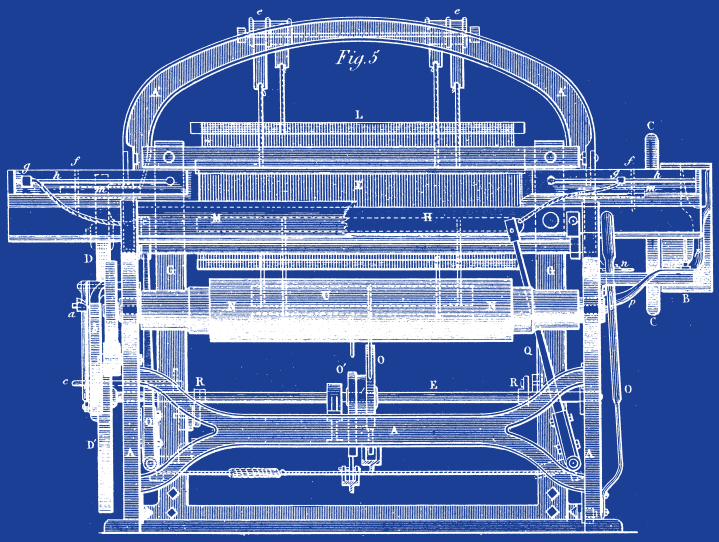
AUGUST 2010 VOLUME 35 NUMBER 4

;login:

THE USENIX MAGAZINE



USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events

9th USENIX Symposium on Operating Systems Design and Implementation (OSDI '10)

Sponsored by USENIX in cooperation with ACM SIGOPS

OCTOBER 4–6, 2010, VANCOUVER, BC, CANADA
http://www.usenix.org/osdi10

Workshops co-located with OSDI '10:

Workshop on Supporting Diversity in Systems Research (Diversity '10)

OCTOBER 2–3, 2010
http://www.usenix.org/diversity10

Workshop on Managing Systems via Log Analysis and Machine Learning Techniques (SLAML '10)

OCTOBER 2–3, 2010
http://www.usenix.org/slaml10

Sixth Workshop on Hot Topics in System Dependability (HotDep '10)

OCTOBER 3, 2010
http://www.usenix.org/hotdep10

2010 Workshop on Power Aware Computing and Systems (HotPower '10)

OCTOBER 3, 2010
http://www.usenix.org/hotpower10

2010 Workshop on the Economics of Networks, Systems, and Computation (NetEcon '10)

Sponsored by USENIX in cooperation with ACM SIGecom

OCTOBER 3, 2010
http://www.usenix.org/netecon10

5th International Workshop on Systems Software Verification (SSV '10)

OCTOBER 6–7, 2010
http://www.usenix.org/ssv10

24th Large Installation System Adinistration Conference (LISA '10)

Sponsored by USENIX in cooperation with LOPSA and SNIA

NOVEMBER 7–12, 2010, SAN JOSE, CA, USA
http://www.usenix.org/lisa10

9th USENIX Conference on File and Storage Technologies (FAST '11)

Sponsored by USENIX in cooperation with ACM SIGOPS

FEBRUARY 15–18, 2011, SAN JOSE, CA, USA
http://www.usenix.org/fast11
Submissions due: September 2, 2010

8th USENIX Symposium on Networked Systems Design and Implementation (NSDI '11)

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

MARCH 30–APRIL 1, 2011, BOSTON, MA , USA
http://www.usenix.org/nsdi11
Submissions due: September 16, 2010

13th Workshop on Hot Topics in Operating Systems (HotOS XIII)

Sponsored by USENIX in cooperation with the IEEE Technical Committee on Operating Systems (TCOS)

MAY 8–10, 2011, NAPA, CA, USA
http://www.usenix.org/hotos11
Submissions due: January 15, 2011

For a complete list of all USENIX & USENIX co-sponsored events, see http://www.usenix.org/events.

contents

;login:

VOL. 35, #4, AUGUST 2010

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
Steve Gilmartin
proofshop@usenix.org

PRODUCTION
Casey Henderson
Jane-Ellen Long
Jennifer Peterson

TYPESETTER
Star Type
startype@comcast.net

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

http://www.usenix.org
http://www.sage.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.

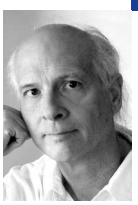
$90 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $125 per year.

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.

RIK FARROW

musings



Rik is the Editor of *;login:*.

rik@usenix.org

ANOTHER DAY, ANOTHER EXPLOIT. Today I read that some researchers plan on demoing a rootkit for Android at DefCon 2010. In May, UW and UCSD researchers announced remote control of a car via the diagnostic port [1] and a wireless link. They could disable the brakes totally, apply the brakes to just one wheel (or more), stall the engine, and control the display panel. And just wait until you learn what can be done with your new smartphone.

I got interested in computer security because I found it both intriguing and challenging. Understanding security involves knowledge of programming, system administration, OS design and implementation, and networking. Few CS disciplines cross so many boundaries. I've done well in the field, but still feel like I am groping in the dark.

Visualization

Speaking of darkness, I decided to visualize what government-sponsored attackers might be using in the near future. It would have been really nice to have a console that was not only used for attacking (or, more politely, penetration testing), like Metasploit, but something that displays the status of the systems and networks that may be involved in the attack. Of course, I also imagine that the operator can call up lists of potential attacks and chose to execute an attack with a few keystrokes (real hackers keep their hands on the keyboard—mouses are for sissies). On that note, here goes . . .

Maxim glanced at his smartphone, checked the time, and stubbed out his cigarette. He really hated himself for smoking, but did it for the sharpened focus it seemed to bring. He looked around the stuffy, windowless room, with its piles of books, magazines, and stacks of paper, then turned out the dim LED lamp over his desk. Finally, he lifted the helmet from its rest and settled it into position.

Maxim sighed deeply. As usual, he felt enthralled by the display. He appeared to be floating in the depths of space, but instead of stars, collections of brightly colored lights surrounded him. With a couple of keystrokes, he could zoom in on an area and view the tags that provided more detail. Or he could add in overlays that showed the amount of network traffic between any of the targets in his display.

Soft pops, crackles, and chimes came from all directions. The UI people had decided that enhanc-

ing the display with sound was a good idea, and Maxim agreed. Humans retain ancient instincts from their hunter-gatherer days, when someone who ignored a cracking branch might miss finding dinner, or quickly become dinner. As new sources of information appeared, they chimed softly, then crackled as more information about the source was added.

What Maxim really hated was the thumper. If the back-end AI decided that the attacker had been detected, it not only produced a tremendous *crump* sound, it also thumped the seat hard enough to jolt his whole body. Full surround experience. Fortunately, he hadn't been detected often since his training days.

Maxim had discovered many years ago that he worked best when he could visualize what he was working with. In junior high, he had scribbled out graphical representations for the systems he was attacking, and how they appeared to be related. His drawings were crude, and trying to decipher scrawled IP addresses and port numbers later always proved troublesome. Plus, trying to recall what he had written to represent passwords was truly frustrating.

By comparison, this new visualization system was a dream come true. Just selecting the object representing a target brought up all the known details, as well as some that were guesses, and allowed him to scroll through lists of actions he could use to probe or attack.

Maxim watched as the display continued to evolve, with new objects glowing and chiming into life in front of him, while others faded away as they moved behind him in the virtual display. He still had a minute or so before the action began, and he allowed himself to daydream. How would things have looked in this display when he first started hacking back in the late '90s?

Instead of objects and their attributes, Maxim conjured up an image made of dots of light in the darkness. Each dot represented an open port, and the text associated with the dot gave the port number and, for a well-known port, perhaps the name of the protocol. Port scans produced the raw data, and seeing it as dots of light made quick assessments of targets easy. A totally black display meant no ports open, while one full of glowing dots suggested a wide-open, poorly configured system just wanting to be owned. Sometimes such systems would already have been exploited many times before he found them. Occasionally, he would be the first.

Times changed, and people became more aware of security. Firewalls became common, so instead of scanning target systems, all you could scan was the firewall. And scanning firewalls was akin to throwing rocks at an armed guard sitting inside a bunker—unless the firewall was poorly configured. Sometimes a port scan of a firewall would produce a display every bit as useful as a wide-open server. Other times, the firewall would stop responding to his scan, leaving nothing but darkness in its place.

Where once the targets had mostly been various UNIX and Linux servers, Windows machines started appearing. Recognizing them from the open ports was as easy as recognizing an old Ford by its square tail lights. UNIX systems, long the victim of attacks, were still common but were much better secured these days. No more easy shell accounts, but Windows systems were now sitting on faster networks, making them useful as relays and exploit stashes.

Maxim mentally tagged each machine he had exploited with a different color: red for firewalls, blue for relays, and orange for code stashes. Over time, the display would change as exploited machines were taken down,

patched, or wiped clean. Often he could re-exploit a system, but he would always wait a few weeks before doing so.

Maxim labeled Web servers as green, because, for a while, they had meant money. Web servers always had open ports because serving data was their job. A firewall could still get in the way, but few people bothered to configure them correctly, something that would have made downloading his tools and attacks more difficult. With a Web server as a relay, he could usually get inside an organization in a snap.

Then the Web servers themselves became sources of income. Maxim would pop Web servers, add a single line to many HTML files that would direct any browser that downloaded them to a drive-by download site, and collect money for each system exploited. It was a great time, like being a highwayman, or even a troll, the storybook kind that lived under bridges.

He still took side jobs, and these were a little more demanding. In most cases, someone wanted to get something from within a particular organization. If his clients provided him with lots of specifics, such as the names and email addresses of a target and his associates, he could use a classic social engineering attack. The target would get an email and by opening the email, silently exploit his Windows desktop. Visualizing this, Maxim imagined a long tunnel stretching from his desktop, through any firewalls, and right into the target organization. Once inside, he would slowly build a picture of the internal network by cautious probing (don't want to set off an IDS by being too eager) and reading emails. Just the headers alone would sometimes be enough to steer him in the right direction.

Maxim popped back into the present moment. His fingers floated ready above his keyboard, waiting to launch the next phase of the attack. His display had become almost muted, as the object he was focused upon moved into a secure corridor, one with little network traffic. After a brief pause, the object moved forward and the screen lit up again. Bingo! He was inside. Maxim's hands flashed as he typed out macros, launching several attacks in swift succession. It was important to establish a beachhead, by setting up relays, before the subject turned off his smartphone. Until then, the smartphone acted as a relay between a tunnel to his attacking system and the wireless network the smartphone had joined. He didn't know, nor did he need to know, the encryption key for the wireless, since the smartphone had handled that for him. And with data rates in metro areas approaching five megabits per second, any downloads he needed to do would go fast.

Maxim perceived the richness of the display grow as the passive sniffing tool he had launched did its work. A print server appeared, and Maxim played a few keys that started a sure-fire exploit against the server. Print servers were often trusted in office networks, and were a great place to "share" malware.

It really wasn't hard to find services on today's networks, what with all the announcements. Routers, printers, file servers, authentication servers, all either routinely broadcast or responded to broadcast queries for attention.

Maxim already knew who would likely hold the key to the data file his employers wanted. All he needed to do was discover the IP address assigned to that user's desktop. There! A plaintext email header with the name in the To: header line gave away the address. Maxim launched the exploit that would do the deed. By sending out a fake ARP reply, the smartphone would relay data between the victim and the default router. Once the victim visited a Web site, the relay would insert a bit of JavaScript code that, once executed, would add a relay to the victim's Web browser. Maxim loved IE6, as IE8 would have made this part of the attack much more difficult.

The sound of a gong, like one from an Asian monastery, announced success. Maxim quickly shut down the ARP spoofing program, as it would be easily detected by IDS (and perhaps already had been). He then wiped his tools from the smartphone and closed the tunnel. Instantly the display went dark as the connection closed, accompanied by the pops that signified systems disappearing.

Maxim took a deep breath, coughed, and removed the helmet. He glanced at his phone as he lit a cigarette: he had been inside for just five minutes.

Fantasy to Reality

Maxim's visualization tool is currently, AFAIK, total fantasy. The attacks I described are not fantasy. The past attacks are part of history. Using a smartphone as a relay is the only new attack presented, and it is technically possible today. On the best secured phones, such as the maniacally controlled iPhone, this attack would be difficult and would rely on jailbreaking. But on phones where users can install any app they choose and grant the app all the permissions it asks for, this attack becomes trivial. The user might notice how sluggish his phone had become but would be unlikely to notice anything else. The entire display of a smartphone is driven by software and can be convinced to display anything at all, easily fooling the user into thinking things are perfectly normal. The display could even fake a complete shutdown, as the only way you can really be sure a smartphone is off is to remove the battery—something you can't do routinely with an iPhone.

I disliked using computers when they were so complex that no single person could understand the entire system. The mainframes I used in the '70s were like that, similar to mainframes that are still in use today. These mainframes required many bookshelves to hold the documentation for system programmers, those allowed to hook into the underlying OS.

When I started using PCs, understanding the entire system was much easier. After all, a system with 64kb of RAM and only floppy disks just couldn't hold very much, and having a complete understanding of how the system worked was possible. Early UNIX systems were the same way, with a total of three binders containing *all* of the documentation. Today's *nix systems are no longer so simple—a kernel alone uses tens of thousands of times the RAM my entire first PC had.

Layered on top of any modern desktop or server, we find libraries. The libraries make life much easier for programmers, as they provide consistent and easy-to-use interfaces with the rest of the system. They also add many layers between the hundreds (for *nix) or thousands (for Windows) of system calls. Each layer of abstraction adds complexity even as it reduces the programmer's burden, while opening up more opportunities for exploitation.

My friends in the business of AV and application auditing have been telling me how much better Windows security has become. They swear Windows 7 has much better security than either Linux or Mac OS X because of many great new features. Yet there are still exploits for Windows 7. The same people tell me that this is because Windows versions control 95% of desktops, and that is certainly true to a point. But as long as there are many layers of software, and users are allowed to install any software they like or run browsers that will do this for them, Windows will not be secure.

And neither will other systems.

The day of the user-controlled system may be coming to an end. For security, this will be a good thing. For freedom, including the freedom to tinker,

it might be terrible. Yet Apple has already done this with its iP* family, and I wonder if Microsoft Windows and Google Android will be far behind.

The Lineup

This issue begins with an article about logging, not in the sense of mining the land for timber, but mining your logs for the interesting parts. Ron Dilley has been obsessed with logs for many years. He tells us how this obsession has lead him to create a set of useful tools and techniques for monitoring systems.

Steve Checkoway and friends have written about some research they published at LEET '10 (reports in this issue). Checkoway warns us that even text isn't safe, and demonstrated this by creating a virus that infects systems via TeX files.

Bo Li et al. explain the Symbian OS security model. Li begins with some background on the original design, then covers the current security features used in the most popular smartphone OS in the world today (based on number of handsets sold). I find that the trade-offs between security and the desire to have lots of apps for your OS tends to favor apps over security—but I suggest you find out for yourself.

Carlos Maltzahn and associates have provided us with an overview of Ceph, a distributed file system in the works for over five years, and they explain how to integrate Ceph with Hadoop. In the April 2010 issue of *;login:*, Konstantin Shvachko discussed how the single namespace server design of HDFS creates very real limits in scaling and performance. Ceph bypasses these limitations by having many metadata servers. The client that allows mounting of Ceph was added to the Linux kernel in May, making Ceph even more accessible. The instructions in this article for setting up your own Ceph distributed file system are clear and simple to follow.

Michael Piatek and Arvind Krishnamurthy explain why the tit-for-tat reward system used in peer-to-peer file-sharing systems like BitTorrent can't work with live video streaming. In a short and very clear article based on their paper for NSDI, they outline a different strategy, one that has already been implemented in a commercial video streaming product.

David Blank-Edelman expands on the theme of his April column, making things up, by covering randomness. If you have been awake and interested in computer security, you will appreciate just how important good sources of randomness can be. David takes us on a tour of Perl modules that go beyond the two system calls POSIX operating systems provide.

Peter Galvin stares into his crystal ball, then chucks it. Peter takes a hard look at the future of various Sun products under Oracle control, basing his column on the few facts that are known and a bit of deduction. I recommend his column for anyone who is interested in the future of Sun.

Dave Josephsen continues where he left off last month in his discussion of Argus. In this column Dave expounds on the basics of collecting packet summary data using Argus, explains how to combine, collate, and reduce Argus data, and demos some commands for extracting interesting stuff from this data.

Robert Ferrell provides a hard look at disaster planning. Possibly inspired by the opening of the hurricane season, with all the destruction, death, and misery that implies, Robert suggests that the best disasters are the ones you plan yourself.

We have reports from the NSDI conference and workshops in this issue. All of the workshops were covered. We also have a report from BSDCan 2010, about an invited talk that considers the interesting side effects you can expect from implementing IPv6.

I just burned a new Linux live-CD for a friend, one that he uses only for online banking. Windows rootkits have become less common: exploiting the browser is so much easier, why bother installing a rootkit? Not that rootkit-like malware does not still exist, and is in fact pretty common in bot clients. By booting from a CD, my friend has a guaranteed secure path from his keyboard, through the browser and its many layers of libraries, to his banking site (which may or may not be secure).

If I had my way, all systems would provide as much security as booting from a live-CD can. I think it is at least possible that we will see such systems soon enough, as Android strives to produce this level of security, and the iPhone already comes close. But that day is still in the future. Be very careful, and perhaps you can avoid the messiness of being exploited.

REFERENCE

[1] Karl Koscher et al., "Experimental Security Analysis of a Modern Automobile," *2010 IEEE Symposium on Security and Privacy*: http://www.autosec.org/pubs/cars-oakland2010.pdf.

RON DILLEY

making sense of logs

Ron Dilley has spent the majority of the past two decades in the fields of programming, technical support, administration, and security engineering. His involvement in information systems and information security is informed by his vast experience, including several open source projects and work on the 2005 DARPA Grand Challenge. After several years as a UNIX administrator, Dilley moved into information systems security. He currently leads an information security team at a Fortune 500 company.

ron.dilley@uberadmin.com

EFFECTIVE LOG ANALYSIS DOES NOT need to be complex or expensive. Nor do you need to have significant prior knowledge of the data to find anomalies and clear indications of infections. This may sound like the preamble on some glossy product sheet, but don't be fooled. I am not trying to sell you anything. I am trying to encourage you to read on about the log-related tools, techniques, and successes I have had over the last few years.

> To secure ourselves against defeat lies in our own hands, but the opportunity of defeating the enemy is provided by the enemy himself.
>
> —*Sun Tzu, 544–496 BC*

A Short History

I have nurtured a love-hate relationship with logs for almost 20 years. It would be fair to say that my feelings for logs border on obsessive. Instead of spending the following paragraphs on a "logging is good" rant, I present a history of the past few years of my logging obsession and some of the ideas, tools, and techniques that I have worked on.

I was a closet logger for many years. It started back when I was a UNIX administrator running a large call center for a cellular carrier in California. I always subscribed to the idea that it was better to have and not need than to need and not have. I found myself in many situations where solving a problem started and ended with reviewing the copious logs diligently collected from across my environments. I started with awk, moved up to perl, and fiddled around with open source tools like swatch [1] and xlogmaster [2]. As my environments got larger and my logging fervor grew, the volume of log data became unwieldy. It also became harder and harder to make sense of or use all the log data I was storing.

My love of logs did not abate when I switched from system administration to information security. My frustration, on the other hand, grew and grew and grew. I became convinced that there were interesting "things" locked in the massive volume of log data that I was compulsively writing to disk every day. Unfortunately, I could not find any way to set these "things" free. I searched year after year for some way, some tool or script. With every new

product that claimed to solve the logging problem, my heart would leap and subsequently break after grilling the sales people or banging on the product showed that it was not doing anything that my scripts could not already do.

A funny thing happened to me on the way to a security conference that changed everything. It would be more accurate to say that I met another obsessed logger at a conference but for the purpose of this story, it started on the way to the conference. I was thumbing through the schedule for USENIX Security on the plane, looking to see what talks I would attend and noticed that a guy by the name of Marcus Ranum [3] was scheduled to talk about logging. It turned out that he was actually scheduled to rant about logging, but I am getting ahead of myself. It was time well spent. Not because he presented any tools or techniques that blew me away or solved all of my problems, but because he too believed that there were "things" locked away in the logs. I sat in the front row and made a point of introducing myself and plying him with booze and Thai food after class.

We have been collaborating on ways to get at those sneaky little "things" in the logs ever since.

Good Analysis Starts with Reading Your Logs

> Log analysis is a lexical problem.
>
> —*Marcus Ranum*

We focused our work on log datasets stored as ASCII text files. These files were generated using syslog. The best and worst thing about logs produced by syslog [4] is that the format is very open. Other than the PRIORITY and HEADER sections, which have some formatting requirements, the MESSAGE should only contain printable characters, and the whole syslog line should not exceed 1024 bytes. The absence of formatting requirements makes for an interesting parsing problem. For years I used regular expressions to chop up log data. Marcus had a different idea. He built a parser using lex and yacc [5] and spent a few days with Abe Singer at the San Diego Supercomputer Center crunching 10 years' worth of log data. The parser converted raw logs into printf [6] style templates that represented each log line. For example, a simple log line like "Oct 11 22:14:15 mymachine su: 'su root' failed for lonvick on /dev/pts/8" was converted to "%s %d %d:%d:%d %s %s %s %s %s %s %s %s". The reason for running this tool against 10 years of data was to get an idea of how much variability there was in the general format of syslog-based log data from a diverse set of UNIX computers. Even with this simple tokenizing strategy, Marcus found about 50,000 different formats across 10 years of data. The graph in Figure 1 (next page) plots the unique templates discovered and clearly shows the increases in new log structures over time. This early version of the parser turned out to be an effective Linux major-release detector, as these spikes aligned well with deployments of new OS versions.

I used the same tokenizing strategy against two years of logs from a Fortune 500 biotechnology company and found less than 8,000 different line formats. This work encouraged me to build a non-regex parser [7] that could extract interesting chunks of data from log data without prior knowledge of the overall log format. In doing so, I came across a couple of interesting and helpful side effects of this parsing methodology. My goal was to make the parser smarter about types of data that I was sure would show up in the logs. To start, this included IP addresses and timestamps.

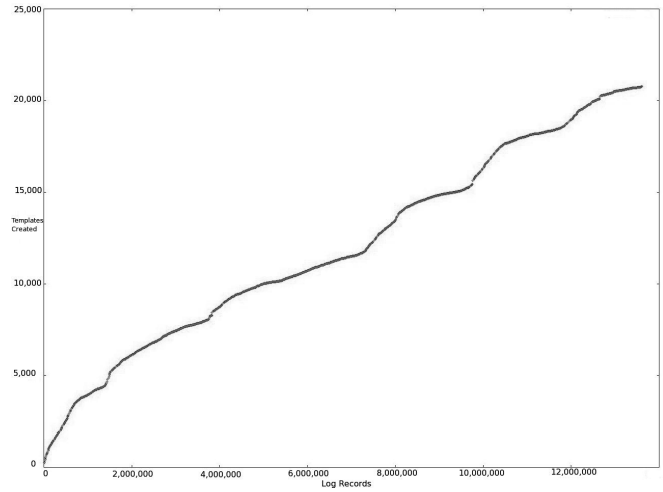


FIGURE 1: PLOT OF UNIQUE TEMPLATES

The utility read logs and converted them into a printf-like template with special tokens for time and date strings as well as IP addresses. White spaces and special printable characters were stored as part of the template. IP addresses and time/date strings were converted to strings. Each non-numeric value associated with a token was stored in a binary tree unique to the template and the token. For example,

```
Feb 5 17:32:18 10.0.0.99 Use the BFG!
```

would be converted into the template "%D %IP %s %s %s!". With the time/date stamp and IP address converted to strings, the five strings would be stored in separate binary trees all associated with the template that the log line matched. At first, I did not realize what I had just built or the interesting and unexpected side effect of organizing the parsed log data in this way. The tool could represent any log line by its template and a unique identifier for each variable. This simple approach was easy to implement, but was inefficient in storing some fields. I experimented with using various compression techniques to save space. A subsequent revision to my code converted recognizable strings such as dates and IP addresses to numbers and encoded them as the numeric value. Treating the fields this way allowed me to see log types I had not seen before, even if some fields had inconsistently variable data, such as the month and day string in a date field. A further optimization was to treat short fields as part of the template so that I did not have to use more space to encode the field than the value consumed in the log. I kept all of the variables in RAM, so I settled for organizing them in binary trees ordered using Huffman encoding [8]. This made it possible to represent variables as a series of bits. Consider this: the example log message above is 39 characters long, and if we assign a two-byte numeric ID to the template (0x01), store the timestamp and IP address as four-byte numbers, and assume that "Use", "the", and "BFG" are the first items in each binary tree, we have stored 39 bytes of data in 10 bytes and the bits required to traverse the Huffman encoded binary tree. My non-regex parser became a tool called logstore [9]. I added some improvements such as assuming that binary trees with only one item could be summarized and moved to the template. Logstore produced 40-to-1 compression of my log data and was much faster than other compression tools that were CPU intensive. The second unexpected benefit of this method came from keeping track of all of the variable strings associated with each log template. By recoding the log line number where each variable occurs, the logstore parsed data format provided a high-speed keyword search capability. I almost regretted encoding some fields as numbers to save space, but good compression was a higher priority than

faster searching at the time. In hindsight, having an option to encode to save space or speed up searches would have been useful.

It was amazing how much fun I could have while taking a stab at regex-less parsing.

Quickly parsing data without significant prior knowledge was nice. We found that non-regex parsing was able to detect new forms of log data that were ignored by regex-based parsers. Unless great care was taken when building the regex parsers, new or edge-case logs were ignored. Using the abstraction of the log templates allowed me to see when a new type of log started showing up by sending the templates into a tool that Marcus wrote called *never before seen* (NBS) [10]. Additionally, I was able to do some simple frequency analysis of the types of logs that were seen without fighting with the complexity of variables.

Alas, it was not enough to satiate my hunger, my need to find those devious "things" that continued to elude detection. A couple of years ago, in response to questions about the efficacy of data-loss prevention systems for my environment, I sat down with Marcus to design a tool that could give me an idea of what we could see on the network beyond simple pattern-matching signatures. Perhaps then I could find those sly little "things" hidden in my log data.

Someone to Watch over Me

> The number of times an uninteresting thing happens—is an interesting thing.
>
> —*Marcus Ranum*

What I needed was a configurable log filter that did not rely on known bad patterns. What came out of our working sessions was a solution with three distinct components. The first components were data collectors that stored network traffic, syslog data, and infrastructure service logs such as DHCP and DNS. The second was a set of parsers that converted the various data sources on the collectors into pseudo-XML format. The last was the tool where all the real magic happened, called overwatch [11]. From a high level, the system works like this:

1. Data collectors generate data which is written to syslog.
2. Syslog data is forwarded to a central repository.
3. Periodically, a batch job kicks off a log parser to convert the syslog data to pXML.
4. The batch job then submits the pXML data to overwatch via a domain socket.
5. Overwatch applies rules and stores scoring information in memory matrixes.
6. Overwatch sends warnings and alerts based on configurable thresholds.

DATA COLLECTORS

Our collectors come in all shapes and sizes. Regardless of how they work, they follow a similar pattern of gathering data and then converting it into text-based logs that are easily parsed. I have built several task-specific programs that extract data directly from the network. I mention several in this article. I have also built scripts, both simple and complex, to convert data from applications such as tcpdump and snort into text-based logs. For consistency and simplicity, I like setting up my collectors to send their data via syslog.

LOG PARSERS

Most of the discussion up to this point has revolved around the evolution of techniques used in our log parsers. Each parser converts the text-based data recorded by the collectors into a pseudo-XML format that is ingestible by overwatch. The name of the game with the parsers is speed: overwatch is a batch-processing environment, so the log parsers determine how fast the data is loaded and processed.

OVERWATCH

Overwatch is a general-purpose filter and pattern detector for arbitrary text-based data. Its simple architecture and configuration belies the power and effectiveness of the tool. The tool maintains n-dimensional matrixes of weighted values. The dimensions, time scales, and weightings are configurable and there is no hard-coded limit to the number of active matrixes. To give you a taste for how overwatch works, I will work through building a configuration to monitor, filter, and alert on DNS logs. Listing 1 shows a record taken from a DNS sniffer [12] that I wrote for use with overwatch:

```
<rec>
<time>Mar 11 18:36:51</time>
<snort dad>123</snort dad>
<srcMac>0:15:c7:c5:22:40</srcMac>
<srcIp>10.131.239.206</srcIp>
<srcPort>32768</srcPort>
<dstMac>0:3:47:de:34:f3</dstMac>
<dstIp>192.41.162.30</dstIp>
<dstPort>53</dstPort>
<dnsId>63706</dnsId>
<qCount>1</qCount>
<qStr0>crl.comodoca.com</qStr0>
<qType>1</qType>
<qClass>1</qClass>
<aCount>0</aCount>
<authCount>0</authCount>
<rCount>1</rCount>
</rec>
```

LISTING 1: A RECORD IN PXML CONVERTED FROM A DNS LOG ENTRY

The pseudo-XML (pXML) begins and ends each log record with <rec> and </rec>. The time attribute is not mandatory; overwatch will use the current time for the record if the field is not present. The <snort dad> field is a unique identifier for the DNS sniffer that sent the record. The other fields are self-explanatory, although we will be using the <aCount> field in another example to discriminate between DNS queries and answers. Next we will build our matrix definition:

```
matrix dnsA
  options {
    alert channel = "/tmp"
    alert recipient =
      "dnsA.alerts"
    warn at 1000
    alert at 5000
    path "/tmp/dnsA"
    rotate hourly
  keyfields {
```

```
        <dnsId>
        <reqIp>
        <aStr0>
    }
}
```

LISTING 2: A MATRIX DEFINITION USED IN OVERWATCH; THE KEYFIELDS DEFINE A THREE-DIMENSIONAL MATRIX.

All matrix definitions begin with the "matrix" keyword. The options section allows you to specify the paths to the data files and high- and low-water marks as well as the time scale for each matrix. The example above puts the data files in /tmp and sets the high-water mark to 5000, the low-water mark to 1000, and the time scale to hourly. This means that data will be placed in matrixes in one-hour increments and that when a value at a given *n*-dimensional position in a given one-hour period exceeds 1000 a warning record will be written, and at 5000 an alert record will be written. The keyfields keyword is where the dimensions are defined. dnsId is the DNS ID associated with the logged query or answer. reqIp is the IP address of the system that sent the query. aStr0 is the first answer string. Listing 2 defines a three-dimensional matrix of dnsId, reqIp, and aStr0.

Now that we have defined our DNS matrix, let's tell overwatch what to do when it reads a DNS log record.

```
records matching {
  <aStr0> startswith "192.168."
} insert into dnsA {
  bump +500
}
```

LISTING 3: AN OVERWATCH INSERTION RULE THAT ADDS 500 TO A MATRIX LOCATION WHEN ASTR0 BEGINS WITH 192.168

Insertion rules begin with the records keyword and have two parts. The first is the match rule and the second is the insert rule. If the match rule is true, the insert rule fires. You can think of the insert rule as a small program that runs against the score values in the matrix. The example above defines a match rule for the first answer string, <aStr0>. If the <aStr0> field starts with the string "192.168.", then the insert rule fires. This is interesting because the DNS logs are being collected from the edge of the network. In general, an external DNS server should not be returning an RFC 1918 address in response to a query for an external hostname. The insert rule adds 500 to the matrix at the position associated with dnsId, reqIp, and aStr0. This rule helped me find some malware that was using DNS answers to issue commands to infected hosts.

