

usenix;login:

AUGUST 2012 VOL. 37, NO. 4

**The Future of Security: Criticality, Rejectionists,
Risk Tolerance**

DANIEL E. GEER, JR.

**Passwords for Both Mobile and Desktop Computers:
ObPwd for Firefox and Android**

MOHAMMAD MANNAN AND P.C. VAN OORSCHOT

**Mosh: A State-of-the-Art Good Old-Fashioned
Mobile Shell**

KEITH WINSTEIN AND HARI BALAKRISHNAN

Conference Reports from NSDI '12 and LEET '12

usenix ASSOCIATION

UPCOMING EVENTS

Eighth Workshop on Hot Topics in System Dependability (HotDep '12)

CO-LOCATED WITH OSDI '12

October 7, 2012, Hollywood, CA, USA
www.usenix.org/hotdep12

2012 Workshop on Power Aware Computing and Systems (HotPower '12)

CO-LOCATED WITH OSDI '12

October 7, 2012, Hollywood, CA, USA
www.usenix.org/hotpower12

2012 Workshop on Managing Systems Automatically and Dynamically (MAD '12)

CO-LOCATED WITH OSDI '12

October 7, 2012, Hollywood, CA, USA
www.usenix.org/mad12

10th USENIX Symposium on Operating Systems Design and Implementation (OSDI '12)

October 8–10, 2012, Hollywood, CA, USA
www.usenix.org/osdi12

ACM/IFIP/USENIX 13th International Conference on Middleware (Middleware 2012)

December 3–7, 2012, Montreal, Quebec, Canada
<http://middleware2012.cs.mcgill.ca/doku.php>

26th Large Installation System Administration Conference (LISA '12)

December 9–14, 2012, San Diego, CA, USA
www.usenix.org/lisa12

11th USENIX Conference on File and Storage Technologies (FAST '13)

SPONSORED BY USENIX IN COOPERATION WITH ACM SIGOPS

February 12–14, 2013, San Jose, CA, USA
www.usenix.org/fast13
Submissions due September 18, 2012

10th USENIX Symposium on Networked Systems Design and Implementation (NSDI '13)

SPONSORED BY USENIX IN COOPERATION WITH ACM SIGCOMM AND ACM SIGOPS

April 3–5, 2013, Lombard, IL, USA
www.usenix.org/nsdi13
Paper titles and abstracts due September 12, 2012

14th Workshop on Hot Topics in Operating Systems (HotOS XIV)

May 13–15, 2013, Santa Ana Pueblo, NM, USA
www.usenix.org/hotos13
Submissions due January 10, 2013

2013 USENIX Federated Conferences Week

June 25–28, 2013, San Jose, CA, USA

2013 USENIX Annual Technical Conference (USENIX ATC '13)

June 26–28, 2013, San Jose, CA, USA

EDITOR

Rik Farrow
rik@usenix.org

MANAGING EDITOR

Jane-Ellen Long
jel@usenix.org

COPY EDITOR

Steve Gilmartin
proofshop@usenix.org

PRODUCTION

Arnold Gatilao
Jane-Ellen Long
Michele Nelson

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 nonmembers are \$125 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.

©2012 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.

usenix;login:

AUGUST 2012, VOL. 37, NO. 4

OPINION

Musings RIK FARROW2

SECURITY

ShareMeNot: Balancing Privacy and Functionality of Third-Party Social Widgets FRANZISKA ROESNER, CHRISTOPHER ROVILLOS, TADAYOSHI KOHNO, AND DAVID WETHERALL.....9

Mosh: A State-of-the-Art Good Old-Fashioned Mobile Shell KEITH WINSTEIN AND HARI BALAKRISHNAN20

Passwords for Both Mobile and Desktop Computers: ObPwD for Firefox and Android MOHAMMAD MANNAN AND P.C. VAN OORSCHOT28

Looking Within to Improve American Cybersecurity RICHARD F. FORNO38

The Future of Security: Criticality, Rejectionists, Risk Tolerance DANIEL E. GEER, JR...42

NETWORKED SYSTEMS

Fast and Interactive Analytics over Hadoop Data with Spark MATEI ZAHARIA, MOSHARAF CHOWDHURY, TATHAGATA DAS, ANKUR DAVE, JUSTIN MA, MURPHY MCCAULEY, MICHAEL J. FRANKLIN, SCOTT SHENKER, AND ION STOICA45

SYSADMIN

When Disasters Collide: A Many-Fanged Tale of Woe, Coincidence, Patience, and Stress, Continued ... DOUG HUGHES52

COLUMNS

Practical Perl Tools: Taking the XPath Less Traveled DAVID N. BLANK-EDELMAN.....59

Import That! DAVID BEAZLEY.....66

iVoyeur: The Gift of Fire DAVE JOSEPHSEN76

/dev/random: Apps for Saps ROBERT G. FERRELL82

BOOKS

Book Reviews ELIZABETH ZWICKY, WITH PETER GUTMANN, SAM STOVER, AND MARK LAMOURINE84

USENIX NOTES

Farewell, Jane-Ellen CASEY HENDERSON.....90

Welcome, Rikki CASEY HENDERSON91

USENIX, Open Access, and You ANNE DICKISON91

USENIX Association Financial Statements for 2011 ANNE DICKISON AND CASEY HENDERSON91

CONFERENCES

9th USENIX Symposium on Networked Systems Design and Implementation (NSDI '12).....94

5th USENIX Workshop on Large-Scale Exploits and Emergent Threats: Botnets, Spyware, Worms, New Emerging Threats, and More (LEET '12)114

Musings

RIK FARROW



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

Once a year, the local Mac users group asks me to talk to them about Mac OS X security. And every year, I search for something exciting and recent, something that I can use to motivate Mac users into being concerned about their security. In April of 2012, I had lots of fodder for a good talk.

Mac users tend to be complacent about their security. They know that Windows users are the predominant target of malware. During the LEET workshop, Tudor Dumitraş of Symantec Research Labs shared information from their Worldwide Intelligence Network Environment, a large dataset collected from millions of Windows hosts. Tudor had little to say about Macs; the amount of malware collected was small, as were the number of Mac users with antivirus and intrusion protection installed.

But does this mean that Macs are inherently more secure? Or is it simply a case of there being many fewer targets, and thus the much greater focus on Windows systems?

The Numbers

Four years ago, Adam O'Donnell wrote an article for *IEEE Security and Privacy* [1] in which he outlined a game theory approach for why Macs have not been targeted by malware writers. Game theory uses simple mathematical equations to calculate the best strategies for players, in this case, malware writers and Mac users, and the associated payoffs. In 2008, O'Donnell calculated that the Macs would have to make up 16.6% of the total number of client systems before they would become an attractive target for malware writers. With Mac adoption at about 11%, Mac users should have been safe for a while longer [2].

The arrival of the Flashback trojan, which infected around 700,000 Macs over a short period of time in April 2012, was certainly a rude awakening [3]. Flashback (which Kaspersky calls Flashfake) used a vulnerability in Apple's version of Java (other versions of Java had been patched in January 2012) to install a trojan downloader. The attack used social engineering to get people to run the Java applet, by posing as a method for updating Adobe Flash player—itsself a notorious source of infection in 2011. If the user starts the applet, which could be part of a Web page or an email, the applet installs the trojan, executes it, and the trojan contacts its C&C (command and control) server—in other words, exactly like a Windows bot.

Apple responded quickly (for Apple) with a patch to its Java implementation. Within a week, Apple produced two more patches, actually disabling the execution

of Java besides updating their JVM. Their first patch would also remove Flashback, which I thought was a nice touch.

But what happened to Apple's "more secure than Windows" veil? Newer Windows versions (since Vista) have included security features designed to prevent exploits from succeeding (NX, DEP, and ASLR). If you read my summary of Tudor's LEET presentation (in this issue) or watch his presentation online, you will learn that these defensive measures have done little to stem the flood of malware. Just as O'Donnell implies, the platforms with the greatest numbers also attract the most viruses and attacks.

A Closer Look

So why was Flashback so successful? I too had hopes that UNIX-based operating systems, such as Mac OS X and Linux, would be less vulnerable to attacks like this. But this turned out to be the usual self-deception, where one has faith in something without there being anything but an illusion of security.

Flashback relies on the Java bug to escape the JVM's sandbox. It then downloads and installs the executable and arranges for it to execute every time the user logs in. As the user can write to his own directories, adding a file to `~/Library/StartupItems`, or to `~/Library/LaunchAgents/` [4] is something malware can easily do once a user has begun executing the malware's code. Flashback and SabPub [5], another Mac trojan, one used for spying rather than acting in a botnet, both add entries in the `StartupItems` directory. `launchctl` can easily work like `cron`, and arrange for execution of a program on user login, routinely (every n seconds) after the user has logged in, or at set times. Neither approach requires administrator privilege.

As for locking these methods down, I don't think you can prevent them. `~/Library/StartupItems/` is a directory (folder if you like) owned by the user, and changing the owner of that directory to root won't stop an attacker working as the owner of the parent directory. I didn't see a way to configure `launchd` [6] that prevents ordinary users from taking advantage of this useful (and yet dangerous) tool. At least in Linux, you can control which users get to use `crontab -e` with `/etc/cron.allow` or `/etc/cron.deny`. But Linux users will soon be getting something like `launchd` in the form of `upstart` [7], a replacement for the `init` daemon, which will not only facilitate faster booting but also eventually replace `cron`. We will have to see if `upstart` can be locked down so only root or sudoers can use it.

Mac and *nix users still have one advantage over Windows users when it comes to client security. If Windows users opt to include Administrator privileges with their user account, this privilege can be used by exploits as well. For example, a Windows user with Administrator privilege can start sniffing network interfaces without being required to authenticate.

Mac and Linux users, on the other hand, get prompted to enter their password, or the root password in Linux, when updating or installing software, before they can perform privileged tasks. Sniffing the network, for example, running `tcpdump` requires that the user work as root. While requiring that the user provide a password, a form of authentication, is nice, I am sure that many users do this without considering or understanding, the implications of doing so.

As a proof of concept, consider the dialog box shown in Figure 1. I began my Mac security talk in 2011 with this illustration, which I left on the screen for 15 minutes, while I waited to begin. I then asked members of the audience if they noticed

anything unusual about the dialog box. I asked the people who I knew did Mac consulting if it alarmed them. Eventually they noticed the poor English, but not that “Cancel,” customarily found in such dialog boxes, had been replaced with “Abort.” This example was taken from actual Mac malware.

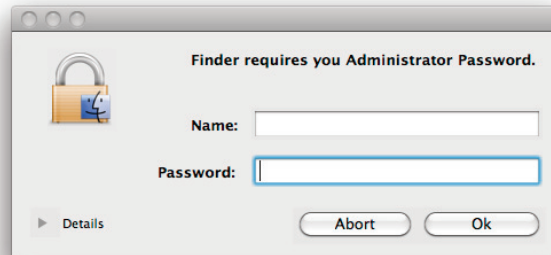


Figure 1: A dialog box found on Mac OS X systems that is requesting an administrator’s password

The SabPub [5] Mac malware I mentioned earlier is a backdoor that is being used in directed attacks. That is, Mac users are singled out, targeted with email that uses social engineering to get them to execute and install the malware, often just by clicking on a link or opening a Microsoft Office Word or Excel file [8].

While I was working on my Mac user group presentation [9], I asked many security friends what they thought about Mac security, knowing that most of them used Macs (laptops, desktops, and/or iPhones). Two people told me that they knew that there is an active, but underground, market in Mac-specific malware, but neither would go on the record. I know these people are not just pulling my leg. So if you are using a Mac, secure in the knowledge that you are safe from targeted attacks, I suggest you adjust your belief system.

Defense

I also wondered what this group of security geeks thought about using antivirus for Macs. Most do not use AV on their Macs. One salient reason is that the market for Mac AV is so small that there is little focus on it by AV vendors. Another person, in a position to know, said that an AV product developed for Linux had sold only six copies before it was discontinued. I am not saying this to suggest that Linux or Macs are immune to malware, just that current AV is not going to be much help.

Currently, iOS has stronger security than Mac OS X. Apple has published a 20-page document [10] describing iOS security features that sound very good to me. They are not perfect, but actually enforcing the use of ASLR on built-in and purchased apps, the use of a sandbox, the use of signed “entitlements” for access to resources, all appear to be evidence of good security design.

Some versions of Linux include SELinux. I’ve been using SELinux for years and have taught classes about it for the past three years. I recently had to check if my new CentOS 6.2 install had it enabled, as I have had no problems with SELinux at all. SELinux helps sandbox services as well as applications that run with privileges, like set-user-ID programs. With the sandbox command, you can run Web browsers in isolated environments that disappear when you exit. What you cannot

easily do is write policy that you can trust to be complete for your own applications. Writing policy is just too complex.

Speaking of Web browsers, it is important to keep in mind that any extension you install in your browser becomes part of the browser, its “chrome.” While extensions have some limitations, the software you have added has access to everything you download, just as the default install of the browser does. It’s this feature that makes it possible for NoScript to block JavaScript, and Ghostery to tell you about Web bugs, ad networks, and other “invisible” stuff added to Web pages and used to track you. It also makes it possible for malware that gets installed as an extension to modify what you see and to collect the information you enter when using your browser. The article by Franzi Roesner in this issue explains how their extension, which controls third-party tracking, takes advantage of rewriting the pages you view.

Your Web browser, and the ability of your mailtool to parse the same set of protocols, is what has made client systems so vulnerable to attacks. If we were still using plain old text, the Mac attacks, and most Windows attacks, would not be possible. Not that I expect anyone to give up the advances we’ve seen in pretty displays over the past 17 years, but it has made us vulnerable. Go back to text-only days, and the most comparable attack would be the inclusion of vi or emacs macros at the beginning of a text file. Today, it is common for image protocol parsers, such as JPEG and PNG, to include exploitable vulnerabilities. I suggest reading the summary of Ulfar Erlingsson’s LEET talk, or the associated paper [11], if you want to learn how Google is working at reducing the risk associated with the many protocols we use today.

The Lineup

Franzi Roesner and her co-authors start us off with research into just what your browser can do for (or to) you. Roesner has researched the various ways your browsing habits can be tracked, using an assortment of cute technology, such as single-pixel iFrames. While many browsers have an option to block third-party trackers, these measures are ineffective against tracking by social media sites. Ever wondered how Facebook “Like” or Google “+1” buttons know who you are? Roesner et al. don’t just explain how these trackers work, they also present an extension that fits into Firefox and Chrome that neuters these buttons—until you want to use them. Roesner’s paper was presented at NSDI ’12, and a summary of her presentation appears in this issue as well.

Keith Winstein and Hari Balakrishnan write about Mosh, a mobile shell application that is actually designed for mobility. SSH uses TCP, and your connections die if your source IP address changes. Also, TCP handles bufferbloat and high latency connections poorly. Mosh leverages SSH for its ability to set up and authenticate a connection, and then takes over, using AES encryption for your data and UDP for transmission. Mosh maintains virtual screens at both the client and the server, and provides local echo to clients using a conservative approach, but one that makes high latency connections much more pleasant to use. Keith presented this paper at USENIX Annual Tech in Boston, and that summary will appear in the October 2012 issue of *login*.

Mohammed Mannan and Paul van Oorschot examine the use of password tools for mobile devices. They evaluate their own tool, ObPwd, and show how it can provide strong passwords while simplifying the entering of passwords for mobile users.

ObPwd also works for desktop users, allowing users to share the same mechanism and secret token (an image or file) on both types of systems.

Rick Forno takes us in a different direction. Rick has been a long-time security professional, a Washington D.C. observer, and is now the director of the cybersecurity program at the University of Maryland, Baltimore Campus. Rick presents the history of failed security policies that have led us to this moment in US history and demonstrates how these proposals have remained essentially the same since 1996. He then makes three suggestions of his own for improving the stance of US security policy.

Dan Geer heads in yet another direction, with a different look at policy. Dan lets me read the text of speeches he makes, and when I read one of his latest, I contacted him immediately and asked him to write for this issue. What caught my attention this time was Dan's concept of the role of the Internet rejectionist, and how important this is for creating an infrastructure that is robust enough to withstand and quickly recover from failures or large-scale attacks using the Internet.

Matei Zaharia and co-authors, winners of the NSDI '12 Best Paper award, present Spark, an interface for distributed data modeling. Spark borrows some concepts from DryadLINQ, but is open source, and already in use outside the lab where it was developed. Spark presents a rich language for interaction with large datasets via Hadoop and HDFS, but also introduces the concept of *resilient distributed datasets* (RDDs). While using Spark, you can persist the result of filtering or manipulating large datasets in memory and speed up applications on Hadoop by as much as 40 times. RDDs are stored in memory and not replicated—if they are lost, a graph of transformations is used to recover the data. You can find out more about Spark on its Web site, <http://www.spark-project.org>, and by reading the paper presented at NSDI '12 and the summary in this issue.

Doug Hughes presents the second part of the story he began in the June 2012 issue. In that issue, Doug hinted at a near disaster involving two racks of drives storing 640 TB that started failing. Doug explains how the combinations of failures managed to overcome a level of redundancy they had felt was surely sufficient. Imagine having the loss of vast amounts of data hanging over you, not for days, but for longer than a week. Doug also provides lessons learned, not just in debugging but also in the design of redundant storage.

David Blank-Edelman almost departs from teaching us about Perl to explain XPath. XPath provides a common syntax for addressing portions of XML documents. XML documents can be viewed as trees, with a single root and many children, and XPath allows you to efficiently specify and extract elements of those trees. In the end, David shows us how to use XPath with Perl libraries.

Dave Josephsen continues his series on tools, written by Mathias Kettner, that work with Nagios. In this column, Dave waxes enthusiastic over Livestatus, a tool that presents Nagio's structures via a UNIX socket, using a simple API. Dave explains how Livestatus is superior to the other commonly used methods of extracting status information from Nagios, as well as demonstrating the use of Livestatus.

Dave Beazley explains how the import statement works in Python. A lot is happening behind the scenes when you import library modules and third-party extensions, and Dave explains how Python searches for these libraries. He also explains how you can vary the searchpath, how the searchpath varies for different versions of Python, and what different types of files may be found using import.

Robert Ferrell is excited about yet-to-be written, but oh so necessary, apps. You might think that with hundreds of thousands of apps already written, there could only be tiny niches left, but Robert's imagination takes us to places few dare to go.

We have a nice batch of book reviews in this issue. Elizabeth Zwicky starts with a book on analyzing big data, covers two books on Agile programming, then a book on building free software communities. She finishes up this set of five reviews with a book on domain modeling. Peter Gutmann tackles a legal tome that covers electronic signatures. The author has left few stones unturned: while not exactly beginning with the Stone Age, he does go back to the Magna Carta. Sam Stover writes an enthusiastic review of Metasploit, a book on the penetration testing framework. Metasploit itself is vast, but Sam tells us that this book carefully relates both how Metasploit fits into pen testing and also how Metasploit is used. Finally, Mark Lamourine reviews three books by Allen Downey, designed to teach students (and the self-taught) how to think like scientists. Downey uses examples in Python to teach problem solving, use of statistics, and computation modeling.

Finally, we have two reports, for NSDI '12 and LEET '12. There were many interesting papers at NSDI this year, and LEET combined academic and commercial research for a very interesting workshop, one that has been one of my personal favorites.

In 2006, I decided that I needed to go out into the world and point out just how badly the various security mechanisms that had been created, starting with MULTICS, were working. The point of my "Security Is Broken" talk [12] was not to beat a dead horse, but to encourage researchers and developers to try some different approaches. I ended my talk by suggesting that capabilities seemed like a fruitful direction, even though it had been explored in the past.

Today, we have capability-like features in iOS [10], as well as Robert Watson's Capsicum [13], now part of FreeBSD 9, and perhaps soon to appear in other OSes. The difference between Capsicum and approaches like SELinux is that the security policy of an application appears within the source of the application, instead of being stored and managed separately. Using Capsicum forces the developer to consider first what resources are needed, then to arrange for those capabilities to be available to an application running with reduced privileges, the set of allowed capabilities.

This still doesn't deal with the basic problem we have today in many of our applications. Web browsers operate with our own user privileges these days, and that allows reading and writing anything that we can. Even a browser or email tool that is effectively sandboxed can still be used to pass a Word document to an exploitable version of Office, and to silently do other things we don't want. We are not used to working in the type of compartmentalized systems that will work best with a capabilities-based system. Similar compartmentalized workstations were tried in the late '80s and were never easy to use.

Consider the LAMP model: Apache, MySQL, and PHP all running over Linux. That's an enormous software stack, all based on having a flexible Web server that can be remotely changed. Wordpress, as an awful example, recommends that you allow Apache write permission to all of its files, for purposes of administration (installing modules and updates). You can download PHP shell from SourceForge, something I imagine gets installed in lots of writeable Web server directories.

As I prepared for my Mac users group talk, I could tell from my email exchanges with the organizers just how disturbed they were. Someone had written software that had *silently* infected 700,000 of Macs—perhaps their very own Macs. The security paradigm we have inherited from MULTICS, that users need to be isolated from one another, does not fit the security problems we face today. Today, via our Web browsers, we execute code and interpret complex files that are provided to us by third parties that are usually unknown to us, in our own user context. So it is no surprise to me that malware still gets installed, when the very systems we use does that by design.

References

- [1] Adam O'Donnell, "When Malware Attacks (Anything but Windows)," *IEEE Security and Privacy*, 2008: <http://www.securitymetrics.org/content/attach/Metricon3.0/j3attAO.pdf>.
- [2] Andy Greenberg, "Cybercrime Game Theory: Why Apple's Malware Grace Period Ended Early," *Forbes*, April 20, 2012: <http://www.forbes.com/sites/andygreenberg/2012/04/20/cybercrime-game-theory-why-apples-malware-grace-period-ended-early/>.
- [3] Igor Soumenkov, "Flashfake Mac OS X Botnet Confirmed," Kaspersky Lab, April 6, 2012: http://www.securelist.com/en/blog/208193441/Flashfake_Mac_OS_X_botnet_confirmed.
- [4] David Lanier, "Cron and launchctl on Mac OS X 10.5 Leopard": <http://www.davidlanier.com/story/cron-and-launchctl-on-mac-os-x-105-leopard>.
- [5] Costin Raiu, "SabPub Mac OS X Backdoor: Java Exploits, Targeted Attacks and Possible APT Link": http://www.securelist.com/en/blog/208193467/SabPub_Mac_OS_X_Backdoor_Java_Exploits_Targeted_Attacks_and_Possible_APT_link.
- [6] Mac OS X Developer Library, launchd.plist(5): http://developer.apple.com/library/mac/#documentation/Darwin/Reference/ManPages/man5/launchd.plist.5.html#//apple_ref/doc/man/5/launchd.plist.
- [7] "What Is Upstart?" *Linux*, Jan. 6, 2011: <http://www.linux-magazine.com/Online/News/What-is-Upstart>; <http://upstart.ubuntu.com/>.
- [8] Dennis Fisher, "MacControl Trojan Being Used in Targeted Attacks Against OS X Users," *ThreatPost*, March 28, 2012: http://threatpost.com/en_us/blogs/maccontrol-trojan-being-used-targeted-attacks-against-os-x-users-032812.
- [9] Rik Farrow, "Malware: No One Is Safe (Mac malware presentation for Mac users):" <http://www.rikfarrow.com/malware2012.pdf>.
- [10] *iOS Security*, May 2012: http://images.apple.com/ipad/business/docs/iOS_Security_May12.pdf.
- [11] Mike Samuel and Ulfar Erlingsson, "Parse to Prevent Pwnage": <https://www.usenix.org/system/files/conference/leet12/samuel.pdf>.
- [12] Rik Farrow, "Security Is Broken": <http://video.google.com/videoplay?docid=1762847950860111011>.
- [13] Robert Watson et al., "Capsicum, Practical Capabilities for UNIX": http://www.usenix.org/event/sec10/tech/full_papers/Watson.pdf.

ShareMeNot

Balancing Privacy and Functionality of Third-Party Social Widgets

FRANZISKA ROESNER, CHRISTOPHER ROVILLOS, TADAYOSHI KOHNO, AND DAVID WETHERALL



Franzi Roesner is a fourth-year PhD student in Computer Science and Engineering at the University of Washington, where she is advised by Yoshi Kohno. Her research focuses on helping users better understand and control how their data is used and shared in a range of scenarios, from third-party tracking on the Web to permission granting for applications in modern operating systems (such as smartphones). She received her BS in Computer Science from the University of Texas at Austin and has done internships at Amazon.com, Microsoft Research, and Google. franzi@cs.washington.edu



Chris Rovillos is a second-year undergraduate student at the University of Washington. He is studying computer engineering and human-centered design and engineering. crovillo@cs.washington.edu



Tadayoshi Kohno is an Associate Professor in the University of Washington's Department of Computer Science and Engineering and an Adjunct Associate Professor in the UW Information School. Kohno received his PhD from the University of California at San Diego and his BS from the University of Colorado. His research focuses on helping protect the security, privacy, and safety of users of current and future technologies. yoshi@cs.washington.edu



David Wetherall is a Professor of Computer Science & Engineering at the University of Washington and led Intel's former Seattle research lab from 2006 to 2009. Wetherall received his PhD in computer science from MIT and his BE in electrical engineering from the University

of Western Australia. His research interests are focused on network systems, especially wireless networks and mobile computing, and Internet measurement and protocol design. He is known for pioneering research on programmable networks, Internet mapping, network de-duplication, and denial-of-service. djw@cs.washington.edu

Web tracking, the practice by which Web sites identify and collect information about users, generally in the form of some subset of Web browsing history, has become a topic of increased public debate. In this article, we summarize what we have learned about the Web tracking ecosystem and describe a taxonomy for understanding Web tracking behavior. In particular, we found that no existing browser mechanisms prevent tracking by social media sites via widgets (such as the Facebook "Like" button) while still allowing those widgets to achieve their utility goals. We then describe ShareMeNot, a browser extension that we developed to balance privacy with the intended functionality of social widgets.

Social widgets like the ones shown in Figure 1 allow their providers (Facebook, Google, Twitter, and others) to track a user's online browsing activities on every site that includes one of these buttons. This tracking is possible even if users never interact with the widgets and (in most browsers) even if users employ common defenses for third-party tracking, such as disabling third-party cookies.

To close the gap in defenses available to users, we introduce ShareMeNot. ShareMeNot is a browser extension (for Firefox and for Chrome) that aims to find a middle ground between allowing social widgets to track users wherever they appear and retaining the functionality of these widgets when users explicitly choose to interact with them (e.g., to "like" or to "tweet" a page). ShareMeNot for Firefox works by removing the browser cookies attached to requests made while loading the buttons, and ShareMeNot for Chrome works by replacing the buttons entirely with local replacement buttons. When users choose to click on a button, ShareMeNot necessarily allows the widget provider to identify that user to carry out the widget's functionality.

In this article, we provide background on how Web tracking works and summarize the taxonomy of tracking behavior that we introduced in our recent paper [8]. We then motivate ShareMeNot by assessing the effectiveness of defenses currently available to users and describe in detail its functionality and effectiveness.



Figure 1: Example social widgets. Social media sites expose social widgets that can be used to track users across all the sites on which these widgets are embedded. We refer to this type of tracking behavior as “personal tracking,” as users visit these Web sites directly during their normal browsing behavior and are often logged in and thus are not anonymous to these trackers.

What Is Web Tracking?

Third-party Web tracking refers to the practice by which an entity (the tracker), other than the Web site directly visited by the user (the site), tracks or assists in tracking the user’s visit to the site. For instance, if a user visits *cnn.com*, a third-party tracker like *doubleclick.net* embedded by *cnn.com* to provide, for example, targeted advertising can log the user’s visit to *cnn.com*. For most types of third-party tracking, the tracker will be able to link the user’s visit to *cnn.com* with the user’s visit to other sites on which the tracker is also embedded. We refer to the resulting set of sites as the tracker’s *browsing profile* for that user. In this section, we briefly review necessary Web-related background and summarize the taxonomy introduced in our recent paper [8].

Category	Name	Profile Scope	Summary	Example	Visit Directly?
A	Analytics	Within-Site	Serves as third-party analytics engine for sites.	Google Analytics	No
B	Vanilla	Cross-Site	Uses third-party storage to track users across sites.	DoubleClick	No
C	Forced	Cross-Site	Forces user to visit directly (e.g., via popup or redirect).	InsightExpress	Yes (forced)
D	Referred	Cross-Site	Relies on a B, C, or E tracker to leak unique identifiers.	Invite Media	No
E	Personal	Cross-Site	Visited directly by the user in other contexts.	Facebook	Yes

Table 1: Classification of tracking behavior. This table summarizes the taxonomy of tracking behavior that we developed in prior work [8]. Note that trackers may exhibit multiple behaviors at once.

Property	Behavior				
	A	B	C	D	E
Tracker sets site-owned (first-party) state.	✓				
Request to tracker leaks site-owned state.	✓				
Third-party request to tracker includes tracker-owned state.		✓	✓		✓
Tracker sets its state from third-party position; user never directly visits tracker.		✓			
Tracker forces user to visit it directly.			✓		
Relies on request from another B, C, or E tracker (not from the site itself).				✓	
User voluntarily visits tracker directly.					✓

Table 2: Tracking behavior by mechanism. In order for a tracker to be classified as having a particular behavior (A, B, C, D, or E), it must display the indicated property. Note that a particular tracker may exhibit more than one of these behaviors at once.

Web-Related Background

When a page is fetched by the browser, an HTTP request is made to the site for a URL in a new top-level execution context for that site (that corresponds to a user-visible window with a site title). The HTTP response contains resources of several kinds (HTML, scripts, images, stylesheets, iFrames, and others) that are processed for display and that may trigger HTTP requests for additional resources. Resources (such as iFrames) fetched from another domain and embedded on the page are known as *third-party content*.

Web tracking relies fundamentally on a Web site’s ability to store state on the user’s machine, as do most functions of today’s Web. Client-side state may take many forms—most commonly, traditional browser cookies. A *cookie* is a triple (domain, key, value) that is stored in the browser across page visits, where domain is a Web site, and key and value are opaque identifiers. Cookies that are set by the domain that the user visits directly (the domain displayed in the browser’s address bar) are known as *first-party cookies*; cookies that are set by some other domain embedded in the top-level page are *third-party cookies*.

Cookies are set either by scripts running in the page using an API call, or by HTTP responses that include a Set-Cookie header. The browser automatically attaches cookies for a domain to outgoing HTTP requests to that domain, using Cookie headers. Cookies may also be retrieved using an API call by scripts running in the page and then sent via any channel, such as part of an HTTP request (e.g., as part of the URL). The same-origin policy ensures that cookies (and other client-side state) set by one domain cannot be directly accessed by another domain.

Users may choose to block cookies via their browser’s settings menu. Blocking all cookies is uncommon, as it makes today’s Web almost unusable (e.g., the user cannot log into any account), but blocking third-party cookies is commonly recommended as a first line of defense against third-party tracking.

Background on Tracking

Web tracking is highly prevalent on the Web today. From the perspective of Web site owners and of trackers, it provides desirable functionality, including personalization, site analytics, and targeted advertising. From the perspective of a tracker, the larger a browsing profile it can gather about a user, the better service it can provide to its customers (the embedding Web sites) and to the user herself (e.g., in the form of personalization).

While some users may benefit from the results of this tracking, larger browsing profiles spell greater loss of privacy for users. A user may not, for instance, wish to link the articles he or she views on a news site with the type of adult sites he or she visits, much less reveal this information to an unknown third party. Even if the user is not worried about the particular third party, this data may later be revealed to unanticipated parties through court orders or subpoenas.

In our recent paper [8], we investigated tracking in the wild today and introduced a taxonomy of third-party Web tracking behavior. This taxonomy (summarized in Table 1) focuses on *explicit* tracking mechanisms, i.e., tracking mechanisms that use assigned, unique identifiers per user rather than *inferred* tracking based on browser and machine fingerprinting. Other work [9] has studied the use of fingerprinting to pinpoint a host with high accuracy.

More specifically, all trackers we considered have two key capabilities:

- ◆ The ability to store a pseudonym (unique identifier) on the user's machine.
- ◆ The ability to communicate that pseudonym, as well as visited sites, back to the tracker's domain.

The pseudonym may be stored using any client-side storage mechanism, including conventional browser cookies, HTML5 LocalStorage, Flash cookies, and others. Stored values are communicated to the tracker either automatically when the browser includes a cookie with a request, or explicitly by tracker-provided JavaScript code that accesses and transmits the stored values. Similarly, the browser may communicate information about the visited site to the tracker, either implicitly via the HTTP Referrer header, or explicitly via tracker-provided code using the `document.referrer` API call.

Depending on the mechanisms used by a tracker, the browsing profiles it compiles can be *within-site* or *cross-site*. Within-site browsing profiles link the user's browsing activity on one site with his or her other activity only on that site, including repeat visits and how the Web site is traversed, but not to visits to any other site. Cross-site browsing profiles link visits to multiple different Web sites to a given user (identified by a unique identifier or linked by another technique [6, 9]).

Our tracking taxonomy categorizes tracking behavior based on client-side observable mechanisms. It distinguishes between within-site and cross-site trackers, and it further distinguishes different types of cross-site trackers. This is in contrast to past work that considered business relationships between trackers and the embedding Web site [4] and past work that categorized trackers based on prevalence rather than user browsing profile size [5]. These distinctions are important, because they have different implications for how to detect and defend against the various behaviors.

We summarize the behavior types defined in our taxonomy in Table 1 and below. Table 2 captures the relationships of specific observable tracking mechanisms to

these behavioral categories. In order to fall into a particular behavior category, the tracker *must* exhibit (at least) all of the properties indicated for that category in Table 2. A single tracker may exhibit more than one of these behaviors.

- ◆ **A (Analytics):** The tracker serves as a third-party analytics engine for sites. It can only track users within sites.
- ◆ **B (Vanilla):** The tracker uses third-party storage that it can get and set only from a third-party position to track users across sites.
- ◆ **C (Forced):** The cross-site tracker forces users to visit its domain directly (e.g., popup, redirect), placing it in a first-party position.
- ◆ **D (Referred):** The tracker relies on a B, C, or E tracker to leak unique identifiers to it, rather than on its own client-side state, to track users across sites.
- ◆ **E (Personal):** The cross-site tracker is visited by the user directly in other contexts.

In the remainder of this article, we focus in more detail on personal tracking behavior.

Personal Trackers

Personal trackers are defined as those whose domains the user otherwise visits intentionally (e.g., facebook.com). Many of these sites, primarily social networking sites, expose social widgets such as the Facebook “Like” button, the Twitter “Tweet” button, the Google “+1” button and others (see Figure 1). These widgets can be included by Web sites to allow users logged in to these social networking sites to Like, Tweet, or +1 the embedding Web page. These widgets allow the corresponding tracker to create a cross-site browsing profile of a user across any Web sites that he or she visits that includes such a widget.

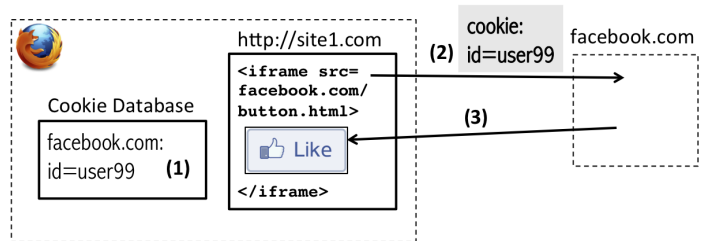


Figure 2: Personal tracking via social widgets. Social sites like Facebook, which users visit directly in other circumstances allowing the site to (1) set a cookie identifying the user, expose social widgets such as the “Like” button. When another Web site embeds such a button, the request to Facebook to render the button (2-3) includes Facebook’s cookie. This allows Facebook to track the user across any site that embeds such a button.

These buttons present a privacy risk for users because they track users even when they choose not to click on any of the buttons. Like traditional third-party tracking content, simply loading a social widget provides the tracker with sufficient information to create a cross-site browsing profile for the user. That is, cookies and referrer information are included with requests to load the widget, with no user interaction required.

Furthermore, because users are often logged into the Web sites that expose such widgets (e.g., Facebook or Google), this tracking may not be anonymous. These

trackers have the ability to link the cross-site browsing profiles they collect about users with the personal information users have entered directly into their accounts at those Web sites.

As an example, Figure 2 overviews the interaction between Facebook, a site embedding a “Like” button, and the user’s browser. The requests made to facebook.com to render this button allow Facebook to track the user across sites. Unlike vanilla tracking behavior, Facebook sets its cookie from a first-party position when the user voluntarily visits facebook.com. As a result, defenses like third-party cookie blocking are ineffective against personal trackers. In the next section, we explore the weaknesses of existing defenses available to users.

Existing Defenses Against Personal Trackers

We find that existing defenses against third-party Web tracking available to users today are not well suited to defend against personal trackers. In particular, existing defenses either fail to prevent personal tracking behavior or disable desired functionality (e.g., the user’s ability to “Like” a Web page and share it back to his or her Facebook account).

Third-party cookie blocking is insufficient for personal trackers, for a number of reasons. First, different browsers implement third-party cookie blocking with different degrees of strictness. While Firefox blocks third-party cookies from being *set* as well as from being *sent*, most other browsers (including Chrome, Safari, and Internet Explorer) only block the setting of third-party cookies. So, for example, Facebook can set a first-party cookie when the user visits facebook.com; in browsers other than Firefox, this cookie, once set, is available to Facebook from a third-party position (when embedded on another page).

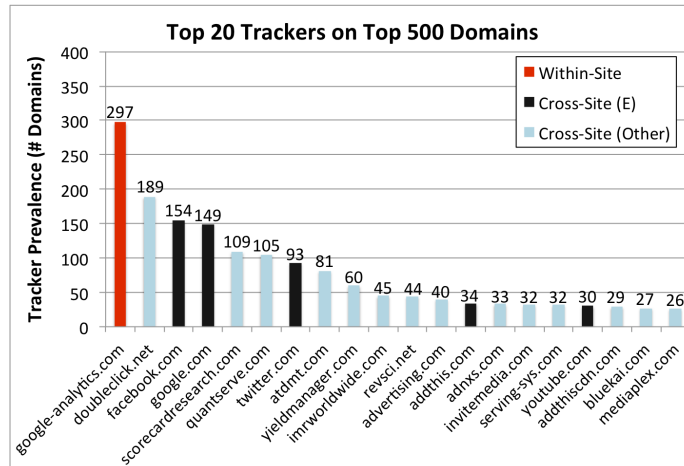


Figure 3: Prevalence of trackers on top 500 domains [8]. This graph shows the prevalence of the top 20 trackers on the Alexa top 500 domains from our 2011 measurement study with no defenses enabled. Compare to Figure 4, in which third-party cookies are blocked.

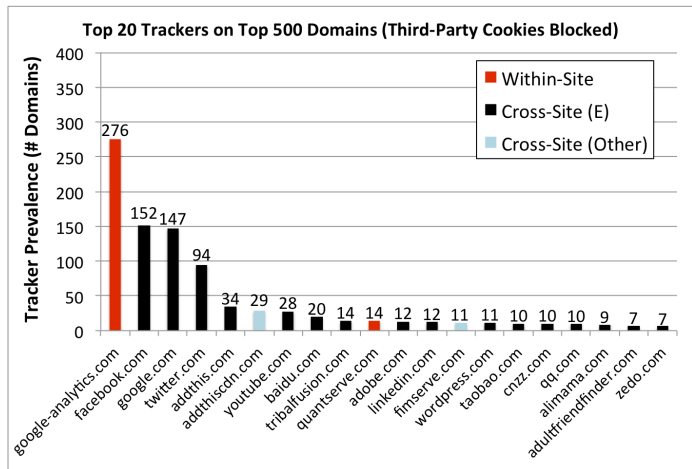


Figure 4: Prevalence of trackers on top 500 domains with third-party cookies blocked [8]. When third-party cookies are blocked, personal trackers dominate the set of top 20 trackers. Personal trackers are not affected by third-party cookie blocking, because users visit the trackers' Web sites directly, allowing them to set cookies from a first-party position.

Thus, in most browsers, third-party cookie blocking protects users only from trackers that are never visited directly. Figures 3 and 4 show data from our measurement study described in [8]. Notice that third-party cookie blocking is effective for many cross-site trackers, but it is ineffective for personal trackers, leaving them as the prominent remaining trackers when third-party cookies are blocked (Figure 4).

Firefox's strict policy provides better protection, but at the expense of functionality like social widgets and buttons (thus prompting Mozilla to opt against making this setting the default [7]).

The recently proposed Do Not Track header and legislation aim to give users a standardized way to opt out of Web tracking via a browser setting that appends a DNT=1 heading to outgoing requests. DNT has prompted a debate over the definition of tracking, as its conclusion determines to which parties the legislation will apply. Facebook, for instance, argues that personal tracking should not be subject to Do Not Track because users already have an explicit relationship with the tracker [3].

Users can attempt to minimize the size of the browsing profiles trackers can create by frequently clearing the client-side state that contains unique identifiers. However, when users log into the Web sites of personal trackers, the new identifier is by definition linked with the old identifier, as both are linked to the user's account on the tracker's Web site.

Logging out of a social media site may not prevent the tracking of a user, or prevent the linking of a user's browsing profile while they are not logged in with their browsing profile while logged in. Logging out of these sites often does not clear any or all cookies containing unique identifiers [1], allowing them to continue to be used for tracking.

Several possible defenses exist in the form of browser extensions that allow users to block trackers. These defenses, including NoScript (<http://noscript.net>), Ghostery (<http://www.ghostery.com>), and Disconnect (<http://disconnect.me>) (which targets personal trackers directly), work by simply blocking the tracker's scripts and their associated buttons from being loaded by the browser at all. This approach effectively removes the buttons from the user's Web experience entirely and thus removes potentially desired functionality.

ShareMeNot

We introduce the ShareMeNot browser extension to protect users from tracking by social widgets while still allowing these widgets to be used. The use of ShareMeNot shrinks the profile that the supported personal trackers can create to only those sites on which the user explicitly clicks on one of the buttons, at which point the button provider must necessarily know the user's identity in order to link the "Like" or the "+1" action to the user's profile. No other existing approach can shrink the profile a personal tracker can create while also retaining the functionality of the buttons, although concurrent work on the Priv3 Firefox add-on [2] adopts the same basic approach; as of May 2012, Priv3 supports fewer widgets and, to our knowledge, was not iteratively refined through measurement.

