source projects in the world. As such, when this highly exploitable vulnerability was discovered, it was promptly used to compromise large swaths of the financial services sector. While Heartbleed (see below) got full media frenzy, many affected by 2013-2251 learned of the problem from FBI victim notifications under 42 U.S.C. § 10607. The FS-ISAC issued guidance [1] telling institutions (read, victims) to scrutinize the security of third-party and open source components throughout their life cycle of use. It is not noteworthy that an open source project could have a severe vulnerability; what *is* of note is that this flaw went undetected for at least seven years (if not a lot longer from WebWork 2/pre-Struts 2 code base)—an existence proof that well-vetted code still needs a backup plan.

OpenSSL (Heartbleed) CVE-2014-0160, April 7, 2014 - CVSS v2 5.0

The Heartbleed vulnerability in OpenSSL garnered tremendous media and attacker activity this past April. While only scored with a CVSS of 5.0, it is a “5 with the power of a 10” since sniffing usernames, passwords, and SSL Certificates provides stepping stones to far greater impact. In contrast to the Struts bug above, this flaw was introduced only two years prior, but it, too, went unnoticed by many eyeballs—it was found by bench analysis [2].

Dependence on Open Source Is Growing

Sonatype, home to author Corman, serves as custodian to Central Repository, the largest parts warehouse in the world for open source components. At the macro level, open source consumption is exploding in Web applications, mobility, cloud, etc., driven in part by increasingly favorable economics. Even (risk averse, highly regulated) government and financial sectors, which previously resisted “code of unknown origin/quality/security,” have begun relaxing their resistance. According to both Gartner surveys and Sonatype application analysis, 90+% of modern applications are not so much written as assembled from third-party building blocks. It is the open source building blocks that are taking the field, and not just for commodity applications (see Figure 1).

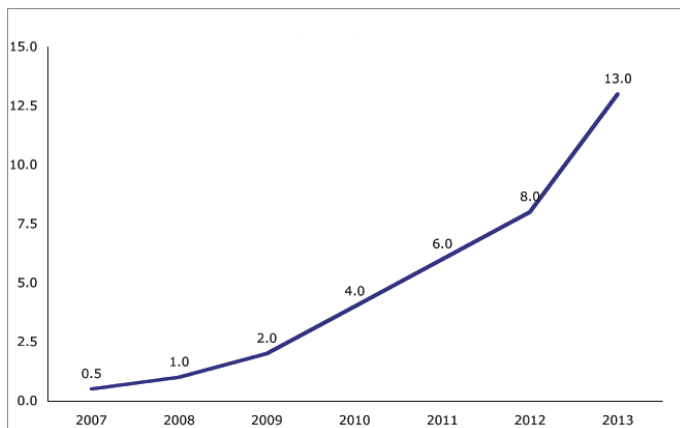


Figure 1: Open source downloads per year measured in billions

Adversary Interest in Open Source Is Growing

Adversary interest tracks component prevalence. The prevalence of open source has grown, ergo so has adversary interest [3]. There are several equivalent ways to characterize that:

- ◆ Payoff: “That’s where the money is.”
- ◆ Cost-effective leverage: Unless you are engaged in one-off targeting, you go after the components that are most depended upon (Struts, OpenSSL, etc.).
- ◆ Accessibility: Obscurity may occasionally contribute to security, but there is nothing obscure about an open source code pool.

Figure 2 shows the pattern of vulnerability disclosure in the Apache Struts project; the vertical axis shows CVSS severity against the horizontal showing calendar time.

While author Geer has written elsewhere [4] about how CVSS scores are *not* the way to steer remediation efforts, Figure 2 does confirm that there is a mounting interest in cataloging open source flaws. (See also author Corman’s “HDMoore’s Law” [5].)

Can We Characterize Flaw Response?

Yes, Virginia, all software has flaws, but one might ask whether we avoid “known bad components” when assembling deliverable code? Not always; consider:

“Bouncy Castle” CVE-2007-6721, November 10, 2007 CVSS v2 10

The “Legion of the Bouncy Castle Java Cryptography APIs” had a CVSS worst-case scenario fixed in April of 2008—more than six years ago. While 2007-6721 is a severe security flaw in a security-sensitive project, nevertheless the unrepaired, vulnerable version was requested from Central Repository 4,000 times in 2013. One can assume it was used in security-related applications/products, perhaps multiple applications per download instance.

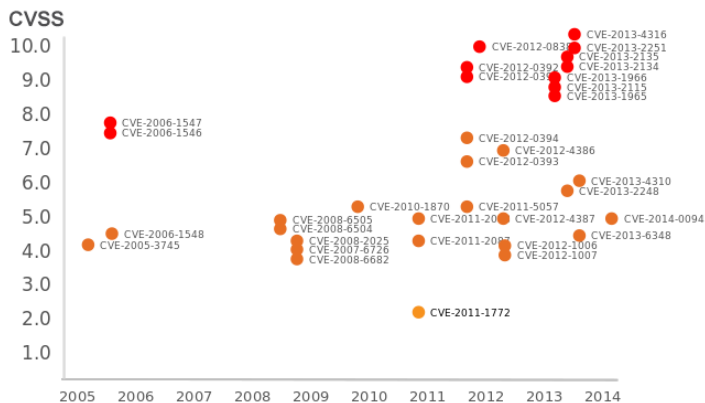


Figure 2: Graphing the CVSS severity (1-10) for disclosed Struts vulnerabilities against the year shows generally increasing severity levels.

Similar (disappointing) consumption patterns exist for Struts. Outside of *CVE-2013-2251* compromised organizations, still-vulnerable versions of Struts 2 continue to remain popular. Worse, Struts version 1-related artifacts still had over a million downloads in 2013, despite its April 5, 2013 official End of Life. In other words, finding and fixing serious flaws in open source does not mean that the repaired versions are the ones that are used. Is this an awareness problem, or is it something else?

Readers will recall that Availability (A) is calculated as

$$A = \frac{MTBF}{MTBF + MTTR}$$

where MTTR is Mean Time To Repair and MTBF is Mean Time Between Failures. Availability is thus perfect (100%) if either the item never fails (MTBF goes to infinity) or the item enjoys instant recovery (MTTR goes to 0). This is where a distinction between open and closed source may be operationally relevant: If the MTBF is a constant, then MTTR is what matters. The 2013 Coverity Scan Report [6] showed comparable defect rates between open and closed source projects (with a slight quality advantage for open source projects). If project sizes are also comparable, then MTBF between open and closed source would likewise be comparable.

We have less data on MTTR, whether for closed or open source, but it is our educated guess that (once fixed) open source project repairs are *available* earlier than closed source projects because the latter will have additional packaging and deployment steps. Open source projects are not responsible for deployment of fixes, only the availability of fixes, and, even then, there is no forcing function for making fixes available. In a sense, Heartbleed was a blessing; it showed us just how widespread one error can be deployed and just how much widespread use led users to assume that it must have been thoroughly scrubbed by somebody else by now.

Almost Too Big to Fail

But to base our discussion on knowledge rather than educated guesses, Sonatype has begun an analysis of the “project integrity” of the open source codebases it hosts. One focus will, in fact, be MTTR. It is central to the open source domain because there are unobvious transitive dependencies between and among open source components. An early analysis of open source projects with already identified vulnerable dependencies revealed some troubling behavior. Direct (aka “1-hop”) vulnerable component dependencies were only remediated 41% of the time. Put differently, more than half (59%) of the vulnerable base components remain unrepaired. Folding multiple components into your projects means inheriting not just the components’ functionality but also their (largely unrepaired) flaws. For the 41% that were fixed at all, the MTTR was 390 days (median 265 days). Filtering for just CVSS 10s brought the mean of this subset down to 224 days. And this is just for 1-hop dependencies—there is as yet no mechanism to cause remediated flaws to flow automatically through the dependency graph, and there may never be.

Making Remediation Possible

In closed source development domains, the command structure will know who uses what and can thus ascertain what code trees have to be rippled when a common component is revised. This is not the case with open source, nor will it be. As Heartbleed made clear, open source is in home electronics, medical devices, industrial controls, etc. The more widespread the use of a particular open source library, the more common mode failure among otherwise unrelated product spaces becomes. An auto manufacturer can recall a particular model, and know that only that model has the faulty component. There is no feasible equivalent for an open source library. We thus suggest that, just as a jar of pickles on the grocery shelf must list its ingredients, products and services that are assembled from open source components need to provide a bill of materials so that when an open source component has a vulnerability, downstream users can tell whether they are affected and whether a particular remediation is one they need to consume (directly or indirectly). Ingredients lists would serve as a framework both for remediation and for further work in security metrics.

To emphasize the concreteness of these issues, embedded systems are largely assembled from open source components, have no field upgrade path once deployed, and had build environments that were not coordinated with source code control. We have work to do.

References

- [1] Financial Services Information Sharing and Analysis Center, “Appropriate Software Security Control Types for Third Party Service and Product Providers”: docs.ismgcorp.com/files/external/WP_FSISAC_Third_Party_Software_Security_Working_Group.pdf.
- [2] F. Berkman, “Researcher Who Discovered Heartbleed Bug Donates \$15K Reward,” *The Daily Dot*: www.dailydot.com/news/heartbleed-neel-mehta-freedom-press-foundation-encryption.
- [3] S. Rosenblatt, “Heartburn from Heartbleed Forces Wide-Ranging Rethink in Open Source World,” CNET: www.cnet.com/news/heartburn-from-heartbleed-forces-wide-ranging-rethink-in-open-source-world.
- [4] D. Geer and M. Roytman, “Measuring vs. Modeling,” *USENIX ;login.*, vol. 38, no. 6 (December 2013): geer.tinhow.net/fgm/fgm.geer.1312.pdf.
- [5] J. Corman, “Intro to HDMoore’s Law”: blog.cognitivedissidents.com/2011/11/01/intro-to-hdmoores-law/.
- [6] Coverity Scan, 2013 Open Source Report: softwareintegrity.coverity.com/rs/coverity/images/2013-Coverity-Scan-Report.pdf.

/dev/random

ROBERT G. FERRELL



Robert G. Ferrell is a fourth-generation Texan, literary techno-geek, and finalist for the 2011 Robert Benchley Society Humor Writing Award. rgferrell@gmail.com

I was reading about the “Do Not Track” initiative on *Engadget* the other day. That’s where, along with the page request, your browser includes an environmental variable or flag indicating that you don’t want your browsing habits tracked. Now, perhaps I’m leaping to conclusions here (I do that so often I had to have a hot tub installed for the cramps), but I’m guessing most advertisers are already aware of that. It’s a bit like telling a rabid dog that you prefer not to be mangled at this time. Unless you have a cat like “Tara” in California, then bring it on.

Tara is the Internet’s darling as I write this, but by the time you read it I’m sure the meme will have faded into obscurity; if you care enough, you’ll have to search for it and wade through stuff about 1930s movies and Buddhism before you find anything related to cats. Writers have been dealing with publication delays for decades if not centuries; they just make life that much more interesting, and they’re not at all awkward. So, how is the late May weather where you are?

Back on task, “Do Not Track” brought to mind a whole host of other opt-outs we should consider implementing. My car, for example, would benefit from a “Do Not Stall” flag. I’d also like to get a “Do Not Food Poison” flag for use at certain restaurants, along with “Do Not Overcharge” and “Do Not Expect a Large Tip Because You Left Me Sitting Here without Taking My Order or Even Anything to Drink for Forty-Five Minutes.”

Perhaps this initiative will spur a whole new series of browser flags. I’m thinking “Do Not Redirect to Some Stupid Two-Minute Advertising Video When All I Want Is the Current Temperature” and “Do Not Create Pop-Ups that Obscure the Story I’m Trying to Read” would be nice. I’d also like a “Do Not Insinuate That Because I Don’t Have the Latest Video Plugin My Browser Is Hopelessly Non-Compliant” flag. That one really pops my garters.

I was a Web designer back in the mid-to-late ‘90s, and I made sure that my clients’ content was visible in every conceivable browser/platform combination, even though it meant having a lot of different systems on hand and a ton of work. Nowadays the lazy so-and-sos just design their content for one specific system and cast aspersions on you if yours doesn’t happen to be that one, as though the fact that their content doesn’t display properly is somehow *your* fault. I don’t know what sort of advertising model they think that represents, but I call it “A Product I Will Cross the Street to Avoid Buying.”

The other headline-grabbing news event, currently neck and neck with Tara the cat tonguing out the first baseball, is the promise of dire consequences from the Chinese as a result of our rather puzzling indictment of five of their nationals for cyber-espionage. If you stop and think about the way the Internet was designed and the rather cavalier approach we as a community have taken to security thereon, charging one nation with spying on another takes on the mantle of a vaudeville act (pause for my younger readers to look that up).

The *mise-en-scène* is a classroom with five desks occupied by students in school uniforms, each representing a sovereign nation. They are taking an online exam, and each is obviously looking at the monitor screen of the student on the right. The student on the right end peri-

odically arises and casually strolls by the student on the left end on his way to the water fountain. He is apparently quite thirsty. After one of these trips the student on the left end jumps up and loudly accuses him of cheating. All of the characters are male, incidentally, because most women are too smart to fall into this trap.

The accused now indignantly levels the same charge at the student to his left. This process repeats sequentially until all stand accused, at which point they leap upon one on another and begin a melee with mice, keyboards, and external drives, pummeling and attempting to strangle one another with the USB cords. There is much pulling of hair, ripping of clothing, and use of shoes as projectiles. During the struggle, the teacher pushes a key on her own workstation and calmly walks to each desk, rotating the screen so the audience can see the big fat red "F" in the center of them. The curtain closes with the crazed student body trying to cram each other head first into their DVD drives, yelling "Cheater! Cheater! Cyber Bleeder!"

I suppose I would be amiss (which is as good as a mile) if I failed to at least mention in passing the infamous Heartbleed bug, which has caused so much heartache, not to mention heartburn, in my own era. This figurative deceased equine has already been subjected to a thorough and prolonged pummeling, so I think I'll just bypass the bug itself and dwell on the lessons learned: There

weren't any. Do you honestly think the next time (and every time after that) software developers are going to import libraries from another source they will subject them to a comprehensive security review before incorporating them into their own products? I chuckle derisively at your naïveté. The World Wide Web is built around the premise that the only way to deal with a mistake is to make another, worse, one that will draw attention away from the original.

When will the next big insufficient bounds checking scandal hit? By the time you read this, I expect. As with Tara, Heartbleed will probably be relegated to the ancient history archives, as Internetland now seems to regard anything that happened more than 14 days ago as the distant past. Yet another oblique corollary to Moore's Law is that the span of time during which any new technology is considered "bleeding edge" seems to be contracting. It is no longer an exaggeration to say that a newly released product ordered from an online vendor may well be obsolete, or at least obsolescent, before it is even delivered to your house. In some areas of technology, products are obsolete before they even roll off the assembly line.

In fact, by the time you read this column it will be thought of, when thought of at all, as a relic from an earlier age. Much like the author.

XKCD



xkcd.com

Book Reviews

RIK FARROW AND MARK LAMOURINE

The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win

Gene Kim, Kevin Behr, and George Spafford
IT Revolution Press 2013; 345 pages
ISBN 978-0988262591

Reviewed by Rik Farrow

I learned of this book while working on the LISA '14 tutorial committee. After reading it, I wondered how I had missed hearing about it previously. And although this is a novel, it also provides a deeper understanding of DevOps, something I hadn't encountered before.

The plot roughly follows that of Eliyahu Goldratt's *The Goal*, which has become a model for systems management ever since it was published in 1984. Bill, the protagonist, is the senior IT guy who has risen to management of the mid-level systems group. Bill is happy where he is and manages his own group well, but that safe harbor disappears in the opening chapter, when his CEO deftly maneuvers Bill into taking the VP of IT position. The former VP has disappeared under a cloud, and Bill quickly finds himself having to deal with one impossible situation after another.

For anyone who has worked in IT, the details of the story will sound familiar: failed releases, the over-ambitious project, a release deadline set by marketing, and an IT department that is not just fragmented, but fractious. Bill gets guidance from a new board member, who takes him on visits to a smoothly functioning factory. Rather than tell Bill what to do, the board member provides hints and leaves Bill to work things out on his own. In real life, you could read other Gene Kim books and get a head start. But Bill progresses through one disastrous release after another, getting a handle on development, quality assurance, security, testing, and release management.

I found the book easy to read and breezed through it. If you usually read novels driven by character development, you will find just traces of that here. The greatest benefit to reading this is getting a visceral, on the ground understanding of how workplace transformation can happen, given the right set of circumstances and personalities. *The Phoenix Project* is not a textbook, but it still gets across key ideas about controlling the acceptance of new projects, uncovering chokepoints, and how continuous integration actually makes software projects more successful and less expensive.

Penetration Testing: A Hands-On Introduction to Hacking

Georgia Weidman
No Starch Press, 2014; 531 pages
ISBN 978-1-59327-564-8

Reviewed by Rik Farrow

This is the book I wish I had when I was teaching my two-day, hands-on Linux security class. At more than 500 pages long, the book covers a lot more material and many more topics than I could in two days.