The example above is very simple, but don't be fooled. The power of overwatch is in its configuration language. It is limited only by the imagination of the user. Consider the following matrix definition and insertion rule for DNS:

```
matrix dnsConficker
  options {
    alert channel = "/tmp"
    alert recipient =
      "dnsConficker.alerts"
    warn at 1000
    alert at 5000
    path "/tmp/dnsConficker"
    rotate hourly
  keyfields {
```

```
        <reqIp>
    }
}

records matching {
    <aCount> = 0
} insert into dnsConficker {
    bump +1
}

records matching {
    <aCount> greaterthan "0"
} insert into dnsConficker {
    bump -1
}
```

LISTING 4: THIS INSERTION RULE DETECTS CONFICKER DNS FLOODING BY WATCHING FOR MISMATCHES BETWEEN THE NUMBER OF DNS REQUESTS AND RESPONSES.

The matrix rule in Listing 3 is very similar to the example in Listing 2, with the exception that keyfields only has one dimension, <reqIp>. The insertion rules are a bit different. There is no hard-coded limit to the number of insertion rules you can have for a given matrix. For this example I have created two rules. The first adds 1 to the score for a given requesting IP address each time the IP address sends a DNS query. The second subtracts 1 from the score each time an answer to a query is received. This turned out to be a simple way to detect Conficker [13] DNS flooding, as the majority of normal DNS traffic gets at least as many answers as requests. Malware that sends large volumes of bogus DNS traffic looks very different.

The match rules support the usual set of Boolean operators along with string matching and regular expressions. It is also possible to define external files with lists of values to test against. The lists turned out to be helpful in applying adjustments to the scores based on known good or bad criteria such as blacklisted (bogon) networks and domains. A special-match rule calling NBS (never before seen) allows for special insert rules when some value is seen for the first time.

The insert rules support addition, subtraction, and multiplication of fixed values or a value from the log records. I have used this several times. One rule that returned unexpectedly useful information loaded firewall log data into overwatch. It added the bytes out to the score for each destination IP address and port and subtracted the bytes in. This provided a list of the destination/port pairs where internal systems were sending more data outside the network than they were receiving back in. Other than a short list of services like email and VPN connections, most Internet services send more into your network than out of it. Here is output from the rule using the dumpdb overwatch command.

```
% dumpdb –d /tmp/fwByteCounts.2010.03.25.07 dump-all | sort –n –r
13080772    post.craigslist.org|443
619852      www.plusone.com|443
574766      63.240.253.71|443
86940       64.23.32.13|443
85037       www.invitrogen.com|443
79966       miggins.aqhostdns.com|2082
73957       198.140.180.213|443
62292       147.21.176.18|443
```

```
61512       159.53.64.173|443
57494       63.240.110.201|443
```

The dumpdb command was built to aid in tuning the high- and low-water marks for sending out warnings and alerts. As I tuned rules, I found it useful as a stand-alone tool for generating actionable reports about data collected in the overwatch matrixes. The score is the number of bytes out minus the number of bytes in over a one-hour period. This simple rule can detect command and control traffic, unauthorized VPN solutions, and other IP leakage points. Most systems that show up on this list warrant special attention from a security analyst.

A Huge Leap Forward in Log Analysis

It is not my intention to give you a complete dissertation on overwatch and all of its capabilities much though I would love to do so. The point of this high-level teaser is to show why I am no longer frustrated about my logs. I now have a tool that is more than capable of finding those shifty "things," or, as Marcus calls them, "needles in the haystack." In closing, I would like to offer up what I see as the next logical step in the use of overwatch. We call it distrust engineering and it goes something like this.

Systems that interact with "bad" systems are less trustworthy than systems that don't. "Bad" systems exhibit repeating and detectable properties. A system that exhibits one of these properties is less trustworthy than a system that does not exhibit any. A system that interacts with an untrustworthy system is itself less trustworthy. Systems become trustworthy over time.

A negative score shows that a system is untrustworthy and a positive score that it is trustworthy. If we use overwatch to keep track of our trust scores then all we have to do is define what log records represent untrustworthy properties. Once defined, we just need to build parsers that will send the log records into overwatch and we will be able to maintain a near-real-time trust map of all systems in our environment. This ever-changing map will show malware incursions and blooms as well as their retreat as we respond to the incursions. These rules don't need to be complex, and the score adjustments can vary depending on the weight of the event or property. Here are a few attributes that would decrease a system's trust score. I leave the "how-to" for gathering data about each of these attributes as an exercise for the reader.

- Systems in un-trusted countries
- Systems with broken or missing DNS records
- Systems listed in DDNS services
- Systems on malware/spyware black lists
- Systems sending spam or email
- Systems sending packets that are blocked by your firewall
- Systems with new DNS records
- Systems running insecure operating systems
- Systems running unsafe browsers
- Systems with vulnerabilities
- Systems you have never seen before
- Systems flagged by your IDS systems
- Systems that have previously been infected with malware

The above is not meant to be an exhaustive list but merely a primer to get you thinking in the hopes that our email will be flooded with suggestions, rules, and parsers for overwatch.

Conclusion

I have been using overwatch for several years now and my only frustration is that I can't spend more time building and testing new rules and adding additional log sources to the system. My goal was to find a reasonable way to find interesting anomalies in my log data that would help me reduce or remove threats in my environment. I tried many approaches over the years, starting with scripts and moving to log analysis tools and suites both open source and commercial. None of them provided the filtering and detection capabilities that I needed without having significant foreknowledge of the threats. Marcus and I were able to design, build, and implement a generalized detector and filter for arbitrary text-based data. It is useful and effective at detecting anomalous events with minimal prior knowledge or understanding of the event. This required some initial planning and discussion about what log data feeds might be interesting and ways to score the scenarios. As with general log analysis, the best way to do it is to look at the logs, build the rules, and prototype and test them.

REFERENCES

[1] http://sourceforge.net/projects/swatch/.

[2] http://www.gnu.org/software/xlogmaster/.

[3] http://www.ranum.com/.

[4] http://www.faqs.org/rfcs/rfc3164.html.

[5] http://dinosaur.compilertools.net/.

[6] http://en.wikipedia.org/wiki/Printf.

[7] http://www.uberadmin.com/Projects/quickparser/index.html.

[8] http://en.wikipedia.org/wiki/Huffman_coding.

[9] http://www.uberadmin.com/Projects/logstore/index.html.

[10] http://www.ranum.com/security/computer_security/code/nbs.tar.

[11] http://www.ranum.com/security/computer_security/code/overwatch.tar.

[12] http://www.uberadmin.com/Projects/pdnsd/index.html.

[13] http://en.wikipedia.org/wiki/Conficker.

STEPHEN CHECKOWAY, HOVAV SHACHAM, AND ERIC RESCORLA

Don't take LaTeX files from strangers

Stephen Checkoway is a PhD student at UC San Diego. He works with Professors Shacham and Savage on embedded systems security, including electronic voting and automotive security.

s@cs.ucsd.edu

Hovav Shacham is an assistant professor in the Department of Computer Science and Engineering at the University of California, San Diego. His research interests are in applied cryptography, systems security, and tech policy.

hovav@cs.ucsd.edu

Eric Rescorla is a principal engineer at Skype. His research interests include communications security, the economics of vulnerabilities, and electronic voting.

ekr@rtfm.com

TEX, LATEX, AND BIBTEX FILES ARE a common method of collaboration for computer science professionals. It is widely assumed by users that LaTeX files are safe; that is, that no significant harm can come of running LaTeX on an arbitrary computer. Unfortunately, this is not the case. In this article we describe how to exploit LaTeX to build a virus that spreads between documents on the MiKTeX distribution on Windows XP as well as how to use malicious documents to steal data from Web-based LaTeX previewer services.

> I wrote out what I thought I would like to type—how my electronic file should look. And then, I said, OK, that's my input, and here's my output—how do I get from input to output? And for this, well, it looks like I need macros.
>
> —*Donald Knuth* [9]

Donald Knuth's TeX is the standard typesetting system for documents in mathematics and computer science. However, like many other text processing systems designed by computer scientists (PostScript, troff, etc.), what it really is is a general-purpose programming language specialized for typesetting documents. This is a fact that most TeX users don't think about much, and they (we) tend to treat TeX documents the way they would treat text files—as something inherently safe. Many a user who would never consider downloading and running a random program off the Internet doesn't think twice before feeding arbitrary data into his local copy of LaTeX.

TeX is extremely (legendarily) wel designed: Knuth actually gives out cash rewards to people who find bugs, and has made only a few minor changes to TeX in the last decade [3]. As one would expect, TeX generally restricts the functionality that documents and the macros they define can invoke. Nevertheless, it allows macros to read and write arbitrary files. This single capability turns out to be enough to allow a carefully crafted document to completely escape TeX's sandbox. As a demonstration, we present a TeX virus that affects recent MiKTeX distributions on Windows XP, and that, with no user action beyond compiling an infected file, spreads to other TeX documents in the user's home directory. Our proof-of-concept virus carries no malicious payload beyond replicating itself, but it could just as easily download and execute binaries or undertake any other action.

The vulnerabilities exposed by TeX's file-I/O capabilities extend beyond a user's personal computer. TeX is the *lingua franca* of mathematics and the mathematical sciences; its notation is frequently used even in communication (e.g., in email between collaborators) that isn't meant to be run through the TeX program. And TeX does such a good job of formatting mathematical formulae (and other programs do such a bad job) that it's common to write one's formulae in TeX, render them into images, and then embed them into a Web page, a Word document, or a PowerPoint presentation. A large number of Web-based TeX previewers exist to facilitate the process of turning TeX equations into an embeddable image or PDF. Unfortunately, many of these previewers fail to properly isolate the TeX program, with the result that it is possible merely by sending them a malicious document to remotely download sensitive information such as the documents rendered by previous users or even—under the right conditions—the remote system's password file. Even here, the danger is potentially more widespread. Because the TeX core has not changed for many years, which makes TeX an archival format, many archive services, such as Cornell University's popular arXiv.org, accept submissions in TeX, which they compile to produce PDFs.

It is important to realize that the file I/O capabilities at the heart of the vulnerabilities we identify are not bugs in TeX; rather, they are intended capabilities exposed by TeX's macro language that were not fully understood and accounted for by the designers of larger systems (such as online previewers) of which TeX is a component. In this way the vulnerability is of a different kind than the programming error frequently reported in image-handling software (including, in one notorious example, Microsoft Windows' handling of animated cursor files [6]), in which insufficient validation by the program of attacker-supplied input leads to memory corruption and arbitrary code execution. No such programming errors are known in TeX, though Knuth, writing recently, did not disclaim their existence [3]:

> Let me also observe that I never intended TeX to be immune to vicious "cracker attacks"; I only wish it to be robust under reasonable use by people who are trying to get productive work done. Almost every limit can be abused in extreme cases, and I don't think it useful to go to extreme pain to prevent such things. Computers have general protection mechanisms to keep buggy software from inflicting serious damage; TeX and MF are far less buggy than the software for which such mechanisms were designed.

We believe that there are two important lessons to draw. First, one must be cautious about which TeX and LaTeX files one compiles. This is actually harder than it sounds: While most people don't routinely compile LaTeX source from untrusted sources, they do compile BibTeX entries. For instance, ACM Portal provides BibTeX entries for each of its articles. Because BibTeX entries can (surprise!) contain LaTeX code, this is equally dangerous and much harder to verify, especially if you download large bibliography files such as Joe Hall's well-known election auditing bibliography [2]. This brings us to the second lesson: Executable code is everywhere, even in formats that you would expect just to be passive data. And because it's so difficult to build an effective sandbox, our intuitions about what formats are inert (and hence safe) can lead us very far astray.

How to Write a TeX Virus

In this section we show how to write a virus that is carried in a TeX file. As explained above, our virus is made possible by the file output capability exposed to TeX documents. Unlike other modern distributions of TeX, MiKTeX, the most common TeX distribution for Windows, places no meaningful restrictions on this capability.

Given the ability to overwrite system files, it is not surprising that TeX documents can compromise the security of the system on which they are compiled. For concreteness, we focus on one convenient target: on Windows XP, a JScript file written to a

user's Startup directory will be executed by the Windows Script Host facility at login; the Windows Script Host exposes COM objects to scripts that allow easy manipulation of the filesystem.

Our JScript startup script, when run on the user's next login, seeks out other LaTeX files on disk and infects them with our virus. The virus lifecycle is summarized in Figure 1.

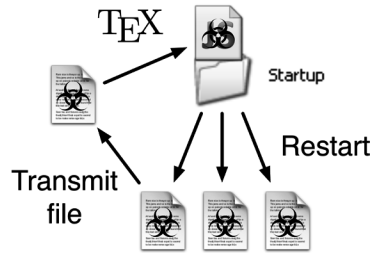


FIGURE 1: LATEX VIRUS LIFECYCLE

WRITING THE MALICIOUS FILE

Writing the malicious JScript file is conceptually simple. The TeX write primitive allows us to write data to a file, like so: \write\file{foo} Since we have the malicious JScript embedded in our document, we can just write it to disk. However, there is one technical hurdle that must be overcome in order to write to the Startup directory: the full path of the directory is C:\Documents and Settings\Administrator\Start Menu\Programs\Startup, but TeX does not ordinarily allow spaces in file paths (this does not appear to be a security feature, just a functional defect). However, we can leverage Windows' compatibility with older programs that expect file and directory names in 8.3 format. For example, Start Menu can be specified as STARTM~1. This mechanism allows us to bypass the path restriction.

In addition to the JScript file, we also write a copy of the virus to the disk at an easily accessible location, for use by our JScript in viral spread. For convenience, we just write the entire original document, virus and all. For this, we take advantage of the fact that the TeX engine used in MiKTeX—and indeed in all modern TeX distributions—is pdfTeX, which contains the ε-TeX extension \readline [7]. We use \readline to read the document being compiled line by line and write an exact copy to C:\WINDOWS\Temp\sploit.tmp.

The complete source for the TeX portion of our virus is given in Listing 1. We give the details of how it accomplishes the tasks listed above in our LEET '10 paper [1].

```
%%%%SPLOIT%%%%
{\newwrite\w\let\c\catcode\c`*13\def*{\afterassignment\d\count255"}\def\d{%
\expandafter\c\the\count255=12}{*0D\def\a#1^^M{\immediate\write\w{#1}}\c`^^M5%
\newread\r\openin\r=\jobname \immediate\openout\w=C:/WINDOWS/Temp/sploit.tmp
\loop\unless\ifeof\r\readline\r to\l\expandafter\a\l\repeat\immediate\closeout
\w\closein\r}{*7E*24*25*26*7B*7D\immediate\openout
\w=C:/DOCUME~1/ADMINI~1/STARTM~1/PROGRAMS/STARTUP/sploit.js \c`[1\c`]2\c`\@0
\newlinechar`\^^J\endlinechar-1*5C@immediate@write
@w[fso=new ActiveXObject("Scripting.FileSystemObject");foo=^^J
<11 lines of JScript omitted>
f(fso.GetFolder("C:\\Documents and Settings\\Administrator"));}m();]
@immediate@closeout@w]}%
%%%%SPLOIT%%%%
```

LISTING 1: THE LATEX COMMANDS THAT CREATE THE SPLOIT.TMP FILE; THE JSCRIPT CODE HAS BEEN OMITTED.

SPREADING THE DISEASE

The second phase, written in JScript, is automatically executed by Windows when the user next logs in. It reads the sploit.tmp file, extracts from it the TeX virus code, finds all the files in the Administrator directory with the extension .tex, and appends the virus onto each of them. To manipulate the filesystem, it instantiates Microsoft's convenient FileSystemObject, which exposes a programmatic interface for filesystem search and manipulation.

In total, the virus requires two marker lines and 21 80-column lines of TeX. Listing 1 omits most of the JScript, in the interests of not providing a complete, working virus; but the remaining code is straightforward and we have tested it in our own systems.

We stress that JScript code run from the file system is unsandboxed. Our virus could manipulate the file system however it wishes, or download an arbitrary program from the Internet and cause it to be executed. The damage caused by the vulnerability could in principle be far greater than just modifying LaTeX files on disk.

APPLICATIONS OUTSIDE WINDOWS

While Windows is the easiest platform to exploit, exploits on other platforms are possible as well. As an example, consider the TeX Live distribution popular on UNIX platforms (including Mac OS X). Like MiKTeX, TeX Live allows any file to be read. Unlike MiKTeX, in its default configuration TeX Live prohibits TeX documents from writing to "dotfiles" (files whose names start with a dot, such as ~/.login, the user startup script for Bourne-derived shells) or files not in its current directory or subdirectories.

Even with these restrictions, however, there may be avenues for attack. For instance, if a makefile is being used to run LaTeX, then the attacker can overwrite it, inducing arbitrary behavior the next time the make program is run. In addition, the popular Emacs-based TeX editing environment AucTeX writes Emacs Lisp cache files to the local directory; an attacker who overwrites these files can execute arbitrary Lisp code inside Emacs, which itself is Turing-complete and unsandboxed. (For an earlier example of a TeX virus that used Emacs for propagation, see [4].)

Attacks on Previewers

We now turn our attention to a slightly harder target. There are more than a dozen Web-based services that compile LaTeX files on users' behalf and make the resulting PDFs available. While some of the operators of these sites seem to be dimly aware that attacks may be possible, in nearly every case we were able to read server files remotely and, in many cases, to write loops that could be used for denial of service via resource consumption. The one previewer we were unable to attack, MathTran [8], uses Secure plain TeX, a reimplementation of plain TeX that prevents using any control sequence other than those meant for typesetting.

We have designed successful exfiltration and denial of service attacks on most of the LaTeX previewer services we studied. Moreover, the filtering mechanisms devised by these services were largely ineffective against our attacks. We disclosed the vulnerabilities of the affected services we found to the operators, with universally positive responses. As a result, a number of operators changed their security policy or removed the previewer altogether.

In the rest of this section we describe some of the details of our attacks.

EXFILTRATING DATA

Our key insight is this: any data that can be read by the TeX script being compiled can be incorporated into the PDF file that is its output. When that PDF file is made

available to the attacker, he can read it to recover the data. A data exfiltration vulnerability is thus created whenever Web-based TeX previewers allow scripts to read files on disk that are not otherwise made public by the Web server.

This attack can be implemented in a number of ways. The most obvious way uses input to interpolate the text of the file being read into the TeX input and hence the output document. A minor problem with this approach is that it loses line breaks in the input file, since TeX will treat them as spaces in the usual manner. To avoid this, we can instead use the ε-TeX \readline extension, as we did in our virus. Using this (rarely-used) control sequence also evades any blacklisting of input by the preview service's developers.

In principle, the procedure is straightforward. Our malicious TeX program opens the sensitive file for reading and, in a loop, reads and typesets each line. When the preview service displays the output in the attacker's browser, the contents of the sensitive file are exposed.

For the preview services we examined, the procedure was, in some cases, slightly more complicated. The first barrier to overcome is that many of these previewers are designed to typeset a single equation, and, as a consequence, interpolate the user input into a mathematics environment in an otherwise-complete LaTeX document for processing. Similar to basic SQL injection attacks, this attack requires the attacker to escape math mode to perform some operations. A further barrier is that some of the preview services explicitly disallow some control sequences, such as \input or \include—rightly recognizing their potential for misuse. This is a very natural defense; however, the availability of other macros for file I/O and the malleability of LaTeX code make possible a host of techniques for defeating blacklist or whitelist filters, ranging from using equivalently powerful internal LaTeX macros to exploiting the way TeX parses its input and, in particular, how it decides what is a control sequence. Again, see our paper [1] for more details.

DENIAL OF SERVICE

Any previewer that allows the TeX looping construct \loop...\repeat or the definition of new macros is at risk of a denial of service attack. One can create a simple loop:

```
\loop\iftrue\repeat
```

or one can define a recursive macro such as:

```
\def\nothing{nothing}
```

In the absence of imposed resource limits, enough such loops executed in parallel will slow the server machine to a crawl and no more useful work will be possible until the processes are killed. One extension of this attack is to cause TeX to produce very large files, potentially filling up the disk.

The Origins of Insecurity in the Breakdown of the Code/Data Distinction

The vulnerabilities described in the previous sections are examples of a much broader problem: the big shift toward active content. It's common to think of there being a sharp distinction between "code" and "data": code expresses behavior or functionality to be carried out by a computer; data encodes and describes an object that is conceptually inert and is examined or manipulated by means of appropriate code. Programs (Web browsers, word processors, spreadsheets, etc.) are code. Documents (Web pages, text documents, spreadsheet files, etc.) are data, and data is safe.

This distinction is increasingly false. All of the "document" formats mentioned above routinely contain active content (JavaScript, macros, etc.) which is run in the context of whatever program you use to work with the data. When those programs do not

properly sandbox the active content, then viewing a seemingly inert document can be just as dangerous as directly executing a program from an unknown source. For example, PDF files can embed JavaScript, which allows PDF files that include malicious JavaScript to exploit bugs in Adobe's Acrobat; by one report [5], some 80% of exploits in the fourth quarter of 2009 used malicious PDF files. Unfortunately, as long experience has shown, proper sandboxing is very hard.

The insecurity we have identified in TeX is one more example of the weakness of this kind of thinking. In TeX, we have a piece of extremely well written software designed for a superficially safe activity (text processing). What's more, whereas PDF files and most other media formats are binary and opaque, the input file formats associated with TeX are all plaintext and thus, naïvely, transparent and auditable. Nevertheless, executing TeX files from untrustworthy sources is fundamentally unsafe: compiling a document with standard TeX distributions allows total system compromise on Windows and information leakage on UNIX. Simply put, every time you compile someone else's LaTeX file or cut and paste a BibTeX entry from a Web site, you are engaging in unsafe computing. Note that the LaTeX source for this article is available from the authors upon request.

You would do well, as Knuth suggested, to avail yourself of those operating-system protection mechanisms designed "to keep buggy software from inflicting serious damage."

REFERENCES

[1] Stephen Checkoway, Hovav Shacham, and Eric Rescorla, "Are Text-Only Data Formats Safe? Or, Use This LaTeX Class File to Pwn Your Computer," *Proceedings of LEET '10*, USENIX, April 2010.

[2] Joseph Lorenzo Hall, "Election Auditing Bibliography," version 3.8, February 2010: http://josephhall.org/eamath/bib.pdf and http://josephhall.org/eamath/eamath.bib.

[3] Donald E. Knuth, "The TeX Tuneup of 2008," *TUGboat*, vol. 29, no. 2, 2008, pp. 233–38: http://www.tug.org/TUGboat/Articles/tb29-2/tb92knut.pdf.

[4] Keith Allen McMillan, "A Platform-Independent Computer Virus," master's thesis, University of Wisconsin—Milwaukee, April 1994: http://vx.netlux.org/lib/vkm00.html.

[5] ScanSafe, "Annual Global Threat Report, 2009": http://www.scansafe.com/downloads/gtr/2009_AGTR.pdf.

[6] Alexander Sotirov, "Windows ANI Header Buffer Overflow," March 2007: http://www.phreedom.org/research/vulnerabilities/ani-header/.

[7] Hàn Thế Thành , Sebastian Rahtz, Hans Hagen, Harmut Henkel, Paw Jackowski, and Margin Schröder, "The pdfTeX User Manual," January 2007: http://www.tug.org/texmf-dist/doc/pdftex/manual/pdftex-a.pdf.

[8] The Open University, "MathTran" (online translation of mathematical content): http://mathtran.open.ac.uk.

[9] Christina Thiele, "Knuth Meets NTG Members," *MAPS,* vol. 16, March 13, 1996, pp. 38–49: http://www.ntg.nl/maps/16/15.pdf.

BO LI, ELENA RESHETOVA, AND TUOMAS AURA

Symbian OS platform security model

Bo Li is a second-year student in the master's program in security and mobile computing at Aalto University, Finland. He got his bachelor's degree in communications engineering in 2008 from Fudan University, China.

Bo.Li@tkk.fi

Elena Reshetova is a senior security engineer at Nokia, as well as a postgraduate student at Aalto University. She is interested in various research areas related to platform security, security aspects of networking, and cryptography.

elena.reshetova@nokia.com

Tuomas Aura is a professor at Aalto University, Finland. His research interests are security and privacy in communications networks and online services.

tuomas.aura@tkk.fi

THE SYMBIAN OS BECAME FULLY OPEN sourced in February 2010, which opens even more possibilities for application developers to understand and analyze its security solution. We present a short introduction to the software features of Symbian platform security: three trust tiers, capability model, data caging, and the Symbian signed process. We also try to compare the security solution with the classical design principles in this area, as well as briefly discuss general design challenges and potential weaknesses.

Introduction

With the development of mobile devices and mobile computers, more and more people rely strongly on them. People use mobile devices and mobile computers to arrange their schedules, contact each other, process emails, and share rich media content. People believe it is safe to do so because it feels secure just knowing it is "right there with you" [8]. But, in fact, even their manufacturers agree that all mobile devices are facing various security threats and attacks [3]. Moreover, the more widespread the phone's operating system is, the higher the risk that it will be targeted by attackers, especially if it allows execution of third-party applications. According to Symbian site information, over 80 million devices running Symbian OS were sold in 2009 [7]. The public development tools for the Symbian platform have been easily accessible for a long time, and Symbian also supports execution of add-on applications. So the Symbian OS had and still has a strong need for a well-designed security solution.

Despite the fact that modern smartphones operate like minicomputers, several characteristics suggest that their security solutions cannot be the same as the general security solutions for PCs.

Compared with the users of computers, the users of mobile devices have some particular expectations. People believe they should be able to make an emergency call whenever necessary. This means that, no matter how many applications are running, the mobile phone must have very high reliability; people also do not wish to restart their mobile phones every day as they do with desktop computers. This requires the mobile OS to be very stable. And although it seems more and more necessary to install antivirus software on smartphones

now [13], most users still do not expect to install any antivirus application on a mobile device, at least not to have to install it themselves.

Alongside these particular expectations from users, the mobile device itself has some constraints which should be taken into account when designing a security architecture. This is mainly due to the limitations, especially when compared with computers, of the mobile devices' hardware: low processing power, limited memory and battery, small screen size, and inconvenient keyboard. All of these require a light but efficient security architecture.

Another important characteristic of mobile devices indicates even more serious vulnerabilities and challenges. "Phones are primarily used to communicate. They are built to make communication as easy as possible," notes Aaron Davidson, CEO of antivirus company SimWorks. "Phone users want to communicate, and viruses want to be communicated" [12]. Every mobile phone has a contact list; if one phone is infected by malware, the people on the contact list will be vulnerable to attack.

To sum up, mobile device security has some similarities with computer security, but faces additional challenges.

Basic Design Principles

Saltzer and Schroeder's classic article on computer security design [9] lays out essential principles to follow when designing a security solution. In this section we will explain some of these principles, as well as show how they were applied in the Symbian platform security design.

- Economy of mechanism: Economy of mechanism means the design should be small and simple. The simpler a system is, the fewer vulnerabilities it should have. When we apply this rule to the Trusted Computing Base (TCB) design, fewer trusted components will result in fewer internal trust relationships and a simpler system. In Symbian OS, the only fully trusted part is the Trusted Computing Base, which is small enough to be well tested and reviewed.
- Fail-safe defaults: In the security field this principle means that, by default, access to a system's resources should be denied. This is needed to prevent the situation where some sensitive resource is forgotten from the list of resources but must be protected. When applied to users of a computer system, the principle may also mean that a user should be given a secure default choice when presented with a security question. Symbian follows the "deny by default" rule in its design, but the graphical interface does not always advise a secure choice for a user. For example, when an application tries to connect to the network, the user is asked whether it is allowed to do so or not. The "yes" button (meaning "allow connection") is situated on the left and is the button the user usually presses. So, the user tends to choose the unsafe option presented in that graphical interface.
- Complete mediation: Every time any system resource is accessed, there should be a check whether this action is authorized for a caller. Symbian Platform Security includes access control checks on all platform resources, but it does not provide any mechanism for applications that need to provide flexible access control to the applications' own resources. We will discuss this problem in more detail in the discussion section.
- Open design: The security solution should not try to build security by obscurity, but, rather, rely on secure storage for cryptographic keys and other security material (e.g., certificates, initialization vectors, etc.). Nowadays, Symbian almost fully follows this principle by open sourcing the operating system itself, including the security solution. However, the information about hardware security support is still not fully public.

- Principle of Least Privilege: Least privilege means that a process should be allocated sufficient privileges to accomplish its intended task but no more. However, this principle does not speak about any granularity of privileges, so often it is interpreted quite broadly. Symbian Platform Security addresses this principle by its capability model.
- Psychological acceptability: Psychological acceptability requires that the user interface should be intuitive and clear. An elegant interface, combined with a precise definition of its behavior, promotes the correct and secure use of the system. For example, Symbian should convert the various low-level capabilities into several user-friendly descriptions, such as: "The application wants to open the Bluetooth connection: allow or not?"

One principle that needs to be added to this classic list is the notion of "trusted root," meaning that there should be a chain of trust going from the hardware to upper parts of the operating system in order to secure the end of the chain from offline attacks.

Concepts of the Symbian OS Security Model

In this section we will introduce three important concepts in Symbian OS platform security: three trust tiers, capabilities, and data caging.

TRUSTED COMPUTING PLATFORM: THREE TRUST TIERS

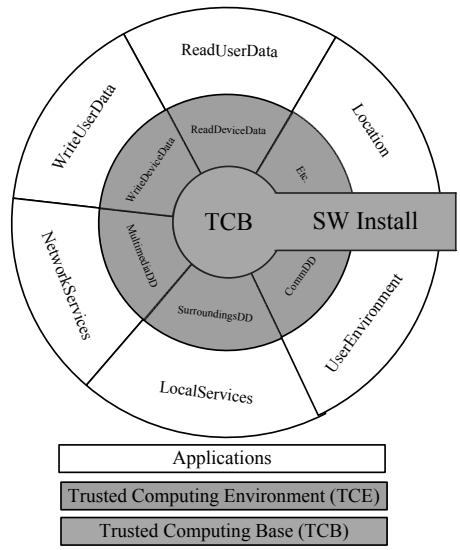


FIGURE 1: THREE TRUST TIERS [8]

The first essential concept in Symbian OS platform security is the "three trust tiers" model, shown in Figure 1. Figure 1 depicts a trusted computing platform comprising a Trusted Computing Base (TCB), a Trusted Computing Environment (TCE), and Applications.

- Trusted Computing Base (TCB): The heart of a trusted computer system is the Trusted Computing Base (TCB), which contains all of the elements of the system responsible for supporting the security policy and supporting the isolation of objects (code and data) on which the protection is based [10]. TCB is the most trusted part of Symbian OS. It has three components: the operating system kernel, the file server (F32), and the software installer. All of them have been carefully checked to ensure that they behave properly and can be completely trusted. The kernel has the responsibility for managing all the processes and assigning appropriate privileges to them.

The file server is used to load the code for running a process. The software installer is used to install applications from files packages. It also checks the digital signatures of the packages to validate the privileges requested for the program binaries, so it is the "gatekeeper" for the mobile device.

- Trusted Computing Environment (TCE): The next tier is the TCE, which is a set of different system servers running with different privileges. TCE consists of trusted software provided in the mobile phone by Symbian and others, such as the UI platform provider and the mobile device manufacturer [8]. The TCE usually implements a system server process and is less trusted, and thus it has only limited privileges to perform a defined set of functions. With this design, Symbian can ensure that failure of one server will not threaten the whole system, and it is also impossible for a misbehaving system server to compromise the security of another server, since it does not have access to the same APIs [16].
- Applications: Third-party applications are the last tier in this trusted computing model. As they are not highly trusted, they only have privileges to access the services that are unlikely to pose a security risk, and they are not allowed to access critical low-level operations. There are actually two kinds of applications: signed applications and unsigned applications. If a signed application wants to use some service provided by TCE, it must request that TCE run the service on its behalf. TCE will only accept the request if the application has been granted enough privileges. The unsigned applications, however, can only perform some operations that do not require privileges, or operations that may be allowed by the user. It is necessary to point out that there are a lot of useful operations which can be performed by unsigned applications and which do not require a signature because they are not security-relevant. A user may allow some operations to be performed even if an application is not signed, via a user-prompt dialog. An unsigned application, then, is not necessarily worthless software or malware: an application should have a signature only when it is necessary.