ShareMeNot supports social widgets from Facebook, Google, Twitter, AddThis, YouTube, LinkedIn, Digg, and Stumbleupon. We chose to support these sites based in part on our initial, pre-experimental perceptions of popular third-party trackers, and in part based on our experimental discovery of the top trackers.

ShareMeNot for Firefox

ShareMeNot for Firefox works by stripping cookies from third-party requests to any of the supported personal trackers that are made during the loading of a social widget. ShareMeNot strips cookies from two types of requests:

- ◆ *Requests to a tracker's domain that have another domain as the referrer.* Most requests made during the loading of a social widget fit this rule.
- ◆ *Specific blacklisted requests, of referrer.* This rule is necessary because of the complexity of some of the social widgets, which include multiple chained requests. For example, loading the Facebook "Like" button involves requests to facebook.com with the referrer facebook.com, rather than the embedding site. ShareMeNot's blacklist covers these requests.

When ShareMeNot detects that a user has clicked on a button by recognizing the characteristic request that follows a click on each supported widget, it allows the cookies to be included with the request. In most cases, this allows the button click to function as normal and as transparent to the user. The Facebook "Like" button is more complex, however, and must first be reloaded in the logged-in state to be active. Thus, a ShareMeNot user must click on this button twice: the first click will reload the button with personalized content (e.g., "5 of your friends have liked this") and the second will actually "like" the page.

ShareMeNot for Firefox does not fully block requests to the trackers, instead only removing cookies from the relevant requests. Thus, it may expose the user's IP address and other fingerprinting information that can be used for implicit tracking. It also does not block programmatic access to document.cookie, which would

allow personal trackers attempting to circumvent ShareMeNot to continue accessing cookie values. ShareMeNot for Chrome addresses these weaknesses.

ShareMeNot for Chrome

Unlike ShareMeNot for Firefox, ShareMeNot for Chrome blocks entire HTTP requests to tracker buttons. The buttons are replaced with locally stored versions of the buttons that offer the same functionality. ShareMeNot for Chrome works in two phases: first, blocking HTTP requests to tracker buttons, then inserting replacement buttons where the tracker buttons were to originally have been.

ShareMeNot leverages the newly introduced XMLHttpRequest API in Chrome to monitor HTTP requests before they are sent by the browser. When a user visits a Web page, the HTTP requests sent as the page is loaded are compared to a predefined set of URL patterns that identify requests for tracker buttons; if a URL matches a pattern, the entire HTTP request is blocked. In that case, ShareMeNot displays an icon in the Chrome location bar notifying the user, who can choose to unblock certain sites by clicking on the icon. To avoid impacting functionality when users directly visit social media sites such as Facebook, ShareMeNot does not block requests for top-level pages. Instead, it only blocks requests for resources that are requested by the page that is loading, such as scripts, images, and other Web pages embedded via iFrames.

In the second phase, ShareMeNot inserts replacement buttons. A Chrome extension content script is executed in the context of the current page after it has loaded. As the original widgets were blocked from loading in the first phase, the content script must search the page using a predefined set of CSS selectors for HTML tags or other clues about where the buttons should have been. It replaces them with iFrame elements that point to the replacement buttons stored within the extension. These replacement buttons either directly activate (for example, opening the appropriate Twitter sharing page if the user clicks on the “Tweet” button) or load the original button when clicked. For example, clicking on the replacement Facebook “Like” button loads the actual “Like” button; as in ShareMeNot for Firefox, the user must click twice to actually “like” the page. This is necessary because some social media sites don’t have direct links for button actions; the user must use that social media site’s real buttons.

By blocking all requests to tracker domains until users click on the replacement buttons, ShareMeNot for Chrome prevents the leakage of the user’s IP address and other fingerprinting information, as well as access to document.cookie. We hope to update ShareMeNot for Firefox to match this more privacy-preserving design in the future.

Effectiveness

We experimentally verified the effectiveness of ShareMeNot for Firefox (we expect similar results for ShareMeNot for Chrome). As summarized in Table 3, ShareMeNot dramatically reduces the presence of the personal trackers it supports to date. ShareMeNot entirely eliminates tracking by most of these, including Twitter, AddThis, YouTube, Digg, and Stumbleupon. While it does not entirely remove the presence of Facebook and Google, it reduces their prevalence to 9 and 15 occurrences, respectively. In the Facebook case, this is due to the Facebook comments widget, which triggers additional first-party requests (containing tracking infor-

mation) not blacklisted by ShareMeNot; the Google cases appear mostly on other Google domains (e.g., google.ca).

Tracker	<i>Without ShareMeNot</i>	<i>With ShareMeNot</i>
Facebook	154	9
Google	149	15
Twitter	93	0
AddThis	34	0
YouTube	30	0
LinkedIn	22	0
Digg	8	0
Stumbleupon	6	0

Table 3: Effectiveness of ShareMeNot. ShareMeNot drastically reduces the occurrences of tracking behavior by the supported set of personal trackers.

To date, ShareMeNot does not fully support the complete set of social widgets exposed by the supported trackers. Facebook, in particular, exposes a broader set of “social plugins” that ShareMeNot renders nonfunctional (because it does not properly activate them when users attempt to interact with them) and/or for which it does not properly prevent tracking (because it has an incomplete request blacklist). We hope to address these issues in future versions.

As of May 2012, we have seen over 25,000 downloads from our own servers (<http://sharemenot.cs.washington.edu/>), in addition to over 8,500 daily users as reported by the official Mozilla add-on site (<https://addons.mozilla.org/firefox/addon/sharemenot/>).

Conclusion

We have introduced ShareMeNot, a browser extension that protects users from personal tracking by third-party social widgets while retaining the functionality of these widgets should users wish to click on them. ShareMeNot can be downloaded from our Web site, <http://sharemenot.cs.washington.edu>.

Acknowledgments

We thank the readers and reviewers of earlier versions of our related NSDI paper [8] for their valuable feedback throughout this work: Jon Crowcroft, Daniel Halperin, Arvind Narayanan, and Charlie Reis. We thank Brandon Lucia for naming ShareMeNot. This work was supported in part by NSF Awards CNS-0722000, CNS-0846065, and CNS-0917341, an NSF Graduate Research Fellowship under Grant No. DGE-0718124, an Alfred P. Sloan Research Fellowship, and a gift from Google.

References

[1] N. Cubrilovic, “Logging Out of Facebook Is Not Enough,” 2011: <http://nikcub.appspot.com/posts/logging-out-of-facebook-is-not-enough>.

[2] M. Dhawan, C. Kreibich, and N. Weaver, "The Priv3 Firefox Extension," <http://priv3.icsi.berkeley.edu/>.

[3] Facebook, "Facebook's Position on 'Do Not Track,'" W3C Workshop on Web Tracking and User Privacy, 2011: <http://www.w3.org/2011/track-privacy/papers/Facebook.html>.

[4] C. Jackson, A. Bortz, D. Boneh, and J.C. Mitchell, "Protecting Browser State from Web Privacy Attacks," WWW 2006: <http://www2006.org/programme/item.php?id=3536>.

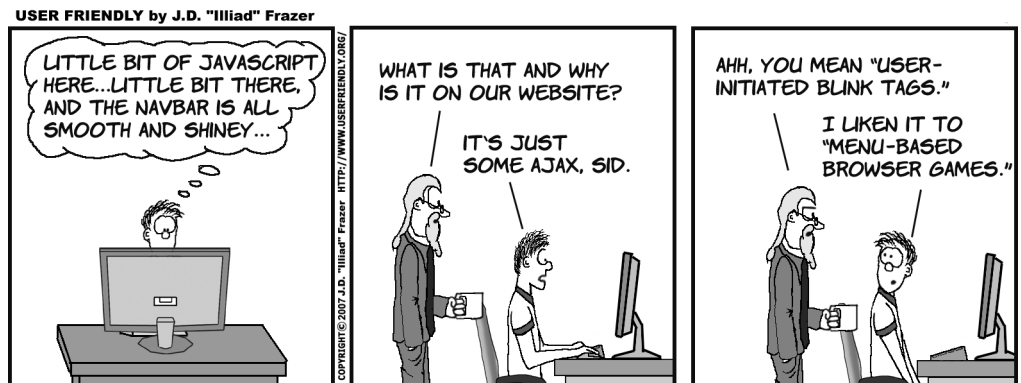
[5] B. Krishnamurthy and C. Wills, "Privacy Diffusion on the Web: A Longitudinal Perspective," WWW, 2009.

[6] B. Krishnamurthy, K. Naryshkin, and C. Wills, "Privacy Leakage vs. Protection Measures: The Growing Disconnect," Web 2.0 Security and Privacy, 2011: <http://www.w2spconf.com/2011/papers/privacyVsProtection.pdf>.

[7] Mozilla, "Bug 417800 Revert to Not Blocking Third-Party Cookies," 2008: https://bugzilla.mozilla.org/show_bug.cgi?id=417800.

[8] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and Defending against Third-Party Tracking on the Web," NSDI '12.

[9] T.-F. Yen, Y. Xie, F. Yu, R.P. Yu, and M. Abadi, "Host Fingerprinting and Tracking on the Web: Privacy and Security Implications," NDSS 2012.



Mosh

A State-of-the-Art Good Old-Fashioned Mobile Shell

KEITH WINSTEIN AND HARI BALAKRISHNAN



Keith Winstein is a doctoral student in electrical engineering and computer science at MIT, where he was named Claude E. Shannon Research Assistant. His work focuses on protocols to make today's Internet more efficient and equitable. Mr. Winstein was previously a staff reporter at *The Wall Street Journal*, covering science and medicine, and the vice president of business development at Ksplice Inc., a Linux software company now part of Oracle Corp.

keithw@mit.edu



Hari Balakrishnan is the Fujitsu Professor of Computer Science at MIT, working on networked computer systems, with current projects in mobile/wireless networking and cloud computing. His previous work includes the RON overlay network, the Chord DHT, the Cricket location system, the CarTel mobile sensing system, and cross-layer wireless protocols such as snoop TCP and SoftPHY. He is an ACM Fellow (2008), a Sloan Fellow (2002), and an ACM dissertation award winner (1998), and has received several Best Paper awards, including the IEEE Bennett prize (2004) and the ACM SIGCOMM Test of Time award (2011). He also co-founded StreamBase Systems, helped devise the key algorithms for Sandburst Corporation's (acquired by Broadcom) high-speed network QoS chipset, is an advisor to Meraki, and is on the Board of Trustees of IMDEA Networks (Spain).

hari@csail.mit.edu

Remote terminal applications are almost as old as packet-switched data networks. Starting with RFC 15 in 1969, protocols like Telnet, SUPDUP, and BSD's rlogin and rsh have played an important role in the Internet's development. The reader of *.login:* has undoubtedly used the most popular of these: the Secure Shell, or SSH, which since 1995 has ruled the wires. This article describes a successor to these venerable applications that was designed to work better in our increasingly mobile and wireless world.

SSH has two weaknesses that can make it unpleasant today. First, because SSH and previous remote terminals run over TCP, they don't support roaming between IP addresses or always preserve sessions when connectivity is intermittent. A laptop will happily suspend for a commute to work, but after wakeup its SSH sessions will have frozen or died.

Second, SSH operates strictly in character-at-a-time mode, with all echoes and line editing performed by the remote host. As a result, its interactive performance can be poor over wide-area wireless (e.g., EV-DO, UMTS, LTE) or transcontinental networks (e.g., to cloud computing facilities or remote datacenters), and sessions are almost unusable over paths with non-trivial packet loss. When loaded or when the signal-to-noise ratio is low, delays on many wireless networks reach several *seconds* because of deep packet queues ("bufferbloat") or over-zealous link-layer retransmissions. Many home networks also suffer from multi-second delays under load. Trying to type, or correct a typo, over such networks is unpleasant.

These problems affect users to varying degrees. For us after 15 years, they eventually bubbled over from the cauldron of smouldering frustration that produces software: in this case, Mosh, the mobile shell (<http://mosh.mit.edu>). Mosh is free and open-source software and is available for most major operating systems.

```
[mosh]
mosh: Last contact 10 seconds ago. [to quit: Ctrl-^ .]
* Benefits of Mosh
** Roam across Wi-Fi networks or to cell without dropping connection.
** More pleasant to type -- intelligent local echo is instant.
** No need to be superuser to install.
** Mosh doesn't fill up buffers, so Ctrl-C works quickly on runaways.
** Designed from scratch for Unicode; fixes bugs in SSH, other terminals.
** Free / open-source software.
-UU-:*--F1 All L19 (Org)-----
```

Figure 1: Mosh in use

Mosh fixes these issues. A user who switches IP addresses (e.g., from WiFi to cellular while leaving a building, or from home to work) keeps the connection without thinking about it. Ditto for suspend and resume. Mosh is careful about flow control and doesn't fill up network buffers: for example, "Control-C" works right away to halt output from a runaway process. Mosh reacts to packet loss intelligently.

In addition, the Mosh client runs a predictive model of the application in the background, and uses the model to do intelligent client-side echoing and line editing. According to data we have collected from contributing users, more than two-thirds of the keystrokes in a typical UNIX session can be displayed instantly with a conservative model of application behavior. Mosh's empirical approach to local echo works in full-screen programs like a text editor or mail reader as well as at the command line, and doesn't require a change to server-side software.

We announced Mosh by accident in April 2012, or, to be fair, somebody announced it for us with a post on Hacker News (<http://news.ycombinator.com>). Over the next 48 hours, Mosh's hurriedly completed Web page received more than 100,000 views, not an indication of merit, but an unexpected amount of attention for a UNIX system utility. Mosh has now been downloaded more than 70,000 times.

This level of interest may be because we scratched an itch: Mosh is a rare example of a gracefully mobile network application. Today, many programs intended for mobility, including email clients and Web browsers on popular smartphones, can't cope gracefully if the client switches network interfaces, roams, or has intermittent connectivity, the very conditions presented by mobile networks.

Here's our view of the ideas behind Mosh, with the hope that new applications could benefit from the same principles. We plan to investigate this hypothesis in future work.

Choose a Good Abstraction

Traditional remote-shell protocols such as Telnet, rlogin, and SSH work by reliably conveying a bytestream from the server to the client, to be interpreted by the client's terminal. Mosh works at a different layer and treats the remote terminal more like a videoconference. With Mosh, the server runs a terminal emulator, and the server and client each maintain a snapshot of the current screen state. The problem becomes one of state-synchronization: fast-forwarding the client to the *most recent* screen as efficiently as possible.

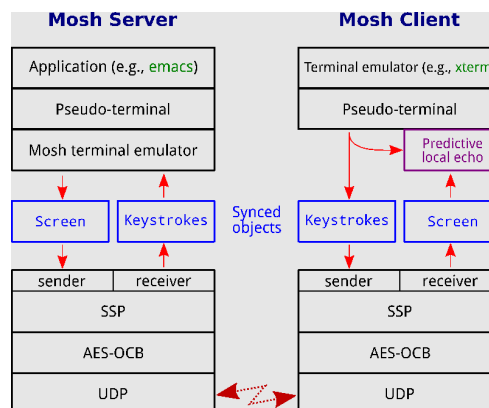


Figure 2: Mosh's design

This synchronization is accomplished using a new protocol we call the *State Synchronization Protocol* (SSP). SSP runs over UDP, synchronizing the state of an object from one host to another. Datagrams are encrypted and authenticated using AES-128 in the Offset Codebook mode [1], which provides confidentiality and authenticity with a single secret key.

Because SSP works at the object layer and can control the rate of synchronization (i.e., the frame rate), it does not need to send every byte it receives from the application. Mosh regulates the frames so as not to fill up network buffers, retaining the responsiveness of the connection. Mosh sets the minimum time between frames to half the smoothed round-trip time (RTT) of the path. By contrast, SSH doesn't know how the client will interpret each octet, and so must send everything the application generates.

A schematic of Mosh's design is shown in Figure 2. Mosh runs two copies of SSP, one in each direction of the connection. The server-side terminal emulator exports an object called the *Screen*, containing the contents of the terminal display. This is the object that SSP synchronizes to the client. Meanwhile, the client records the user's keystrokes in a verbatim transcript, and synchronizes this object to the server.

Why is TCP the wrong abstraction? TCP presents a reliable, in-order, bytestream abstraction and assumes continual connectivity between a pair of IP addresses. For applications like Mosh, this interface is problematic: In marginal conditions or after a period of disconnectivity, the server ought to try to “fast-forward” the client to the current screen state, not resend old data that has been queued. Even if the server application knows that much of the data queued up isn't useful, it is not possible to “pull back” stale data from a kernel socket buffer, much less from the network.

TCP doesn't allow intermittent connectivity and roaming. Previous work in this area, including proposals for TCP connection migration and Mobile IP, have not been widely deployed; TCP migration requires kernel modifications and Mobile IP third-party home agents, and neither supports intermittent connectivity. And TCP's minimum retransmission timeout is at least one second: fine for bulk transfers but not for human-generated interactive flows.

Idempotency for Security and Roaming

SSP is a novel secure-datagram protocol with a design considerably simpler than previous work. We agree that this statement is just cause for skepticism. It will take time for the security community to become comfortable with SSP; protocols like SSH, Kerberos, and TLS have had security holes and design weaknesses surface only after years of scrutiny. We didn't use Datagram TLS because it doesn't support roaming and requires the endpoints to generate public key pairs to authenticate each other.

The security of SSP rests on the principle of *idempotency*. Each datagram sent to the remote site is encrypted and authenticated with AES-OCB and represents an idempotent operation at the recipient, a “diff” between a numbered source and target state. The diff is a *logical* one: the object itself calculates the diff between itself and a future object of the same type, and an object on the other end of the connection “applies” the diff.

Field	Value	Explanation
type:	Screen	<i>type of object</i>
protocol version:	2	
source state:	#17	<i>what state diff is coming from</i>
target state:	#20	<i>will be created when diff is applied to source</i>
ack:	#6	<i>latest state sender has constructed from other side</i>
received ack:	#17	<i>latest state other side has constructed from sender</i>
contents:	"1st\r\n2nd ln"	
random chaff:	...	<i>frustrates packet-length analysis</i>

Table 1: Example "diff"

For example, a diff might tell the receiver how to get from frame #17 to frame #20. An attacker who repeats the datagram containing this diff, or changes the ordering of datagrams, won't compromise the security of the system.

Roaming becomes simple to accommodate. Every time the server receives an authentic datagram from the client with a sequence number greater than any before, it sets the packet's source IP address and UDP port number as its new target for future outgoing datagrams. As a result, client roaming happens without any timeouts or a notion of reconnection. The client doesn't even know (or need to know) it has roamed. This is helpful when the client is behind a network-address translator (NAT), and it is the NAT that has changed the public-facing IP addresses (common when "tethering" to a smartphone).

Make Each Packet Your Best Packet

SSP tries to wring the most benefit out of each packet and get the receiver to the current object state as efficiently as it can. The sender can formulate diffs between whatever pairs of states it thinks make for the swiftest way to accomplish that goal. For lossy links, one technique we have developed is the *prophylactic retransmission*, or p-retransmission. We illustrate this technique with an example:

1. Consider a situation where the receiver has acknowledged the sender's state #3. Then the application changes the object state (e.g., changes the contents of the screen) to a new state, #4.
2. The sender knows that the receiver already has state #3, so it creates a diff from #3 and #4 and sends it.
3. Soon after, and before the diff is acknowledged, the object state changes again to #5.
4. If the previous diff hasn't timed out, the sender will formulate a diff from state #4 and #5, with the assumption that both diffs will arrive and be applied. This is the "normal transmission." If the #3 to #4 packet was lost, the receiver won't be able to apply the new diff and will stall until the sender times out and retransmits.
5. But another option is to formulate a diff all the way from state #3 through #5: the p-retransmission. Sometimes this diff will actually be shorter or the same size as

the normal transmission, and it's more likely to be useful to the receiver because it doesn't have state #4 as a prerequisite. (It's a retransmission of sorts, because implicitly we're re-sending the diff between state #3 and #4 before it has timed out.)

The algorithm says that if the p-retransmission is shorter or only slightly longer (relative to packet overhead) than the normal transmission, we send it instead. The advantage is that if a loss occurs, we get the benefits of resending without necessarily repeating anything and without a timeout and stall, with a tunable maximum overhead.

Speculate, but Verify

Because Mosh operates at the terminal emulation layer and maintains the screen state at both the server and client, it's possible for the client to make predictions about the effect of user keystrokes and later verify its predictions against the authoritative screen state coming from the server.

Most UNIX applications react similarly in response to user keystrokes. They either echo the key at the current cursor location or not. As a result, it's possible to approximate a local user interface for arbitrary remote applications. We use this technique to boost the perceived interactivity of a Mosh session over network paths with high latency or high packet loss. When the RTT exceeds a threshold, we underline unconfirmed predictions so the user doesn't become misled. This underline trails behind the user's cursor and disappears gradually as responses arrive from the server. Occasional mistakes are removed within one RTT.

Previous work in this area, such as Telnet's LINEMODE option, only works when the kernel itself is echoing user keystrokes. This is rare today, as full-screen applications (such as emacs and vi) and even command-line shells now disable these kernel echoes and process keystrokes within the application. Mosh's approach is more general and doesn't require a change to server software.

The challenge is that about a third of user keystrokes are "navigation" (such as going to the next email message, or changing modes in vi) and shouldn't be echoed. Of course, password entry shouldn't be echoed either.

Mosh deals with this conservatively. The client makes predictions in groups known as "epochs," with the intention that either all of the predictions in an epoch will be correct, or none will. An epoch begins tentatively, making predictions only in the background. If any prediction from a certain epoch is confirmed by the server, the rest of the predictions in that epoch are immediately displayed to the user, along with any future predictions in the same epoch.

Some user keystrokes are likely to alter the host's echo state from echoing to not, or are otherwise hard to predict, including the up- and down-arrow keys and control characters. These cause Mosh to lose confidence and increment the epoch, so that future predictions are made in the background again.

We evaluated Mosh using traces contributed by six users, covering about 40 hours of real-world usage and including 9,986 total keystrokes interacting with a variety of UNIX applications. These traces included the timing and contents of all writes from the user to a remote host and vice versa. The cumulative distributions and statistics of keystroke response time are shown in Figure 3. When Mosh

was confident enough to display its predictions, the response was nearly instant. This occurred about 70% of the time. Our USENIX ATC paper [2] reports similar results on other Internet paths.

Get the Details Right, or, If There Is No “Right,” Defensible

Character set handling on UNIX—maybe on any operating system—remains a dark and under-specified area. We learned, for example, that there is no specification for a Unicode terminal emulator (the ECMA-48 specification was last revised in 1991). Existing terminal emulators show a variety of interpretations on issues such as the order of normalization and the placement of combining accents when mixed with control characters (such as newline) and escape sequences.

Figure 4 illustrates this problem. The same sequence, in which a combining accent is mixed with an escape sequence and newline, gives rise to four different interpretations in four popular terminal emulators. The most interesting of these is the Mac OS X Terminal.app, which normalizes the input before parsing the escape sequences, then triggers a bug that freezes the terminal.

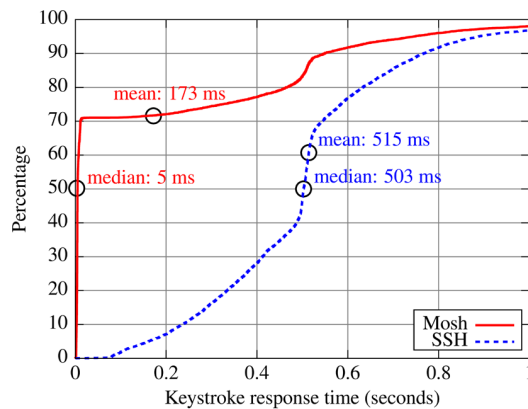


Figure 3: Cumulative distribution of keystroke response times with Sprint 1xEV-DO (3G) Internet service

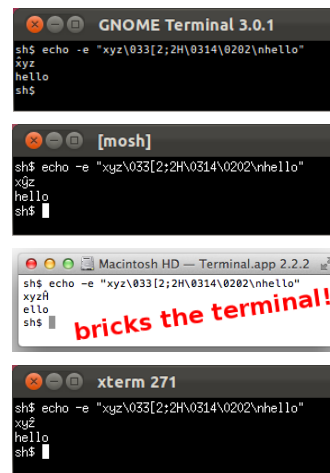


Figure 4: Same string, four interpretations

We can't argue that Mosh's interpretation of Unicode corner cases is *correct*, because there is no authority on such matters, but we did put effort into making it *defensible*.

We also learned that POSIX and the Single UNIX Specification don't provide a mechanism to delete multibyte characters (including UTF-8 sequences) in "canonical mode," because the kernel doesn't know how many bytes to delete, or even what character set the user is in. Linux and Mac OS X have responded by creating an IUTF8 termios flag to tell the kernel that it should interpret input as UTF-8.

Mosh sets this flag where available, but that only fixes part of the problem: the kernel still doesn't know how many columns the character occupies on the display and how many backspaces to send. With IUTF8 set, deleting wide Chinese, Japanese, and Korean characters won't leave garbage in memory, but it still leaves garbage on the screen. There's no easy solution to this problem; we hope CJK users aren't often trying to type and then delete wide characters in canonical mode.

Solve a Small Problem

With Mosh, we tried not to solve any problems we could avoid. Mosh doesn't authenticate users, run a daemon, or contain any privileged (root) code. Users don't need root to install Mosh on the client or server. Mosh doesn't use public-key cryptography. Users continue to authenticate and log in through their existing means: probably SSH with passwords, public keys, or Kerberos.

The mosh program is a script that uses SSH to log into the server and execute an unprivileged mosh-server process, which prints out an AES key and binds to a high port number. The script then shuts down the SSH connection and executes the mosh-client with the supplied key and port. The client contacts the server directly over UDP.

Mosh doesn't support multiple windows, split-screen modes, multiple clients connected to the same server, or reattaching if the client has rebooted or the user has moved to a different machine. For these features, users often run terminal multiplexers like GNU screen or OpenBSD tmux inside a Mosh session.

Limitations and Future Work

The current version of Mosh (version 1.2.2, June 2012) has several limitations:

- ◆ Because Mosh only synchronizes an object representing the current contents of the screen, there is no guarantee that the user will be able to scroll back to history that was missed. If the application is sending large volumes of output or the user was disconnected, the history could have been in lines that were skipped over. The user must use a server-side pager, such as screen or less, to have accurate scrollback.
- ◆ Mosh doesn't tunnel X11 sessions or ssh-agent requests.
- ◆ Mosh doesn't work through TCP-only proxy servers.
- ◆ The Mosh server requires a separate UDP port for each concurrent client—when the client's IP address can vary, the server-side UDP port is the only invariant connection identifier. Some users have objected that opening the default port range of 60000–61000 is not a realistic request of security-conscious network administrators.
- ◆ Mosh clients for Android and iOS are still in development.

- ◆ Mosh doesn't support IPv6, or roaming from IPv6 to IPv4 addresses for the same server.
- ◆ Mosh provides transport-layer security, but doesn't attempt to hide the existence of a session.

Mosh has attracted a cadre of online contributors, and we plan to address some of these issues in future versions. Limited as Mosh is, we've received valuable and encouraging feedback from users. Releasing early and often was the right decision.

References

- [1] T. Krovetz and P. Rogaway, "The Software Performance of Authenticated-Encryption Modes," *18th Intl. Conf. on Fast Software Encryption*, 2011.
- [2] K. Winstein and H. Balakrishnan, "Mosh: An Interactive Remote Shell for Mobile Clients," *USENIX Annual Technical Conference*, Boston, Mass., June 2012.

Passwords for Both Mobile and Desktop Computers

ObPwd for Firefox and Android

MOHAMMAD MANNAN AND P.C. VAN OORSCHOT



Mohammad Mannan is an Assistant Professor at the Concordia Institute for Information Systems

Engineering, Concordia University, Montreal. He has a PhD in Computer Science from Carleton University (2009) in the area of Internet authentication and usable security. He was a postdoctoral fellow in the Department of Electrical and Computer Engineering at the University of Toronto from 2009 to 2011 (funded by an NSERC PDF, and ISSNNet). His research interests lie in the area of Internet and systems security, with a focus on solving high-impact security and privacy problems of today's Internet.

mmannan@ciise.concordia.ca



Paul C. van Oorschot is a Professor of computer science at Carleton University in Ottawa, where he is a

Canada Research Chair in authentication and computer security. He was program chair of USENIX Security '08, program co-chair of NDSS 2001 and 2002, and co-author of the *Handbook of Applied Cryptography* (1996). He is on the editorial board of IEEE TIFS and IEEE TDSC. His current research interests include authentication and identity management, security and usability, software security, and computer security. He is a member of the IEEE.

paulv@scs.carleton.ca

Many users now access password-protected accounts and Web sites alternately from desktop machines and mobile devices (e.g., smartphones, tablets). The input mechanisms of the mobile devices are often miniature physical or virtual on-screen keyboards, posing challenges for users trying to type passwords with mixed-case and special characters expected by Web sites and more easily entered on desktop keyboards. We begin with a review of these challenges and existing proposals addressing cross-device password entry, including some password managers. We then bring the issues into focus with detailed discussion of the interoperational challenges and implementation and interface details of the object-based password (ObPwd) mechanism, as implemented for the Android platform, plus compatible browser-based and stand-alone implementations for desktop environments. ObPwd generates a password from a user-selected digital object (e.g., image), does not require changes to server-side software, and avoids the text-input challenges of mobile devices. We also briefly evaluate ObPwd using a recently proposed evaluation framework for password authentication schemes. A major goal is to increase attention to the cross-device password authentication problem.

We all wish passwords would go away. Their faults are many and are well-documented elsewhere. However, for the present time, the Internet world continues to have a deep investment in them. Millions of Web sites continue to demand passwords.

Almost all users with a few years of experience are familiar with the task of typing passwords on full-size Qwerty keyboards. For many accounts and Web sites, users are encouraged or required to choose passwords with mixed-case letters, digits, and special characters. How such policies affect the overall security and usability requires an entirely separate discussion. In recent years, the flood of new devices in the marketplace has included smartphones, tablet PCs, Internet-enabled TVs, and game consoles. Many of these offer only a limited on-screen keyboard, e.g., touch/stylus-based, remote control, or miniature physical keyboards. When these devices are used for Web browsing, passwords must be input to access password-protected sites. Many applications ("mobile apps") also require password input, some of which are also accessible on desktops as independent or Web applications.

The authentication task is now more complicated than when using a full-size keyboard: text password input is more error-prone and frustrating in terms of locating the keys and entering the characters, especially if entering special characters requires multiple keys to reach alternate (shifted) keyboards. Such text-unfriendly input interfaces may also influence user-chosen passwords to be even weaker

than otherwise. Indeed, Jakobsson et al. [9] reported from a survey of 50 smartphone users that 88% of device passwords are digit-only. Arguably, user-choice is influenced by available screen-lock options and the default option presented to users. They also reported that 46% of users enter a password once or more per day on mobile devices, and that 56% mistype a password at least one in 10 times. Users even stated that mobile-device password entry is more annoying than lack of coverage and poor voice quality. On the other hand, a new smartphone dynamic is underway largely due to the input problem: mobile apps often require users to enter an app-password only once (e.g., upon installation or first run), which is then saved by the app, and thereafter the stored password is used to authenticate the user. This is analogous to a desktop browser permanently remembering passwords for Web sites. However, the general password input problem remains, especially for browser-accessed services in mobile devices.

Others also have recognized the problem of text-password input in mobile devices. Several graphical and image-based schemes have been proposed: e.g., Windows 8 picture password, mobile Blue Moon authentication (<http://mobile-blue-moon-authentication.com>). Another idea, as discussed later, is a password scheme involving multiple real words [7, 8] and leveraging widely available auto-correct and auto-suggest features on mobile devices. However, using these same passwords becomes challenging in the desktop world if similar auto-correction functionality is unavailable. (For further reading on a similar topic, i.e., discussion of need for scheme that allows users to alternate between mobile phone and desktop keyboard, see Bicakci and van Oorschot [2].)

More generally, user-friendly remote authentication mechanisms custom-designed for mobile devices may not find wide acceptance, as the same online services will often be accessed from multiple devices with different input mechanisms. A major design criterion that must not be overlooked, and which is more challenging than initially apparent, is that such authentication mechanisms must also be suitable back on a desktop computer with a standard keyboard, since users alternate access devices. Consequently, the design of a user-friendly authentication system must suit a wide variety of devices, including those with input-constrained and conventional keyboards. We consider this challenge herein. As a specific example, we revisit a password mechanism called object-based password (*ObPwd*) originally designed in the context of the desktop world, and its implementation adopted to the mobile world.

The ObPwd mechanism constructs text-compatible passwords from digital objects such as pictures, generating strong passwords without requiring that users type mixed-case or special characters (see Fig. 1). A previous publication [3] outlines the basic idea and detailed results of a user study and security analysis. Our main focus here is an illustrative example, with concrete implementation details and design choices, of one solution to this challenge of designing a password scheme supporting *password entry modes across devices with a variety of input mechanisms* to stimulate further innovation and better solutions. In what follows, we describe the implementation of ObPwd adapted to the mobile space (on the Android platform), as well as in the desktop PC environment. Beside the basic design, we also emphasize the *domain-salted* variant of ObPwd that generates unique passwords from the same object (e.g., a picture) for different Web sites. This variant is particularly interesting in terms of addressing the widespread password reuse behavior among users that enables passwords leaked from low-security Web sites to be used in more sensitive sites.

Password Entry Challenges from Nonstandard Keyboards

Numerous usability issues arise when conventional text passwords must be entered on mobile devices that do not have standard physical keyboards. We review a few of these here.

Multitude of Devices with Different Keypad Layouts

No standard keyboard layout is followed across mobile devices. Common text entry methods include (1) multi-tap: multiple letters are mapped to the same physical key; (2) T9: text on 9 keys, a predictive text entry method allowing words to be entered by a single keypress; (3) full Qwerty keyboard; and (4) on-screen alternate keypads. Even different versions of devices from the same manufacturer may vary significantly in keypad layout (e.g., Nokia E7 vs. Nokia E63). Layout across devices from different classes of the same vendor (e.g., Nokia E vs. N series) and across devices from distinct vendors differ sufficiently to cause vendor lock-in for some users.

Keyboard forms. Various form factors are available, with no clear winner. Examples include physical keypad, touch-based, stylus, or pointer-based on-screen keypad (e.g., on the Wii game console). Forms other than physical keypads typically lack tactile feedback during input. Some on-screen keypads offer feedback via audio and visual channels, and vibrating the device. Interestingly, such user-friendly feedback may also leak information to nearby attackers when these devices are used in public places (e.g., through shoulder-surfing or video recording [11]).

Multiple Steps for Keyboard Shifting

In the default layout, small keypads (whether physical or touchscreen) offer a limited number of characters. Accessing additional characters requires tapping or pressing special keys (e.g., using shift to switch between keypad modes). Thus inputting a strong password with mixed-case letters and digits requires more keystrokes than characters in a password.

Cold Weather Issues and Fat Fingers

Most modern touchscreens use a capacitive sensing panel which operates by completing an electric circuit with the human body through the fingers. This hinders providing input to these devices with gloved fingers, as common gloves are made of electrically insulating materials. *Fat fingers* (“working men’s fingers”) may also hinder input precision in mobile keypads, due to small keypad size and visual interference.

Existing Proposals Supporting Cross-Device Password Entry

Here we mention a few of the existing proposals providing password authentication compatible across devices with varying text-input mechanisms. Security and usability problems with passwords are well known, and many techniques have been proposed to replace passwords altogether, e.g., biometrics, graphical passwords, and security tokens. The online-only appendix covers some other examples of mobile password systems.

Stand-alone Password Managers

Password managers are becoming more popular, for reasons including convenient access to passwords, need to maintain numerous accounts, browser integration, and online access. Several of the existing password managers (both stand-alone applications and Web services) that support multiple devices may alleviate the password input problem to some extent. Below we discuss two example password managers.

KEEPPASS

KeePass (<http://keepass.info>) is an open source password manager for multiple desktop platforms. It saves several Web site/application-specific items such as the site URL, user ID, and password in an encrypted database file on a user's local PC; the encryption key is derived from a user-chosen master password and, optionally, a user-selected or randomly generated file. The database files can additionally be locked with the user's Windows user account. The saved passwords can be copied onto the clipboard and then pasted into the intended application or Web site. A saved URL can be launched through the default system browser directly from the KeePass application; the KeeFox extension for Firefox can automate password entry to the Web site. KeePassSync is a KeePass plugin that offers synchronization of password databases between devices using online storage providers (e.g., Amazon S3). The database files can also be manually transferred. Third-party developed apps (e.g., iKeePass, KeePassDroid, and KeePassMobile, for iOS, Android, and J2ME devices, respectively) enable users to access KeePass databases from mobile devices.

LASTPASS

LastPass (<http://lastpass.com>) is a free online password manager available in most major desktop and mobile platforms. Passwords and other user data (e.g., notes, form data) are encrypted locally using a user-chosen master password; the encrypted result is saved on the LastPass server. LastPass and similar online password managers offer two distinct features to alleviate password problems in general: (1) portability—all user passwords are accessible from anywhere with a browser and Internet connection; and (2) backups—passwords are backed up without involving the user. However, these highly appreciated features come with side-effects. For example, the encrypted password list may be vulnerable to dictionary attacks. This vulnerability depends on the strength of the user-chosen master password; historical experience of user-choice issues makes this worrisome.

Some password managers facilitate generating random passwords as site passwords, but the master password itself remains user-chosen. Solutions such as Kamouflage [4] which store decoy passwords along with user passwords can also be adopted to frustrate dictionary attacks on stolen encrypted password storage. These online password managers offer an attractive target to attackers: compromising such a server allows access to a large number of user accounts. Indeed, in May 2011, LastPass was possibly compromised in such an attack [13]. When users were forced to reset their master password after the attack, some users were stuck, as the reset process required logging into their pre-registered email address, the password of which was also saved with LastPass. Such account lock-out appears to be an intrinsic problem with online password managers that use email for account recovery.

A general drawback of password manager services is the tangible privacy concern: despite assurances that user data is stored in encrypted form, the service providers may be compelled to make user data “available” to government agencies (e.g., see the recent changes in DropBox privacy policy [6]).

Karole et al. [10] conducted a usability study comparing three widely used password managers: LastPass (online), KeePassMobile (phone-based), and Roboform2Go (USB-based). Overall, LastPass was least preferred, and specifically the non-technical users in the study favored the phone-based manager. The authors attributed this finding to the fact that users prefer control over their passwords, rather than trusting a third party. However, they also reported that the usability of phone-based managers is not on par with user expectations. Another study [1] on the security of smartphone-based password manager apps reported several implementation weaknesses, including easy verification of master password and hard-coded encryption keys.

Browser-Based Password Synchronization

Synchronization functionality is built into Firefox 4 (older versions can use the Sync add-on), and Firefox Mobile (available on Android OS and Maemo/Nokia N900). For example, Firefox Sync saves user bookmarks, passwords, open tabs, form data, and browsing history for access from PCs and mobile devices. Saved content is also accessible from iOS (iPhone, iPad, iPod) via the Firefox Home application. User data is encrypted locally, and then stored and shared via a Mozilla hosted server; users can set up their own server to be used with Sync (e.g., if they do not trust Mozilla with their data).

Tapsure is a Firefox Mobile add-on (<https://addons.mozilla.org/en-us/mobile/addon/tapsure/>) that enables users to input saved text passwords by tapping a rhythm/pattern on the phone’s touchscreen. Users can save a password entered on a Web site (through usual input methods), by tapping a personal pattern on the screen; the same password can be accessed from any sites that use it via Tapsure. Tapsure uses Firefox’s built-in password manager for storing passwords. The tapping pattern serves as an easy-to-use unlock password that enables access to the saved text password. Tapsure differs from browser password managers in a subtle but important way. A Tapsure-saved password can be accessed from any site when the specific tapping pattern is entered, thus facilitating easier input of reused passwords. In contrast, browser password managers save pairs of user ID-password for individual sites which are made available only at the specific sites (from the saved password list, which is optionally encrypted under a user-chosen master password).

Cross-platform/Cross-device ObPwd Implementations

We outline here the basic ObPwd scheme and several implementations in different platforms. The implementation details may help you understand the challenges such as design and user interface issues in cross-platform/cross-device implementations. The ObPwd FAQ and implementations are publicly available from <http://www.ccs.l.carleton.ca/~mmannan/obpwd/> as an OS-agnostic Firefox browser extension, and as stand-alone applications in Android, Microsoft Windows, Mac OS X, and Linux. An initial Firefox extension and prototypes on other major platforms have been well received, and public feedback has resulted in modifications and upgrades. To our knowledge, the technology is free of patents.

While we explain specific implementation choices made in order to convey a concrete sense of the functionality and interfaces available, different design choices could be made for reasons of preference, usability, and security, matching different intended contexts and use environments.

The Basic ObPwd Mechanism

The core functionality in all ObPwd tools is to output a text password from user-selected content; see Figure 1. The current implementations use SHA-1 to hash password objects. The hash output is mapped to a base-64 character set, then converted to an alphanumeric password (default 12 characters) by known techniques [12]; for example, assuming a local file is used as the password object, $\text{pwd} = \text{Hash2Text}(\text{Hash}(\text{fileContent}))$. Up to the first $n = 100,000$ bytes are used from an object; 160 bytes are required. Variants of the basic mechanism are discussed elsewhere [3]. One of these variants, discussed later, is the *domain-salted variant*, which involves using a local file and the Web site domain as a salt (i.e., $\text{pwd} = \text{Hash2Text}(\text{Hash}(\text{URLdomain} || \text{fileContent}))$). This domain-salted variant is provided by both the Firefox add-on and the Android app implementations mentioned above, starting with versions 1.0.1 and 1.0.



Figure 1: ObPwd basic mechanism

ObPwd Firefox Extension (Desktop)

This extension can be activated from the browser context menu (i.e., right-/secondary-click menu). Under the “Object-based Password (ObPwd)” menu, several sub-menus appear (depending on the right-click context): (1) “Get ObPwd from Local File” brings up a file dialog box for selecting a local file as a password object; (2) “Get Unique ObPwd: Local File + Domain” offers choosing a local file, and then the domain name of the current page is used with the file content to generate a site-specific password; (3) “Get ObPwd from Selected Text” generates a password using the selected text block on the Web page (if there is any selected text string); (4) “Get ObPwd from Image” generates a password from the selected image (i.e., the one right-clicked on, if any); (5) “Get ObPwd from Link” generates a password from the content as pointed by the URL right-clicked on, if any. Certain types of relatively stable HTTP and HTTPS links are supported by default (e.g., pdf, mp3, avi, txt, jpg, zip, wav), but not several common URL extensions (e.g., html, php, asp) which commonly host dynamic content—e.g., news page content may change as user comments are added, precluding regeneration of the original password.