After a brief introduction, Weidman spends a long chapter on setting up four VMs: one for the pen testing, and three as targets. Although the pen tester's VM runs Kali, a Linux distro that already includes many tools for security, the author takes the time to explain how to install additional tools that will be used in exercises throughout the book. The target systems also get extra attention, with vulnerable apps getting installed. I like this approach, because it prepares the reader for what's to come, as well as encouraging the reader to do more than just read.

The next couple of chapters are the weakest, but they will need to be there for some readers. I do wonder how many people who can't use basic Linux commands will be successful with pen testing, even when using GUI-based tools like Wireshark, or how showing someone a short shell script or C program will help.

Once past this point, Weidman progresses rapidly, providing a quick overview of Metasploit from the command line. In part two, she guides the reader through vulnerability scanning, port scanning, and network packet capture. Weidman's explanations are clear and accurate, if a bit terse. And although she tells the reader to start up Wireshark as root and "click through the warnings about using Wireshark as root being dangerous," I wished she had explained why: that Wireshark itself can be exploited while parsing packets, and that being root makes any exploit much more useful to the attacker. In a book that teaches about vulnerabilities and exploits, I thought explaining this issue would both help with the pen tester's mind-set as well as act as a warning. I found myself imagining an organization's security team exploiting the pen tester's laptop, something which I know has been done, including by one of the people Weidman lists in her acknowledgments. At least Weidman describes running Kali from within a VM, partially excusing her exclusive use of the root account for all exercises throughout the book.

Weidman does a very nice job of explaining how stack-overflow exploits work, as well as going through examples of how to build these exploits. She does also point out that stack-overflows only work on older OS versions, before data execution prevention (DEP) became the norm for most software. Still, with the use of examples, she walks the reader through how these exploits work, essential knowledge for the person who wants to understand exploitation in the post-DEP and address space layout randomization (ASLR) defensive environment. And understanding how to write exploits forms the basis for modifying existing Metasploit exploits or writing new ones, which she covers in a chapter.

Weidman has developed the Smartphone Pentest Framework herself and covers this in the final chapter. SPF works with Android emulators, whose setup is described in the first chapter, but Weidman uses the framework to explain how attacks outside of the simulated environment should work. I did find myself cringing when she suggested changing the SSH password for the iPhone (alpine is the root password), but for the most part, her writing and exposition are solid.

There are also chapters on exploiting Windows and bypassing antivirus, among other topics.

If you are interested in understanding security from the perspective of the practitioner—that is, a pen tester or hacker—*Penetration Testing* will certainly do more than get you started. For many people I taught over the years, this book will explain more about the tools we used then, and about new tools and techniques.

Understanding Computation

Tom Stuart

O'Reilly Media, 2013. 315 pages.

ISBN 978-1-449-32927-3

Reviewed by Mark Lamourine

In *Understanding Computation*, Stuart sets out to provide something you don't often find in the computing aisle of the retail bookstore chains. Most books in this area are tutorials and references designed to achieve a level of popularity by focusing on the most recent languages and frameworks. Stuart, by contrast, takes on an Honest-To-God Theory of Computation. Although this would typically be an academic book, Stuart has put this one together with the professional computing audience in mind.

Stuart breaks the book into two sections (if you exclude the brief introduction to Ruby—more on that later).

In the first half, Stuart builds up simple computational machines, starting with parsers and finite automata and finishing with the development of a universal Turing machine. He explores the capabilities and limits of each machine and then investigates how to extend the machine to the next level.

Stuart has chosen to express the logic of the machines using Ruby rather than a formal language. He acknowledges that this approach poses some tradeoffs in clarity, but he thinks this is offset by the fact that the reader can actually execute and observe the behavior of the machines he describes. I applaud the attempt to invite the reader to experiment and explore, but I think that he might have made the concepts clearer by presenting them in proper notation as well as in code. This would have given a concise representation that could be compared to the working code. As it is, it can be difficult to separate the topical content from the Ruby code artifacts.

Stuart spends the remainder of the book exploring the capabilities and limitations of the universal Turing machine. Again, he starts with the language of computation, this time the lambda calculus. After producing a working implementation in Ruby, he shows that the lambda calculus is equivalent to a universal Turing machine, as are, in the end, several possible alternative computational models. Again, it would have been clearer to include the operations and explanation of the lambda calculus in traditional notation followed by the translation into Ruby.

Finally, in a chapter entitled “Impossible Programs,” Stuart confronts the truly difficult problems of modern computation. In a mere 30 pages he treats the identity of code and data, decidability, and the Halting Problem. Godel's Incompleteness Theorem gets mentioned, but there is no real discussion of its deep implications.

Stuart quotes from and provides a reference to Turing's original paper on computability. In at least half a dozen other places, he makes a passing comment about some other research or information that could have added depth to the discussion. In some of those places, he includes a link to Wikipedia (which I think is a fine place to learn more), but in others he just moves on.

Stuart has done a fine job presenting the content of this theory, but the presentation lacks a sense of the significance and wonder that I find in the idea that my laptop is, conceptually, no more powerful than a Turing machine. Nevertheless, *Understanding Computation* is still the only offering that I've seen aimed at computer professionals, and it will serve that audience well.

Conference Reports

NSDI '14: 11th USENIX Symposium on Networked Systems Design and Implementation

April 2–4, 2014, Seattle, WA

Summarized by: Chen Chen, Michael Coughlin, Rik Farrow, Seyed K. Fayazbakhsh, Pan Hu, Chien-Chun Hung, Sangeetha Abdu Jyothi, Rishi Kapoor, He Liu, Feng Lu, Oliver Michel, Muhammad Shahbaz, Doug Woos, Qiao Zhang

Opening Remarks

Summarized by Rik Farrow

Ratul Mahajan, Microsoft Research, opened with a comment about how it was nice that we got to see Seattle when it wasn't raining. The rain held off until after the workshop concluded. Ratul went on to say that there was a new record in the number of submissions, 223, with 38 papers accepted for presentation over the next three days. The workshop included a new session track, operational systems track, on Thursday morning. The workshop this year drew over 250 attendees. Program Committee members had to review more than 27 papers each; fortunately, said Ratul, no one remembered (or at least commented on) the promise that each member would have fewer than 27 papers to review.

Ion Stoica, UC Berkeley, thanked Ratul for making his experience as co-chairperson a pleasant one. Ion announced that the award for Best Paper went to Mihai Dobrescu and Katerina Argyraki for "Software Dataplan Verification." The Community Award was given to EunYoung Jeong et al. for "mTCP: A Highly Scalable User-Level TCP Stack for Multicore Systems."

Finally, the Test of Time awards, for papers published at NSDI at least 10 years earlier that have had a lasting impact on their field, went to two papers (chosen by Tom Anderson, Stefan Savage, and Robert Morris of the Test of Time committee): "Operating Systems Support for Planetary-Scale Network Services" (better known as PlanetLab today), by Andy Bavier et al., and "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," by Philip Levis et al.

Datacenter Networks

Summarized by Sangeetha Abdu Jyothi (abdujyo2@illinois.edu)

Circuit Switching under the Radar with REACToR

He Liu, Feng Lu, Alex Forencich, Rishi Kapoor, Malveeka Tewari, Geoffrey M. Voelker, George Papan, Alex C. Snoeren, and George Porter, University of California, San Diego

He Liu presented REACToR, a hybrid Top-of-Rack (ToR) prototype that combines optical circuit switching and electrical packet switching to enable high-throughput networks. Although a 100 Gbps fat-tree would be an ideal choice for improving the throughput of an existing 10 Gbps datacenter network, this is an expensive option. The author mentioned previous work that attempted to improve throughput using

optical circuit switching for large flows and drew attention to the fact that these techniques ignored the performance of short flows. In order to allow easy expansion of current 10 Gbps networks to 100 Gbps networks, the authors proposed the use of REACToR. REACToR combines the best of both worlds—it uses electrical switching, which allows buffering of packets for low bandwidth flows, and optical switching, which supports higher bandwidth for large flows.

In this model, the end hosts maintain a single queue for all packets belonging to low bandwidth flows destined for a packet-switched network and a per-destination queue for large flows to be sent on the circuit-switched network. REACToR sends Priority Flow Control (802.1Qbb) frames to pause and unpause flows at the end hosts. Since the link connecting user and REACToR is 100 Gbps, the optical bandwidth is limited to 90 Gbps in order to accommodate the sum of circuit-switched and packet-switched transmissions. Performance of the system was evaluated under various conditions. The TDMA scheduling is invisible to large TCP flows and guarantees fairness across flows irrespective of the schedule. The system also responds to demand changes in a fast and robust manner within 1.5 ms. The difference in performance between simulation and real world implementation was less than 1%. The system also performs well with a skewed workload (one flow constituting 50% of the traffic) and a very skewed workload (a single flow contributing 99% of traffic).

Changhoon Kim, the session chair, pointed out that benefits of the system depend on the speed and accuracy of traffic demand estimation. Liu responded that work on traffic estimation is ongoing. Currently, application-level information is used. Rik Farrow asked whether the experiment results were based on simulations only or involved real optical switches and the use of mirrors for the circuits. Liu replied that the results were based on a real implementation, and the Mordia switch does indeed use mirrors. Peter Hill asked about the diameter of the network. Liu responded that the testing was done on a small network with eight hosts. Liu also observed that scaling this system to accommodate thousands of hosts in a datacenter could be a challenge and requires further work.

Catch the Whole Lot in an Action: Rapid Precise Packet Loss Notification in Data Center

Peng Cheng, Fengyuan Ren, Ran Shu, and Chuang Lin, Tsinghua University

Peng Cheng presented cutting payload (CP), a mechanism that allows precise packet loss feedback. TCP faces several challenges—TCP incast, out-of-order packet arrival, etc.—and several mechanisms have been proposed to tackle each of these problems. But there exists no single solution that can mitigate all the problems associated with TCP. In order to improve the performance of TCP, three main challenges need to be addressed: (1) avoid TCP timeout caused by insufficient ACKs,

(2) distinguish packet loss and packet delays accurately, and (3) reduce packet loss detection time.

The authors address these challenges by cutting payload (CP). Switches use CP to cut off the payload of packets during congestion; only the headers are transmitted to the receivers. Upon receiving a payload-cut packet, the receiver parses the header and generates a PACK for the missing payload. The PACK is transmitted to the sender, which parses the message and triggers fast retransmission for the dropped payload. The authors showed that CP only increases resource usage by 2% and delays by 56 ns. The query completion time improved by 40% using this mechanism. Finally, Peng Cheng mentioned that CP is a TCP-complementary mechanism that is compatible with other versions of TCP used in datacenter networks.

Costin Raiciu (University Politehnica Bucharest) focused on the importance of PACK and asked about the need for a response to the sender. Peng replied that PACK gave the sender more information regarding lost payloads. Costin then wanted to know more about the implementation of buffers in the switch. Peng responded that the switches had byte-oriented buffers of size 128 kB. The next questioner asked about a scenario where a bottleneck link is shared by a large flow that uses CP and several small flows which do not—how would the fairness criteria be met in this situation? Peng acknowledged that such scenarios were not tested. Changhoon Kim wanted to know more about the difference between ECN and CP and was referred to the paper. Another attendee asked about the choice to make changes at end-hosts and not use feedback from the switches. Peng answered that replies from switches would require network-wide changes whereas CP requires modifications at end-hosts only. Costin Raiciu returned to make another point: He referred to a scenario where the switch buffers contain only packet headers with no goodput in the network. Peng pointed out that CP has high goodput as demonstrated by the results in the paper.

High Throughput Data Center Topology Design

Ankit Singla, P. Brighten Godfrey, and Alexandra Kolla, University of Illinois at Urbana—Champaign

Ankit Singla put forward a systematic approach for high throughput datacenter design. The presentation focused on the design of throughput-optimal network topologies and dealt with two main questions: (1) How close can we get to optimal network capacity? and (2) How can we handle heterogeneity? In order to compare the performance of networks, throughput is computed as the solution to a linear program whose objective is to maximize the minimum flow in the network. Ankit pointed out that this is a more accurate measure of throughput than bisection bandwidth. Next, he presented an upper bound for throughput in a homogeneous network that is inversely proportional to the average path length. He showed that the throughput of random graphs is very close to this upper bound.

Next, Ankit dealt with the design of heterogeneous networks with multiple types of links and switches. The switches in the

network are grouped into two clusters of high-degree switches and low-degree switches. The key challenges associated with the heterogeneous topology design for high throughput are (1) identification of the appropriate interconnection between the two category of switches and (2) determination of the best distribution of servers across these switches. Ankit showed that throughput improved initially when the number of cross-cluster links increases and then reached a plateau due to the upper bound imposed by the average path length. This allowed greater freedom in the cabling of heterogeneous networks. In addition, throughput was found to be optimal when servers were added in proportion to the port-count of the switches. Ankit pointed out that by using this technique of topology design, throughput performance of virtual layer 2 (VL2) topology could improve by 40%.

Tom Anderson (University of Washington) asked whether the workloads were realistic. Ankit responded that specific workloads cannot be representative for all datacenters. The traffic matrices used in the experiments were within two times the worst-case traffic matrix. As a follow up, Tom mentioned that optimizing the network for a particular traffic matrix is a dangerous trend and requested a clarification on the choice of the traffic matrix. Ankit replied that the workload is ideal for a generic high-capacity interconnect that can support a wide variety of applications. Hence, the results are applicable to any workload that is nearly uniform. Costin Raiciu inquired about the fraction of servers that contribute to the workload—the network is designed for 100% load, but the realistic loads could be 30%. Ankit replied that the network could be designed for the expected average load using the proposed mechanism, and the peaks could be tackled using hybrid approaches such as optical networks. Vyas Sekar (CMU) pointed out that the shortest path might not be optimal for traffic engineering. Ankit referred to results in the paper that show that the performance of packet level experiments using multipath-TCP is very close to the flow-level results obtained in simulations.

Debugging Complex Systems

Summarized by Rishi Kapoor (rk Kapoor@ucsd.edu)

Adtributor: Revenue Debugging in Advertising Systems

Ranjita Bhagwan, Rahul Kumar, Ramachandran Ramjee, George Varghese, Surjyakanta Mohapatra, Hemanth Manoharan, and Piyush Shah, Microsoft

Ramjee started by describing advertising (ad) systems as large distributed systems that are complex for two reasons: scale (millions of queries, users, publishers, ads) and the number of entities these systems interact with. The paper discusses revenue debugging: Why is revenue down at a given time anomalously, and how much does it cost?

Ramjee presented three interesting case studies/scenarios explaining the root cause for revenue anomaly. In the first scenario, a datacenter in Ireland had latency issues, which resulted in fewer ads being shown and consequent revenue decline. In the second scenario, revenue loss was caused by buckets using a new relevance algorithm. Finally, during the papal election, a lot of

people were searching for “pope results,” but there were very few advertisers who could show ads for these terms.

Ramjee said they focused on three aspects of these problems: a novel algorithm for root-cause analysis, attributes for derived measures (e.g., revenue per search), and they built a real-time time system for such root-cause diagnosis. For identifying the root cause of an anomaly, they picked a metric which deviated most from the expected value (JS-divergence). Throughout the talk, Ramjee gave examples of scenarios where derived measures such as cost per click (unlike individual metrics such as clicks or revenue) were able to detect anomalies by capturing correlations between the different metrics.

Ramjee also presented a demo of their tool: Adtributor. In their evaluation they show that their tool has an accuracy of more than 95% and saves troubleshooting time by 1+ hour per anomaly.

Evgeny Vinnick (Simon Fraser) asked whether they could expand this application to new markets and whether the code was available for experimentation. Ramjee replied that currently evaluation is done on four markets, but the system is running on more than 20 markets. Ramjee also said that the code is not currently open sourced, but they can consider that in the future. Someone asked whether the data collected by Adtributor could be used to create trending analysis: for example, could this tool be used to move an advertisement from x side to y side to earn more advertising revenue? Ramjee replied that there are lots of complementary problems, which are related but also distinct. They could use Adtributor but the question asked falls into a separate problem. The same person asked about network latency, and Ramjee replied that the network latency bubbles up to the top layer via the datacenter metric. Were there different datacenter metrics with a layer-wise approach? Ramjee replied to the smallest set question: If all your results are 100 elements long, but only one or two elements are the root cause, the smallest set is in regard to that. Ravi Bhoraskar asked whether they could apply this tool to any kind of troubleshooting: for example, could Azure use it? Ramjee replied that they could use it to detect failures and slowdowns in Azure or more generally. Ramjee mentioned the Bodik paper [Eurosys 2010] and added that these principles are fundamental, but he couldn't claim how easy it would be to port to other systems.