To summarize, the TCB is the most trusted part and controls access to all sensitive low-level operations such as the communication device drivers. The TCB ensures that only the privileged TCE components are able to perform these operations. The TCE then provides system services, such as the telephony server, to applications. This is the only way for the applications to perform the low-level operations [8].

CAPABILITIES DETERMINE PRIVILEGE

The second basic concept in the Symbian OS platform security model is capabilities. A capability is an unforgeable ticket, which when presented can be taken as incontestable proof that the presenter is authorized to have access to the object named in the ticket [9].

Capabilities Basics in Symbian OS

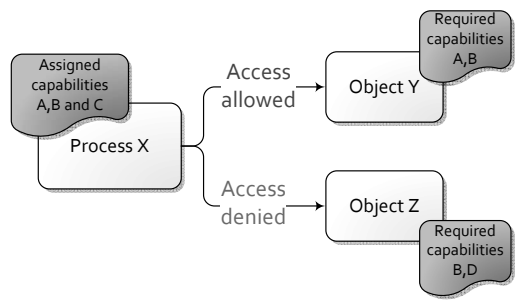


FIGURE 2: ACCESS CONTROL BASED ON CAPABILITIES

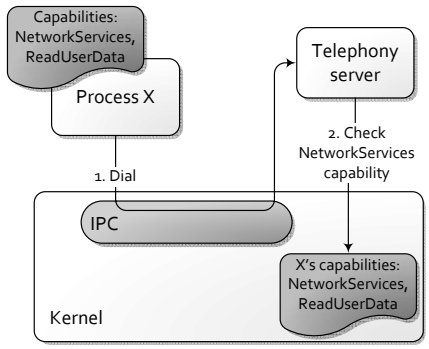


FIGURE 3: ACCESS TO SERVICES VIA THE TCE

In Symbian OS, each process runs with a list of capabilities, and these capabilities determine whether a given resource can be accessed by the process. As shown in Figure 2, process X (with capabilities A, B, and C) can access the resource Y, which requires capabilities A and B, but cannot access the resource Z, which also requires capability D. The concrete form used to access the resources is through APIs, and different APIs will require different capabilities for the services they provide.

The Symbian OS uses capabilities to represent access privileges. Each live process and its corresponding capabilities are listed and monitored by the system kernel. A process will ask the kernel to check the capabilities of another process before deciding whether to carry out a service on its behalf [8]. Figure 3 shows how a well-known example of a client application X accesses the Dial service via the telephony server (one component of the TCE). First, process X sends the Dial request to Inter-Process Communication (IPC, part of the kernel), which then delivers this request to the telephony server process. The Dial service requires NetworkServices capability, and the system kernel holds the capability list of X. The telephony server then checks X's capability list in the kernel and finds the capability NetworkServices there, enabling access from X to the Dial service. The capabilities architecture is discrete but not hierarchical; each capability is only associated with its corresponding resource, and capabilities do not overlap.

When the capabilities are designed and divided, the outcome is actually a trade-off between the principles of "economy of mechanism" and "least privilege" introduced in the section on Basic Design Principles, above. If only " economy of mechanism" is considered, the system will have only one capability with full access to all resources. This system is simplest and is easy to review, but it obviously doesn't work. If "least privilege" is the only principle followed, every API would have a dedicated capability, amounting to over 1000 capabilities in the Symbian OS. This is also impossible to manage and realize. The number of capabilities should be small enough to manage but also reasonably big enough to show the differences between different privileges. To meet this requirement, Symbian OS defines 20 capabilities with their special privileges.

Categories of Capabilities in Symbian OS

The capabilities are divided into several categories according to the three trust tiers model; basically, the three kinds of capabilities are TCB capability, system capabilities, and user capabilities. The categories and brief descriptions of the capabilities are shown in Table 1.

TABLE 1: CATEGORIES OF CAPABILITIES IN SYMBIAN OS [3, 6]

Category	Capability	Brief Description
User Capabilities	LocalServices	Grants access to sending or receiving information through USB, IR, and point-to-point Bluetooth profiles
	ReadUserData	Grants read access to confidential user data
	WriteUserData	Grants write access to confidential user data
	NetworkServices	Grants access to remote services such as dialing a number or sending a text message
	UserEnvironment	Grants access to recording the user's voice and using the camera
	Location	Grants access to data about the location of the device
System Capabilities (extended capabilities)	SwEvent	Grants the right to simulate key presses, pen input, and capture such events from any program
	ProtServ	Grants the right to a server to register with a protected name
	TrustedUI	Grants the right to create a trusted UI session, and, therefore, to display dialogs in a secure UI environment
	PowerMgmt	Grants the right to kill any process in the system, to power-off unused peripherals, and to cause the mobile phone to switch its machine state
	SurroundingsDD	Grants access to logical device drivers that provide input information about the surroundings of the device
	ReadDeviceData	Grants read access to sensitive system data
	WriteDeviceData	Grants write access to sensitive system data
System Capabilities (manufacturer-approved capabilities)	CommDD	Grants access to communication device drivers
	DiskAdmin	Grants the right to disk administration functions that affect more than one file or directory such as formatting a drive
	MultimediaDD	Controls access to all multimedia device drivers (sound, camera, etc.)
	NetworkControl	Grants the right to modify or access network protocol controls
	AllFiles	Grants visibility to all files in the system and extra write access to files under /private
	DRM	Grants access to alter DRM-protected content
TCB Capability	TCB	Grants access to the /sys and /resource directories in the device

- TCB is fully trusted and has the highest privilege, so there should be a capability owned only by TCB. This capability is just called *TCB*, which allows one process to create new processes and assign capabilities to them. Generally, the TCB capability is not granted to other parts besides TCB, as it is really critical and closely related to the integrity and security of the whole OS.
- System capabilities are assigned to access sensitive operations. Basically, system capabilities are divided into two parts: manufacturer-approved capabilities and extended capabilities. Manufacturer-approved capabilities are the most sensitive capabilities. They are reserved by the mobile device manufacturers and are enforced by TCB. Extended capabilities control access to higher-level services and are enforced by TCE.
- User capabilities are also called "basic capabilities." Following the psychological acceptability principle, user capabilities should be simple and easy to understand, as they are defined for user interaction. User capabilities are

usually granted to third-party applications to access the service provided by the TCE, and the TCE is responsible for enforcement.

Capability Rules

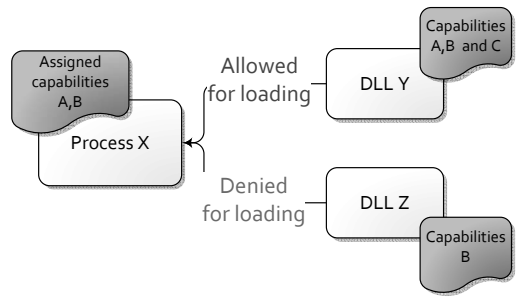


FIGURE 4: LOADING OF DLLS

There are two basic capability rules in the design of Symbian. The first rule is that every process has a set of capabilities and its capability never changes during its lifetime [8]. This rule is included in Symbian OS for simplicity and security. The second rule states that a binary cannot load any DLL that has fewer capabilities than itself (see Figure 4) [6]. This prevents untrusted code being loaded into sensitive processes. The capabilities of a DLL do not affect the capabilities of the process that loads it; process capabilities are entirely defined by the capabilities of the EXE.

FILE ACCESS CONTROL SYSTEM

The last essential principle in Symbian OS platform security architecture is the file access control system. It is designed to protect the integrity and confidentiality of critical files. The concrete solution is called "data caging" or "file caging."

Basic Data Caging Principles

The original idea of data caging is simple: put the most important treasures into a coffer and enforce strict access control to it, so that no one else can easily access the treasures inside it. In Symbian OS, data caging is achieved by providing special directories that "lock away" files in private areas [8]. Data caging is a lightweight mechanism, as the access control of the files only relies on their path; thus, it works without another access control list, which would consume additional system resources.

Data caging also follows the Principle of Least Privilege, by which a process cannot access these special directories unless it is authorized. If we take a close look at this aspect, there are two special capabilities closely related with data caging: TCB and AllFiles. These are enforced by the TCB. If these capabilities are granted, they will apply to all the files in the system. So they are really critical and should not be assigned to one specific application. In Symbian, the solution for this problem is to provide several different system services with more limited privileges such as ReadUserData, WriteUserData, and so on, so the access permission is narrowed and divided.

Caged Paths in Symbian OS

In Symbian OS, there are three caged top paths: \sys, \resources, and \private. All their subdirectories are also access restricted. Only the processes with TCB and AllFiles capability may access these directories.

- \sys
  There are two important subdirectories under \sys: \sys\bin and \sys\hash. \sys\bin is the default path where all the binary files are stored. The pre-installed binaries are located at z:\sys\bin, which is a path in the ROM; and add-on applications are under \sys\bin on some writable devices. In Symbian OS, there are two rules defined for this path. First, only TCB can write new executables into this path (via SWInstall) or execute the binaries under this path (via F32). Second, only the binaries under this path are runnable; all the other paths will be ignored by the loader. This ensures the security and integrity of the system, since only TCB, not malware, can create a new process.
  \sys\hash is used to check whether a binary on removable media can be launched. It is managed by the software installer. The installer will check the presence and correctness of the hash, and the hash entry for a binary will not be generated unless the installation has been validated. This mechanism ensures the security of running a binary on removable media.
- \resources
  Similar to \sys, \resources exists in both ROM and writable storage. The files under \resources are read-only for most of the applications, because this path is used to store the resource files, which will not be changed after installation. Only an application with TCB capability (installer application) can modify the files under this path, ensuring that resources installed for one application will not be destroyed by other applications.
- \private
  In Symbian, every EXE is assigned its own caged subdirectory under \private. The subdirectory is open to its own process but is inaccessible to other normal processes. Particularly, if two processes are loaded from the same EXE, they share the same subdirectory [8]. The subdirectory under \private is the default path to store the data of the process, but the developer can also choose to store the data of their application in a public directory.

Software Installer in Symbian OS

The software installer plays an important role in Symbian OS platform security, as it is one of the gatekeepers of the system. It is responsible for ensuring that add-on native software is installed on the mobile phone with the correct set of security attributes [8]. The software installer discussed here only handles the installation of the software directly running on Symbian OS, but not the software running on a Java Virtual Machine.

There are three main tasks for the installer of Symbian OS: first, to validate and install the native Software Install Scripts (SIS files) on the mobile device; second, to validate pre-installed software on removable media; lastly, to manage upgrades and removals and provide the package management service to the rest of the system.

IDENTIFIERS

Identifiers are used by the servers to identify processes. There are several different kinds of identifiers in Symbian OS which are important for the software installers. They are the Secure Identifier (SID), Vendor Identifier (VID), and package UID (pUID).

The SID is used to identify the binaries, and it is locally unique. The VID is used to distinguish the origin of the executable. If an application needs a VID, it must be signed [17]. The pUID is the identifier of a package, or set of

files, that forms an installable unit [8]. For example, if we download one installable package of a game from Electronic Arts (EA)'s Web site and install it into our Symbian mobile phone, the pUID is the ID of this package, the VID is the ID of the EA company, and the SID is the ID of the executable binary of the game after installation.

Identifiers are split into protected and unprotected ranges. Every identifier will be allocated in these ranges, according to or not whether the application will be signed.

SIS FILES AND REMOVABLE MEDIA

SIS (Software Installation Script) files in Symbian are used to deliver software packages to mobile phones for installation and can be installed from a PC, downloaded via a browser, or sent to a mobile device by MMS [8]. When the software installer validates the SIS files, it checks a lot of parameters. In general, it checks the capabilities the software wishes to use for authorization. First, it checks to see whether the software is signed; second, it checks that the certificate chain can be followed back to the root; then it checks whether the certificate has been revoked. It can also mark root certificates as mandatory, ensuring that all installable packages must be signed with it.

After the validation of authority, the installer compares the capabilities requested by the installable package with those that the root certificate can grant, and calculates the largest set of capabilities that can be granted. A configuration option allows the end user to grant additional capabilities to the SIS file if these capabilities are not granted by the installer directly. If the user refuses to grant the additional capabilities, the software will not be installed [17]. Another possibility is to ask for the user grantable permissions during runtime, but this is annoying to some users. The overall process of installing a signed application is shown in Figure 5.

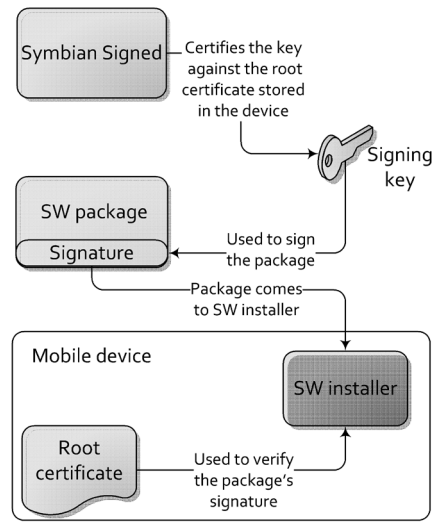


FIGURE 5: INSTALLATION PROCESS FOR SIGNED PACKAGES

Besides the installation from the SIS files, the software can also be installed onto removable media. The most important security issue is to prevent tampering with the binaries installed on removable media. This is achieved by the software installer computing and storing (in the tamper-proof \sys\ directory) a reference hash of the program file on installation, which is recom-

puted and compared whenever the program is to be run [17]. If the reference hash does not exist, or the two hashes do not match each other, the program will not be executed.

Code Example

There are many Symbian C++ code examples of differing complexity on the Internet, as well as many developers' forums where people ask security-related questions. The most common question regarding access control concerns capabilities, which are needed in order to use a particular class or function. Let's take as an example a class CTelephony. The official API reference documentation [1] explains that this is the class that can be used to make a call (and for some other purposes, too). The actual function which initiates a call is CTelephony::DialNewCall, and the documentation says that it requires the user capability "Network Services" to use this function. The full example of the class, which can be used in order to handle a call, can be found in the Forum Nokia pages [2]. Listing 1 shows a small fragment of the code, which was modified in order to check the return status of the function.

```
1 iTelephony = CTelephony::NewL();
2 CTelephony::TTelNumber telNumber(aNumber);
3 iCallParams.iIdRestrict = CTelephony::ESendMyId;
4 iTelephony->DialNewCall(iStatus, iCallParamsPckg,
5                                   telNumber, iCallId);
6 User::WaitForRequest(iStatus);
7 if ( iStatus == KErrPermissionDenied )
8   {
9     // if we are here, then the call
10    // was stopped by the access control
11 }
```

LISTING 1: MAKING A CALL CODE EXAMPLE

The code (Listing 1) simply creates a class object CTelephony (line 1), prepares the function parameters, such as phone number (line 2) and CallID restriction setting (line 3), makes a new call (lines 4–5), and checks for the "KErrPermissionDenied" error code in order to check whether access was denied (lines 7–11).

Symbian Signed Model

The purpose of Symbian Signed is to enable third-party access to protected APIs and give users a basis for trusting third-party applications. If platform security is like a firewall locking down access to the system, then Symbian Signed is the mechanism that allows developers to negotiate entry through the firewall [14].

There are three different options for Symbian Signed: Open Signed, Express Signed (online and offline), and Certified Signed [15]. Open Signed can grant user and system capabilities to an application, and Open Signed offline can also grant the restricted capability set. However, the signature is limited to a device IMEI number and therefore is supposed to be used only for development purposes. The Open Signed online option also doesn't require a Publisher ID, which costs money for developers and can be granted only to a member of a company or organization. The Express Signed enables developers with a Publisher ID to sign and distribute their applications without IMEI restrictions. The cost per application's signature is small (10

euros), and the submitted applications are randomly tested for compliance with the Symbian Signed Test Criteria. This signature can be used for commercial purposes, but the restricted capability set can be granted to such an application. Certified Signed is the most common option for commercial software developers and can grant, in some cases, even access to manufacturer capabilities.

THE SIGNING PROGRESS

According to [8], the signing process consists of four key steps: (1) development, (2) developer authentication, (3) testing against industry-defined criteria, and (4) signing against the mobile device's root certificate.

The basic mechanism of the signing model works as follows: sensitive APIs are protected by capabilities; if the programs wish to access protected APIs, they must be granted the corresponding capabilities. The signing process is responsible for granting these capabilities, and it will encode the capabilities into digital certificates accompanying the application. The application certificates are validated at install time and the accesses to the critical APIs are also policed at runtime. Finally, when the process is running, the resource and privacy of the application will be protected by the additional security mechanisms of the system.

Analysis and Discussion

Earlier, the Symbian OS platform security was closed, no one was talking about the real implementation, and there were no publications about it. Nowadays, Symbian is fully open sourced, which is a very good step towards understanding its security solution. As there is no absolutely right solution, in this section we will discuss some problems of Symbian OS platform security and try to suggest some possible solutions. But before we start to go into detail, it is important to remember that while analyzing the solution to any problem, the constraints and given environment should be taken into account. One big constraint on the devices running the Symbian OS, at the time when the platform security was designed, was the processing power, which had a huge impact on the design of the security solution. However, today the processing power is no longer so constrained, and some decisions can be revised.

LIMITATIONS OF THE CAPABILITY MODEL

First, to ensure the security of the OS, Symbian OS restricts the capability set so that no one else can define a new capability. Of course, a fixed set of capabilities is easier to manage both inside the platform and during the Symbian Signed verification process, but sometimes having a flexible capability set is a very useful feature. For example, Symbian OS currently has no way for an application which would like to protect its own data with its own capability to enforce its own rules on how this capability can be granted to other applications.

Another problem is that the capability set is not fine-grained enough. As previously discussed, in order to keep the capability system simple, Symbian only defines 20 capabilities, not enough nowadays to achieve user-desired security. A lot of APIs are associated with the same capability, such as ReadUserData. Since user data is not fine-grained, one process can access either all user data or nothing. The result is that although a process—say,

a music player—should only be allowed to access part of the user data (the music files), through this capability it can also access data that should be off limits, such as the user's photos, documents, and even private email if they are stored in the user's folder.

Many people would argue that users should not be bothered with fine-grained security, which is of course true, but where is the border between needed flexibility and fine-grained solution? Who should define the granularity level? One possibility is to leave this choice to application developers and create a flexible mode which allows them to specify the granularity they need. Some developers prefer not to know much about platform security features, however, but simply to develop their applications "as before." Having a flexible but complicated system may frighten some beginners starting to develop applications for a particular platform.

Another possibility is to let the application certification authorities define the granularity, since they have to maintain the tools to check what permissions can be granted to a particular application. The last choice is to define the granularity of the level that an end user understands. Taking the end user into consideration is especially important if the user has to make any access control decisions on a platform, simply because users have to understand the ramifications of their actions in order to make a correct choice.

The question of granularity, then, should balance the needs of application developers, certification authorities, and end users. In the case of access control, end users become more and more security educated over time, especially users of high-end devices such as smartphones and minicomputers, as the generations change. Today, mobile device users understand the difference between their media files and email contents. However, Symbian OS platform security was designed when the notion of mobile device security was not very commonly understood, and ordinary users were not educated enough to think about their security and privacy to the deeper degree that Symbian proposed. At that time, it was reasonable to make the capability set small and simple, but for future solutions, the granularity probably should be increased.

PROBLEMS OF THE USER PROMPTING

As we mentioned before, when applications are installed or executed, the system will sometimes ask whether the user would like to grant some additional capabilities to the application; this can be dangerous [18]. Most applications downloaded from the Web are not signed, and, as normal users usually do not have enough security knowledge, this mechanism leaves open the possibility for malware to ask for some "user capabilities" from the users directly and then misuse the privilege. Many usability studies have shown that users tend to click "yes, yes, ok" in response to questions, after having become familiar with the process, which happens quickly. Moreover, since Symbian platform security does not fully provide a Trusted UI, there is no guarantee that a user prompt is fully secure and that it's actually an end user who presses the "allow" button on the screen.

Of course, many mobile operating systems use user prompts, mostly to guard themselves against liability, and put the consequences of a bad security decision on the user's shoulders. So how can we help the users to make better security decisions? What can be an alternative to a user prompt? One possible solution might be to allow a user access to a good software reputation system during the installation process. Such a system, if trusted and easy to use, may help users to make the right security decisions. Reputation

systems are hard to build, but if built correctly, they could provide a great service for users.

ISSUES WITH THE SYMBIAN SIGNED MODEL

Before starting to speak about issues with the Symbian Signed Model, let us recall its purpose. Symbian Signed allows third-party application developers to certify their software and therefore to get access to needed protected resources. This process is very important for any operating system that wants to support third-party applications and still check their quality. However, the process should be organized very carefully in order not to become a nightmare for applications developers and process maintainers.

Most software submitted to the Symbian signing process is checked by an antivirus engine; only some samples are checked manually. As the antivirus engine cannot guarantee detection of all potential malware, it is possible for evil codes to pass the signing process and acquire a digital signature. Recently, a Trojan called Sexy Space passed the security check of Symbian and obtained the digital signature [11]. Thus, there is always a possibility that malware can bypass the binary checks of Symbian Signed.

Also, the Symbian Signed process used to be quite painful for developers: it is slow, costly, and difficult. If your application needed to be certified for the sales version, for example, it used to take at least seven steps and one week to wait for the result. You also had to pay for the Publisher ID (200 USD/year) and the testing costs (at least 250 EUR) to get the signature [15]. The situation is slowly changing now, since the platform became open sourced and processes are evolving, but originally the service was not well executed.

This issue led to the current situation in China. The biggest Symbian fans community [5] in China made a survey on the N95 discussion board [4] asking if users wanted to "crack" their mobile phones. "Crack" here basically means disabling the installer's capability check mechanism to bypass the Symbian Signed problem. The results, which we translated from Chinese, showed that over 70% of the 1127 active users participated in the survey would like to do so (or already had done so) for a number of reasons. One reason is that they wanted to install some cool application which was not certified by Symbian Signed but which requires some system capabilities. Some other users bypassed the system by submitting the IMEI code of their devices in order to get their own developer certificates, so they could self-sign the application and therefore allow it to access needed resources on the devices. This simple example shows that if the process of application submission and certification is problematic, users and developers can always find a way to bypass it.

The last issue is that it is actually difficult for developers to know the exact capability set for their applications. As there is no automatic tool for the developers to detect the capabilities needed by their software, sometimes they just try to include as many capabilities as possible, whether or not the capability is needed.

Conclusion and Recent Developments

Symbian, as a leading mobile operating system, has a complete set of security solutions. The design of Symbian Security resolves the most important issues of mobile device security: the reliability of the OS, and the integrity and privacy of the critical data on the mobile device. It follows several classic design principles and makes careful decisions between trade-offs, given

the constraints imposed on the platform security solution at the time of its creation. The three essential concepts ensure the security of the system, while the software installer and signing model also keep the platform open for third-party developers. However, Symbian also has some weaknesses and drawbacks, which we hope will be taken into account when new platform security solutions for mobile devices are designed.

Recently a number of new OS security designs for mobile devices have been introduced, such as the Android and Maemo security frameworks. Their main difference from the Symbian solution comes from trying to utilize basic existing UNIX security in order to implement the desired functionality. The Maemo security solution tries to overcome a number of the earlier mentioned issues by providing an extensible set of capabilities (called "resource tokens" in Maemo), better support for development tools and processes, and a special mode of device operation where advanced developers can get almost full control over their devices. Time will tell, based on actual usage by users and developers, whether any of these frameworks will succeed, which is possible only if the lessons are learned from such examples as Symbian OS security.

REFERENCES

[1] CTelephony class API reference: http://carbidehelp.nokia.com/help/index.jsp?topic=/S60_5th_Edition_Cpp_Developers_Library/GUID-35228542-8C95-4849-A73F-2B4F082F0C44/sdk/doc_source/reference/reference-cpp/ETel_3rd_Party_API/CTelephonyClass.html.

[2] Example of CTelephony class usage: http://wiki.forum.nokia.com/index.php/Make_call_with_CTelephony.

[3] Nokia Online Documents: Symbian security model: http://www.forum.nokia.com/document/Cpp_Developers_Library/?content=GUID-232258EC-D3B4-4D72-B12B-FFC34F070B4B_GUID-644A092A-6999-46F8-A81F-363591CC0C03.html.

[4] Symbian China community discussion: http://bbs.dospy.com/viewthread.php?tid=2474219&extra=page%3D1%26amp%3Bfilter%3Dpoll&bbsid=147.

[5] Symbian China fans community: http://bbs.dospy.com/.

[6] Symbian SDK: http://www.forum.nokia.com/info/sw.nokia.com/id/05c63dfd-d6e9-4c0e-b185-d365e7001aeb/S60-SDK-0548-3.0-f.3.215f.zip.html.

[7] Symbian, "The Future of Your App": http://www.symbian.org/yourapp.

[8] Craig Heath, *Symbian OS Platform Security: Software Development Using the Symbian OS Security Architecture* (Wiley, 2006).

[9] Jerome H. Saltzer and Michael D. Schroeder, "The Protection of Information in Computer Systems," *Proceedings of the IEEE*, vol. 63, September 1975, pp. 1278–1308.

[10] Donald C. Latham, Department of Defense Trusted Computer System Evaluation Criteria, DoD 5200.28-STD, December 1985: http://csrc.nist.gov/publications/history/dod85.pdf.

[11] George Lawton, "Sexy Space Threat Comes to Mobile Phones," *Computing Now*, August 2009.

[12] Neal Leavitt, "Mobile Phones: The Next Frontier for Hackers," *Computer*, April 2005, pp. 20–23.

[13] Robert Lemos, "A Moving Target," *PC Magazine*, June 2006, p. 124.

[14] Ben Morris. *Platform Security and Symbian Signed: Foundation for a Secure Platform* (Symbian Developer Network, 2008): http://developer.symbian.com/main/downloads/papers/PlatSec_and_Symbian_Signed.pdf.

[15] Ben Morris and Ashlee Godwin, *A Guide to Symbian Signed*, 3rd edition (Symbian Software Ltd, 2008): http://developer.symbian.org/wiki/index.php/Complete_Guide_To_Symbian_Signed.

[16] Joe Odukoya, Elise Korolev, and Ashlee Godwin, *Platform Security for All* (Symbian Software Ltd, 2008): http://developer.symbian.com/main/documentation/books/books_files/pdf/Plat+Sec+FINAL+-+WEB.pdf.

[17] Mark Shackman, *Platform Security: A Technical Overview* (Symbian Developer Network, 2008): http://developer.symbian.com/main/downloads/papers/plat_sec_tech_overview/platform_security_a_technical_overview.pdf.

[18] Niu Xuelian and Ling Li, "Research and Improvement of the Symbian OS Kernel Platform Security Design," *Computer Engineering*, June 2006, pp. 194–196.

CARLOS MALTZAHN,
ESTEBAN MOLINA-ESTOLANO, AMANDEEP KHURANA, ALEX J. NELSON,
SCOTT A. BRANDT, AND SAGE WEIL

Ceph as a scalable alternative to the Hadoop Distributed File System

Carlos Maltzahn is an associate adjunct professor at the UC Santa Cruz Computer Science Department and associate director of the UCSC/Los Alamos Institute for Scalable Scientific Data Management. His current research interests include scalable file system data and metadata management, storage QoS, data management games, network intermediaries, information retrieval, and cooperation dynamics.

carlosm@soe.ucsc.edu

Esteban Molina-Estolano is a PhD student at UC Santa Cruz. His research interests include parallel file and storage systems, and their modeling and simulation. He received his BS from Harvey Mudd College.

eestolan@cs.ucsc.edu

Amandeep Khurana completed his master's in computer science from UC Santa Cruz, specializing in distributed systems. He has been dabbling with Hadoop and HBase and is now at Amazon, helping them scale their systems. Prior to this, he worked with Cisco on building scalable data integration frameworks.

akhurana@soe.ucsc.edu

Alex Nelson is a PhD student at UC Santa Cruz. His research interests include secure, large-scale, and long-term storage.

ajnelson@cs.ucsc.edu

Scott Brandt is a professor of computer science at the University of California, Santa Cruz, and director of the UCSC/Los Alamos Institute for Scalable Scientific Data Management (ISSDM) and the UCSC Systems Research Laboratory (SRL), in which Ceph was developed. His current research includes high-performance storage systems, real-time systems, and distributed system performance management.

scott@cs.ucsc.edu

Sage Weil built Ceph as part of his PhD research in storage systems at UC Santa Cruz. Prior to his graduate work, Sage helped found New Dream Network, the company behind Dreamhost Web hosting (dreamhost.com), which now supports a small team of Ceph developers.

sage@newdream.net

THE HADOOP DISTRIBUTED FILE System (HDFS) has a single metadata server that sets a hard limit on its maximum size. Ceph, a high-performance distributed file system under development since 2005 and now supported in Linux, bypasses the scaling limits of HDFS. We describe Ceph and its elements and provide instructions for installing a demonstration system that can be used with Hadoop.

Hadoop has become a hugely popular platform for large-scale data analysis. This popularity poses ever greater demands on the scalability and functionality of Hadoop and has revealed an important architectural limitation of its underlying file system: HDFS provides only one *name-node,* which has to store the entire file system namespace in main memory. This puts a hard limit on the amount of metadata, in particular the number of files, that HDFS can store. The single name-node limitation is well-recognized in the Hadoop user and developer community (see, for example, [8, 16]). Large clusters frequently run out of capacity at the name node to track new files even though there is plenty of storage capacity at the data nodes. The single name-node also creates a single point of failure and a potential performance bottleneck for workloads that require relatively large amounts of metadata manipulations, such as opening and closing files.

Ceph [11] is an object-based parallel file system with a number of features that make it an ideal storage system candidate for Hadoop:

- Ceph's *scalable metadata server* [14] can be distributed over hundreds of nodes while providing consistent, reliable, high-performance metadata service using dynamic subtree partitioning with near-linear scalability.
- Each file can specify its own *striping strategy* and object size. Flexible striping strategies and object sizes are important tuning parameters for Hadoop workloads [2, 6, 10].
- Data is stored on up to 10,000 nodes which export a single, *reliable object service* [13] with a flat namespace of object IDs, not unlike Amazon's Simple Storage Service (S3) [1]. Changes in the storage cluster size cause automatic (and fast) failure recovery and rebalancing of data with no interruption of service and minimal data movement, making Ceph suitable for very large deployments.
- The state of the entire storage cluster, includ-

ing data placement, failed nodes, and recovery state, has a very compact representation due to calculated data placement [12] as opposed to allocation tables, and is known in every part of Ceph. As in HDFS, Hadoop's scheduler can take advantage of this information to place mapping close to where the data resides.

- Ceph is an open source project (ceph.newdream.net) written in C++ and C that started as a PhD research project at UC Santa Cruz over four years ago and has been under heavy development ever since. A Hadoop module for integrating Ceph into Hadoop has been in development since release 0.12, and Hadoop can also access Ceph via its POSIX I/O interface, using ioctl calls for data location information.
- Since Ceph is designed to serve as a general-purpose file system (e.g., it provides a Linux kernel client so Ceph file systems can be mounted), if it supported Hadoop workloads well, it could also be a general solution to other storage needs.

In this article we describe the Ceph file system architecture, how to install Ceph on a 10-node cluster, and how to use Ceph with Hadoop.