Configuration preferences support changing the default resulting password length (6 to 20 characters, default 12) and including special characters. If a password is generated with certain preferences, the same preferences must be selected to re-create that password (irrespective of where the password is used).

When a password object (an image, highlighted text, URL, or local file) is selected, the extension generates a password from the underlying content and displays

the password in a dialog box in plaintext to enable users to record it for backup in a secure way. If the OK button is clicked, the password is copied to the system clipboard, allowing pasting anywhere by the user. Note that by default, for security and privacy reasons, Firefox clipboard data is not accessible to JavaScript embedded on a site. The password is inserted directly into a password input box on a login page, if the extension is activated from such a box (i.e., the context menu is brought up by right-clicking on the password box). This auto-filling both automates the password copy-paste step and protects the password from shoulder-surfing.

ObPwd Android App (Mobile Devices)

Installing the ObPwd app adds a menu item (labeled “ObPwd”) to the “Share” menu of Gallery, the default media app for Android devices. Users can browse their media files stored on the device and from Picasa Web albums (if the user’s Google account is linked to Picasa). When the user selects an image or video and chooses the ObPwd app from Gallery’s Share menu, the user is asked if the domain of the last visited Web site should be used in the password generation (i.e., whether to use the *domain-salted variant*). Then the corresponding text password is displayed. The password display dialog offers two choices: “Copy to clipboard” and “Quit.” If copied, the password can be pasted to any password field (e.g., in Web sites and other apps) without requiring typing the password.

Usability, Limitations, and Evaluation

Discussion on Usability and Features of the Basic ObPwd

A hybrid in-lab/at-home user study using the ObPwd desktop extension was conducted involving 32 participants (see [3] for full details). Participants were asked to use 11 new passwords (eight test Web sites and three real-world sites) in a span of 7–10 days. The study reported encouraging results in terms of several usability factors. The login success rate was more than 90% (on the first attempt) in a return-to-lab session. The average login time was about 20 seconds, which despite being longer than text password logins in the desktop environment, was reflected in a positive affect by participants: they reported that they enjoyed browsing their password objects both when creating passwords and logging in. This result is atypically positive compared to other new password proposals. Whether similar positive results occur for other ObPwd platforms and/or implementations requires further user studies; we presently have only anecdotal praise from users of the Android app, but these are self-selected, technically savvy users not representative of the general population.

The user interfaces of ObPwd tools described above differ depending on the device/environment, reflecting hardware and software interfaces which vary significantly across these devices. Instead of implementing a separate image/video-browsing interface on Android, the implementations described rely on the default Gallery app for object selection. This provides a familiar interface to users, but on the down side, the implementation shortcut overloaded the “Share” menu to initiate the ObPwd app. “Share” is an unfortunate name for this function menu, since security is defeated if users openly share their password objects (as facilitated by several other applications in the Share menu). Indeed, ObPwd security relies heavily on the assumption that users’ password objects remain private, as opposed to, for example, publicly posted photos.

Some features of the ObPwd scheme may favor its adoption. Personal digital content is now easily available on both desktop and mobile platforms. As ObPwd requires no server-side changes, users can immediately benefit upon installing freely available implementations. ObPwd passwords are typically as strong as system-generated passwords, with respect to guessing attacks (see [3] for further discussion), yet the password objects are user-chosen. On the other hand, many visible and invisible barriers exist to installing any new authentication mechanism intended to replace passwords; widespread adoption of ObPwd, or any other alternative, is likely to occur only if adopted by a major platform vendor or browser provider.

ObPwd is a hybrid mechanism both in terms of authentication method (part what-you-know, part what-you-have-access-to), and input type (involving media such as image/video/music-based). It is not a password manager in a traditional sense (i.e., does not store passwords), but empowers users to better manage several strong passwords (as apparent from our user testing) by taking advantage of the positive attachment users already have to their personal content.

The ObPwd Domain-Salted Variant

For generating ObPwd passwords, we assume the *domain-salted variant* in the evaluation. This variant is arguably the safest, and, from a user interface viewpoint, indistinguishable from that used in the user study [3]; we note, however, that this variant itself and the ObPwd Android app have not been formally user-tested. We also assume that a single local file is used as the password object for all Web sites: i.e., the password object is used as a master password from which unique, site-specific passwords are generated (cf. PwdHash [12]). This assumption is rooted in the current practice of reusing the same or few text passwords across many accounts. We argue that access to site-specific passwords does not allow an attacker or a malicious site operator to (easily) create passwords for other sites, since this will require guessing the file-content of the password object (recall that $\text{pwd} = \text{Hash2Text}(\text{Hash}(\text{URLdomain} \parallel \text{fileContent}))$). In effect, the best attack remains the exhaustive search; note that, for an attacker not in possession of the password object, exhaustive search requires on the order of 2^{70} guesses in default settings (see [3] for details). In contrast, a compromised site-specific password generated from a user-chosen master password (e.g., $\text{pwd} = \text{Hash2Text}(\text{Hash}(\text{URLdomain} \parallel \text{masterPassword}))$) may reveal the master password under offline dictionary attacks. Thus, from a security viewpoint, ObPwd is analogous to using a random string stored on the user's machine. However, from a usability viewpoint, in contrast to ObPwd's use of a user-chosen object, a random string is neither user-friendly nor recognizable to users, and provides no positive feedback.

Comparison and Usability-Deployability-Security Evaluation

We compare and evaluate ObPwd against basic text passwords, using the UDS (usability, deployability, security) framework of 25 baseline properties for user authentication schemes [5]. For context, we show the UDS rating for (Web passwords, desktop) as the first row of Table 1 herein. The appendix (online only) explains how the authors came up with the ratings that appear in Table 1. We use separate table rows for implementations of each mechanism for desktop (assuming a full-size keyboard) and mobile platforms (e.g., mobile phones with small hardware or touch-based keypads, and tablets), as the usability ratings in particular differ.

	Usability	Deployability	Security
	U1: Memorywise-Effortless U2: Scalable-for-Users U3: Nothing-to-Carry U4: Physically-Effortless U5: Easy-to-Learn U6: Efficient-to-Use U7: Infrequent-Errors U8: Easy-Recovery-from-Loss	D1: Accessible D2: Negligible-Cost-per-User D3: Server-Compatible D4: Browser-Compatible D5: Mature D6: Non-Proprietary	S1: Resilient-to-Physical-Observation S2: Resilient-to-Targeted-Impersonation S3: Resilient-to-Throttled-Guessing S4: Resilient-to-Unthrottled-Guessing S5: Resilient-to-Internal-Observation S6: Resilient-to-Leaks-from-Other-Verifiers S7: Resilient-to-Phishing S8: Resilient-to-Theft S9: No-Trusted-Third-Party S10: Requiring-Explicit-Consent S11: Unlinkable
Web passwords (desktop)	● ○ ● ● ○ ●	● ● ● ● ● ●	○ ● ● ● ● ● ● ● ● ● ●
Web passwords (mobile)	● ● ● ● ●	○ ● ● ● ● ●	○ ● ● ● ● ● ● ● ● ● ●
ObPwd (desktop)	○ ● ● ● ○ ● ●	● ● ● ○ ● ●	● ● ● ● ● ● ● ● ● ● ● ●
ObPwd (mobile)	○ ● ● ● ○ ● ●	● ● ● ● ● ●	● ● ● ● ● ● ● ● ● ● ● ●

Table 1: Comparing conventional text passwords (web passwords) to ObPwd passwords. Key: ● (offers the benefit); ○ (almost offers the benefit); blank (benefit not offered).

Concluding Remarks

The ubiquity of traditional keyboards in desktop systems has played an important role in the proliferation of text passwords as the primary mode of user authentication on the Internet. For user authentication from mobile devices, the opportunity exists to exploit device-specific features such as multi-touch, GPS, accelerometer, and camera to improve both security and usability (e.g., see [9]). However, a critical requirement is that the authentication of users who alternate between desktop and mobile systems must be accommodated. Greater customization of authentication schemes, such as allowing user selection of per-login authentication modes, may be the path to better support the emerging multi-device/multi-platform usage scenarios. The system side could automatically detect the type of device the user is on, and offer a different login interface or variation based on that. The interaction between user authentication and evolving password managers (and their support across platforms and devices, including cross-device password synchronization) is likely to become an increasingly important part of the user authentication equation. Another important but unexplored aspect of cross-device authentication is whether current user behavior is affected by input difficulties and, if so, to what extent; for example, do users significantly change which sites they visit depending on which access device they use specifically to avoid accessing sites that require password input?

The implementations discussed herein are an illustrative example intended to motivate further discussion and innovation addressing the problem of user-friendly password authentication from alternating computing devices supporting divergent user input capabilities. No doubt better mechanisms will appear over time. Their chances of deployment success in practice will be much higher if designed from the start keeping in mind cross-device requirements as discussed herein. We hope this article motivates and expedites further progress.

References

- [1] A. Belenko and D. Sklyarov, “‘Secure Password Managers’ and ‘Military-Grade Encryption’ on Smartphones: Oh, Really?” Blackhat Europe 2012: <http://www.elcomsoft.com/WP/BH-EU-2012-WP.pdf>.
- [2] K. Bicakci and P. van Oorschot. “A Multi-Word Password Proposal (gridWord) and Exploring Questions about Science in Security Research and Usable Security Evaluation,” New Security Paradigms Workshop (NSPW’11), Marin County, CA, USA, Sept. 2011.
- [3] R. Biddle, M. Mannan, P. van Oorschot, and T. Whalen, “User Study, Analysis, and Usable Security of Passwords Based on Digital Objects,” *IEEE Transactions on Information Forensics and Security (TIFS)* 6(3):970-979, Sept. 2011.
- [4] H. Bojinov, E. Bursztein, X. Boyen, and D. Boneh, “Kamouflage: Loss-Resistant Password Management,” *European Symposium on Research in Computer Security (ESORICS’10)*, Athens, Greece, Sept. 2010.
- [5] J. Bonneau, C. Herley, P.C. van Oorschot, and F. Stajano, “The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes,” IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 2012.
- [6] BusinessInsider.com, “DropBox: We’ll Turn Your Files Over to the Government If They Ask Us To,” *Business Week*, Apr. 18, 2011: <http://www.businessinsider.com/dropbox-updates-security-terms-of-service-to-say-it-can-decrypt-files-if-the-government-asks-it-to-2011-4>.
- [7] W. Cheswick, “Rethinking Passwords,” invited talk at USENIX LISA ’10: <http://www.usenix.org/event/lisa10/tech/slides/cheswick.pdf>. See summary in *login*: 36(2):68-69, Apr. 2011.
- [8] M. Jakobsson and R. Akavipat, “Rethinking Passwords to Adapt to Constrained Keyboards (Short Paper),” Mobile Security Technologies (MoST) Workshop, Oakland, CA, USA, May 2012.
- [9] M. Jakobsson, E. Shi, P. Golle, and R. Chow, “Implicit Authentication for Mobile Devices,” USENIX Workshop on Hot Topics in Security (HotSec’09), Montreal, Canada, Aug. 2009.
- [10] A. Karole, N. Saxena, and N. Christin, “A Comparative Usability Evaluation of Traditional Password Managers,” International Conference on Information Security and Cryptology (ICISC’10), Seoul, Korea, Dec. 2010.
- [11] R. Raguram, A. White, D. Goswami, F. Monrose, and J.-M. Frahm, “iSpy: Automatic Reconstruction of Typed Input from Compromising Reflections,” ACM Computer and Communications Security (CCS’11), Chicago, IL, USA, Oct. 2011.
- [12] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J.C. Mitchell, “Stronger Password Authentication Using Browser Extensions,” USENIX Security Symposium, Baltimore, MD, USA, 2005.
- [13] “LastPass Potentially Hacked, Users Urged to Change Master Passwords,” TheNextWeb.com., May 5, 2011: <http://thenextweb.com/apps/2011/05/05/lastpass-potentially-hacked-users-urged-to-change-master-passwords/>.

Looking Within to Improve American Cybersecurity

RICHARD F. FORNO



Dr. Richard Forno directs the graduate cybersecurity program at the University of Maryland Baltimore County.

His twenty-year career includes helping build the first formal cybersecurity program for the US House of Representatives, serving as the first Chief Security Officer at Network Solutions (the InterNIC), and co-founding the Maryland Cyber Challenge. Richard was one of the early researchers on the subject of “information warfare,” and he remains a longtime commentator on the influence of Internet technology upon society. The views expressed in this commentary are his and may not reflect those of his employer.

richard.forno@umbc.edu

If history is any guide, the ongoing attempts in 2012 to improve American cybersecurity through legislation will continue the practice of “admiring the problem” but not provide significant or lasting cybersecurity improvements for the nation. Many of the Internet security concerns first discussed in the mid-1990s still exist today, and the assorted recommendations to remedy them have not changed much, either. What has changed, however, is the importance of the Internet to contemporary society and the increased risks we face as a result of failing to learn from history in our attempts to address the problem of securing America’s cyberspace.

For example, in 1996, just before the “Dot-Com Revolution,” the Senate Permanent Subcommittee on Investigations conducted a survey of Fortune 1000 companies to explore the then-emerging concerns of Internet security and its importance to the country. In 1997, the “Manhattan Cyber Project” again examined these issues and developed additional recommendations regarding computer and networked systems security. That same year, the report of the Presidential Commission on Critical Infrastructure Protection (PCCIP) echoed those prior findings and offered its own recommendations from a national security perspective. After 9/11, the “National Strategy to Secure Cyberspace” (2003), the President’s Information Technology Advisory Committee’s cybersecurity report (2005), the “Comprehensive National Cybersecurity Initiative” (2008), the Center for Strategic and International Studies’ “Securing Cyberspace for the 44th Presidency” (2008), and the “White House Cybersecurity Policy Review” (2009) all offered similar recommendations, as have numerous other proposals, reports, analyses, task forces, and pundits over the years. Among the most prominent and oft-repeated recommendations include the sharing of cybersecurity information, fostering closer cooperation between government and commercial firms on cybersecurity activities (“public-private partnerships”), funding new cybersecurity research, acknowledging an ever-changing threat landscape, respecting privacy and civil liberties, and developing a public education campaign about cybersecurity.

Unfortunately, technology changes and time passes, but people remain the same. Longtime cybersecurity professionals, myself included, are frustrated that such ideas, when dusted off and reiterated anew, are greeted warmly by Washington policymakers, who typically end up proposing the same solutions again while hoping for a different outcome [1]. (Einstein had a term for this behavior.) Therefore, I offer three new recommendations to counter this trend and, perhaps, lead to meaningful discussions about actually improving American cybersecurity:

First, ***we must realize that many, but certainly not all, of our cybersecurity problems are self-inflicted.*** Although we are enamored by the promise and allure of new computing technologies and services, in our rush to embrace these innovations we often overlook their potential risks to our well-being and ability to remain resilient in the face of adversity. For example, much has been said over the past fifteen years about protecting our national critical infrastructures as they became more interconnected—and as we became more dependent upon them. However, a fundamental problem remains: if, as a nation, we consider something to be a critical national resource whose disruption by a cyber-attack might endanger public safety and security, effectively securing it requires more than a new National Strategy, greater numbers of cybersecurity professionals, or more technological countermeasures. It necessitates spending the time and resources to “raise the bar” at a fundamental technical and operational level to make things more difficult for adversaries or accidents to cause us problems in the first place. Admittedly, this may incur significant upfront costs for critical infrastructure providers to implement but will likely provide them a higher degree of operational resilience, assurance, and cost savings over the long term. The “Build Security In” initiative of the Department of Homeland Security (DHS) represents such an approach toward software development; however, the same concepts also are applicable to new hardware systems and entire technology infrastructures.

Being truthful with ourselves also means that we acknowledge facts that make us uncomfortable regarding our cybersecurity activities and reviewing where we have gone wrong, both in practice and policy. We are barraged constantly with warnings about the dangers (both real and perceived) that Internet-based hackers, terrorists, foreign agents, or criminals pose to the security of our water treatment facilities, power grids, and other national critical infrastructures so vital to society, but we continue to connect them to the public Internet where such adversaries lurk—and then wonder why incidents happen. Evidently, such preventable vulnerabilities exist in our critical infrastructures: a March 2012 White House-sponsored cybersecurity exercise centered around an incident triggered by a hacker gaining access to New York’s power grid systems from the public Internet using an email attack known as “phishing” [2, 3]. Yet none of the current legislative initiatives to improve American cybersecurity proposes that these critical systems be removed from the public Internet—an action that would reduce the number of Internet-based attacks on those systems, if not eliminate the danger entirely. In other words, our own behavior makes it easy for costly, and in many cases preventable, cybersecurity problems to occur.

Second, ***our national dialogue about cybersecurity must transcend sensationalism and special interests.*** Apocalyptic and overused terms like “Digital Pearl Harbor” and “Cyber Katrina” or “Cyber 9/11” must be eliminated from the national lexicon in order to allow rationality and facts to form the basis of proactive public discourse about cybersecurity instead of the reactionary fear-based perceptions implied by the use of such phrases. National cybersecurity policy discussions must include experts from across the spectrum of Internet users and the cybersecurity community instead of drawing on the same people representing the same organizations who recite the same talking points as they have over the past fifteen years—including those who stand to profit by providing solutions to persistent and unresolved cybersecurity problems [4], whose unrelated legislative concerns might disrupt normal cybersecurity activities and routine Internet

operations [5], threaten constitutional liberties [6], or override existing legal protections [7, 8].

Information is crucial to establishing a common understanding and for conducting objective analysis of problems. Therefore, since many cybersecurity concerns are shared by government, business, and individual users alike, minimizing the amount of cybersecurity information that is classified, instead of increasing the number of people granted access to the allure of secret information (much of which is needlessly made secret to begin with), will assist this process of public understanding and is something that even former CIA and NSA Director Michael Hayden agrees with [9]. An expanded public understanding of our cybersecurity situation can indirectly, if not also more objectively, inform national lawmakers about the current state of cybersecurity when deliberating far-reaching national policies related to it. These simple actions will help ensure that cybersecurity policy discussions are based on the professional knowledge of those most closely working in the field and not exclusively on apocalyptic speculation, special-interest posturing, or information that cannot be analyzed or disputed independently and publicly.

Finally, and perhaps most important, ***there must be meaningful accountability for cybersecurity***. Inculcating and sustaining a national and deep-seated commitment to cybersecurity success has become marginalized by the all-too-human tendency to cut corners for the sake of convenience. Although many of America's cybersecurity problems are self-inflicted, there are few real consequences faced by those responsible beyond short-term financial costs and adverse publicity. Absent meaningful consequences for cybersecurity failures, there is little incentive to overcome the inertia of the problematic status quo.

In terms of accountability, commercial and government organizations would not integrate emerging technologies [10] into their computing environments without first considering the risks and their ability to function should those resources become denied, disrupted, or degraded through intentional or unintentional incidents. This commits them either to forgo such innovations or spend the requisite resources to ensure that the security and availability of critical infrastructures and citizen data over the long term are not sacrificed in the name of enticing cost savings or operator convenience in the near term. Thus, "accountability" need not be externally applied but can emerge through voluntary internal process improvements—which may or may not involve new expenditures or dramatic changes to daily operations. However, when likely preventable cybersecurity incidents do occur, meaningful external accountability and consequences must be brought to bear upon organizations failing to implement effective cybersecurity practices that would have prevented the incident [11]. Instead, new legislative proposals purporting to improve cybersecurity offer broad indemnification to companies, which allows them to escape responsibility for their actions—or lack thereof—when cybersecurity problems occur [12]. Accordingly, implementing accountability for cybersecurity must also extend to policymakers and regulators for their roles in developing the laws and oversight mechanisms that create or perpetuate such problems, and for creating bureaucratic situations that preclude effective, long-term national leadership on cybersecurity matters.

New laws alone will not solve America's cybersecurity problems. Successfully managing America's cyber risks depends on our ability to be honest with ourselves and having the courage to act from a position of fact and objective, rational knowl-

edge while accepting responsibility for our actions. Only then will we be able to achieve greater effectiveness and long-term sustainability in our national cybersecurity posture, practices, and policies. To do otherwise will demonstrate that we have not learned from history and will perpetuate the problematic situation witnessed today.

References

- [1] D. McCullagh, "A Cybersecurity Quiz: Can You Tell Obama from Bush?" *CNET*, May 29, 2009: http://news.cnet.com/8301-13578_3-10252263-38.html.
- [2] M. Masnick, "If Phishing Email Can Kill NY Power Grid, Lack of Cybersecurity Legislation Is Not the Problem," *Techdirt*, March 12, 2012: <http://www.techdirt.com/articles/20120309/16470618060/if-phishing-email-can-kill-ny-power-grid-lack-cybersecurity-legislation-is-not-problem.shtml>.
- [3] J. Martinez, "White House Tries Cyber Scare Demonstration to Spur Senate," *Politico*, March 8, 2012: <http://dyn.politico.com/printstory.cfm?uuid=BCEC37C2-ABCD-4973-9858-569B77D9EFA5>.
- [4] J. Brito and T. Watkins, "Loving the Cyber Bomb? The Dangers of Threat Inflation in Cybersecurity Policy," *Harvard Law School National Security Journal*, 3(1). Accessed on 15 June 2012 at http://harvardnsj.org/wp-content/uploads/2012/01/Vol.-3_Brito_Watkins1.pdf.
- [5] T. Daly, "SOPA: Why Do We Have to Break the DNS?" *Dyn*, December 12, 2011: <http://dyn.com/sopa-breaking-dns-parasite-stop-online-piracy/>.
- [6] B. Ritholtz, "SOPA, PIPA, ACTA . . . What's Next?" *The Big Picture*, January 31, 2012: <http://www.ritholtz.com/blog/2012/01/sopa-pipa-acta-%E2%80%A6-what%E2%80%99s-next/>.
- [7] R. Cringley, "CISPA: Big Brother's Best Friend Forever," *Infoworld*, April 27, 2012: <http://www.infoworld.com/print/191952>.
- [8] D. McCullagh, "How CISPA Would Affect You (faq)," *CNET*, April 27, 2012: http://news.cnet.com/8301-31921_3-57422693-281/how-cispa-would-affect-you-faq/.
- [9] K. Poulsen, "Former NSA, CIA Chief: Declassify Cyber Vulnerabilities," *Wired*, March 14, 2012: <http://www.wired.com/threatlevel/2011/03/hayden-cyber/>.
- [10] For example, when examining the benefits of cloud computing services, prospective customers seem concerned with cost savings and how it may "empower" them in new and exciting ways, but overlook what may happen if that cloud (and the data, applications, and services within it) becomes unreachable by users who now are totally dependent on it.
- [11] Data loss or privacy incidents resulting from the theft of unencrypted corporate or government laptops are easily preventable using any number of accepted technical and procedural best practices. Failing to implement appropriate data safeguards in such cases should result in meaningful—and motivating—consequences both for the organization and for the employee(s) involved.
- [12] L. Beadon, "New Draft of CISPA Announced: Some Progress, Still Big Problems," *Techdirt*, April 13, 2012: <http://www.techdirt.com/articles/20120413/15420218488/new-draft-cispa-announced-some-progress-still-big-problems.shtml>.

The Future of Security

Criticality, Rejectionists, Risk Tolerance

DANIEL E. GEER, JR.



Dan Geer is the CISO at In-Q-Tel and likes to list the following milestones: the X Window System and Kerberos (1988), the first information security consulting firm on Wall Street (1992), convener of the first academic conference on electronic commerce (1995), the “Risk Management Is Where the Money Is” speech that changed the focus of security (1998), the Presidency of the USENIX Association (2000), the first call for the eclipse of authentication by accountability (2002), principal author of and spokesman for “Cyberinsecurity: The Cost of Monopoly” (2003), co-founder of SecurityMetrics. Org (2004), convener of MetriCon (2006–present), author of “Economics & Strategies of Data Security” (2008), and author of “Cybersecurity & National Policy” (2010). Creator of the Index of Cyber Security (2011) and the Cyber Security Decision Market (2011). Six times entrepreneur. Five times before Congress.

dan@geer.org

Pew reports [1] that

One in five American adults does not use the Internet. . . . Among adults who do not use the Internet, almost half [said] that the main reason they don't go online is because they don't think the Internet is relevant to them. . . . Though overall Internet adoption rates have leveled off, adults who are already online are doing more.

It may no longer be possible to live your life without dependence on the Internet. Unlike television, you cannot entirely unplug from the Internet even if you want to. If you are dependent on those who are dependent on television, then so what? If, however, you are dependent on those who are dependent on the Internet, then so are you. Dependence with respect to television is not transitive. Dependence with respect to the Internet is.

The source of risk is dependence, and security is the absence of unmitigatable surprise. It is thus obvious that increasing dependence means ever more difficulty in crafting mitigations, and that increasing complexity embeds dependencies in ways such that while surprises may grow less frequent, they will be all the more unexpected when they do come.

Because dependence on the Internet is transitive, those who choose “leave it” with respect to the Internet only get to say that in the first person; they are still dependent on it unless they are living a pre-industrial life. That rejectionists depend on people who are not rejectionist is simply a fact. Everyone has a stake in the game, but rejectionists have impact on the Internet-happy—rejectionists are now a kind of fail-safe. If we begin to penalize the rejectionists, that is to say, force them to give up on their rejectionism, we will give up a residuum of societal resiliency.

On November 13, 2002, a total computer outage at Boston's Beth Israel Hospital began [2]. The initiator was inadvertent high volume of data sharing among researchers; the impact was reverting to paper for four days, during which time doctors and laboratory personnel over 50 years old could cope; most of the rest could not. That a fallback to manual systems was possible saved the day, and it was those who could comfortably work without network dependence who delivered on that possibility, because they had done so at earlier times.

Thus the central thesis of this essay: accommodating rejectionists preserves alternate, less complex, more durable means and therefore bounds dependence. Bounding dependence is the core of rational risk management.

Common mode failure comes from under-appreciated mutual dependence. In NIST's High Integrity Software System Assurance documentation [3], they say, "A more insidious source of common-mode failures is a design fault that causes redundant copies of the same software process to fail under identical conditions." This is exactly what can be masked by complexity precisely because complexity ensures under-appreciated mutual dependence.

In an Internet crowded with important daily-life functions, the possibility of common-mode failure is no idle worry. The Obama administration is notably increasing dependence on the Internet on two fronts, either of which might be said to be, in the words of President Clinton's Presidential Decision Directive 63, "essential to the minimum operations of the economy and government": the press for electronic health records, and the press for the Smart Grid.

Electronic health records depend on the smooth functioning of electric power, networks, computers, displays, and a range of security features [4]. The Smart Grid depends on good clocks, industrial controls operated flawlessly at a distance and guaranteed not to lie about their state, and another range of security features.

Both of these involve new levels of exposure to common-mode risk; both add new failure modes to the world we live in. On good days, both will deliver cost-effective benefits. On bad days, doing without those benefits will be easier for those who can remember not having had them.

Each new dependence raises the magnitude of downside risk, the potential for collateral damage, and the exposure of inter-relationships never before contemplated. Forget the banks: it is the Internet that is too big to fail. While there is no entity that can bail out the Internet, there is no meaningful country that is not developing ways to disrupt the Internet use of its potential adversaries.

When 10% of the population sees nothing in the Internet for them, should we respect and ensure that, as with the Amish, there is a way for them to opt out without choosing to live in a cave? Should we preserve manual means?

I say "YES" and I say so because the preservation of manual means is a guarantee of a fallback that does not have a common-mode failure with the rest of the interconnected, mutually vulnerable Internet world. That this is not an easy choice is an understatement. I do not (yet) claim to have a fully working model here, but neither do our physicist friends (yet) have a unified field theory.

Summing up, risk is a consequence of dependence. Aggregate societal dependence on the Internet is not estimable. When dependencies are not estimable, they are underestimated. If they are underestimated, they will not be made secure over the long run, only over the short. As risks become increasingly unlikely to appear, the interval between events will grow longer. As the latency between events grows, the assumption that safety has been achieved will also grow, thus accelerating dependence in what is now a positive feedback loop. If the critical infrastructures are those physical and cyber-based systems essential to the minimum operations of the economy and government [5], and if aggregate risk is growing steadily [6], then do we put more of our collective power behind forcing security improvements that will be sharply diseconomic, or do we preserve fallbacks of various sorts in anticipation of events that become harder to mitigate as time passes? Is centralizing authority the answer, or is avoiding further dependence until we can fix things the better strategy? Should the individual who still prefers to fix things he

or she already has be celebrated, or are those individuals to be herded into National Health Information Networks, Smart Grids, and cars that drive themselves?

Resources

[1] “Digital Differences,” Pew Research Center, April 13, 2012: http://pewinternet.org/~media/Files/Reports/2012/PIP_Digital_differences_041312.pdf; tinyurl.com/d7eqo7v.

[2] P. Kilbridge, “Computer Crash—Lessons from a System Failure,” *New England Journal of Medicine*, vol. 348, no. 10, 6 March 6, 2003, pp. 881-882: http://ehealthcon.hs.network.com/NEJM_downtime_2003-03-06.pdf; tinyurl.com/75fjmbb.

[3] NIST, High Integrity Software System Assurance, section 4.2: http://hissa.nist.gov/chissa/SEI_Framework/framework_16.html; tinyurl.com/canwggd.

[4] Simon S.Y. Shim, “The CAP Theorem’s Growing Impact,” *IEEE Computer*, vol. 45, no. 2, February 2012, pp. 21–22.

[5] Presidential Decision Directive 63, May 22, 1998, <http://www.fas.org/irp/offdocs/paper598.htm>; tinyurl.com/4974j.

[6] The Index of Cyber Security: <http://cybersecurityindex.org>.

This article was abridged by the author from his keynote address for the Rocky Mountain Information Security Conference, Denver, May 17, 2012, which can be found at www.usenix.org/publications/login/august-2012/future-security.

Fast and Interactive Analytics over Hadoop Data with Spark

MATEI ZAHARIA, MOSHARAF CHOWDHURY, TATHAGATA DAS, ANKUR DAVE, JUSTIN MA, MURPHY MCCAULEY, MICHAEL J. FRANKLIN, SCOTT SHENKER, AND ION STOICA



Matei Zaharia is a fifth-year PhD student at UC Berkeley, working with Scott Shenker and Ion Stoica

on topics in computer systems, networks, cloud computing, and big data. He is also a committer on Apache Hadoop and Apache Mesos.

matei@eecs.berkeley.edu



Mosharaf Chowdhury is a PhD student at the University of California, Berkeley, working with Ion Stoica on topics in

cloud computing and datacenter networks. He received his undergraduate degree at Bangladesh University of Engineering and Technology (BUET) and a Master's degree from the University of Waterloo in Canada.

mosharaf@cs.berkeley.edu



Tathagata Das is a PhD student at the University of California, Berkeley, working with Scott Shenker on topics

in cloud computing, datacenter frameworks, and datacenter networks. He received his undergraduate degree at the Indian Institute of Technology, Kharagpur, India.

tdas@cs.berkeley.edu



Ankur Dave is an undergraduate at UC Berkeley and a Spark contributor since 2010. His projects include

Arthur, a replay debugger for Spark programs, and Bagel, a Spark-based graph processing framework.

ankurd@eecs.berkeley.edu



Justin Ma is a postdoc in the UC Berkeley AMPLab. His primary research is in

systems security, and his other interests include machine learning and the impact of energy availability on computing. He received BS degrees in Computer Science and Mathematics from the University of Maryland in 2004, and he received his PhD in Computer Science from UC San Diego in 2010.

jtma@cs.berkeley.edu



Murphy McCauley is a Master's student at UC Berkeley. His current focus is on software defined

networking.

murphy.mccauley@gmail.com



Michael J. Franklin is the Thomas M. Siebel Professor of Computer Science and Director of the Algorithms,

Machines and People Laboratory (AMPLab) at UC Berkeley. His current research is in the areas of scalable query processing, data integration, hybrid human/computer data processing systems, and cross-datacenter consistency protocols.

franklin@cs.berkeley.edu



Scott Shenker spent his academic youth studying theoretical physics but soon gave up chaos theory for

computer science. Unable to hold a steady job, he currently splits his time between the UC Berkeley Computer Science Department and the International Computer Science Institute.

shenker@cs.berkeley.edu



Ion Stoica is a Professor in the EECS Department at the University of California at Berkeley. His research

interests include cloud computing, distributed systems, and networked computer systems.

istoica@cs.berkeley.edu

The past few years have seen tremendous interest in large-scale data analysis, as data volumes in both industry and research continue to outgrow the processing speed of individual machines. Google's MapReduce model and its open source implementation, Hadoop, kicked off an ecosystem of parallel data analysis tools for large clusters, such as Apache's Hive and Pig engines for SQL processing. However, these tools have so far been optimized for one-pass batch processing of on-disk data, which makes them slow for *interactive* data exploration and for the more complex *multi-pass* analytics algorithms that are becoming common. In

this article, we introduce Spark, a new cluster computing framework that can run applications up to 40× faster than Hadoop by keeping data in memory, and can be used interactively to query large datasets with sub-second latency.

Spark started out of our research group's discussions with Hadoop users at and outside UC Berkeley. We saw that as organizations began loading more data into Hadoop, they quickly wanted to run rich applications that the single-pass, batch processing model of MapReduce does not support efficiently. In particular, users wanted to run:

- ◆ More complex, *multi-pass* algorithms, such as the iterative algorithms that are common in machine learning and graph processing
- ◆ More *interactive* ad hoc queries to explore the data

Although these applications may at first appear quite different, the core problem is that both multi-pass and interactive applications need to *share* data across multiple MapReduce steps (e.g., multiple queries from the user, or multiple steps of an iterative computation). Unfortunately, the only way to share data between parallel operations in MapReduce is to write it to a distributed filesystem, which adds substantial overhead due to data replication and disk I/O. Indeed, we found that this overhead could take up more than 90% of the running time of common machine learning algorithms implemented on Hadoop.

Spark overcomes this problem by providing a new storage primitive called *resilient distributed datasets* (RDDs). RDDs let users store data in memory across queries, and provide fault tolerance *without* requiring replication, by tracking how to recompute lost data starting from base data on disk. This lets RDDs be read and written up to 40× faster than typical distributed filesystems, which translates directly into faster applications.

Apart from making cluster applications fast, Spark also seeks to make them easier to write, through a concise language-integrated programming interface in Scala, a popular functional language for the JVM. (Interfaces in Java and SQL are also in the works.) In addition, Spark interoperates cleanly with Hadoop, in that it can read or write data from any storage system supported by Hadoop, including HDFS, HBase, or S3, through Hadoop's input/output APIs. Thus, Spark can be a powerful complement to Hadoop even for non-iterative applications.

Spark is open source at <http://www.spark-project.org> and is being used for data analysis both at Berkeley and at several companies. This article will cover how to get started with the system, how users are applying it, and where development is going next. For a detailed discussion of the research behind Spark, we refer the reader to our NSDI '12 paper [6].

Programming Interface

The key abstraction in Spark is *resilient distributed datasets* (RDDs), which are fault-tolerant collections of objects partitioned across cluster nodes that can be acted on in parallel. Users create RDDs by applying operations called *transformations*, such as `map`, `filter`, and `groupBy`, to data in a stable storage system, such as the Hadoop Distributed File System (HDFS).

In Spark's Scala-based interface, these transformations are called through a functional programming API, similar to the way users manipulate local collections. This interface is strongly inspired by Microsoft's DryadLINQ system for cluster

computing [5]. For example, the following Spark code creates an RDD representing the error messages in a log file, by searching for lines that start with “ERROR,” and then prints the total number of error messages:

```
val lines = spark.textFile("hdfs://...")
val errors = lines.filter(line => line.startsWith("ERROR"))
println("Total errors: " + errors.count())
```

The first line defines an RDD backed by an HDFS file as a collection of lines of text. The second line calls the `filter` transformation to derive a new RDD from `lines`. Its argument is Scala syntax for a function literal or closure. It’s similar to a lambda in Python or a block in Ruby. Finally, the last line calls `count`, another type of RDD operation called an *action* that returns a result to the program (here, the number of elements in the RDD) instead of defining a new RDD.

Spark lets users call this API both from stand-alone programs and *interactively* from the Scala interpreter to rapidly explore data. In both cases, the closures passed to Spark can call any Java library, because Scala runs on the Java VM. They can also reference read-only copies of any variables in the program. Spark will automatically ship these to the worker nodes.

Although simply providing a concise interface for parallel processing is a boon to interactive analytics, what really makes the model shine is the ability to load data in memory. By default, Spark’s RDDs are “ephemeral,” in that they get recomputed each time they are used in an action (e.g., `count`). However, users can also *persist* selected RDDs in memory for rapid reuse. If the data does not fit in memory, Spark will automatically spill it to disk, and will perform similarly to Hadoop. For example, a user searching through a large set of log files in HDFS to debug a problem, as above, might load just the error messages into memory across the cluster by calling:

```
errors.persist()
```

After this, she can run a variety of queries on the in-memory data:

```
// Count the errors mentioning MySQL
errors.filter(line => line.contains("MySQL")).count()

// Fetch the MySQL errors as an array of strings
errors.filter(line => line.contains("MySQL")).collect()

// Fetch the time fields of errors mentioning PHP as an array
// (assuming time is field number 3 in a tab-separated format):
errors.filter(line => line.contains("PHP"))
  .map(line => line.split('\t')(3))
  .collect()
```

In-memory data provides a significant speed boost for these queries. For example, in one test, we loaded a 50 GB Wikipedia dump onto a 20-node Amazon cluster, and found that a full-text search through it took 20 seconds with on-disk data or with Hadoop. The same search took only 0.8 seconds with an in-memory RDD.

Fault Tolerance

Apart from providing in-memory storage and a variety of parallel operators, RDDs also automatically recover from failures. Each RDD tracks the graph of transformations that was used to build it, called its *lineage graph*, and reruns these

operations on base data to reconstruct any lost partitions. For example, Figure 1 shows the RDDs in the last query above, where we obtain the time fields of errors mentioning PHP by applying two filters and a map. If any partition of a dataset is lost, e.g., a node holding an in-memory partition of errors fails, Spark will rebuild it by applying the filter on the corresponding block of the HDFS file. Recovery is often much faster than simply rerunning the program, because a failed node typically contains multiple RDD partitions, and these can be rebuilt in parallel on other nodes.

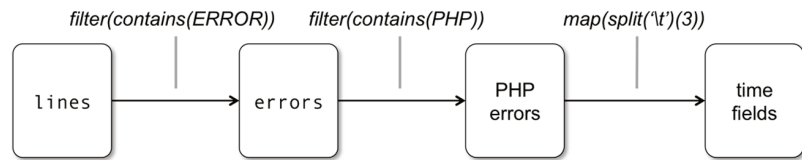


Figure 1: Lineage graph for the third query in our example. Boxes represent RDDs, and arrows represent transformations between them.

Lineage-based fault recovery is powerful because it avoids the need to replicate data. This saves both time in constructing RDDs, as writing data over the network is much slower than writing it to RAM, and storage space, especially for precious memory resources. Even if *all* the nodes running a Spark program crash, Spark will automatically rebuild its RDDs and continue working.

Other Examples

Spark supports a wide range of operations beyond the ones we’ve shown so far, including all of SQL’s relational operators (`groupBy`, `join`, `sort`, `union`, etc.). We refer the reader to the Spark Web site for a full programming guide [8], but show just a couple of additional examples here.

First, for applications that need to aggregate data by key, Spark provides a parallel `reduceByKey` operation similar to MapReduce. For example, the popular “word count” example for MapReduce can be written as follows:

```

val counts = file.flatMap(line => line.split(" "))
                .map(word => (word, 1))
                .reduceByKey((a, b) => a + b)
  
```

Second, to give an example of an iterative algorithm, the code below implements logistic regression, a common machine learning algorithm for classifying objects such as, say, spam vs. non-spam emails. The algorithm runs MapReduce operations repeatedly over the same dataset to optimize a mathematical function by gradient descent (specifically, to find a hyperplane that best separates the objects). Thus, it benefits greatly from storing the input data in RAM across iterations.

```

val points = spark.textFile(...).map(parsePoint).persist()
var w = Vector.random(D) // Current separating plane
for (i <- 1 to ITERATIONS) {
  val gradient = points.map { p =>
    (1 / (1 + exp(-p.y*(w dot p.x))) - 1) * p.y * p.x
  }.reduce((a, b) => a + b)
  w -= gradient
}
println("Final separating plane: " + w)
  
```

To show the benefit of Spark for this algorithm, Figure 2 compares its performance against Hadoop for varying numbers of iterations. In this test, with 100 GB of data on a 50-node cluster, Hadoop takes a constant time per iteration of about 110 seconds. In contrast, Spark takes 80 seconds for the first iteration to load the data in memory, but only six seconds for each subsequent iteration.

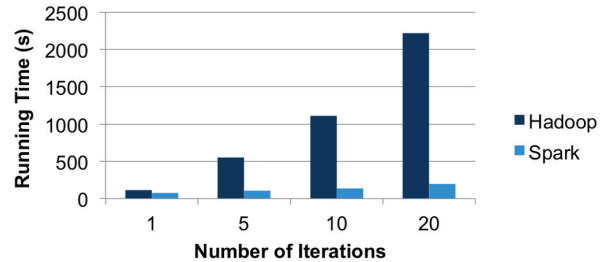


Figure 2: Performance of logistic regression in Hadoop vs. Spark for 100 GB of data on a 50-node cluster. Hadoop takes a constant time of 110s per iteration, much of which is spent in I/O. Spark takes 80s on the first iteration to load the data in memory, but only 6s per subsequent iteration.

User Applications

Spark is in use both at several Internet companies and in a number of machine learning research projects at Berkeley. Some of our users' applications are discussed below.

In-Memory Analytics on Hive Data

Conviva Inc., an online video distribution company, used Spark to accelerate a number of analytics reports that previously ran over Hive, the SQL engine built on Hadoop. For example, a report on viewership across different geographical regions went from taking 24 hours with Hadoop to only 45 minutes with Spark (30× faster) by loading the subset of data of interest into memory and then sharing it across queries [2]. Conviva is also using Spark to interactively search large collections of log files and troubleshoot problems, using both the Scala interface and a SQL interface we are developing (discussed in the next section) called Shark.