DECAF: Detecting and Characterizing Ad Fraud in Mobile Apps

Bin Liu, University of Southern California; Suman Nath, Microsoft Research; Ramesh Govindan, University of Southern California; Jie Liu, Microsoft Research

Bin Liu started by explaining how the ad ecosystem works. First, the app developer registers with an ad network, which then provides an ad plugin that the app developer incorporates into the app. The ad network selects an ad to be displayed to a user based on the user's location, interests, etc. When a user clicks on an ad, the ad network pays the app developer. Thus the app developer has an incentive to commit ad fraud by changing the way the ad

is displayed (e.g., placement layout, invisible impressions) so that the user accidentally clicks on it. Bin Liu showed visual screenshots of such ad frauds. The focus of their work is to build an automated system that can detect such frauds.

The main challenge with building such a system is that it would need to scale to thousands of visually complex apps (an almost infinite number of pages and tens of clickable elements) and accurately and quickly identify the fraud. Other technical challenges include the sliding screen problem and the fact that the z-index is not available to identify hidden ads. Their automated system, DECAF, analyzed 50k phone apps and 1150 tablet apps. The system takes as an input an app, uses an automated UI navigation (Monkey), and outputs whether the app may contain placement frauds.

Someone asked whether they assume that DECAF can correctly extract all UI actions. Bin replied that they don't make that assumption, and there are indeed some apps that they can't recognize, mostly because the Windows Automation Framework can't identify them. Evgeny Vinnick asked whether they would be able to detect fraud if publishers use ads from different providers, like Google and Microsoft. Bin replied that they wouldn't be able to detect fraud for other providers. The system they designed needs a sub-checker for each provider. If another provider has a similar policy, they can use the same sub-checker but otherwise they will need to extend it. Someone asked whether the application developer could bypass their system, using techniques that Monkey can't detect such as a captcha (e.g., typing in three numbers to go to the next screen). Bin argued that they can design a new sub-checker to avoid application bypass. Microsoft is also working on adding a smarter ad-control checker. Bin added that their work is focused on speed, and they get better benefits than other tools. Ravi Bhoraskar asked why they couldn't use the page class to determine structure—for example, two different post pages on Reddit would be in the same class. Bin argued that the page class can't be used to determine things dynamically. It is necessary to compare pages at runtime.

I Know What Your Packet Did Last Hop: Using Packet Histories to Troubleshoot Networks

Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, David Mazières, and Nick McKeown, Stanford University

Nikhil Handigol started his talk by giving an example of a simple network bug—one that required hours of manual network debugging to identify the root cause and fix the issue. He argued that the current state-of-the-art tools in network debugging, tools like ping and traceroute, are tedious to use and require deep understanding of the tools and the system. Moreover, these methods are Band-Aid solutions—that is, they don't guarantee that they will be helpful in solving the outage. Nikhil emphasized that these tools provide close to zero visibility on what is happening in the system. He further argued that complete visibility of every event in the network is needed, which is challenging because this amounts to a huge amount of data to collect, process, and store.

Nikhil claimed that they can achieve complete visibility using packet histories (i.e., the path taken by a packet, modifications it encountered, and the current switch state) and a platform the authors built called NetSight. These histories can be used to diagnose faults in the system—for example, dropped/lost packets. Nikhil mentioned that naive implementation would result in huge overhead (of total bandwidth), and storing packet information would result in a huge storage costs. Nikhil described how they implemented such a solution using diff-based history and MapReduce-style parallel computation.

The authors developed four applications: NDB, netwatch (a live network debugger), nprof (a hierarchical network profiler), and netshark (a network-wide path-aware packet logger). Their evaluation demonstrated low overhead and the feasibility of complete network visibility. Nikhil mentioned that code is available at the following link: <http://yuba.stanford.edu/netsight>.

Aaron Gember (University of Wisconsin) asked how they ensure that they don't lose the tags on packets. Nikhil said that it is necessary to ensure that these tags are immutable. This should be enforced at the controller or proxy layer. Rodrigo Fonseca (Brown University) asked about the modifications needed at the switches to generate the post cards, and whether the method is very similar to sFlow, which would make it very slow. Nikhil replied that their work is different from sFlow in the way they correlate the exact state present on the switch. Unlike sFlow, the proxy adds a few extra actions to generate a copy of the packet header in the data plane and not in the control plane. Rodrigo followed up to ask whether the operations mentioned by Nikhil are standard OpenFlow actions. Nikhil mentioned that their prototype works with unmodified prototype OpenFlow switches; currently, OpenFlow doesn't have the ability to truncate the headers in hardware, making it highly inefficient as they copy entire packet headers. Rodrigo then asked whether the numbers mentioned in the talk, 31% and 7% compression, were with the prototype. Nikhil mentioned that these numbers are not for the prototype.

Timothy Wood (George Washington University) asked how they use these packet histories. Nikhil showed a few packet filter examples to detect reachability, isolation, and forwarding loops in networks. Timothy followed up by asking whether it is clear what these queries should look like. Nikhil mentioned that packet history appears like a string and that you apply a filter to this string using regular expression.

Libra: Divide and Conquer to Verify Forwarding Tables in Huge Networks

Hongyi Zeng, Stanford University; Shidong Zhang and Fei Ye, Google; Vimalkumar Jeyakumar, Stanford University; Mickey Ju and Junda Liu, Google; Nick McKeown, Stanford University; Amin Vahdat, Google and University of California, San Diego

Hongyi Zeng started his talk with a graph showing three common problems in Google datacenters. These problems are missing forwarding entry, forwarding loops, and black holes. Each

of these trouble tickets is very expensive because it takes a very long time to debug them. Hongyi also mentioned that diagnosing these problems is difficult because it involves complex interactions between multiple protocols on the same switch and complex interactions between states on different switches. This arises from uncoordinated writes in the system. Hongyi mentioned that the static data plane verification does not work for datacenters for two reasons. First, in a large datacenter network the forwarding state is constantly changing, which makes it hard to take an accurate snapshot of the state for static analysis. Second, static analysis tools do not scale for large datacenters.

Hongyi said that they created a tool (Libra) that is fast and scalable and that can quickly detect loops, black holes, and other failures in large networks. First, Libra captures stable and consistent snapshots across large network deployments. Second, in contrast to prior tools that deal with arbitrarily structured forwarding tables, they substantially improve scalability by assuming packet forwarding based on longest prefix matching. The authors focused on the problem of obtaining a stable snapshot across thousands of switches. The gist of their solution is that they only consider the stable and consistent snapshots (discarding shady areas) and thus avoid false positives. Libra uses a divide and conquer algorithm that can be implemented using MapReduce to overcome the large scale of datacenters.

The data set is open sourced at <http://eastzone.github.io/libra/>.

Juan Francisco asked Hongyi to compare the previous paper's complete dynamic approach with their static analysis. Hongyi mentioned that these approaches solve two different sides of the problem. The static analysis is useful for checking functional problems that can be solved using a forwarding table. Performance problems are another class of problem that can be tackled only by using dynamic checking. Other sets of problems where incoming packets modify network state can only be solved in a dynamic state. Hongyi concluded that these approaches apply to different problems. Someone else asked about accounting for NTP inaccuracies, and whether there are any hardware switches that implement Lamport clocks (i.e., precisely the problem that solves the snapshot problem). Hongyi said that this problem can be solved if these systems have global clocks. NTP is more popular and deployed on the switches, and by adjusting the epsilon value they can achieve the same effect. Someone else asked whether they could gather events from the black box switches or required SDN switches. Hongyi answered that Google uses SDN-based switches. With traditional switches, it is hard to dump the state of the network, and you can't subscribe to the network. The final question was whether they could use the tool to do testing on controller software. Hongyi replied that it could be used as an independent checker to check the rules.

Verification and Testing

Summarized by Seyed K. Fayazbakhsh (seyed@cmu.edu)

Software Dataplane Verification

Mihai Dobrescu and Katerina Argyraki, École Polytechnique Fédérale de Lausanne

Awarded Best Paper!

Katerina Argyraki explained that software data planes are composed of packet processing elements. This is expected to make it easier to reprogram network functionalities. But in reality, bugs are present. The authors wanted to be able to take the binary along with the properties of interest, have a tool to verify the properties, and do so by adopting symbolic execution. Essentially, a section of code is represented using a tree that branches where the actual code branches. Exploring a path means examining the properties along the path. The problem with a naive application of symbolic execution is path explosion. The use of composition enables exploration of a subtree only once using domain-specific knowledge. The opportunity here is to utilize the pipeline structure of the data plane, which means usually there is no shared mutable state across data-plane elements for a packet.

The problem is that loops make reasoning about data-plane elements in isolation difficult (e.g., going over the IP options of a packet). The solution to this challenge is to share a small amount of state information between loop iterations. Another challenge is that there are many possible values that data structures may take. The solution here is to make explicit APIs by the programmer to enable data-access decomposition. A hash table and longest-prefix-match table are the proof-of-concept data structures that the authors implemented. The downside is that they cannot use dynamic memory allocation if they want to be able to verify the data structure. They have proved bounded instruction and crash freedom for Click data planes.

Costin Raiciu asked about the need to change the Click code and binary to make it work. Katerina answered that they did not need to change Click itself, but they had to change the loop structure part of metadata, which took a few lines of code. They also needed to extract data structures (e.g., for a NAT) using APIs. Costin then asked about whether the authors had any plans to build on this work, since what was done seemed practically limited. Katerina answered that they did support elements such as IP routing elements and NAT boxes. These are simple, but existing tools cannot capture them. Someone from Princeton asked what kind of interface would be needed to add to data structures. Katerina answered that key-value store interfaces were required to read or write or expire a value. Any well-defined interface would do. Someone asked about how easy it would be to determine verification requirements. Katerina said that the vision is that admins should not be doing this. Rather, the programmers should follow the authors' guidelines.

NetCheck: Network Diagnoses from Blackbox Traces

Yanyan Zhuang, Polytechnic Institute of New York University and University of British Columbia; Eleni Gessiou, Polytechnic Institute of New York University; Steven Portzer, University of Washington; Fraida Fund and Monzur Muhammad, Polytechnic Institute of New York University; Ivan Beschastnikh, University of British Columbia; Justin Cappos, Polytechnic Institute of New York University

Justin Cappos began by saying that applications like Skype are complicated to troubleshoot. Candidate solutions include using tools like ping or Wireshark, or trying to model apps and network elements to try to find bugs. These solutions are not completely practical. The insight in NetCheck is that humans make mistakes but not perfectly random mistakes. So we use tools like ktrace or strace for capturing system traces and then process and order these logs. This lets us diagnose problems. A trace is a locally ordered series of system calls. Each call has arguments and return values. Getting the exact global timestamps across traces captured in different locations is very difficult; we just need an approximation, though, as long as the extracted ordering is equivalent to the ground truth. NetCheck reconstructs what actually happened in the network using return values to infer the correct orders.

The network model is a simulated invocation of a system call. The runtime of the ordering algorithm is a linear function of the trace size. The fault classifier component of NetCheck decides what to output as the relevant information. We can configure what level of detail we want to receive. For evaluation of NetCheck, the authors reproduced reported bugs from bug trackers and found more than 95% of the bugs. NetCheck is extremely fast for logs larger than 1 GB.

Minlan Yu (University of Southern California) asked whether NetCheck needed the complete trace. Cappos answered no, NetCheck can deal with missing information. Wyatt Lloyd (Facebook) asked whether NetCheck works only for client-server applications and wondered about multi-party settings. Cappos answered that NetCheck could handle such scenarios. Someone from LinkedIn asked about performance issues, not just connectivity. Cappos answered that performance bugs were not targeted in this work. Someone asked how middleboxes come into the picture. Cappos responded that they can detect the problems in the middle as long as the applications are written to work with middleboxes.

Exalt: Empowering Researchers to Evaluate Large-Scale Storage Systems

Yang Wang, Manos Kapritsos, Lara Schmidt, Lorenzo Alvisi, and Mike Dahlin, The University of Texas at Austin

Yang Wang presented Exalt, a library that gives back to researchers the ability to test the scalability of today's large storage systems. Researchers usually do not have access to enough resources to test storage systems. The problem gets worse because sometimes the scale of the required test resources grows superlinearly with the system size. Quite often the I/O is the bottleneck, so we cannot simply add more resources. We

need to abstract away data to scale the test method. The authors use data compression to solve this problem. The requirements for compression here include CPU efficiency, high compression ratio, and losslessness. Furthermore, the compression algorithm should be able to work with mixed data and metadata.

Our algorithm uses Tardis compression. Tardis allows data to be identified and efficiently compressed even at low-level storage layers that are not aware of the semantics and formatting used by higher levels of the system. This compression enables a high degree of node co-location, which makes it possible to run large-scale experiments on as few as a hundred machines. The authors used Exalt to identify several performance problems in HDFS and HBase.

Minlan Yu noted that the authors considered single nodes; what about the case of multiple nodes communicating with each other, and how efficient would the system be? Yang Wang answered that the work could not capture many nodes cooperating with each other. Wyatt Lloyd asked whether there were scalability restrictions with very large systems. Yang Wang answered that the authors could not guarantee full coverage, which would be the case with other tools, too. Someone from LinkedIn asked whether the authors considered a bottleneck caused by the hardware itself, such as network cards, when a lot of servers were emulated. Yang Wang answered that the implementation currently would not support that but that the authors were planning to consider a device-modeling approach to incorporate, for example, models of disks.

Security and Privacy

Summarized by He Liu (h8liu@cs.ucsd.edu)

ipShield: A Framework for Enforcing Context-Aware Privacy

Supriyo Chakraborty, Chenguang Shen, Kasturi Rangan Raghavan, Yasser Shoukry, Matt Millar, and Mani Srivastava, University of California, Los Angeles

Supriyo Chakraborty pointed out that some cell phone applications that provide utility to users read sensor data (e.g., to monitor user fitness); these applications can also infer sensitive information from the sensor readings (like user password) and hence violate user privacy. ipShield is an inference firewall that protects users from such attacks. Different from previous protection systems that statically restrict sensor data accessing, ipShield recommends privacy sensor accessing rules based on an inference whitelist/blacklist of higher-level privacy abstractions.

To use ipShield, a user first specifies an inference whitelist and a blacklist with priorities assigned to each inference. ipShield then will build the sensor-accessing rules based on an accuracy table. The table lists the inference accuracies that an application can achieve with different combinations of sensor readings. The recommended rules that ipShield outputs tend to maximize the accuracy of the whitelisted inferences and minimize the accuracy of the blacklisted inferences. Based on

the recommendation, users can manually override the rules and create their own fine-grained policies.

Supriyo then talked about the complexity of implementing ipShield on Android systems, and showed the latency introduced and memory overhead of running ipShield. Source code of ipShield can be downloaded at <http://tinyurl.com/ipshieldgit>.

Jaeyeon Jung (Microsoft Research) asked how ipShield captures the correlations of different inferences, since information on one inference (such as location) might disclose information on another (such as activity). Supriyo said that in this paper, ipShield does not model it, but the team was in the process of integrating these correlations. Another person asked how ipShield tracks indirect data flow. For example, an application that has access to the GPS sensor can transfer the sensor data to another application via the storage card. Supriyo answered that as an extension to ipShield, they were trying to perform static analysis on the applications to track such data flows. Seungyeop Han (University of Washington) asked how ipShield could cover all possible inference types. Supriyo answered that crowd-sourcing could play a role here, but that is hard in general because important inference types in the future could be currently undefined.

Building Web Applications on Top of Encrypted Data Using Mylar

Raluca Ada Popa, MIT/CSAIL; Emily Stark, Meteor, Inc.; Steven Valdez, Jonas Helfer, Nikolai Zeldovich, and Hari Balakrishnan, MIT/CSAIL

Raluca Ada Popa explained that Web applications store sensitive data on servers, but it is challenging to protect the data from attackers who could have full server access. To handle this threat, Raluca presented Mylar, a framework that stores data encrypted on untrusted servers, where the data is only decrypted in a user's browser and presented via verified Web applications. On the untrusted server, Mylar stores user data with different encryption keys, yet the framework allows data sharing among multiple authorized users and data searching across multiple keys.

Raluca showed how Mylar works using a chat room example. First, Mylar generates the Web pages on the client side with certified Web application code (rather than on the server side). Since the server only acts as remote storage for signed code and encrypted data, it cannot tamper with the Web page. Second, Mylar manages shared data with a principal graph that is enforced by encryption key chains of certified keys. Users encrypt their data with different keys based on the principal graph. Third, Mylar introduces a new encryption scheme that enables multi-key search. If a user knows two keys, it can compute a "key delta" of the two keys and send it to the server, where the server can morph the encrypted data from one key to another so that searching is possible in encrypted form. With this setup, Mylar protects a user's data from other users and also fully compromised servers.

Implemented in 9000 lines of JavaScript and C++, the developer's effort to use Mylar is around 36 lines of code change on average, and the performance overhead introduced is also modest for Web applications. During the presentation, the PowerPoint slide failed to show the performance overhead figure, but Raluca managed to describe the results to the audience in her own words without the figure. The implementation can be downloaded from <http://css.csail.mit.edu/mylar/>.