Ceph

Ceph was designed to fulfill the following goals specified by three national laboratories (LLNL, LANL, and Sandia) back in 2005:

- Petabytes of data on one to thousands of hard drives
- TB/sec aggregate throughput on one to thousands of hard drives pumping out data as fast as they can
- Billions of files organized in one to thousands of files per directory
- File sizes that range from bytes to terabytes
- Metadata access times in μsecs
- High-performance direct access from thousands of clients to
  - different files in different directories
  - different files in the same directory
  - the same file
- Mid-performance local data access
- Wide-area general-purpose access

The challenges of such a file system are that it needs to be able to deal with huge files and directories, coordinate the activity of thousands of disks, provide parallel access to metadata on a massive scale, handle both scientific and general-purpose workloads, authenticate and encrypt at scale, and grow or shrink dynamically because of frequent device decommissioning, device failures, and cluster expansions. You can't just shut down the system because of a disk failure or shortage of space.

Ceph is an *object-based parallel file system* whose design is based on two key ideas. The first key idea is object-based storage, which splits the traditional file system architecture into a client component and a storage component. The storage component manages disk scheduling and layout locally, relieving clients and servers from low-level per-disk details and increasing scalability. This design allows clients to communicate with storage nodes via a high-level interface and manage data in terms of objects, which are chunks of data much larger than 512-byte blocks. The T10 standard of the SCSI Object Storage Device (OSD) command set [5] is an example of an object interface specification. Ceph uses and significantly extends the concept of OSDs. For all practical purposes, think of a Ceph OSD as a process that runs on a cluster node and uses a local file system to store data objects.

The second key idea in the Ceph design is the separation of data and

metadata. Management of data differs fundamentally from management of metadata: file data storage is trivially parallelizable and is limited primarily by the network infrastructure. Metadata management is much more complex, because hierarchical directory structures impose interdependencies (e.g., POSIX access permissions depend on parent directories) and the metadata server must maintain file system consistency. Metadata servers have to withstand heavy workloads: 30–80% of all file system operations involve metadata, so there are lots of transactions on lots of small metadata items following a variety of usage patterns. Good metadata performance is therefore critical to overall system performance. Popular files and directories are common, and concurrent access can overwhelm many schemes.

The three unique aspects of Ceph's design are:

- distributed metadata management in a separate metadata server (MDS) cluster that uses dynamic subtree partitioning to avoid metadata access hot spots and that is robust against non-byzantine failures;
- calculated pseudo-random data placement that allows for very compact state that can be shared easily throughout the system (CRUSH); and
- distributed object storage using a cluster of intelligent OSDs which forms a reliable object store that can act autonomously and intelligently (RADOS) (see Figure 1).

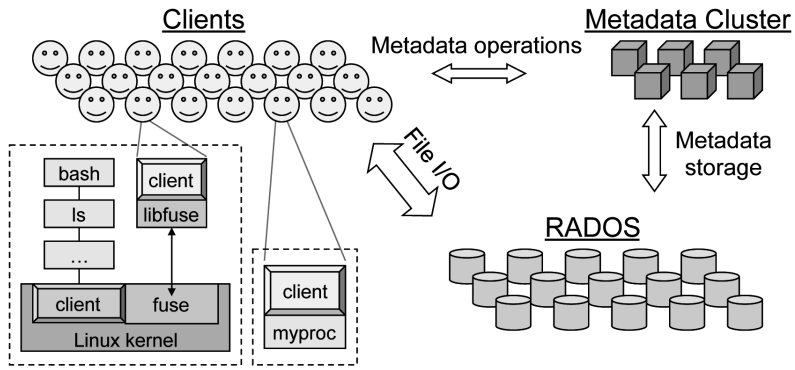


FIGURE 1: ARCHITECTURE. CEPH CONSISTS OF FOUR SUBSYSTEMS: (1) FILE SYSTEM CLIENTS, (2) DATA PLACEMENT USING A FORM OF CONSISTENT HASHING (CONTROLLED REPLICATION UNDER SCALABLE HASHING, OR CRUSH), (3) A CLUSTER OF METADATA SERVERS (MDS), AND (4) A RELIABLE AUTONOMIC, DISTRIBUTED OBJECT STORE (RADOS), WHICH INCLUDES THE MONITOR SERVICE AND OBJECT STORAGE DEVICES (OSDS).

The high-level interaction of these components is shown in Figure 2.

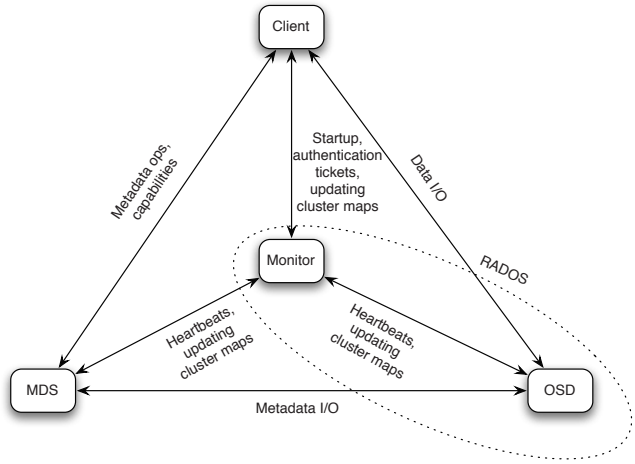


FIGURE 2: CEPH COMPONENT INTERACTIONS

CLIENT

A user or an application interacts with Ceph through the client component. The client exposes a POSIX file system interface. The interface also supports a subset of POSIX I/O extensions for high-performance computing, including the O_LAZY flag for the POSIX open command, which allows performance-conscious applications to manage their own consistency [15]. Ceph has a user-level client as well as a kernel client. The user-level client is either linked directly to the application or used via FUSE [9]. The kernel client is now available in the mainline Linux kernel (since 2.6.34) and allows the Ceph file system to be mounted.

For a simple example of how a client interacts with the rest of the Ceph system, consider a client opening a file /foo/bar for reading: the client sends an "open for read" message to the MDS. The MDS reads directory /foo from the appropriate OSDs (unless the directory is already cached in memory) and returns the capability for reading "/foo/bar" to the client. The client then calculates the name(s) and location(s) of the data object(s) for the file and reads the data from the corresponding OSD(s). Finally, the client closes the file by relinquishing the capability to the MDS. Remember that all these messages between these different components of Ceph are invisible to the application: the application simply issues POSIX open, read, and close commands.

The client is the only place in Ceph where metadata meets data: the capability returned by the MDS contains the inode number, the replication factor, and information about the *striping strategy* of a file, which can be file-specific and is set at file creation time. The striping strategy, the inode number, and an offset allow the client to derive the object identifier. A simple hash function maps the object identifier (OID) to a *placement group*, a group of OSDs that stores an object and all its replicas. There are a limited number of placement groups to create an upper bound on the number of OSDs that store replicas of objects stored on any given OSD. The higher that number, the higher the likelihood that a failure of multiple nodes will lead to data loss. If, for example, each OSD has replica relations to every other OSD, the failure of just three nodes in the entire cluster can wipe out data that is stored on all three replicas. Placement groups are mapped to OSDs by CRUSH, a consistent hashing function that takes as parameters (1) the placement group ID, (2) the replication factor, (3) the current cluster map (see section on RADOS, below), and (4) placement rules (see section on CRUSH, below). CRUSH then returns an ordered list of OSD IDs, one for each replica. The client picks the first OSD (the "primary") on the list as a location of the object.

METADATA SERVICE (MDS)

Ceph provides a cluster of metadata servers which continually load-balances itself using dynamic subtree partitioning [14]. The responsibility for managing the namespace hierarchy is adaptively and intelligently distributed among tens or even hundreds of metadata servers. The key to the MDS cluster's adaptability is that Ceph metadata items are very small and can be moved around quickly. This would be impossible if Ceph were to use the approach of many file systems, in which file byte streams are mapped to disk blocks using allocation tables.

Each directory is stored as an object. Inodes are embedded in directories and stored with the corresponding directory entry (file name). This organization optimizes the common access pattern of listing a directory and retrieving metadata for each file. Embedded inodes complicate the management of hard links, but it turns out that hard links are rare and exhibit useful locality properties (most hard links are referring to entries within the same or parallel directory).

To enable failure recovery, the MDS journals metadata updates to OSDs. The journals are striped across large objects, which leads to efficient, sequential writes. All in-memory metadata that is updated and journaled but not written as an updated directory object is marked dirty and pinned to main memory until the corresponding entries in the journal must be trimmed from the tail of the journal. Journals are allowed to grow very large (hundreds of megabytes). By the time updates need to be trimmed, many metadata updates have been invalidated by subsequent updates and can be consolidated into a small number of directory updates. With this approach the MDS acts as an intelligent metadata cache that turns random updates of small metadata items into efficient data I/O.

The file system's namespace is partitioned across the cluster of MDS nodes along directory subtrees, as shown in Figure 3. This *dynamic subtree partitioning* ensures that the distribution of subtrees is as coarse as possible to preserve locality (which is often aligned along subtrees) and that it is continually adapted to keep the MDS workload-balanced. Subtrees are migrated between MDSes as workload changes. To alleviate hot spots, heavily read directories are replicated on multiple MDSes so that the number of clients knowing about a particular replica can be bounded. Large or heavily written directories are fragmented for load distribution and storage. Rebalancing of the MDS at even extreme workload changes is usually accomplished within a few seconds. Clients are notified of relevant partition updates whenever they communicate with the MDS.

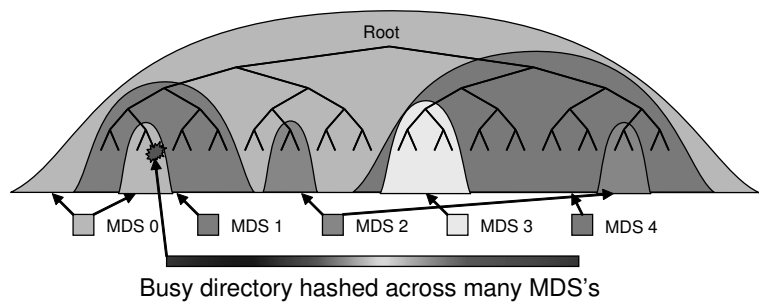


FIGURE 3: DYNAMIC SUBTREE PARTITIONING

RELIABLE AUTONOMIC DISTRIBUTED OBJECT STORAGE (RADOS)

Data in Ceph is stored in a distributed object store that can scale up from tens to hundreds of thousands of object storage devices (OSDs). RADOS can

also be used as a stand-alone system. In fact, the Ceph distribution includes a simple gateway, a fastcgi daemon, that implements Amazon's S3 API [1] for RADOS.

Initiators—in Ceph's case, clients and the MDS—see the object storage cluster as a single logical object store with a flat namespace, called RADOS. RADOS achieves its scalability by significantly expanding the T10 SCSI OSD concept: while T10 OSDs only respond to read and write, Ceph OSDs actively collaborate with peers for failure detection, data replication, and recovery.

To direct requests, initiators use a *cluster map* which they receive from RADOS. The very compact cluster map contains information about participating OSDs, their status, and the CRUSH mapping function (see CRUSH section, below). The master copy of the cluster map is managed by a distributed *monitor service* which consists of a set of monitors that maintain consistency using the Paxos algorithm [7]. When an OSD alerts the monitor service of a new failure or cluster expansion, the monitor replies with an updated version of the cluster map, which is then lazily propagated throughout the cluster. The total size of the cluster map is in the range of megabytes (depending on the size of the cluster). Propagation overhead is reduced by only communicating deltas between cluster map versions and combining them with existing inter-OSD messages.

Unlike other parallel file systems, replication is managed by OSDs instead of clients, which shifts replication bandwidth overhead to the OSD cluster, simplifies the client protocol, and provides fully consistent semantics in mixed read/write workloads. RADOS manages the replication of data using a variant of primary-copy replication. As mentioned in the Client section, above, replicas are stored in placement groups. Each placement group includes a primary OSD which serializes all requests to the placement group. In case of writes, the primary forwards the request to the other OSDs of the placement group. The primary applies the write locally after the other replicas have applied theirs. Only then does the client receive an acknowledgment from the primary.

Writes are applied in two phases, as shown in Figure 4. This two-phase approach separates writing for the purpose of sharing with other clients from writing for the purpose of durability and makes sharing data very fast. The client receives an *ack* from the primary after the data has been replicated in memory on all of the replicas. At this point, the client still has the data in its buffer cache. Once the data is committed to the disk on all replicas, the primary sends a *commit* to the client, confirming that the data is now durable. The client may then delete the data from its buffer cache.

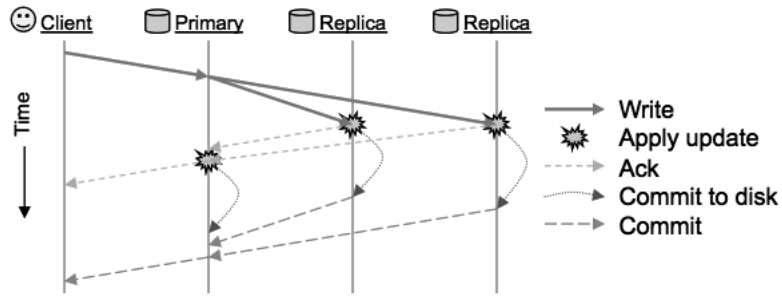


FIGURE 4: WRITE SEMANTICS

By default, OSDs use Btrfs [3] as their local file system (but ext3 works too). Data is written asynchronously using copy-on-write, so that unsuccessful write operations can be fully rolled back. Each OSD maintains a log of

object versions for each placement group in which it participates. If one OSD fails, the remaining OSDs can quickly identify stale or missing objects by comparing these logs; OSDs which intermittently fail can quickly recover.

An OSD monitors the state of other OSDs in the same placement groups using heartbeat messages, which are usually piggybacked on replication traffic. OSDs that discover an unresponsive OSD alert the monitor service and receive a new cluster map that marks the unresponsive OSD as *down*. Other OSDs temporarily take over any primary roles the unresponsive OSD might have had. If within a configured time the OSD does not come back up, the monitor service issues another cluster map that marks it as *out*, some other OSD is elected to be the new primary, and the re-establishment of the full complement of replicas begins.

In summary, Ceph's failure detection and recovery are fully distributed and the monitor service is only used to update the master copy of the cluster map. The monitor service does not broadcast these updates to the cluster. Instead, the cluster map updates are communicated by OSDs using epidemic-style propagation that has bounded overhead. This procedure is used to respond to all cluster map updates, whether due to OSD failure, cluster contraction, or expansion. OSDs always collaborate to realize the data distribution specified in the latest cluster map while preserving consistency of read/write access.

DATA DISTRIBUTION WITH CRUSH

The small size of metadata items in the MDS and the compactness of cluster maps in RADOS are enabled by CRUSH (Controlled Replication Under Scalable Hashing) [12]. Ceph uses this hash function to calculate the placement of data instead of using allocation tables, which can grow very large and unwieldy. CRUSH is part of the cluster map and behaves like a consistent hashing function in that failure, removal, and addition of nodes result in near-minimal object migration to re-establish near-uniform distribution.

As mentioned in the Client section, CRUSH maps a placement group ID to an ordered list of OSDs, using a hierarchically structured cluster map and placement rules as additional input. The length of the list of OSDs depends on the replication factor. The first available OSD in the list is the primary. Any list output by CRUSH meets the constraints specified by *placement rules*. These rules are defined over cluster maps which an administrator can hierarchically structure according to, say, failure domains such as racks or shelves (since they often share the same electrical circuit or power supply). Thus, placement rules can prevent two replicas from being placed in the same failure domain. This awareness of failure domains during data placement is critically important for the overall data safety of very large storage systems where correlated failures are common.

Installing and Configuring Ceph

A Ceph installation requires at least one monitor, one OSD, and one metadata server. These may all be on the same node, although the use of additional nodes allows for greater performance, robustness, and capacity. Here we give a configuration walk-through for a multi-node Ceph installation. For a single-node installation, adapt this example to start up one of each daemon, all on the same node.

Our hypothetical cluster has 10 nodes, with hostnames node0 through node9. Each node has a raw, unused partition at /dev/sdb1 and another at /dev/sdb2. We will configure the nodes as follows:

- Three monitors, on nodes 0–2
- Three MDSes, on nodes 0–2
- Eight OSDs, on nodes 2–9

Less hardware could be used for a multi-node cluster. There should always be an odd number of monitors. For robustness, two MDSes (active and standby) are sufficient. For data replication, two OSDs would be sufficient.

We use a file system setup user named `setupuser`, whose public key grants access to `setupuser` on all nodes and root accounts on the storage nodes, due to the need to format partitions. This key should be password-locked and used with SSH-Agent or an equivalent session manager—it will only be used to set up and tear down the Ceph cluster. However, if you use the same account and key for later Hadoop setup, this key cannot have a password due to Hadoop's passwordless SSH requirement.

These instructions are sufficient to set up and run a Ceph cluster. If you want further documentation, see the Ceph wiki (http://ceph.newdream.net/wiki/), and src/sample.ceph.conf within the downloaded Ceph server code. Alternate packaging is available for Debian and Fedora Core from Ceph's wiki; also for Fedora, there is a `ceph.spec` file in the source. This tutorial works with Ceph 0.20.1 from source and Linux kernel 2.6.34.

INSTALLING CEPH DAEMONS

First, we install the prerequisite packages on each node. For Fedora Core 12 or Ubuntu 9.10, run these commands:

```
#Ubuntu 9.10
apt-get install autoconf automake libtool libboost-dev libedit-dev libssl-dev

#Fedora Core 12
yum install rpm-build fuse-devel libtool libtool-ltdl-devel boost-devel
libedit-devel openssl-devel gcc-c++ btrfs-progs
```

Second, we configure and build the source. For simplicity, we will assume that you are building the source in a shared NFS directory, visible on all nodes. We will run the daemons from <ceph directory>/src/ in the build tree without installation. We will also create the ceph.conf configuration file in the source tree.

To avoid relying on NFS, you can `make install` Ceph on each node, or install Ceph using the Debian packages (see http://ceph.newdream.net/wiki/Debian). In this case, you will also need to deploy ceph.conf to /etc/ceph/ceph.conf on each node, instead of keeping it in the source tree.

```
cd <ceph directory>
CXXFLAGS="-g" ./configure
make
```

The cluster initialization script detects when these are run out of a local directory instead of being fully installed; thus there is no need to set the Ceph executables' directory in a configuration file. If you do want to have a local installation on each node, run `sudo make install` after `make`. This places the Ceph binaries into /usr/local/bin and /usr/local/sbin as necessary.

CONFIGURING

Ceph's configuration is stored in a single file, `ceph.conf`, which is identical across all nodes. There is a section in the configuration file for each daemon and a common section for each type of daemon. For instance, configuration

parameters common across all monitors are placed in the [mon] section, and per-monitor settings are placed in [mon0], [mon1], and so on.

For logging and other local data purposes, create a local directory on all nodes, /data, readable and writeable as setupuser. Then create src/ceph.conf as follows.

```
[global]
   user = setupuser
   ; where the mdses and osds keep their secret encryption keys
   keyring = /data/keyring.$name

; monitors
[mon]
   ;Directory for monitor files
   mon data = /data/mon$id

[mon0]
   host = node0
   mon addr = 192.168.0.100:6789
[mon1]
   host = node1
   mon addr = 192.168.0.101:6789
[mon2]
   host = node2
   mon addr = 192.168.0.102:6789

; metadata servers
[mds]

[mds0]
   host = node0
[mds1]
   host = node1
[mds2]
   host = node2

; OSDs
[osd]
   ; osd data is where the btrfs volume will be mounted;
   ; it will be created if absent
   osd data = /data/osd$id
   ; osd journal is the regular file or device to be used for journaling
   osd journal = /dev/sdb2
   ; The 'btrfs devs' partition will be formatted as btrfs.
   btrfs devs = /dev/sdb1
   host = node$id

[osd2]
[osd3]
[osd4]
[ . . .}
[osd9]
```

Observe that by setting host = node$id, and by instantiating OSDs 2 through 9 (skipping 0 and 1), we can exploit the sequential hostnames and avoid explicitly setting the hostname for every OSD.

OSD journaling is more efficiently done on a raw partition than on a regular file. If you use a raw partition, setupuser should be a member of the disk group on the OSDs, so that it has write access to the device.

Next, create and start the file system as follows. The -allhosts flag makes

each command work on all the nodes specified in ceph.conf, via SSH; therefore, you should have passwordless root SSH configured for each OSD. The root account's authorized_keys on each OSD should have the public key of setupuser. Once the public key is distributed, run the following commands:

```
./mkcephfs -c ceph.conf --allhosts --mkbtrfs
./init-ceph -c ceph.conf --allhosts start
```

-mkbtrfs formats the btrfs devs to Btrfs; if you wish to use another file system or regular directory instead, ensure that the resulting or underlying file system has extended attributes enabled (user_xattr for ext3).

Should you need to stop (./init-ceph -c ceph.conf -allhosts stop) and restart the Ceph services, the log directories from prior server sessions may cause a hanging failure on mount. Clearing them prevents this issue.

Mounting requires kernel support. Ceph support is in the Linux kernel as of version 2.6.34; for earlier kernel versions, see http://ceph.newdream.net/wiki/Building_kernel_client for instructions on building the Ceph kernel module. Mount as follows, using the IP address of one of the monitors:

```
mount -t ceph 192.168.0.100:/ /mnt/ceph
```

At this point, Ceph is usable like any other part of your local file system.

Using Hadoop with Ceph

There are two ways to use Ceph as the file system for Hadoop: (1) mounting Ceph as in the end of the previous section and using it as a local file system in Hadoop (file:///mnt/ceph); (2) patching the Hadoop Core with the patch available in the HADOOP-6253 JIRA [4]. This uses the user-level client of Ceph.

When starting up Hadoop, do not use bin/start-all.sh, as this will launch HDFS. Start up only the daemons you need (e.g., bin/start-mapred.sh for MapReduce).

RUNNING HADOOP ON CEPH THROUGH THE KERNEL INTERFACE

You can run Hadoop on Ceph via POSIX using Ceph's kernel interface. Using Hadoop on top of Ceph instead of HDFS requires two configuration tweaks in conf/core-site.xml (conf/hadoop-site.xml in 0.20.2):

```
<configuration>

  <property>
    <!--
    Note that in release 0.20.2 this name is fs.default.name;
    afterwards this is fs.defaultFS.
     -->
    <name>fs.defaultFS</name>
    <value>file:///mnt/ceph</value>
  </property>
</configuration>

<property>
  <name>hadoop.sharedtmp.dir</name>
  <value>/mnt/ceph/hadoop-tmp/hadoop-${user.name}</value>
</property>
```

Using Ceph through the above option will use Hadoop's Raw File System interface to communicate with Ceph as a local file system. There are optimizations that can be added to this interface in order to better leverage Ceph's performance. For example, locality information is not exposed to the Raw

File System interface; therefore Hadoop will not be able to schedule tasks close to the physical location of the data. These optimizations are being put into the HADOOP-6779 JIRA [17].

RUNNING HADOOP ON CEPH VIA JNI CODE

There is a second way to run Hadoop on Ceph which does not require the POSIX kernel interface—and hence does not require updating your cluster's kernels. You can instead use JNI code to connect Hadoop to Ceph in user-space. This feature is not yet in a Hadoop release; to use it, check out the Hadoop development trunk, and apply the patch from the HADOOP-6253 JIRA [4]. (See http://ceph.newdream.net/wiki/Hadoop_File system for more details.) Building Hadoop from the trunk source is outside the scope of this article.

Then add the following to conf/core-site.xml:

```
<property>
  <name>fs.defaultFS</name>
  <!--ip address of the mds here-->
  <value>ceph://<MDS_IPAddress:port></value>
</property>

<property>
  <name>fs.ceph.monAddr</name>
  <value><monitor_server:port></value>
  <description>The location of the Ceph monitor to connect to.  This
  should be an IP address, not a domain-based web address.</description>
</property>

<property>
  <name>fs.ceph.libDir</name>
  <value>/usr/local/lib</value>
  <description>The folder holding libceph and libhadoopceph</description>
</property>
```

After adding these configurations, Hadoop is ready to use with Ceph as its data store.

Summary

Since the Ceph kernel client was pulled into Linux kernel 2.6.34, interest in Ceph has greatly increased. Ceph is currently the only open source (LGPL licensed) parallel file system that offers a distributed metadata service that is linearly scalable to at least 128 metadata service nodes, supports the POSIX I/O API and semantics, and is able to expand and contract with low overhead without interrupting service.

The last point allows Ceph to be deployed in virtual environments such as Amazon's EC2 cloud service, where frequent and significant cluster size changes are the norm. Overall Ceph addresses a number of shortcomings of HDFS, i.e., HDFS's limited name-node scalability, its heartbeat overhead, and its highly specialized file access semantics.

As we write this, Ceph is still experimental and not yet ready for production environments. Sage Weil, Yehuda Sadeh, and Gregory Farnum are working full-time on making Ceph production-ready, with new releases coming out every 2 to 4 weeks. We hope this article will encourage people to participate in this effort by trying out Ceph with workloads they care about and reporting any bugs, performance problems, or bug/performance fixes.

REFERENCES

[1] Amazon, Simple Storage Service—Developer Guide (API Version 2006-03-01): docs.amazonwebservices.com/AmazonS3/2006-03-01/.

[2] Rajagopal Ananthanarayanan, Karan Gupta, Prashant Pandey, Himabindu Pucha, Prasenjit Sarkar, Mansi Shah, and Renu Tewari, "Cloud Analytics: Do We Really Need to Reinvent the Storage Stack?" USENIX HotCloud '09, San Diego, CA, June 15, 2009.

[3] Valerie Aurora, "A Short History of Btrfs," LWN.net, July 22, 2009: http://lwn.net/Articles/342892/.

[4] Gregory Farnum, "Add a Ceph File System Interface," ASF JIRA, May 2010: https://issues.apache.org/jira/browse/HADOOP-6253.

[5] International Committee for Information Technology Standards, SCSI Object-Based Storage Device Commands - 3 (OSD-3), project proposal for a new INCITS Standard T10/08-331r1, International Committee for Information Technology Standards, September 11, 2008.

[6] Hadoop Project, Hadoop Cluster Setup: hadoop.apache.org/core/docs/current/cluster_setup.html.

[7] Leslie Lamport, "The Part-time Parliament," *ACM Transactions on Computer Systems*, vol. 16, no. 2, 1998, pp. 133–169.

[8] Konstantin V. Shvachko, "HDFS Scalability: The Limits to Growth," *;login:*, vol. 35, no. 2, 2010.

[9] Miklos Szeredi, "File System in User Space," 2006: http://fuse.sourceforge.net.

[10] Wittawat Tantisiriroj, Swapnil Patil, and Garth Gibson, "Data-intensive File Systems for Internet Services: A Rose by any Other Name..." Technical Report CMU-PDL-08-114, Parallel Data Laboratory, CMU, Pittsburgh, PA, October 2008.

[11] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D.E. Long, and Carlos Maltzahn, "Ceph: A Scalable, High-Performance Distributed File System," *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)*, Seattle, WA, November 2006.

[12] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, and Carlos Maltzahn. "CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data," *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (SC '06)*, Tampa, FL, November 2006.

[13] Sage A. Weil, Andrew Leung, Scott A. Brandt, and Carlos Maltzahn. "Rados: A Fast, Scalable, and Reliable Storage Service for Petabyte-Scale Storage Clusters," *Proceedings of the 2007 ACM Petascale Data Storage Workshop (PDSW '07)*, Reno, NV, November 2007.

[14] Sage A. Weil, Kristal T. Pollack, Scott A. Brandt, and Ethan L. Miller, "Dynamic Metadata Management for Petabyte-Scale File Systems," *Proceedings of the 2004 ACM/IEEE Conference on Supercomputing (SC '04)*, Pittsburgh, PA, November 2004.

[15] Brent Welch, "POSIX I/O Extensions for HPC," *Proceedings of the 4th USENIX Conference on File and Storage Technologies (FAST)*, December 2005.

[16] Tom White, "The Small Files Problem," February 2, 2009: http://www.cloudera.com/blog/2009/02/02/the-small-files-problem/.

[17] Alex Nelso, "Support for Ceph Kernel Client," ASF JIRA: https://issues.apache.org/jira/browse/HADOOP-6779, May 2010.

MICHAEL PIATEK AND
ARVIND KRISHNAMURTHY

improving the performance and robustness of P2P live streaming with Contracts

Michael Piatek is a graduate student at the University of Washington. After spending his undergraduate years working on differential geometry, his research interests now include incentive design in distributed systems, network measurement, and large-scale systems building.

piatek@cs.washington.edu

Arvind Krishnamurthy is an associate research professor at the University of Washington, Seattle. His research interests are primarily at the boundary between the theory and practice of distributed systems, in the topics of peer-to-peer systems, network measurements, and network protocols. His recent projects include iPlane, a distributed service that performs network measurements, BitTyrant, an optimized content distribution system, Botlab, a pervasive monitoring infrastructure for Botnets, and OneSwarm, a privacy-preserving peer-to-peer system.

arvind@cs.washington.edu

THE INCREASING POPULARITY OF ONLINE streaming video is defining a major shift in the Internet's workload. To cope with the demands of distributing video, many popular services take a peer-to-peer approach, relying on users to redistribute video data after receiving it. PPLive is one such system, used daily for live streaming by millions of people worldwide. As with any P2P design, the scalability of PPLive depends on users contributing capacity to the system. But, currently, these contributions are neither verified nor rewarded. This article describes Contracts, an extension of the PPLive protocol, that improves performance by recognizing and rewarding users who contribute. For example, in our experiments, the fraction of PPLive clients using Contracts experiencing loss-free playback is more than four times that of native PPLive.

Live Streaming

The Internet is rapidly becoming one of the main distribution channels for video. Hulu, Netflix, and the BBC's iPlayer are just a few examples of well-known video streaming services. Increasingly, video distribution services support live content as well, e.g., sporting events.

The popularity of video streaming creates scalability challenges. Publishers would prefer that playback start quickly, continue without interruption, and have high quality. But achieving these goals is difficult given the realities of transient congestion, flash crowds, and network bottlenecks at the broadcast source.

To achieve scalability, many Internet video services use peer-to-peer (P2P) content distribution. P2P designs rely on users who have already received part of the video stream to redistribute that data to others. This decreases load on the broadcaster and provides more redundant data sources, increasing robustness.

PPLive is one of the most widely deployed live streaming services on the Internet today, serving more than 20 million active users spread across the globe. As with any P2P design, PPLive's scalability

depends on its users contributing their capacity by redistributing video data. But the current PPLive design neither verifies nor rewards contributions. All users are treated equally by the protocol, even if they contribute nothing whatsoever.

In this article, we examine how best to provide incentives for users to contribute resources to P2P live streaming systems. Our goal is to structure the system so that rational users will *want* to contribute resources because doing so will improve their performance. We use PPLive as a concrete example; it is one of the most popular P2P live streaming systems available today, and its developers were willing to work with us to provide modified binaries and workload data.

The goal of motivating contributions is common to many P2P designs, notably the popular BitTorrent file-sharing protocol. BitTorrent and PPLive take a similar approach to distribution: data is broken up into blocks and distributed by a source, with peers redistributing individual blocks after receiving them. Unlike PPLive, however, BitTorrent incorporates contribution incentives into its design. BitTorrent's policy is "tit-for-tat"—clients *reciprocate* by providing data to peers that give data in return. As a result, users interested in faster downloads should increase their upload contribution.

The similarity of BitTorrent and PPLive raises the question: *will tit-for-tat work for live streaming?* While intuitive and simple, we find that tit-for-tat is much less effective for live streaming than file sharing. We consider three challenges to applying tit-for-tat that motivate the design of our protocol, Contracts.

CAPACITY HETEROGENEITY

The picture quality of a video stream is determined by its data rate. For a P2P system, choosing the data rate depends on the total capacity of all the users in the system, including any seed capacity provided by the source. The maximum rate is the average capacity—beyond the average, bandwidth demand exceeds supply, and a rate lower than the average wastes capacity that could be used to increase quality.