Interactive Queries on Data Streams

Quantifind, a startup that specializes in predictive analytics over time series data, used Spark to build an interactive interface for exploring time series. The system periodically loads new data from external feeds (e.g., Twitter streams), runs an entity extraction algorithm to parse the data, and builds an in-memory table of mentions of each entity. Users can then query this table interactively through a Web application that runs Spark queries on the backend [4].

Traffic Modeling

Researchers in the Mobile Millennium project at Berkeley [3] parallelized a learning algorithm for inferring traffic conditions from crowd-sourced automobile GPS measurements. The source data were a 10,000 link road network for the San Francisco area, as well as 600,000 position reports for GPS-equipped automobiles (e.g., taxi cabs, or users running a mobile phone application) collected every minute. By

applying an iterative expectation maximization (EM) algorithm to this data, the system can infer the time it takes to travel across individual road links.

Twitter Spam Detection

The Monarch project at Berkeley used Spark to identify link spam in Twitter posts. They implemented a logistic regression classifier on top of Spark, similar to the example in “Other Examples,” above. They applied it to over 80 GB of data containing 400,000 URLs posted on Twitter and 10^7 features/dimensions related to the network and content properties of the pages at each URL to develop a fast and accurate classifier for spammy links.

Conclusion and Next Steps

By making in-memory data sharing a first-class primitive, Spark provides a powerful tool for interactive data mining, as well as a much more efficient runtime for the complex machine learning and graph algorithms that are becoming common on big data. At the same time, Spark’s ability to call into existing Java libraries (through the Scala language) and to access any Hadoop-supported storage system (by reusing Hadoop’s input/output APIs) make it a pragmatic choice to complement Hadoop for large-scale data analysis. Spark is open source under a BSD license, and we invite the reader to visit <http://www.spark-project.org> to try it out.

Our group is now using Spark as a foundation to build higher-level data analysis tools. Two ongoing projects that we plan to open source in 2012 are:

- ◆ **Hive on Spark (Shark):** Shark is a port of the Apache Hive SQL engine to run over Spark instead of Hadoop. It can run over existing Hive data warehouses and supports the existing Hive query language, but it adds the ability to load tables in memory for greater speed. Shark will also support machine learning functions written in Spark, such as classification and clustering, as an extension to SQL [1]. An alpha release is available at <http://shark.cs.berkeley.edu>.
- ◆ **Spark Streaming:** This project extends Spark with the ability to perform online processing, through a similar functional interface to Spark itself (map, filter, reduce, etc. on entire streams). It runs each streaming computation as a series of short batch jobs on in-memory data stored in RDDs, and offers automatic parallelization and fault recovery for a wide array of operators. A short paper on Spark Streaming appears in HotCloud ’12 [7].

For more information on these projects, or on how to get started using Spark itself, visit the Spark Web site at <http://www.spark-project.org>.

Acknowledgments

Research on Spark is supported in part by an NSF CISE Expeditions award, gifts from Google, SAP, Amazon Web Services, Blue Goji, Cisco, Cloudera, Ericsson, General Electric, Hewlett Packard, Huawei, Intel, MarkLogic, Microsoft, NetApp, Oracle, Quanta, Splunk, and VMware, by DARPA (contract #FA8650-11-C-7136), and by a Google PhD Fellowship.

References

- [1] C. Engle, A. Lupper, R. Xin, M. Zaharia, M. Franklin, S. Shenker, and I. Stoica, “Shark: Fast Data Analysis Using Coarse-Grained Distributed Memory,” SIGMOD, 2012.
- [2] D. Joseph, “Using Spark and Hive to Process Big Data at Conviva”: <http://www.conviva.com/blog/engineering/using-spark-and-hive-to-process-bigdata-at-conviva>.
- [3] Mobile Millennium project: <http://traffic.berkeley.edu>.
- [4] K. Thiyagarajan, “Computing Time Series from Extracted Data Using Spark,” Spark User Meetup presentation, Jan. 2012: <http://files.meetup.com/3138542/Quantifind%20Spark%20User%20Group%20Talk.pdf>.
- [5] Y. Yu, M. Isard, D. Fetterly, M. Budiú, Ú. Erlingsson, P.K. Gunda, and J. Currey, “DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language,” USENIX OSDI ’08.
- [6] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica, “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing,” USENIX NSDI, 2012.
- [7] M. Zaharia, T. Das, H. Li, S. Shenker, and I. Stoica, “Discretized Streams: An Efficient Programming Model for Large-Scale Stream Processing,” USENIX HotCloud, 2012.
- [8] Spark homepage: <http://www.spark-project.org>.

When Disasters Collide

A Many-Fanged Tale of Woe, Coincidence, Patience, and Stress, Continued . . .

DOUG HUGHES



Doug Hughes is the manager for the infrastructure team at D. E. Shaw Research, LLC., in Manhattan. He is a past

LOPSA board member and was the LISA '11 conference co-chair. Doug fell into system administration accidentally, after acquiring a BE in Computer Engineering, and decided that it suited him.

doug@will.to

In my previous column we left off with the third of four major incidents. As a recap, we had a 160 TB storage backup server almost lose all of its data, a peculiar WAN failure involving some closely spaced and mismatched fiber-optic transceivers, and a flash SSD failure on a primary application server, all around the same time. I left a teaser about our big storage system. The story continues . . .

Issue 4

Prior to this issue happening, we had an unfortunate misunderstanding with the storage vendor about how we wished the metadata to be arranged. We wanted it mirrored across two RAM/flash devices in case one died. This required a policy-level declaration in the file system (GPFS) that had never been set! So, at the inception of this incident, we were several weeks into trying to correct this oversight by restriping all of the metadata back to spinning media in preparation for repairing the RAM/flash LUN setup. A restripe operation also takes care of ensuring appropriate metadata replication at the same time. Alas, going from RAM/flash to spinning media drops the IOPS by about 1000:1, so this is a tedious process and has many other impacts.

A little bit of background about the architecture is necessary to understand the failure that I'm about to describe. The storage "system" is made up of four Linux hosts which serve GPFS and NFS to clients and also talk to two large storage cabinets attached to a Fibre Channel SAN network. Everything else I will be describing lives in these large cabinets (except for the RAM/flash devices described earlier; those are separate). The first storage cabinet has two storage controllers connected to a number of disk shelves. Each shelf is divided into a left half and a right half. The storage controllers are each directly connected to all shelves. Controller 1 normally services the left half and controller 2 normally services the right half of each shelf. If one of the storage controllers dies, the other controller will be able to service the disks. The second cabinet has an identical set of shelves but no head controllers. Each shelf in the second cabinet is chained to the corresponding shelf in the first cabinet (i.e., cabinet *A* first shelf is chained to cabinet *B* first shelf, and so on). The storage controllers each have five, two-channel SAS cards. Each SAS channel (port) connects to an I/O module serving one-half of a storage shelf. Each storage controller also has four Fibre Channel ports for talking to the Linux hosts, for a total of eight optical ports (A and B path) across the two storage controllers. Since we have four Linux hosts with two Fibre Channel ports, each host is connected to both controllers simultaneously using direct-attach point-to-point

Fibre Channel links. Every host and controller can communicate on all channels (SAS and FC) simultaneously for purposes of throughput, balancing, and failover.

Each storage shelf (holding all of the disks) has two I/O cards, one of which is an A path, and the other of which is a B path. All SAS and Fiber Channel paths are active at all times. Internally stored configuration and physical cabling together control which paths are in use as primary and which are secondary, and the primary/secondary paths are alternated to achieve the best balance.

Here is a summary of various failure scenarios:

- ◆ GPFS Linux server fails: NFS and GPFS failover to another of the three directly attached servers.
- ◆ A Linux/controller Fibre Channel cable or card fails: a path to that controller is disabled on that host, and three other paths are available to that controller. Half the capability of a given Linux node is disabled, but the other FC path remains active.
- ◆ A storage controller fails: half of the paths to storage are lost. Uncommitted blocks may be lost. Disk commit journals are lost (see below) because journals are kept per storage controller for the RAID groups for which it is defined as the primary.
- ◆ A storage controller SAS card fails: half the storage in two shelves is inaccessible, because it is a two-port card. Because the storage controller itself is still alive, a failover for these two paths does not occur to the other controller. Only if this controller totally fails does the second controller take control of all paths. It's all or nothing. The controller also keeps journals of RAID logical units (LUNs [5]) to replay when the card is replaced. The LUNs are RAID-6 [4] (double parity), so everything keeps running even though half of the RAID-6 LUNs could be degraded by two disks. Another single disk loss in the wrong stripe would cause data loss.
- ◆ A storage shelf card fails: half the disks in that shelf are unavailable. Half the logical units are down (one disk out of two). Storage controller journals mark blocks to rebuild on the missing disks once the card is replaced.

A commit journal is a journal kept on each of the disk storage controller nodes that keeps track of the state of the dual parity blocks in that system. If an I/O card (on the controller or in the shelf) fails, the journal will buffer RAID information so that a full rebuild is not necessary when the card is replaced; only blocks that have been written while the disks have been unavailable will need to be affected. Also, if a disk is pulled from the system and put back in, the storage controller will recognize the disk as one of its own and will rebuild the blocks of data that it missed while it was unplugged, allowing for a fast rebuild on insert. This is important, and I'll explain why in a few paragraphs.

The Incident

Sometime in the midst of the WAN outage and just prior to the backup server outage, one of our storage controller heads had an incident. It was a lot like a seizure. The storage system got very slow. We saw lots of disturbing messages on the console about timeouts and retries of various components, and disks started to disappear from the storage controller view. Eventually, a storage controller timed out two of the disk channels and all of the disks attached to that storage controller node on those two channels. This made half of our RAID-6 LUNs degraded by two disks. (Half because the other controller was fine, and each controller is primary

for half of all disks, as explained above.) The normal procedure here is to swap out the controller I/O module, which we did, and then reset the controller. Unfortunately, this failed. More serious messages happened. We got the vendor on the phone, and they suggested we leave the storage controller off while they shipped us a replacement. They shipped overnight. We were degraded, but we could wait. It wasn't a comfortable wait, given that we were down by two disks in many RAID-6 LUNs, and one more disk failure out of about 450 meant lots of lost data, but we had little choice. At the time, what we didn't realize was that we lost all of the journals for this storage controller too, or we would have been all the more apprehensive.

The vendor was able to beat the 4 p.m. shipping deadline for priority overnight, and we had the replacement storage controller the next morning. We swapped out the controller in about an hour (they have a lot of cables, and they need to be attached to the right places!) and started the rebuild process. It was then that we realized that things were worse than they had seemed: the journals were gone and we would have to rebuild 38 16TB logical units and a number of smaller ones. Worse, with this generation of storage controller node, only one disk out of each pair (two were missing) could be rebuilt at a time, and only six rebuilds could run in parallel per storage controller head node. This was a week-long series of rebuilds where any single drive failure in any one of the logical units would result in permanent data loss at a massive scale.

Then the second controller failed . . .

Appallingly, while the first storage controller was rebuilding, within two hours of its being replaced, the second storage controller failed! Luckily, this failure did not start with I/O modules failing (for some value of lucky). This failure resulted in losing two disks on every other RAID-6 stripe in the system because of the channel failures to those shelves. So, now we are down two disks in every RAID-6 stripe in the storage system! Previously, it was only half of the stripes.

After replacing the second storage controller node, we started getting more errors on the same two channels that had failed in the first place. The vendor, now on high alert, advised us that we should power cycle the two storage shelves full of disks to reset their I/O modules. Knowing that we could not tolerate another single disk loss without data loss on a very large scale (because data blocks are striped across all RAID-6 logical units by GPFS), this operation had to be done exactly right. It's worth noting that at some point in the past, the entire cabinet full of disks had been moved from another building, and some of the power cables were mislabeled. To be specific, the labels were correct at the time, but they had been attached to the wrong chassis on re-cabling. The person who was cycling the power looked at the label, flicked the power switch, and hysterics ensued. I was watching the console and started seeing a continuous stream of errors on the third channel, muttered an expletive, and took the extreme action of doing a forced shutdown of all four Linux nodes to try to preserve file-system integrity. Had the technician looked at the overall situation carefully instead of in a hurry, he would have noticed that the disk shelf he was power cycling could not have been either of the shelves in question. The only ones that should have been cycled were the last two in the system, the bottommost two. The shelves are identified in such a way that this is a fairly easy thing to double check. He had turned off the third from the bottom. OUCH.

Since the replacement of the second controller, we had been running the file system un-shared and un-mounted. Normally, it serves around 640 TB of primary chemistry data to analysis clusters and serves as the primary write target for the

chemistry supercomputers, but we wanted to get things stable. After the unfortunate cycling of the third shelf, and the emergency shutdown of the file system, we started bringing things up carefully. First the shelf was turned back on. Then we started the Linux storage servers one at a time to make sure they could see the disks, and then we mounted the cluster file system one node at a time. So far, so good. The file system was mountable, basic commands like `ls` and `df` were okay, and there didn't seem to be any permanent harm done. However, now we had to deal with the repercussions of the loss of the third disks in many stripes. And we dreaded that we would have to run an `fsck`!

Thus, we got the vendor on the phone again, although this wound was self-inflicted. The I/O card in the storage shelf was replaced, channel messages were no longer occurring, and things were generally stable with a lot of broken disks. Here's what we knew at the time:

- ◆ All RAID-6 stripes were now running bare-minimum.
- ◆ One more disk failure out of a little over 800 would result in a lot of data loss.
- ◆ Half of the disks marked failed did have RAID journals, so recovery should be relatively quick for those.
- ◆ Many of the remaining disks had been in low-power mode (MAID [1]).
- ◆ The emergency shutdown might require an `fsck`.

Our data flow is such that the current working set of chemistry data is typically data that has been written in the last month. Any files that have not been modified in two months and accessed in three months are migrated to MAID storage, and those disks spin to low power mode to save energy. These migrations happen every Sunday. This is important because we know that these disks were not being written during the failure, so it should be possible to mark them as "okay."

First, we targeted the low-hanging fruit. Step 1: run the command to rebuild all the LUNs with journals. This brought half of the disks online in about two hours. This was a good start! At this point we decided to bring the file system online. The supercomputers, expensive resources, were stalled at not being able to write data anywhere, and the top priority was to have them running again. So we took the risk and brought the file system online so it could start collecting data again. We put network quality of service (QoS) policies in place to throttle any analysis jobs (NFS reads) from overwhelming the file system. We were running severely degraded, with each storage controller rebuilding the maximum of six logical units simultaneously, albeit quite slowly. We knew this was going to take one to two weeks to complete. We contemplated bringing other storage online in place to collect the partial writes and then moving things over, but a lot of things are more ingrained that they ought to be. This would have been a time-consuming job with a lot of logistics and double changes followed by data migration, and time was not exactly an abundant resource.

Next we targeted the MAID RAID-6 stripes. There is an undocumented command that can be used, under strict vendor supervision, to mark a 10-disk stripe unit as okay as long as you are absolutely positive that nothing has been changed. If you use this command and are wrong, the consequences are rather dire. So, like Henry V unto the breach, we charged, marking all of them as good! Except there was a problem. The vendor informed us that, according to the data parity logs, something *had* been written there! We pondered this for a while. There couldn't have been anything written to there, could there? After some thought we concluded that what happened was that during the emergency shutdown initiated by the third shelf loss

(see OUCH, above), the Linux nodes tried to shut down the file system gracefully, instead of just powering off the hosts immediately. This initiates a GPFS shutdown which will flush any superblock updates and other bookkeeping data.

We theorized that the master RAID-6 block records on the MAID disks got updated during this event. Luckily, there's an undocumented way to fix this as well. The vendor had us run several commands that generated 128,000 rows of data about disk blocks on the suspected LUNs for each storage controller. They analyzed them and gave us a series of commands to run to regenerate the parity for each of these blocks. This is basically the equivalent of telling it to rebuild the parity on this disk from all of the others, and just assume that everything is good. With a little help from Expect [2], this took another couple of hours. Finally, we could mark the stripes as OK and that left just ~30 LUNs to rebuild the hard way. But we could do this while the data continued to roll in and out.

So we continued to operate, QoS throttles in place (adjusted as needed), partially degraded, and still hadn't run the fsck, yet. We waited for the full rebuild to complete, which took five days for the first disk in each RAID-6 group, and another five days for the second disk. A very large sigh was had after the first five days concluded and all LUNs were only degraded by one disk each. Ideally, the storage controller would be a little bit smarter about this and build both RAID-6 parity blocks simultaneously to economize the presence of data already in memory. Actually, this should be required in a disk storage architecture, but I know there are many instances out there of similar behavior.

Finally, all of the RAID parity had been rebuilt; it was time to run the fsck, just to make sure everything was okay. So, we started an online fsck. After about 10 hours of running, it reported that it had encountered a series of errors requiring an offline fsck! This was not heartening. A number of things about this were bad:

1. It was somewhere in phase two of multiple undisclosed phases.
2. The fsck had no estimate of how much time was left to go. We had no bounds of what to expect from an offline fsck and how much downtime would be required.
3. We'd had enough downtime.
4. The messages pointed to some kind of a mismatch in a file that appeared to be perfectly fine when examined by hand.

So we contacted the vendor again. They suggested we run it in offline mode, as indicated. "@\$(&% no," we said. We can't afford to be down for an undisclosed number of further days. What does this error mean? They had to consult with IBM about the message that we received. The next day we had an answer. It turned out that this error was an internal conflict of some kind. They suggested that we update to the next release of GPFS and try again, since it was an identified bug. We did this. Rolling upgrades in GPFS are fairly easy and require no downtime. We started the fsck again and the error was gone! The fsck ran successfully and without incident; it took 18 hours. Where did this leave us? Unfortunately, we were still without mirrored metadata. However, as I write this article that situation has finally been corrected by the installation of a larger flash media device pair capable of holding all metadata with room to grow. (The mirroring to the new devices took less than a day!)

Ponderables

- ◆ Explore all failure modes of your storage. Some may be surprising.
- ◆ Pay attention to your hardware architecture. Play “what if” scenarios.
- ◆ Insist on on-site spares for components that can leave you in severely degraded states. In our case, we had spares of disks, I/O modules, and power supplies, but the most serious loss is when a storage controller head gets lost after losing an I/O module. We did not have this spare. (We do now!)
- ◆ Demand explanation on error messages that you don’t understand, that appear to be contradictory, or that seem patently false.
- ◆ Never defer label maintenance on critical systems!
 - ◆ Correct labels are imperative in a crisis.
 - ◆ Don’t trust that labels are correct when getting it wrong could mean catastrophe. If the person powering down the shelves under this stressful condition had used a sanity check, he would have realized that he was power cycling the wrong storage shelf.
- ◆ Router/switch-based throttles and QoS are useful for asset protection. Use them to limit maximum bandwidth to clusters when needed.
- ◆ Sometimes it’s better to keep the full implications of how close one has been to disaster close to the vest until the worst is over. Worrying about things that are outside of one’s control can cause a lot of stress. On the other hand, some management groups insist on knowing the worst case scenario, so withholding in those situations can be “bad for your career.” This is a tough call. I simply told management that things were quite dire, while not sharing that it was a 10 out of 10 sort of situation until afterwards.
- ◆ At some point, most people need to make a call about whether to leave storage online in degraded mode while the problem is being debugged—so people can get work done—or to take it offline if it is safer to do so. These are always excruciatingly hard decisions. I have decided in each direction at different times. May you never have to make this choice in your career! This is a decision where management backing is very helpful, but not always available (weekends, holidays, vacations, the middle of the night, etc.).
- ◆ You’ve tested your backups recently, right? You have, HAVEN’T YOU!?
- ◆ Is your tape data something that can be restored from easily?

Finally, the trials were over. Feeling some empathy for Job [3], we heaved a collective sigh of relief. Looking back over the whole series of unfortunate events (with apologies to Lemony Snicket), it seems highly improbable that they were unrelated, but they were. One is tempted to think of things like localized power disturbances, or other things, but some of the incidents were in completely different buildings, involving an entirely different electrical service substation from the utility, and different power sources and high voltage feeders.

One final note on backups and restores. During the heart of the incident, one member of the team was working on restoring the most recent working set of data—the last two months—to an alternate server in case the worst happened and we had to zero out the entire file system from scratch and restore from tape. This was a frighteningly real scenario, so he was testing out the most optimal way to get the data from tape. This is one of those cases where I’m sure most of us have read advice about testing our restores, and most of us back-burner this because there always seem to be higher-priority things. You’ve heard it before, you’ll hear it again:

go do it. If you need a full day, show this horror story to your management and get them to hold off the customers while you complete this important task.

We're still testing various alternatives to make retrieval of the most recent data as efficient as possible, which means as few tapes as possible. We are currently at a 2:1 overhead. This means to retrieve N TB of data, we need 2*N tapes, which isn't terrible, but definitely has room for improvement.

May your transceivers be well-matched, your RAID controllers have sufficient redundancy, your journals be recoverable, your power be reliable, your flash devices be well behaved, and your users be tolerant and patient. I'd say "may your restores be fast and optimal," but I think you'll better appreciate the sentiment of: may your tapes be write-only.

References

[1] MAID: http://en.wikipedia.org/wiki/Massive_array_of_idle_disks.

[2] Expect: <http://www.nist.gov/el/msid/expect.cfm>.

[3] Job: http://en.wikipedia.org/wiki/Book_of_Job.

[4] RAID-6 is dual-parity, allowing the failure of three disks before data is lost.

[5] LUN: a logical partition of a disk or multiple disks to provide aggregation and/or partitioning that can be used with permissions models to enable or restrict access to storage. http://en.wikipedia.org/wiki/Logical_Unit_Number.

For part 1 of this article, see the June 2012 *login*: page at <https://www.usenix.org/publications/login/june-2012-volume-37-number-3>.

Practical Perl Tools

Taking the XPath Less Traveled

DAVID N. BLANK-EDELMAN



David 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 my last column I made a statement about liking XPath so much that I just might be tempted to make it the subject of the next column. I've decided to make good on that threat and talk a bit about XPath and how its power can be harnessed from Perl. One strange thing about this choice is it will force the column to include more details about another language that isn't Perl. Hopefully, you, faithful reader, are willing to hang with that. The plus is the language in question is XPath, something you'll be able to use with Perl or virtually any other language that is currently wetting your whistle.

What's XPath and Why Do I Care?

So what is XPath? I think you'd be a bit hard-pressed to figure out the answer to this question, and why XPath is so cool, if you just read the introduction to the W3C's technical report on the subject ("XML Path Language (XPath), Version 1.0"):

XPath is the result of an effort to provide a common syntax and semantics for functionality shared between XSL Transformations and XPointer. The primary purpose of XPath is to address parts of an XML document. In support of this primary purpose, it also provides basic facilities for manipulation of strings, numbers and booleans. XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values. XPath operates on the abstract, logical structure of an XML document, rather than its surface syntax. XPath gets its name from its use of a path notation as in URLs for navigating through the hierarchical structure of an XML document.

Let me try to provide a different, pragmatic, and perhaps overly blunt explanation of what XPath is and why you might care, as seen from the porthole of a Perl programmer. XPath provides a concise, elegant syntax for referencing a specific thing or things in a document. Let's say you want your program to retrieve the text of the third headline in a document. Or maybe you need to extract the weather forecast part of an XML document (as we did in the last column). Both of these things are pretty easy to do with the XPath syntax, provided your document is marked up well. XPath was originally invented to interact with XML documents, but it works quite well for well-formed HTML documents as well.

To understand how XPath works, there's one very important idea you must wrap your head around: documents as trees. And just to be clear, when I say "trees" I don't mean the "I think that I shall never see. A poem lovely as" kind of tree. I'm

referring to the computer sciency, data structure/hierarchical organization like “directory tree” kind of tree. Going from an XML or HTML document to a tree structure may require a bit of squinting and pondering if you’ve never heard of this idea before, but let me see if a couple of examples might help. Take for instance the following XML document:

```
<poem>
  <poet>
    <name>Joyce Kilmer</name>
    <born>1886</born>
    <died>1918</died>
    <movement locale="American">modernist</movement>
  </poet>
  <title>Trees</title>
  <excerpt>I think that I shall never see
    A poem lovely as a tree.
  </excerpt>
</poem>
```

In XML-speak, the “root” element of this XML document is the `<poem>` element. And when I say “element” in this case I’m referring to the `<poem>` tag and everything inside it up until the closing `</poem>` tag. The `<poem>` element has three sub-elements, namely `<poet>`, `<title>`, and `<excerpt>`. Within `<poet>`, there are four sub-elements: `<name>`, `<born>`, `<died>`, and `<movement>`.

So far, so good, right? Okay, let me turn this document into a tree for you by massaging that last descriptive paragraph a bit: At the top of the tree (at the root node) is the `<poem>` element. It has three child nodes: `<poet>`, `<title>`, and `<excerpt>`. One of these nodes, `<poet>`, has four children of its own: `<name>`, `<born>`, `<died>`, and `<movement>`. Tah-dah. It’s a tree.

How would this work with an HTML document? Same sleight-of-hand applies. Take for example this document:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Joyce Kilmer</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  </head>
  <body>
    <h1>Trees</h1>
    <p>I think that I shall never see...</p>
  </body>
</html>
```

I could start speaking in tree-language and say something like: the root node of the document is the `<html>` element. It has two children, `<head>` and `<body>`. The `<head>` child has two children of its own, `<title>` and `<meta>`. Similarly, the `<body>` node has an `<h1>` element and a paragraph element (`<p>`) as its children.

Once we’re certain that we’re dealing with data that is in a tree structure, then doing all of the things we talked about before having to do with referencing parts of the document or extracting certain bits all become tree operations. We need a way to take a walk up and down the tree structure, find certain elements in the tree, extract parts of the tree, and so on. And that’s where XPath comes in.

Before we actually see any XPath syntax, I feel that I must confess that we're only going to stick with the simple, but highly useful, parts of XPath in this column. Let me unburden my soul by mentioning the syntax we'll be seeing will be all XPath 1.0 syntax, even though a 2.0, and a 2.0 schema-aware specification (both not nearly as widely implemented as 1.0) exist. XPath lets you specify things using an abbreviated or an unabbreviated form; we're going to use the former. XPath defines eight separate ways to specify the direction a parser should go when walking the tree (called an axis), but we're not going to use most of them in this column (and you may find you never use most of them in real life either). All of this is just to say that there's considerable depth to explore when dealing with XPath. Consider doing more reading on the subject if you decide XPath is a tool you'd like to use with proficiency.

XPath Path Time

Time to put the "Path" in XPath. For these examples let's use the sample XML document from above so we have something simple for demonstration purposes. As a first test, let's start by figuring out how to reference the <movement> element. In XPath we walk down the tree from the root node, just like you would if you were dealing with a file system path. The resulting XPath specification is:

```
/poem/poet/movement
```

Yup, it is that simple to reference the <movement> element. Note: this lets us point to that node in the tree. If we want the textual contents of that node (i.e., "modernist") we would add the text() function to our specification (as in "/poem/poet/movement/text()").

Let's follow the file system path analogy a little farther. Like most common file-system semantics, you can also use the . (dot) and .. (dot-dot) characters to refer to relative places in the tree. A dot by itself refers to the current node, while dot-dot refers to the parent node. Given our example document:

```
/poem/poet/death/..
```

refers to the <poet> element. That example may be a bit obtuse because it isn't clear why you'd want to go that deep in the tree before coming back up a level. We'll see when you might want to do this in a moment once we discuss a few other path-related items. The first is the use of an at (@) character in a path. The @ gets used to refer to an attribute of an element. So if we wanted to access the "locale" part of <movement locale="American"> we could write:

```
/poem/poet/movement/@locale
```

The last part of the path syntax I want to mention also has its roots in file system syntax. XPath lets you include wildcards in your expressions, so I could write:

```
/poem/poet/*
```

to refer to all of the child nodes of the <poet> element or even something like:

```
/poem/*/born
```

to reference all of the nodes that have a <born> sub-element. Or I could write:

```
/poem/poet/*/@locale
```

to find all of the elements under <poet> that have locale attributes.

Or even:

```
/poem/poet/movement/@*
```

to reference all of the attributes of the <movement> element.

This globbing-like syntax is similar to the way we use globbing with file systems. With a file system, if you say *.txt, you expect to get back a list of files that end in .txt. With XPath, when you use a wild card, your program gets back a list of nodes that match the specification. I'll be sure to show you an example of this in Perl so you can see how we might handle dealing with a list of nodes (foreshadow: the same way we deal with any other list in Perl).

There's one more very important path-related piece of syntax to show you that is kind of like globbing, only cooler. It doesn't have a direct file system analog, although I sometimes wish it did. If you use two slashes (//) in your specification, an XPath parser will start at that place in the tree and walk down the tree until it finds a match. Our tree isn't particular deep or complex, so examples of this may seem a bit weak. But when you start dealing with more complex documents, it can be awfully convenient to tell the parser "Go find this specification any place in your tree" without having to do that searching yourself. For our document, we could write something like:

```
//died
```

and it would walk down from the root until it found that node deep in the <poet> element. The double-slash notation also works in the middle of a specification, as in:

```
/poem//@locale
```

Remember the mention above about the "dot-dot" operator? Here's one place where it makes more sense. If we wanted to extract the textual contents of all of the nodes that contained a @locale attribute (vs. just referencing the locale attributes themselves as we just wrote), we could write:

```
/poem//@locale../text()
```

An XPath parser would return the string "modernist" if asked to parse our current sample document with this specification.

XPath has a ton of other "navigational" filigrees that let you say things such as "move to the next sibling node in the tree," but that gets you into talking about the different XPath axes (plural of axis). Rather than bore you with any more of them, there's just one more concept we should discuss before we actually write some Perl code.

XPath Predicates and Functions

You may not have noticed, but something about our sample document has made our XPath specifications easier than they might ordinarily be. Every one of our elements has been unique at its level of the tree. For example, there's only one <poem> element. That seldom happens with real documents. Most documents contain more than one of a certain piece of information. Let's construct a new sample that incorporates our previous sample so we have something more interesting to work with. Imagine we had a new document with this in it:


```

<anthology>
  <poem>
    <poet>
      <name>Joyce Kilmer</name>
      <born>1886</born>
      <died>1918</died>
      <movement locale="American">modernist</movement>
    </poet>
    <title>Trees</title>
    <excerpt>I think that I shall never see
      A poem lovely as a tree.
    </excerpt>
  </poem>
  <poem>
    <poet>
      <name>Ogden Nash</name>
      <born>1902</born>
      <died>1971</died>
      <movement locale="American">light verse</movement>
    </poet>
    <title>Song of the Open Road</title>
    <excerpt>I think that I shall never see
      A billboard lovely as a tree.
    </excerpt>
  </poem>
</anthology>

```

Given this document, how do we walk down the tree? We can start off with “/anthology”, but then how do we tell the parse which <poem> element we want to reference? This is where XPath predicates come into play. XPath predicates are specified using square brackets right at the decision point in the path. XPath has a whole bunch of predicates, the simplest of which is simply to use the number of the branch, starting at the number 1 (yup, it is 1-based not 0-based as in Perl):

```
/anthology/poem[2]/poet/name/text() # Ogden Nash
```

You can also use string comparison predicates against attributes and text contents of nodes. If we wanted to find the names of all of the poets who died in 1971, we could write something like this:

```
//died[text()='1971']/../name/text() # Ogden Nash
```

This example uses a combination of the different things we’ve covered up until this point, so let me go over it once just so it is clear. The double slash at the beginning says “walk down the tree until you find a match for what follows.” In this case, it walks down until it finds a node called <died> whose contents equal 1971. When it finds a node that matches this condition, it walks back up one level to find a <name> element and returns the textual contents of that <name> element. Given our document, this will return “Ogden Nash.”

If we wanted to return the names of all of the American poets, we could write:

```
//movement[@locale="American"]/../name/text() # Joyce Kilmer & Ogden Nash
```

and we would get back two strings: “Joyce Kilmer” and “Ogden Nash.”

XPath also has a bunch of functions you can use. For example, if we wanted to know how many American poets were in the document, we could instead write:

```
count(//movement[@locale="American"]) # 2
```

So far I've just been describing the XPath syntax without providing much commentary, so let me be sure my bias is clear. I *love* this language. It may just be that my brain is wired strangely, but I find the path analogy lets you write concise and elegant specifications for document references and document extraction queries.

Forget Anything?

Given that this is a Perl column, I'm pretty sure I'd be remiss if I didn't include any Perl code. Let me warn you ahead of time, the Perl samples we're about to see won't themselves be anything exciting. This is not because Perl isn't exciting (hey, hey, quiet down, peanut gallery), but it is because all of the magic super powers reside in the XPath language itself. Perl programs that use XPath largely get to say, "Here's a document. Here's an XPath specification. Go to town and hand me back the results when you are done."

There are a number of different Perl modules that understand XPath or a reasonable subset of it (in fact, in the last column, I mentioned `Class::XPath`, which lets you graft on an XPath-lite interface to an object tree of your own making). Even though there is actually an `XML::XPath` module (last touched in 2003), the two modules I use for XML and HTML XPath parsing are `XML::LibXML` and `XML::Twig`. I've also used `HTML::TreeBuilder::XPath` for my HTML parsing. For our big anti-climactic use of XPath in Perl, we'll use `XML::LibXML`. Here's some code that returns all of the American poet names from our document:

```
use XML::LibXML;

my $prsr = XML::LibXML->new();
$prsr->keep_blanks(0);

my $doc = $prsr->parse_file('poetry.xml');

foreach my $tnode (
    $doc->findnodes('//movement[@locale="American"]/./name/text()')
) {
    print $tnode->data . "\n";
}
```

First we load the module and initialize a parser object. We tell the parser object it should feel free to discard any "blank" nodes it would create during parsing (i.e., a node that gets created from whitespace). The parser gets pointed at our document, and `XML::LibXML` goes to work parsing it and bringing it into memory. At this point we can execute the XPath query we described above using the `findnodes()` method. This method performs a query and returns a list of nodes returned by that query. We iterate over each returned node (we're going to get back a list of text nodes holding the textual contents of each element), finally printing out the data in the node. It prints:

```
Joyce Kilmer
Ogden Nash
```

as expected. Code that queries the value of an element's attribute looks quite similar:

```

use XML::LibXML;

my $prsr = XML::LibXML->new();
$prsr->keep_blanks(0);

my $doc = $prsr->parse_file('poetry.xml');

# yes, this can be written as just //@locale, but I think it is good
# form to specify a bit more context so it is clear which elements'
# attributes you are targeting
foreach my $attrib ( $doc->findnodes('//movement/@locale') ) {
    print $attrib->value . "\n";
}

# output:
# American
# American

```

XML::LibXML also has a `find()` method that can be used to execute XPath queries that don't return nodes. This would be used for something like retrieving the result of the `count()` function example from above:

```

use XML::LibXML;

my $prsr = XML::LibXML->new();
$prsr->keep_blanks(0);

my $doc = $prsr->parse_file('poetry.xml');

print $doc->find('count(//movement[@locale="American"])'), "\n"; # 2

```

That's mostly all there is to it—feed the right XPath 1.0 specification to either the `findnodes()` or `find()` method and cope with what is returned.

And with that, I think it is time to bring this column to a close. Hopefully, having gotten a taste of XPath, you're going to rush right out to try it from Perl. Take care and I'll see you next time.

Import That!

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>). He is based in Chicago, where he also teaches a variety of Python courses. dave@dabeaz.com

I have a small confession: I often think that I don't fully understand the Python `import` statement. As a seasoned Python veteran, it might be surprising to admit something like that, but I think I might be in good company—most of the Python programmers I meet are just as mystified by some of the inner workings of it as I am.

Obviously, the `import` statement is one of the most important parts of Python. You use it to load standard library modules and third-party extensions. It's at the heart of various Python packaging and distribution tools. However, it's also the bane of anyone who needs to set up and maintain a custom Python installation (or anyone who has had to debug a broken setup).

In this article, I'm hoping to go behind the scenes and explain some of the mysteries surrounding module imports with the hope that it might be useful for anyone who has deal with setting up or debugging their Python installation.

Import Basics

As you know, the `import` statement is used to load library modules. For example:

```
import math
```

Modules define namespaces, so to access anything contained inside, you have to use the module name as a prefix. For example:

```
x = math.sin(2)
y = math.cos(2)
```

Alternatively, you can use the `from module import` form of import:

```
from math import sin, cos
```

This loads the module the same as before, but makes references to the listed symbols in the namespace of the code making the import. You can use those symbols without a prefix. For example:

```
x = sin(2)
y = cos(2)
```

Various details about *using* the `import` statement can be found in most Python books. That's not our focus here—instead, let's peel back the covers and go a bit deeper into its inner magic.

What Can Be Imported?

When you use a statement such as `import spam`, Python looks for one of three things. First, it checks for the existence of a package. A package is a directory with the same name as what is being imported and which contains an `__init__.py` file. On import, the `__init__.py` file executes.

If a package can't be found, the interpreter checks for a compiled C/C++ extension module in the form of a shared library or DLL. Depending on platform, this is typically a file such as `spam.so` (UNIX) or `spam.pyd` (Windows). Many of Python's standard library modules are actually compiled C code (math, time, etc.).

If a package directory or extension module can't be found, an attempt to load a simple Python source file is made. For this, Python first checks for a `.pyc` file containing compiled bytecode. If the format of the `.pyc` file matches the correct Python version and the timestamp is newer than the corresponding `.py` file, then the `.pyc` is loaded. Otherwise, the `.py` file is loaded and the matching `.pyc` file is overwritten with a more up-to-date version. It should be stressed that `.pyc` files are mostly a detail that you don't need to worry about. They are automatically created and managed by Python behind the scenes.

There is an underlying precedence for packages, extensions, and source files that happen to share the same name. For example, if you had a package `spam` and a source file `spam.py`, using `import spam` will always load the package and ignore the source file. Similarly, a C extension `spam.so` takes precedence over a source file `spam.py`.

It's All About the Path

To find packages, extensions, and modules, Python relies on the setting of the `sys.path` variable. You should launch Python on your machine and take a look at it:

```
>>> import sys
>>> sys.path
['', '/usr/local/lib/python2.7/site-packages/setuptools-0.6c11-py2.7.egg',
 '/usr/local/lib/python2.7/site-packages/pip-1.1-py2.7.egg',
 '/usr/local/lib/python2.7/site-packages/python_dateutil-1.5-py2.7.egg',
 '/usr/local/lib/python2.7/site-packages/pandas-0.7.3-py2.7-macosx-
10.4-x86_64.egg',
 '/usr/local/lib/python2.7/site-packages/tornado-2.1-py2.7.egg',
 '/usr/local/lib/python2.7.zip',
 '/usr/local/lib/python2.7',
 '/usr/local/lib/python2.7/plat-darwin',
 '/usr/local/lib/python2.7/plat-mac',
 '/usr/local/lib/python2.7/plat-mac/lib-scriptpackages',
 '/usr/local/lib/python2.7/lib-tk',
 '/usr/local/lib/python2.7/lib-old',
 '/usr/local/lib/python2.7/lib-dynload',
 '/Users/beazley/.local/lib/python2.7/site-packages',
 '/usr/local/lib/python2.7/site-packages']
>>>
```

At a simple level, `sys.path` is simply a list of directory names. When you type `import spam`, Python starts scanning the list looking for the first match. That is,

each directory on the path is checked for a package, extension module, or simple source file that matches the import name.

Close inspection of `sys.path` reveals a bit more, however. For example, in addition to directories, you will often see `.egg` and `.zip` files. When a `.zip` file is added to the path, Python treats its contents as a logical directory. That is, matching files will be transparently loaded from a zip archive as if the `.zip` file had been unzipped into an actual directory. Files with an `.egg` type are typically associated with third-party libraries and add-on packages. An egg is either a directory or a `.zip` file in disguise, with some extra metadata stored inside. If you examine the contents of an egg, you'll find a directory `EGG-INFO` that has this information.

Almost all of the problems related to managing a complicated Python installation are due to issues concerning the setting of `sys.path`. For instance, if the `import` statement fails to load code, it means that code is located in a directory not listed on the path. If you are getting bizarre name clashes or Python is using the wrong version of a module, it also points to some issue concerning the path. If you tried to install a third-party package but the code doesn't work, you might have installed it into the wrong version of Python. The bottom line is that spending a few minutes to deconstruct the `sys.path` is a worthwhile experiment if you want to better understand how Python installs itself on your machine.

Adjusting the Path

The Python import system path is something that can be adjusted. Since `sys.path` is a list, you can simply change its contents to any set of directories that you wish. For example, you can write code that inserts new directories:

```
import sys
sys.path.insert(0, "/some/new/directory")
```

Even though you can do this, don't! In fact, writing code that manually tweaks the import path is the first step towards a future of endless installation sorrow. In part, you don't want to be writing code that directly messes around with hard-coded directories and file names like this (what if your code gets moved somewhere else?). Second, it can be hard to predict how your changes to the path will interact with other parts of Python (other libraries, frameworks, package managers, and so forth). Frankly, you're often best off leaving the path alone.

A somewhat more sane way to alter the path is to set the `PYTHONPATH` environment variable prior to running the interpreter. For example:

```
bash % env PYTHONPATH=/some/new/path:/another/path python
Python 2.7.1 (r271:86832, Feb 27 2011, 11:47:28)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.path
['',
 '/usr/local/lib/python2.7/site-packages/setuptools-0.6c11-py2.7.egg',
 '/usr/local/lib/python2.7/site-packages/pip-1.1-py2.7.egg',
 '/usr/local/lib/python2.7/site-packages/python_dateutil-1.5-py2.7.egg',
 '/usr/local/lib/python2.7/site-packages/pandas-0.7.3-py2.7-macosx-
10.4-x86_64.egg',
 '/usr/local/lib/python2.7/site-packages/tornado-2.1-py2.7.egg',
```

```

    '/some/new/path',          # Note new paths added here
    '/another/path',
    '/usr/local/lib/python2.7.zip',
    '/usr/local/lib/python2.7',
    '/usr/local/lib/python2.7/plat-darwin',
    '/usr/local/lib/python2.7/plat-mac',
    '/usr/local/lib/python2.7/plat-mac/lib-scriptpackages',
    '/usr/local/lib/python2.7/lib-tk',
    '/usr/local/lib/python2.7/lib-old',
    '/usr/local/lib/python2.7/lib-dynload',
    '/usr/local/lib/python2.7/site-packages',
>>>

```

As you can see, setting `PYTHONPATH` causes new search directories to be added to the path. This is somewhat better than hard-coding such changes to `sys.path` in your code (as no source code modifications are needed). However, it's still a bit hard to predict where your new paths will be added to `sys.path`. For example, in the above example, why did the new directories get inserted into the middle of `sys.path` instead of the beginning or end? Mysteries abound.

Understanding Python's Installation

To better understand `sys.path`, it helps to step back and discuss a typical Python installation. Python typically installs itself in a directory such as `/usr/local`. Although this may vary, the top-level directory used by the interpreter can be found in the `sys.prefix` variable:

```

>>> import sys
>>> sys.prefix
'/usr/local'
>>>

```

Under this top-level prefix, the Python interpreter itself is usually found in an executable `sys.prefix + '/bin/python2.7'` such as `/usr/local/bin/python2.7`. If you have multiple versions of Python installed, they are accessible using their respective version numbers such as `python2.5`, `python2.6`, or `python3.2`. The unversioned Python interpreter is often a link to one of these installed versions.

Python-related scripts are also installed in the same directory as the interpreter binary. For example, if you are using `easy_install` or `pip` to install third-party packages, those installer programs are found in `/usr/local/bin` alongside the interpreter. If multiple versions of Python have been installed, you actually have to be somewhat careful using such scripts. A common mistake is using a package installer to install things into the wrong version of the interpreter.

Python's standard library is located in a directory `sys.prefix + '/lib/python2.7'` such as `/usr/local/lib/python2.7`. To see the raw structure of the standard library, you can run Python with the `-S` option. This runs the interpreter, but in a mode that ignores third-party add-ons. You'll just get the bare interpreter and standard library. Try it:

```

bash % python -S
Python 2.7.3 (default, May 24 2012, 22:03:52)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
>>> import sys

```

```

>>> sys.path
['',
 '/usr/local/lib/python27.zip',
 '/usr/local/lib/python2.7/',
 '/usr/local/lib/python2.7/plat-darwin',
 '/usr/local/lib/python2.7/plat-mac',
 '/usr/local/lib/python2.7/plat-mac/lib-scriptpackages',
 '/usr/local/lib/python2.7/lib-tk',
 '/usr/local/lib/python2.7/lib-old',
 '/usr/local/lib/python2.7/lib-dynload']
>>>

```

Within the library, modules and packages consisting of pure Python code are located in `sys.prefix + '/lib/python2.7'`. You should go take a look at it yourself. For example:

```

bash % ls /usr/local/lib/python2.7
... look at the output ...

```

In the output, you'll see a large number of hopefully recognizable files from the standard library.

C extension modules are located in `sys.prefix + '/lib/python/lib-dynload'`. Take a look at this as well:

```

bash % ls /usr/local/lib/python2.7/lib-dynload
... look at the output ...

```

Here, you'll see some of the lower-level system libraries related to sockets, timing, files, and so forth. All of the files are typically C shared libraries in the form of `.so` files.

Other parts of the library are a bit more special-purpose. For example, the `plat-mac` has some files containing platform-specific constants, and `tib-tk` contains some files related to the optional Tkinter (Tcl/Tk) package.

A Brief Word About Path Construction

The core part of `sys.path` is hardwired according to where Python is installed. When you run `python -S` as shown in the previous section, you get the basic path settings that Python starts with.

One subtle feature of the interpreter is that it can be “homed” to a different installation directory using an environment variable. For example:

```

bash % env PYTHONHOME=/my/python python -S
Python 2.7.3 (default, May 24 2012, 22:03:52)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
>>> import sys
>>> sys.path
['',
 '/my/python/lib/python27.zip',
 '/my/python/lib/python2.7/',
 '/my/python/lib/python2.7/plat-darwin',
 '/my/python/lib/python2.7/plat-mac',
 '/my/python/lib/python2.7/plat-mac/lib-scriptpackages',

```



```
    '/my/python/lib/python2.7/lib-tk',
    '/my/python/lib/python2.7/lib-old',
    '/my/python/lib/python2.7/lib-dynload']
>>>
```

The interpreter can also be re-homed through the use of “landmark” files. For example, when the interpreter starts, it determines its location on the filesystem and checks for the existence of `./lib/python2.7/os.py` and `./lib/python2.7/lib-dynload`. If those two files exist, Python assumes that you’ve relocated the installation and adjusts the path settings accordingly.

Of course, unless you’ve taken steps to set up the new directories as a proper installation, nothing much will work if you do this.

The `PYTHONPATH` environment variable specifies directories to be included just before the standard set of library directories. For example:

```
bash % env PYTHONPATH=/some/new/path:/another/path python -S
Python 2.7.3 (default, May 24 2012, 22:03:52)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
>>> import sys
>>> sys.path
['',
 '/some/new/path',
 '/another/path',
 '/usr/local/lib/python27.zip',
 '/usr/local/lib/python2.7/',
 '/usr/local/lib/python2.7/plat-darwin',
 '/usr/local/lib/python2.7/plat-mac',
 '/usr/local/lib/python2.7/plat-mac/lib-scriptpackages',
 '/usr/local/lib/python2.7/lib-tk',
 '/usr/local/lib/python2.7/lib-old',
 '/usr/local/lib/python2.7/lib-dynload']
>>>
```

Third-Party Packages and Extensions

As Python programmers know, there are a huge number of third-party packages that can be added to Python. Within the Python standard library, there is a special directory dedicated to these extensions called site-packages (note: sometimes it’s called dist-packages on certain Linux distributions). Typically, it is located at `sys.prefix + '/lib/python2.7/site-packages'`. Here is a partial listing of the site-packages directory of my machine:

```
bash % ls /usr/local/lib/python2.7/site-packages
IPython
certifi
chameleon
chardet
dateutil
easy-install.pth
mako
markupsafe
matplotlib
```

```

mpl_toolkits
numpy
oauthlib
pandas-0.7.3-py2.7-macosx-10.4-x86_64.egg
paste
pip-1.1-py2.7.egg
...

```

Handling of site packages is a little more complicated than it might initially seem. When Python starts, it imports a special module called `site.py`. One of the first things that happens is that `site.py` adds the `site-packages` directory to `sys.path`. You can often see it appended at the end of the normal library directories.