Regarding the chat room example, one attendee asked whether two chat rooms created by the same user would share the same key. Raluca responded that two different chat rooms have different keys, and this depends on how the user/Web application chooses to handle it. Another attendee asked why some service providers that make profits from looking at user data would want to use Mylar. Raluca said that although there are clouds that make profits from user data, there are also clouds that rent resources to users. Mylar fits the second type; enabling the first type over encrypted user data is interesting future work. Finally, Sophia Xiao Wang (University of Washington) asked about Mylar's trust base. Raluca responded that those who use Mylar trust their own machines and Web browsers, not to mention the client-side code of the Web application developer.

PHY Covert Channels: Can You See the Idles?

Ki Suh Lee, Han Wang, and Hakim Weatherspoon, Cornell University

In this talk, Ki Suh Lee presented a covert timing channel called Chupja ("spies" in Korean) that works at the physical layer, with high bandwidth (100s Kbps), low bit error rate (less than 10%), and very hard to detect by upper-layer software. The threat comes from a passive adversary in the middle, equipped with commodity servers and NICs, trying to detect and monitor the communication between a sender and a receiver who want to exchange secrets. Both the sender and the receiver have full control of their own physical layers.

The design of Chupja is simple: It encodes information into a packet stream of the same packet length and inter-packet gaps (IPG) by varying the length of the gaps, where a slightly longer gap encodes a 1 bit and a slightly shorter one encodes a 0. Chupja is implemented as 50 lines of C code on top of SoNIC [NSDI '13], a software-defined network interface card that has full control of the physical layer.

Through evaluations, Lee showed that, throughput-wise, Chupja can achieve 81 Kbps of covert throughput over 1 Gbps overt throughput. Robustness-wise, the bit error rate increases with the number of hops on the route, but with a larger distance on the two coding points (the time difference between a 0 gap and 1 gap), Chupja can achieve a bit error rate of less than 10% even sharing the link with external traffic and, at the same time, still remain undetectable to software that runs on commodity servers that only have kernel timestamps for packet timing. The implementation can be downloaded from <http://sonic.cs.cornell.edu>.

Dongsu Han (KAIST) asked two questions: For the covert channel to remain undetected, the overt channel has to look innocent, but how is that done? Lee responded that the overt channel can simply transmit upper-layer application data as a cover. The second question was does it work with radar? Lee responded that Chupja's timing encoding scheme is general and can be applied to many communication channels.

cTPM: A Cloud TPM for Cross-Device Trusted Applications

Chen Chen, Carnegie Mellon University; Himanshu Raj, Stefan Saroiu, and Alec Wolman, Microsoft Research

Chen Chen stated that mobile devices have started to use trusted hardware, such as the Trusted Platform Module (TPM). However, protecting data with TPM across multiple devices is hard, because TPM-created keys are bound to a single TPM chip. cTPM (short for cloud-TPM) provides a solution for this. It embeds an additional root key pre-shared with the cloud. This enables seamless sharing of TPM-protected data with the cloud's assistance. It also provides additional benefits, such as a fast and large remote NVRAM storage and a trusted real-time clock.

One alternative to cTPM is to leverage secure execution mode (SEM), which is TPM's extensibility mechanism. However, SEM suffers from performance and engineering overhead, and lack of support on mobile devices. Instead, cTPM trusts the cloud, and provisions a unique random seed value pre-shared with the cloud. Based on this shared seed value, cTPM deterministically generates a "cloud root key" (CRK) and a "cloud communication key" (CCK). With these keys, the cloud can securely distribute shared keys to multiple cTPMs. The shared key is encrypted by the CRKs, and the communication channel is protected by the CCKs.

The cloud can also offer remote NVRAM storage. To handle disconnections and network latency, the cTPM also maintains a cache of the remote NVRAM storage. Each cache entry in the cTPM has a time-to-live field that dictates when the entry becomes stale. The untrusted OS and applications are responsible for re-syncing the cache; the synchronization protocol is protected with the CCK shared between cTPM and the cloud. Finally, a trusted clock can be implemented simply as a special NVRAM entry updated by the cloud. To offer this functionality, the cTPM proposes three new commands over the TPM 2.0 specification, and the protocol is verified with ProVerif.

Chen's implementation of cTPM is 12x faster than a typical hardware TPM for creating 2048-bit RSA keys; this is because software-based entropy source and crypto computation (on the cloud) are much faster than that of a TPM chip. Chen's team reimplemented Pasture [OSDI '12] and TrInc [NSDI '09] on top of cTPM as application demos.

One attendee mentioned that by storing the key on the cloud, the cTPM now provides a security property inherently different from TPM. A cloud compromise would inherently be a cTPM compromise. Chen responded that cTPM works as an addition

on top of the current TPM design. While a cloud compromise would affect the root key pre-shared with the cloud (including the CRK and CCK), the unshared TPM-only root keys (e.g., the storage and endorsement keys) would remain safe. Also, making the cloud trusted is an active research area; emerging technology like Intel SGX could help with this issue.

Posters

First group summarized by Doug Woos (dwoos@cs.washington.edu)

Garbage Collection and Heap Growth Heuristics for Mobile Systems

Gabriel Arellano, Eduardo Dragone, Md. Jahid, Rogelio Ochoa, David Pruitt, Adrian Veliz, and Eric Freudenthal, University of Texas at El Paso

Garbage collection time represents a large problem for the performance and responsiveness of Android applications. Garbage collection is triggered in response to heap growth. The authors propose an alternative GC heuristic which allows the heap to grow and limits collection to at most once in a given time period (which they set to two seconds for evaluation). Their results were preliminary, but they found that with their policy, applications experienced no prolonged pauses and only modest additional heap usage.

GENI: Global Environment for Network Innovation

Vicraj Thomas, Niky Riga, and Sarah Edwards, BBN Technologies

GENI is a global-scale research testbed for networking and distributed systems projects. Similar to PlanetLab or VICCI, GENI allows researchers to provision virtual machines, OpenFlow-enabled switches, and WiMax base stations at many sites at US universities. Researchers can configure, via a simple drag-and-drop UI, multiple independent VLANs. GENI is now available for research use and is currently being used for multiple cloud-computing projects.

Online Censorship Resistance with *freedom.js*

Will Scott, Raymond Cheng, Arvind Krishnamurthy, and Thomas Anderson, University of Washington

The authors present *freedom.js*, a system for peer-to-peer communication in the browser. *freedom.js* is implemented in JavaScript and enables cross-platform communication. The authors have built several applications, including a file-sharing application, a VPN in which a user's traffic is routed through his or her friends, and *activist.js*, a system for avoiding censorship by enabling peer-to-peer access to blocked Web sites.

User Scripting on Android Using *BladeDroid*

Ravi Boraskar, University of Washington; Dominic Langenegger, ETH Zurich; Pingyang He and Michael D. Ernst, University of Washington

User scripting, as enabled by Greasemonkey and other site-modifying browser extensions, has become an important part of the way users interact with the Web. *BladeDroid* uses byte-code rewriting and dynamic class loading to bring the same functionality to Android applications, allowing users to modify application behavior without the assistance or permission of the application developer. The authors have written several extensions, including an ad blocker and an input automation system.

A Platform for At-Scale Wideband UHF MU-MIMO Systems

Ryan Guerra, Narendra Anand, and Edward W. Knightly, Rice University

The authors present a platform for research into wideband networking. Their card array consists of four Wideband UHF Radio daughter-Cards (WURCs), as well as four standard WiFi antennas; each radio is programmable using the WARP Software-Defined Radio system. To demonstrate the system, the authors implemented an application that measures and displays both UHF and WiFi channel capacity.

Enabling Multi-Layer Provisioning and Optimization for Core Transport Networks with Unified Packet-Optical Control Plane

Abhinava Sadasivarao, Infinera Corporation; Henrique Rodrigues, University of California, San Diego; Sharfuddin Syed, Chris Liou, and Sivaram Balakrishnan, Infinera Corporation; Andrew Lake, Eric Poyoul, Chin Guok, and Inder Monga, Energy Sciences Network; Tajana Rosing, University of California, San Diego

The authors present a system for provisioning and optimizing network paths in Tier-1 networks that include both packet-switching and optical components. The system is implemented on top of the Floodlight OpenFlow controller and allows the use of unmodified OpenFlow for data path programming. Rather than having to consider optical transport separately, the system maps practical transport concepts to OpenFlow flow abstractions. They have implemented the system on actual packet-based and optical routers, and demonstrate it with a bandwidth-policy-based optimization system.

Ib-KV: Using Infiniband Effectively for Key-Value Services

Anuj Kalia and David G. Andersen, Carnegie Mellon University; Michael Kaminsky, Intel Labs

The authors implement a key-value store on top of Infiniband's RDMA primitives, and demonstrate that by making two unconventional design choices they can achieve significantly better latency and throughput than similar systems (FaRM and Pilaf) on read-intensive workloads. First, Ib-KV client requests begin with a write operation rather than a read; clients write the request directly into server memory. Second, the server uses RDMA's built-in messaging system for its response, replying with a SEND message over an unreliable datagram transport. The primary benefit of these decisions is to avoid multiple round trips when making a request.

Second group of posters summarized by Chen Chen (chen.chen@inf.ethz.ch)

WiSense: A Client-Based Framework for Wireless Diagnosis

Ashish Patro, Prakhya Panwaria, and Suman Banerjee, University of Wisconsin—Madison

WiFi technology has been extensively deployed in homes and enterprises. Today, however, WiFi networks suffer from several performance issues, such as RF coverage, WiFi link/non-WiFi interference, traffic hotspots, channel contentions, etc. For users to better identify location-specific WiFi problems, the authors built an Android-based platform called WiSense, which

can monitor WiFi network status without access points. Current WiSense implementation includes a NEXUS 7 device and an external USB card to collect fine-grained RF statistics. WiSense supports spectrum energy level monitoring, non-WiFi activity, per-channel airtime utilization, neighboring WiFi networks, aggregate per-link statistics, and active network measurements. The authors have demonstrated these features during demo sessions.

Pruning Masstree

Huanchen Zhang and David G. Andersen, Carnegie Mellon University; Michael Kaminsky, Intel Labs

Key-value stores are critical building blocks behind many cloud and network services like Facebook. The authors focus on building a space-efficient, high-performance key-value store that also enables range queries. The underlying data structure is called Masstree, which is a concatenation of layers of B+ trees that conceptually form a trie. The authors further designed pareto-optimal improvements to increase performance and reduce memory consumption, including designing more efficient memory allocation and garbage collection to save memory consumption, and serializing the B+ tree into a sorted array with binary indexing for higher performance. Preliminary results show that key-value stores could achieve 1.2 million items/sec with only around 200 MB memory consumed, and the designed improvements help increase the performance for range queries up to 3x.

Making the Live Network the Honeygot

Michael Coughlin, Oliver Michel, and Eric Keller, University of Colorado, Boulder; Adam J. Aviv, United States Naval Academy

Today's dedicated honeypots usually have different network topology, different applications, and different data than production networks. The differences not only leave the honeypot distinguishable from the production networks, but also make it difficult for network administrators to monitor the effects of attacks on production networks. In this work, the authors propose to combine live migration technique and the SDN to use the production networks as honeypots while isolating it from the attackers. According to their method, each time an attack is identified, the victim machine will be cloned to isolate the attacker from the production networks. The attack traffic will also be isolated from the production network by SDN. Finally, fake data are provisioned to replace actual data on the original machine to protect sensitive data. The authors have partially implemented the design based on KVM offline migration and SDN traffic dissection on top of Floodlight.

Towards an Open Middlebox Platform for Modular Function Composition

Shinae Woo, Korea Advanced Institute of Science and Technology (KAIST); Keon Jang, Microsoft Research; Dongsu Han and KyoungSoo Park, Korea Advanced Institute of Science and Technology (KAIST)

As the features of middleboxes continue to diversify, software-based middleboxes, which help consolidate multiple functionalities into a single box, have become increasingly popular over hardware-based middleboxes. However, the authors have

identified three challenges for implementing software-based middleboxes. First, existing network stacks lack support for exacting flow-level information. Second, existing software stacks lack support for diverse transport-layer or application-layer events. Third, existing platforms do not provide programmable pipelines to process packets. In this project, the authors aim to provide a software-based middlebox development platform with three key building blocks—flow management, user-defined events, and module composition—to effectively address key issues in middlebox development.

Erwin: A Low-Latency Network Monitoring Platform

Jeff Rasley, Brown University; Brent Stephens, Rice University; Colin Dixon, Eric Rozner, Wes Felter, Kanak Agarwal, and John Carter, IBM Research—Austin; Rodrigo Fonseca, Brown University

In software-defined networks (SDN), the state-of-the-art network measurement system (sFlow) takes hundreds of milliseconds to collect a view of network conditions, which is much longer than installing a new route. The authors repurpose the port mirroring features provided by switches to efficiently produce traffic samples by oversubscribing the mirror ports. The mirror ports of different switches are connected to a collector, which is responsible for determining input/output ports for the packets, estimate the flow rates based on TCP sequence numbers, and answer queries about network status. The resulting measurement latency is demonstrated to be 250 μ s–6.5 ms compared to current 100 ms–5 sec latency.

WiFi Mobility without Fast Handover

Andrei Croitoru, Costin Raiciu, and Dragos Niculescu, University Politehnica of Bucharest

WiFi networks are mostly static networks today, and mobile devices moving across different WiFi networks suffer from WiFi handover. The authors focus on reducing the handover duration for WiFi networks on mobile devices. The key idea is to leverage multi-path TCP to simultaneously connect to all available WiFi access points to avoid WiFi handover. The authors also demonstrate by experiments that the TCP congestion-control algorithm could well handle the interference caused by using a single channel and still provide high throughput. The authors also discuss ways to support multiple channels on mobile devices: using multiple NICs and using a channel switch to emulate multiple NICs.

Efficient Deployment of Network Management Policy Using Distributed Database Abstraction

Kamran Ali Akhtar, National University of Sciences and Technology, Pakistan; Muhammad Shahbaz, Georgia Institute of Technology; Saad Qaisar, National University of Sciences and Technology, Pakistan

The basic observation made by the authors is that the flow tables for controllers in software-defined networking (SDN) can be abstracted as relational databases. First, the controller would support Create, Read, Update, and Delete (CRUD) operations on the flow tables. Second, the flow tables on controllers must maintain Atomicity, Consistency, Isolation, and Durability

(ACID) properties. As a result, the authors propose to leverage existing Distributed Database Management System (DDBMS) schemes to manage flow tables for SDN and SQL as the interface to manipulate the flow rules. DDBMS could further provide features, including transaction validation, check-pointing, and deadlock mitigations, which are not present on current SDNs. The authors have used Mininet to build an implementation to validate the design based on POX and OpenFlow.

Toward a Precision Network Scripting with a User-Programmable Dataplane

Yohei Kuga and Takeshi Matsuya, Keio University; Hiroaki Hazeyama, NARA Institute of Science and Technology (NAIST); Kenjiro Cho, IJ Research Laboratory; Rodney Van Meter and Osamu Nakamura, Keio University

Today, network processing is much more difficult than file processing on an OS for applications for network testing and diagnosis. Therefore, the authors built a network scripting framework called EtherPIPE, which provisions a character device interface to bridge the gap between user-mode UNIX command tools and the underlying network devices. Moreover, to provide highly precise timestamps for the packets sent and received, the authors further implement a NIC using NetFPGA to assign highly precise timestamps for packets for network diagnosis and to transmit packets at highly precise time points. The authors have demonstrated that the frameworks could support network programming with shell script tools and achieve time precision in microseconds.

Operational Systems Track

Summarized by Michael Coughlin (michael.coughlin@colorado.edu)

Network Virtualization in Multi-Tenant Datacenters

Teemu Koponen, Keith Amidon, Peter Balland, Martín Casado, Anupam Chanda, Bryan Fulton, Igor Ganichev, Jesse Gross, Natasha Gude, Paul Ingram, Ethan Jackson, Andrew Lambeth, Romain Lenglet, Shih-Hao Li, Amar Padmanabhan, Justin Pettit, Ben Pfaff, and Rajiv Ramanathan, VMware; Scott Shenker, International Computer Science Institute and the University of California, Berkeley; Alan Shieh, Jeremy Stribling, Pankaj Thakkar, Dan Wendlandt, Alexander Yip, and Ronghua Zhang, VMware

Teemu Koponen began his presentation by explaining that the presented paper represents five years of work by various researchers, especially those named as authors. He continued by discussing whether the issue of network virtualization is, in fact, already a solved problem by various technologies such as VLANs, MPLS, NATs, VRF, OpenFlow, etc. He argues that each of these technologies is only a point solution for a specific aspect of the network but does not address the network as a whole. Koponen then addressed datacenters directly, arguing that a network virtualization layer is hard to achieve, as tenant workloads are highly coupled to the infrastructure, and that a virtualization layer similar to what is provided to VMs by a hypervisor is needed for the network layer.