Although streaming at the average capacity maximizes quality, doing so is incompatible with incentive strategies based on *strict reciprocation*; i.e., users trade one block sent for one block received. Strict reciprocation creates a strong incentive to contribute—to view the stream, users must contribute as much as they receive. But such a policy would exclude users with a capacity less than the average. In other words, for strict tit-for-tat to work well in live streaming systems, all users should have roughly the same capacity.

In practice, a defining feature of P2P workloads is that users' upload capacities vary significantly, and PPLive is no exception. Capacity measurements of more than 90,000 clients show that the top 10% of PPLive users contribute 58% of total capacity. This yields a discouraging trade-off. On one hand, streaming at the average rate maximizes quality, but would exclude 86% of PPLive clients when insisting on strict reciprocation. Alternatively, supporting 95% of users reduces capacity utilization to just 15%. In short, when applying strict reciprocation to live streaming, we can have high quality or robust incentives, but not both.

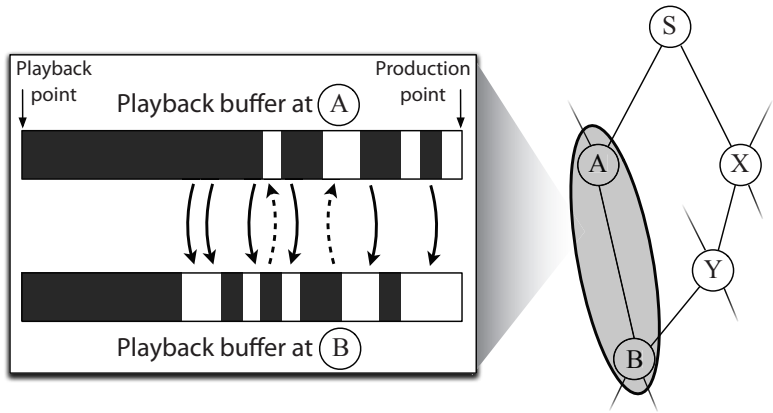


FIGURE 1: BLOCK TRADING OPPORTUNITIES IN A LIVE STREAMING MESH. DISTANT CLIENTS HAVE FEW OPPORTUNITIES FOR RECIPROCATION WITH PEERS CLOSER TO THE SOURCE.

LIMITED TRADING OPPORTUNITIES

An alternative to strict reciprocation is to allow imbalanced exchange. Rather than insisting on one block received for one block sent, imbalanced reciprocation schemes simply prioritize peers that contribute the most. But imbalanced exchange still depends on trading opportunities arising between peers—i.e., two peers exchanging blocks of mutual interest.

Unfortunately, live streaming provides clients with few opportunities for mutually beneficial trading between peers. The key property is that unlike bulk data distribution, where blocks have roughly equal value over time and among clients, *the value of blocks in live streaming varies over time and client*. A block has little value if it is received after the playback point at a client. Thus, the data useful to an individual client is limited to a narrow range between the production point and the local playback point.

This effect is shown in Figure 1. In this case, a snippet of the overlay mesh is shown. A source S sends blocks to directly connected peers A and X, who forward it to other peers in turn. Consider the trading opportunities between A and B. A is directly connected to the source—it receives most data immediately after it is produced. B is more distant, receiving data only after it has been forwarded by others. This reduces the trading opportunities between A and B, since virtually all of B's data blocks have already been received by A.

This example illustrates a general property: peers close to the source enjoy a near monopoly on new blocks, creating a trade imbalance that puts more distant peers at a perpetual disadvantage under any incentive scheme based on pairwise reciprocation between peers.

NO COMPELLING REWARD

The final obstacle we consider is the lack of a *compelling reward* for increasing contribution in live streaming. For bulk data distribution, the incentive to increase upload rate is a corresponding increase in download rate. But, live streaming is *inelastic*. To watch the stream, each user needs to download video data at the production rate. Even a small loss rate can quickly degrade the quality of a live video stream. On the other hand, neither can we increase download speeds to reward users—for live streams, additional video data has not yet been produced.

Streaming Incentives with Contracts

The challenges of using reciprocation lead us to take a different approach to providing contribution incentives in live streaming. Our scheme, Contracts, is based on two key design choices, which we summarize here. More details are available in our paper [1].

Contracts evaluates contributions according to both their *amount* and *effectiveness*. As in any P2P system, PPLive benefits from users contributing as much as possible. But, in live streaming, contribution alone is not sufficient. Because data blocks must arrive in time to meet playback deadlines, Contracts prioritizes requests from peers with the greatest capacity—i.e., the peers that can replicate new data most quickly.

In Contracts, evaluating contributions and effectiveness must be *verifiable*. While we could rely on honest reporting of these values, a strategic client could easily misrepresent its contributions to game the system. To prevent this, each Contracts client bases its calculations of a peer's value on cryptographically strong *receipts* of contribution. Each receipt acknowledges that one user has sent some data blocks to another. Receipts are gossiped among peers, allowing clients to evaluate nearby peers in the mesh.

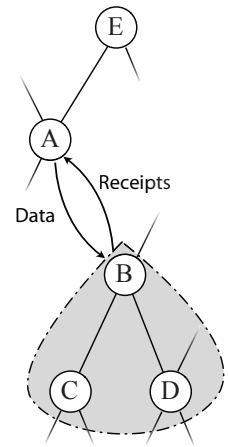


FIGURE 2: EVALUATING PEERS IN CONTRACTS. CLIENT *A* RECEIVES CRYPTOGRAPHIC RECEIPTS FROM *B* ACKNOWLEDGING DATA TRANSFER AND FORWARDS THESE TO *E* TO DEMONSTRATE ITS CONTRIBUTION AMOUNT. TO DEMONSTRATE EFFECTIVENESS, *E* ALSO INCLUDES RECEIPTS FROM ITS ONE HOP NEIGHBORHOOD OF PEERS (SHADED).

As an example of how Contracts evaluates contributions, consider Figure 2. In this case, A is being evaluated at E, and E uses receipts from peers in the shaded region to perform its calculation. As A sends data to B, it receives receipts attesting to its contributions. A forwards these receipts to E. Receipts from B to A allow E to compute the *amount* of A's contribution. To compute *effectiveness*, E needs receipts from both A and B. Receipts from C, D to B show that A has made *effective* contributions to a peer (B) that replicated the data to others. In general, A would forward receipts from all of its peers (and peers of peers) to E. Contracts restricts the propagation of receipts to one hop, however, to limit the overhead of the protocol.

In addition to prioritizing service, receipts are used to update the mesh topology. Contracts restructures connections to move high-capacity peers toward the source, promoting efficiency. In Figure 2, for example, suppose peer E is the closest to the broadcaster, and B has higher capacity than A.

During its evaluation of peers, E can recognize the mismatch while inspecting B's receipts and connect to B directly. This moves B closer to the source.

Performance and Incentives

Our evaluation of Contracts shows two main results. (1) PPLive with Contracts significantly outperforms both unmodified PPLive and PPLive modified to support tit-for-tat (TFT). (2) Contracts provides our intended contribution incentives; when the system is bandwidth constrained, increasing contribution improves performance. PPLive has adopted some of our techniques in their production code, and we continue to work toward full support in the public client. In this section, we report performance measurements of our Contracts PPLive prototype.

PERFORMANCE

We define performance as the fraction of data blocks received by their playback deadlines, and compare the performance of PPLive and PPLive using Contracts. We conducted a test broadcast of 100 users with arrivals and departures and upload capacities based on measurements of PPLive users. Crucially, in order to provide a meaningful comparison, we use a video data rate which exercises capacity constraints. (It would be easy to achieve good performance by simply over-provisioning the system substantially.)

We find that Contracts significantly improves performance relative to unmodified PPLive: 62% of Contracts clients experience loss-free playback compared with just 13% when using unmodified PPLive. In other words, the fraction of PPLive/Contracts clients experiencing loss-free playback is more than four times that of unmodified PPLive.

INCENTIVES

Contracts rewards contribution with increased robustness. We evaluate this by comparing the performance of PPLive using Contracts with that of PPLive using tit-for-tat. In both cases, the system is bandwidth constrained.

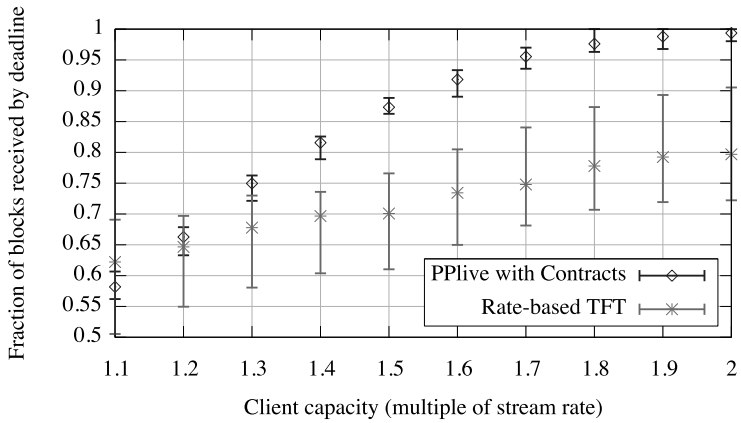


FIGURE 3: DELIVERY RATE AS A FUNCTION OF CONTRIBUTION. CONTRACTS PROVIDES LARGER AND MORE CONSISTENT REWARDS FOR INCREASING CONTRIBUTIONS.

In Figure 3, averages are shown with error bars giving the full range of block delivery rates for clients with a given capacity. While tit-for-tat does provide some correlation between contribution and performance, the amount of

improvement varies significantly because tit-for-tat does not update the topology. In contrast, Contracts combines both topology updates and priority service for contributors to provide a consistent improvement in performance, strengthening incentives.

Summary

P2P technology has emerged as a powerful technique for achieving scalability in live streaming systems. But scalability depends on users contributing their capacity, and in many systems, contributions are neither verified nor rewarded. The unique features of P2P live streaming limit the effectiveness of widely used incentive strategies based on reciprocation, e.g., tit-for-tat. Contracts provides an alternative: a new incentive strategy that rewards contribution with quality of service by evolving the overlay topology. Experiments using our prototype PPLive implementation show that Contracts both improves performance relative to PPLive and strengthens contribution incentives relative to existing approaches.

REFERENCES

[1] Michael Piatek, Arvind Krishnamurthy, Arun Venkataramani, Richard Yang, David Zhang, and Alexander Jaffe, "Contracts: Practical Contribution Incentives for P2P Live Streaming," *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI '10)*, 2010.

DAVID N. BLANK-EDELMAN

practical Perl tools: random acts of kindness

David N. Blank-Edelman is the director of technology at the Northeastern University College of Computer and Information Science and the author of the O'Reilly book *Automating System Administration with Perl* (the second edition of the Otter book), available at purveyors of fine dead trees everywhere. He has spent the past 24+ years as a system/network administrator in large multi-platform environments, including Brandeis University, Cambridge Technology Group, and the MIT Media Laboratory. He was the program chair of the LISA '05 conference and one of the LISA '06 Invited Talks co-chairs. David is honored to be 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

IF YOU SUBSCRIBE TO THE NOTION THAT it is a good thing to practice random acts of kindness, how exactly do you ensure they are random? In this column, we're going to take a pseudo-random walk through a number of ways people interact with randomness using Perl and various Perl modules. You'll encounter randomness in many application domains, including cryptography, data protection, software testing, Web development, voting, games, statistics, to name just a few. Rather than focusing on any one domain for use, this column will point at some tools to make your use of randomness easier.

Perl's Built-in Functions

The best place to start talking about randomness in Perl is with its two random number–related internal functions: rand() and srand(). The former provides you with access to your operating system's random number generator (which is either good or bad, depending on your operating system). By default, it returns "a random fractional number greater than or equal to 0 and less than the value of an optional argument." You'll often see code that looks like this:

```
print int(rand(20));        # prints a random integer
                            from 0 to 19
```

Some random number generation algorithms produce "more random" numbers than others. Which algorithm your operating system uses isn't always obvious or documented, hence my comment above. The one thing you can say about all of them is that in order to get the best possible (or even correct) use of rand(), you have to first wisely choose an initial value to "seed" it with. This is where the srand() function comes into play.

The srand() function accepts a seed value that rand() will use. If you give srand() the same value each time, rand() will generate the same "random" numbers each time. That repeatable quality in a random number generator may seem a bit strange, but we'll discover at least one use for it later in this column.

rand() calls srand() by default the first time it is called. srand() tries to use a decent default seed value "based on time of day, process ID, and memory allocation, or the /dev/urandom device if available," but you still see professionally paranoid

programmers calling this function when they want to be extra careful. We'll see one module in the very next section that some people use to super-duper-secure seed their random number generator.

Better Generators

If you are not satisfied with using rand() to access the OS's built-in random number generator, Perl offers a ton of alternatives. I should note that I'm not a mathematician (nor do I play one on TV), so I am not qualified to recommend one random-number-generator algorithm/module over another. I can point to a few that seem popular and talk a good game, but if you are going to need to use one because you have a serious need for the most "secure" RNG (random number generator) available, best to talk to someone more serious about this stuff than I am. Three modules in this vein are:

1. BSD::arc4random—to make use of the same RNG algorithm used by OpenBSD and others
2. Math::Random::MT (and related modules)—to use the Mersenne Twister RNG (though MT RNGs produce "high-quality" random numbers they are known not to be suitable for crypto uses)
3. Math::Random::ISAAC—to use the ISAAC RNG (still apparently crypto-safe)

Let's look at how to use the last two just so you get a feel for how they work.

The easiest way to use a Mersenne Twister RNG is through the Math::Random::MT::Auto module (there is a Math::Random::MT module, but this one has the added feature of making it easier to seed the RNG from a number of sources). To use Math::Random::MT::Auto, your code can be as simple as:

```
use Math::Random::MT::Auto 'rand';
# Perl's built-in overridden with an MT
```

This one line effectively substitutes the built-in rand() function with one backed by the MT algorithm. If that sort of overloading magic gives you the heebie-jeebies, you can be more explicit in how it gets used:

```
use Math::Random::MT::Auto;
my $rng = Math::Random::MT::Auto->new();
print $rng->irand(20)     # mimics the rand() example above
                          # use ->rand() instead for a rand() clone
```

Math::Random::MT::Auto also offers bonus functions like shuffle() to shuffle arrays in an MT-inspired fashion.

Math::Random::ISAAC (and its accompanying fast version, Math::Random::ISAAC::XS) is similarly easy to use:

```
use Math::Random::ISAAC;
my $rng = Math::Random::ISAAC::XS->new( @seeds );
$rng->irand();
```

Math::Random::ISAAC makes a point of not trying to pick a good seed value by default because the author believes that's a decision the user should make in a careful and concerted fashion. The documentation does point out a number of ways to generate a good seed value, one of which I want to mention because it addresses the issue we left open before when discussing srand(). At the moment, one of the better ways is to use the Math::TrulyRandom module that attempts to generate values based on "interrupt timing discrepancies." Math::TrulyRandom gets used like this:

```
use Math::TrulyRandom;
$random = truly_random_value();
```

According to the documentation, "The random numbers take a long time (in computer terms) to generate, so are only really useful for seeding pseudo random sequence generators."

Before we move on to the next section, I want to mention one of the more intriguing places you could turn to for randomness, namely the Web. The Perl module Net::Random queries one of two Web-based random number sources. One is fourmilab.ch, home of the HotBits project [1]. According to their site:

> HotBits is an Internet resource that brings *genuine* random numbers, generated by a process fundamentally governed by the inherent uncertainty in the quantum mechanical laws of nature, directly to your computer in a variety of forms. HotBits are generated by timing successive pairs of radioactive decays detected by a Geiger-Müller tube interfaced to a computer. You order up your serving of HotBits by filling out a request form specifying how many random bytes you want and in which format you'd like them delivered. Your request is relayed to the HotBits server, which flashes the random bytes back to you over the Web. Since the HotBits generation hardware produces data at a modest rate (about 100 bytes per second), requests are filled from an "inventory" of pre-built HotBits. Once the random bytes are delivered to you, they are immediately discarded—the same data will never be sent to any other user and no records are kept of the data at this or any other site.

The other random number source is random.org, a site devoted to randomness. Their site says:

> RANDOM.ORG offers *true* random numbers to anyone on the Internet. The randomness comes from atmospheric noise, which for many purposes is better than the pseudo-random number algorithms typically used in computer programs.

Random.org has a quota system in place that makes sure people don't abuse the system and try to consume all of the random bits it produces. You can, however, buy a larger quota, i.e., more bits. It amuses me that it is now possible to figure out how much randomness costs (at the time of this writing: $150 USD will get you 600,000,000 bits, or 4 million bits per dollar, which seems like quite a bargain, no?).

The Net::Random module knows how to query both Web services. Here's an example from its documentation:

```
use Net::Random;
my $rand = Net::Random->new();# use fourmilab.ch's randomness source,
        src => 'fourmilab.ch',          # and return results from 1 to 2000
        min => 1,
        max => 2000
);
@numbers = $rand->get(5);          # get 5 numbers
```

As the documentation for Net::Random points out, there are certainly security concerns with using a service like this (especially when you add caching Web proxies to the mix), so best check out the documentation before you start to use this for a serious project.

Modules for Generating Random Data

That last bit provides a nice segue into a section on ways Perl can help make using randomness in your applications easier. We saw a bit of this in my last column. In that column we explored a number of Perl modules like

Data::Generate and Data::Maker for creating random but plausible-looking data records. Let's look at a further expansion of the theme.

The first set of modules, similar to those we saw last time, are those that take some sort of specification and return random data in the form of your choice. For example, String::Random lets you write code like this:

```
use String::Random;
my $srobj = new String::Random;
$rand_string = $srobj->randregex('\d[a-z]\d');
```

Once run, $rand_string will consist of a random string containing a digit, a letter from a to z, followed by another digit. String::Random can either take a regular expression (using a subset of Perl regexp syntax), as was done above, or take a pattern more like pack() and create random strings based on that specification.

To create a more targeted set of data, you might find Data::Random and Data::Rand::Obscure more useful. The former lets you request different kinds of data using functions like:

```
rand_words()     - produce random words from a list
rand_chars()     - produce random characters from a defined set
rand_set()       - produce random array elements from a given array
rand_date()      - generate random date strings
rand_time()      - generate random time strings
rand_datetime()  - generate random date/time strings
```

and my favorite:

```
rand_image()     - generate a random PNG-format image
```

All of these functions behave the way you would expect—you hand them a few initialization parameters (such as the size of the character string you want to get back and the number of strings to return) and they return the data requested. The one thing you may find Data::Random doesn't do is provide a facility for only returning values not previously returned (i.e., only unique responses). For that you may wish to check out Data::Rand instead. It lets you provide a flag called 'do_not_repeat_index' which keeps the module from using any one array index into the given set more than once.

Data::Rand::Obscure is slightly more focused than either of these modules. Despite its name, it bears a closer resemblance in function to Data::Random than to Data::Rand. Data::Rand::Obscure provides a few functions along the lines of those we saw for Data::Random:

```
create_hex()   - create a random hex string (also create())
create_b64()   - create a random base64 string
create_bin()   - create a random binary value
```

The module "first generates a pseudo-random 'seed' and hashes it using a SHA-1, SHA-256, or MD5-digesting algorithm." The documentation goes on to say, "You can use the output to make obscure 'one-shot' identifiers for cookie data, 'secret' values, etc." It also makes dandy session keys for Web programming.

Make the Randomness Go Away

I tend to get a kick out of modules that are both counterintuitive and solve a clear problem using this twist from their usual expectations. Let's bring this column to a close by looking at two modules that solve basically the same problem.

Here's the issue in a nutshell: sometimes you write Perl code that should be tested using random data as input. Creating such tests is a commendable endeavor, but they add another layer of complexity to the development process. If you find one of your module's tests failing when presented with a certain input (randomly generated or not), you can't really be sure whether later programming efforts have squashed the bug unless you have some way to precisely reproduce the initial input to the code. In this scenario, we need a method for making previously random inputs reproducible (which essentially means removing the randomness from the "random inputs").

Both of these packages can intercept your tests' calls to rand(). In the case of Test::Random, you simply:

```
use Test::Random;
```

in front of your usual test code. If you call your test program with the TEST_RANDOM_SEED environment variable set, your code will use that particular seed every time. By default, Test::Random will display its current random seed so you can feed it back into the program. For example (from the Test::Random documentation):

```
$ perl some_test.t

    1..3
    ok 1
    ok 2
    ok 3
    # TEST_RANDOM_SEED=20891494266

$ TEST_RANDOM_SEED=20891494266 perl some_test.t

    1..3
    ok 1
    ok 2
    ok 3
    # TEST_RANDOM_SEED=20891494266
```

From this example you can see how the Test::Random magic lets you run the same test each time using the same predictable "random" input. Test::MockRandom is slightly more complicated and meant for intercepting rand()-like calls from within object-oriented programs. Be sure to read its documentation for further information on how to actually use it.

Take care, and I'll see you next time.

REFERENCES

[1] http://www.fourmilab.ch/hotbits/.

PETER BAER GALVIN

Pete's all things Sun: the status of Sun products

Peter Baer Galvin is the chief technologist for Corporate Technologies, a premier systems integrator and VAR (www.cptech.com). Before that, Peter was the systems manager for Brown University's Computer Science Department. He has written articles and columns for many publications and is co-author of the *Operating Systems Concepts* and *Applied Operating Systems Concepts* textbooks. As a consultant and trainer, Peter teaches tutorials and gives talks on security and system administration worldwide. Peter blogs at http://www.galvin.info and twitters as "PeterGalvin."

pbg@cptech.com

EVERY SUN OBSERVER—WHETHER CUSTOMER, partner, or competitor—is trying to get as much information as possible about the directions that Oracle is taking its Sun products. Each observer has their own reasons for doing so, and each has different hopes and goals. Certainly there is a lot of speculation and opinion—again, as varied as the reasons for the speculation.

In this column I collect together all that is known and some of what is being guessed about the future of Sun products under the new regime. Questions and comments are welcome, and probably best posted at the blog posting based on this column at http://ctistrategy.com. Also, where possible, I include a reference to the source of each data point I discuss here, so that you can read the details for yourselves and use that extended information to draw your own conclusions. I am just our humble guide on this journey through product status and futures.

Sources of Information

Nothing is certain until Oracle makes official product change announcements. Some of those have been made, but it is likely that more are to come. Which products will be canceled, which emphasized, and which perpetuated with little change? While waiting for firm product direction, all we have to go on for datacenter planning in some areas is what Oracle has announced and scattered reports about product directions. There is no NDA information included in this article for obvious reasons, but Oracle has clamped down on NDA information and presentations, making it more difficult than ever for IT managers to plan datacenter changes. Without further ado, here are the Sun product categories and their future, as far as I can tell.

Storage

The Sun 7000 Open Storage line (also known as the "ZFS Storage Appliance" by Oracle) is clearly going forward [1, 23]. Oracle has stated that the Sun 7000 will be the heart of its storage strategy. Likewise, the Sun tape libraries are market leaders and will be a continuing product line from Oracle [14]. As Oracle competes with IBM (as stated in its ads), storage and tape must be included in its product offerings to have a full line of solutions. Part of

the Oracle plan to differentiate from other storage vendors is to aggressively add flash storage throughout the storage lines.

The Sun partnership with Hitachi Data Systems (HDS) has been terminated, and Oracle/Sun does not sell the HDS storage arrays any longer [2]. This move seems to be part of a trend by Oracle to cancel agreements that involve Oracle reselling other companies' products. If that is the case, then Oracle either will not have a high-end storage product or believes that they can create their own. The recent addition of the FC protocol to the Sun 7000 software stack makes it a player in the SAN space, but it still lacks some features required in enterprise storage, such as near-instant failover of components when a fault occurs and synchronous replication. At the rate that the Sun 7000 feature set is evolving, such features are certainly possible. Could this be where Oracle is placing its enterprise storage hopes?

Servers

Sun had a variety of servers, in a variety of configurations and form factors. Many IT executives have been asking about the future of SPARC M servers and x86 servers. This product area has been especially rife with uncertainty and rumors. Again, a lack of NDA roadmaps adds to this confusion. The current plan appears to be as follows.

Intel-based x86 servers continue on in the Oracle portfolio. By the time you read this there are likely to have been more new products announced, but already Oracle has upgraded one product—the x2270 [3]. Less clear is whether the other existing products in the x2??? line get updated, or whether the x2270 becomes the lone server in the low-cost, fast, single-power supply race. Such servers tend to be for HPC and other fault-tolerant uses and the lowest margin servers, so it's a bit of a surprise that Oracle even has an offering in this space. Still, it is convenient to have a bare-bones server available for low-cost computing.

Sun was one of the first vendors to embrace the AMD CPUs for servers, having some exclusivity long ago as part of the deal. That deal is long past, and many other vendors have also shipped AMD-based servers. The great leap that Intel has made with its Nehalem-based CPUs is undeniable, to the point where AMD seems to be having trouble competing. Thus it is not much of a surprise that Oracle has no plans to continue creating AMD-based servers. It will be Intel CPUs only in the x86 servers [4].

Based on the ads Oracle has placed in various publications, they clearly want to compete (and win) in the server space. Oracle has stated that they will invest more in SPARC and Solaris R&D than Sun did, for example [5]. In a talk given by Sun's John Fowler and others at the Oracle/Sun launch after the approval of the merger, some details about SPARC futures were revealed [6]. In the T-server space, Oracle plans to release the T3 chip in 2010, including doubling the number of cores and increased performance all around. Beyond that, they plan on the T-servers moving from "throughput" to high-performance—that is, not just lots of cores and threads, but lots of fast cores and threads.

In that same announcement, Oracle had fewer details about the future of the M-servers. The next release is to include higher frequencies, increased I/O throughput, and larger caches. Assuming Fujitsu continues to iterate their SPARC CPUs, there should be more updates beyond that. Of course, an entirely new product line is possible as well, but nothing like that was announced. Generally, Oracle is hiring into the hardware engineering groups

with the announced goal of accelerating the multi-year roadmap for the SPARC servers (i.e., bringing them to market sooner than Sun had planned).

The question of which CPU is the "best" to run Oracle remains open, and is complicated by software pricing. The normal analysis method for determining platform direction is to look at the total cost of ownership (TCO) over a period of time, typically three years. Into that calculation go initial purchase cost of the servers, maintenance cost, and sometimes personnel costs, power, cooling, datacenter rack space and cost, and software licensing and support costs. Frequently, the software costs dwarf the other costs, so maximizing the performance per software license is a driving factor. Oracle publishes a document describing software licensing to help sort through the options [7]. They also publish a document discussing partitioning mechanisms and what are considered valid ways to limit the number of CPUs on which the Oracle database runs (and needs to be paid for) [8]. For example, a Solaris container running in a Dynamic Resource Pool (DRP) can be used to limit Oracle to a specific number of CPUs, and the Oracle license is needed only for that number. Finally, Oracle publishes a chart providing a "core factor"—a CPU-specific multiplier of core count that is used to determine how many Oracle licenses are needed for a given system [9]. For example, the UltraSPARC T2+ has a factor of .5, while the M series has a factor of .75. Multiply that times the number of cores used in the partition in which Oracle is running to determine how many licenses are needed. The factor for all x86-based systems is .5, giving them a pricing advantage at the moment. However, a change in the factors could change the TCO equation, so I recommend monitoring that core factor table.

Operating Systems

One of the more confusing aspects of the Oracle purchase of Sun is the operating system direction of the combined company. Oracle has been a big contributor to Linux and a big driver of its acceptance in the enterprise. But Oracle doesn't own Linux. Solaris has a leading feature set and is owned by Oracle. Add to the confusion the use of Linux in the Oracle Exadata appliance, and IT managers have a challenge in choosing operating system direction. What we do know is that Oracle has stated that Solaris is "The World's Best Operating System" [6]. That is not something one would say about a technology being put out to pasture. On the other hand, Oracle has also stated that they would continue contributing to the development of Btrfs, a next-generation file system for Linux [10]. It now becomes clear that Oracle is firmly behind both operating systems for the long term.

Unfortunately (in my view), Oracle has pulled back on the reins for spreading the use of Solaris. Solaris is no longer free to use, for example, and, in fact, a site can only buy support for Solaris if it is running on Sun hardware (rather than the previous policy of allowing hardware-independent Solaris maintenance contracts) [11]. On the other hand, Oracle seems to be allowing the selling and support of Solaris on other platforms when a contract with Oracle is in place. For example, the resale of Solaris on HP x86 servers appears to be continuing [12]. It would be a shame if Solaris got less emphasis over time. Solaris x86 gets Oracle database patches after other first-tier operating systems such as Solaris on SPARC and RHEL. Increasing its priority at Oracle would translate well to customers worried about its future and would increase its installed base. Solaris remains a great operating system (even a survey commissioned by HP shows Solaris as the number one mission-critical OS [13]), so the ball is in Oracle's court to foster use of Solaris.

OpenSolaris is a different kettle of fish, with no clear direction yet given by

Oracle. The OpenSolaris community is a bit miffed by this lack of direction, as can be seen by reading the exchanges at http://opensolaris.org. Certainly, OpenSolaris as the testing ground for new Solaris features will go forward, as Oracle has no other development environment for Solaris. Personally, I find it unlikely that OpenSolaris would move away from the open source path, due to the legal terms under which it was originally released. However, could some components added to Solaris "Next" (the next major release of Solaris) be close-sourced? Will Oracle continue to sponsor (via engineering and financial support) the OpenSolaris community? Those questions seem unanswered at the moment.

Appliances

Previous to the Sun purchase, Oracle had two appliances. Exadata V1 was jointly engineered with HP. Once the Oracle plan to purchase Sun was announced, that version was terminated and Exadata V2 was announced, being a joint venture between Oracle and Sun [1]. Oracle takes great pride in Exadata V2 and makes broad performance and success claims, but just how many orders have been placed for it is difficult to determine. Nevertheless, Oracle plans on creating more appliances. At least two more are expected to be released at Oracle World in September 2010. In their own words, "Where we think we'll make our money—where we think we're able to differentiate ourselves from IBM and everybody else—is by building complete and integrated systems from silicon all the way up through the software, all prepackaged together" [1].

What will be interesting about those next appliances are the technologies from which they are composed. That could indicate where Oracle is putting its emphasis within the Sun line, or maybe just what they thought would make compelling appliances with potential high demand and relatively quick development times. The Exadata appliances include Oracle Linux as the operating system, for example—what will the new appliances be based upon?

Other Areas

Some other areas of the Oracle/Sun product line have been fleshed out by Oracle. Virtualization, for example, has a full spectrum of solutions from a full suite of products. Those products include Oracle VM VirtualBox for desktops, Oracle VM Server for x86 (based on Xen), Containers for Solaris, Oracle VM server for SPARC (previously called LDOMS) for T-servers and Domains for M-servers [15]. Other Sun products seem to be continuing unabated, including SunRay desktops, Secure Global Desktop, and VDI. One sign to watch for is whether a product has been rebranded with "Oracle" at the front of its name—a sure sign it is now part of the Oracle family.

In many areas where Sun and Oracle had overlapping products, Oracle has announced at least general direction guidance, if not detailed roadmaps. Oracle's Identity Manager seems to be the winner over Sun's, for example, while Glassfish will continue alongside WebLogic and MySQL will live on as yet another database in the Oracle portfolio.

Management products is an area that Sun was continually weak in, and it hurt them competitively. Nowhere in the Sun catalog was a product like IBM's SMIT, for example. While SMIT might not the be best possible solution to the system administration problem, at least it is *a* solution (and has an ethos). Going forward, Oracle Enterprise Manager (OEM) will be the um-

brella under which products like XVM Ops Center live, and likely will gain features from those products as the products are merged into OEM.

As mentioned above, Oracle seems to be canceling some long-standing Sun agreements with other product vendors. The state of those relationships is not clear at this point, but some partnerships seem to be going forward even if product resale might not. For example, Brocade still touts their partnership with Oracle [16], but Oracle may not be reselling Brocade products. Likewise, Oracle and Red Hat are working together on multiple fronts [17], but Oracle does not seem to be reselling RHEL (especially since it competes with Oracle via Oracle Enterprise Linux—essentially RHEL but with lower support costs).