```

>>> import sys
>>> sys.path
['',
...
'/usr/local/lib/python2.7/lib-dynload',
'/Users/beazley/.local/lib/python2.7/site-packages # Added by site.py
'/usr/local/lib/python2.7/site-packages', # Added by site.py ]
>>>

```

You may also see a directory for user-level site packages being added as well. Since Python is often installed system-wide, it may not be possible for individual users to install Python packages, due to permissions. The user-level site package directory is reserved for this (each user can put custom packages in their per-user `site-packages` directory instead).

However, there's more to the management of packages than simply adding a few directories to the path. `site.py` also scans the `site-packages` directory for `.pth` files with additional path configuration. For example, here are some of the `.pth` files on my machine:

```

bash % cd /usr/local/lib/python2.7/site-packages
bash % ls *.pth
PasteDeploy-1.5.0-py2.7-nspkg.pth
easy-install.pth
repoze.lru-0.5-py2.7-nspkg.pth
setuptools.pth
zope.deprecation-3.5.1-py2.7-nspkg.pth
zope.interface-3.8.0-py2.7-nspkg.pth

```

These `.pth` files contain additional directories that must be added to the `sys.path` variable. For example, let's look at `easy-install.pth`:

```

bash % cat easy-install.pth
import sys; sys.__plen = len(sys.path) ./setuptools-0.6c11-py2.7.egg
./pip-1.1-py2.7.egg
./python_dateutil-1.5-py2.7.egg
./pandas-0.7.3-py2.7-macosx-10.4-x86_64.egg
./tornado-2.1-py2.7.egg
import sys; new=sys.path[sys.__plen:]; del sys.path[sys.__plen:];
p=getattr(sys,'__egginsert',0); sys.path[p:p]=new; sys.__egginsert = p+len(new)

```

In this file, lines starting with `import sys` are executed as Python code and can be seen to be doing things with `sys.path`. The other lines of the file list additional entries that need to be added to the path. It should be noted that the above code is a bit complicated due to the fact that it moves added directories from the end of `sys.path` to the beginning. This is the purpose of the rather cryptic line at the end.

If you want, you can make your own `.pth` files and put them in the `site-packages` directory. For example, here's a simple experiment you can try. Make a file called `mypath.pth` that contains the following:

```
# mypath.pth
/etc
/tmp
```

Now, copy this file to `/usr/local/lib/python2.7/site-packages` or `~/local/lib/python2.7/site-packages` and run Python. Here is an annotated listing of `sys.path` that shows the output and how the different parts are set:

```
bash % env PYTHONPATH=/some/new/path:/another/path python
Python 2.7.1 (r271:86832, Feb 27 2011, 11:47:28)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.path
['',
# --- Added by site-packages/easy_install.pth files
'/usr/local/lib/python2.7/site-packages/setuptools-0.6c11-py2.7.egg',
 '/usr/local/lib/python2.7/site-packages/pip-1.1-py2.7.egg',
 '/usr/local/lib/python2.7/site-packages/python_dateutil-1.5-py2.7.egg',
 '/usr/local/lib/python2.7/site-packages/pandas-0.7.3-py2.7-macosx-
10.4-x86_64.egg',
 '/usr/local/lib/python2.7/site-packages/tornado-2.1-py2.7.egg',
# --- Added by PYTHONPATH environment
 '/some/new/path',
 '/another/path',
# --- Standard Python library (initialized at startup)
 '/usr/local/lib/python2.7.zip',
 '/usr/local/lib/python2.7',
 '/usr/local/lib/python2.7/plat-darwin',
 '/usr/local/lib/python2.7/plat-mac',
 '/usr/local/lib/python2.7/plat-mac/lib-scriptpackages',
 '/usr/local/lib/python2.7/lib-tk',
 '/usr/local/lib/python2.7/lib-old',
 '/usr/local/lib/python2.7/lib-dynload',
# --- Added by site.py
 '/usr/local/lib/python2.7/site-packages',
# --- Directories list in mypath.pth
 '/etc',
 '/tmp']
>>>
```

Don't forget to delete the `mypath.pth` file after you're done experimenting with it.

At this point, I would encourage you to dissect the `sys.path` setting on your own machine to see if you can figure out how it gets put together.

How to Use This Knowledge

Knowing how `sys.path` gets constructed is useful if you have to do any kind of maintenance on a Python setup. For instance, you'll know where third-party packages go (site-packages) and be able to diagnose common installation problems.

Custom Python distributions (e.g., Enthought Python Distribution, ActivePython) as well as vendor-supplied Python versions (e.g., those provided by Apple, Linux distributions, etc.) often involve customizations of the `site.py` library to adjust module import paths.

If you've ever wondered how various Python package management tools such as `setuptools`, `distribute`, or `pip` work, they also mostly involve manipulation of `sys.path` through the use of `.pth` files in site-packages. Additionally, they may make their own changes to `site.py` as well.

Other popular tools such as `virtualenv` also play games with the Python installation (see <http://pypi.python.org/pypi/virtualenv>). For example, with `virtualenv`, you can make isolated Python environments:

```
bash % virtualenv example
New python executable in example/bin/python
Installing setuptools.....done.
Installing pip.....done.
bash %
```

This creates a new Python installation under a directory "example." If you look at it, you'll see something that looks like a typical Python installation:

```
bash % cd example
bash % ls
bin  include lib
bash % ls bin
activate  activate_this.py pip
activate.csh  easy_install  pip-2.7
activate.fish  easy_install-2.7  python
bash % ls lib
python2.7
bash %
```

If you run the Python interpreter in this environment and inspect the path, you'll see how the path has been configured for an entirely new location, yet parts of the path incorporate directories from the original Python installation. Again, it's a bunch of clever path hacking.

```
bash % example/bin/python
Python 2.7.3 (default, May 24 2012, 22:03:52)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin Type "help", "copyright",
"credits" or "license" for more information.
>>> import sys
>>> sys.path
['',
```

```

'/Users/beazley/example/lib/python2.7/site-packages/setuptools-0.6c11-
py2.7.egg',
'/Users/beazley/example/lib/python2.7/site-packages/pip-1.1-py2.7.egg',
'/Users/beazley/example/lib/python2.7.zip',
'/Users/beazley/example/lib/python2.7',
'/Users/beazley/example/lib/python2.7/plat-darwin',
'/Users/beazley/example/lib/python2.7/plat-mac',
'/Users/beazley/example/lib/python2.7/plat-mac/lib-scriptpackages',
'/Users/beazley/example/lib/python2.7/lib-tk',
'/Users/beazley/example/lib/python2.7/lib-old',
'/Users/beazley/example/lib/python2.7/lib-dynload',
'/usr/local/lib/python2.7',
'/usr/local/lib/python2.7/plat-darwin',
'/usr/local/lib/python2.7/lib-tk',
'/usr/local/lib/python2.7/plat-mac',
'/usr/local/lib/python2.7/plat-mac/lib-scriptpackages',
'/Users/beazley/example/lib/python2.7/site-packages']
>>>
```

Final Words

Figuring out how Python installations are put together is often one of the most mysterious parts of Python for newcomers and even for experienced programmers. It's important to stress that if you've ever had problems importing libraries, it's invariably some kind of configuration issue related to `sys.path`.

Believe it or not, there are even more tricks that can be played with in handling the `import` statement. However, I'll leave that for a future article.

iVoyeur

The Gift of Fire

DAVE JOSEPHSEN



Dave Josephsen is the author of *Building a Monitoring Infrastructure with Nagios* (Prentice

Hall PTR, 2007) and is senior systems engineer at DBG, Inc., where he maintains a gaggle of geographically dispersed server farms. He won LISA '04's Best Paper award for his co-authored work on spam mitigation, and he donates his spare time to the SourceMage GNU Linux Project.

dave-usenix@skeptech.org

It was Prometheus, so the story goes, who gave us fire. The other titans being otherwise engaged, it fell to him and his equally guilty but rarely talked about brother Epimetheus to populate the earth with flora and fauna, or at least to furnish them with the good stuff. Being neither solid architects nor effective planners, they may not have been the best choice for the job, for by the time they got around to the humans, they'd completely run out of horns, and antlers, and fangs, and shells, and wings and whatnot.

One imagines them standing wide-eyed over their empty basket of measures and countermeasures, like a spent bucket of Legos with the tower half-built, each accusing the other of some fatal error in accounting. "Gah," says Prometheus, being the bigger man [1] of the two. "I'll take care of it," and off he stomps into the heavens, eventually returning with fire and, bestowing it upon us (perhaps along with some fingernails that happened to be lying unwanted in the bottom of the basket), he calls it a day.

It was, for him, a fatal gift. A gift for which, as you may have heard, he was chained to a rock for eternity to have his entrails repeatedly devoured by a vulture—being divine entrails, they always grew back. That punishment always puzzled me. I mean being chained to a rock for eternity already seems like overkill, but the entrails thing, that smacks of a virulent and concentrated malice. Some ancient analog of whatever it is that makes a person reload the gun so that they can continue shooting bullets into the already dead body of their spouse, or boss, or spouse's lawyer, or boss's lawyer, or really any lawyer. And for what? Giving us fire? It seems like we would have probably figured that out on our own eventually. Why so severe a penance?

I suspect it was not the fire itself, which we ourselves could re-gift to a rhinoceros tomorrow with no substantial effect (save perhaps what would normally be expected from a freaked-out rhinoceros), but, rather, the propensity to use that gift that got him in trouble. The gift of fire is not the same thing as the gift of being able to use fire. Quite a bit more is implied by the latter. Perhaps the word "fire" is meant metaphorically, and what we were truly given that day was an inner-fire; that indescribable self-awareness that makes us human. The flame in us that not only enables us to use fire, but allows us to think about fire, and what we think about when we think about fire—now *there's* a gift over which I can imagine someone wanting to sic a vulture on a dude.

Awakened Rhinoceros

Over the course of the last few months I've been having what can only be described as conversations with my Nagios servers. Admittedly, these conversations are entirely composed of bits of information that I could get from the Nagios Web Interface, but the relevant point is that questions are not only being asked but answered. There is a dialog. Where before there was a black box, a wholly self-contained system comprised of a daemon that spoke to its own UI and nothing else, there is now an introspective entity to which I can posit queries, and from which I can expect replies. Before there was a rhinoceros and now there is, well, it's still a rhinoceros, but now it can tell me how it's feeling, which I think you'll agree is a cool trick for a rhinoceros.

Have I finally gotten with the program and installed NDOUtils [2] on my Nagios servers, pushing the state into a MySQL database so that I can query the DB? No. I've avoided writing an article on NDOUtils for the simple reason that I don't use it. It complicates things, doesn't scale well, and encourages a mindset I don't think works very well in distributed monitoring architectures, specifically, that all UI problems are nails and PHP/MySQL is the hammer. But that's another article.

I'm talking about MK Livestatus, the second of the three gems I plan to cover written by Mathias Kettner, a man who should be wary of rocks and vultures, because Livestatus does for Nagios something very similar to what Prometheus did for you and me, imbuing Nagios with, if not the flame of self-awareness, then at least the spark of introspection. If you've ever tried to script against a Nagios system, you understand the problem all too well. There just aren't a lot of ways to get data out of it, the three most popular being to scrape the Web interface, to parse status.dat, or to use NDOUtils. Over the years I've covered a few add-ons to assist in this endeavor, including my own beloved filesystem event-broker module, which exports the Nagios state to a file system.

Livestatus has all of these beat. Instead of replicating the state data to a log, or a database, or even a file system, it simply provides you with a UNIX socket (not a network socket) through which you can query state information directly from the Nagios daemon itself. You ask questions via a simple query language, and receive immediate answers directly from the same data structures resident in memory that are in use by the Nagios daemon. There is no overhead associated with replicating state or querying external databases. There is no disk I/O whatsoever. There is no latency and no risk of the data being out of date, and, importantly, there is no risk of blocking the daemon process with a hung ODBC thread or something similar.

MK Livestatus is implemented as a Nagios Event Broker (NEB) module, which is a compiled, shared-object file that is loaded into Nagios at startup. NEB modules communicate with the Nagios daemon by subscribing to announcements made by the Event Broker, and generally have access to everything that the daemon itself has to offer. I covered the Event Broker and the creation of NEB modules in depth in [3], and there are now several good articles in the tubeosphere, including official documentation at [4].

Livestatus is included in the Check_MK tarball, but can also be downloaded as a stand-alone package. If you're installing Check_MK, which we covered in last month's article, simply answer "yes" when the setup script asks if you'd like Livestatus installed. The stand-alone build is the autoconf standard configure, make,

and make install. Either way, Livestatus.o will be compiled and placed in /usr/lib/check_mk by default. Edit the nagios.cfg to enable the event broker with:

```
event_broker_options=-1
```

... and tell Nagios to load Livestatus with:

```
broker_module=/usr/local/lib/mk-livestatus/livestatus.o /var/lib/nagios/rw/live
```

Restart Nagios, and you're all set. The second argument is the location of the Livestatus socket on the local file system. Again, this is a UNIX IPC socket, so you'll need a tool like socat [5] or unixcat [6] to work with it in shell. Livestatus actually comes with its own copy of unixcat, which it will install for you in /usr/local/bin by default. I'm unsure whether this is the same tool as the one in ucspi-unix or if it's original code. Caveat emptor. The suggested method for making remote queries to Livestatus is by piping them over SSH, but there's no reason why the UNIX socket couldn't be front-ended by a TCP server program such as xinetd or DJB's tcpserver. Sample configuration for xinetd is available at [7].

Several other options may be passed to the NEB module on the broker_module line to, for example, place it in debug mode, or tune the various cache, buffer, and thread sizes. These are fully documented at [7]. Once Nagios is restarted, check to make sure the socket exists in the expected location, and test that it's working with:

```
echo 'GET columns' | /usr/local/bin/unixcat /var/lib/nagios/rw/live
```

If everything has gone according to plan, a dizzying block of text should scroll by. The "GET" statement in the above command was an "LQL" or Livestatus Query Language command. LQL commands are made up of two parts, the first line, which always begins with "GET", identifies the object type you're interested in inspecting. The Livestatus documentation refers to these as "tables," but they roughly correspond to object types in Nagios. The available tables, shamelessly pasted from the official documentation at [7], are:

- *hosts - your Nagios hosts*
- *services - your Nagios services, joined with all data from hosts*
- *hostgroups - you Nagios hostgroups*
- *servicegroups - you Nagios servicegroups*
- *contactgroups - you Nagios contact groups*
- *servicesbygroup - all services grouped by service groups*
- *servicesbyhostgroup - all services grouped by host groups*
- *hostsbygroup - all hosts group by host groups*
- *contacts - your Nagios contacts*
- *commands - your defined Nagios commands*
- *timeperiods - time period definitions (currently only name and alias)*
- *downtimes - all scheduled host and service downtimes, joined with data from hosts and services.*
- *comments - all host and service comments*
- *log - a transparent access to the nagios logfiles (include archived ones)*
- *status - general performance and status information. This table contains exactly one dataset.*
- *columns - a complete list of all tables and columns available via Livestatus, including descriptions!*

As you can see, these are mostly Nagios object types. You may have noticed "logs" in the list. Your eyes do not deceive you; Livestatus can retrieve log messages for

you, even if they've been archived. It does this by subscribing to log message events from the event broker and then caching them in memory. The number of cached messages in this memory buffer is one of the options you may pass in on the `broker_module` line. The default value is 500,000.

The second part of an LQL command is made up of any number of subsequent lines, collectively referred to as "headers." Headers generally allow you to modify your query, restricting the output to a given set of columns, filtering it for objects that meet a criterion, transforming the output into a statistic (e.g., returning how many objects matched, or computing the average of a returned value from several hosts), and even modifying the output format (to JSON vs. CSV, for example).

I used `GET columns` as the initial test command because a command like `GET hosts` will return a massive amount of data from a moderately sized Nagios server. Now, with the help of the "Columns" header, we can return just the host attributes we're interested in, like so:

```
Q='GET hosts
Columns: address state
' echo "${Q}" | unixcat /var/lib/nagios/rw/live
,
```

The "Columns" header causes Livestatus to return only the named attributes. The query above should return a list of the address of every host monitored by Nagios, followed by its current state. The returned list will use semicolons for line separators. Because LQL queries are themselves multi-line, things get a little weird from the command line. In the above example, I'm setting a variable to the multi-line query and then echoing it to `unixcat` (the double quotes around the variable in the `echo` statement are important. Without them the shell will eat the line-feeds in the variable). The official documentation recommends that you save the query to a file, and then redirect it to `unixcat`, like so:

```
echo 'GET hosts
Columns: address state' > query.lql

unixcat /var/lib/nagios/rw/live < query.lql
```

I quickly tired of both of these methods and took a minute to write a small shell wrapper that I call "qls" (query Livestatus) to make it easy to interact with Livestatus. It works kind of like an interactive Livestatus shell and is available for download from the [USENIX ;login: site](#) at [8].

If I wanted to narrow my list to just the hosts that were in a critical state, I could modify my query accordingly:

```
GET hosts
Columns: address state
Filter: state = 2
```

I can continue to narrow and modify my results with as many headers as I'd like, placing each new header on a subsequent line. The "Filter" header is quite powerful, supporting 12 equality operators, including negation, and case sensitive and insensitive POSIX extended regular expressions ("~" for case sensitive, and "~~" for case insensitive). I could, for example, refine my search to all the hosts whose description began with "DB" followed by a number between 1 and 4 (inclusive) with:

```
GET hosts
Columns: address description state
Filter: state = 2
Filter: description ~ ^DB[1-4]
```

Filter headers can be combined by suffixing them with a logical AND or OR header. This makes it possible to, for example, return the DB hosts that are in either a critical or warning state, like so:

```
GET hosts
Columns: address description state
Filter: state = 1
Filter: state = 2
OR : 2
Filter: description ~ ^DB[1-4]
```

The ordinal after the OR header defines how many of the previous headers are included in the logical OR.

Sometimes you don't want a list of the actual things, but, rather, the number of things that meet your criteria. The "Stats" header can not only provide counts but also compute sums, averages, and standard deviation, as well as return Min and Max, and much more. The following query returns the number of hosts in a critical state:

```
GET hosts
Stats: state=2
```

Stats headers also support logical operators. Here is the number of hosts that are in a critical state whose state has not been acknowledged:

```
GET hosts
Stats: state=2
Stats: acknowledged=0
StatsAnd: 2
```

The last example I have room to give is of the "Stats" headers grouping support. By adding a "Columns" header to my Stats query, I tell Livestatus to output a line each time the given Column value is different. For example, the following query will break down the number of unacknowledged critical hosts per hostgroup:

```
GET hosts
Stats: state=2
Stats: acknowledged=0
StatsAnd: 2
Columns: groups
```

I'm barely scratching the surface here, so if any of this was the slightest bit interesting to you, I'd highly recommend checking out [7]. Mathias Kettner really has knocked it out of the park with Livestatus. It's probably the coolest Nagios add-on I've encountered. It's so great it should just be in nagios-core. Next time we'll take a look at Multisite, a new Web GUI for Nagios which uses Livestatus as its data-source.

Take it easy.

References

- [1] “The Bigger Man,” American slang for the first person to back down from a conflict. <http://www.urbandictionary.com/define.php?term=Bigger%20man>.
- [2] NDOUtils: <http://exchange.nagios.org/directory/Addons/Database-Backends/NDOUtils/details>.
- [3] Dave Josephsen, “You Should Write an NEB Module,” *login*, vol. 33, no. 5, Oct. 2008: <https://www.usenix.org/publications/login/october-2008-volume-33-number-5/ivoyeur-you-should-write-neb-module>.
- [4] NEB API: <http://nagios.sourceforge.net/download/contrib/documentation/misc/NEB%20x%20Module%20API.pdf>.
- [5] socat, the multipurpose relay: <http://www.dest-unreach.org/socat/>.
- [6] ucspi-unix: <http://untroubled.org/ucspi-unix/>.
- [7] Livestatus docs: http://mathias-kettner.de/checkmk_livestatus.html.
- [8] www.usenix.org/publications/login/august-2012/ivoyeur.

/dev/random

Apps for Saps

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

Ahoy, Social Medians: new smartphone apps for 2012 are now available from the prestigious (and yet wholly fictional) Ferrell Institute for Technology Sarcasm (FITS). We here at FITS are always on the lookout for new ways to mock you and your ridiculous addiction to being connected twenty-four hours a day to people you don't really even know or like that much.

N.B.: I don't use apps on my phone (it has moments of feigned cleverness, but I wouldn't really call it "smart"), and I don't track the app development community. If one or more of these actually exists, my apologies to—and concern for the mental health of—the developer(s), along with my deepest sympathy for our species. All we can do at this point, it seems to me, is to keep it comfortable, give it plenty of fluids, and hope it gets better.

Splat: When this app is activated and the user points the smartphone's camera at an offending pest of the insect/arachnid variety, a reticle appears on the screen. The user positions the phone above the pest until the reticle turns red and then smashes that sucker with the phone. Splat probably voids your warranty, but is that too high a price to pay for one less vermin-carrying, blood-sucking mosquito in the world? I think not, so long as it's your phone, not mine.

Power Play: The phone emits a high, piercing noise when the battery level reaches 10%. When the noise stops, either the battery is dead or some irritated passers-by have clubbed you unconscious with their iPads.

Text Wrex: Combined with selected elements from Google's Self-Driving Car, this app detects when you are sending text messages from the driver's seat and immediately steers the vehicle into the nearest stationary object. A real time-saver that gets you off the road quickly.

Dork-B-Gone: Using voice actuation, this app detects when you are making a verbal ass of yourself at parties and gives you increasingly potent electric shocks until you change the subject or shut up entirely. May require optional external battery pack or small portable generator for some users.

Flatline: Monitors your vital signs and, in case of cessation (see "Dork-B-Gone," for example), sends text messages to people you've indicated are in your will to let them know they can come pick up your stuff now.

What's-Her-Name: Stores photos of people you've met and then when you casually bring the phone up as though you're reading a text, uses facial recognition

software to match the person's face to that database and tell you his or her name in large letters. Don't forget to turn off the speaker, though.

Andy C-app: Phones your spouse with a pre-recorded excuse (thought up and created while you were sober and nominally rational) as to why you're going to be late getting home tonight. Excuse can be selected from a menu or, if you're too far into your cups to summon that level of manual dexterity, just have one of your more sober friends (designated dialer) select "shuffle."

The Rockin' Butt Dialer: Allows the user to call up to four speed-dial numbers by wriggling their posterior in a certain manner. Often referred to as the "Ants in the Pants" app. Not advisable for use on board aircraft.

Domestic Tranquility: Allows the user to designate numbers, texts, and photos as private, so that when the Significant Other picks up the phone and scrolls through it without entering the secret "full access" code first, none of those entries are visible. Also comes with a non-functional "decoy" secret code that gives the impression of unlocking the hidden files, but actually does nothing, in order to throw off more tech-savvy SOs. An optional add-on module is available that routes an incoming call from one of the private numbers directly to voice mail while playing a pre-recorded donation solicitation message (voice or text) on the receiving end if the secret code is not entered before answering.

App-athy. Doesn't do anything at all but register itself as an app. For people who are tired of being labeled Luddites because they don't have any apps on their phone.

Occupy Yourself: This is an app that refuses to let you use the phone until a set list of demands is met. Usually they entail taking the day off, buying yourself something nice, and cheating on your diet. At random times it may insist you break a window or set a trash barrel on fire, but what's a protest without a little property damage?

Angry Nerds: Customizable geek avatars fling large sacks of cash at flimsy structures containing the fast-food workers, janitorial personnel, and used car salesmen who taunted them incessantly in high school.

Virtual Vengeance: Lets you take a photo of a vehicle whose driver has cut you off or committed some other unforgiveable offense in traffic and then obliterate it repeatedly in a variety of creative and violent ways. Sociopathic, but cathartic.

Super Walkie-Talkie: A third-party company assigns you a unique series of integers. You share this sequence with your friends. When they enter this secret number on their keypad, it establishes a walkie-talkie connection to you. Best of all, you don't even have to press a "talk" button every time: the app does that for you. It works over surprisingly long distances (up to 12,450 miles under certain conditions).

Next up from FITS R&D: a new FB plugin called RedFace. It selects a rumor, accusation, or sordid confession from our extensive database and posts it from your account to a random selection of your friends. Then you get to spend the next few days/weeks denying it and doing damage control. A great way to get to know your online acquaintances much better.

If you have suggestions for apps you'd like to see developed by our team of part-time programmers here at FITS, write them on the back of a stained napkin, crumple it up, and drop it in the nearest dumpster. That's where we get all of our ideas.

Book Reviews

ELIZABETH ZWICKY, WITH PETER GUTMANN, SAM STOVER, AND MARK LAMOURINE

Machine Learning for Hackers

Drew Conway and John Myles White
O'Reilly, 2012. 293 pp.
ISBN 978-1-449-30371-6

Suppose you have a pile of data, and you have a sense that what the fancy guys do with this stuff involves machine learning, but you have neither the time nor the inclination to go out and learn a whole bunch of new theory. This is the book for you, although you should note that for many, many people there is simply no practical approach to machine learning without picking up a language or two along the way, so you should be prepared to learn more than straightforward techniques. This book uses R for its examples in order to get something with a lot of power and less overhead than installing an entire Hadoop cluster. This is a tradeoff; if you already have Big Data around, somebody probably has already installed a Hadoop cluster or so for you, so that using R may actually be more work, both in installation and in trimming data sets to fit on a single machine. Still, it gives a consistent base to work from.

If you don't already have machine-learning people doing specialized stuff, you may be surprised at the amount of progress that can be made with algorithms that are not particularly smart. A few tricks will go a long way. If you do have such people, a little learning will help you talk to them. I found this an approachable introduction to the base techniques, although not without quirks. Some sections are made much more difficult to follow by having lovely big graphs. That sounds like a plus, but when a page introduces 5 graphs, which then show up on the 5 following pages with supporting text across the next 10, the text and the picture get punishingly out of sync, with text about illustration 6 showing up on the same page as illustration 2. Admittedly, this problem is definitely worse in the electronic editions I prefer to review, where flipping back and forth 2 or 3 pages is not easy.

The Scrum Field Guide: Practical Advice for Your First Year

Mitch Lacey
Addison-Wesley, 2012. 365 pp.
ISBN 978-0-321-55415-4

As the author points out, Scrum (which is one of the extended and incestuous family of Agile programming techniques) is simple but not easy. All you need to do is change everything about how your programmers work and how that work is specified and directed; what could possibly go wrong? For most organizations, converting to Scrum is like moving to another hemisphere. This book is aimed at providing useful advice for this transition, and the author speaks with the practical voice of experience, providing effective ways of making the transition.

As a "first year" book, it's not just about techniques; it's also an introduction to the common problems, inevitable miseries, and eventual joys of the transition. This kind of roadmap is a great thing to have when you are wondering whether you screwed things up, your team is hopeless, or this is just one of those bad patches that happen in every transition. It is inspirational; it did a good job of making me feel that this is a plausible transition. Furthermore, it covers a relatively wide range of situations, including outsourcing, operational/support groups, and at least one way of handling common functions like security and user interface design that span multiple teams. Daily team meetings are one thing if you're on a single team. What if you're on three or more? It is more applicable to isolated teams than to entire organizations going to Scrum, which brings up new issues. Getting buy-in for a single team is a different deal from getting buy-in when it's an edict from above.

User Stories Applied for Agile Software Development

Mike Cohn

Addison-Wesley, 2004. 262 pp.

ISBN 0-321-20568-5

User Stories makes an interesting pair to *The Scrum Field Guide*. It has a reasonably similar loose, practical tone, and a noticeably different take on some issues. Which issues? It's right up there with league vs. union in rugby, which is to say, I could tell you, but if you're not an aficionado, you're just going to be baffled. *User Stories* believes in half-point stories! What kind of madness is this? Everybody knows you only assign points from the Fibonacci series. Everybody who does Scrum, that is. The rest of you are wondering if I am actually making this up.

User stories are one of the key parts of Agile; they replace requirements documents. It is entirely possible that Agile is in fact entirely a plot to keep people from ever having to write 400-page documents with seven levels of nesting. I would consider restructuring the entire programming culture as a reasonable response to this experience. However, getting user stories to actually be more useful than requirements documents and getting people who are used to requirements documents to accept them are both tricky.

This is one of those books with a summary and quiz questions at the end of every chapter, which always annoys me, but it's a quick, straightforward guide to how you would get user stories, including advice on what generally goes wrong and what to do when, for instance, you don't get to speak to actual users to get user stories. I love the advice on what to do with stories when you're finished; tear them up if they're written on paper, but keep them if they're electronic. They're not very useful—but destroying bits isn't very satisfying, either. This kind of advice is also evidence of solid experience.

The Art of Community, Second Edition

Jono Bacon

O'Reilly, 2012. 526 pp.

ISBN 978-1-449-31206-0

This book's central audience is people who are trying to build free software communities, although it has wider applicability, particularly to communities at the tricky interface of the commercial and the volunteer. It does a good job of covering this tricky territory, with advice that blends the practical and the theoretical. (I am amused to note that it has the best burn-down graphs of the lot, better than either of the books that are actually about programming models.) In this second

edition, the author brings in some interviews to help flesh out the picture of other kinds of communities, but the bias is still pretty strong. If you want to build a group of system administrators, or a coalition of commercial companies (to name the communities I'm most familiar with), you will be doing some translating and some skipping, but you will still find useful information.

I have two mild regrets about the book. First, it does not at any point talk about how you deal with community members who are simply incapable of participating in the community as part of the consensus reality. It strongly implies that all community members are willing and able, if approached correctly, to come into harmony; in my experience, almost every community has members who are not willing and able to do so, for one reason or another. Managing these people is its own art. The author does encourage readers to try to bring people, even difficult people, into the community, which I think is a good choice. However, I think there are situations where removing people is also a valid if difficult choice, and it is useful to community managers to have some idea how to negotiate these incredibly difficult waters.

Second, the author uses technical terms of the online community overly broadly. This may be intentional, as a way to speak to people who come from other backgrounds, but it clashes on my sensitive ears and in one case obscures an important distinction. "Trolling" is not any type of disruptive online behavior; it's online behavior motivated by a desire to create controversy and dissent. It's important to know this and to discuss it, for a couple of reasons. People who don't understand that trolls exist can easily be sucked into dealing with fictional people, whereas people who do understand it can leap to deciding that real, if deluded, people are trolls. Either way, you get bad effects on the community. There is no point trying to help a troll, and there is no point ignoring real people. Lumping all of this together does a disservice to people who're trying to deal with online communities. The other case that bothered me was "bikeshedding" being described as all forms of discussion rather than getting something done; I think this is wrong, but probably not importantly so.

Domain-Driven Design

Eric Evans

Addison-Wesley, 2004. 519 pp.

ISBN 0-321-12521-5

This is a book about domain modeling for people who model domains for a living. It is not a book for people who might want to bring a bit of domain modeling into their existing process, although I suppose if you're already doing architecture for large development teams that are strongly separated

from the domain they're building for, it might be a valuable tool for you. But for the most part, it's about adding depth and finesse to your domain modeling.

Because of this, I'm not a great reviewer for it. I'm nearer to being a domain expert than a domain modeler, and I found it hard to sympathize with. Plus, the author is fond of patterns, and carefully puts the name of every pattern in small caps. This leads to entire sentences in which all the noun phrases are in small caps, which rapidly became annoying and added to the sensation that I was reading a careful exegesis of somebody else's religion. I'm pretty sure that this is dry going for all but a small audience, for whom it is probably stuffy but enlightening.

Electronic Signatures in Law

Stephen Mason

Cambridge University Press, Third Edition, 2007. 728 pp.

ISBN 978-1-84592-425-6

As any IT person knows, in order to create an electronic signature you need (depending on your particular fashion tastes) public and private keys, digital signatures, certificates, smart cards, and all manner of other paraphernalia. Lawyers—the people who actually work with signatures and signature law on a daily basis—don't see it quite that way, and this book explains why. For example, the function of a signature isn't necessarily to form a contract but may be merely to provide a cautionary function, encouraging the person affixing the signature to a document to take extra care in reading it, or a channeling function in which the signature records the time at which a document was accepted by the signer. In extreme cases it serves purposes quite unrelated to what we'd normally associate with signatures, such as allowing documents to be taxed in the form of a stamp tax or stamp duty.

The book starts with precedents going back to the Magna Carta, at which time "signatures" consisted of drawing a cross as a sign of Christian truth because writing your name on a piece of paper would have meant nothing, so that only non-Christian could sign their names on a contract. The book traces the history of what in legal terminology is called a manuscript signature, traditionally a pen-and-paper process, through to more modern forms such as telegrams and telexes. In none of these cases was special legislation necessary, since courts interpreted existing signature law and practice to cover newer technology that was introduced after the laws were written. This part of the book is a fascinating look at just how flexible the interpretation of what constitutes a signature actually is, with courts finding that "signatures" can include things like signing as "Mum" or saying "yes" (verbally), and more recently typing a name in an

email message, sending an SMS, or clicking "Send" on email that has your name in the "From:" field. In fact, provided that the identities of the parties to the agreement are fairly obvious, even a document containing no conventional signature at all may be enough to meet the legal requirements for a contract, if the intent of the participants is manifest and the method of conveying this is appropriate to the particular transaction. The extreme flexibility of existing contract law is demonstrated by the fact that a court case over the validity of email that's been "signed" by having the sender's name on it was supported by a quote from the Statute of Frauds Act of 1677, predating the existence of email by several centuries. The later parts of the book cover electronic and digital signature laws in great detail, including the twisty maze of signature laws, all different, that were passed in various countries around the time of the dot-com boom. This portion is probably of interest only to lawyers (or someone having to deal with the mess of subtly incompatible laws), and it appears to be an absolute minefield compared to the relative simplicity of the earlier case-law-based portions. In any case, much of what's contained in the laws seems to present little more than unnecessary complications. As the book points out when discussing the calisthenics required by electronic signature laws (p. 101), "contracts conducted by post . . . were commonplace two hundred years before the Internet, and it is to be wondered why businesses need such guidance when they have been dealing with such issues for such a long period of time."

One thing to be aware of is that this book will take a bit of getting used to for someone who's not familiar with the format of legal documents. The narrative progresses in two parallel channels: the main body text at the top of the page and extensive references, footnotes, comments, and annotations at the bottom, and because of the emphasis on case law there are plenty of those. The first half of the book, which covers existing case law and precedents for allowing various forms of non-manuscript signature, will be a real eye-opener for anyone who has grown up expecting to have to use certificates and smart cards and assorted other paraphernalia in order to form an electronic legally binding agreement. The second half, covering the ins and outs of electronic and digital signature legislation, will probably be of interest mostly to lawyers.

One downside to the book is that it's quite pricey, around \$200 US. On the other hand, if you can find it in a library or get your employer to buy it for you, it's definitely worth a read if you're in a position where you have to deal with electronic/digital signatures.

—Peter Gutmann

Metasploit

David Kennedy, Jim O’Gorman, Devon Kearns, and Mati Aharoni

No Starch Press, 2011. 299 pp.

ISBN 978-1-59327-288-3

This book was designed as a “useful tutorial for the beginner and a reference for practitioners” for the Metasploit penetration testing framework. I think the authors definitely achieved this goal: the book is well written, contains tons of useful information, and is extremely thorough. If you haven’t already picked up a book on Metasploit, this is the one to buy, and even if you already have a decent book on Metasploit, you probably should give this one a hard look. The first couple of chapters deal with the basics of penetration testing and where Metasploit fits into the process. Metasploit certainly handles the exploitation aspect of a pen-test, but there are lots of other things, such as gathering intel on your targets, that require time and other tools. While the book definitely focuses on Metasploit, a fair bit of time and effort was put into making you a better overall pen-tester, not just a Metasploit expert.

Chapter 5, “The Joy of Exploitation,” is where you really start digging into the awesomeness of Metasploit. By this point, you should be comfortable with the various Metasploit Framework (MSF) interfaces, as well as several other standard tools/utilities such as nmap, whois, Nessus, and NeXpose. The ability to take Nessus/NeXpose/nmap output and import it into Metasploit makes it a lot easier to keep track of your targets and their likely vulnerabilities. This book doesn’t just walk you through using Metasploit, it walks you through how to use other tools that make the Metasploit experience even more rewarding. Good stuff. Chapter 6 focuses on using the Meterpreter, which is a mini-shell delivered as a payload to an exploit. Once you’ve owned a box, you can use Meterpreter to interact with the system instead of being limited to the victim command prompt. In other words, just owning the box is only half the battle—once you’re there, you want to perform local attacks on the system (e.g., grab password hashes), and Meterpreter makes your job that much easier.

Chapters 7–9 go a little deeper into the more advanced features of the MSF. Detection avoidance using MSFencode to create custom binaries, client-side attacks, and auxiliary modules, such as SSH brute-forcing and protocol (FTP, HTTP, SMTP, SSH) fuzzers, are all tools that advanced pen-testers can rely on. You may not need them the way you need the exploits, but when you do, Metasploit has them.

Another vector that can’t be ignored is social engineering. One of the authors of this book developed the Social-Engineer Toolkit (SET), which is a separate tool that relies on the

Metasploit Framework. Each example comes with a little story, and I think this is a great way to teach/demonstrate the capability, because we can all imagine day-to-day scenarios like these. Lots of great stuff in this chapter, from creating your own malicious PDF to cloning a target’s Web site and harvesting credentials. Fantastic.

Chapter 11 introduces Fast-Track, which is another stand-alone tool that uses the MSF. Unlike the SET, this is an advanced pen-testing tool which offers exploitation methods that complement the MSF. This tool provides some MS-SQL attacks, different exploits from those bundled with Metasploit, and some browser attack vectors. While Fast-Track is quite advanced, its interface is pretty intuitive, and this chapter walks you through it.

Chapter 12 is dedicated to Karmetasploit, which is “Metasploit’s implementation of the KARMA attack.” This involves setting up a fake access point and enticing users to connect to it instead of to the “real” access point. Successfully executing this attack allows access to all network traffic (e.g., passwords) as well as the ability to launch client-side attacks on unsuspecting users. Karmetasploit is fairly simple to set up and execute, and this chapter provides all the necessary bits.

Chapters 13–16 are for the hard-core Metasploit user. If you want to write your own module, create your own exploits, port your exploits into the Framework, or write your own Meterpreter API scripts, these are the chapters for you. Each chapter deals with one of the topics and presents them in a very accessible manner. If you want to write or import your own exploits, you’ll need a firm grasp of assembler, so this is not for the faint of heart. If you’re up for a challenge, though, look no further.

The final chapter walks you through a simulated pen-test from soup to nuts. Complete and succinct, the chapter encapsulates everything you’ve learned in this book. Two appendices give some help on configuring your target systems and a “cheat sheet” of MSF commands. No stone is left unturned in this book, I’ll tell ya.

This is a book about how to become a (better) pen-tester, written by people who know what they are talking about and focusing on one of the best tools available, open source or otherwise. You simply cannot go wrong with this book if you want to learn more about Metasploit, or even if you want to learn more about penetration testing. Hands down, my new favorite Metasploit book.

—Sam Stover

Python for Software Design: How to Think Like a Computer Scientist

Allen B. Downey

Cambridge University Press, 2009. 270 pp.

ISBN 978-0-521-72596-5

Think Python: How to Think Like a Computer Scientist

Allen Downey

Green Tea Press, 2012. 212 pp.

<http://www.greenteapress.com/thinkpython/thinkpython.html>

It has always seemed to me that there were three types of technical books: tutorials, user guides, and references. Allan Downey's "Think" series has reminded me that there is a fourth type: teaching texts.

A typical tutorial tries to guide the reader down a specific path, controlling and limiting the reader's experience. It assumes that the user has little or no prior experience with the subject. A user guide is similar but might be more comprehensive and more useful once the user has some experience. A reference provides the most detail, but generally no learning narrative. All of these are focused on the single topic at hand and try to minimize diversions.

A good teaching text can be special in several ways (Downey's books being special in all of them):

- ◆ Promotes classroom discussion
- ◆ Provides a learn-by-experience framework
- ◆ Encourages exploration

At first glance, "Think Python" is just another beginner's guide to a popular scripting language. The subtitle, while less catchy, describes what's really going on. The first two paragraphs of the first chapter make it explicit: "The goal of this book is to teach you to think like a computer scientist" and "The single most important skill for a computer scientist is *problem solving*."

Most chapters introduce a single programming language concept. Sprinkled in among these are case study chapters that expose and discuss concepts such as interface and data structure design. Each chapter concludes with a glossary, a set of exercises, and a section on debugging specifically addressing the new material.

While the book begins with such basic concepts as variables, expressions, and statements, by the end the student will be working with techniques of classic object-oriented programming. Along the way the reader is introduced to the concepts and practice of lists (including map and reduce) and hashes (dictionaries in Python).

The real point of the book is to teach the underlying concepts of software development. The examples in the book are all written in Python, but Python is only the framework on which the real lessons are hung. The book was originally written and taught using Java.

There is very little exposition of Python syntax. The examples are clear and brief, and the text explains the intent and meaning, but almost no time is spent on particular language elements. A true beginner without the benefit of a classroom and teacher will probably want to have a traditional language manual as well while working through this book. A professional developer who is not familiar with Python will want access to the Net or at least a pocket reference.

Think Stats: Probability and Statistics for Programmers

Allen B. Downey

O'Reilly Media, 2011. 120 pp.

ISBN 978-1-449-30711-0

Again, the subtitle of *Think Stats* gets to the heart of why Downey's second book is different from most ordinary textbooks on probability and statistics. While most programmers do not need to be statisticians, the wide variety of fields that are using mathematical methods makes it unlikely that most will be able to escape without some exposure.

Think Stats introduces statistics as a means to answer a certain class of questions involving populations and data sets. In the first chapter, Downey asks if there is a way to confirm the common belief that first children are born later in a pregnancy than subsequent ones. He addresses that question in new ways as the book progresses, adding nuance to the answers each time.

Throughout this book Downey uses references to online resources for additional reading and for data sets to use for exploring. He introduces topics with appropriate mathematical notation, but, rather than taking up space with formal derivations or proofs, he refers to online articles, usually from Wikipedia, for students to explore on their own (or as directed by the instructor). Downey focuses, instead, on illustrating each of the concepts in the context of some problem or question in need of a solution.

Downey covers the traditional topics for an introductory stats course: cumulative distribution functions, population distribution curves and characteristics, and probability. And, unusually for an introductory text, he also discusses Bayesian statistics. The last half of the book covers hypothesis testing, estimation, and correlation, again standard topics, but again including Bayesian Estimation.

The book illustrates examples and provides exercises in Python. It uses pyplot for graphing and provides other statistical demonstration code as downloads from the book's Web site. It follows *Think Python's* pattern of ending each chapter with a glossary and exercises.

This book is aimed at a guided introductory college-level classroom, but the frequent external references make it suitable for a motivated self-learner with working knowledge of Python and enough calculus to be able to interpret the few formal mathematical expressions: simple summation and integration. The presentation is also ideal for a knowledgeable programmer to explore and understand the basis of statistical methods.

Think Complexity: Exploring Complexity Science with Python

Allen B. Downey

O'Reilly Media, 2012. 142 pp.

ISBN 978-1-449-31463-7

Think Complexity is Downey's most recent book and the least conventional. As before, the subtitle gives a better feel for the content, but this time it still needs some clarification. The book really explores the techniques and applications of computational modeling. "Complexity Science" is the name given to the study of the set of problems where traditional analysis is ineffective and to which computational modeling is a better approach.

Downey devotes the first chapter to explaining what he means by "Complexity Science," although in the end you've only really learned what it is not. He defines it largely in contrast to traditional deterministic reductionist science and engineering. A deep understanding of what that means has to wait until the reader has gotten some experience with modeling complex systems. Downey also promises to address questions of the philosophy of science that are raised by the development and use of computational modeling.

Chapters 2 through 5 are used to build up the base of tools and concepts that will be needed for application to real problems. These include graphs and some analysis of algorithms and statistics (there is some overlap from *Think Stats* here).

Chapters 6 through 10 are the meat of the book. Downey proceeds to introduce and explore the characteristics and uses of cellular automata, self-organizing critical systems, and, finally, agent-based models.

Each of the sections contains some code, but more is provided as downloadable modules, which are then used as a base for the student's work. The samples are implemented in Python and use the NumPy and SciPy modules, but almost no text is devoted to them. It is assumed either that the teacher will provide what is needed or that the reader has the ability to find what they need online.

These first two sections of the book are liberally laced with references to classic papers and studies which have informed the development of computational modeling as a discipline. These often include links to Wikipedia articles and, occasionally, to the original source papers. Downey invites the reader to question and understand the limits of classical scientific methods and the utility of computational methods for understanding systems that resist traditional analysis.

The last few chapters of the book contain a set of case studies presented by his students and evaluated by members of the faculty. Each uses some form of non-deterministic modeling to explore the characteristics of some dynamic system. Downey invites instructors and readers to submit additional case studies for inclusion in the evolving text for the course.

In this latest book, Downey makes explicit something that has been integral to the first two as well, but without recognition. The preface includes a short section of guidance for teachers who intend to use the book as the basis for a course, and then a section dedicated to self-learners. He addresses the fact that many popular books on these kinds of topics are aimed at a casual audience and tend to avoid details and depth that would scare off the lay reader (or a publisher's concept of a lay reader). These books are addressed to the dedicated learner who is interested in the details and is motivated to follow leads to the source materials.

As teaching texts these books are special. At about \$30 US they are each much less expensive than comparable textbooks. They are smaller, with many external references; rather than including redundant and static expositions of concepts, they take advantage of existing online resources.

All three of these books have one other characteristic that sets them apart: they are all freely available under the Creative Commons Attribution-Noncommercial-ShareAlike 3.0 license. All three are available as HTML and PDF online from Downey's own Web site at <http://www.greenteapress.com>.

—Mark Lamourine

NOTES

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 October 1997 to this month:
www.usenix.org/publications/login/

Discounts on registration fees for all USENIX conferences.

Special discounts on a variety of products, books, software, and periodicals:
www.usenix.org/member-services/discounts.

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

Margo Seltzer, *Harvard University*
margo@usenix.org

VICE PRESIDENT

John Arrasjid, *VMware*
johna@usenix.org

SECRETARY

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

TREASURER

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

DIRECTORS

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

Sasha Fedorova, *Simon Fraser University*
sasha@usenix.org

Niels Provos, *Google*
niels@usenix.org

Dan Wallach, *Rice University*
dwallach@usenix.org

CO-EXECUTIVE DIRECTORS

Anne Dickison
anne@usenix.org

Casey Henderson
casey@usenix.org

Farewell, Jane-Ellen

Casey Henderson, USENIX Co-Executive Director

The August 2012 issue of *login*: serves as a bittersweet milestone. This month Jane-Ellen Long begins her well-deserved retirement from USENIX, to which she has dedicated herself for the past 14 years, in multiple key roles including those of *login*: Managing Editor and Director of Information Systems and Production. During her tenure, USENIX gained a machine room, a full-time system administrator, an online conference registration system, and electronic membership renewals, among many other major advances for this small non-profit organization. She led us as we moved from paste-up conference proceedings to electronic-only proceedings that are open to everyone, the keystone of USENIX's Open Access Policy.

Shortly after I came to USENIX in 2002 as Member Services Assistant, she chose to involve me in the redevelopment of our membership Web forms, demonstrating her characteristic knack for collaboration and awareness of multiple perspectives; unfortunately, very few managers have such a good sense of the big picture in taking into account all users of a system, from the front end to the back end. She asked me to join the production team in 2003 and over the course of the near-decade since has taught me more than could fit into all of the issues of *login*: she's skillfully edited. I'm not alone in having been fortunate in benefiting from her

wisdom and experience. I know that the hundreds of authors, speakers, program chairs, and other volunteers who have had the pleasure of her leadership and assistance in building our conferences and programs join me in giving her our heartfelt thanks for her contributions to USENIX's successes.

Please join us in celebrating Jane-Ellen's many years of devoted service at LISA '12 in San Diego. We'll be toasting her at the Conference Reception on Thursday evening.

Welcome, Rikki

Casey Henderson, USENIX Co-Executive Director

As this chapter concludes for USENIX and Jane-Ellen, we're excited to announce the beginning of a new era for *:login:*. Rikki Endsley will become *:login:* Managing Editor beginning with the October 2012 issue. Many of you will recognize Rikki as USENIX's Community Manager; she's also a familiar face in the high-tech publishing industry. In addition to her freelance writing work, which continues to be a primary passion, in the past she worked as the managing editor for *Sys Admin* magazine and *UnixReview.com*, and she was the Associate Publisher of *Linux Pro Magazine*, *ADMIN Magazine*, and *Ubuntu User* until 2011. We welcome Rikki's greater role on the USENIX team.

USENIX, Open Access, and You

Anne Dickison, USENIX Co-Executive Director

The open access issue is gaining strength. Researchers are tired of seeing their groundbreaking work locked up behind a pay wall. USENIX not only agrees that research should be free and open to all, but we're also doing something about it. Since 2008, we have offered free online access to conference proceedings. In addition, all videos of conference technical session presentations are now available to the general public on the USENIX Web site shortly after they take place. Being a USENIX member supports our community's right to open access, and we thank you. However, if you feel as strongly as we do about the importance of the issue, please help us in spreading the word about our policy and asking your colleagues to consider supporting open access by joining USENIX. Together we can build the momentum to change the course of access to research.

USENIX Association Financial Statements for 2011

Anne Dickison and Casey Henderson, USENIX Co-Executive Directors

The following information is provided as the annual report of the USENIX Association's finances. The accompanying statements (p. 92) have been reviewed by Michelle Suski, CPA, in accordance with Statements on Standards for Accounting and Review Services issued by the American Institute of Certified Public Accountants. The 2011 financial statements were also audited by McSweeney & Associates, CPAs. Accompanying the statements are several charts (p. 93) that illustrate where your membership dues go. The Association's complete financial statements for the fiscal year ended December 31, 2011, are available on request.

**USENIX ASSOCIATION
STATEMENTS OF FINANCIAL POSITION
As of December 31, 2011 & 2010**

**USENIX ASSOCIATION
STATEMENTS OF ACTIVITIES
For the Years Ended December 31, 2011 & 2010**

ASSETS	2011	2010
Current Assets		
Cash & cash equivalents	\$ 787,118	\$ 189,444
Receivables	56,975	150,798
Prepaid expenses	53,320	52,588
	<hr/>	<hr/>
Total current assets	897,413	392,830
Investments at fair market value	5,177,383	5,887,267
Property and Equipment		
Office furniture and equipment	351,131	320,397
Website - construction in progress	345,394	-
Leasehold improvements	29,631	29,631
Less: accumulated depreciation	<u>(330,721)</u>	<u>(308,778)</u>
Net property and equipment	395,435	41,250
Other assets	337,960	326,676
	<hr/>	<hr/>
	\$ 6,808,191	\$ 6,648,023
	<hr/>	<hr/>
LIABILITIES AND NET ASSETS		
Current Liabilities		
Accounts payable	\$ 543,430	\$ 103,181
Accrued expenses	46,250	68,714
Deferred revenue	84,435	82,090
	<hr/>	<hr/>
Total current liabilities	674,115	253,985
Long-Term Liabilities	337,960	326,676
Total liabilities	1,012,075	580,661
Net Assets		
Unrestricted Net Assets	5,796,116	6,067,362
	<hr/>	<hr/>
Net Assets	5,796,116	6,067,362
	<hr/>	<hr/>
	\$ 6,808,191	\$ 6,648,023

	2011	2010
REVENUES		
Conference & workshop revenue	\$ 3,033,125	\$ 3,096,793
Membership dues	364,450	480,459
Product sales	902	860
SAGE dues & other revenue	96,863	150,185
General sponsorship	5,000	8,000
	<hr/>	<hr/>
Total revenues	3,500,340	3,736,297
OPERATING EXPENSES		
Conferences & workshops	2,604,390	2,827,141
Membership - login:	431,357	476,001
Projects & GoodWorks	161,180	185,969
SAGE	150,115	98,741
Management and general	466,559	453,946
Fund raising	57,503	44,738
	<hr/>	<hr/>
Total operating expenses	3,871,104	4,086,536
Net operating deficit	(370,764)	(350,239)
NON-OPERATING ACTIVITY		
Interest & dividend income	172,953	156,884
Gains (losses) on marketable securities	(5,701)	352,582
Investment fees	(63,285)	(61,909)
Other non-operating	<u>(4,449)</u>	<u>(2,616)</u>
Net investment income & non-operating expense	99,518	444,941
Change in net assets	(271,246)	94,702
Net assets, beginning of year	6,067,362	5,972,660
Net assets, end of year	\$ 5,796,116	\$ 6,067,362

Chart 1: USENIX 2011 Membership Dues Revenue

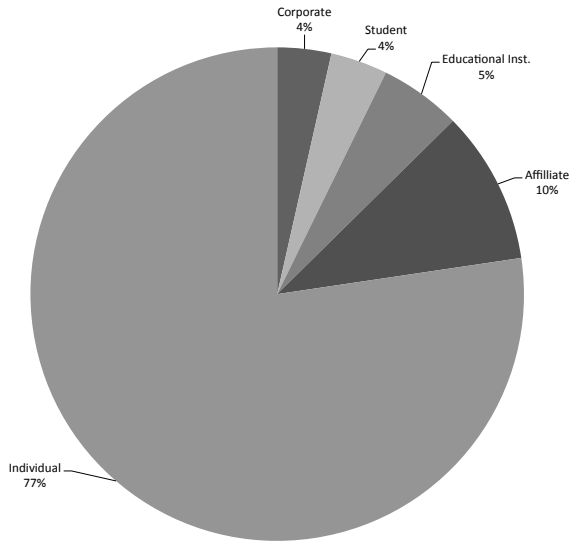


Chart 2: Where Your 2011 Dues Went

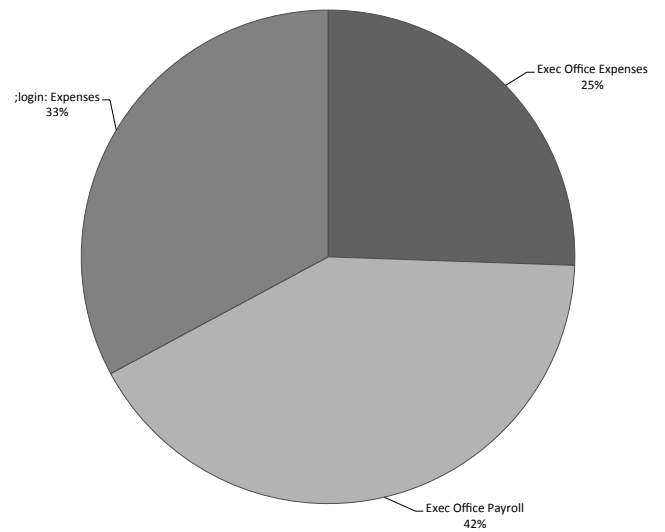
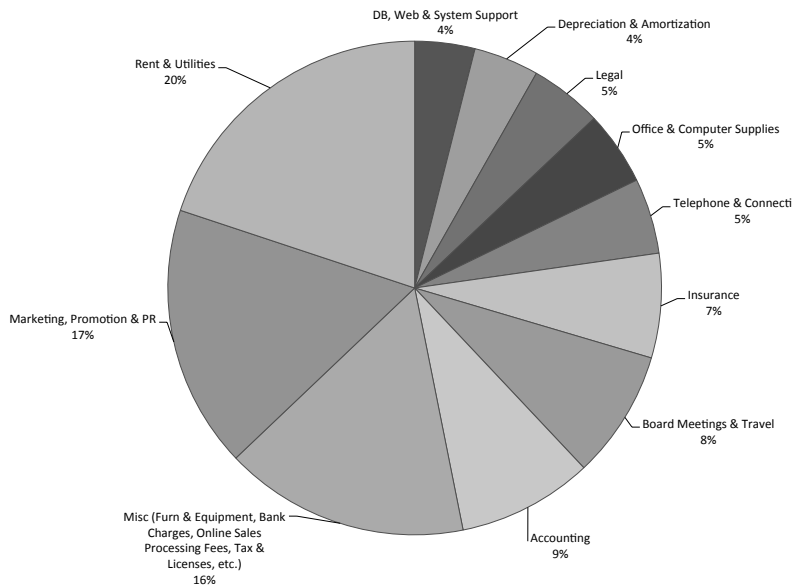


Chart 3: USENIX 2011 Administrative Expenses



Conference Reports

CONFERENCE

In this issue:

9th USENIX Symposium on Networked Systems
Design and Implementation 94

*Summarized by Advait Abhay Dixit, Rik Farrow, Andrew Ferguson,
Katrina LaCurtis, Marcelo Martins, Karthik Nagaraj, Kevin Ngo, and
Will Scott*

5th USENIX Workshop on Large-Scale Exploits and Emergent
Threats: Botnets, Spyware, Worms, New Emerging
Threats, and More 114

Summarized by Manuel Egele, Rik Farrow, and Engin Kirda

9th USENIX Symposium on Networked Systems Design and Implementation (NSDI '12)

San Jose, CA
April 25–27, 2012

Opening Remarks and Awards Presentations

Summarized by Rik Farrow (rik@usenix.org)

Steven Gribble introduced the workshop, saying that 30 papers out of the 169 submitted were accepted, and all submitted papers were shepherded as well. There were over 250 attendees on the opening day—not a record, but close to it. The Best Paper award went to Matei Zaharia and his co-authors for “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing” (see their “Fast and Interactive Analytics over Hadoop Data with Spark” article in this issue of *login*). Steven announced a new award, the Community Award, for papers that make code available to the community. This went to “How Hard Can It Be: Designing and Implementing a Deployable Multipath TCP,” by Costin Raiciu et al., with the Zaharia paper and “Serval: An End-Host Stack for Service-Centric Networking,” by Erik Nordström et al., receiving Honorable Mentions.

Big Data

Summarized by Marcelo Martins (martins@cs.brown.edu)

CORFU: A Shared Log Design for Flash Clusters

Mahesh Balakrishnan, Dahlia Malkhi, Vijayan Prabhakaran, and Ted Wobber, Microsoft Research Silicon Valley; Michael Wei, University of California, San Diego; John D. Davis, Microsoft Research Silicon Valley

Mahesh Balakrishnan started by saying that flash drives create optimization opportunities in data centers that cannot be achieved with hard disks. He claimed that flash drives can be leveraged in a distributed environment without partitioning, hence keeping strong consistency, while guaranteeing cluster performance. Mahesh described how CORFU achieves this objective starting from its software and hard-

ware components. CORFU exposes a simple API to server applications that view the flash cluster as a single shared log supporting read and append operations. In parallel, CORFU utilizes custom-designed, network-attached flash units to reduce power draw and cost. That flash drives allow random reads with minimum overhead is the key to CORFU's scalability. Mahesh proceeded to explain the client-centric protocol used to access the shared log and how it guarantees ACID (atomicity, consistency, isolation, durability) conditions. Each client keeps a projection that maps logical entries into flash units and needs to request tokens from a sequencer machine to reserve an append operation to a specific unit. A write-once semantics is used to easily identify the tail of the log. Mahesh mentioned that the sequencer does not represent a single point of failure, as it is used only as an optimization asset. CORFU leverages chain replication for integrity and durability.

He then proceeded to describe how CORFU handle two types of failures: flash-unit and client failures. In the case of a flash-unit malfunctioning, reads are served by the remaining original drive, while new writes are posed to both original and substitute drives. In the background, the substitute drive replicates old data from the original drive. Projections storing the new mapping are interchanged between clients using the Paxos protocol. Mahesh claimed that the entire recovery procedure takes between 10 and 20 milliseconds for a 32-drive cluster. In the case of a client failure, if a client reserves a disk from the sequencer but crashes before completing its write, other clients can fill in the referred disk with invalid data to guarantee appending ordering.

Finally, Mahesh presented CORFU's performance evaluation. For throughput, CORFU reads scale linearly as more flash drives (up to 32) are added to the cluster. Appends also scale linearly with the number of flash drives, although the sequencer becomes the bottleneck when more than 31 drives are presented to the cluster. In its current implementation, the sequencer can serve up to 190,000 appends per second without performance degradation, although Mahesh believes that this number can go up to one million appends per second if the sequencer logic is implemented in hardware. In his final slide, he commented on the relationship between CORFU and Paxos. While Paxos-based protocols suffer from I/O bottlenecks due to space partitioning, CORFU stitches multiple flash-unit chains into a single virtual storage unit. The partitioning occurs over time, and therefore there is almost no I/O bottleneck.

David Andersen (CMU) asked about the influence of newer approaches to NVRAM technology (e.g., memresistors, PRAM) on CORFU's design. Mahesh explained that CORFU takes into account that current flash units are block-

addressable, while the newer technology is byte-addressable. CORFU assumes a sequential-appending design given the write-block property of flash drives, but Mahesh believes that a byte-addressable flash storage could enable arbitrary appending and avoid data padding. John Dunagan (MSR) asked for clarification of the benefits of separating the flash units from the client. Mahesh explained that in CORFU, servers should communicate with all flash units in order to provide scalability, and therefore its design does not preclude data locality. Ben Reed (Yahoo! Research) questioned the validity of a global shared-log abstraction for different server applications. Mahesh agreed with Ben's position and clarified that CORFU does not make such an assumption and can provide a different log per application.

Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing

Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael Franklin, Scott Shenker, Ion Stoica (University of California, Berkeley)

► *Awarded Best Paper!*

► *Awarded Community Award Honorable Mention!*

Matei Zaharia gave a lively presentation about the internals of Spark. The recipient of the Best Paper Award started by commenting on the inefficiency of the MapReduce programming abstraction in attending the data-sharing needs of particular "big data" applications (e.g., iterative processing and interactive queries). He gave examples of how Hadoop (an open-source implementation of MapReduce) does not scale properly, due to its design based on replication and network and disk I/O between stages. Matei then enticed the audience by asking how one could get in-memory data sharing and fault tolerance without replication across nodes to avoid I/O bottlenecks, and introduced his solution: Resilient Distributed Datasets (RDDs). RDDs work as an immutable, distributed collection of records that can only be recovered in a coarse-grained manner: the system only logs the deterministic operations between stages and reapplies the logged operations to the missing record sets in case of failure. To determine which record sets should be rebuilt, RDDs track the graph of transformations that builds them (*lineage*). High write throughput is guaranteed, as RDDs work close to memory-bandwidth limits.

Matei then presented Spark, a programming interface that uses RDDs as its abstraction for operations on large datasets. Matei proceeded with examples of how the Spark API can be used to solve real-world analytical problems. He showed how an administrator can use the API to solve a large log-mining problem and how the API requests translate into interactions between the master and workers in a

distributed environment. Spark, implemented in Scala, uses lazy evaluation to avoid unnecessary data operations. As the workers generate the first result sets, they store these sets in local memory. Subsequent queries take advantage of the cached results, leading to almost instantaneous responses. As evidence, Matei claimed that Hadoop takes about 170 seconds for responding to an on-disk search query from a 1 TB Wikipedia dataset, while Spark takes only 5–7 seconds. His next example showed the latency savings for an iterative machine-learning algorithm. Although the system pays the price of from-disk data loading in the first Spark query, subsequent iterations take advantage of in-memory shared data and respond quickly. Even in the case of failures, the recovery overhead is far from the full disk access (119s vs. 81s).

Spark is built from scratch, can load data from the same sources as Hadoop, including HDFS and S3, and is available as open-source software (<http://www.spark-project.org>). Matei also mentioned that RDDs can be used to easily express different parallel models, such as graph processing, iterative MapReduce, and SQL. Matei proceeded with a demonstration of his system running on a 20-node EC2 cluster. He showed how a user would use the Scala interpreter to run Spark queries and again emphasized the benefits from in-memory data sharing. He showed a live example of how a large Wikipedia query that took about 19 seconds to load data from disk only took 1 second for subsequent queries. Finally, Matei pinpointed the extensions to Spark that companies and other research projects have been developing since its open-source announcement as proof of its applicability to diverse scenarios.

Matei was asked about the possibility of memory leaks due to many RDD structures kept in memory as a result of intensive use of Spark. He replied that his system implements garbage collection to avoid leaks. John Dunagan (MSR) followed with a comment on the benefits of persistence for data sharing so that nodes will not lose sharing context once they switch tasks and asked how that would affect in-memory RDDs. Matei agreed with John's comment and said that his system leverages data locality via delay scheduling to keep the benefits of low I/O overhead. Rik Farrow asked whether he had considered persisting some of the RDDs instead of using LRU for replacement in case of low memory. Matei argued that one could take advantage of a hybrid solution that would consider both cases. He then showed one of his backup slides demonstrating the performance overhead relative to the amount of working set kept in memory (the missing data would be recomputed or retrieved from disk). In his scenario, for each 25% less working set kept in memory, the iteration time would increase by 15 seconds on average.

Camdoop: Exploiting In-network Aggregation for Big Data Applications

Paolo Costa, Microsoft Research Cambridge and Imperial College London; Austin Donnelly, Antony Rowstron, and Greg O'Shea, Microsoft Research Cambridge

Paolo Costa started with a brief recap of MapReduce and then focused on a particular problem his group at MSR was trying to solve: improving the performance of the shuffle and reduce phases of MapReduce. He argued that the all-to-all traffic pattern of the shuffle phase is an encumbrance to data-center networks and that the final results of a MapReduce job are usually much smaller than the intermediate results before the shuffle phase. As an example, Facebook reported that its usual final results were 5.42% the size of intermediate data. A few solutions exist to reduce the input size of shuffle and reduce phases. Combiners are limited to in-node aggregations, while rack-level aggregation trees easily create network bottlenecks on the switch links.

Camdoop's goal is to avoid this bottleneck, performing the combiner function as part of the network protocol by harnessing packet aggregation at each hop. Paolo then introduced CamCube, MSR's solution to perform in-network processing. Instead of using routers/switches, CamCube builds a 3D-torus topology, where servers are directly connected to each other and can intercept and process packets. The 3D topology facilitates server naming and fault-tolerance mapping. Paolo then explained the logic behind the topology choice and network design. Each server has the capability of processing packets and aggregating them before forwarding, avoiding the link-contention problem of rack-level aggregation trees.

In the second part of his talk, Paolo talked about Camdoop's innards. As a first goal, Camdoop does not modify the MapReduce model. Each server runs map tasks locally. The system keeps a spanning tree where combiners aggregate results from mappers and send them to their parents. The root node runs the reduce task. To guarantee network locality, the parent-children relationship is mapped to physical neighbors (using the 3D-topology naming). One problem with this solution is the load distribution. To overcome this issue, Paolo described a striping mechanism that distributes data across disjoint trees formed from the same cube topology. Not only load is distributed, but all possible links can be used if six disjoint trees are built.

The third part of Paolo's talk encompassed evaluation results and more implementation details. Paolo gave a quick overview of the 27-server CamCube testbed built by MSR and also a 512-server (8x8x8) packet-level simulator. One of the insights we can take from Camdoop is that the shuffle and

reduce phases run in parallel, resulting in a performance boost from the traditional MapReduce sequential model. Considering that all data streams are ordered, once the root of the spanning tree receives a packet, it can start the reduce function and no intermediate results need to be stored. For evaluation purposes, Paolo compared Camdoop with an implementation of MapReduce that runs shuffle and reduce in parallel, and a restricted version of Camdoop without in-network aggregation. First, he showed via benchmark applications that Camdoop can compete with the state-of-the-art Hadoop and Dryad. Next, he considered the effects of data-aggregation size and reduce-phase types (all-to-one vs. all-to-all) on task running time. For the all-to-one case, as we move from no aggregation (output size is equal to intermediate size) to full aggregation (output size is minimal compared to intermediate size), Camdoop presents up to a tenfold latency improvement over no-aggregation Camdoop. When comparing no-aggregation Camdoop with MapReduce, he pointed out that the aggregation size brings no improvement, since there is no actual data aggregation. Still, Paolo noted that there is a threefold latency improvement of no-aggregation Camdoop over MapReduce, due to other optimizations. Considering the full-aggregation scenario, Paolo also talked about the impact of the number of reduce tasks on running time. For both no-aggregation and MapReduce, as the number of reduce tasks increase, the running time becomes smaller. However, Camdoop returns the same performance numbers independent of the number of reduce tasks. Furthermore, Camdoop presents a tenfold and fiftyfold latency improvement over the other two solutions, respectively. Finally, Paolo showed that even for interactive applications (e.g., Bing Search, Google Dremel) which produce small-sized intermediate data, Camdoop is capable of taking advantage of in-network aggregation to quickly run MapReduce stages.

David Oran (Cisco) asked whether Paolo had any measurements on the effects of latency on Camdoop. Paolo replied that latency exists due to server-based forwarding on the order of hundreds of milliseconds, but this loss could be compensated by traffic reduction resulting from in-network aggregation. Rishan Chen (Peking University) asked about the scalability of Camdoop in terms of machines. Paolo answered that no matter the number of servers participating in the system, each node still maintains the same number of neighbors. What changes is the number of hop counts between nodes, which may increase latency, but this can be compensated for by on-path data aggregation. Ankit Singla (UIUC) asked for clarification of the differences between the full-bisection bandwidth topology and the one used by Camdoop. His impression was that in the former, all shuffle-phase transfers occur in parallel, and in Camdoop, parents

have to wait for their children to finish before transferring data. Paolo partially disagreed with his observation, saying that transfers in the full-bisection topology do not occur as early as one might imagine and also involve waiting time.

Wireless

Summarized by Marcelo Martins (martins@cs.brown.edu)

WiFi-NC: WiFi Over Narrow Channels

Krishna Chintalapudi, Microsoft Research India; Bozidar Radunovic, Microsoft Research UK; Vlad Balan, USC; Michael Buettner, University of Washington; Srinivas Yerramalli, USC; Vishnu Navda and Ramachandran Ramjee, Microsoft Research India