Koponen then introduced NVP (Network Virtualization Platform): The function of this network hypervisor system is to present a packet abstraction to VMs such that the network view seen by the VMs appears to be a physical network, which allows

for applications to be run unmodified, and a control abstraction, which should allow for an ability to control the flow table pipeline on network hardware. NVP provides these abstractions, which allow for a tenant to create a logical data path, which, in turn, allows for the creation of any kind of logical topology. This system is implemented by using the virtual switches inside of standard hypervisors and connecting them using IP tunnels, with the switches being controlled from a central cluster.

Because this paper presents the product of five years of work, Koponen offered several lessons learned while operating NVP. First, some assumptions about logical networks cannot be made, because more complex workloads implicitly require more complex topologies that must be supported so that workloads are not modified; fortunately, this was addressed by the initial design decision to support arbitrary topologies. Second, OpenFlow proved to be insufficient for pushing state: Connections to OpenFlow controllers proved to be unreliable, which would lead to switches being left in an undefined state when a connection ended before completion. Third, OpenFlow is difficult to scale: Small operations still entail a large number of flow entries and may be redundant, and OpenFlow is highly coupled to switches. To address these last two issues, the authors are investigating a replacement for OpenFlow for the virtualization layer, but not necessarily the elimination of OpenFlow completely. Koponen concluded his presentation by restating the guiding principle of the project, which was to not modify workloads.

The first questioner asked whether the system obviates the virtual appliances that are built by networking companies and deployed in datacenters. Koponen explained that some appliances cannot be implemented in a distributed manner but can still be supported. He continued by stating that there will be pressure to implement these devices in a distributed manner, and he highlighted several applications built by VMware that illustrate this. Nodir Kodirov (University of British Columbia) asked if there was any experience with using the out-of-band network information debugging. Koponen replied that a large number of bits could be used in the encapsulation header that can and are used to debug workloads without affecting them. Marcin Kowalski (Amazon EC2) asked whether the researchers had looked at the performance implications of the software switches, in respect to 10G or 40G networks. Koponen replied that these implications had been investigated in the paper, including encapsulation optimization on x86 and Open vSwitch flow caching. Rick Schlichting (AT&T) asked whether there are any implications or lessons for virtualization in wide area networks. Koponen replied that wide area networks have not been investigated, but many deployments can span multiple datacenters, although the virtualization of wide area networks was not the goal of these deployments.

Operational Experiences with Disk Imaging in a Multi-Tenant Datacenters

Kevin Atkinson, Gary Wong, and Robert Ricci, University of Utah

Robert Ricci began by discussing disk images as one of the key tools for restoring a clean state to datacenter machines, and introduced his talk as a study of the disk imaging system of the Emulab datacenter, with an eye toward future research. He continued by explaining that the disk imaging system is an important consideration when provisioning a datacenter because of the need for large amounts of storage and bandwidth. He then asked three questions that he later answered in his presentation: What does the working set look like; what do the images themselves look like; and what are the key factors in preloading? The data set that was analyzed was collected over four years, consisting of 714 unique images from 1300 users and roughly 600 physical machines. Ricci noted that Emulab is not representative of all datacenters, but it provides an available data set.

Ricci then presented statistics gathered from the data set, starting with the number of requests for facility-provided images vs. user-defined images; this was a roughly even split, with 55% of requests being for facility images. Further analysis revealed that most users do not mix-and-match user and facility images and that heavy users tend to use user-defined images. Ricci continued by discussing the popularity of facility and user images. Analysis found that there was a small number of popular facility images, with the other facility images seldom used, whereas the user images were used more evenly by users, requiring a larger number of images to satisfy the same number of load requests. He then presented a scalability analysis based on the image usage trends: as the facility scales, all of the facility images will be used, because there is a limited number of facility images, but the number of user images will continue to grow.

Ricci described image content, noting that Emulab uses block-level similarity for users to create custom images, where users take a facility image and customize it, and then a diff can be taken from the base to store the image. Analysis of user images found that most were more than 60% similar to base images, which makes differential loading more useful. Ricci proceeded to discuss pre-loading of disk images onto disks. The first important consideration was the working set ratio (free pool of idle disks to number of images that can be pre-loaded), which allows for higher pre-loading effectiveness when this ratio is high. The second consideration was the rate of pre-loading an image; Ricci stated that pre-loading is more effective when investment is made in fast and scalable disk imaging. Ricci concluded by stating that a large number of images can be stored in slower storage, with a cache of popular images, and that facility images and user images are used differently and should be treated differently. All of the data and scripts used to write the paper are available at <http://aptlab.net/p/tbres/nsdi14>.

Vyas Sekar (CMU) asked whether Emulab has considered advertising images, in relation to how videos are advertised by

sites like YouTube, in order to tweak popularity toward pre-loaded images. Ricci replied that Emulab provides a default image, but it is of poor quality and is not used by many users. However, this image is not aggressively advertised. Peter Hill (Microsoft Cloud) asked whether many images could be loaded onto servers to make use of extra space and reduce the number of active machines. Ricci replied that Emulab machines are pre-loaded with two images for these reasons, as many tenants will reside on the same disk. Katerina Argyraki (EPFL, and session chair) asked whether the designers would change the interface for user images if the system were to be re-designed. Ricci replied that a new design would keep the current interface, as other packaging options were made available, but these methods were not used by users.

VPN Gate: A Volunteer-Organized Public VPN Relay System with Blocking Resistance for Bypassing Government Censorship Firewalls

Daiyuu Nobori and Yasushi Shinjo, University of Tsukuba

Daiyuu Nobori began his presentation by introducing censorship firewalls, which are intended to censor access to the Internet, of which the Great Firewall of China (GFW) is a well-known example. A common method of circumventing these technologies is to use relay servers, but censorship authorities can easily block access to these relay servers, and resistance to blocking is difficult. VPN Gate's approach is to use thousands of volunteers and distribute the list to potential users using a central server. In order to prevent the authorities from blocking all volunteer addresses, innocent IP addresses (such as root DNS servers) are mixed into the list to force authorities to probe all addresses, and collaborative spying by volunteer hosts is used to identify authority IP addresses so that volunteer hosts can pretend to be innocent to requests from authorities.

To increase the number of volunteers in the system, Nobori explained, the relay program must be easy to use and install, and the program must function behind a NAT. These issues are addressed in the current implementation. The server program supports multiple VPN technologies, is based on the SoftEther VPN Server (<http://www.softether.org>), and supports NAT traversal. The system was launched on March 8, 2013, which led to a response from the authorities four days later using manual and then automated blacklisting, which was solved by the insertions of innocent IP addresses. The authorities then began to probe IP addresses, which led to the implementation of collaborative spying, where multiple abnormal connections to multiple servers are classified by a central server as an authority IP address.

Nobori presented an evaluation of the effectiveness of VPN Gate at evading censorship. At the start of the system's deployment, users were forced to attempt to connect to five different servers, on average, before a connection could be made, but after the deployment of collaborative spy detection on April 24, 2013, this decreased to 1.2 connections. Once the NSDI paper was completed, the GFW ceased attempts to censor the system.

Currently, the system is approaching 7,000 active servers and 400,000 active daily users, with 32,000 unique Chinese IP addresses, which is 13 times the estimated users of Tor. The system has also been deployed in Iran and North Korea, and the total bandwidth is approaching 3 Gbps. In conclusion, Nobori compared Tor and VPN Gate: They have a similar number of servers, but Tor has a much smaller number of unique Chinese users than VPN Gate, only supports certain technologies, and has weaker firewall resistance. Nobori concluded by saying that the system allows for people to help others overseas and may promote friendship, that no laws were violated in Japan, where the research took place, and that the system is based on open source software available at <http://github.com/SoftEtherVPN/>.

Aaron Gember (University of Wisconsin) asked how many different spy IP addresses from the authorities were observed. Nobori replied that approximately 2700 IP addresses were observed, but they were used for long periods of time and were reused. Matvey Arye (Princeton) asked how blocking of access to the central server list is prevented. Nobori replied that there are two mechanisms to access the list, via either an HTML table or an indirect protocol, which is accessible using VPN Gate proxy servers and HTTP mirror sites. Jon Howell (Microsoft Research) asked what would be the next move for the GFW. Could the GFW probe from many locations inside of the firewall? Nobori replied that it is difficult to predict what the GFW authority will do, but its ability to disturb the activity of Chinese users is limited. Someone asked whether it was possible to poison the server list to create a black hole, just as VPN Gate was able to poison the firewall address list. Nobori replied that this is possible, but the user can keep trying until a server is found. Someone asked whether the GFW performs deep packet inspection. Nobori said that the firewall uses three techniques: fake TCP RST packets, DNS IP reply poisoning, and IP address blacklisting. However, due to the high level of traffic, DPI is not scalable to all Chinese traffic.

Data Storage and Analytics

Summarized by Chien-Chun Hung (chienchun.hung@usc.edu)

Bolt: Data Management for Connected Homes

Trinabh Gupta, The University of Texas at Austin; Rayman Preet Singh, University of Waterloo; Amar Phanishayee, Jaeyeon Jung, and Ratul Mahajan, Microsoft Research

Trinabh Gupta started the presentation by talking about the need for data management of connected devices in the home environment. Some motivation scenarios include: (1) applications would generate data on a time basis and retrieve based on the time window, which means the data management system would need to support time-series, tagged data; (2) applications would access the data from multiple locations, and so the data management system would need to leverage the cloud servers for availability; (3) applications might share sensitive home data, and so the data management system would need to ensure confidentiality and integrity.

Gupta then presented Bolt, the data management system addressing the above-mentioned design concepts. There are four main features. First, end-users perform data encryption, while Bolt supports the append-only data abstraction “<timestamp, tag, value>” and batching data for efficiency. Second, Bolt supports data query based on the time-window “<start, end>,” while the lookup and computation are performed locally at home. Third, Bolt leverages the cloud to support data availability. Finally, Bolt enables data security by decentralized access control.

The current Bolt implementation supports Windows Azure and Amazon S3. Its main performance gain over OpenTSDB is mainly due to data prefetching for subsequent data; the current data query is most likely to initiate the need for subsequent data records, batching for data query, and Bolt is 3–5x more space efficient than OpenTSDB.

Rik Farrow asked whether the system is mainly designed for Microsoft Home OS, and Gupta answered that Bolt can run outside of Home OS. George Porter (UCSD, session chair) asked how the authors collected the home sharing data from cameras. Gupta replied that currently all the results shown are based on synthetic data.

Blizzard: Fast, Cloud-Scale Block Storage for Cloud-Oblivious Applications

James Mickens, Edmund B. Nightingale, Jeremy Elson, and Darren Gehring, Microsoft Research; Bin Fan, Carnegie Mellon University; Asim Kadav and Vijay Chidambaram, University of Wisconsin–Madison; Osama Khan, Johns Hopkins University

James Mickens described the goal of this work as providing virtual block storage over commodity storage hardware in the cloud for cloud-oblivious applications using POSIX or WIN32 APIs, as if it is mounted locally, while achieving throughput greater than 1000 MB/s. Mickens outlined three key design challenges: (1) I/O dilation, that is, subsequent I/O operations generated by the end-host users may not be exactly subsequent when they arrive at the backend storage due to the delay dilation; (2) cross-rack I/O congestion limits execution parallelism; and (3) fsync() only returns when all writes complete.

This paper proposes Blizzard, which uses nested striping that assigns virtual disks (provided for the end-host users) randomly among physical disks. This random matching surprisingly achieves good performance. Blizzard also uses a Flat Datacenter Storage (FDS) style full bisection bandwidth network, and an RTS/CTS mechanism to avoid edge congestion. Finally, Blizzard delays later writes until earlier writes are flushed.

Keith Smith (NetApp) wondered if the paper’s response to I/O dilation could be solved using prefetching data. Mickens pointed out that one of the key design goals for this work is to support unmodified applications, while prefetching does requires application modification to some extent despite the fact that prefetching does help alleviate I/O dilation. Someone from UBC said that fsync is evil, and that using dsync and osync should be preferred

for ordering and durability. Mickens reminded the questioner that a design goal was not to modify existing applications. Someone who works on DynamoDB (Amazon) wondered about reads when writes were still buffered. Mickens responded that they deal with that issue. Ashton Goel (University of Toronto) worried about applications, such as mailtools, that expect durability. Mickens answered that some applications are fine with delayed durability and are willing to engage in that tradeoff.

Aggregation and Degradation in JetStream: Streaming Analytics in the Wide Area

Ariel Rabkin, Matvey Arye, Siddhartha Sen, Vivek S. Pai, and Michael J. Freedman, Princeton University

Ariel Rabkin began by showing that backhaul bandwidth is intrinsically inefficient, because it is sometimes under-provisioned and sometimes over-provisioned for streaming data analytics. The authors addressed this problem and optimize the use of WAN bandwidth by proposing a data streaming framework, JetStream, with aggregation and degradation mechanisms. Aggregation is done by merging records with the same property, whereas degradation is achieved through sampling and filtering data according to the current bandwidth capacity.

To realize its goals, JetStream requires a few characteristics for data abstraction: that data be updateable (locally and incrementally), reducible in size (with predictable accuracy cost), and mergeable (without accuracy penalty). A key design component is the data cube, which is the multi-dimensional array indexed by a set of dimensions. Cubes can be rolled up and unify the storage and data aggregation. Another key design for supporting auto degradation is to coordinate between network operators so that the degradation can be achieved to match the bandwidth capacity appropriately, although there is a tradeoff between bandwidth saving and accuracy.

Someone from Microsoft Research asked how to quantify the accuracy penalty during degradation. Rabkin confirmed that the accuracy penalty is generally well-behaved based on the granularity of sampled data records, e.g., from second level to minute level. The same person followed up with some scenarios—e.g., calculating the maximum values within a data set, the accuracy penalty is therefore not clear during degradation. Rabkin replied that in that kind of case, degradation could be achieved by dropping the low-ranked values, instead of coarsening the data granularity. Kurt Colovson (VMware) suggested that they might want to impose some hysteresis. Rabkin replied that they could add something for stability in the controllers.

GRASS: Trimming Stragglers in Approximation Analytics

Ganesh Ananthanarayanan, University of California, Berkeley; Michael Chien-Chun Hung, University of Southern California; Xiaoqi Ren, California Institute of Technology; Ion Stoica, University of California, Berkeley; Adam Wierman, California Institute of Technology; Minlan Yu, University of Southern California

Ganesh first pointed out that approximation analytics, which trades complete results for quicker responses, is becoming more trendy and general in big data analytics. What makes scheduling

for approximation jobs challenging are the certain requirements (e.g., deadline, accuracy) and the existence of stragglers. Specifically, should the stragglers be mitigated by speculation, dynamically prioritizing between original and speculative tasks, while meeting the deadline/accuracy requirement?

Two heuristics for straggler mitigation are Greedy Speculation (GS, which is more aggressive) and Resource Aware Speculation (RAS, which is more conservative). Ganesh talked about the guidance from analysis model, which shows that optimal scheduling strategy is to start speculation conservatively in the early stage of a job, then turns aggressive as the job gets close to completion. Based on the guidance, the authors propose GRASS, which starts with RAS first and switches to GS as the job is about to complete, while the switching point is learned from job samples. Experimental results obtained from Hadoop and Spark implementations suggest GRASS improves deadline and accuracy by 47% and 38%, respectively.

One person asked about the task duration's dependency on data content. Ganesh replied that GRASS assumes each data unit contributes equally to the computation volume within a job. Moreover, GRASS results show that when the estimation accuracy is low, sticking to RAS is better than switching to GS.

Interpreting Signals

Summarized by Pan Hu (panhu@cs.umass.edu)

Bringing Gesture Recognition to All Devices

Bryce Kellogg, Vamsi Talla, and Shyamnath Gollakota, University of Washington

Bryce Kellogg started his presentation by asking for new interaction technology that goes beyond mouse and keyboard. He focused on gesture recognition in this presentation. After pointing out the drawbacks of Kinect-based gesture, such as failing to work in non-line-of-sight scenarios and consuming too much power, he demonstrated in a mobile application the authors' ultra low-power gesture recognition system using hand gestures near his pocket, where he kept his mobile device, to change songs and increase and decrease volume.

Bryce further explained the technology detailed behind the demo. He introduced AllSee, the first gesture recognition system that runs without batteries, their prototype leveraging the ambient signal from RFID and TV signals with the intent of expanding into WiFi and cellular. The hardware architecture consists of a wireless signal receiver, some digital logic, and an energy harvester circuit. Bryce pointed out that traditional receivers that actively generate carrier waves consume too much power. Since the power AllSee gets is from ambient, the energy harvester is very small (~40uW) and it is not possible to employ traditional radio on these kinds of devices. The power budget also limited the capability of the AllSee receiver. By using a technology that is similar to Ambient Backscatter introduced in SIGCOMM '13, Bryce managed to get the amplitude information from the receiver. This poses another challenge to the digital

signal process since it contains neither frequency nor phase information, and thus traditional gesture recognition technology based on Doppler or angle of arrival could not be implemented on AllSee. What's more, Bryce pointed out that it is unlikely that a machine-learning algorithm would be implemented on battery-free devices.