Maintenance of Sun hardware and software has changed drastically under Oracle. Before Oracle, Sun maintenance was complicated—single and multi-year options, multiple "metal" levels of support depending on the desired response time and weekend support requirements, and so on. Oracle has turned this model on its head, providing just one level of support—"Oracle Premier Support" for software and systems. It is similar to the previous "gold" Sun level support, but with many complex details, including expensive recertification of a system if it goes off maintenance. Oracle has published a document describing the full program [18].

Sun Futures

It is expected, but unfortunate, that there has been quite a bit of turnover at Sun as a result of the purchase by Oracle. Some of the biggest names at Sun are not employed by Oracle, including McNealy, Schwartz, Gosling, Bray, Phipps, and Tripathy [19]. Some will be missed more than others, but Oracle has to counter the brain-drain by adding excellent management and engineering to the Sun staff.

But not all Oracle changes to Sun have been for the worse. For example, Oracle is now offering free online courses about Solaris [20]. Oracle's increased investments in SPARC and Solaris is certainly welcome, but its effect will remain unknown in the short term. Also, Oracle is showing signs of making the Sun portfolio and R&D more sane by canceling non-core projects and emphasizing those that are core. An example of this is Project Darkstar, Sun's open-source, scalable, flexible architecture for massively multiplayer online games. Of course, the downside of these difficult decisions is that by limiting more long-term speculative R&D efforts, Oracle/Sun restricts its ability to come up with breakthrough technologies and solutions.

Oracle ownership of Sun is in its infancy, but already some are saying that Oracle is having difficulty managing the Sun purchase [21]. Many aspects of the hardware business are new to Oracle, and Oracle is rapidly changing how Sun does business. Hopefully, Oracle will be a fast learner, recognizing what is good and bad at Sun and sorting those out. Of course, doing that without alienating customers and partners would be the best result. Only time will tell if that is a result Oracle can attain.

Tidbits

As this issue of *;login:* is security based, I thought I should include at least something about security. As described in a previous column, the CIS Benchmark [22] is a tremendous tool for gauging the current security stance and improving the stance of Solaris systems. The document has recently

been updated, so whether you've previously had a look or not, now would be a good time to do so.

Also, Oracle has produced a new quick reference document covering their Sun offerings [23].

REFERENCES

[1] http://www.reuters.com/article/idUSTRE64B5YX20100512.

[2] http://www.crn.com/storage/223101244.

[3] http://sun.systemnews.com/articles/147/3/server/23153.

[4] http://www.theregister.co.uk/2010/05/27/oracle_spikes_opterons/.

[5] http://www.osnews.com/story/22147/Oracle_Promises_to_Commit_to_SPARC_Solaris.

[6] http://www.oracle.com/ocom/groups/public/@ocom/documents/webcontent/044518.pdf.

[7] http://www.oracle.com/corporate/pricing/sig.pdf.

[8] http://www.oracle.com/corporate/pricing/partitioning.pdf.

[9] http://www.oracle.com/corporate/contracts/library/processor-core-factor-table.pdf.

[10] http://www.linux-magazine.com/Online/News/Future-of-Btrfs-Secured.

[11] http://www.sun.com/software/solaris/licensing/policies.xml.

[12] http://sun.systemnews.com/articles/147/3/Solaris/23104.

[13] http://www.itpro.co.uk/623683/unix-still-a-hit-for-mission-critical-systems.

[14] http://www.enterprisestorageforum.com/hardware/news/article.php/3861446/Oracle-to-Keep-Suns-Data-Storage-Tape-Businesses.htm.

[15] http://www.oracle.com/us/technologies/virtualization/index.htm.

[16] http://www.brocade.com/partnerships/technology-alliance-partners/technology-alliances/Oracle/index.page.

[17] http://www.oracle.com/partnerships/hw/redhat/index.html.

[18] http://www.oracle.com/support/collateral/hardware-systems-support-policies.pdf.

[19] http://www.infoworld.com/d/the-industry-standard/suns-stars-where-are-they-now-and-why-did-they-leave-765.

[20] https://learning.sun.com/olc/smartstart.

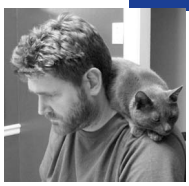
[21] http://www.fool.com/investing/general/2010/05/28/the-sun-isnt-shining-for-oracle.aspx.

[22] http://cisecurity.org/en-us/?route=downloads.benchmarks.

[23] http://www.oracle.com/us/products/servers-storage/039224.pdf.

DAVE JOSEPHSEN

iVoyeur: pockets-o-packets, part 2



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

SOMEONE ONCE TOLD ME THAT OUR perception of temporal reality changes as we grow older not because our aging brains are slowing down but, rather, because we're gaining context. Things seemed to move so slowly around us when we were young because every new thing we learned made up a huge percentage of what we already knew. When you only knew four people's names, the next new name you learned made up 1/5 of your total knowledge on the subject. How many names do you know now?

I find this a comforting thought, but despite my best efforts I have trouble believing it. The fact that I can't even remember who said it to me is a telling example (I'm almost certain it was a "she"). At any rate, reality seems to me to be speeding up and changing disproportionately to the meager amount of context I've managed to gather (I'm only thirty . . . uh . . . five? six?). But even though I'm barely middle-aged, it seems to me the fundamentals of the world are changing more rapidly than they used to. The incandescent light bulb was invented about 100 years ago. An above-average human life-span to be sure, but it's possible today to have been born in a time when reading by lantern was the norm in North America, and to have lived from that time, through the construction of the grids, air conditioning, the moon landings, nuclear power, solid-state electronics, and the Internet—a mind-boggling influx of reality-shattering changes. Fifty years before that light bulb there were places on the continent for which there were no maps.

For my part, I'm not sure if I'll be able to tell today's children about life before the Internet. I have all sorts of fond memories of how it began—getting my first UNIX shell account from PrimeNet for $6 per month (to MUD), asking my first questions at altavista.digital.com, etc., but I have trouble remembering how I learned things before I could pull my pocket-sized everything-machine out of my pants and ask the magical Google answer hive-mind any question that happened to pop into my head. Before the Internet, how did we research things? I was there, but now I have trouble imagining it. There are several changes that have occurred in my lifetime that I could say this about; before things changed, I didn't know anything was amiss, but afterwards I couldn't imagine what life was like before. I suppose this was what was meant by the

phrase "paradigm shift" before the marketeers got hold of it: when things change so fundamentally that the previous reality no longer even seems possible.

Argus

This month I promised I'd talk more about Argus, because I didn't get nearly as deep as I wanted to with it last time. If you read this column with any regularity, you may have noticed I'm prone to gushing, but there is no amount of gushing I could do over Argus that wouldn't be deserved. Argus is one of the changes I alluded to above in that I'm not sure how you'd manage a network without it, but I assume it would involve an inordinate amount of guesswork. Argus knows everything about everything that ever transpired on my network. It knows so much about network interaction that it regularly teaches me things about networking in general that I thought I already had a practical understanding of.

I've learned so much myself from my interactions with Argus that I recommend it as an aid to people who want to study networking. Not only has Argus taught me things about my network and about networks in general, it has taught me objective truths about the universe. For example, Argus has taught me that I cannot ever, under any circumstances, trust a DBA. I may like them personally, and even respect them professionally, but trust them I simply cannot do, and while this is something I'd often suspected, it is because of Argus that I know it to be the truth.

So let's get our hands dirty with an examination of our HQ network's Argus infrastructure, and then I'll walk through a couple of examples of how I use Argus on a day-to-day basis. Argus is actually two projects: Argus itself [1] is a flow capture daemon, while the Argus Client Programs [2] are the tools that do everything else. The client utilities all have names that begin 'ra' (read argus), while the Argus daemon is just called argus. From now on, when I'm talking about Argus the project, to include the client utilities, I'll use the capital *A*, and when I'm referring to the argus packet capture binary, I'll use the lowercase *a*.

It can be difficult to understand what all the different little Argus programs do, because there are a lot of them, and there is a lot of functionality overlap between them. argus, for example, really is a packet capture daemon, but it can be used just like one would use tcpdump, to watch packet flows in real time from a given interface. In our design, and in most of the Argus setups I think you'll likely run into, argus is run on the router, firewall, or device we're interested in gathering traffic from, and various Argus client programs are used to collect, parse, retransmit, store, and analyze the collected traffic. The Argus documentation refers to argus in this configuration as a "probe."

Probe Setup

Let's start with my probe setup. The core routers in our HQ building are OpenBSD boxes on commodity hardware with 10 1 Gbps interfaces each (see Figure 1). Given that they are gigabit devices and have a lot of ports, it would be expensive to use hardware network taps with them, but since they're multi-purpose OSes on commodity hardware, Argus is a natural fit. Since argus can only listen to a single port at a time, we'll run one argus instance for each device and then consolidate them into a single data flow before exporting it off the box. It's possible to run argus as a stand-alone daemon (argus -d), but I prefer to run it under daemontools [3]. My daemontools service directory looks like this:

argus.em0 argus.em1 argus.em2 argus.em3 argus.em4 argus.em5 argus.em6 argus.em7 argus.em8 argus.em9 radium

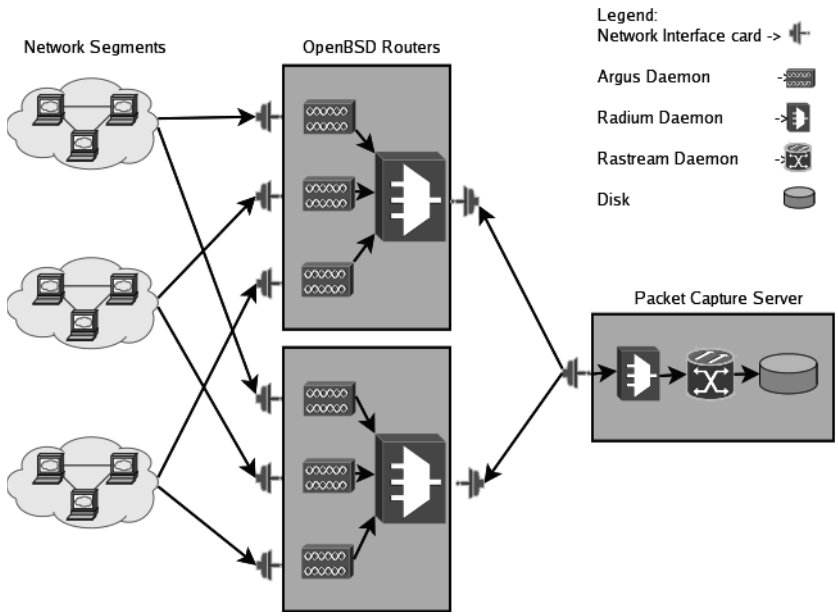


FIGURE 1: HOW ARGUS COMPONENTS FIT TOGETHER IN MY HQ NETWORK ARCHITECTURE.

Each argus.x directory has a daemontools run script, but really these are all just symlinks of the same script, only differing in interface name. Once the symlinks, variables, et al. have been resolved, the argus command for em0 comes out looking like this:

```
/usr/local/sbin/argus -i em0  -B '127.0.0.1' -P "4000" 2>&1
```

And for em1:

```
/usr/local/sbin/argus -i em1  -B '127.0.0.1' -P "4001" 2>&1
```

So you've probably guessed that -i specifies the interface name (just like tcpdump). If I were to stop there, argus would have given human-readable output to STDOUT, but since I specified -P, argus will instead provide Argus-readable data to port 40XX. This is an important concept to understand—every Argus program can give two kinds of output: human-readable and binary Argus data stream. The default is always to output human-readable, but if you want to save output to a file, or pipe the output of one Argus program to another, you need to specify -w. A -w <filename> writes Argus output to a file, while -w - writes it to STDOUT, suitable for piping to other Argus programs. Most Argus programs also support -P, which will write the output to a network socket, and this will always be Argus binary data format. In my daemontools scripts I've also specified the optional -B switch, which tells argus to bind to a specific address rather than 0.0.0.0, which is the default.

So my router has 10 argus daemon processes, each writing to localhost on its own high-number port. Now I want to combine the output from all these daemon processes and provide them as a singular data stream to my syslog/pcap server. To do this I use radium, which you may have noticed among the services directories listed above (radium also supports -d if you want it to run as a stand-alone daemon). radium, which describes itself as a "real-time Argus Record multiplexor," is the first of the Argus clients I'll be

talking about, and it's the only Argus client I use on the capture endpoint. radium is intended to do exactly what I need here: it reads argus data from any number of network sockets and outputs a single aggregated data stream to the destination of your choosing. Here's what my radium command looks like when daemontools execs it:

```
radium -S localhost:4000 -S localhost:4001 -S localhost:4002 -S
localhost:4003 -S localhost:4004 -S localhost:4005 -S localhost:4006 -S
localhost:4007 -S localhost:4008 -S localhost:4009 -B 10.20.1.2 -P 3999
```

You've probably guessed that -S specifies a network socket from which to read argus data. You can also read from files with -r, but there are limitations. radium can only read one file at a time, and you cannot read from both sockets and files at the same time. radium, like argus and most of the other Argus tools, supports -B and -P for writing out to sockets and -w for writing out to files. In my environment we provide our aggregated data stream on port 3999 on the router's internal IP, where the syslog/pcap server can read it, and we lock it down with pf locally so only the pcap server can connect.

Data Collector Setup

The /service directory on the pcap server has a radium daemon and a rastream daemon. The radium command on the pcap server looks like this:

```
/usr/local/sbin/radium -S 10.20.1.2:3999 -S 10.20.1.3:3999 -B 127.0.0.1 -P
4001
```

If you are abnormally observant you may have noticed that the internal address of the router ended in ".2". This is because there are actually two routers in a failover setup using CARP. So the pcap server has a radium service that aggregates the flows from each of these routers and provides this stream on localhost:4001. Even if you only have a single probe to read from, I'd recommend using radium to collect it on your pcap box, because this future-proofs things. If/when you get more probes, incorporating them is just an additional -S. rastream's job is to write the data stream to log files, and I could have simply piped radium straight into rastream, but I prefer to write to a socket here instead. This makes it possible for me to come to the pcap box run any client tools against localhost 4001, and get real-time info from all of my probes. radium supports 128 simultaneous client connections, so reading the localhost port on the pcap box doesn't interfere with the logging process. My rastream command looks like this:

```
/usr/local/bin/rastream -S localhost:4001 \
-M time 10m -w "/var/log/argus/%Y-%m-%d/argus.%H.%M.%S" 2>&1
```

The rastream command reads the aggregated probe data from port 4001 and then splits the resulting output into consecutive sections of records based on criteria I give it. These can include file size, time intervals, record count, or even events in the data stream (e.g., rotate the file when you see an SSH connection to a specific host). I use time mode (-M time) with an interval of 10 minutes. I also specify the names of the resultant log files with the -w switch. rastream supports an extended -w option that includes variable expansion with the $. Since I split the logs based on time interval, I use time-based variables, but rastream can actually resolve any printable field in the flow record, which includes byte-counts and the like.

Record Management

Now that the data is on a disk, we can use any of the argus client tools to

read it with -r. We can also put some thought into file management and compression. Larger Argus installations can easily generate hundreds of gigs of data per day, so it may not be enough to write the files and forget about them. Let me tell you the three most popular ways to get the data file sizes under control.

The first option you have is a client called racluster. racluster is an amazing Swiss Army knife of a client. In fact, it's the client I use to do most of my data analysis, and I'll be talking more about it in the next issue. With its default options, however, it takes argus input from -S or -r and merges status records from the same flow and argus probe. As argus captures packets from the network, it reports flow statistics every 5 seconds or so. That means there will be *n*/5 records per network connection, where *n* is the number of seconds each connection lasts. Running racluster against an argus dataset merges all of these little records into singular records that represent the entire flow from beginning to end. In my configuration, doing this reduces the size of the data files by 40–60%. We do this via cron once per day. The racluster command looks like this:

```
/usr/local/bin/racluster -R /var/log/argus/${DAY}/ -w /var/log/argus-${DAY}.log
```

racluster is one of several argus clients that support recursively reading directories full of Argus data files with -R. We simply read in a directory of 10-minute-interval files and write out a single file representing the day. racluster needs no other options whatsoever to drastically shrink the size of your archives.

The second option is filtering. Every Argus client, as well as argus itself, supports the use of tcpdump-style traffic filters. Simply give the Argus command as normal, then add a - followed by a tcpdump filter, and you're there. For example, if I wanted to get rid of everything that wasn't TCP traffic, as well as use racluster to shrink my files, I could do:

```
/usr/local/bin/racluster -R /var/log/argus/${DAY}/ -w /var/log/argus-${DAY}.log - tcp
```

The full subset of tcpdump filtering is supported, so things like - tcp and dst host 192.168.5.1 and dst port 888 work fine. I could also do this earlier in the stream by adding the filter to any of the argus, radium, or rastream commands I've pasted above. Many larger installations run multiple filtered rastream clients against their radium service to store very specific subsets of network traffic in different files, because of storage requirements as well as maintaining separation of duties. This plus racluster is usually enough, but if not, your third option is bzip2. Enough said.

That pretty much covers the Argus infrastructure we're currently using for our office network, which is, I think, a pretty fair example of how end-to-end Argus installations are generally implemented. Unfortunately, I'm out of space this month, so next month, in the third (!) part of this series, I'll cover the really fun stuff: data mining and problem solving with Argus.

Take it easy.

REFERENCES

[1] http://qosient.com/argus/dev/argus-3.0.2.tar.gz.

[2] http://qosient.com/argus/dev/argus-clients-3.0.2.tar.gz.

[3] http://cr.yp.to/daemontools.html.

ROBERT G. FERRELL

/dev/random: a debatably helpful guide to IT disaster planning

Robert G. Ferrell is an information security geek biding his time until that genius grant finally comes through.

rgferrell@gmail.com

HOT TOPICS COME AND GO ON A VER- tigo-inducing rotation in IT. Sometimes one will gain popularity because of some blog post or news story; other times they seem to be generated out of thin air, apropos of not much. Nothing brings the theme of disasters into the forefront quite as effectively as a bona fide "there but for the grace of the Flying Spaghetti Monster go I" event like a plane crash or oil spill. Or maybe a plane crashing *into* an oil spill. During an earthquake. After being shot down by a rogue Predator drone whose unencrypted data stream was hacked by the widow of a Nigerian Deputy Finance Minister trying to leave you her husband's ill-gotten millions while dying of cancer. Dressed like Lady Gaga.

In IT, though, we seldom call a spade a spade, preferring content-free technical jargon like "wireless earth displacement unit," so we sequester the rather inelegant concept of disasters in a rich, velvety coating of Business Continuity Planning. That's a lot less likely to spook the stakeholders. Let us now suppose, for the sake of stretching this column out to an acceptable length, that after 24 straight hours of Anderson Cooper total immersion your CEO finally decided to jump on the Business Continuity bandwagon and you got elected to lead the orchestra. The path to BCP enlightenment is about as smooth as a drug-runner's illicit jungle runway after a meteorite shower, but fortunately I'm here to provide you with an insider's knowledge of at least the lefthand side of the equation. I've been involved in more disasters than the Great Dirigibles' Last Flights Reenactment Working Group.

BCP is a lot easier if you skip the boring stuff about RPOs, RTOs, crisis management command structures, and so on, and plunge headlong into the disaster itself. Planning a proper disaster is lots more work than most people realize. They get so wrapped up in the before and after details that they put almost no effort into implementing what is, after all, the raison d'être for the entire process. All you really need to get a good disaster going is something worth losing and plausible deniability for your involvement in said loss. Ferrell's First Law of Business Entropy states that, once set into motion, all systems tend toward catastrophic failure.

The first and foremost priority for IT disaster planning is risk analysis. If you have functional IT

equipment a fair amount of risk is present right out of the box, so *that* one is something of a no-brainer in my book. The successful disaster will be well mapped out beforehand. Which servers will be involved? How many hard drives? What percentage of your organization's irreplaceable data will be lost? Whose jobs will be on the chopping block so management appears to be taking this seriously? Which senior executive bonuses, if any, will be affected, and by how much? How will you make it up to them under the table? It is critically important that the culpable party or parties be identified well in advance so that public statements can be crafted and ready for release once a suitable "incident investigation" interval has passed.

Essential logistical planning completed, the project can now move on to the implementation phase. This is where operating system patches are ignored or not properly tested on a non-production system before deployment, hardware is placed in thermally suboptimal environments containing lots of aerosolized particulate matter, and data centers are strategically located on active fault zones, in flood plains, on the slopes of questionably dormant volcanoes or, if you want to go for the trifecta, all of the above. Improperly configured firewalls and lack of an IT acceptable-use policy, anti-malware guards, or safe surfing briefings will be of great assistance during this phase.

The disaster sequence itself doesn't require much participation from the staff. As with death, taxes, and bad legislation, it just happens. It is possible and in fact advisable to influence the timing, however. Those who will be involved in the recovery effort are probably going to want to arrange it such that they can claim maximum overtime/comp time as a result. Senior management will need to minimize negative publicity, so popping on the heels of some much more newsworthy calamity is ideal. The hourly consultants who will invariably be called in to lend the impression that people who actually know what they're doing are on the case will, of course, want to drag things out as long as possible. Juggling these somewhat competing priorities is a challenge that may itself require outsourcing. It should be noted that infinite recursion is a distinct hazard where consultants are concerned.

The aftermath of an IT meltdown can be a fruited plain, ripe for the harvest, if handled correctly. A surprising number of suddenly untraceable assets may be written off, as well as any accounts payable that the payees don't realize they now need to pursue much more aggressively. With a little luck and some skilled wringing of hands you can slough off a sizeable chunk of that pesky indebtedness. Disasters can also put the *amor* in *amortize*. Subpoenas to testify before Congress are worth more than their weight in gold. By the time you finish exploiting the resulting talk show circuit and bestseller list for a few months you'll more than likely have forgotten exactly what the hoo-ha was all about to begin with. Fortunately, they say that vast amounts of money are a great balm for the amnesiac.

Caveat: inattention to detail can ruin a good disaster. For example, installing adequate uninterruptible power supplies, religious patching, and other solid configuration management practices can be extremely deleterious for failure. Another potential kiss of death for IT Armageddon is the practice of periodic off-site backups, although this threat can be ameliorated somewhat if the archive media aren't being checked for integrity or the courier regularly leaves your backups in plain sight on his car seat while he runs into the local pub for a quick one or four.

If you follow my guidelines carefully, you can be almost totally assured of an organization-altering IT cataclysm and national press attention. After all, if you're going to have a disaster, why skimp? Anything worth doing is worth doing spectacularly badly.

Bon calamité.

book reviews

ELIZABETH ZWICKY, WITH BRANDON CHING AND SAM STOVER

THE MYTHS OF SECURITY: WHAT THE COMPUTER SECURITY INDUSTRY DOESN'T WANT YOU TO KNOW

John Viega

O'Reilly, 2009. 232 pp.
ISBN 978-0-596-52302-2

I like the book, but hate the subtitle. Most of the security industry desperately wants you to know the truth. I don't think the author really believes the subtitle either—surely the security industry, even in its most evil fear-spreading moments, does not actually want everybody to believe that the antivirus companies spread viruses? That is one of the myths he talks about.

Quibbling aside, this is a nice common-sense discussion of security, in the form of a lot of short essays. They are casual in tone and mostly non-technical. It's written by an insider, and the audience varies a bit; some essays are solidly aimed at non-technical people, some at technical people, and the occasional one is really insider-to-insider (those are the only people who need to be told to suck it up and live with the name "antivirus" for all sorts of malware prevention). There are lots of people who could enjoy this (I did), but its best audience is probably developers, who ought to know something about security, and the young and security-obsessed, who are prone to myths. I suspect the scandal-mongering subtitle is aimed at sucking in the latter group.

FATAL SYSTEM ERROR: THE HUNT FOR THE NEW CRIME LORDS WHO ARE BRINGING DOWN THE INTERNET

Joseph Menn

Public Affairs, 2010. 265 pp.
ISBN 978-1-58648-748-5

This is a believable, gripping non-fiction story about computer crime and the Russian mafia. You'd think that would be somewhat redundant; how could you write a boring story about computer crime and the Russian mafia? Sadly, I recently heard three speakers talk about computer crime and the Russian mafia, and one of them managed to make it considerably less gripping than, say, billing procedures. So boring is possible, and unbelievable is easy, which makes this book all the more surprising.

It lays out the story clearly, so you can see how it makes human sense. Some bits are obvious; my life is full of people who could easily have gotten in the middle of multiple crime families by a combination of technical focus and inability to recognize when people are really not trustworthy. Shadowy secretive gangs protected by the police sound a lot harder to believe in when brought up out of context, but actually make plenty of sense when set in their proper background. These days, when people ask me if I really believe that there are international crime rings involved in most computer crime, I tell them that it's like the drug trade. Sure, people whip up viruses in their own homes; they also grow marijuana in their own homes. But sooner or later, if they keep it up, they're going to run into serious criminals, and then they're going to become very small players in loosely coupled but large international crime rings.

This book reads like a novel, only with endnotes, and, like good spy fiction, is simultaneously enjoyable to read and somewhat depressing.

THE NUMBERS GAME: THE COMMONSENSE GUIDE TO UNDERSTANDING NUMBERS IN THE NEWS, IN POLITICS, AND IN LIFE

Michael Blastland and Andrew Dilnot

Gotham Books, 2009. 202 pp.
ISBN 978-1-592-40485-8

This is another book of nice, digestible essays loosely held together. It has some perceptive explanations I haven't seen elsewhere, including a good discussion of why numbers might, in fact, mean nothing at all. It's aimed primarily at helping you interpret the news, but is also a good start on thinking about numbers you may have dug up yourself. The section on how performance measurement changes things is especially important reading; it may be relevant to your management structure, and it will certainly be critical to you if you want to display metrics from your trouble-ticket or bug-tracking system.

I particularly resonated with the chapter in which you ask yourself, "Is this really a big number?" having recently spent a good bit of time contemplating the question, "Is 500,000 computers a big botnet?" (Yes, but not fascinatingly large.)

STATISTICS EXPLAINED: AN INTRODUCTORY GUIDE FOR LIFE SCIENTISTS

Steve McKillup

Cambridge University Press, 2006. 262 pages.
ISBN 978-0-521-54316-3

I have to admit that even system administration is generally not considered a life science, and almost everything else involving computers is even farther away. On the other hand, I haven't yet found a good book on statistics for the computer industry. You'd think business books on statistics would be the way to go, but those I can only find for students, they weigh 10 pounds and are printed in full color with the highlighting already done for you, and they make me too cranky to read, let alone review, them. (Also, they cost the better part of $200 each, which makes it very hard to suggest that anybody buy them.)

Apparently, biology students are less free-spending, at least on math books, because this is a reasonably sized black-and-white paperback. It assumes that you are intelligent and motivated and not terrified by mathematics, but probably not all that mathematical, and it goes through a *lot* of statistics and experimental design. It speeds right through some of the basics (probability gets a page-and-a-half sidebar) and doesn't really slow down until it gets to things like ANOVAs. This makes it a good follow-up to really beginning statistics books. There's also a helpful multipage decision chart on picking experimental designs and tests which, in itself, is worth the price of entry. If you really don't know any statistics, start somewhere else, but if you've mastered dice rolls and how to tell averages apart, this is a good next place to go. You may not be measuring fish, but it will still be useful to you.

RESTFUL WEB SERVICES COOKBOOK: SOLUTIONS FOR IMPROVING SCALABILITY AND SIMPLICITY

Subbu Allamaraju

Yahoo! Press, 2010. 293 pp.
ISBN 978-0596801687

REVIEWED BY BRANDON CHING

The *RESTful Web Services Cookbook* by Subbu Allamaraju is an excellent companion text for Web developers in search of REST solutions. Like most texts in the O'Reilly Cookbook series, the book is not an all-encompassing manual. Rather, Allamaraju outlines a number of commonly faced problems in building RESTful applications and provides concise solutions to those problems.

The book is divided into 14 chapters and a number of appendices. Allamaraju covers a wide range of topics, from resource identification and Atom to security and caching issues. While no single book can encompass all possible problems a REST developer may face, Allamaraju does an excellent job in answering what are likely the most common and far-reaching questions. Coming in at under 300 pages, this book nevertheless contains an impressive breadth of coverage.

I found the coverage of security (Chapter 12) quite useful. While all Web developers know basic HTTP authentication, Allamaraju extensively covers digest and OAuth (both two- and three-legged) authentication methods as well. The sections on conditional requests, cache validation, and concurrency control were also quite valuable. Allamaraju includes the handling of JSON, XML, Atom, and SOAP in a number of examples throughout the book.

Allamaraju's writing is clear and easy to read. All examples in the book use HTTP and XML to demonstrate implementation. While the REST architecture is not the most complicated to understand, it is very often poorly implemented. The author does a very good job of delivering solutions in an understandable way so that when you have to implement a RESTful application, it will be done right.

This book is written for experienced Web developers in any programming language and belongs on the shelf of anyone working in REST applications. However, since the book focuses on higher-level REST methodology, I would recommend a language-specific REST resource in addition to this book (unless you are a serious guru in your project language).

SECURING THE BORDERLESS NETWORK: SECURITY FOR THE WEB 2.0 WORLD

Tom Gillis

Cisco Press, 2010. 125 pp.
ISBN 978-1-57805-886-8

REVIEWED BY SAM STOVER

I should have known. The author of this book is the VP and General Manager for the Security Technology Business Unit at Cisco and was part of

the founding team at IronPort Systems (Senior VP of Marketing). Had I read the bio, I wouldn't have been surprised, but I didn't, so I was. I'll be honest: I think this book is more about how awesome Cisco is, not really about securing the "borderless network." That's not to say that it doesn't have merit; Cisco *can* be awesome, and there is a bit of history throughout the book that's interesting. But $45 is a little pricy for what you get.

The book is actually small—really small, only 125 pages. The 15 chapters are very short, most no more than ten pages, so that's great for anyone with a short attention span. Chapter 1 runs through the evolution of the firewall and warns that this type of technology will never handle Web 2.0 traffic properly. Chapter 2, "Collaboration and Web 2.0 Technologies," was especially disturbing to me; it seemed to say that companies not dialed into the new way of the Web will lose their appeal to "Gen Y" employees and ultimately fail. The unfortunate point, as it seemed to me, was that this trend is inevitable, so you had better just give in. Personally, I prefer to believe that while it might be inevitable, it needs to be done right, and I didn't find the emphasis on doing it right that I was hoping for in a (alleged) security book. Well, at least in a book that has "Securing" in its title.

Chapters 3 and 4 go on to show how productive you can be if you just give in to cloud computing and online collaboration. Chapter 5 is basically an advertisement for Cisco's Telepresence and WebEx tools. Chapters 6 and 7 extol the virtues of the smartphone but, again, don't really dive into what can be done to actually secure anything.

Chapters 8 and 9 give an overview of malware and the people who use it. Lots of history and perspective, but a little lean on mitigation and how those technologies impact companies embracing Web 2.0 technologies. For a while there, I forgot that this book is about Web 2.0 and securing it. Chapter 10 claims to offer "Signs of Hope," but all I got out of it was that Cisco's global product footprint and their multicore technology are two great answers to the malware problem. Chapters 11 and 12 discuss AUPs and data loss events: AUPs are too restrictive and just get bypassed, and data loss is bad. If you don't already know this, then maybe this book is for you.

Chapter 13 claims that we need to re-engineer our IT departments to stop denying Web 2.0 technologies by "Saying 'No Thanks' to the 'Culture of No.'" Again, too much of "this is inevitable" (I get it already, seriously) and not much on how to do it right. Chapter 14 follows on its heels with the challenges of authentication and how Cisco is readying their "Identity Fabric." This was pretty interesting, but I'm not sure it came to a solid conclusion—it was more like a discussion on a hard problem, with some ideas Cisco is proposing to address it.