Krishna Chintalapudi started with a high-level overview of the differences between conventional WiFi radio and WiFi-NC. Conventional radio leverages one wide channel at a time for data transmission/reception and permits only one pair of communicating devices. Conversely, WiFi-NC proposes the simultaneous usage of several narrow channels for the same purpose and allows multiple simultaneous transmissions and receptions from several devices. Krishna said that despite the increase in WiFi data rates in the last 10 years, there was no corresponding increase in throughput. While the time taken to actually transmit data has decreased with higher transmission rates, the overhead associated with setting up the transmission has not changed at all. He explained that WiFi-NC increases the communication efficiency by slicing a wide data channel into lower data-rate channels and uses them to transmit data in parallel, which also parallelizes the associated overhead (CSMA, preamble, SIFS, and ACK). Krishna claimed that although the progress on the WiFi standard resulted in more bits being encoded per Hz, the current trend from new revisions of the 802.11 protocol to make channels wider does not always work. Access points operating in different frequencies must resort to the lowest frequency so that clients can coexist and avoid starvation. WiFi-NC can solve the coexistence problem by dividing the wide frequency channel into narrow channels and allowing independent data transmission and reception. Finally, he said that WiFi-NC can use the whitespace fragmented spectrum by creating low-rate channels that exist between other medium sources.

The key component behind WiFi-NC is its compound radio. Krishna said that the new radio design uses digital signal processing to create narrow channels without modifying the analog radio front end. To permit simultaneous transmissions and receptions, the compound radio should avoid radio interference. For transmissions and receptions, elliptic filters are the best for interference attenuation. He briefly went over other complex design challenges that should be over-

come before enabling simultaneous communication without interference. Next, Krishna briefly touched on how the NC radio selects the right operating point in the white spectrum using an optimization algorithm named TMax, although no details were provided during the talk.

The evaluation of WiFi-NC basically demonstrated that its design and implementation can overcome the coexistence, multiple simultaneous communication, and starvation-avoidance problems. Krishna produced graphs showing that the greater the number of narrow channels, the more efficient the communication is for a single link. For instance, by increasing the slicing of a 20 Hz channel from two to eight smaller channels, the efficiency of a 300 Mbps link increased from 20% up to around 50%.

Aaron Gember (University of Wisconsin) asked whether one could assume that all nodes have the same white spectrum available to them and whether there would not be interference issues between nodes. Krishna said that there is no such assumption and that if nodes have overlapping visibility on parts of the spectrum, they will use narrower channels to fairly share this available spectrum. Suman Banerjee (University of Wisconsin) asked about the percentage of actual data transmission given the parallel transmissions using WiFi-NC compared to the amount of data related to radio control. Krishna first explained how his solution confronts preamble dilation on the physical layer resulting from the interference-avoidance algorithm. Later, he made a data transmission estimate of roughly 40%. Brad Karp (University College London) asked about the effect on the radio cost from using multiple components to enable greater communication efficiency, as in the case of WiFi-NC. Krishna replied that he is not a radio manufacturer and could not give an estimate. He mentioned that the front-end already comes with the filters, but we need to rely on Moore's Law to expect the prices to go lower. Finally, he added that the cost would be proportional to the number of transmitter/receivers.

Catching Whales and Minnows Using WiFiNet: Deconstructing Non-WiFi Interference Using WiFi Hardware

Shravan Rayanchu, Ashish Patro, and Suman Banerjee, University of Wisconsin Madison

Shravan Rayanchu asked how a user can determine the source of the WiFi disturbance that leads to his annoying video-buffering issue. It turns out there are many reasons for it, including weak signal and interference from WiFi and other devices (cordless phones, microwave ovens, Bluetooth receivers, etc.) that share the spectrum with the access point (AP). The idea behind WiFiNet is to construct a system

capable of identifying the sources of WiFi interference due to non-WiFi devices and quantifying the impact on the link quality. He said that identifying the culprits of non-WiFi-to-WiFi interference is harder, since there is no simple way for a WiFi card to detect interference from, say, a cordless phone, and also there is no easy way to identify the non-WiFi device (cf. MAC addresses on WiFi cards).

WiFiNet leverages the power of AirShark, software running on access points capable of detecting non-WiFi devices. AirShark relies on WiFi cards capable of taking spectral samples from the communication medium and, with the help of a decision-tree classification algorithm, categorizing non-WiFi devices using tags. The classification algorithm works based on the signature of each spectral sample. Shravan's group at Wisconsin extended AirShark to perform interference quantification and localization. A WiFiNet controller coordinates the distributed view of different non-WiFi devices from access points. The first challenge of WiFiNet is to differentiate and identify non-WiFi samples. To do this, multiple synchronized APs observe the medium and collect samples that will later be analyzed. The samples can tell whether they belong to the same device if they share the same period and frequency range. Still, we need to separate different instances from the same device type. Shravan did not go into detail, but he said he used a clustering algorithm to identify multiple instances of a given device.

The next step is to localize the non-WiFi device. Assuming that the AP locations are known, one can implement several algorithms using the received signals. WiFiNet's choice is based on a propagation model from the devices. However, since the transmit power of a device cannot be estimated by the controller, WiFiNet uses the difference in the received power between pairs of APs to estimate the non-WiFi device position. Shravan followed with an evaluation of two different deployments (of 4 and 8 APs) that shows that the propagation model produces the lowest median error (less than 4 meters) compared to other algorithms (Fingerprint, Centroid, and Iterative). The last piece of WiFiNet is an estimation module for the impact of interference of detected devices on a wireless link. Given that each AP point is capable of identifying transmission overlaps between WiFi frames and non-WiFi pulses, the WiFiNet controller can correlate frame losses with the transmission overlaps to generate a relative number corresponding to the loss probability.

Shravan then made a few remarks on different experiments for testing the effectiveness of his proposed framework. The first experiment concerned the interference impact from a single non-WiFi device of various types on the link quality. It turns out that in most cases the WiFiNet estimate is very

close to the ground truth, with errors of less than 10% for more than 90% of the cases. For the scenario with multiple devices, WiFiNet is still capable of producing estimation values close to the ground truth, with errors within 15% for more than 85% of the cases.

Michael Nowlan (Yale University) asked whether WiFiNet used a dictionary to identify device signatures and, if so, whether it would require updating. Shravan replied that AirShark contains a dictionary of devices that only includes the most popular ones, and it does require updates. Srinivas Narayana (Princeton University) had two questions: (1) the reason behind the need to differentiate instances of the same device type, (2) the sensitivity of the estimation algorithm to the number of available access points. Shravan answered that identifying an instance could help one localize the culprit and proceed to mitigate the interference problem. Regarding algorithms, he said that the localization algorithm is actually more sensitive to the number of available APs, while the estimation could still be performed with a single AP. Still, both the estimation and localization are fairly accurate even with a low number of APs. Brad Karp (University College London) wondered about the possibility of using more advanced radio hardware and its impact on WiFiNet design. Shravan answered that sophisticated hardware definitely can improve the interference estimation problem and there are works that make use of it; however, WiFiNet's contribution was designed to harness current off-the-shelf WiFi cards and see to what extent they can be used to solve the interference-identification problem.

Content and Service-Oriented Networking

Summarized by Katrina LaCurtis (katrina@csail.mit.edu)

RPT: Re-architecting Loss Protection for Content-Aware Networks

Dongsu Han, Carnegie Mellon University; Ashok Anand and Aditya Akella, University of Wisconsin—Madison; Srinivasan Seshan, Carnegie Mellon University

Dongsu Han addressed the problem of minimizing data loss in delay-sensitive communications. Minimizing loss is a particular challenge here because of time constraints; it often takes too long to retransmit data. Today, FEC (Forward Error Correction) is one of the most popular methods to protect against loss, but it's difficult to tune and is susceptible to bursty losses.

Content-aware networks, however, are equipped to do caching at the routers, which minimizes the bandwidth cost of redundancy (this is referred to as redundancy elimination, or RE). The RPT (Redundant Packet Transmission) system

exploits this capability, introducing redundancy in a way that the network understands.

To retain robustness, sources in RPT send one copy of the original packet along with multiple “redundant packets.” These redundant packets are very small and contain a reference to the original content in the cache. If the original packet is lost, the content can be recovered using the redundant packets and the cache.

Compared to FEC using Reed-Solomon, RPT has less overhead and a lower data loss rate. When tested on streaming video, the quality of the received video stream (measured via PSNR) was better with RPT than FEC. RPT has the added benefit that delay is decoupled from redundancy, so the user is able to more easily control both the quality of redundancy and the maximum delay incurred. One downside of RPT is that its flows get prioritized, which can lead to unfair bandwidth allocation when non-RPT traffic also exists in the network.

Zili Francisco asked for clarification on the redundant packets. Dongsu noted that these packets are not compressed packets—rather, they're deduplication packets that only contain a pointer to what the router should look up in its cache.

Fitz Nowlan (Yale) asked why the received quality of FEC (10,9) was so poor compared to other FEC schemes (Fig. 8 in the paper). This happens because there is a high chance that two or more packets are lost in the same batch, and if two or more packets are lost, you lose data.

David Nolan asked whether they had studied the amount of state that has to be received by each router when many RPT flows are coming into it, and whether they've looked at doing explicit expiration from the cache once a copy of the data has been successfully delivered. Dongsu noted that they're using RE as-is on content-aware networks, and so aren't actually modifying anything about the underlying network.

Serval: An End-Host Stack for Service-Centric Networking

Erik Nordström, David Shue, Prem Gopalan, Rob Kiefer, and Matvey Arye, Princeton University; Steven Ko, University of Buffalo; Jennifer Rexford and Michael J. Freedman, Princeton University

► *Awarded Community Award Honorable Mention!*

Erik Nordström began by noting that the Internet of the 1970s was designed for accessing hosts. Today, we access services running in data centers replicated across many machines. Users are agnostic to the location/host and are only interested in the service.

Accessing a service involves locating a nearby data center, connecting to the service (establishing a flow, load balancing, etc.), and maintaining connectivity to the service, even through migration. Today's overloaded abstractions don't support these operations particularly well. Serval provides a new naming abstraction that gives clean role separation in the network stack by providing a service-level control/data plane split.

In Serval, the network layer stays unmodified, and a service access layer (SAL) is inserted above it. Applications now connect to services via ServiceIDs, which are allocated to service providers. Data for the service is sent over flows which are addressed by FlowIDs, which are then attached to particular endpoints (IP addresses).

The SAL contains a flow table and service table. The service table maps ServiceIDs to actions such as forward, delay, etc. In the control plane, the Service Controller uses this table and communicates with other service controllers, with DNS, with OpenFlow, etc. Serval can handle load balancing and flow migration and is incrementally deployable.

Performance-wise, having a single Serval TCP connection that never breaks saves hundreds of MBs of data, and is just slightly slower than TCP/IP (in their tests, roughly 933 vs. 934 Mbps). The throughput of the service table is a bit worse, but hasn't been optimized yet. Today, it can support 140K connections per second. Serval extends the SDN model to the network edge, while OpenFlow primarily focuses on layer-2/layer-3 abstractions.

Emin Gün Sirer (Cornell) asked about cross-layer interactions (e.g., how Serval interacts with congestion control). Serval can tell certain layers to freeze state, so bad cross-layer interactions can be avoided.

Reliable Client Accounting for P2P-Infrastructure Hybrids

Paarijaat Aditya, Max Planck Institute for Software Systems (MPI-SWS); Mingchen Zhao, University of Pennsylvania; Yin Lin, Duke University and Akamai Technologies; Andreas Haeberlen, University of Pennsylvania; Peter Druschel, Max Planck Institute for Software Systems (MPI-SWS); Bruce Maggs, Duke University and Akamai Technologies; Bill Wishon, Akamai Technologies

Paarijaat Aditya discussed hybrid CDNs. In these CDNs, clients download from peers as well as CDN servers. This gives us the scalability of P2P networks as well as the reliability/manageability of centralized systems. However, clients are untrusted, and the infrastructure can't observe P2P communication, leading to clients that can mishandle content, affect service quality, or misreport P2P transfers. CDNs need

a mechanism to reliably account for client activity. This is where Reliable Client Accounting (RCA) comes in.

RCA looks at two types of attacks: misbehaving client software (e.g., deviations from correct protocol, collusion) and suspicious user behavior (e.g., repeatedly downloading content to drive up demand). In RCA, clients maintain a tamper-evident log of their network activity. These logs are based on those in PeerReview, but are able to take advantage of the CDN infrastructure. The logs are periodically uploaded to the infrastructure and verified, and anomalous clients are quarantined.

The overhead of RCA is primarily dominated by signature verification, which is on the order of the number of communicating client pairs. This overhead is lower than that of PeerReview, which is on the order of the number of messages. To verify client activity, plausibility checking verifies whether the log is consistent with a valid execution of software. To detect collusion and suspicious user behavior, RCA uses statistical methods. Prior work blacklists potentially misbehaving clients; RCA quarantines them, allowing the client to download but not upload. This enables the use of aggressive anomaly detectors.

The authors studied RCA in the context of Akamai NetSession, which is software used to distribute large files in CDNs. This software is bundled with a customer-specific installer. The authors attacked NetSession by having a client report fake downloads. The network overhead of RCA (as well as the CPU overhead) was less than .5% of the actual content downloaded; log storage accounted for around 100 KB/day. To handle log uploading and processing requires about 35 machines, compared to the 10 used now. RCA was able to catch all of the attacks the authors inserted into NetSession.

This talk generated much discussion. Vyas Sekar (Intel Labs) asked why Akamai was worried about these types of attacks if they control the software. Paarijaat clarified that since the clients are untrusted, they could modify client software and cause significant accounting errors. Srinivas Narayana (Princeton) asked about incentives for carrying out these attacks. A client might want to carry out click fraud against a particular customer, or undermine the CDN's reputation. Colin Dixon (IBM) pointed out that when RCA classifies a client as being malicious, the client is then served directly from the infrastructure. Clients might find this desirable (as the infrastructure usually provides better service), so what is the encouragement to not be malicious? Paarijaat pointed out that clients can opt out of P2P service if they so choose, and that one could be stricter and blacklist clients who were definitely malicious. Tudor Dumitraş (Symantec Research) asked whether RCA could help with an attack where a cli-

ent downloaded a file he wasn't supposed to have access to. These types of attacks should primarily be handled by the CDN via access control lists. Could RCA be used by clients to verify billing statements from the CDN? In theory, this is possible, as RCA is providing a mechanism to ensure honesty in reporting. Emin Gün Sirer asked about scalability, since RCA scales with $O(n^2)$ rather than $O(n \cdot S)$, where n is the number of clients and S is the size of the content (admittedly very large). Paarijaat said that they had designed the system with scalability in mind.

Network Robustness

Summarized by Andrew Ferguson (adf@cs.brown.edu)

Header Space Analysis: Static Checking for Networks

Peyman Kazemian, Stanford University; George Varghese, UCSD and Yahoo! Research; Nick McKeown, Stanford University

Peyman Kazemian presented Header Space Analysis, a mathematical foundation for answering questions about a network's configuration. In this approach, packets are interpreted as points in a multi-dimensional binary space. That is, a packet with header of length L is represented by a point in an L -dimensional space, where each dimension can have the value zero or one, or be a wildcard; this space contains all possible packets. Networking equipment, such as switches and routers, is modeled as transfer functions from the two pairs: (packet/point, input port) to (packet/point, output port).

Armed with algebraic operators, Header Space Analysis can be used to answer questions such as: can host A talk to host B? does the network contain traffic loops? or is a network slice X fully isolated from slice Y? For example, to check for loops in a network, an all-wildcard packet is injected at every switch port, and the transfer functions from attached networking equipment are applied. If the transformed packet eventually returns to the starting port, and within the same header space, then a loop has been detected.

Header Space Analysis can be performed using a Python library, Hassel, available at <https://bitbucket.org/peymank/hassel-public/>. Hassel includes a Cisco IOS configuration parser which generates the packet transfer functions, and implements checks for reachability, loop detection, and slice isolation. Kazemian used Hassel to detect loops in Stanford's backbone network in less than ten minutes of runtime on a single laptop.

After the presentation, Kazemian was asked about modeling middleboxes that operate on a sequence of packets. He replied that it is difficult to do so currently because they require state, but such equipment can be over-approximated with a

function modeling the largest header transformation it could make. Srinivas Narayana asked about the runtime complexity. Kazemian replied that some tests had complexity growing with the square of the number of VLANs in the network. Arjun Guha asked how the results are presented to the user. Hassel currently prints a textual representation of packet headers that match the query.

A NICE Way to Test OpenFlow Applications

Marco Canini, Daniele Venzano, Peter Perešini, and Dejan Kostić, EPFL; Jennifer Rexford, Princeton University

Software Defined Networks (SDNs), such as those made possible by OpenFlow, allow the network to be managed by configuration programs running on general-purpose computers. These programs are often logically centralized, yet manage an inherently distributed system. This mismatch is a source of potential bugs, which can have serious consequences in production networks.

Marco Canini presented NICE (No bugs In Controller Execution), a tool which can uncover bugs in OpenFlow controllers written in Python for the NOX platform. NICE takes as input the unmodified controller program, a network topology, and a set of correctness properties (such as no forwarding loops or packet black holes). It then models the controller running on the network and uses model checking to identify packet traces that cause the controller to violate one or more of the properties.

However, the space of all possible packet traces is extremely large. To make NICE scalable, symbolic execution is applied to the controller's event handlers to determine equivalent classes of packets, and domain-specific knowledge is used to minimize the number of packet orderings that must be checked. The authors ran NICE on three previously published OpenFlow controllers, which identified eleven bugs. NICE is publicly available at <https://code.google.com/p/nice-of/>.

Ranveer Chandra requested references to the tested controller applications, which can be found in the paper. Ratul Mahajan asked why the controller needed to be symbolically executed during each state of the model checking. The reason is that some event handlers' execution depends on the state of the network; making this state symbolic as well would create a scalability challenge. Sambit Das wondered about the specificity of this work to OpenFlow. Canini replied that they make some OpenFlow-based assumptions about how the controller can interact with the switches to make NICE scalable.

Toward Predictable Performance in Software Packet-Processing Platforms

Mihai Dobrescu and Katerina Argyraki, EPFL, Switzerland; Sylvia Ratnasamy, UC Berkeley

Packet-processing platforms such as RouteBricks (SOSP 2009) and PacketShader (SIGCOMM 2010), which run on general-purpose hardware, offer flexibility unavailable in specialized network equipment. However, when multiple packet-processing applications, such as IP forwarding and traffic monitoring, are run together on a software router, the performance can degrade in unpredictable ways. These performance drops are due to contention for shared resources such as CPU caches and memory controllers.

Mihai Dobrescu presented the results of a benchmark study in which five different packet-processing applications ran in combinations inside a single contention domain. The results revealed that contention for the L3 cache was the dominant factor in the performance drop, as most of the profiled applications had few memory accesses.

The authors developed a synthetic application which allowed them to vary the number of L3 cache references each second. When the synthetic application contended for resources with each packet-processing application, the performance drops were within 3% of the corresponding drops caused by realistic applications. Therefore, this simple synthetic application can be used to predict offline the performance impact on an existing packet-processing application when it is run alongside a second one.

Dobrescu was asked by several attendees whether the results depend on the workload used for the benchmark. He responded that the results did, and that network operators generally provision for the worst case of a specific, assumed workload. Rishi Kapoor asked about other potential bottlenecks, such as the NIC, and was referred to analysis performed in the RouteBricks paper.

Privacy

Summarized by Will Scott (wrs@cs.washington.edu)

Detecting and Defending Against Third-Party Tracking on the Web

Franziska Roesner, Tadayoshi Kohno, and David Wetherall, University of Washington

Franzi Roesner presented her work looking at how companies track Web site visitors. The basic threat is that online trackers today can create profiles that capture a large fraction of user activity. Franzi started by laying out a classification system that captured the evolution of tracking Web

tracking. Beginning with the use and outsourcing of analytics code, which only tracks activity on a single site, the taxonomy then looks at the different forms of cross-site tracking present on the Web today. In particular, it differentiates standard “vanilla” tracking from mechanisms that create profiles even when third-party cookies are disabled, those that don’t rely on the client-side state at all but are included by other trackers, and social widgets that can associate Web browsing with additional profile information even when third-party cookies are blocked.

Franzi then presented a set of measurements showing that 91% of the top 500 domains include tracking code, and simulations of browsing from search logs showing that individual trackers could capture up to two-thirds of visited pages. These measurements also showed that the average domain included between four and five trackers, with DoubleClick being the most popular cross-site tracker. Also notable was that Facebook and Google social widgets were both present on approximately 30% of the top 500 domains.

Franzi then talked about potential ways to defend against Web tracking. She showed that blocking third-party cookies can dramatically reduce anonymous tracking, but doesn’t impede social tracking. Additionally, Firefox’s method of blocking third-party cookies is undesirable, because it breaks the functionality of embedded social widgets. ShareMeNot (<http://sharemenot.cs.washington.edu>) is a new alternative for Firefox and Chrome. In Firefox, it removes cookies from social widget requests until they are clicked, and in Chrome it replaces these buttons with local versions until clicked. Franzi showed that this browser extension is largely effective at preventing tracking by social widgets, but cautioned that the method was based on a blacklist and couldn’t handle complex widgets such as Facebook comments.

The majority of questions after the talk focused on what additional profiling can’t be detected by the client. Vyas Sekar from Intel Labs asked how the discovered trackers were grouped into administrative domain, and both Aaron Gember from University of Wisconsin and John Dunagan from Amazon asked about back-end correlation and side channels. Franzi responded that there likely was sharing of data and ownership between the trackers, and agreed that data handling would largely need to be tackled by legislative policy. The point of this work was to give users control before a longer-term policy solution is found. Saikat Guha from MSRI asked why some sites would embed 43 trackers, and Franzi explained that those cases were largely due to ad networks, where a range of third parties could get pulled in.

Towards Statistical Queries Over Distributed Private User Data

Ruichuan Chen, Alexey Reznichenko, and Paul Francis, Max Planck Institute for Software Systems (MPI-SWS); Johannes Gehrke, Cornell University

Ruichuan presented Practical Distributed Differential Privacy (PDDP), a system allowing providers to analyze users, while allowing users to maintain exclusive control of their information. He began by explaining why previous work leaves something to be desired: previous attempts either can't scale, can't handle churn, or can't handle untrusted users. He then introduced PDDP, explained that the two key insights were requiring clients to only respond with binary answers and the addition of unknown noise at the proxy. By limiting the space of values and having the clients and proxy collaborate to create additional blind noise, answers remain confidential and malicious clients are not able to over-influence the results.

Ruichuan then went through an example of PDDP processing a query. In the PDDP system, analysts will send the honest but curious proxy SQL queries and bucketing specifications. The proxy will then select clients and forward the query. Clients will execute the query and, for each bucket, return a 1 or 0 encrypted with the analyst's key, using the Goldwasser-Micali system. The proxy next aggregates the results and then shuffles and adds blind noise to each bucket before returning the results to the analyst. The analyst is able to decrypt the binary values and tally the final, noisy result. The system is currently deployed as a Firefox plugin, allowing querying of online browsing activity. While performance is often a concern, PDDP's encryption is fast enough to respond efficiently. Ruichuan noted that they had observed tens of thousands of encryptions per second on desktop browsers, and more than 800 encryptions per second on smartphones. The system is also capable of widespread queries, with a query against one million clients taking a total of less than 30 CPU minutes and a total bandwidth and storage usage of 1.2 gigabytes.

The two points from the talk eliciting questions were the proxy design and the relationship between bucketing and noise. Jacob Lorch from Microsoft Research asked who was envisioned in control of the proxy, and Michael Freedman from Princeton asked if "honest but curious" was chosen because designing for malicious proxies would be too expensive. Ruichuan clarified that proxies could be controlled by a privacy advocate or a company with an external auditor and that the paper does cover design for a malicious proxy. Matvey Arye from Princeton asked if the differential noise could be reduced, due to the uncertainty introduced by bucketing, and another questioner asked who chose the number of

buckets and their boundaries. Ruichuan responded that the analyst chooses the bucketing, and because of that control the differential noise needed to be orthogonal from the bucketing.

Koi: A Location-Privacy Platform for Smartphone Apps

Saikat Guha, Mudit Jain, and Venkata N. Padmanabhan, Microsoft Research India

Saikat Guha presented the design for Koi, a system providing a useful and secure interface to location data. He began by explaining that the reason so many mobile applications transmit location data is because the current abstraction of latitude and longitude presented by the operating system isn't useful. Instead, he proposed that applications should express their intent to the operating system and receive event notifications when locations they are interested in are reached. When building the system, they thought of six ways applications interact with location: advertising, content tagging, search, recommendations, social networking, and navigation, and they believe the Koi model will work with all of them.

Saikat then explained how the Koi model provides location privacy. In particular, in order to provide privacy but still answer personalized queries, Koi unlinks information, so while the cloud gets individual queries and locations, it doesn't know how they are linked together. To do this, the cloud system is split into two components: a matcher and a combiner. The matcher does most of the processing, while the combiner keeps information unlinked and can be provided by the user or a trusted organization. In addition, a model checker, ProVerif, has been used to formally prove that the protocol provides privacy guarantees. The specific proofs show that users can't be linked to their attributes, user attributes can't be linked together, and users with matching attributes can't be linked.

In addition, Saikat showed the flow of how a query takes place in the Koi system. Before the query is made, entities and users maintain their status by sending the matcher their identifier, and attributes (like category or location) encrypted first with the matcher's key, and then with the combiner's. The matcher forwards encrypted data to the combiner and gets the unencrypted data back, so it knows all of the information, while the combiner knows the links but not the actual data. A user can then ask the matcher for matching entities, without the cloud being able to link the match to the user. The benefit of this approach is that the matcher works on plaintext attributes (a user might query for entities within five blocks of a GPS position) and the phone can automatically update the location for all applications.

Michael Nowlan from Yale asked if companies would be willing to give up the ability to track location. Saikat agreed that many companies do want the data, but that the OS abstraction would allow significantly more transparency for the user about what applications are doing. Bryan Ford (Yale) asked what would happen if there was a malicious cloud component, or if two cloud components colluded. Saikat answered that the privacy guarantees only work if there isn't collusion, but the proofs of correctness provide some guarantee against a single malicious party. Marcelo Martins from Brown asked how this system would work for users with limited data plans. Saikat said that Koi reduces energy costs, because the OS can amortize requests from multiple applications.

Security and Availability

Summarized by Will Scott (wrs@cs.washington.edu)

Aiding the Detection of Fake Accounts in Large Scale Social Online Services

Qiang Cao, Duke University; Michael Sirivianos, Telefonica Research; Xiaowei Yang, Duke University; Tiago Pogueiro, Tuenti, Telefonica Digital

Qiang Cao presented Sybil Rank, a system for detecting fake accounts in online social networks. Facebook believes 5–6% of accounts are fake, and these fake accounts are hard to detect automatically. Tuenti, the Spanish social network where this work was done, hand reviews 12,000 suspicious accounts every day. Only a small percentage of reviewed accounts are actually fake, so there is a need for a system that is both fast and effective at detecting these fake accounts in large social networks. Sybil Rank uses the landing probabilities of short random walks from known non-Sybil users to rank nodes. The intuition between these walks is that walks from real users are unlikely to enter Sybil regions.

Qiang then explained the details of how the ranking is computed. In order to handle multiple communities within large social networks, initial trust-seeds are picked from different communities. This process is performed by detecting communities and then hand-labeling a set of seed nodes in those communities as real or fake. Trust is then pushed along social links, although only a small number of iterations are performed, in order to produce the same effect as if nodes individually made short random walks.

Qiang concluded the talk with evidence that the method is effective. The algorithm was tested against the Stanford large network data set and remained effective even with a high number of attack edges. It is also deployed on the Tuenti network. In the Tuenti context they found several different structures of Sybil communities, and, more importantly, over

90% of the nodes that the algorithm ranks as least trusted are Sybil accounts.

Two questioners asked about the distribution of Sybils in the network, wondering if some would appear as a crust around the edge of the graph, or as disjoint nodes rather than as communities. Qiang responded that graph schemes tend to use attack edges to detect Sybils, and this approach would also likely miss Sybils which are well integrated with real users compared to other Sybils. Srinivas Narayana from Princeton asked how humans would classify nodes as fake. Qiang explained that they use lots of factors, including correlating photos with age and address, checking the posts an account has sent, and the IP addresses used for login. Another questioner asked whether the same mechanism would work in a peer-to-peer context, and Qiang answered that while this solution is meant for social networks, where there is a global view of the network, some of the techniques could also apply in the distributed context.

Don't Lose Sleep Over Availability: The GreenUp Decentralized Wakeup Service

Siddhartha Sen, Princeton University; Jacob R. Lorch, Richard Hughes, Carlos Garcia Jurado Suarez, and Brian Zill, Microsoft Research; Weverton Cordeiro, Universidade Federal do Rio Grande do Sul; Jitendra Padhye, Microsoft Research

Siddhartha presented the GreenUp service, which keeps desktop computers in an enterprise available, even when most of them are asleep. He began by laying out the problem space that GreenUp explores: desktop computers in companies use a lot of power and many are left on at night. There's a tension between the energy savings that accompanies putting machines to sleep, and the desire to have at-will connections or perform upgrades on machines. He then explained the motivation for GreenUp: requiring dedicated servers to manage machine state was expensive, and transitioning machines to VMs would require disruptive changes to the desktop environment.

Siddhartha then described the design of the service. In order to maintain accurate state of which machines are active and which are not, all active machines are set to randomly and repeatedly probe other machines on the subnet. By carefully choosing the number of probes to send, there is a high probability that all machines will get probed, and an awake machine will notice a state change soon after a computer goes to sleep. A resolution protocol resolves conflicts if multiple machines notice at the same time, so that only one awake machine will become the manager of the asleep machine. GreenUp takes care of several important points to make this work: machines use subnet broadcasts to distribute their IP and MAC addresses, and listening ports to other machines

so that they can become managers. Enough machines need to stay awake for the service to work effectively, and if a manager falls asleep another needs to take over for both it, and all the machines it was managing.

Siddhartha then covered the evaluation of the system. GreenUp was deployed at Microsoft on 1100 machines and was reliably able to wake up machines 99% of the time, with most errors caused by Wake-on-LAN issues. Of wake-ups, 87% took less than nine seconds, and 85% occurred before the incoming connection was closed (13% of connections gave up in under three seconds, and were likely port scanners or other automated systems). In their deployment, GreenUp never consumed more than 7% of CPU resources. They simulated larger management workloads and found that a manager could handle up to one hundred machines without noticeable CPU impact. Finally, they found that with GreenUp, machines were asleep an average of 31% of the time in their environment, but this was due to an IT-mandated sleep policy. In time, they believe GreenUp's availability guarantees will make more users comfortable, allowing their machines to sleep for increased savings.

After the talk, Bryan Ford from Yale asked if they needed to have a server for each subnet, rather than using VLANs to span larger areas. Siddhartha responded that in Microsoft's environment, the hundreds of separate subnets made this impractical, and they didn't want to send broadcast messages to large segments of the enterprise. With follow-up prompting from David Anderson at Carnegie Mellon, he added that in an ideal world an alternative solution to this problem would be for employee desktops to run in VMs that are migrated efficiently to awake servers when inactive. A second focus of questions was on the applicability of this method. Syed Amin from Yale asked about security issues, and Aaron Gember from the University of Wisconsin asked if it would work on home networks. Siddhartha responded that the focus had been on the enterprise domain and that the design isn't meant for an environment with malicious devices or only a small number of machines.

Data Center Networking

Summarized by Katrina LaCurts (katrina@csail.mit.edu)

Jellyfish: Networking Data Centers Randomly

Ankit Singla and Chi-Yao Hong, University of Illinois at Urbana-Champaign; Lucian Popa, HP Labs; P. Brighten Godfrey, University of Illinois at Urbana-Champaign

Chi-Yao Hong presented Jellyfish, a new topology for data center networks. Old-school datacenter networks are constrained to tree topologies, which make it easier for span-

ning-tree protocols to operate. Today, however, we can build topologies with much more freedom. Data center topologies have two goals: high throughput and incremental expansion. The second goal is of particular interest in industry. Topologies with structure, e.g., hypercubes and fat trees, constrain this type of expansion, since they require a particular number of switches.

Jellyfish's solution is to forget about structure. Each switch uses some ports to connect to the servers, and the other ports to form a random graph with the other switches. To add nodes, it breaks an existing link and connects it to the new switch. This same procedure works for graph construction as well as incremental expansion. As a result, Jellyfish is cheaper and provides better incremental expansion than existing work. It also achieves higher throughput than fat-tree topologies, especially when there are many servers.

Jellyfish achieves high throughput by making the average path length shorter. When using all capacity, the problem of increasing throughput is equivalent to decreasing mean path length. In a fat-tree topology, most of the edges are used for redundancy rather than to decrease path length. In fact, Jellyfish is within 9% of the throughput for the best known degree-diameter graphs (those graphs represent the theoretically optimal throughput). These types of graphs are difficult to find and are not as expandable.

Jellyfish also addresses the problem of routing/congestion control and cabling. The authors found that using k-shortest paths for routing worked well combined with either TCP or Multipath TCP. For cabling, Jellyfish topologies actually have fewer switch-to-switch cables than fat-tree topologies. It even retains its gains when constrained to only using the same number of long cables as in a fat-tree topology. Long cables are of interest here because they're quite expensive.

In response to a question from Rohan Gandhi (Purdue), Chi-Yao noted that Jellyfish is quite robust, because if a switch fails, the topology is still a random graph. Siddhartha Sen (Princeton) had two concerns: one about the effect on routing table state (since we can no longer aggregate prefixes), and another about how Jellyfish affects multipath protocols. Chi-Yao pointed out that for the routing tables they can use dictionary lookup, but was unsure about the effect on state or on multipath protocols.

Ben Zhao (UCSB) asked what happens when the network is full, i.e., you've added all of the links and filled up all the slots, and have a full clique rather than random paths. Chi-Yao responded that most data centers aren't large enough for that to happen, but that if your budget allows for a full mesh, then that's actually great.

The last question came from Charlie Hu (Purdue). Since Jellyfish gives up hierarchy, how easy is it to make allocations? The authors are exploring this problem now and have found that you can impose a structured graph on top of the random graph with only a small loss of path length.

OSA: An Optical Switching Architecture for Data Center Networks with Unprecedented Flexibility

Kai Chen, Northwestern University; Ankit Singla, UIUC; Atul Singh, Kishore Ramachandran, Lei Xu, and Yueping Zhang, NEC Labs America, Inc.; Xitao Wen and Yan Chen, Northwestern University

Kai Chen presented OSA, an all-optical data center architecture. Today's data centers provide the infrastructure for many "big data" applications. Conventional three-tiered data centers can be problematic, causing communication bottlenecks due to oversubscription. Fat-tree and B-cube architectures provide high bandwidth but have high wiring complexity, use lots of power, and cost a lot. Hybrid (electrical + optical) efforts reduce complexity, power, and cost, but provide insufficient bandwidth.

The insight into the OSA work is that data center traffic exhibits regionality and some stability. By using optical link technology, OSA can dynamically change its top-of-rack (ToR) topology and link capacity to adapt to changing traffic matrices. OSA takes advantage of two technologies in particular: MEMS, switches that allow any input port to be connected to any output port (providing the flexible topology), and WSS, switches that allow for flexible link capacity.

As a result, OSA can achieve any k -regular topology with flexible link capacity among the ToRs. Currently, OSA is a container-sized data center, with a logically centralized control plane that can optimize the network to better serve the traffic. OSA can use systems such as Hedera to estimate traffic demand for the purpose of assigning links and wavelengths. In the prototype implementation, the bisection bandwidth is very close to the theoretical bandwidth. In simulation, OSA is almost non-blocking. Additionally, OSA costs less and uses less power and wiring than fat trees, and has better performance than hybrid architectures (and typically better cost/power/wiring as well). One caveat, however, is that OSA is not intended for all-to-all, non-stable traffic.

In response to a question from Srinivas Narayana (Princeton), Kai noted that they observed traffic stability in production networks on the order of seconds to minutes, which is appropriate, since OSA only takes ~100 ms to react to a change in traffic.

Less Is More: Trading a Little Bandwidth for Ultra-Low Latency in the Data Center

Mohammad Alizadeh, Stanford University; Abdul Kabbani, Google; Tom Edsall, Cisco Systems; Balaji Prabhakar, Stanford University; Amin Vahdat, Google and U.C. San Diego; Masato Yasuda, NEC Corporation, Japan

Mohammad Alizadeh presented HULL, an architecture for achieving both low latency and high bandwidth in data centers. Many data centers run low-latency applications today (e.g., high-frequency trading, RAMClouds). They would like predictable low-latency delivery of individual packets. With large-scale Web applications, where data is typically stored on machines separate from the ones containing the application logic, we have to communicate across the network to get data to an application, limiting the data access rate. Even the latency in the tail is important: the application won't respond if one operation is taking a long time. The goal of HULL is to reduce delay—in particular, queueing delay—to zero.

HULL first uses DCTCP (Data Center TCP) to allow the network to react to congestion in proportion to the extent of the congestion. DCTCP is helpful here, but does not get latency down as far as we'd like. The main idea in HULL is to use "phantom" queues, which signal congestion before queueing occurs. This approach sacrifices some bandwidth, but lowers latency. Since data center traffic often exhibits both mice and elephant flows, HULL can steal some bandwidth from the larger flows without severely impacting their performance. Unfortunately, because TCP traffic is bursty, to get down to near zero latency we must sacrifice a lot of bandwidth. To deal with this, HULL implements hardware pacers, which dynamically figure out at what rate to pace. These pacers bring the 99th-percentile latency down to almost zero.

HULL implements hardware pacing at the network edge, DCTCP in the racks, and phantom queues at the switches. In simulation and on a 10-server testbed, the authors saw a 93% decrease in latency for roughly a 17% increase in completion time.

John Dunagan (Amazon) asked how DCTCP interacts with workloads that don't typically run over TCP, as well as how much latency endhost software adds. Mohammad responded that, in principle, you could apply HULL to non-TCP flows, and that we could likely optimize out the overheads of software. Mark Handley (UCL) asked how long HULL takes to yield resources to a new flow. Mohammad showed that it's only a constant factor increase on top of TCP. Jeongkeun Lee (HP Labs) asked why HULL uses hardware pacing rather than software pacing, and Mohammad pointed out that software pacing requires disabling things such as LSO. In response to Costin Raiciu's (University Politehnica Bucha-

rest) question, “Why not just use two priority queues?” Mohammad noted that one usually assigns an application, not a specific flow, a priority. It is much harder to use priority queues at a flow level and to dynamically assign priorities.

Big Data (2)

Summarized by Andrew Ferguson (adf@cs.brown.edu)

PACMan: Coordinated Memory Caching for Parallel Jobs

Ganesh Ananthanarayanan, Ali Ghodsi, and Andrew Wang, University of California, Berkeley; Dhruva Borthakur, Facebook; Srikanth Kandula, Microsoft Research; Scott Shenker and Ion Stoica, University of California, Berkeley

Modern data parallel clusters are increasingly capable of storing large amounts of data in RAM. For analytic processing jobs, such as those executed by Hadoop or Dryad, caching file inputs can result in a significant performance boost. However, because of synchronization steps, or barriers, which occur after some execution stages, a single uncached input can eliminate the end-to-end latency improvement from caching other inputs in the same stage.

Ganesh Ananthanarayanan presented PACMan (Parallel All-or-nothing Cache Manager), a service for data parallel clusters that coordinates file caching across the cluster. PACMan uses a global view of the cluster to focus evictions on incompletely cached stages, stages for which not all compute nodes are able to cache the inputs. This approach ensures that stages with inputs fully cached across the cluster do not lose caching’s benefits.

PACMan can use two metrics to determine which file to evict. The first is user-centric, minimizing jobs’ completion times, and the second is system-centric, maximizing the utilization of the cluster. The service contains two cache eviction policies: LIFE, which is optimized for the first metric, and LFU-F, which is optimized for the second. Details of these policies can be found in the paper. PACMan has been evaluated using traces of Facebook’s and Bing’s production workloads, and reduced average job completion time by 53% and 51%, respectively.

Because caching reduces disk accesses, Ananthanarayanan was asked about its importance relative to other resources, such as CPU. He replied that Map-style phases, which are generally disk-intensive, account for 60–70% of job completion times. Sriram Rao asked about the effect of adding more jobs to the cluster. Performance would continue to improve, but only to a limit. John Ousterhout asked about the ability to cache data parallel jobs over the long term. Ananthanarayanan replied that as long as job sizes continue to follow a power law-like distribution, there will be small jobs which can be cached effectively.

Reoptimizing Data Parallel Computing

Sameer Agarwal, University of California, Berkeley; Srikanth Kandula, Microsoft Research; Nico Bruno and Ming-Chuan Wu, Microsoft Bing; Ion Stoica, University of California, Berkeley; Jingren Zhou, Microsoft Bing

The performance of data parallel jobs running in Hadoop or Dryad clusters is heavily influenced by task parallelism, input data partitioning, and implementation and sequence of operators such as Map, Reduce, or Join. These settings are described by the job’s execution plan, which can be determined manually, or automatically using frameworks such as Hive, Pig, or SCOPE. Unfortunately, configuration choices made at a job’s start may not be optimal later.

Sameer Agarwal presented RoPE, a Re-optimizer for Parallel Executions, which uses statistics from recurring and similar jobs, as well as previous stages of the same job, to improve execution plans. RoPE captures both data properties, such as cardinality and common values, and code properties such as CPU consumption and peak memory utilization. The key challenge is to measure these properties with both high accuracy and low overhead.

RoPE provides statistics modules that can be integrated into a job’s execution, as they are distributable, maintain sub-linear state, and are restricted to a single pass over the data. The results are then collected and composed as inputs to an existing query plan optimizer. Using RoPE’s improved inputs in Bing’s production clusters yielded a 2x improvement in job latency at the 75th percentile while using 1.5x fewer resources, at a cost of only 2–5% increased overhead.

Agarwal was asked about applying RoPE’s techniques to other frameworks such as DryadLINQ. He replied that the feasibility depends upon such frameworks’ own plan optimizers, as the statistics gathered by RoPE are quite general. Matvey Arye asked about potential benefits from users annotating their operators with statistics. The presenter found that while SCOPE supports such annotations, in practice users do not have the necessary knowledge to make use of them. Rohan Gandhi wondered about the consistency of the collected statistics. The authors considered additional statistics, but restricted RoPE to those which remain stable.

Optimizing Data Shuffling in Data-Parallel Computation by Understanding User-Defined Functions

Jiaying Zhang and Hucheng Zhou, Microsoft Research Asia; Rishan Chen, Microsoft Research Asia and Peking University; Xuepeng Fan, Microsoft Research Asia and Huazhong University of Science and Technology; Zhenyu Guo and Haoxiang Lin, Microsoft Research Asia; Jack Y. Li, Microsoft Research Asia and Georgia Institute of Technology; Wei Lin and Jingren Zhou, Microsoft Bing; Lidong Zhou, Microsoft Research Asia

Today’s data parallel frameworks such as Hadoop and Dryad require shuffle steps during which output data from one

stage is re-sorted and re-partitioned to become the input data for a second. Because shuffle steps can require all-to-all communication between processing nodes, they are both time-consuming and resource-intensive. Zhenyu Guo presented statistics showing that shuffle operations cause more than half of all cross-pod communication in a production system at Bing.

To eliminate unnecessary shuffle steps between stages, the authors developed an optimization framework called Sudo, which has three components. First, Sudo tracks the partitioning properties of the data as it is being processed: for example, is the data hash-partitioned? is it range-partitioned? is it sorted within partitions? Second, Sudo determines which functional properties are required by the user code processing the data; perhaps the code expects the input data to be monotonically increasing. To determine these functional properties, Sudo performs program analysis on user-defined functions and can incorporate user-provided annotations as well.

Finally, Sudo determines the least-costly operation that will transform a stage's output data such that it will have the properties required for input to the subsequent stage. Ideally, data would not need any shuffling between stages. In other cases, Sudo may find that a local sort within a partition is necessary and sufficient. The paper contains further details about Sudo's reasoning and the properties present in various data-partitioning strategies. Experimental results show that Sudo can save up to 47% of disk and network I/O costs incurred by naive shuffling in Bing's production systems.

After the presentation, Mike Freedman asked how the data-partitioning and functional properties Sudo tracks were chosen. Guo replied that the data-partitioning properties were the ones provided by the underlying system, and the functional properties were the ones used in production.

Poster Session

Summarized by Karthik Nagaraj (knagara@cs.purdue.edu)

SNMiner: A Rapid Evaluator of Anomaly Detection Accuracy in Sensor Networks

Giovani Rimon Abuaithah and Bin Wang, Wright State University

Deployments of sensor nodes consist of many nodes constantly recording environmental measurements and later transmitting them to a central place for analysis. It is possible that some of these sensors are either returning faulty readings or have been compromised, leading to wrong readings. This work describes the design of a framework for identifying anomalous behavior in sensor networks, using statistical and data mining techniques. Historical data from

nodes can help predict if a node has suddenly started exhibiting faulty behavior. This framework allows introduction of faulty readings within the network through different fault models, identifies faulty readings, and removes those from the collected data. The authors have developed a framework to collect all the sensor data so that different analysis techniques can be easily performed on the data and the detection accuracy of such techniques can be re-evaluated over time.

BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data

Sameer Agarwal and Aurojit Panda, UC Berkeley; Barzan Mozafari and Samuel Madden, MIT CSAIL; Ion Stoica, UC Berkeley

With the recent explosion of data on the Web, many user decisions and activities can be improved, but they depend on the ability to process this data quickly. With such data quickly aggregating to terabytes and even petabytes, even the use of large compute clusters cannot easily produce query results within a few seconds. An interesting tradeoff is explored in this work between processing large volumes of data and producing results within deadlines, by tweaking the precision of the results. BlinkDB aims to produce slightly less accurate results within a time bound by running the queries across samples of the data. Using historical query information to gather usage patterns to create intelligent multi-dimensional stratified samples, BlinkDB achieves high accuracy with low storage overhead. It also caters to different query time bounds by using samples of different sizes. Some initial experiments show that BlinkDB can process 17 TB of data on 100 nodes in under 2 seconds, with 2–10% error.

PhoneLab: A Large-Scale Participatory Smartphone Testbed

Rishi Baldawa, Micheal Benedict, M. Fatih Bulut, Geoffrey Challen, Murat Demirbas, Jay Inamdar, Taeyeon Ki, Steven Y. Ko, Tevfik Kosar, Lokesh Mandvekar, Anandathirtha Sathiyaraja, Chunming Qiao, and Sean Zawicki, University at Buffalo, The State University of New York

PhoneLab is the initiative, equivalent to PlanetLab, to enable testing of smartphone applications and protocols. Smartphones are ubiquitous—every other mobile phone user carries a smartphone and uses it to access the Internet, take pictures, play music, control other devices, etc. However, research and development of new technologies for smartphones is still limited to small testbeds with a handful of devices, limiting extensive testing of ideas under realistic workloads. PhoneLab aims to solve this problem by deploying Android smartphones to participating users who would use them normally. Researchers and developers, on the other hand, can provide new applications, tweak underlying operating system functionality and protocols, and collect