Their solution was to build an amplitude library of gestures and focus on the trends. Then they proved their algorithm was simple but accurate—94% gestures were correctly classified. By duty cycling the microcontroller, Bryce pushed the energy consumption of AllSee to merely 30uW, which fit into the power budget perfectly.

Multiple people showed deep interest in their technology. Deepak Ganesan (University of Massachusetts) first asked about the working range and orientation sensitivity of AllSee. Bryce replied that the working range is about 2.5 feet. Direction could have an effect on the performance of AllSee since it uses a dipole antenna, but the effect could be cancelled through calibration. Deepak then asked whether it was always possible to harvest 40uW from the environment—for example, in rural areas. Bryce replied that although TV signals are quite pervasive, they are looking to extend the technology to cellular and WiFi so that energy harvesting is guaranteed. Pengyu Zhang (University of Massachusetts) asked whether the sampling rate (200Hz) would limit the accuracy of gesture recognition or not. Bryce answered that 200Hz works well for most scenarios. It also provides a good balance between power consumption and accuracy. Joshua Smith (University of Washington) asked about future work. Bryce told us they hope to integrate AllSee into commercial off-the-shelf devices.

3D Tracking via Body Radio Reflections

Fadel Adib, Zach Kabelac, Dina Katabi, and Robert C. Miller, Massachusetts Institute of Technology

Fadel Adib introduced a motion tracking system that uses reflected signals only. To answer the question “Can we see through walls with wireless signals?” he introduced the WiTrack, a system that tracks 3D motion of a user behind a wall by using radio signal reflected back from the user. After that, he demonstrated his system with a video of a man walking on a white line. A laptop in another room showed the real time position of the man, which was very accurate. He also showed that WiTrack could track body part movements by gesturing with an arm to turn off a light or a TV. At the end of the video he shows that the man could turn off the light in another room, which is impressive and attendees started to laugh. Fadel concluded that WiTrack could achieve centimeter-level accuracy, which is suitable for gaming, gesture control, rescue, or monitoring the elderly.

Fadel introduced the architecture of WiTrack after the interesting demo. The first step is to measure the distance in order to get the position of the user. This can be done by computing the time of flight of the reflected signal, but it requires a very high

sampling rate. In addition, this method is susceptible to noise, which is not suitable for sensing behind a wall. Instead, Fadel introduced frequency modulation continuous wave radar, which sends out a chirp signal rather than a signal of a single frequency. By analyzing the frequency difference between the sent and reflected signal, WiTrack can get very accurate distance measurements. Fadel also shared one of the challenges in implementing the system: multipath. Not only does the human reflect the signal, other objects including the wall also reflect. However, background signals can be eliminated by doing subtraction by assuming that the user is moving while the environment is not. However, it is still possible to see multiple peaks due to dynamic multipath such as people interacting with a table. This can be solved by always looking at the shortest path because the direct path is always shorter than reflected paths.

After obtaining the distance, Fadel explained how to achieve localization by putting three antennas for trilateration. The WiTrack system works from a bandwidth of 5.5 to 7.2GHz with a transmit power of 0.75mW. A motion tracking system is used to serve as ground truth. Experiment results showed that WiTrack could achieve 10 cm, 13 cm and 21 cm in three directions. Fadel also showed that WiTrack could have achieved accurate motion detection with an orientation error of 11 degrees. The accuracy of fall detection is also very high.

Joshua Smith (University of Washington) asked about the regulation of this kind of device since it may involve privacy issues. Fadel answered that people could try to block these signals or the government could regulate their usage. Joshua also asked whether it is easy to filter out background signals. The answer is not easy because there are other moving objects such as fans in the room. Fadel also pointed out that steering the RF signal beam might help to filter out the background signal. Deepak Ganesan asked whether segmentation is needed in the gesture recognition and Fadel answered yes. Jie Liu (Microsoft Research) asked about the difference between WiTrack and traditional radar and sonar. Fadel pointed out that traditional radar fails to deal with multipath in indoor scenarios; what's more, it cannot provide centimeter-level accuracy. Jie Liu also asked about the possibility of trying to distinguish between people. Fadel replied that it could be done by looking at the reflected pattern. Finally, Jie asked how the properties of the wall might affect the wireless signal and its accuracy. Fadel answered that the wall may have a small impact on the accuracy although they haven't tested on different kinds of walls yet.

Epsilon: A Visible Light Based Positioning System

Liqun Li, Microsoft Research, Beijing; Pan Hu, University of Massachusetts Amherst; Chunyi Peng, The Ohio State University; Guobin Shen and Feng Zhao, Microsoft Research, Beijing

Localization is a hot (and old) topic throughout the last decade but Guobin Shen presented something new. Recent localization technology based on WiFi, cellular, FM, or magnetic fields either failed to provide high accuracy or required a complicated infrastructure. Guobin presented Epsilon, an indoor localization

system based on visible light. Guobin suggested that LED lighting is the future light for industry due to its high efficiency and longer life. And LED as a semiconductor component intrinsically supports rapid switching on-and-off. By taking advantage of this character, current dimmer switches can change the light intensity smoothly by switching on-and-off using different duty cycles. Inspired by this fact, Guobin said we could use LEDs as beacons. The advantage of using visible light includes higher beacon density due to the pervasive existence of LED lights and getting humans into the loop. The system leverages the current infrastructure and thus has minimal cost.

Guobin then explained how to implement this system. Epsilon adopted a model-based rather than a fingerprint method to reduce the cost needed to deploy the system. Since both the LED and light sensor are directional to some extent, Epsilon models the irradiation angle from the LED and the incidence angle to the light sensor. Evaluation shows that the accuracy of the model is pretty good when both angles are small but that relative error increases as the angles increase. Guobin also shared some practical design considerations for Epsilon. Firstly, commercial LEDs could not be modulated with a frequency higher than 100 kHz, and they need to modulate faster than 200 Hz to avoid perceptible flicker. What's more, power line frequency could also affect the positioning system. By carefully choosing the frequency and modulation method, Epsilon managed to meet all these needs. Guobin also talked about how to deal with a scenario with only one LED light. Epsilon could involve human motions to improve the position results. For example, by letting users tilt their cell phones while using the IMU, it would still be possible to get the position. Experimental results showed that the 90th percentile accuracy is 40 cm, 70 cm, and 80 cm in three dimensions.

Joshua Smith asked about privacy issues of this system when compared with other ones. Guobin replied that Epsilon might have better privacy preservation because light can be more easily blocked than wireless signals. Joshua also asked whether it is possible to achieve higher accuracy by increasing the modulation frequency. The answer was no, because Epsilon only uses signal strength to measure distance in the model, which is irrelevant to frequency. Deepak Ganesan asked whether it is possible to achieve higher accuracy by deploying more LED lights. Guobin answered yes, although Epsilon will always choose the four LED lights with the highest signal strength. Deepak also asked whether the orientation of the user would affect the accuracy or not. The answer was yes, but they have taken this into consideration when evaluating their system.

Improving Throughput and Latency (at Different Layers)

Summarized by Muhammad Shahbaz (shahbaz@cc.gatech.edu)

Enabling Bit-by-Bit Backscatter Communication in Severe Energy Harvesting Environments

Pengyu Zhang and Deepak Ganesan, University of Massachusetts Amherst

Zhang started the talk by giving a short description of their new design for backscatter systems, which operates in severe energy harvesting conditions. He then talked about the current trends in lower-power sensors, specifically, the micro-powered sensors. These micro-powered sensors are equipped with energy harvesters and don't need batteries to power the whole system, which forces them to operate at very low energy. That's why these sensors are finding applications, specifically in the bio-sensing domain, such as delivering glucose in the blood stream and measuring vital signs using wearable sensors such as Google contact lenses. The physical limitations of these sensors makes it hard to attach batteries and thus designers are trying to find ways of using ambient and external energy sources to power the system—for example, harvesting energy from solar cells and from wireless signals.

Zhang et al. were interested in answering the question of how these devices interact with the outside world and what are the limiting factors. They looked into the example of RF harvesting and backscatter communication and found that the maximum operating range of these systems is 3.6 feet, or five times less than their communicating range of around 18.6 feet. This is because the atomic unit of existing network stacks didn't fit into the single discharge cycle and, thus, was limiting the overall operating range of the system. Using 1-bit as an atomic unit, they were able to achieve an operating range of 18 feet, but in that case throughput suffered at close distances. Later in the talk, Zhang gave different insights into maximizing communication throughput while maintaining the capability to operate at maximum range. By carefully selecting sleep time, the number of TX bits, and interleaving sensors on the packet level using a novel packet fragmentation and transmission scheme, they were able to achieve a 3.5 times increase in the operating range and a 5.8 times improvement in the overall throughput compared to the Dewdrop system.

Joshua Smith (a WISP designer) asked if the data doesn't come through, how will retransmission interact with the fragmentation scheme? Zhang argued that their scheme fragments the packets including the CRC into small microframes and performs all the error correction and detection once the packet is reassembled at the receiving end using the existing methods. In reply to Josh's follow-up question about the cost of the fragmentation scheme, Zhang answered that the packet is encoded into a fixed structure with a particular form, so they used trailing and leading bits to identify where packets began and ended. This adds some cost to the fragmentation scheme.

Full Duplex MIMO Radios

Dinesh Bharadia and Sachin Katti, Stanford University

Bharadia started the talk with a refresher on why full duplex radios have been considered an impossibility. He gave an example of two radios where the sending radio cannot receive the weaker signal because its own transmitting signal is acting as strong self-interference, which drowns the incoming signal. He said that it's analogous to hearing a whisper when one is shouting at the top of one's lungs. He later stated that the recent work in their lab, presented in last year's SigComm, has invalidated this assumption, and they have been able to demonstrate a fully working single antenna full duplex radio.

Before jumping into the details of their full duplex MIMO (multiple-input and multiple-output) design, Bharadia raised a question of why full duplex matters. Is it just the doubling of speed? He answered himself saying that the current performance curve is flattening, and we are running out of link layer techniques as well as reaching the channel limit. That's why full duplex is considered the next logical step and there is active interest from industry. Bharadia argued that for full duplex to be viable it needs to support MIMO. For full duplex MIMO radio, the design of an adaptive filter that matches the environmental reflections and cancels them should impose low complexity and minimal residue after cancellation.

Bharadia discussed the technical contributions of their work, where they designed a MIMO cancellation filter that has linear complexity in the number of antennas and doesn't scale quadratically compared to SISO replication-based design. In addition, the cancellation residue is ideal and doesn't degrade with the increasing number of antennas. In the end, he showed that their design does provide the proposed throughput scaling. The speedup is 1.95x, compared to the 35% improvement in SISO replication design, when matched against the standard half duplex design.

Shyam Gollakota (University of Washington and session chair) asked Zhang to compare the cost of WARP with the typical WiFi chipset, which by comparison is way more expensive than the WARP board's low cost. Gollakota also asked how the WARP components have greater linearity and the board has less noise compared to the typical chipset provided in mobile phones and access points. For the second question, Bharadia answered that linearity is typically based on standards. For example, WiFi needs 25 dB of maximum SNR (signal-to-noise ratio) so they design it at 30 dB and WARP also has around 30 dB of linearity. In his reply to the first question, he said that WARP may be \$10,000 but the transceivers (Maxim chips) the board uses are fairly cheap, around \$2 each. One can buy these off the shelf but this would require a lot of integration effort. He said that his point was when you buy these cheap transceivers they don't optimize linearity beyond a certain range. Thus, you have to cancel a lot of nonlinearities and noise. This requires building an analog

design rather than a digital design because you cannot model noise in the digital domain.

Recursively Cautious Congestion Control (RC3)

Radhika Mittal, Justine Sherry, and Sylvia Ratnasamy, University of California, Berkeley; Scott Shenker, University of California, Berkeley, and International Computer Science Institute

Radhika Mittal started her talk by presenting a different view of the congestion control problem and how it's still unsolved. She mentioned the importance of classic work on congestion control by Van Jacobson and others and how it has helped in avoiding congestion collapse in the networks. But, she argued, this is not the only goal of congestion control, and users actually respond to fast completion time, which indirectly results in better revenue and profit.

Radhika presented a really interesting scenario: David Clark is a service provider providing services to the sender, Van Jacobson, and receiver, Vint Cerf. She showed how both resources (provisioned by the service provider) and link capacity are underutilized because of the current congestion control schemes. These schemes try to find a sweet spot between two conflicting goals of maximizing the throughput without adversely affecting other flows. RC3 tries to decouple these goals using priorities. RC3 utilizes spare capacity by sending additional packets using low priorities. This reduces the flow completion time and increases utilization of the bandwidth resources. She showed that assigning priorities required minimal changes in the TCP stack. Two parallel control loops, one for the regular TCP and the other for sending packets with multiple levels of priorities, were used to achieve max-min fairness.

Simulation results showed that RC3 performed 43.54% and 74.35% better on average over flows and bytes, respectively, compared to regular TCP. RC3 also showed better gains in completion time for short flows than existing schemes like RCP. Although better than regular TCP, the unoptimized RC3 implementation on Linux behaved relatively poorly compared to the simulated RC3 results, because the low priority packets were being processed by the slow path, thus causing high CPU overhead. The authors showed that leveraging NIC offloading capabilities like TSO and LRO reduced the overhead by half.

They also listed some cases where RC3 provides smaller benefits: for example, low delay bandwidth product and heavily utilized link, and some deployment concerns like partial support for priorities in switches. Looking toward the future, performance will eventually improve with the increasing bandwidth and round-trip time (RTT) product. The authors forecast a 45% and 66% reduction in average flow completion time over flows and bytes, respectively, in the futuristic datacenter with 100 Gbps bandwidth.

Dongsu Han (KAIST) asked about fairness among low priority packets. Radhika answered that this is ensured using the recursive multi-level priority scheme. Han asked whether there

is an upper limit to the number of priority levels needed. Radhika replied that because they are using an exponential increase in the amount of packets sent per priority level, eight levels, for example, should be able to handle large flows with terabytes of data. Shyamnath Gollakota asked whether increasing the size of the buffer would have any effect on overall performance and delay. Radhika argued that it won't have much effect but will require more memory, but because memory is getting cheaper, this might not be an issue.

How Speedy Is SPDY?

Xiao Sophia Wang, Aruna Balasubramanian, Arvind Krishnamurthy, and David Wetherall, University of Washington

Xiao Wang gave a brief history of HTTP 1.1 and its global usage and mentioned, at the same time, that HTTP is not working as fast as desirable due to the increasing complexity of Web pages. Around 2009, Google introduced a new protocol called SPDY to solve many of the concerns in HTTP: e.g., opening a single TCP connection vs. opening individual connections for each object; the client's ability to prioritize objects; the server's ability to initiate requests, thus, avoiding extra RTTs; and compressing page headers along with the data. SPDY is now deployed in Chrome, Firefox, and on many Web sites. It's also the basis of HTTP 2.0, which is currently under standardization.

After briefly discussing the advantages of SPDY over HTTP, Wang listed various challenges, like variance in page load times and dependencies between network and browser computation. She then stated the goals of their paper: to perform a systematic study of SPDY and identify the dominant factors that are causing variability in its performance. The authors' approach was to extensively sweep the parameter space (e.g., network parameters like RTT and BW, TCP settings, and Web page effects) and isolate the dominant factors. Their finding was that SPDY helps on small objects because SPDY can batch up small objects into a TCP segment; SPDY helps on large objects under low packet loss due to reduced retransmissions from a single flow; and SPDY hurts on large objects under high packet loss because it backs off the TCP congestion window more aggressively than HTTP. Wang also found that object number, object size, and packet-loss rate are stronger indicators of SPDY's performance than RTT, BW, and TCP's initial cwnd. Their tests on real objects, although ignoring browser effects, showed that SPDY helped a lot, mainly due to its single TCP connection.

To capture the effects of the browser without worrying about variability in page load times, the authors introduced a new tool called Eload. They found that SPDY helps marginally when browser effects are preserved and suggested that the performance impact decreased due to browser computation and dependencies in real pages. For further improvement, the page load process needs to be restructured, for example with server push. Wang mentioned that people can download WProf and Eload at <http://wprof.cs.washington.edu/spdy>.

Someone asked about considering the effect of caching at the client or network or both. Wang answered that caches present similar behavior as having a small number of objects. If the number of objects is very large, SPDY actually helps less, but they don't consider the caching case in the network. James Mickens (MSR) started his question by saying that all browsers are terrible, which made the audience laugh. He said that they not only differ in their computation but also along other axes like storage and the network stack they use. He asked how the emulator, Eload, generalizes across all these different browsers. Wang explained that Eload is based on WProf, the work they presented in last year's NSDI. Using WProf, Eload captures these dependencies in a browser-independent manner. She said that they ran test cases under different browsers in order to capture dependencies across browsers. Wang did mention that the emulator cannot capture the computation time variability, but based on the platform, they can hypothetically increase or decrease the compute time.