Finally, Chapter 15 looks like it will answer all of our questions. It's entitled "Security for the Borderless Network: Making Web 2.0 and 3.0 Safe for Business" and it contains, well, pretty much nothing we don't already know. Starting with instructing companies to make their policies more flexible and ending with promises of next-gen scanning tools from Cisco, there's not much here for the person looking to actually apply security to their Web 2.0 users and technology.

Maybe I'm just not the right audience for this book, and if that's the case, then most of the *;login:* readership probably isn't either. If you want to learn about next-gen Cisco multicore silver bullets, head to their Web page. If you want to hear that your policies and procedures aren't sufficient to account for Web 2.0 or (shudder) 3.0, ask, well, any audit or assessment firm. If you want someone to tell you that new technologies are coming and that you'd better be ready, well, I'll tell you that for free.

USENIX 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 summaries of sessions at USENIX conferences.

ACCESS TO ;LOGIN: online from October 1997 to this month: www.usenix.org/publications/login/

ACCESS TO VIDEOS from USENIX events in the first six months after the event: www.usenix.org/publications/multimedia/

DISCOUNTS on registration fees for all USENIX conferences.

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

THE RIGHT TO VOTE on matters affecting the Association, its bylaws, and election of its directors and officers.

FOR MORE INFORMATION regarding membership or benefits, please see www.usenix.org/membership/ or contact office@usenix.org; 510-528-8649.

USENIX BOARD OF DIRECTORS

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

PRESIDENT

Clem Col
Intel
clem@usenix.org

VICE PRESIDENT

Margo Seltzer
Harvard University
margo@usenix.org

SECRETARY

Alva Couch
Tufts University
alva@usenix.org

TREASURER

Brian Noble
University of Michigan
brian@usenix.org

DIRECTORS

John Arrasjid
VMware
johna@usenix.org

David Blank-Edelman
Northeastern University
dnb@usenix.org

Matt Blaze
University of Pennsylvania
matt@usenix.org

Niels Provos
Google
niels@usenix.org

EXECUTIVE DIRECTOR

Ellie Young,
ellie@usenix.org

USENIX LIFETIME ACHIEVEMENT AWARD

PRESENTED AT THE 2010 USENIX FEDERATED CONFERENCES WEEK TO WARD CUNNINGHAM

Ward Cunningham has always been a computer scientist on the cutting edge of programming, with a remarkable ability to see the future. His work has focused on the art of programming and helping people do things in a better, speedier, more "agile" way. While others in the community were arguing C vs. Pascal, Ward was experimenting with object-oriented programming, developing Extreme Programming and pattern languages. Realizing that the most significant projects are developed collaboratively, he began to consider how to facilitate communication in collaborative work. The result? His most famous and perhaps his most lasting contribution: the invention of the first wiki, which he called the WikiWikiWeb after the Hawaiian word for "fast."

STUG AWARD

PRESENTED AT THE 2010 USENIX FEDERATED CONFERENCES WEEK TO MEDIAWIKI AND THE WIKIMEDIA FOUNDATION

The STUG Award recognizes significant contributions to the community that reflect the spirit and character demonstrated by those who came together in the Software Tools User Group (STUG).

Most computer users, particularly our youngest ones, may find it hard to imagine a world without wikis and especially hard to imagine one without Wikipedia. We are proud to present the USENIX STUG Award to MediaWiki and to its parent organization, the Wikimedia Foundation, which operates some of the largest collaboratively edited reference projects in the world, including Wikipedia.

But we also know that this wonderful piece of code did not just appear. The ideas behind them were not new, but a few people had to come forward to implement a wiki that could underlie a real scalable, worldwide service. That service was MediaWiki. It took a com-

munity to collaborate, putting in many many hours of work to make the idea great and make it lasting. We recognize Magnus Manske, Lee Crocker, Brion Vibber, and Tim Starling as the major contributors to MediaWiki, but we all realize that their work has been refined and improved by many others. MediaWiki is a wonderful example of a software tool that changed our world.

USENIX ASSOCIATION FINANCIAL STATEMENTS FOR 2009

Ellie Young, Executive Director

The following information is provided as the annual report of the USENIX Association's finances. The accompanying statements (p. 79) 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 2009 financial statements were also audited by McSweeney & Associates, CPAs. Accompanying the statements are several charts that illustrate where your membership dues go. The Association's complete financial statements for the fiscal year ended December 31, 2009, are available on request.

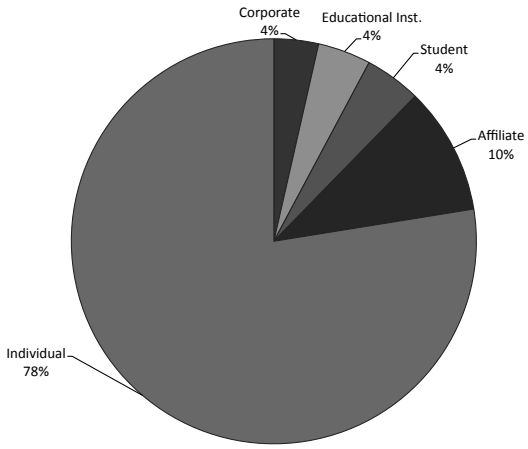


CHART 1: USENIX MEMBER REVENUE SOURCES 2009

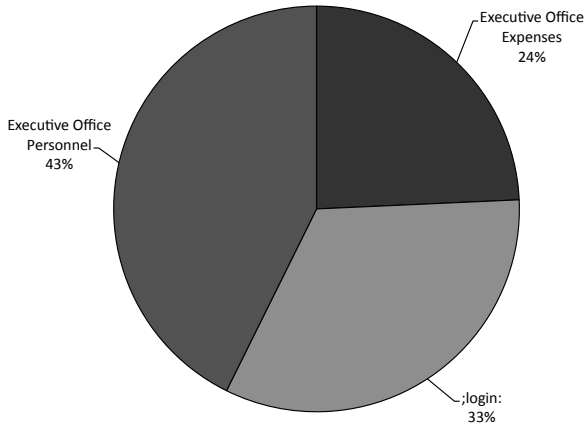


CHART 2: WHERE YOUR 2009 MEMBERSHIP DUES WENT

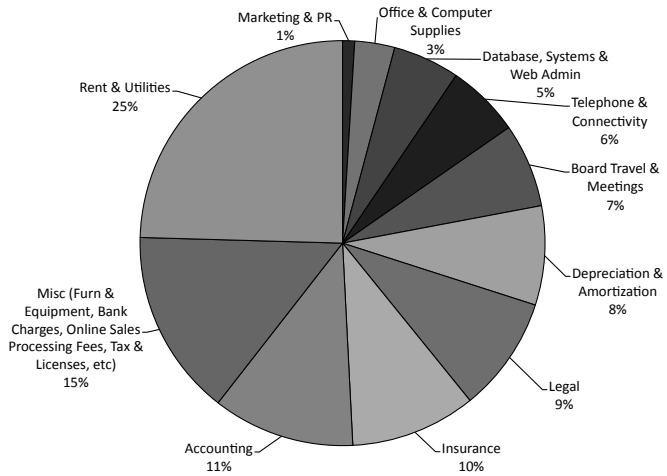


CHART 3: USENIX ADMINISTRATIVE EXPENSES 2009

USENIX ASSOCIATION
STATEMENTS OF FINANCIAL POSITION
As of December 31, 2009 & 2008

ASSETS	2009	2008
Current Assets		
Cash & cash equivalents	$ 210,710	$ 507,714
Receivables	40,184	47,511
Prepaid expenses	93,180	53,319
Inventory	-	2,433
Total current assets	344,075	610,977
Investments at fair market value	5,765,887	5,530,751
Property and Equipment		
Office furniture and equipment	306,308	564,353
Leasehold improvements	29,631	29,631
Less: accumulated depreciation & amortization	(277,837)	(509,139)
Net property and equipment	58,102	84,845
Other assets	284,757	194,341
	$ 6,452,821	$ 6,420,914

LIABILITIES AND NET ASSETS		
Current Liabilities		
Accounts payable	$ 41,066	$ 219,009
Accrued expenses	56,528	157,830
Deferred revenue	97,810	108,240
Total current liabilities	195,404	485,079
Long-Term Liabilities	284,757	194,341
Total liabilities	480,161	679,420
Net Assets		
Unrestricted net assets	5,972,660	5,741,494
Net Assets	5,972,660	5,741,494
	$ 6,452,821	$ 6,420,914

USENIX ASSOCIATION
STATEMENTS OF ACTIVITIES
For the Years Ended December 31, 2009 & 2008

	2009	2008
REVENUES		
Conference & workshop revenue	$ 2,282,899	$ 3,360,576
Membership dues	489,578	567,608
Product sales	1,542	4,333
SAGE dues & other revenue	115,102	141,504
Total revenues	2,889,121	4,074,021
OPERATING EXPENSES		
Conferences & workshops	2,175,669	2,848,684
Membership; login:	385,268	434,737
Projects & GoodWorks	148,687	311,129
SAGE	122,931	185,494
Management and general	461,173	573,645
Fund raising	39,984	42,649
Total operating expenses	3,333,712	4,396,338
Net operating deficit	(444,591)	(322,317)
NON-OPERATING ACTIVITY		
Interest & dividend income	193,439	238,825
Gains & losses on marketable securities	543,214	(997,095)
Investment fees	(57,531)	(64,891)
Other non-operating	(3,364)	(89,299)
Net investment income & non-operating expense	675,757	(912,460)
Increase/(decrease) in net assets	231,166	(1,234,777)
Net assets, beginning of year	5,741,494	6,976,271
Net assets, end of year	$ 5,972,660	$ 5,741,494

conference reports

THANKS TO OUR SUMMARIZERS

7th USENIX Symposium on Networked Systems Design and Implementation (NSDI '10)

San Jose, CA
April 28–30, 2010

OPENING REMARKS AND AWARDS PRESENTATION

Summarized by Rik Farrow (rik@usenix.org)

Alex Snoeren started the conference with the numbers: 224 attendees, 175 papers submitted, three rejected for not following formatting rules, and 29 accepted for presentation. He went on to thank the Program Committee, sponsors, and USENIX staff before announcing the Best Paper award.

The "Reverse Traceroute" paper (Ethan Katz-Bassett, University of Washington; Harsha V. Madhyastha, University of California, San Diego; Vijay Kumar Adhikari, University of Minnesota; Colin Scott, Justine Sherry, Peter van Wesep, Thomas Anderson, and Arvind Krishnamurthy, University of Washington) got the award. Snoeren then introduced Laura Hess, vice chair of the Computer Research Association (CRA), who explained that the CRA mission is to strengthen CS research as well as to support participation by women and minorities. Hess announced the female winner of the CRA, Justine Sherry (also one of the authors of the Best Paper award). Sherry has done a lot of work using timestamp algorithms in IP headers, has just graduated from University of Washington, and has been accepted into UC Berkeley.

CLOUD SERVICES

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

- ***Centrifuge: Integrated Lease Management and Partitioning for Cloud Services***

Atul Adya, Google; John Dunagan and Alec Wolman, Microsoft Research

Many distributed applications partition responsibility across multiple application servers and employ leases on those responsibilities in order to handle server churn. Alec Wolman spoke about Centrifuge, a reusable component which implements these services for Microsoft's Live Mesh platform. Centrifuge is targeted at interactive applications whose servers keep many small data objects in memory and operate on data cached from either the clients or back-end storage.

The Centrifuge architecture consists of three components: lookup nodes, which make requests for data objects; owner nodes, which have a lease on the cached data; and the Centrifuge manager service. The Centrifuge service hands out leases on ranges in order to scale to very large numbers of objects and uses consistent hashing to partition the namespace. To ensure consistency when handling an application-specific operation, an

owner checks that it owns the lease both before and after performing the operation; this is known as the "check-lease continuous" call. Leases are granted for 60 seconds and are renewed every 15 seconds.

When an application server crashes, systems using Centrifuge recover data from the client. This is cheaper than holding multiple copies in other servers and avoids the complexity and performance issues of quorum protocols. Wolman argued that this does not add extra complexity, because such recovery is already necessary for tolerating correlated failures. Centrifuge is currently designed for about 1000 servers in a single cluster.

Ben Reed asked if they had investigated lowering the lease time from 60 seconds, which he felt was a long time for an interactive system. Wolman responded that higher lease times lead to greater system stability and that the delay from re-fetching the data from stable storage (particularly if it is on the client) has a greater impact on system interactivity. Gün Sirer remarked that they are implementing load-link/store-conditional semantics and wondered what happens when the "check-lease continuous" call fails; he also asked if the operations needed to be side-effect free. Wolman confirmed that the owner nodes can't expose state and that if "check-lease continuous" failed, the server was expected to contact the lookup server in order to properly redirect the client's request.

- ***Volley: Automated Data Placement for Geo-Distributed Cloud Services***
  Sharad Agarwal, John Dunagan, Navendu Jain, Stefan Saroiu, and Alec Wolman, Microsoft Research; Harbinder Bhogan, University of Toronto

Cloud service providers use multiple data centers in order to reduce the latency when serving requests from globally distributed users. Of course, provisioning capacity at these data centers and WAN bandwidth between them are expensive. Sharad Agarwal presented Volley, an algorithm for partitioning data across data centers while minimizing end-user latencies and datacenter costs. Volley handles the data placement calculations for the Live Messenger service, which supports on the order of 100 million users conducting around 10 billion conversations per month.

Volley processes log data to produce a graph with connections between IP address locations and the data items accessed from those locations. There are also connections representing interactions between data items. All connections are weighted by how frequently the interactions occurred. The Volley algorithm then proceeds in three phases. In the first phase, a good initial solution is calculated by finding the weighted spherical mean of accesses to each data item. In the second phase, this solution is iteratively updated by using a weighted spring model which attempts to pull data closer to users. In the final phase, Volley assigns each data item to the data center closest to the optimal location while respecting capacity constraints.

After explaining the algorithm, Agarwal presented a comparison of Volley against alternate placement strategies for data in Microsoft's Live Mesh service. Volley reduced WAN traffic by 1.8x, reduced data center skew by 2x, and reduced user latency by 30% at the 75th percentile.

Chip Killian asked if there was a way to determine which latencies mattered the most to users. Agarwal replied that the services that use Volley do not currently collect that information. He mentioned that they had considered using the location information specified in users' account profiles, but found the listed location to match the use locations for only 50% of cases. Agarwal was also asked how they account for users traveling and he responded that they weight the importance of each access location by the number of requests the user made from that location. Ben Zhao asked about the role of CDNs in Volley. Agarwal said that Volley is designed for operators of dynamic services who don't want to pay the cost and overhead for the data to be replicated and consistent across the CDN.

- ***Optimizing Cost and Performance in Online Service Provider Networks***
  Zheng Zhang, Purdue University; Ming Zhang and Albert Greenberg, Microsoft Research; Y. Charlie Hu, Purdue University; Ratul Mahajan, Microsoft Research; Blaine Christian, Microsoft Corporation

Ming Zhang discussed the problem of traffic engineering in online service provider (OSP) networks, such as those run by Google, Microsoft, and Yahoo!. The goal of traffic engineering in an OSP network is to reduce the round-trip time experienced by end-user requests. This is a difficult objective, because OSP networks contain dozens of data centers and connect to hundreds of ISPs, and each ISP offers paths to only certain destination prefixes. Furthermore, path capacity constraints create dependencies among prefixes.

To solve this problem, a system called Entact was proposed. Entact injects /32 routes into the network in order to measure the performance of unused alternate paths to each prefix without affecting production traffic. These measurements are then used in a linear programming model which maps the cost/performance trade-off curve of a given traffic engineering strategy. The network operator is then able to select the strategy that makes the appropriate trade-off. Entact was applied to a simulation of Microsoft's OSP network and achieved a 40% cost reduction without degrading end-user performance.

Zhang was asked about the overhead of probing all of the prefixes to determine their performance. Entact only probes at a rate of about 1 Mbps from each data center, sending the probes from the two nearest data centers to each prefix, and it doesn't evaluate all alternative paths equally. Could Entact use actual traffic statistics for performance prediction instead of doing route injections? By only examining traffic, Entact might miss some alternative routes which could be evaluated by route injections.

WIRELESS 1

Summarized by Ann Kilzer (akilzer@cs.utexas.edu)

- ***Exploring Link Correlation for Efficient Flooding in Wireless Sensor Networks***
  Ting Zhu, Ziguo Zhong, Tian He, and Zhi-Li Zhang, University of Minnesota, Twin Cities

Ting Zhu explained how they created a flooding protocol that provides high reliability with fewer retransmissions. They began by challenging the common assumption of link independence in wireless networks. Through an empirical study conducted both indoors and outdoors, they found that packed receptions are in fact highly correlated. The authors leveraged this discovery to reduce redundancy and provide increased reliability by developing "collective acknowledgment" and using Dynamic Forwarder Selection, the details of which are in the paper.

Next, Zhu illustrated the Collective Flooding (CF) protocol and its difference from traditional flooding with a simple three-node network. In traditional flooding, a sender needs to wait for both receiving nodes to send ACKs. However, suppose that receiver 1 has a 100% conditional packet reception probability for receiver 2, i.e., the probability that receiver 2 has received the packet given that receiver 1 has received the packet is 100%. Then the sender only needs to get an ACK from receiver 1 to infer that both receivers got the message. Zhu then detailed the CF protocol and its three stages: maintenance, sender, and receiver. The nodes need different information at each stage. Nodes in the maintenance stage need link quality and conditional packet reception probability. Nodes in the sender and receiver stages need to know which nodes have not received the message (uncovered nodes), as well as coverage probabilities of neighboring nodes. A back-off timer is used to determine which node sends retransmissions. Throughout the protocol, nodes calculate coverage probabilities for all one-hop neighbors. When the coverage probability exceeds a threshold, the flooding protocol terminates.

To evaluate their flooding protocol, the authors used indoor and outdoor tests, as well as large-scale simulations. Additionally, they compared CF to other flooding algorithms such as RBP8 (a variation of RBP), RBP, and standard flooding. They found that CF reduced total transmissions by 31.2% and dissemination delay by 55% against RPB8. Additionally, CF had similar reliability to RBP8 and reduced the number of redundant retransmissions. Further information is available at www.cs.umn.edu/~tzhu.

Aaron Schulman from the University of Maryland asked why there was a correlation in packet reception between nodes. Zhu explained that background noise can cause correlation. Additionally, receivers have different sensitivity to packet receptions, so some nodes will always receive more packets. Philip Levis from Stanford University asked why there was a correlation in the outdoor context, because there should be fewer physical obstacles. Zhu again noted that this was because the receivers have different sensitivities to packets.

- ***Supporting Demanding Wireless Applications with Frequency-agile Radios***
  Lei Yang, University of California, Santa Barbara; Wei Hou, Tsinghua University; Lili Cao, Ben Y. Zhao, and Haitao Zheng, University of California, Santa Barbara

Lei Yang began by noting the recent growth of multimedia streaming in home and office environments, emphasizing the high bandwidth and quality-of-service requirements. Their work was motivated by a desire to provide streaming in a wireless environment. They aimed to build a system to provide continuous access to radio spectrum with high-bandwidth transmissions, even with concurrent traffic. An obvious solution would be to use WiFi; however, Yang noted that there is no dedicated access, and CSMA contention causes many unpredictable disruptions.

The authors proposed per-session frequency domain sharing, which enables concurrent media sessions on separated frequencies. This solution would avoid interference and allow continuous spectrum access for each media session. Yang noted that the FCC has auctioned and released new spectrum, making this work all the more feasible. He then introduced Jello, a decentralized home media system based on frequency domain sharing. This system uses frequency-agile radios, which sense and select free spectrum for media sessions.

Yang noted some of the interesting research and engineering problems in building the Jello system. First, there is the problem of finding free spectrum. Conventional methods use energy detection, which is unreliable. Yang demonstrated how to find free spectrum by using an edge-detection algorithm on a power spectrum density map. A second problem is choosing frequency blocks. Jello uses a best-fit approach, meaning that the radio will broadcast on the narrowest available frequency block that will sustain the current demand. A third challenge is fragmentation of the spectrum as links enter and leave. This can result in situations where there are enough bands to support a request, but they are not contiguous. Yang compared spectrum fragmentation to disk fragmentation, but noted that unlike on hard drives, a centralized defragmenter would not work because it would require stopping all transmissions. The authors proposed a solution of voluntary spectrum changes to lower fragmentation. They also proposed a second solution of non-contiguous spectrum access. However, this adds more overhead. Additionally, more fragmentation means more spectrum lost to guard bands, which are required between each link.

For evaluation, they used an 8-node GNU radio testbed with four concurrent flows. They evaluated a static system with equally partitioned spectrum, Jello with contiguous frequency access (Jello-C), Jello with non-contiguous frequency access (Jello-Full), and a simulated optimal solution without fragmentation or overhead. They measured the percentage of time a video stream was disrupted. In Jello-

Full, disruption was under 5%, while Jello-C had disruption around 18%. The static solution had 23% disruption. A full implementation of Jello can be found at http://www.cs.ucsb.edu/~htzheng/papyrus.

Sayandeep Sen from the University of Wisconsin-Madison wanted to know if the disruption metric referred to frame freezes or glitches in the video. Yang explained that disruption was not based on perception; rather, it meant that the allocated spectrum could not support the current video demand. Basically, it referred to packet loss beyond a certain threshold. Sen asked about the band used for edge detection, noting cases where noise reaches the level of the signal. Yang noted that when the transmission signal is comparable to noise, Jello combines energy detection and edge detection. If the transmission signal is so low it's indistinguishable from noise, it's considered free. Ratal Mahajan of Microsoft Research asked how close links were to one another. They used fixed-size guard bands to prevent interference in these experiments, but for heterogeneous devices in complex environments, adaptive guard bands could be useful.

PEER-TO-PEER

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

- ***Contracts: Practical Contribution Incentives for P2P Live Streaming***
  Michael Piatek and Arvind Krishnamurthy, University of Washington; Arun Venkataramani, University of Massachusetts; Richard Yang, Yale University; David Zhang, PPLive; Alexander Jaffe, University of Washington

Piatek presented Contracts, a system that improves the performance and robustness of P2P live streaming. Current P2P live streaming systems (e.g., PPLive) offer no incentive for peers to contribute by sharing downloads. Piatek first discussed the possibility of using incentive mechanisms that rely on bilateral reciprocation (e.g., tit-for-tat), but noted that these are ineffective in live streaming systems, for three main reasons. First, peers are heterogeneous with respect to upload capacity, making it difficult to maximize capacity as well as limiting the number of users the system can serve. Second, there are limited trading opportunities between peers, as there are few "interesting" blocks in the system at a given time (due to the nature of live streaming), and peers far away from the source tend to receive blocks after everyone else. Third, there is no compelling reward in live streaming; clients can't download faster than the streaming rate.

Instead of relying on bilateral reciprocation, Contracts uses a global evaluation function to evaluate peers' contributions as well as the effectiveness of those contributions. These contributions are verified using cryptographic receipts. Peers who give more to the system are rewarded with robustness by being moved closer to the source. As a result, if the system becomes constrained, contributors fail last. Contracts also has defenses against collusion, which are presented in the paper. Piatek noted in the question-answer session that this is the main attack strategy against Contracts. Additionally, he agreed that if a large number of high capacity peers colluded and disconnected at once, there would be a significant service disruption, but connectivity could always be recovered by going to the source.

In the talk, Piatek presented results of his Contracts implementation in PPLive on Emulab. Results show that Contracts improves performance and strengthens contribution incentives over tit-for-tat. Further results in the paper examine Contracts' overhead, its performance vs. FlightPath (OSDI '08), the speed of its topology convergence, and its performance when over-provisioned.

- ***Experiences with CoralCDN: A Five-Year Operational View***
  Michael J. Freedman, Princeton University

Freedman presented a summary of his experiences from his five-year deployment of CoralCDN, a content distribution network (CDN) designed to make content widely available regardless of the publisher's resources. This talk focused on CoralCDN's interactions with the external environment (clients, origin domains, PlanetLab slivers, etc.) and usage patterns, rather than data from user feedback on their experience, which, as Freedman noted in the question-answer session, would be extremely difficult to collect. He also mentioned that it was a conscious decision not to provide active user support for the system, due to the amount of time that would consume. Currently CoralCDN takes very little time to maintain.

Freedman focused on three areas of CoralCDN: its naming scheme, its fault tolerance capabilities, and its resource management. CoralCDN's naming scheme provides a programmatic, open API for adopters that they began using as an elastic resource. However, it doesn't mesh well with domain-based access control policies, in particular those that use cookies (due to the nature of its naming scheme, CoralCDN's design can allow JavaScript-managed cookies from evil.com to interact with those from target.com). CoralCDN handles internal failures well, but has more trouble dealing with external failures due to problems such as misleading return values from origin sites. Finally, CoralCDN employs fair-sharing algorithms to enforce per-domain control over its bandwidth consumption (a reaction to some sites using CoralCDN—and thus PlanetLab—to serve a large amount of content), but suffers from a lack of control and visibility into the environment's resources. This motivates the need for lower layers to expose greater control over their resources.

Based on these experiences, Freedman discussed potential improvements to CoralCDN's design. Currently, CoralCDN is often used to resurrect old content or access unpopular content, neither of which it is particularly well suited for, as it wasn't designed for these purposes. It is also used to serve long-term popular content as well as to survive flash crowds

(Freedman noted that flash crowds tend to occur on the order of minutes rather than seconds, though it's hard to predict whether this will continue to be a trend). One could imagine redesigning CoralCDN to use little cooperation between nodes and focus on serving long-term popular content, or to focus on serving flash crowds and cooperate at a regional level, using more global cooperation only as insurance for origin failures. The latter could be done by leveraging the Coral hierarchy for lookup, and preliminary results for doing so are presented in the paper. In an attempt to reach Internet scale, Freedman announced the browser-based Firecoral, which can be downloaded at http://firecoral.net/.

- ***Whānau: A Sybil-proof Distributed Hash Table***
  Chris Lesniewski-Laas and M. Frans Kaashoek, MIT CSAIL

In this talk, Chris Lesniewski-Laas presented Whānau, a Sybil-resistant DHT that is oblivious to the number of Sybils in the network. Whānau involves two phases: setup and lookup. In the setup phase, nodes build routing tables using links from a social network. Relying on an assumption of sparse cuts between the honest nodes in the network and Sybils, random walks through the social network will return mostly honest nodes, and Whānau uses these nodes to build its routing table. The lookup phase is similar to lookup in a standard DHT. By keeping a sufficient number of fingers and keys at each node, with high probability Whānau can find a key in one hop. In the worst case, an honest node makes many connections to Sybil nodes, but Lesniewski-Laas noted in the question-answer session that Whānau allows for a certain fraction of these types of nodes, while still guaranteeing a one-hop DHT with high probability. He also noted that all nodes can utilize the same random walk length (this could be set as a system parameter); no node would benefit from choosing its own walk length.

Delving deeper into the construction of Whānau, Lesniewski-Laas presented the idea of layered identifiers. Layered IDs prevent the "cluster attack" on DHTs, where Sybils pick their IDs in such a way that they can overtake part of the DHT. Essentially, Whānau nodes choose multiple IDs—one per layer for log(n) layers—in such a way that there is a low probability of Sybil nodes clustering in any given layer. In the question-answer session Lesniewski-Laas mentioned that it would be harder to prevent this attack in multi-hop DHTs, as each hop moves a node further from its social network, and thus further from peers it trusts.

In the talk, Lesniewski-Laas presented simulation results verifying that popular social networks do exhibit the type of structure that Whānau requires (in particular, that they are fast-mixing), and that Whānau can deliver high availability even in a network with a large number of attack edges. Further results on layered identifiers, scalability, and churn are presented in the paper, along with an evaluation on PlanetLab.

WEB SERVICES 1

Summarized by Michael Chow (mcc59@cornell.edu)

- ***Crom: Faster Web Browsing Using Speculative Execution***
  James Mickens, Jeremy Elson, Jon Howell, and Jay Lorch, Microsoft Research

James Mickens described how in order to achieve a more responsive Internet we need to reduce user-perceived fetch latency. Prefetching has been suggested as a way to reduce latency. In Web 1.0 it was easier to prefetch data. There was a static graph connected by declarative links that had to be traversed. But in Web 2.0, the content graph is no longer static and the links are now dynamically linked. The new challenges to prefetching are handlers triggered by imperative JavaScript event handlers that cause side effects. This prevents the pre-execution of handlers. Thus, the prefetcher must understand JavaScript and it has to understand the side effects of execution fetches driven by user-generated inputs (e.g., clicking buttons or typing a search form). It is impossible to speculate on all possible user inputs. Prior solutions have used custom code.

Mickens then introduced Crom, a framework for creating speculative jobs. It is written in regular JavaScript with no modification to browsers. It is a generic speculation engine for JavaScript. Crom handles the lower-level details of speculative execution. It has to replicate browser state and rewrites event handlers. Replicating browser state requires cloning two pieces: the application heap and the DOM tree. To clone the application heap, it walks the object graph and deep-copies everything. To clone the DOM tree, the browser calls body.cloneNode() and then Crom has to do a fix-up traversal. Event handlers are rewritten to use the speculative contexts. It uses the JavaScript with(obj) function to insert obj in front of the scope chain. Speculative contexts are committed if two conditions are satisfied: the start state is equivalent to the application's current state and speculative input that mutated it must be equivalent to the real input.

The application must define two functions. It must define an equivalent function, which is a hash function over the global namespace, and a mutator function, which tells Crom how to change a new speculative context before running the speculative event handler. Mickens then demonstrated Crom speculating on a user Bing search. Crom has three speculation modes: full copy mode, checked lazy mode, and unchecked lazy mode. Full copy mode clones the entire heap for each speculation. It is always safe, but it may be slow. Checked lazy mode has lazy copy and parent tracking. It is also always safe, and parent mapping costs are amortizable across speculations. This may also be slow. Unchecked

lazy mode is fast. It is often safe in practice but, strictly speaking, it is unsafe without checked mode referencing.

The DHTMLGoodies Tab Manager was used to evaluate Crom. In evaluating Crom, they wanted to know if they could hide the overhead incurred from speculative execution in user think time. In unchecked lazy mode, pre-speculation costs sum to a trivial 24 ms. Committing the speculative context requires 77 ms. However, 99% of this is for committing the DOM pointer, which needs to be paid even for non-speculative execution, so this adds no overhead. In the checked lazy mode, there is a new source of overhead in creating the parent map pre-speculation, and at commit time it needs to fix state parent references. The pre-speculation time is 182 ms and the commit overhead was 5 ms, which is proportional to the number of objects cloned, not all JavaScript objects. Usually this is fine, but sometimes it is not acceptable for some applications. In conclusion, prefetching in Web 2.0 is different. JavaScript has side effects that need to be considered for speculative execution. Crom is a generic JavaScript speculation engine. Applications express their speculative intents, and Crom automates the low-level tasks for speculation. Crom can reduce user-perceived latency in loading dynamic pages.

Ryan Peterson from Cornell asked how Crom handles side effects from speculative executions on the server side. Mickens answered that the logs will show that these requests are speculative and that Crom relies on the developer to implicitly understand the side effects incurred on the server side (e.g., the developer should not speculatively execute purchases on Amazon). There is more on this in the paper. Wyatt Lloyd from Princeton asked about contention with running multiple JavaScript applications. Mickens replied that most JavaScript applications are not CPU bound, but it shouldn't be a problem when using unchecked lazy mode.