usage data or patterns. The use of these phones by real people provides ample opportunity to analyze new techniques and identify new research problems with smartphones. Two of the toughest challenges that hamper the setup of PhoneLab are funding for the devices themselves, and charting out guidelines for the use of the phones by participants.

Precise Anomaly Detection in Network Flows

Sriharsha Gangam, Purdue University; Puneet Sharma, HP Labs; Sonia Fahmy, Purdue University

Actively monitoring network flows is an important and challenging task for network operators, allowing them to verify the current state of the network and also identify faulty or malicious behaviors. With faster networks it is difficult to collect and analyze large volume of flow records at current line rates (of the order of millions of flows per minute). Existing monitoring solutions resort to sampling and sketching, which are lossy in nature and provide only approximate answers. This work advocates the use of co-located compute and storage units (blades) to the switches to provide a distributed passive monitoring infrastructure. With such in-network processing units, aggregation queries and network wide anomalies can be adaptively mined to reduce the communication overhead and improve the anomaly detection accuracy. Since anomalies can span multiple devices and a centralized approach is too expensive, this work provides techniques for anomaly detection in a decentralized manner. Briefly, the detection starts by aggregating coarse-grained flow summaries and iteratively converges to the solution by successively gathering finer-grained flow summaries on demand.

Scaling WiFi Performance for Large Audiences (poster and demo)

Jeongki Min, Arpit Gupta, and Injong Rhee, North Carolina State University

Arpit Gupta described how WiFi scales poorly at access points with many subscribers, mainly because traffic characteristics in such setups produce an asymmetry in sharing the physical medium. Analysis of traffic patterns in large audiences indicates that 76% of the traffic is incoming TCP data, from large request responses. In essence, access points need higher priority to transmit on the wireless medium, whenever download data from the wired network starts to backlog. This work explores the technique for dynamically boosting the access point priority as a logarithmic function of queue size. They further show that prioritization improves good-put of transmission and also reduces re-transmissions.

Non-Linear Compression

Michael F. Nowlan and Bryan Ford, Yale University

Compression is an integral part of many modern systems. However, the popular multicore and parallelization movement is at odds with traditional linear compression schemes, which impose sequential inter-block dependencies on applications. Unfortunately, serial processing significantly impacts parallelism on newer multi-core architectures and also forces network packets to be decompressed in order. Non-Linear Compression fills this gap by supporting both traditional linear compression similar to gzip and “modern” parallel and adaptive compression. NLC gives applications full control on where to form inter-block dependencies, structuring relationships as a parent-child hierarchy. This allows compression and decompression in parallel and independent of other blocks, so applications can make progress with only a subset of the data. The evolving history from all children can later be reconciled intelligently to avoid losing much compression ratio. This work pairs well with distributed storage as well as recent work on uTCP that allows developers to use unordered reliable transmission of packets, which can then be possibly compressed.

Performance Isolation and Fairness for Multi-Tenant Cloud Storage

David Shue and Michael J. Freedman, Princeton University; Anees Shaikh, IBM T.J. Watson Research Center

Cloud storage services are very popular among cloud users, because of high availability and reliability of such storage mediums. However, as the number of users increases, resource fairness gains serious concerns, as users would like to get performance equal to their expense. Currently, cloud providers strive to provide fairness between various virtual machines (VMs) executing on a single host but fail to provide guarantees on other shared infrastructures such as the storage service, network, etc. This work aims to provide weighted fair shares and performance isolation between users sharing a storage layer, such as Amazon S3, through a novel combination of four techniques. Using partition placement, weighting, replica selection, and fair queuing, storage requests are made more streamlined and amenable to performance fairness. Even in situations where client requests are unpredictable and may be bursty, these techniques keep fairness under control.

This same group also had a demo of Serval (presented as a paper), where they demonstrated playing a video on a smartphone without losing frames, while disabling different network types (WiFi, 4G).

Composable Reliability for Asynchronous Systems

Sunghwan Yoo and Charles Killian, Purdue University; Terence Kelly, HP Labs; Hyoun Kyu Cho, University of Michigan; Steven Plite, Purdue University

Distributed systems are traditionally implemented to handle failures using timeouts—a technique that treats really slow nodes as having failed, and restarting them. With the advent of more managed environments, such as data centers and campus setups, observed network delays are real delays, and nodes are quickly restarted if they fail. Therefore, if the state of the process were not lost, applications could be recovered in a cleaner and quicker manner. This work outlines the design and implementation of MaceKen, a transparent distributed systems programming framework that allows application developers to avoid dealing with process crashes and proceed on the assumption of slow nodes. MaceKen achieves this through carefully crafted techniques to persist the state of the application process and network messages sent or received, so that restarted nodes can resume execution from the last correct checkpoint. MaceKen marries the persistent protocol Ken with the popular Mace programming toolkit, so that applications can be easily developed in Mace and easily leverage the benefits of Ken.

New Architectures and Platforms

Summarized by Kevin Ngo (ngoke@onid.oregonstate.edu)

XIA: Efficient Support for Evolvable Internetworking

Dongsu Han, Carnegie Mellon University; Ashok Anand, University of Wisconsin—Madison; Fahad Dogar, Boyan Li, and Hyeontaek Lim, Carnegie Mellon University; Michel Machado, Boston University; Arvind Mukundan, Carnegie Mellon University; Wenfei Wu and Aditya Akella, University of Wisconsin—Madison; David G. Andersen, Carnegie Mellon University; John W. Byers, Boston University; Srinivasan Seshan and Peter Steenkiste, Carnegie Mellon University

Hyeontaek Lim introduced a new network architecture, eXpressive Internet Architecture (XIA). XIA provides a unique foundation for diverse communication styles. Lim first pointed out flaws in today's Internet architecture, specifically IP. IP is described as “the narrow waist of the Internet,” an outdated model for host-centric communication that is still being used despite increasing demand for service and content-oriented communication. XIA seeks to support heterogeneous communication types (e.g., service, content, mobility, cloud) on a single Internet architecture rather than focusing on one communication type, all while being extensible for future communication types.

Lim introduces two primary design pillars: “principal types” and “fallbacks.” Principal types allow the defining of ad

hoc communication models in XIA. The current Internet architecture has one principal type, the IP address. In XIA, a principal specifies a type and a type-specific identifier (e.g., service type and hash of service's public key). Each principal has type-specific semantics to allow XIA routers to logically process it. XIA routers do not have to support all principal types, although they must support host-based communication. However, there must be a way of supporting legacy routers that would not understand these new principal types.

Fallbacks are mechanisms that allows incremental deployment of new communication types and backwards-compatibility. With direct acyclic graph (DAG) addressing, intent is encoded into packets. If necessary, the architecture can fall back to different routing choices if the intent of the packet is not understood by a router. DAG addresses, routes with indicated possible fallbacks, are encoded into packet headers (the encoding method can be found in the paper). Fallbacks can even be nested. With a click-based implementation on commodity hardware with 351,000 table entries based on a Route Views snapshot, they found that with XIA routers, forwarding provides throughput comparable to that of high-speed IP routers with minimal slowdown for small packets with three fallbacks. The prototype of XIA is hosted on GitHub at <https://github.com/XIA-Project/xia-core>, which supports LAN, XIA-over-IP, and GENI.

An attendee wasn't clear whether the source address in XIA was the legacy source address or a DAG. Lim replied that there is a symmetry between the destination address and the source address, and that the source address can be a service to allow flexible communication between any pair of entities in the network. Another attendee noted that the original Internet was intended to be evolvable and extensible but over time that idea was lost, for several reasons, and asked how, if deployed, XIA would avoid the noted pitfalls that TCP and IP fell into. Lim answered that XIA had a security feature, intrinsic security, by which the host identifier can be used to identify the source of a packet, which can be used to amend some problems on the Internet. The attendee followed up with the reservation that people would end up adopting only a select few principal types and asked if the architecture could prevent that. Lim responded that they had not addressed that at this point. David Oran remarked that content-networking people preferred no source addresses at all and asked how Lim would answer that. Lim supported source addresses for security reasons and noted that they do not hinder flexibility in content networks. Oran also asked why they did not use partial orders rather than DAGs. Lim did not find much difference between them and found no side-effect of using DAGs over partial orders.

Design and Implementation of a Consolidated Middlebox Architecture

Vyas Sekar, Intel Labs; Norbert Egi, Huawei; Sylvia Ratnasamy, UC Berkeley; Michael K. Reiter, UNC Chapel Hill; Guangyu Shi, Huawei

Vyas Sekar talked about a new approach for building and managing middlebox deployments. As networks evolve, new applications and devices appear, along with evolving threats to exploit these applications. Then, as companies move these applications to be Internet-enabled, there is a question about policy compliance. Looking at an enterprise network, Sekar's group found that rather than changing routers and switches, the current standard was to add specialized appliances, or middleboxes. Middleboxes are often shipped with narrow interfaces that can be frustrating to manage. As network requirements grow, the only option for network operators is to buy more of these middleboxes, which is not only costly but limits extensibility and flexibility. This work seeks to consolidate the building of individual middleboxes and the managing of a network of middleboxes.

CoMb is top-down design with a logically centralized middlebox controller. On the management layer, the goal was to balance the load across the network and to assign processing responsibilities across different middleboxes. However, reuse and policy dependencies made this difficult. The need for explicitly capturing these dependencies is eliminated if all applications relating to a given session run on the same node. Sekar introduced hyperapps, which take the logical union of all the actions that need to be run on a given packet, so that common actions are not duplicated. With hyperapps in place, the management problem becomes a simple and near-optimal program. Sekar went on to explain the design in depth, which consists of a policy enforcement layer that logically routes packets.

Sekar then showed that consolidation has a low overhead for existing applications, takes little time to index a network, has five times the throughput of VM-based consolidation, sharply reduces maximum load many-fold, and cuts provisioning costs by about half. Addressing isolation, he noted that CoMb currently relies on process-level isolation and leverages user-space for high-performance networking. He also addressed the problem of changing vendor business models and said that consolidating middleboxes is already happening in the form of virtual appliances and that with the benefits it's likely that someone will adopt it.

An attendee had reservations about using CoMb in the presence of topology constraints. Sekar noted that the paper addresses those cases. Another attendee from Cambridge asked about hidden policies in middleboxes. Sekar responded that the policies are not coming from the middlebox vendor

but, rather, from the network operator running an optimization. The attendee then plugged a project from Cambridge, Mirage, which addresses an isolation-reuse problem, which pleased Sekar. John Dunagan asked why they did not run everything on every server and simply choose the number of servers, which is the alternative he saw becoming adopted. Sekar replied that in some sense that is the kind of resource management they are doing by removing the physical coupling. Dunagan followed up and remarked that if every server provided uniform functionality, rather than policy specification, an easier approach to sizing would be to route one person to the traffic to measure CPU utilization. Sekar responded that policy specification was more for deciding what kinds of traffic needed what kinds of processing than for resource management. Aaron Gembar asked whether there was a way to get expected resource consumption. Sekar affirmed and said they had thoughts on more fine-grained fairness issues. Michael Sirivianos referenced an earlier point and asked why different middleboxes had different peak times. Sekar said different boxes have different types of functionality, which causes non-overlapping traffic. Lastly, Dongsu Han wondered whether they accounted for vendor-specific hardware optimizations and, if so, whether that created issues. Sekar responded that if an application has an affinity for a particular sort of hardware, it will be run on that hardware.

An Operating System for the Home

Colin Dixon, IBM Research; Ratul Mahajan, Sharad Agarwal, A.J. Brush, Bongshin Lee, Stefan Saroiu, and Paramvir Bahl, Microsoft Research

Colin Dixon presented HomeOS, a home operating system that eases extensibility and management by providing a PC abstraction for home technology. The motivation was to bring to life the long-envisioned smarthome, where devices are able to come together to cater to wants and needs such as climate control and energy monitoring. They talked to homeowners of existing smarthomes and found that although the smarthomes were convenient, they had poor extensibility and were frustrating to manage. It was difficult to add new devices and configure each individual device with access control. Dixon presents two existing abstractions for technology within the home: a network of devices with interoperability protocols, and fixed appliances with fixed tasks. A network of devices is easily extensible but difficult to manage, whereas appliances are easier to manage, due to their closed nature, but extensibility is difficult. To achieve both extensibility and management, Dixon urged us to view the home as a computer where adding devices consists of plugging in a peripheral and adding tasks consists of installing an application.

Applications run on top of HomeOS and communicate with the devices using drivers. This way, users can interact with HomeOS's centralized high-level interface rather than interacting with individual devices. Dixon presented three challenges with HomeOS in the field. First, non-technical users suddenly have to become network managers. Second, heterogeneity can make it difficult for developers to build applications. Lastly, it can be difficult for HomeOS to keep up with new classes of devices and applications that arrive frequently. HomeOS respectively addresses these challenges with management primitives that align with the user's mental models, with protocol-independent abstract APIs for devices, and with a kernel agnostic toward device functionalities. These goals are mapped onto a stack with layers for device connectivity, device functionality, management, and applications. This stack respectively covers the heterogeneity of topology, devices, control, and tasks.

Dixon then did a live demo of HomeOS on a laptop. He installed an application from a list of compatible applications from the HomeStore. Through a wizard, two cameras were slowly added and configured, and the video from the cameras were shown on-screen. Dixon's group evaluated HomeOS based on usability and extensibility, using field experiences and controlled experiments. They found that the field experiences were positive; users could manage their HomeOS deployments, and developers found the programming abstractions and layering to be natural. Still, users found it hard to diagnose faults, interoperability protocols could be fragile, and not all device features were exposed over the network. In the controlled experiments, they found that most non-technical users were able to complete management tasks without training and found that developers were able to very quickly write one of two realistic applications.

Where would HomeOS physically be located within the home? HomeOS would run on a single Windows PC, where it would have access to the network to communicate with devices. What about conflicting devices? There was little to do to stop two devices conflicting. Would there be configuration management for devices? There would not be.

Cloud Performance

Summarized by Advait Abhay Dixit (dixit0@purdue.edu)

Structured Comparative Analysis of Systems Logs to Diagnose Performance Problems

Karthik Nagaraj, Charles Killian, and Jennifer Neville, Purdue University

Karthik Nagaraj started with a performance comparison of two implementations of BitTorrent: Azureus and Transmission. Transmission was about 20% slower than Azureus in all executions. For this performance problem, logs did not

contain any error messages. Due to non-determinism and concurrency in the system, a line-by-line comparison of logs from Azureus and Transmission is not possible. To address such performance problems, the authors developed Distalyzer, a tool that does a comparative analysis of logs to find the root cause of a performance problem.

Distalyzer automatically analyzes large volumes of logs using machine-learning techniques. Distalyzer first extracts features (such as states and events) from the logs that represent characteristics of the execution. Then it uses predictive modeling to extract those features that diverge between the two sets of logs. Divergence in features may not impact performance, so Distalyzer identifies dependencies between system components and performance using dependency graphs. It then adds weights to the edges by exploiting the underlying statistical model and weighs the node sizes based on divergence between the two sets of logs. After pruning edges with low weights, the root-cause graph is output to the application developer for analysis.

The presenter briefly described case studies on three mature distributed systems, which uncovered six performance problems. They uncovered a constant delay (sleep) in Transmission, which, although it was written in C, was slower than Azureus (Vuze), which was written in Java, because of this added delay.

Distalyzer is available at: <http://www.macesystems.org/distalyzer/>.

Srinivas Narayana (Princeton University) asked for guidelines for developers on how to record state and event transitions in logs. Karthik responded that the amount of logging to add is a known hard problem. Adding too many logs increases runtime overheads, but adding too few logs does not tell you enough about the system. There is no good guideline on this, but Distalyzer allows developers to use their existing high-performance logging infrastructure. Later, developers can use a parser to translate their logs for analysis by Distalyzer, and that worked well for them. Fitz Nowlan (Yale University) asked if they have a graph for Transmission and Azureus after they fixed the performance problem. Karthik displayed the graph. Transmission and Azureus had identical performance.

Orchestrating the Deployment of Computations in the Cloud with Conductor

Alexander Wieder, Pramod Bhatotia, Ansley Post, and Rodrigo Rodrigues, Max Planck Institute for Software Systems (MPI-SWS)

Alexander Wieder listed the choices available to a user who wants to run a MapReduce application using Amazon's cloud service. The user has choices for storage (Amazon's S3 storage, local storage in EC2 nodes) and computation (EC2,

other private infrastructure). Choosing a strategy for deploying cloud applications becomes challenging because of the variety of services and providers available to the user, where each one has different performance characteristics, pricing models, and interfaces.

Conductor is a system that chooses a strategy for deployment of applications in the cloud. It optimizes resource allocation with a set of user goals (e.g., a deadline or reducing monetary costs) and resources available for use. Conductor optimizes by generating a formal linear programming-based execution model and feeding it to a LP-solver along with the user goals. Conductor also provides a resource abstraction layer which sits between the MapReduce framework and the actual resources. The abstraction layer provides abstractions for storage and computation. This allows the cloud application to use different computation and storage services simultaneously. Once the application is deployed, Conductor monitors the execution and changes the deployment to adapt to performance variations or changes in price.

Using EC2 and S3, Wieder described an experiment that showed how Conductor finds the optimal execution plan for a given job completion deadline. The experiment also illustrated how Conductor adapts the execution (by allocating more EC2 nodes) when the observed performance of EC2 nodes is lower than the estimated performance.

Dongsu Han (Carnegie Mellon University) asked about the Amazon policy of having different pricing for different regions and whether Conductor handles this case. Alex said that they can handle this case. This adds to the diversity of products available. In the model, each instance type can be considered as a different service in each region, with different prices but the same performance characteristics. Dongsu Han then asked if computation costs generally dominate over bandwidth costs, and Alex said that it is very hard to come up with realistic cases of what people usually do with a cloud. He couldn't really say whether jobs are computationally bound or I/O bound.

Transport

Summarized by Advait Abhay Dixit (dixit0@purdue.edu)

Fitting Square Pegs Through Round Pipes: Unordered Delivery Wire-Compatible with TCP and TLS

Michael F. Nowlan, Yale University; Nabin Tiwari and Janardhan Iyengar, Franklin and Marshall College; Syed Obaid Amin and Bryan Ford, Yale University

Janardhan Iyengar began by saying that currently, applications have a choice of just two transport protocols: UDP and TCP. While UDP offers no guarantees, TCP guarantees in-order delivery at the cost of performance. This is an all-or-

nothing choice. The authors' goal is to break up the functionality of the transport layer to pieces that can be composed by the application. The architecture, called Minion, uses TCP, UDP, and TLS as a substrate to ensure compatibility with middleboxes. The kernel component of Minion, called uTCP for unordered TCP, stores untransmitted data in priority queues on the sender side. The application can determine the order in which data is transmitted by specifying the priority in the write system call. On the receive-side, uTCP delivers unordered byte fragments to the application but guarantees that all bytes will be eventually delivered. Minion's uCOBS protocol provides a datagram delivery service atop uTCP. Janardhan spoke briefly about uTLS, which provides encrypted out-of-order delivery using TLS wire format. He illustrated the application-level benefits of Minion through an experiment in which uTCP provides low latency for high priority data.

Charles Killian (Purdue University) asked whether previously sent messages are canceled. Janardhan said that, due to middleboxes, the hard constraint is keeping the format on the wire unchanged. They can, however, suppress messages that are in TCP's queue.

How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP

Costin Raiciu, Universitatea Politehnica Bucuresti; Christoph Paasch and Sebastien Barre, Université Catholique de Louvain; Alan Ford; Michio Honda, Keio University; Fabien Duchene and Olivier Bonaventure, Université Catholique de Louvain; Mark Handley, University College London

► *Awarded Community Award!*

Costin Raiciu shared his experience in designing and implementing a deployable multipath TCP. Mobile devices with multiple interfaces that have different characteristics and coverage are becoming common. In data centers, there are multiple paths between end hosts. Content providers have multi-homed servers for robustness, but while networks are becoming multipath, TCP has remained single-path.

The authors' goal was to develop a deployable multipath TCP that works with unmodified applications and without changing network hardware and configurations. The authors used TCP options to set up the MPTCP connection and send MPTCP protocol-specific acknowledgments. This design ensures that packets sent over the wire do not differ from other TCP packets in any way other than the TCP options that they carry. Thus, MPTCP works with middleboxes that randomize sequence numbers or modify IP addresses and port numbers. However, if middleboxes remove TCP options, MPTCP reverts back to TCP. Costin also described

an optimization to improve the performance of MPTCP over multiple interfaces that have a large difference in round-trip times (such as MPTCP over Ethernet, WiFi, and 3G). Finally, Costin demonstrated a streaming video over MPTCP running on a host with Ethernet, WiFi, and 3G. Code is available at <http://mptcp.info.ucl.ac.be/>.

Vyas Sekhar (Intel) said that MPTCP makes sense in data centers, but they do not have middleboxes. So do we need it in the real world, or can we just have multiple connections? And can we do things differently for data centers? Costin replied that using multiple TCP connections instead of MPTCP will still require changing end-host applications. Even then, MPTCP behaves better during handovers. For data centers, we still need separate sequence number spaces for each connection. MPTCP has been shown to perform better than bonding multiple ports as well. Janardhan Iyengar (Franklin and Marshall College) wondered if it is possible to avoid deadlock by reserving some receive window space for the data acknowledgment. Costin said that middleboxes might remember advertised receive windows and hold or drop data that exceeds the receive window, so data acknowledgments cannot be sent if there is a receive window full of unacknowledged bytes. However, if some part of the receive window is reserved for data acknowledgments, this is possible. Bryan Ford (Yale University) asked what happens when a path and middleboxes change during the lifetime of a subflow—for example, when a network link fails. Costin replied that if things change during the lifetime of a subflow, that subflow is killed if there are other subflows available. If there are no other subflows, MPTCP falls back to TCP. This is a one-way switch, meaning MPTCP never tries to switch back to open multiple subflows again.

The TCP Outcast Problem: Exposing Unfairness in Data Center Networks

Pawan Prakash, Advait Dixit, Y. Charlie Hu, and Ramana Kompella, Purdue University

Pawan Prakash described an unfairness problem that may arise when switches employ drop-tail queues and rely on TCP to fairly allocate bandwidth to flows at a bottleneck link. Pawan described a scenario where flows from two input ports at a switch drain to the same output port. One of the input ports carries a large number of TCP flows, while the other has only a few flows. In such a scenario, the flows from the port with fewer flows do not receive their fair share of bandwidth at the output port. The happens because of a temporal phenomenon which the authors call “port blackout,” wherein many consecutive packets from one of the input ports are dropped due to synchronization of packet arrival times. Either of the two input ports may experience port

blackout. However, the smaller set of flows suffers more as a result of port blackout and hence receives a smaller share of the bandwidth at the output port. The authors observed the problem and validated their hypothesis on a testbed of 16 hosts connected in a fat-tree topology.

Ashok Anand (Bell Labs India) asked if they experimented with DCTCP where buffer utilization is low. Pawan answered that they do not have ECN support in their testbed and hence could not experiment with DCTCP, but they feel that the problem will not happen with DCTCP. Dongsu Han (Carnegie Mellon University) asked why consecutive packet drops matter. Pawan replied that flows in the large bundle respond differently to consecutive packet drops from flows in the small bundle. When the large bundle of flows sees consecutive packet drops, there is a smaller reduction in throughput. However, when the flows in the small bundle see the same number of consecutive packet drops, they suffer timeouts that result in a severe reduction in throughput. Mark Handley (University College London) pointed out that the problem is phase lock between packets and asked whether randomizing packet sizes to break phase lock would help. Pawan replied that it is something they hadn’t tried, but they plan to.

5th USENIX Workshop on Large-Scale Exploits and Emergent Threats: Botnets, Spyware, Worms, New Emerging Threats, and More (LEET ‘12)

San Jose, CA
April 24, 2012

Opening Remarks

Summarized by Rik Farrow (rik@usenix.org)

Engin Kirda (Northeastern University) opened the workshop by saying that they had decided on combining speakers from industry with academic researchers this year, with six industry and seven academic position or research papers accepted. He asserted that he wanted LEET to retain its workshop format and that people should ask questions throughout each presentation unless the speaker asked that questions be held until the end.

Engin then listed types of attacks and pointed out that social networks provide a huge new set of targets. Fabian Monrose (UNC), obviously rising to the invitation to ask questions, challenged Engin’s list, saying that there is nothing new there. Engin responded that he is not trying to predict the future, except to say that malware will become more sophisticated. He also pointed out that with virtualization and data

moving into the cloud, there will be more loss of control over data, and it will be more difficult to determine the correctness of computations.

New Challenges

Summarized by Rik Farrow (rik@usenix.org)

Challenges in Network Application Identification

Alok Tongaonkar, Ram Keralapura, and Antonio Nucci, Narus, Inc.

Alok Tongaonkar explained that Narus is a subsidiary of Boeing that focuses on traffic classification by categorizing traffic. Narus has its own research staff, as well as being a source of funding for external research into network attacks. They fund 20 research fellows.

Alok described the focus of their talk as traffic classification. In the past, most tools used the server port number as a hint for determining application protocol. But this works poorly now, because applications like BitTorrent can use any port. Another type of classification uses header features to identify the protocol—for example, similarities in packet length. Most commercial tools today use deep packet inspection (DPI), using strings like EHLO or HTTP to guess the protocol. But now we have thousands of apps (think games running within Facebook or a smartphone, which are tunneled within HTTP). And they need a way to learn about applications automatically, as there are thousands of new applications.

Rik Farrow pointed out that it's easy to hide one application within another, like steganography. Vern Paxson (UCB) suggested that Alok provide a threat model. Alok said that they want to be able to identify applications even when encrypted. It is true that applications hide within other applications and that they consider this a real problem. Niels Provos said, just to be clear, they are not trying to identify traffic where people are trying to hide their intentions. Alok said that network operators need to identify known traffic before they can even begin to consider uncovering covert channels.

Fabian Monroe asked what percentage of traffic, from Narus's point of view, they have no way of analyzing. Alok said they used `tstat` to analyze cellular traffic, and it was only able to identify 70% of the traffic, and most of that was HTTP, which is not good enough, as HTTP is used to tunnel many other apps. Vern Paxson wondered if this isn't a losing battle. How much of the traffic is over SSL? Alok said he was just worried about cleartext traffic, with hopes that their packet header analysis may work for encrypted traffic too. Most cellular data traffic is not encrypted, except for authentication.

Alok then said that one of the big challenges was how to generate signatures for 0-day applications. Paul Ferguson thought it was inappropriate that they use 0-day to identify new applications, and Alok agreed. They just want to identify new apps in traffic. But given flow sets, how do they automatically learn to match traffic for new applications. There are lots of challenges here.

Vern Paxson asked whether crowdsourcing, such as is used to create Wireshark recognizers, might work well. Alok wondered if you should trust signatures developed by volunteers for a commercial system. Vern thought that network operators would appreciate something over nothing. Also, classification is different from intrusion detection, so there are really two different types of applications. Alok agreed that there really are two types. Fabian said he liked Vern's suggestion about crowdsourcing. They looked at the traffic on two campuses and found that 13% of streams used ports that indicated encryption was in use, and another chunk was object streams, which would be opaque to analysis. Alok said that analyzing object streams is an interesting problem, but their header analysis might work there as well.

Sherlock Holmes and the Case of the Advanced Persistent Threat

Ari Juels and Ting-Fang Yen, RSA Laboratories

Ting-Fang Yen started by listing several well-publicized APT attacks: Shady RAT, Operation Aurora Google in China 2009, and against RSA in 2011 to gain access to SecureID. What makes these attacks different is that the attackers have access to lots of tools and can create new attacks; they remain in the victim's network; they are as well resourced as nation-states; and they are strongly motivated. She then pointed out the differences between traditional attackers and APT attackers, where APT attackers have different goals (espionage), objectives, and targets (individual users or systems). APT attackers are not bound by a playbook, although well-known attacks do follow a pattern that starts with spearphishing.

Ting-Fang then used four Sherlock Holmes stories to suggest other techniques that could be used: surround a victim in a more general attack that hides the attack specifics; expropriate data using an indirect method rather than simply uploading the data; conceal unauthorized communications within a commonplace object; and use other methods to bypass a physical perimeter (e.g., a robot through ductwork or digital photoframes). For example, a large botnet might include a bot within a target organization, and an attacker could simply pay to install probe software. Ting-Feng ended with three

points that might help in detecting attacks: behavior profiling, defensive deception, and information sharing.

Engin asked about the role of education in stopping APT. Ting-Feng responded that, given enough time, you can get anybody to open an email. Manuel Egele (UCSB) said that banks would make good targets and wondered why we never hear about them. Ting-Feng said there are a lot of attacks we won't hear about. Jason Brett of UCSB asked whether they had seen instances through RSA of the use of bots. Ting-Feng said she had not, but does not know all that is going on in RSA.

Fabian Monroe wondered what RSA does since they had been attacked, based on her last three points. Ting-Feng said she focuses on log analyses, and RSA collects lots of logs but hasn't done anything with them; she wants to focus on working with these logs. Fabian then asked what the research community might do to help. Ting-Feng said that logs are messy, missing fields, and hard to process. Also, there are many false alarms. Niels Provos wondered whether the logs they had collected would have been sufficient to detect the attacks in the past. She responded that they are not sufficient in themselves, as they don't have total visibility. Niels asked what else they might need. Ting-Feng said the Windows security logs would be useful. Tudor Dumitraş of Symantec Research asked about the role of 0-days in detecting attacks. Ting-Feng said that since they were impossible to detect (by definition), they really weren't of much use in detection.

Let's Parse to Prevent Pwnage

Mike Samuel and Ulfar Erlingsson, Google

Ulfar Erlingsson said that Mike had done most of the work, and since this was a position paper, they hadn't finished it until recently. Ulfar provided an opaque link (via tiny-url.com) and mentioned that you might want to be cautious about following such links. Instead of opening the PDF file pointed to by the link, you might allow Google to process the PDF and present it to you as HTML. In the same way, you could take advantage of a third party to process photos instead of trusting your own system to correctly parse a photo format that comes from an unknown source.

Vulnerabilities can be caused by inconsistencies in processing by different software applications. Ulfar named this data confusion. For example, processing JPEG and PNG files has resulted in vulnerabilities in just about everything since 1998, including a bug in libpng in 2012. He suggested performing lowering, using protocol buffers. As another example, antivirus doesn't work because of data confusion. AV companies cannot write correct parsers for all file

formats. But we can prevent data confusion, and we have technical work appearing in the paper. Ulfar said that rather than have parsers for all content (d) and contexts (c) ($O(c*d)$), you just need $O(c+d)$.

Niels Provos pointed out that there are thousands of protocols, and Ulfar responded that the problem is linear, and that they actually did this work for a Web browser. Then Ulfar showed a demo of a thin client, with all protocol parsing done in a sandbox, and the results displayed in the thin client. Vern Paxson said that work had been done before on thin clients, and Ulfar agreed that the client doesn't have much functionality. Vern wondered if the greed for rich interaction will kill this idea, and Ulfar agreed that this approach by itself is not enough, but parsing using annotated grammars is required. Google already needs to understand protocols and has been doing this. Vern argued that sooner or later they will have to parse a blob of bytes, and Ulfar responded that they already plan on only allowing normalized blobs, such as a single bitmap format.

Ulfar said they have developed one grammar for HTML, CSS, URI, and JSON, plus grammars for languages such as C, Java, and Python. They use lower encoders, sanitization (stripping out scripts), and contextual templating, doing the processing and computation securely and separately from the display. Niels again said that this sounds like a lot of work. Ulfar answered that Mike Samuels has been doing this for years. Fabian Monroe wondered whether they will ever reach a point where everything is sound. Ulfar responded that you don't get to a point where you resolve everything, but you do resolve a lot. Fabian then asked about getting programmers to buy in, and Ulfar said that they want this to happen automatically, within libraries, so that the programmers don't have to do anything.

Emerging Threats

Summarized by Engin Kirda (ek@ccs.neu.edu)

Observations on Emerging Threats

Paul Ferguson, Trend Micro, Inc.

Paul Ferguson from Trend Micro talked about the emerging threats they have been seeing. Trend Micro's Threat Research group is specially tasked with looking forward on the threat landscape. He talked about how the toolkits the attackers are using are getting more sophisticated, and he suggested that mobile threats are probably the next big thing. After the presentation, Tudor Dumitraş from Symantec commented that the problem is that kits make it easier for cybercriminals, because they create work for the defenders. Fabian

Monrose from UNC asked how we know that nation-states are carrying out targeted attacks. Do we assume things? Do we have a pointer for this? Paul's answer was that there is a lot of ad hoc evidence out there and although there are no scientific measurements, there is general awareness and knowledge in the community that such attacks are indeed taking place.

W32.Duqu: The Precursor to the Next Stuxnet

Eric Chien and Liam OMurchu, Symantec; Nicolas Falliere

Nicolas Falliere from Symantec gave a talk on the Duqu analysis they recently conducted in the company. After Nicolas mentioned that Duqu was based on the codebase of Stuxnet, Vern Paxson from UC Berkeley asked what it means that the code is based on Stuxnet. Nicolas answered that there is evidence that the attackers had full access to the source code of Stuxnet. Vern followed up by asking if the Duqu was targeting a specific country. Nicolas said that unlike Stuxnet, Duqu was not targeting a specific nation. One of the questions asked was whether there was authentication in the Duqu command and control mechanism. Nicolas said that the creators of Duqu had not built in any authentication mechanisms.

Botnets and DDoS Threats

Summarized by Manuel Egele (maeg@cs.ucsb.edu)

So You Want to Take Over a Botnet...

David Dittrich, University of Washington

Dave Dittrich discussed what should be taken into consideration when a botnet is to be taken over or down. First, he addressed liability questions that arise when people demand that compromised command and control (C&C) infrastructure should be used to remove malware from infected machines. Dave said that this is a slippery slope and the liabilities are not clear in case that harm is done.

While discussing possible legal approaches, which could include declaring the possession of malicious software illegal, Engin Kirda asked for clarification of the expression "malicious software possession," which David clarified as attack tools and not malware samples per se.

Dave highlighted nomenclature as a big problem. Additionally, reported numbers are often inflated and unverifiable, as the counting methodologies are commonly not disclosed. This includes missing information on timing, or error rates. Dave described the takedown of the Kelihos.B botnet, which turned out not to be exhaustive enough, as the attackers were able to establish the modified Kelhios.C botnet very quickly

after the takedown effort. This led Dave to ponder whether the five most recent takedowns were all successes or just one big failure.

Dave then said that the size estimates of botnets vary hugely, to which Vern Paxson responded that the frequency at which DHCP leases are reissued is dependent on the individual ISPs, and thus it is hard to get the size right. However, Dave pointed out that sometimes bots use a unique ID for each installation and the count can be quite reliable. Engin Kirda wondered why a displayed graph did not indicate novel infections, and Dave replied that the graph shows information gathered after the C&C was sinkholed, and thus no new infections took place.

Dave did mention that relying on the legal process effectively describes the involved ethics as a by-product. That includes the identification of all stakeholders, a detailed analysis of harm and benefits, the likelihood of success, the intentions for the requested action, and automatic external review by the court. This description raised the question of whether the courts are technically competent enough to base their decisions on solid grounds. Dave agreed that it is challenging to put such technical matters into lay terms and that it can be hard to find courts and lawyers that are technically savvy enough.

Dave was then asked what would be done to alleviate the problems that arise from the bad naming choices made in the industry. Apparently, there are efforts on the way in Europe and throughout the industry to come up with a taxonomy to define the necessary terms.

Classification of UDP Traffic for DDoS Detection

Alexandru G. Bardas, Loai Zomlot, Sathya Chandran Sundaramurthy, and Xinming Ou, Kansas State University; S. Raj Rajagopalan, HP Labs; Marc R. Eisenbarth, HP TippingPoint

Alex Bardas explained why one has to consider UDP-based distributed denial-of-service attacks as well as those carried out over TCP. Readily available software packets and free tools such as the Low Orbit Ion Cannon (LOIC) rely on UDP to perform their flooding attacks. Alex then described their detection mechanism that implements a counting approach for packets grouped by source IP address. Although many attacks use the same payload for all packets, this is not a necessity. The fact that UDP is stateless makes detection challenging.

The underlying assumption is that during an attack there are almost solely incoming packets but no outgoing packets that would suggest legitimate communication. The authors call this observation the Proportional Packet Rate

Assumption for legitimate UDP conversations. This inherent two-way communication results mainly from applications implementing their own acknowledgment mechanisms for the unreliable UDP protocol. Thus, the proposed detection system calculates packet ratios of sent and received packets for each sender at the enterprise level. The proposed system was evaluated on synthetic and production networks such as departmental networks, and non-DNS packet captures from industry partners (e.g., ISPs, universities, and financial institutions).

The authors used traffic generated by the LOIC tool to evaluate their approach. The results indicate that there is no silver bullet to detect all UDP-based DDoS attacks. However, by adapting the thresholds for their methods according to each network independently, the authors were able to achieve good results.

The first post-talk question was about understanding how the involved thresholds are chosen. Alex mentioned that the thresholds are best set by observing the network under consideration for a while (i.e., training) and then choosing the corresponding thresholds. He acknowledged that a continuous reevaluation of these thresholds might be necessary, as the optimal values for thresholds can change over time, depending on how the network is used. The second questioner, from the University of British Columbia, tried to analyze the possible impact of spoofed IP addresses, and whether an attacker could disrupt the communication between two peers by spoofing one of the IP addresses. Alex agreed that their system would take down the connection between these two clients in this case but also mentioned during the talk that most ISPs perform ingress filtering and thus IP spoofing (e.g., from dial up connections) is not that easy to accomplish. Furthermore, Alex clarified that their approach relies on observing all traffic entering the network. That is, asymmetric routes where not all used routes are covered pose a problem to the proposed approach.

Tracking DDoS Attacks: Insights into the Business of Disrupting the Web

Armin Büscher, Websense Security Labs; Thorsten Holz, Ruhr University Bochum

Armin Büscher provided some insights into the business behind DDoS attacks. Like the preceding speaker, he mentioned that readily available tools such as the LOIC allow attackers such as botmasters and hacktivists to attack a variety of targets. The authors extracted the motives of these cyber-crooks by evaluating underground forums and identified blackmail, the disruption of competition and adversar-

ies, and political protest as the main areas of motivation. Armin then went on to present a more detailed analysis of the DirtJumper Crimeware Kit attack tool, which is sold and pirated on different underground forums. In their study, the authors analyzed 274 bots which received 1,968 unique target URLs during the observation period, with the majority of the attacked services being HTTP and MySQL database installations. An analysis of the target URLs indicates that they cover a wide variety of businesses such as shopping, gambling, and adult entertainment sites. The authors were thinking about probing the victims of the observed DDoS attacks to evaluate whether the ongoing attacks were successful. However, the probe requests would of course contribute to the attack, so the authors decided against taking this step.

Niels Provos wondered how the authors could evaluate whether an attack was successful. Armin named etrade.com.au, which was attacked by DirtJumper and as a result was taken offline for four days. This information was publicly disclosed and acknowledged by etrade.com.au around Christmas 2011. Armin mentioned that they are aware of at least two other attacks which were also successful, but he was not at liberty to discuss any details about these incidents. Another person asked whether the motivation to attack banks is blackmail. Armin said they had interviewed banks, and blackmail was indeed the motivation for the attacks. An unanswered follow-up question was whether attackers could expect higher success rates for phishing attacks during a DDoS attack, as the legitimate banking Web site would be unavailable to customers.

Fabian Monroe wondered how the landscape is changing, as he had a flashback to HotBots five years ago when the community discussed similar topics. Armin stated that the Internet and how businesses use the Internet have significantly changed in recent years. Today, we have businesses whose business model relies solely on their Web site. Thus, these companies are prime targets for such attacks.

Mobile Security

RGBDroid: A Novel Response-Based Approach to Android Privilege Escalation Attacks

Yeongung Park, Dankook University; ChoongHyun Lee, Massachusetts Institute of Technology; Chanhee Lee and JiHyeog Lim, Dankook University; Sangchul Han and Minkyu Park, Konkuk University; Seong-Je Cho, Dankook University

No reports are available for this session.

Studying Cyber-Criminals

Summarized by Rik Farrow (rik@usenix.org)

Clustering Potential Phishing Websites Using DeepMD5

Jason Britt, Brad Wardman, Dr. Alan Sprague, and Gary Warner,
University of Alabama at Birmingham

Jason Britt gave some background about phishing sites and said that producing a good-looking phishing site takes time. Even though they may look different, they often use the same support files. In their research, they collected MD5s of support files to group phishing sites. Jason then mentioned phishing kits, tools that are sold to help people create phishing sites. Their goal is to go after the bigger players, the people who make or sell phishing kits by looking at the support files found on phishing sites.

They collected data over five months, mostly financial phish sites, but also security and private companies. They checked 265,611 potential sites: 38% were manually confirmed to be phishing sites. If the hash of the top-level page doesn't match a known phish kit page, they collect hashes of support files. Then they use a clustering technique (SLINK) with a minimum similarity threshold of .8 to find sites that are likely to come from the same phishing kit. They did find phishing kit similarities, but they would like to have more assurance that they have really found clusters.

Someone from Paypal asked why there were so many singleton clusters. Jason said that they felt they had enough data, but innocuous files confuse their analysis technique. The same person asked about sites that were unreachable when they did their analyses after the five months was up, and Jason said that they had not looked at that. Would it work better to look at the DOM structure rather than using hashes, which are easy to change? Jason said that their syntactical analysis might solve that problem. Did Jason feel that phishing would be a problem in 10 years? He expects things will change, with more targeted attacks.

Ask WINE: Are We Safer Today? Evaluating Operating System Security through Big Data Analysis

Tudor Dumitraş and Petros Efstathopoulos, Symantec Research Labs

Tudor Dumitraş started with the notion of a worm that could infect all vulnerable machines in 15 minutes. He then pointed out that malware variants are multiplying exponentially, even as OS protection has improved, with DEP and ASLR commonly used in Windows. Tudor presented actual data collected using antivirus telemetry (100 million reports from 6 million hosts over 21 months) and intrusion detec-

tion telemetry (65 million reports from 5 million hosts over 31 months). Vern Paxson asked whether this is the actual number of reports, or was this data sampled. Tudor answered that this information is after filtering, not all the data collected. Vern then asked whether this was background noise or something evil. Tudor said that this was malware detection or network packets that triggered IDS.

Tudor wanted to provide some answers to questions such as: do the new OS protections make it less likely that malware will be created for newer platforms? does it help if the platform software is completely updated? are certain platforms less susceptible to attacks? The answers come from the Worldwide Intelligence Network Environment (WINE), a tool recently developed by Symantec, that is available to researchers in academia. Someone asked how the data is available, and Tudor replied that this is the same data Symantec uses internally, but this is also a filtered sample designed for data-intensive experimentation (SQL, MapReduce, R). WINE can only be accessed on Symantec's premises (in Culver City, CA, and Herdon, VA).

Someone asked about the difference between AV and IDS. Tudor answered that both products are operating on the hosts, and network attacks that trigger the IDS have already bypassed the platforms' firewalls. Tudor then pointed out that these reports indicate that AV or IDS has blocked the attack, so it was a detected, but not successful, attack. The attacks are on various Windows platforms, as that is the most common platform for running Symantec software. Although most viruses are reported variants, they wanted to provide the most specific identification possible, so they present distinct viruses, identified by non-heuristic signatures.

Their first finding is simple: the more hosts running a particular platform, the more distinct virus signatures will be found on that platform. The OS versions with the highest numbers of distinct virus signatures are the ones with the newest security technologies, including DEP/SafeSH, NX bit support, ASLR, User Account Control, etc. He then addressed Mac OS X, where there is a similar trend, but the numbers are three orders of magnitude smaller, so there is not enough data for a statistically significant result. He next looked at how the passage of time affects the number of intrusions or distinct viruses; the number of intrusions seems to increase over time, and the number of viruses does not start from zero even for new releases, but it was not clear that the rate increases over time. Tudor mentioned that finding a virus on a host did not mean that the malware could successfully execute on that platform. In another graph, the introduction of a security technology like DEP, NX bit, and firewall enabled by default

had no detectible effect on the detection of viruses on that platform.

Tudor said that their study has several biases: their data (1) only comes from hosts where the owners actually installed and maintained AV or IDS, (2) is almost entirely from Windows, and (3) does not include servers in their analysis. WINE includes other data sets, such as binary reputation (signatures of all binaries downloaded to hosts around the world), historical URL reputation, samples of spam, and a large collection of malware samples. In conclusion, the production of malware is driven by the number of target platforms deployed, and the introduction of new defenses does not mean a reduction in the number of attacks seen.

Fabian Monroe wondered if there was a correlation between the number of attacks they see and vulnerability disclosures. Tudor said they are doing something very similar, looking at

0-day attacks. One of the things they have observed is that, after disclosure of a vulnerability and 0-day attack, the number of attacks grows a lot.

Threats in Social Networks

Key Challenges in Defending Against Malicious Socialbots

Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, and Matei Ripeanu, University of British Columbia

Adapting Social Spam Infrastructure for Political Censorship

Kurt Thomas and Chris Grier, University of California, Berkeley; Vern Paxson, University of California, Berkeley, and International Computer Science Institute

No reports are available for this session.

SAVE THE DATE!



LISA'12

26th Large Installation
System Administration Conference

sponsored by USENIX
in cooperation with LOPSA

DECEMBER 9–14, 2012 | SAN DIEGO, CA

**Come to LISA '12 for training and face time
with experts in the sysadmin community.**

LISA '12 will feature:

6 days of training on topics including:

- Virtualization
- Security
- Configuration Management
- Cloud


Plus a 3-day Technical Program:

- Invited Talks
- Paper Presentations
- Practice and Experience Reports
- Guru Is In Sessions
- Vendor Exhibition
- Workshops
- Posters and WIPs

**The LISA '12 keynote address will be delivered
by Vint Cerf, Vice President and Chief Internet Evangelist, Google.**

www.usenix.org/lisa12

STAY CONNECTED:

 FACEBOOK.COM/USENIXASSOCIATION

 TWITTER.COM/LISACONFERENCE



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



usenix

10th USENIX Symposium
on Operating Systems Design
and Implementation

October 8-10, 2012 • Hollywood, CA

The 10th OSDI seeks to present innovative, exciting research in computer systems. OSDI 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

- **HotDep '12**: Eighth Workshop on Hot Topics in System Dependability, **October 7**
- **HotPower '12**: 2012 Workshop on Power Aware Computing and Systems, **October 7**
- **MAD '12**: 2012 Workshop on Managing Systems Automatically and Dynamically, **October 7**

STAY CONNECTED ON  

www.facebook.com/usenixassociation

www.twitter.com/usenix #osdi12

www.usenix.org/osdi12

Register by September 17 for the greatest savings!