In-Memory Computing and Caching

Summarized by Qiao Zhang (qiao@cs.washington.edu)

FaRM: Fast Remote Memory

Aleksandar Dragojević, Dushyanth Narayanan, Orion Hodson, and Miguel Castro, Microsoft Research

Aleksandar Dragojević started by discussing two hardware trends that make FaRM timely and important. He noted that it is becoming cost-effective to store almost all application data in memory. While new datacenter networks promise larger throughput and lower latency, network communication remains a bottleneck for systems that use TCP/IP. Fortunately, RDMA technology that allows direct read/write of remote memory is now available at competitive prices. To take advantage of the performance gains from kernel bypassing, FaRM builds a message passing primitive using fast RDMA writes. Microbenchmark for remote random reads between RDMA, RDMA-based messaging, and TCP shows that FaRM's RDMA-based messaging can achieve an order of magnitude improvement on both throughput and latency compared to TCP.

FaRM explores how to use RDMA to build distributed systems. Aleksandar explained that in order to program a modern cluster with terabytes of memory, hundreds of CPUs, and RDMA network, we want to keep data in memory, access data using RDMA as much as possible, and co-locate data and computation because accessing data locally is a factor of 20 faster than accessing remotely even using RDMA. Moreover, RDMA lends itself to a symmetric model where machines not only store data but also execute applications in order to exploit data locality and to avoid idle server CPUs.

FaRM simplifies programming by providing a shared address space and general distributed transactions with strong consistency guarantees. FaRM allows applications to read/write and create/free objects in a shared address space using ACID transactions. FaRM supports locality-aware optimizations that allow

new objects to be created closer to existing objects to exploit locality. For performance-critical operations, FaRM allows applications to perform an efficient lock-free read in a single RDMA operation by taking advantage of cache-coherent DMA.

Aleksandar presented the implementation of a distributed key-value store and a Tao-like in-memory graph store on top of the FaRM APIs. The FaRM key-value store achieves 16x the throughput and two orders of magnitude improvement in latency compared to the state-of-the-art TCP-based key-value store. The FaRM graph store achieves 10x the throughput and a 40x–50x improvement in latency compared to the TCP-based in-memory graph store.

Someone asked whether it would be more accurate to compare a FaRM messaging primitive to UDP since RDMA requires loss-less networks. Aleksandar replied that UDP can achieve lower latency but it is nowhere close to RDMA. Hein Meling (University of Stavanger) asked whether the next step for FaRM is to support across-datacenter applications. Aleksandar said that the latency between datacenters would be too high. The next step is to explore the right primitives to put in the NIC to further improve performance and enable new functionality, and also to implement other applications using FaRM. Someone asked what the difference was between FaRM and RAMCloud. Aleksandar answered that the main difference stems from motivation: RAMCloud is a key-value store, whereas FaRM exposes a much richer and more general programming model with shared address space and distributed transactions. Jeff Rasley (Brown) asked why the evaluations are done using only 20 machines and whether that signals scalability limitations with RDMA. Aleksandar replied that they only had 20 machines and commented that FaRM can scale to 100 machines. Wang (Cisco) asked whether FaRM migrates data to exploit locality. Aleksandar replied that FaRM does not migrate data but keeps data and computations together, and further commented that they expect FaRM to run a single large application, so there would be no contention.

Easy Freshness with Pequod Cache Joins

Bryan Kate, Eddie Kohler, and Michael S. Kester, Harvard University; Neha Narula, Yandong Mao, and Robert Morris, MIT/CSAIL

Bryan Kate started his talk by arguing that application caches should support materialized views natively because in-cache materialized views are easy to use and have good performance. Existing application caches, like Memcached and Redis, provide fast key-value cache to offload reads from database, but the burden of maintenance rests on applications to keep the cache fresh. As a motivating example, Bryan explained how the Twitter timeline is constructed from a join between subscription lists and user data, and pointed out that the result of the join should be cached because the timeline is checked very frequently. While it is easy to cache the join, it is difficult or cumbersome to update the cache when there are new posts. A simple solution is to use a materialized view supported by modern database systems

that compute, store, and automatically update the query results. However, the database is often designed for durable storage and therefore becomes a performance bottleneck when tasked to handle frequent reads and writes.

In order to help applications to avoid the complexity of keeping the cache fresh and to provide good performance, Bryan presented Pequod, a distributed application cache that provides materialized views in a key-value cache with operations such as get, put, scan, and join. Bryan introduced the key idea of Pequod cache joins that allow applications to relate computed data (e.g., timeline) to base data (e.g., posts and subscriptions). Pequod also offers a number of advanced features such as partial and dynamic materialized views, incremental updates, and eager or lazy updates. Bryan highlighted that distributed Pequod can scale to handle large data sets. Computation is kept local while base data is partitioned and transparently replicated when necessary to allow cache joins to be computed anywhere. Finally, Pequod supports cache eviction under memory pressure and allows for eventual consistency.

Pequod was evaluated on a Twitter-like benchmark, including timeline checks, new subscriptions, and new posts. The first experiment compared Pequod with fast key-value caches, such as Memcached and Redis, and a DB-as-cache database, such as Postgres. Results showed that Pequod performed no worse than existing caches. The second experiment tested how Pequod performance scaled with additional servers. Results showed a 3x increase in performance when the number of servers increased from 12 to 48. The imperfect scaling resulted from data movement between servers. The overhead was noticeable but not crippling. Bryan mentioned related work, e.g., DMV, DBProxy, and PNUTS.

Someone asked about latency in Pequod and how latency is affected when data are evicted under memory pressure since Twitter-like applications expect low latency. Bryan replied that there is a latency spike the first time a cache join is computed, and that applications can scale out to hold a larger cache to minimize eviction and avoid high latency. Moreover, Pequod allows tunable eviction of computed data vs. base data. A further question concerned eventual consistency in Pequod when Twitter users expect up-to-date results. Bryan answered that application developers have grown used to eventual consistency, and in Twitter, writes may take up to a second to trickle to user timelines.

MICA: A Holistic Approach to Fast In-Memory Key-Value Storage

Hyeontaek Lim, Carnegie Mellon University; Dongsu Han, Korea Advanced Institute of Science and Technology (KAIST); David G. Andersen, Carnegie Mellon University; Michael Kaminsky, Intel Labs

Hyeontaek Lim presented MICA, a fast in-memory key-value store. MICA aims to improve per-node performance and mainly targets small k-v items that can fit in single packets. Hyeontaek compared the end-to-end performance over the network on the

YCSB workload between MICA and current systems (e.g., Memcached, RAMCloud, MemC3, Masstree) using a single server node. While the performance of the current systems collapses under a write-intensive workload, MICA achieves orders of magnitude improvement for both uniform and skewed workload and attains close to maximum packets/sec possible using UDP.

Hyeontaek explained that the significant improvement comes from MICA's approach, which applies new software architecture and data structures to general-purpose hardware in a holistic way. He outlined three key design choices: First, to avoid frequent cache line transfers between cores and to allow CPU performance to scale with the number of cores, MICA partitions data and gives each core exclusive access to a partition. Experiments show that exclusive access outperforms concurrent access in MICA. Second, MICA uses client-assisted NIC-based request direction to ensure correct and high-throughput delivery of requests to the respective cores for exclusive access. Third, MICA proposes a new "cache" data structure that uses a circular log and lossy concurrent hash index for each partition to provide high throughput for both reads and writes. Hyeontaek emphasized that these unconventional design choices allow MICA to achieve good performance for both throughput and latency, and for both uniform and skewed workload, compared to current systems. Source code can be found at github.com/efficient/mica.

Someone asked how much each of the three tricks contributes to the performance gain. Hyeontaek replied that the paper contains experimental results that show performance comparisons between each design choice and its alternative, e.g., concurrent and exclusive access. Someone from the Voldemort project commented that the performance gain might be much smaller, taking into account the management overhead once MICA becomes fully featured, because it is not fair to compare a research prototype with a fully featured product like Redis or Memcached. Someone from UCSD asked how MICA compares to partitioned k-v stores such as single-threaded Memcached in each core. Hyeontaek answered that such deployment can reproduce most of the gains from MICA design. However, Hyeontaek pointed out that MICA has a hybrid mode that allows both concurrent and exclusive access that is only possible in MICA architecture. Ryan Stutsman (Microsoft Research) asked how MICA keeps track of where cache misses are coming from since circular logs can evict data and result in dangling references. Hyeontaek answered that MICA does not do dynamic reconfiguration of the system. As a solution, one can use lossless data structure or try to predict how much of the losses are coming from data structures.

Scalable Networking

Summarized by Feng Lu (f1lu@cs.ucsd.edu)

NetVM: High Performance and Flexible Networking Using Virtualization on Commodity Platforms

Jinho Hwang, The George Washington University; K. K. Ramakrishnan, Rutgers University; Timothy Wood, The George Washington University

Jinho Hwang began by reviewing recent developments in high performance networking such as PacketShader, Intel DPDK, etc. When the datacenter is virtualized, the performance of these systems could suffer due to the virtualization overhead. He then turned his focus to a specific domain—network function virtualization—and identified two potential challenges: high speed/low latency processing and efficient inter-function (VM) communication. Both challenges motivated his work, NetVM, which aims to provide complex network functionality at line rate (10 Gbps) using commodity hardware (e.g., Intel DPDK and commercial off-the-shelf servers). Jinho discussed two possible placements of DPDK in a virtualized environment: in the hypervisor and in the VM with on-NIC L2 switch. He pointed out that neither could achieve a full line-rate network speed in VMs.

Having explained the limitation, Jinho described the overall design of NetVM, which allows memory sharing between hypervisor and VM, and among VMs themselves. Subsequently, he sketched out the design challenges faced by NetVM: how to ensure zero copy, huge-page sharing between the VM and the hypervisor, lockless and NUMA-aware design, and security domains. He then talked about how NetVM addressed each challenge in turn. For zero-copy, NetVM directly DMAed packets into shared memory and only packet control structures are passed around. For sequential packet processing, packet references are passed between VMs. At any time, only one VM can process the packet to limit concurrency. For lockless and NUMA-aware design, NetVM has dedicated core-queue matching, data-structure separation, and processing path alignment in both the hypervisor and the VMs. For huge-page virtual address mapping, pre-calculation was used to speed up offset lookup. Finally, trusted VMs were using shared memory in NetVM while untrusted VMs see packets via the hypervisor.

Afterward, Jinho moved to the evaluation section and compared NetVM to SR-IOV-VM, Click_NetVM, and Click_Native_Linux. NetVM was able to attain packet delivery and forwarding rate at full line speed, while second best Click_NetVM, was only able to achieve 6 and 6.8 Gbps, respectively. For inter-VM forwarding evaluation, due to the limited number of cores, NetVM could only sustain a line rate of up to three VMs. However, Jinho also mentioned that for a more realistic workload (60% partial forwarding), NetVM would maintain a 10 Gbps line rate up to five VMs.

Wang (Cisco) said that it is common to pass packets in and out of applications, but in reality, people are used to socket interface; his question was about the programming model. Wang said that Jinho mentioned netmap; the beauty of netmap is that it also supports a socket-like interface. Jinho restated the question as

a protocol stack issue, mentioning that NetVM was targeted for middlebox applications, which handle packets. However, there was some software running within applications, such as mTCP, that provided a user-level network protocol within it. In most cases, people use a customized user-level network stack. Wang then asked how much code needed to be changed. Jinho mentioned that KVM was modified to achieve queue/core alignment and to add a PCIe device. They didn't touch any kernel module and only one kernel module was introduced to support user space I/O. Finally, Wang asked whether they dedicated one interface for one VM or could traffic from one interface be shared with multiple VMs? Jinho replied that NetVM allowed one interface for multiple VMs. But to maximize performance, they dedicated one interface to one VM.

ClickOS and the Art of Network Function Virtualization

Joao Martins and Mohamed Ahmed, NEC Europe Ltd.; Costin Raiciu and Vladimír Olteanu, University Politehnica of Bucharest; Michio Honda, Roberto Bifulco, and Felipe Huici, NEC Europe Ltd.

Joao began by pointing out the various drawbacks associated with hardware middleboxes: price, power, management, scalability, and so on. Given these limitations, Joao suggested pushing middlebox functionality to software for the following benefits: shared hardware across multiple tenants, reduced equipment/power costs through consolidation, and flexibility to try out new features. He then articulated the problem solved by ClickOS: Middleboxes can be built on commodity hardware while still achieving high performance. He briefly went over the achievements made by ClickOS: fast boot (30 ms), small memory footprint (5 MB), isolation by Xen, 10 Gbps line rate processing, and flexibility based on the Click library.

Joao continued the talk with more details on ClickOS architecture: Instead of a traditional guest OS on Xen, ClickOS built on top of MiniOS with a single application on a single core. In addition, the Click control plane was emulated over MiniOS/Xen and the boot time was reduced to 30 ms (uncompressing the VM takes longer than booting). He also mentioned that various optimization tricks were used to enable line rates of 10 Gbps. Joao next moved to performance analysis with a native implementation of ClickOS and identified three pieces on the packet processing pipeline: open vSwitch, netback, and netfront. Unfortunately, the performance of these modules was far behind line rate. He quoted that 14.88 million packets would be processed to sustain line rate given a 64-byte packet size. However, these modules can only process 250k–350k packets per second. He attributed the poor performance to packet copy between Click/Xen (772 ns), packet metadata allocation (600 ns), and a slow back-end switch. To address these problems, several optimization techniques were used: reuse Xen page permission, replace OVS with Vale switch, increase I/O batch size, and use the netmap API all the way to the NIC buffer (kernel bypass).

He then presented an overview of the ClickOS prototype and emphasized that the Click changes were small. He explained the

evaluation setup: Intel Xeon with Sandy bridge, 16 GB RAM, one Intel NIC and one CPU core for VMs, and the remaining cores dedicated to dom0. He navigated through the evaluation section, started with base transmission performance while varying the TX ring size. With ring size ≥ 1024 , ClickOS could achieve line rate at all packet sizes. Next, the virtualized middlebox performance was evaluated in a three node setup (host1→ClickOS→host2) for a wide range of middlebox functionalities. Finally, the effectiveness of their Linux optimization was presented. He also mentioned that the entire source code is now available at: <http://cnp.neclab.eu>. Joao concluded his talk with future work, which aimed to consolidate thousands of VMs, improve inter-VM communication, and exploit boot times to achieve reactive VMs.

Marv Ayre (Princeton) wondered about CPU load and utilization among the cores. Joao replied that there was just one core to handle the NIC if it receives packets on the virtual port—in short, just one core in dom0. For guest OS usage, if it is not receiving the packet, the CPU usage is zero. The guest OS did not use anything like DPDK, and there was no polling. Ayre then asked whether they changed any code to handle high load. Joao said their design handled high load well and referred people to the paper for the results. They booted 100 VM instances, and these instances could fill up the pipe. Someone asked about the 100 VMs example: Was anything done about the CPU scheduling or about latency? Joao replied that for 100 VMs and beyond there is likely a bit of a scheduling issue, but he emphasized that the VMs were scheduled fairly. Although the individual contribution per VM rate decreased slightly, overall they did not see any issue. Someone else asked whether they used Xen's virtual IRQ to trigger interrupt. Joao answered yes and confirmed that they used the event channel to deliver interrupt. The same person asked whether they used polling, and Joao replied that since the interrupt happens on a per-batch basis, constant interrupts aren't occurring. In addition, the guest OS does not poll the back end.

SENIC: Scalable NIC for End-Host Rate Limiting

Sivasankar Radhakrishnan, University of California, San Diego; Yilong Geng and Vimalkumar Jeyakumar, Stanford University; Abdul Kabbani, Google Inc.; George Porter, University of California, San Diego; Amin Vahdat, Google Inc. and University of California, San Diego

Siva began his talk by discussing server consolidation, multi-tenancy, and network resource management and sharing in datacenters. He further identified a key piece of functionality, namely a programmable rate limiter on the end host to ensure performance isolation and effective congestion control. He then discussed two options for rate limiters in existing systems. Software rate limiters are scalable but not accurate and precise. Hardware rate limiters (available on NICs) could easily achieve the latter two but are not scalable. The authors propose SENIC as a way to combine the advantages of both approaches by reorganizing the responsibility of the operation system and the NIC functionalities.

Siva reviewed the current NIC design and identified that existing NICs unnecessarily pre-DMAed packets from host memory

to NIC buffers, which are limited in size and not able to scale up. Instead, Siva proposed per-class queues in host memory and having the NIC compute the schedule on demand for each packet. The late binding of packet transfer to NIC enables accurate and precise rate limiting in SENIC. He continued with SENIC implementation details and discussed how they implemented SENIC on both software (as a Linux kernel module) and hardware (NetFPGA). For the software prototype, a dedicated CPU core is used for network scheduling, and for the hardware prototype, token bucket scheduling is implemented.

Siva reviewed microbenchmark results on 10G with NetFPGA. Their current NetFPGA prototype supports 1000 rate limit classes. The inter-packet delay for a traffic class was studied, and the authors found that the average and standard deviation was within 0.038% and 1.7% of ground truth, respectively. Siva next discussed scheduling latency. It takes SENIC 50 ns to compute the schedule for the next packet, which is well within the allowed budget (300 ns for 1500 bytes for 40G). In the last part of the evaluation, Siva talked about tenant isolation by configuring 10 memcached tenants (6 Gbps) sharing the network with one UDP tenant (3 Gbps). He presented the 99.9th percentile tail latency of memcached tenants while varying the number of requests. The tail latency was below 5 ms even when the aggregated load approached 6 Gbps. Additionally, the UDP client was able to attain 3 Gbps irrespective of memcached workload. Both results significantly outperformed the two existing approaches, namely HTB and PTB, provided by the current Linux kernel. Siva explained how SENIC supported other NIC features and chose TSO as an example. Source code for SENIC is available at <http://sivasankar.me/senic>.

Someone wondered how easy it is to implement their solution on commodity NICs. Siva replied that most of the parts used in SENIC, such as packet scheduler and DMA engine, are already available on the NIC. Packets are in the host memory and DMAed into the internal ring buffer. What is needed is a scalable scheduler to compute scheduling on demand and pull the packet for transmission.

mTCP: A Highly Scalable User-Level TCP Stack for Multicore Systems

EunYoung Jeong, Shinae Woo, Muhammad Jamshed, and Haewon Jeong, Korea Advanced Institute of Science and Technology (KAIST); Sunghwan Ihm, Princeton University; Dongsu Han and KyoungSoo Park, Korea Advanced Institute of Science and Technology (KAIST)

Awarded NSDI '14 Community Award!

Shinae Woo motivated mTCP by arguing the need to handle a large number of short flows and the high cost of connection management in datacenter networks. She explained why the current TCP implementation in Linux is unsatisfactory: in particular, the kernel is not well designed to sustain line rate with small flows and does not scale well with respect to the number of cores. She then presented a Web server example, where the CPU spent more than 80% of time in kernel with 34% spent in

the TCP/IP stack. She attributed the performance bottleneck to shared resource contention, broken locality, and per-packet processing overhead in the Linux kernel. She then discussed several related works in addressing the kernel inefficiency and pointed out that none of them solved all of the aforementioned problems.

mTCP is a clean-slate design for a user-level TCP implementation. Woo emphasized that mTCP is explicitly designed for multicore systems and listed mTCP design features, such as independent cores with no resource sharing, resource affinity, batch packet processing, and a portable API compatible with the current Linux kernel. In particular, she mentioned that each mTCP thread corresponds to one application thread and that mTCP threads extend the PSIO library to support efficient packet I/O interface with lock-free and per-core data structures, and she addressed the context switch overhead. Additionally, she talked about porting existing apps on mTCP; most apps required less than 100 lines of changes. She continued with some implementation details, such as 11,473 LOC for mTCP, 552 lines to patch the PSIO library, and that mTCP follows RFC 793.

Woo cited some microbenchmark results and showed that mTCP could scale linearly with CPU cores when handling 64-byte packet connections. She then discussed the performance improvements of ported application on mTCP versus Linux and MegaPipe, using two application examples: `lighttpd` and `SSLShader`. With `lighttpd`, mTCP improved over Linux and MegaPipe by 3.2x and 1.5x, respectively. With `SSLShader`, where one-byte objects were downloaded, mTCP boosted performance by 18% to 33% over Linux. The source code for mTCP is available at: <http://github.com/eunyoung14/mtcp>.

Someone asked whether they used timers to batch I/O. Woo responded that they did not use any timers to batch threads. This naturally happens with the context switching. The next questioner pointed out that from `netmap` they've learned that allocating packet structures such as `sk_buff` is expensive and that, in this work, scheduling is killing performance, maybe system call overhead as well. What was the comparative overhead given that they measured the whole kernel, that is, where were the performance gains coming from? Shinae answered that they were not aware of exactly where the performance gains come from, such as which part contributed most benefit. Context switching is generally more expensive than system calls. Shinae stressed that they batched 2000 events per context switch, and it seemed that batching provided more benefit. A third questioner wondered whether they could move the scheduling portion of mTCP into the kernel. Shinae replied that it is hard for the kernel to provide an event-driven API to applications since the two communicate via system calls. Applications would have to be changed to support an event-driven model. In mTCP, applications do not need to be changed and they still use the typical `connect()` and `send()` calls. However, the batching is provided by the mTCP stack. Shinae believed it would be hard to support batching in kernel. The final questioner noted that they had divided the application

into two parts so that applications were aligned to cores; the questioner wondered whether they had to rewrite the application so that they had a process running per core. Shinae replied that the ported applications normally support a single-process multiple-thread model, in which they can easily change the API to use enough cores. For a single-process single-thread model, they would have to make changes to turn it into a single-process multiple thread.

New Programming Abstractions

Summarized by Oliver Michel (oliver.michel@colorado.edu)

Warranties for Faster Strong Consistency

Jed Liu, Tom Magrino, Owen Arden, Michael D. George, and Andrew C. Myers, Cornell University

Jed Liu started with a discussion about the tradeoff between consistency and scalability in distributed systems. He illustrated this tradeoff by comparing relational database systems and Web-scale NoSQL systems only providing weak, eventual consistency. Essentially, these systems are harder to program against because consistency failures are likely.

To bridge this gap, the authors introduce warranties, which are time-limited assertions about the state of a system. State warranties ensure that a certain key has a specific value until some point in time, whereas computation warranties guarantee that a certain computation result holds true until some point in time. For example, such a warranty could assert that there are at least x seats available on a flight in an airline booking system until a specified time. Warranties are strictly serializable and provide a generalized form of optimistic concurrency control. They provide improved scalability and throughput in read-heavy environments, whereas write access may become a bottleneck because writes that would violate assertions are delayed until the warranty expires.

The authors' evaluation showed that a key parameter, heavily influencing the system performance, is the duration of a warranty. Thus, these durations must be chosen carefully. Warranties must be valid long enough to be useful but short enough to keep the system from blocking access completely. Finally, Liu provided performance figures showing a speedup of $2x$ is achieved by warranties for a system with only 2% of data access being writes. Performance worsens in comparison to not using warranties at about 9%–10% writes.

Aaron Gember (University of Wisconsin) asked how long warranties typically are. Jed answered that warranties are in the range of a few seconds depending on performance characteristics of the system. Someone from Microsoft Research asked whether a detailed knowledge of the performance characteristics of the system is needed to set durations for computational characteristics appropriately. The author answered that a good knowledge of the workload is necessary. Eddie Kohler (Harvard University) asked whether some warranties are more useful than others. Jed answered that this is certainly true and some

messages are not even worth having against the warranty overhead. Alec Wolman (Microsoft Research) asked how developers would tune the system if the workload changed over time. Jed replied that they haven't thought deeply about this, but would have to think about which methods get memoized. Aditya Akella (University of Wisconsin-Madison and session chair) asked whether the higher throughput resulted in higher latency. Jed said that the latency was just for write transactions, and for a read-mostly workload, this is acceptable.

Tierless Programming and Reasoning for Software-Defined Networks

Tim Nelson, Andrew D. Ferguson, Michael J. G. Scheer, and Shriram Krishnamurthi, Brown University

Tim Nelson opened by discussing the three main application tiers in programming software-defined networks: the flow-rules in the switches, the controller program, and the store for the controller state. In most languages, these three layers are abstracted differently. In fact, there is a gap between the code that the controller application executes to produce a new flow modification message and the interface that receives this information on the switch side, which leads to errors and inconsistencies between the tiers.

Tim introduced Flowlog, the authors' approach to a completely tierless programming environment for software-defined networks. Flowlog provides a cross-tier interface such that applications written in this language are easier to statically verify since the gaps between the tiers are covered in the language runtime directly. This runtime handles compilation to flow tables while including state and state updates. With this approach (where also all flows are proactively pushed to the switches), bugs can be found before the rules are executed on the switch. For verification, standard utilities like Alloy can be used because Flowlog programs are equivalent to first-order logic programs.

Aaron Gember (University of Wisconsin) asked whether the authors thought about supporting middlebox programming in their system, which they did not. Concerns were raised that this abstraction (with a SQL-like language) is too high-level; how would languages like Erlang fit into this project? In fact, the authors really thought about using a functional language instead of a custom DSL. Someone asked whether they could still run into inconsistencies between layers. Tim replied that they don't allow state to expire due to lack of use. They are adding the ability to have timeout events for removing rules. Aditya Akella (University of Wisconsin-Madison) asked about how the compiler works and whether it can handle all types of consistency semantics. Tim replied that Flowlog supports some things that OpenFlow does not. When their compiler detects forwarding rules, it produces netcore code and uses the netcore compiler. Andrew Ferguson (another author) pointed out that some parts of the policy might be directed toward a single switch, and that that is an abstraction introduced by netcore.

Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions using FlowTags

Seyed Kaveh Fayazbakhsh, Carnegie Mellon University; Luis Chiang, Deutsche Telekom Labs; Vyas Sekar, Carnegie Mellon University; Minlan Yu, University of Southern California; Jeffrey C. Mogul, Google

Seyed Kaveh Fayazbakhsh gave a short introduction to SDN and middleboxes and pointed out that integration of middleboxes in software-defined networks is a non-trivial task since middleboxes modify packets based on traffic dynamics. This behavior violates two key tenets of software-defined networking, namely origin binding and the paths follow policy. Origin binding means that there should be a strong binding between a packet and its origin, which is violated by NAT boxes. The paths follow policy states that explicit policies should determine the paths that packets follow, which Web proxy boxes violate. As a result, management, debugging, and forensics are more complicated, and, furthermore, it is difficult to ensure service-chaining policies.

Seyed explained that to solve this issue, it is necessary to add the missing contextual information that middleboxes otherwise may rewrite. The authors achieve this goal by adding tags to flows, where a central flow tag controller configures the tagging logic. Subsequently, critical information that middleboxes need is encoded in tags, and forwarding is performed based on tags not overwriting information like the source IP address. Through this central configuration, enhanced policies are possible and are expressed in dynamic policy graphs. These graphs hold transition conditions between middlebox nodes from sources to destinations, which makes chaining devices more easily feasible.

To implement this system, middleboxes need to be modified. This is possible because middleboxes already rewrite packets at a high level. That means that actually no changes to their internal logic are needed. The code to add in a middlebox is negligible (25–75 LOC). The processing overhead is less than 1%. Based on their evaluation, only 15 bits are needed to encode tags, which can be done in, for example, the IP-ID or the IPv6 flow label. Additionally, flow labels add enhanced semantics to a packet, which allows for extended analysis and debugging.

Questions were raised about how granular flow tags need to be. In the worst case, every flow tag corresponds to a transport layer flow. The flow tags also caused confusion around how exactly flow tags are possible in OpenFlow. Seyed said that because OpenFlow and most middleboxes support rewriting IP-ID and other possible headers, flow tags can easily be implemented, and that 15 bits for flow tags supports 70k-80k flows per second.

Closing Remarks

Summarized by Rik Farrow

USENIX Executive Director Casey Henderson closed the symposium by thanking the chairs and handing a bottle of sparkling wine to Ratul. Paul Barham (Microsoft Research) and Arvind Krishnamurty (University of Washington) will be the co-chairs for NSDI '15.

LISA14

Nov. 9–14, 2014 | Seattle

More Craft. Less Cruft.

Wednesday Keynote Speaker:

Ken Patchett, Director of Data Center Operations, Western region, Facebook

Thursday Keynote Speaker:

Gene Kim, former CTO and founder, Tripwire,
co-author of *The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win*

Closing Plenary:

Janet Vertesi, Princeton University

Featuring talks and training from:

Michael “Mikey” Dickerson
Caskey Dickson, Google
Garrett Honeycutt, LearnPuppet.com
Dinah McNutt, Google
Laura Thomson, Mozilla
James Turnbull, Docker
Avleen Vig, Etsy
Mandi Walls, Chef

**Full Program and Registration
Coming August 2014**

www.usenix.org/lisa14

u s e n i x

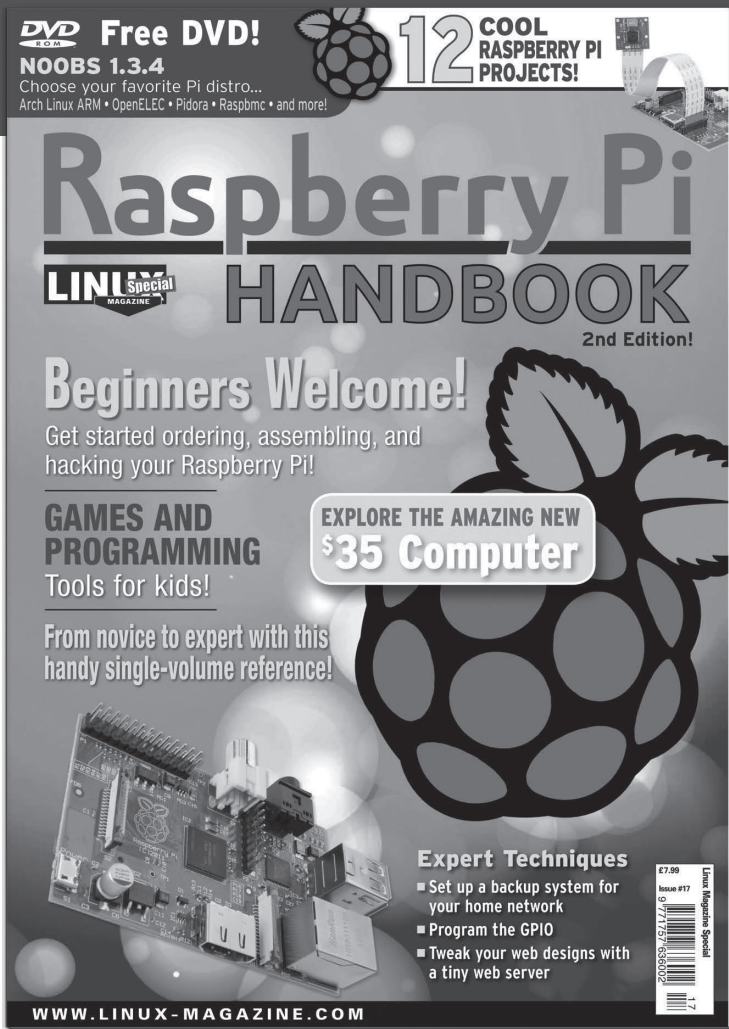
Shop the Shop

shop.linuxnewmedia.com

Raspberry Pi

HANDBOOK

2nd Edition!



DVD Free DVD!
NOOBS 1.3.4
Choose your favorite Pi distro...
Arch Linux ARM • OpenELEC • Pidora • Raspbmc • and more!

12 COOL RASPBERRY PI PROJECTS!

Raspberry Pi

LINIX Special **HANDBOOK**
2nd Edition!

Beginners Welcome!

Get started ordering, assembling, and hacking your Raspberry Pi!

GAMES AND PROGRAMMING
Tools for kids!

From novice to expert with this handy single-volume reference!

EXPLORE THE AMAZING NEW \$35 Computer

Expert Techniques

- Set up a backup system for your home network
- Program the GPIO
- Tweak your web designs with a tiny web server

WWW.LINUX-MAGAZINE.COM

Linux Magazine Special
Issue #17
9 771 91 830002
£7.99

Your companion for a strange and wonderful adventure...

**You ordered your Raspberry Pi...
You got it to boot...what now?**

The Raspberry Pi Handbook takes you through an inspiring collection of projects. Put your Pi to work as a:

- media center
- photo server
- game server
- hardware controller
- and much more!

Discover Raspberry Pi's special tools for teaching kids about programming and electronics, and explore advanced techniques for controlling Arduino systems and coding GPIO interrupts.

**WATCH YOUR NEWSSTANDS FOR
THE ONLY RASPBERRY PI REFERENCE
YOU'LL EVER NEED!**



**RASPBERRY PI ON NEWSSTANDS NOW OR ORDER ONLINE AT:
shop.linuxnewmedia.com/rpi**



USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER

Send Address Changes to *login*:
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE

PAID

AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

OSDI 14

11th USENIX Symposium
on Operating Systems Design
and Implementation

October 6–8, 2014 • Broomfield, CO

Join us in Broomfield, CO, October 6–8, 2014, for the **11th USENIX Symposium on Operating Systems Design and Implementation (OSDI '14)**. The Symposium brings together professionals from academic and industrial backgrounds in what has become a premier forum for discussing the design, implementation, and implications of systems software.

Don't miss the co-located workshops on Sunday, October 5

- **Diversity '14:** 2014 Workshop on Supporting Diversity in Systems Research
- **HotDep '14:** 10th Workshop on Hot Topics in Dependable Systems
- **HotPower '14:** 6th Workshop on Power-Aware Computing and Systems
- **INFLOW '14:** 2nd Workshop on Interactions of NVM/Flash with Operating Systems and Workloads
- **TRIOS '14:** 2014 Conference on Timely Results in Operating Systems



www.usenix.org/osdi14