- ***WebProphet: Automating Performance Prediction for Web Services***
  Zhichun Li, Northwestern University; Ming Zhang, Microsoft Research; Zhaosheng Zhu, Data Domain Inc.; Yan Chen, Northwestern University; Albert Greenberg and Yi-Min Wang, Microsoft Research

Zhichun Li talked about the prevalence of Web services and how slow performance in these services leads to loss in revenue. An extra 100 ms delay at Amazon leads to 1% sales loss. Performance optimization is a hard problem. Web services are complicated, and knowing complete object dependencies is difficult: there is a lot of JavaScript, and services are hosted in data centers around the world. The metric used for performance optimization is page-load time (PLT). The user-perceived PLT is the page or portion with the most visual effects. PLT depends on a number of factors: client delay, net delay, and server delay. And each one of these delays comes from others such as DNS delay, TCP connection setup delay, and data transfer delay. Given the number of possible factors causing delay, there are a large number of possible performance optimization strategies, but there are also limitations to existing techniques. The A/B test, which uses controlled experiments, sets up a group of users to test the Web service. These tests are hard to fully automate, however, and are slow. Service provider-based techniques have problems with multiple data sources and object dependencies. Regression-based techniques usually require independence assumptions on delay factors of each object.

Li proposed a new tool for automated performance prediction with fast client-side prediction. It makes use of a timing perturbation based on dependency discovery and a dependancy-driven page load simulation. In order to extract dependencies, the authors used a lightweight black box–based approach that is browser-independent. Simple HTML parsing and DOM traversal are not enough, since there are event triggers, and object requests generated by JavaScript depend on corresponding .js files. The timing perturbation-based technique injects a delay through an HTTP proxy and sees how the delay propagates. HTML objects are special; they are stream objects, so they allow incremental rendering. The page is loaded slowly and the offset for each embedded object is measured. In order to simulate this process, the simulator loads the page according to a constructed dependancy graph and then adds corresponding delays, figuring out how long the new page load time is.

The WebProphet framework is a Web agent that extracts dependencies, predicts performance, and analyzes traces. The evaluation was on a Google and a Yahoo search. Checking the dependencies was done manually. In evaluating WebProphet, controlled experiments were run with a baseline of high latency experiments. The new scenario would use low latency connections, and a control gateway would inject and remove delays. In their PlanetLab setup, they used international nodes as the baseline and the new scenario for low latency nodes was nodes in the U.S. Looking at the results of the PlanetLab experiment, there was a maximal error of about 11%.

A possible usage scenario described in the talk was analyzing how to improve Yahoo! Maps. They only wanted to optimize a small number of objects. They evaluated 21,767 hypothetical scenarios in 20 seconds and found that moving five objects to the CDN results in a 14.8% speedup. Reducing the client delays of 14 objects to half the time results in a 26.6% speedup. Combining both leads to a 40.1% speedup or a reduction of load time from 4 seconds to 2.4 seconds.

Are there some services for which there are variable times depending on what kind of requests are made (e.g., a maps service)? If object and dependencies don't change, then the dynamic loading times will be the same. If not, they won't be the same. What about Internet routing delays? They are only looking at the client-side user perspective. Their tool only considers factors that the user sees.

- ***Mugshot: Deterministic Capture and Replay for JavaScript Applications***
  James Mickens, Jeremy Elson, and Jon Howell, Microsoft Research

In his second appearance as a presenter (and wearing a long, blond wig as part of his "intense commitment to sartorial flair"), James Mickens began by talking about when things go wrong with modern Web sites. Modern Web sites have event-driven functionality, with hard errors, such as unexpected exceptions, and soft errors, such as broken event handlers. When things go wrong, developers normally perform a core dump, stack trace, error log, etc., but it would be useful to them to see the path to how the error occurred. The developer cannot rely on users to report nondeterministic events either. The authors' solution is Mugshot, which logs nondeterministic JavaScript events, uploads an event log, and allows developers to replay the buggy program. Mugshot is a lightweight JavaScript library.

Mickens then presented an example of a page with a parent frame with a parent button and two child frames, each with a button. Each frame requires a copy of the logger.js file, but there is only one log maintained by the parent frame. In the example, the logger would need to log these two sources of nondeterminism: clicking the button and the return value of the date. Although it sounds simple, Mickens pointed out that for any DOM node/event name pair there can be, at most, one DOM 0 handler, while there can be an arbitrary number of DOM 2 handlers. Firefox calls the DOM 0 handler before DOM 2 handlers, which is difficult for Mugshot since Mugshot needs to run before any app-defined handlers in order to log these events. Multiple tricks, outlined in the paper, were used to get around both this and additional problems, such as IE having no capture phase, which makes logging GUI events tricky in IE.

In logging the value of loads, a mapping of a file name to bits may change from logging time to replay time. So a replay proxy is needed. At replay time, the developer uses the replay proxy as a Web proxy. On the developer machine, a transparent iframe is put on top of the page. Functions are interposed from the log, the log is fetched, and a VCR control is displayed. The log is stepped through on replay. In evaluating Mugshot, Mickens looked at how big the logs got. He showed that they grow at about 16 Kbps. The CPU overhead incurred from logging was only 2% on Firefox and about 7% on IE. This is important, since the user should not notice the logging.

Was there integration with the server-side, since applications run both in the browser and on the server? This was handled in many cases, because Mugshot logs HTTP headers, Ajax, and cookies. Would this work with other browsers such as Opera or Safari? Code used for Firefox could mostly be reused for Opera and Safari, since they are mostly DOM compliant. Was there really a need for the proxy and why couldn't they put it all in the client side? The proxy was needed because JavaScript code cannot inspect the cache, and there is no way for the Mugshot JavaScript to look in the cache after an image fetch.

WIRELESS 2

Summarized by Kevin Bauer (kevin.bauer@colorado.edu)

- ***AccuRate: Constellation Based Rate Estimation in Wireless Networks***
  Souvik Sen, Naveen Santhapuri, and Romit Roy Choudhury, Duke University; Srihari Nelakuditi, University of South Carolina

Souvik Sen presented a system that selects the optimal bit rate for a wireless channel between a client and the wireless access point. Choosing the optimal bit rate for a given wireless link is an important problem, since higher bit rates translate directly into better throughput for clients. However, achieving high throughput in a wireless link is not as simple as just picking the highest supported bit rate. If the wireless channel quality is poor, higher bit rates may result in excessive packet loss and reduced throughput for clients. Furthermore, unpredictable channel fluctuations caused by interference from other clients or environmental factors can affect a link's quality over time and space. This makes it difficult to predict the optimal bit rate for the link in advance.

Sen presented AccuRate, a system that uses physical layer information to improve bit rate selection, with the following intuition. First, the wireless physical layer encodes sequences of bits into a "symbol" in constellation space. AccuRate analyzes the dispersion between the transmitted and received symbol positions, where symbol dispersion is the distance between the received signals and the signal constellation. Small symbol dispersions indicate a strong channel that could potentially support a higher bit rate.

AccuRate arrives at the optimal bit rate for a given wireless link by measuring the symbol dispersions across many different bit rates. By replaying packets at increasing bit rates and measuring the resulting symbol dispersions, AccuRate determines the highest possible bit rate at which a packet could have been received successfully.

The system was implemented and evaluated using USRP hardware and GNURadio software with 802.11-like encoding and modulation schemes. In addition, the system is evaluated in simulation to characterize the performance with highly mobile clients. Sen explains that AccuRate is able to predict a packet's optimal bit rate 95% of the time when the packet is correctly received, and 93% of the time when only the preamble and postamble are correctly received. In terms of throughput, AccuRate is shown to achieve 87% of the optimal throughput.

Is it really necessary to estimate the optimal bit rate on every packet, or is it possible to perform more sporadic sampling? If the channel properties are known to be stable for a certain time period, it would be permissible to use the same bit rate for that time duration and then use Accu-

Rate again after this time elapses. Michael Freedman asked how frequently they need to run their algorithm, and Sen answered that you can use the rate for 100 ms and only calculate the rate once each time period.

- ***Scalable WiFi Media Delivery through Adaptive Broadcasts***
  Sayandeep Sen, Neel Kamal Madabhushi, and Suman Banerjee, University of Wisconsin—Madison

Sayandeep Sen presented an approach for achieving scalable delivery of high definition (5 Mbps or higher) media to clients at a wireless hotspot. Current approaches to media distribution over wireless networks rely on either unicast or broadcast. For unicast media delivery, the wireless access point receives one copy of the video and distributes *N* distinct copies of each video packet to each of the *N* clients. Unicast allows for individual packet retransmissions (via MAC layer acknowledgments) and physical layer rate adaptation, but is inefficient since each client needs its own copy of each packet. In contrast, wireless broadcast requires only a single copy of each packet for all clients. However, broadcasts have no channel feedback mechanisms (since MAC layer acknowledgments are disabled) and, consequently, offer no retransmissions or rate adaptation. To truly achieve scalable media delivery in wireless with improved performance, the authors sought to combine the best features of unicast and broadcast in their proposed system.

The key contribution of this work is the recognition that certain MAC layer functions such as retransmissions and rate adaptation can be improved with knowledge of the value of each packet to the application. Through value-aware adaptive broadcast, it is possible to retransmit valuable packets instead of packets with little or no value to the receiver. Sen presented MEdia Delivery USing Adaptive (pseudo)-broadcast, or Medusa, which offers rate adaptation, packet prioritization, and retransmission planning based on the priority of packets. A live demonstration of the Medusa system was shown to the audience, along with a comparison to wireless broadcast. The demo showed a significant increase in video quality.

Experimental results demonstrate that Medusa is able to achieve high PSNR (a measure of perceptual video quality) even as the number of clients watching the video increases from just one to over 20. This performance is comparable to unicast delivery to a single wireless client, which is considered to be near-optimal, and is significantly better than wireless broadcast.

An audience member observed that in the experimental evaluation, throughput decreases as more clients watched the video when multiple clients use unicast simultaneously for video delivery. Sen responded that because the link quality degrades with more clients, missed packets are not acknowledged. Michael Freedman said that this sounded like Scalable Reliable Multicast to him. Sen answered that they are working at the MAC layer, and the higher layers cannot do this.

- ***Maranello: Practical Partial Packet Recovery for 802.11***
  Bo Han and Aaron Schulman, University of Maryland; Francesco Gringoli, University of Brescia; Neil Spring and Bobby Bhattacharjee, University of Maryland; Lorenzo Nava, University of Brescia; Lusheng Ji, Seungjoon Lee, and Robert Miller, AT&T Labs—Research

Aaron Schulman presented a partial packet recovery protocol for 802.11 called Maranello. Partial packet recovery protocols try to repair corrupted packets, rather than retransmit complete packets, when an error is detected. This is done by retransmitting only the corrupted parts of the packet. Eliminating retransmission of unnecessary data can result in improved throughput and a better user experience. Maranello's goals are to provide partial packet recovery to 802.11 while maintaining compatibility with existing 802.11 deployments, be deployed incrementally, run on existing hardware, and require no extra bits for correct packets (since most packets will have no errors).

The Maranello system employs a block-based partial packet recovery technique. Packets are divided into 64-byte blocks, and checksums are computed over each block. If an error is detected, only the blocks containing errors need to be retransmitted. By retransmitting only the blocks that are corrupt, throughput increases because block repairs are faster than retransmissions, short block retransmissions have higher probability of delivery than larger whole-packet retransmissions, and early delivery avoids back-offs and low-rate retransmissions.

The block-based repair scheme is implemented in firmware so that it can be deployed incrementally on existing 802.11 cards. Relative to prior partial packet recovery designs, Maranello is the first to be implemented in commonly available firmware. Compatibility with existing 802.11 is achieved by having the receivers construct and send a negative acknowledgment (NACK) containing the block checksums before the transmitter decides to retransmit the entire packet. This ideally occurs after the short inter-frame space interval during which the transmitter expects an ACK. Thus, fast block-level checksums are required to meet this time constraint and the Fletcher-32 checksum is used for its speed.

Maranello was evaluated through trace-driven simulations and a real implementation using OpenFWWF and the b43 Linux driver. Simulation results indicate that successfully retransmitting earlier increases throughput. Also, an increase in throughput is confirmed, and packet recovery latency is reduced in a real deployment with other 802.11 hardware. Since most 802.11 implementations simply ignore NACKs, Maranello is able to fully interoperate with unmodified 802.11.

What is the size of the NACKs, since checksums are included in these packets? The NACKs can hold roughly 20 checksums, where each checksum is four bytes. Is it always possible to receive the NACK during the short ACK response window? As long as the data rate is greater than 12

Mbps, the NACK can be received during the normal ACK window. Are NACKs done during Clear Channel Assessment (CCA)? They are.

More information about Maranello, including firmware binaries (source code coming soon) can be found at http://www.cs.umd.edu/projects/maranello.

ROUTING

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

- ***Reverse traceroute***
  Ethan Katz-Bassett, University of Washington; Harsha V. Madhyastha, University of California, San Diego; Vijay Kumar Adhikari, University of Minnesota; Colin Scott, Justine Sherry, Peter van Wesep, Thomas Anderson, and Arvind Krishnamurthy, University of Washington

  Awarded Best Paper

Standard traceroute is one of the most commonly used tools for diagnosing network failures and anomalies. However, the asymmetry of paths on the Internet makes traceroute an imperfect instrument. For a datacenter operator attempting to diagnose problems seen by a client, a traceroute in the opposite direction may actually be necessary. Ethan Katz-Bassett presented the reverse traceroute project, which implements a novel technique for identifying the reverse path taken by a packet.

The reverse traceroute technique starts from the observation that routing on the Internet is based on destination. By maintaining an atlas of known Internet paths, the reverse traceroute tool only needs to stitch together the initial hops along the reverse path until it intersects one in the atlas. The initial hops in the reverse path are identified using the Record Route IP option, which is able to record the first nine routers encountered along a path. However, the host of interest is often further away than nine hops. To overcome this limitation, reverse traceroute uses other nodes as "vantage points," distributed on the PlanetLab testbed. By sending spoofed packets with the Record Route option from vantage points closer to the host of interest, it becomes possible to map the initial reverse hops.

Katz-Bassett presented an example where reverse traceroute was able to illuminate the cause of an inflated round-trip delay which had been noticed using normal traceroute. Correspondence with the network operators confirmed that reverse traceroute had identified a routing misconfiguration between Internap and TransitRail. Reverse traceroute can also be used to compute the latency along paths between arbitrary routers on the Internet. This ability was evaluated by comparing the calculated latencies with the published latencies of Sprint's inter-POP links. A Web interface to the reverse traceroute tool is available at http://revtr.cs.washington.edu.

Rodrigo Fonseca asked about packets with IP options being dropped when traversing the Internet and how that affects reverse traceroute. Katz-Bassett responded that support for IP options is improving and reverse traceroute uses alternate vantage points to try to get around networks which are dropping packets. Nick Feamster asked how they could take further advantage of the stability of paths in the Internet. Katz-Bassett noted that they are currently caching many parts of the path and are keeping an eye on other studies of the dynamics of paths in the Internet.

- ***Seamless BGP Migration with Router Grafting***
  Eric Keller and Jennifer Rexford, Princeton University; Jacobus van der Merwe, AT&T Labs—Research

Migrating a BGP router is currently a difficult but sometimes necessary process. The service it provides needs to be highly available and router changes generally must be coordinated with external organizations. Eric Keller identified the monolithic view of a router in which the hardware, software, and links are treated as one entity as the reason for this difficulty. By separating the components, Keller showed that it is possible to migrate the pieces independently without impacting the BGP session.

Router grafting requires migrating the four layers of a BGP session. The physical link can be moved by using a switchable, programmable network. The IP link is migrated by transferring the IP address, which, to routers in a BGP session, is simply an identifier. BGP relies upon long-running TCP sessions, requiring that the sequence numbers, queues, etc. be passed to the new router. Finally, the BGP state is migrated by dumping the existing configuration on the old router and importing it into the new router, while carefully tracking any updates that arrive during the migration. Keller presented a prototype implementation of router grafting for the Quagga software router and showed that the process is practical and transparent.

Keller was asked how this work relates to his group's previous work on migrating virtual routers (VROOM). He responded that this work is both complementary to and at a finer grain than the whole-router migration in VROOM. Nick Feamster asked if this technique can be applied to other routing protocols such as OSPF. Keller replied that migrating a link has a greater effect in OSPF installations because the shortest path may have changed and would need to be recomputed throughout the network.

DATACENTER NETWORKING

Summarized by Amin Tootoonchian (amin@cs.toronto.edu)

- ***ElasticTree: Saving Energy in Data Center Networks***
  Brandon Heller, Stanford University; Srini Seetharaman, Deutsche Telekom R&D Lab; Priya Mahadevan, Hewlett-Packard Labs; Yiannis Yiakoumis, Stanford University; Puneet Sharma and Sujata Banerjee, Hewlett-Packard Labs; Nick McKeown, Stanford University

Brandon Heller explained that ElasticTree aims to create an energy-proportional datacenter network from non-energy-

proportional components, because networking devices are not designed to be energy proportional. However, turning ports on/off affects their power consumption. ElasticTree compresses the network as much as possible based on the traffic by turning off unneeded ports/links and switches carefully in a scalable and resilient manner.

Input to the ElasticTree optimizer is the network topology, routing policy, power model, and the traffic matrix. The optimizer optimizes for power efficiency and later balances for fault tolerance and utilization. The solution is an active network subset and a set of flow routes. The rest of the talk was mainly about the three optimizers: formal optimization model (most optimal, least scalable), greedy bin-packer, and topo-aware heuristic (least optimal, most scalable). The optimizers come up with a single spanning tree, and the solution is modified to provide fault tolerance and a bound on utilization. ElasticTree was evaluated with simulations and using a running prototype with a Fat-Tree topology.

Costin Raiciu (University College London) raised the point that the graph shown for the traffic demand in a sample network was for the incoming traffic from outside, which does not necessarily reflect the inter-server communication pattern. Heller said they tried to capture that distribution by making some assumptions. Guohui Wang from Rice University asked about MapReduce workloads where all-to-all communications is necessary. Heller said their solution works regardless of workloads, but the underlying assumption is that the traffic demand does not need all the network capacity most of the time. Another attendee asked how ElasticTree would react to a sudden shift in traffic. Heller said that it depends on how fast switches/links can be turned on or put to sleep.

- ***SPAIN: COTS Data-Center Ethernet for Multipathing over Arbitrary Topologies***
  Jayaram Mudigonda and Praveen Yalagandula, HP Labs; Mohammad Al-Fares, University of California, San Diego; Jeffrey C. Mogul, HP Labs

Yalagandula started by arguing that today's datacenter networks require a low-cost flat network with a high bisection bandwidth. Ethernet, as has been pointed out by other recent works, is an appropriate choice: it is commodity, provides flat address space, and is self-configuring. However, having a spanning tree makes it not scale well. SPAIN uses commodity layer-2 switches and makes Ethernet scale under arbitrary topologies using multipathing via VLANs, and has end hosts spread load across VLANs.

SPAIN uses VLANs not for isolation, but for multipathing. Paths are computed and laid out as VLANs, and the end hosts choose a VLAN to send their traffic on. SPAIN wants to have as few paths as possible that provide enough reliability. The path computation module uses modified Dijkstra's shortest-path algorithm to find paths among edge switches with preference given to edge-disjoint paths (to provide higher bisection bandwidth). To reduce the number of VLANs, SPAIN merges the computed paths that do not form a loop. SPAIN was evaluated with simulations and a small testbed using Cisco datacenter, Fat-Tree, Hyper-X, and B-Cube topologies. SPAIN provides 1.6x–24x throughput improvement over STP in different topologies and, depending on the topology, may require a small or a large number of VLANs compared to the number of switches. Finally, the evaluation shows that incremental deployment of SPAIN is beneficial.

Amin Vahdat from UCSD asked about the assertion made in the talk about the deployability of VL2 and PortLand; he claimed that both are deployable today. Yalagandula argued that many data centers may end up buying cheap switches that do not provide features required by VL2. Also SPAIN works with arbitrary topologies (not only Clos or Fat-Tree). Albert Greenberg from Microsoft Research said that he had a paper about Monsoon (2008) which used VLAN tricks, but they moved away from Monsoon because of scalability and reliability: the L3 control plane provides built-in scalability and reliability which can be used. SRM said they are going to fix VLAN reliability issues using MSTP. Also, in the current solution, end hosts can work around the reliability concerns.

- ***Hedera: Dynamic Flow Scheduling for Data Center Networks***
  Mohammad Al-Fares and Sivasankar Radhakrishnan, University of California, San Diego; Barath Raghavan, Williams College; Nelson Huang and Amin Vahdat, University of California, San Diego

Mohammad Al-Fares said that datacenter workloads like MapReduce are often bottlenecked by the network. The standard approach is to use equal-cost multi-path routing (ECMP), but these routes are static and oblivious to link utilization. Hedera dynamically schedules flows under a dynamic traffic matrix in a network with any arbitrary topology. Hedera estimates demand and uses placement heuristics to find good flow-to-core mappings.

Hedera detects large flows, estimates their demand and pushes good paths for them into switches. For small flows, default ECMP load-balancing is efficient; therefore Hedera is complementary to ECMP. The placement heuristics considered are: global first-fit (where the first path which fits the flow requirements is chosen) and simulated annealing (probabilistic search for flow-to-core mappings). Hedera routes around failed components if a failure is detected, and in the case of the scheduler failure, default ECMP takes over. Evaluation on a simple testbed shows that simulated annealing outperforms global first-fit and is very close to the ideal case. Hedera also achieves close-to-ideal performance in a 120 GB all-to-all in-memory data shuffle. For larger topologies, simulations were used. They show that, again, simulated annealing is close to ideal, and Hedera is quite reactive to changes.

Someone raised the point that the power of two choices may significantly help to address the concern about ECMP. Al-Fares responded that static load-balancing is fundamentally flawed because it cannot satisfy dynamic traffic loads. Another attendee said large flows could be avoided in some scenarios, getting around the problems addressed in the paper. Al-Fares said the motivation was to provide a general solution without imposing any restriction on the traffic and applications. In a follow-up the questioner noted that because of small RTT in datacenter networks, the packets could be spread across multiple links and with the aid of a small reassembly buffer at the end-hosts, the problem could be alleviated. Al-Fares said they are currently working on that. Another attendee raised the point that if the number of flows is significantly larger than the number of paths, the distribution of flow sizes should not matter. Al-Fares said that they assume the traffic patterns obey the hose model. If the number of flows is far larger than the network can support, not much can be done. The point behind demand estimation is that the fair allocation can be found. David Anderson from CMU asked why the authors did not consider multicommodity flow routing with randomized rounding, since it is polynomial, fast, and reasonable. Al-Fares said that is part of their ongoing work.

IMPROVING MAPREDUCE

Summarized by Krishna Puttaswamy (krishnap@cs.ucsb.edu)

- ***Airavat: Security and Privacy for MapReduce***
  Indrajit Roy, Srinath T.V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel, The University of Texas at Austin

Indrajit Roy presented Airavat. The MapReduce programming model is increasingly used today for large-scale computing. Many users upload their data to companies, which third parties later mine to provide better services to their users. For example, users might contribute their healthcare data, which may be mined to research new drugs or design cheaper insurance policies. However, a common fear is that the third parties that mine user data may target a specific user's personal data. Simply anonymizing the data is not sufficient because attackers can combine anonymized data with external information and de-anonymize them.

Airavat attempts to solve this problem, running untrusted code on the original data and yet ensuring that users' privacy is protected. The Airavat framework runs on the cloud, which includes modified MapReduce, DFS, JVM, and SELinux to provide privacy guarantees. Untrusted mapper programs from third parties are run in this Airavat framework, and yet Airavat guarantees bounded information leak about any individual's data after performing the MapReduce computation. Airavat prevents sensitive data leakage via network/storage by using mandatory access control, and Airavat controls data leakage from the output of the computation using differential privacy and a small collection of trusted reducers. Although Airavat supports only a small set of reducers (SUM, COUNT, THRESHOLD), it can still be used to perform many interesting computations. The paper used four benchmarks to demonstrate this. The overhead due to the modifications introduced by Airavat was less than 32% in these experiments.

Fang Yu (MSR) asked if Airavat was subjected to side-channel attacks. For example, the running time of the mappers can reveal some information about the computation—if something runs for five hours, it can be used to suggest something about the data. Roy responded saying that probabilistic channels and timing channels are exploitable and Airavat cannot prevent all such leaks. However, Airavat attempts to mitigate these problems by limiting access to system resource to the mappers. For instance, untrusted code cannot access the correct system time, and hence cannot directly know the time taken for running the mappers. Andrew Ferguson (Brown University) asked why the JVM was modified in Airavat. Why not instead reboot JVM after each MapReduce computation? This would easily prevent access to the heap after the job, etc. Roy agreed that this was a good idea, but suggested that this leads to higher overhead in scheduling the jobs.

- ***MapReduce Online***
  Tyson Condie, Neil Conway, Peter Alvaro, and Joseph M. Hellerstein, University of California, Berkeley; Khaled Elmeleegy and Russell Sears, Yahoo! Research

Tyson Condie presented MapReduce Online. MapReduce is tuned for massive parallelism and batch-oriented computations over massive data. But MapReduce is often used for analytics on streams of data that arrive continuously. There are two domains of interest in particular—online aggregation and stream processing. However, the MapReduce model is a poor fit for such computations, since MapReduce only produces the final answer and cannot work on infinite streams of data.

MapReduce Online is a new model based on MapReduce that can operate on infinite data and export early answers. The implementation proposed in this paper, called Hadoop Online Prototype (HOP), also preserves Hadoop APIs and implements pipelined fault tolerance. HOP pushes data from mappers to reducers concurrently with operator computation, which the reducers reduce periodically to produce intermediate results. These intermediate results are then published to HDFS. The paper presented extensive evaluation to show the benefits of generating such early results. For example, in an approximation query example, HOP was shown to produce good results, with low error, that was close to the final result. HOP code is available at http://code.google.com/p/hop/ and more information about the project is available at http://boom.cs.berkeley.edu.

John Dunagan (MSR) asked why not run a small job with sampled input, and what benefits HOP provides beyond sampling. Condie responded that this approach provides the output for one sampled point. If the programmer is not

happy with the results, then new jobs need to be run with different samples. However, in the case of HOP the programmers can watch the results as they are produced and decide when to stop the job. Y. Charlie Hu (Purdue) asked if HOP helps in resolving the problem of tuning and setting the right values to the myriad configuration parameters in Hadoop, so that an ideal set of parameter values can be assigned for executing a given job. Condie responded that HOP may not help in setting all the parameters, but it does help with setting the block size parameter in Hadoop.

WEB SERVICES 2

Summarized by Amin Tootoonchian (amin@cs.toronto.edu)

- ***The Architecture and Implementation of an Extensible Web Crawler***
  Jonathan M. Hsieh, Steven D. Gribble, and Henry M. Levy, University of Washington

Jonathan M. Hsieh presented a system named XCrawler. XCrawler is an extensible crawler service where consumers can set filters and only the documents matching the filters are delivered to the consumers. Hence, XCrawler develops an approach to outsource the resource-intensive crawling tasks of the consumers. The key insight behind XCrawler is to separate the act of crawling from application-specific tasks. This makes XCrawler a shared service that many consumers can use. This work focuses on providing flexibility for the consumers to specify the filter, making the crawler scalable to support many documents and filters, and achieving low latency to crawl even real-time data.

After presenting the core design, Hsieh also presented several optimizations that improved the scalability of XCrawler significantly. They evaluated XCrawler with three applications: a copyright violation detector, a malware detection application, and an online identity service.

Zhichun Li (Northwestern) asked if XCrawler crawled dynamic content, and if they considered using a complex language with more features for designing the filter language. Hsieh responded that XCrawler didn't support dynamic content or complex language as of now. But it would be possible to extend XCrawler to support these features. Stefan Saroiu (MSR) asked if they planned to extend XCrawler to supporting more general query types to enable future applications that may do more processing on the index rather than simply running regexes, and how one would refactor XCrawler to support such applications. Hsieh observed that their goal was to reduce the number of documents delivered to the consumer. The heavyweight processing of future applications can be done locally by the consumer on a small set of documents they receive on their own machines.

- ***Prophecy: Using History for High-Throughput Fault Tolerance***
  Siddhartha Sen, Wyatt Lloyd, and Michael J. Freedman, Princeton University

There has been significant work on BFT protocols in recent years, but the throughput of these systems is still low. In an effort to improve the throughput of Internet services that have a read-mostly workload, Siddhartha Sen presented Prophecy, which aims to extend the middleboxes that already exist in Internet services (such as load balancers) to act as trusted proxies. With this trusted proxy, Prophecy shows that it is possible to get a nearly fourfold improvement on the throughput of these systems. With a measurement study, the authors also show that many Internet services have mostly static workload (90% of requests are for static data in the Alexa top-25 Web sites) and hence can benefit from Prophecy.

The traditional BFT protocols provide consistency guarantees called linearizability, but provide low throughput. A way to improve the throughput for read workload is to store a collision-resistant hash (called a sketch) of the response on the servers, ask only one of the servers in the replica group to send the actual response, and ask the others to send only the sketch. The client can then compare the response with the sketches to decide whether the response is valid. This optimization means that only one of the replicas (not all) in the replica group needs to execute the request. A traditional BFT system with this optimization is called D-Prophecy. However, the problem with this approach is that the throughput can still go down when the session length that must be established between the client and the replica server is short. To resolve this issue, Prophecy introduces a sketcher (trusted proxy) that establishes the appropriate sessions with the replica servers, but the client only needs to maintain a long session with the sketcher. This leads to improved throughput. Sen also talked about how to scale Prophecy with multiple sketchers and presented evaluation results that showed Prophecy's performance for various amounts of reads in the workload.

Ryan Peterson (Cornell) asked why one should trust the proxy if one is not willing to trust the servers. Sen responded that many services already have semi-trusted middleboxes, and they can be easily extended with a small amount of code to act as the proxy. Peterson then pointed out that the proxy is a single point of failure, as the attackers can attack the proxy and start sending arbitrary responses. Sen agreed that this is a problem and if one is not willing to place this trust, then the D-Prophecy model is the most appropriate. Adding more proxies would also help, but then it only helps with fail-stop failures. Miguel Castro (MSR) asked whether not having an authenticated connection between the client and the proxy means that an attacker can inject arbitrary data. How does Prophecy deal with it? Depending on the application, Sen said, there should be some application-level security to deal with such issues. Some applications,

such as banking, need stronger mechanisms while other applications, such as Facebook, might be able to work with cookie-based authentication. Castro next asked why not use cookie-based authentication (and get rid of the sessions) between clients and the replica servers, and also eliminate the proxy? Sen pointed out that such a design would be a problem when there are larger replica groups, since the client will have to establish a session with each of the replicas.

MALWARE

Summarized by Adam Bender (bender@cs.umd.edu)

- ***Carousel: Scalable Logging for Intrusion Prevention Systems***
  Vinh The Lam, University of California, San Diego; Michael Mitzenmacher, Harvard University; George Varghese, University of California, San Diego

Millions of "interesting" network events, such as anomalies, routing changes, and packet drops, occur at a high rate. Network operators want to log such events, but any logging mechanism is often constrained, in terms both of memory and of bandwidth. The standard solutions for dealing with these limitations are sampling and calculating online summaries, which do not capture a complete event log. Terry Lam presented Carousel, a system for efficient, nearly complete event logging. While presented in terms of networking, specifically logging packet source addresses, Carousel has a wide variety of applications.

Lam presented the following model: packets from *N* sources are arriving at rate *B*. The router has a buffer that can store M bytes and write to a remote storage "sink" at a rate *L*. In many scenarios, *L* is much less than *B*, and *M* is much less than *N*. After describing why standard solutions are inefficient for complete event logging (they are a factor of $\log_n(N)$ worse, due to the coupon collector's problem), and why Bloom filters do not help, Lam also showed how an adversary can cause such collection never to terminate.

Lam then described the design of Carousel, which, given these constraints, eventually logs all events (given that the events are persistent) in nearly optimal time. Carousel stores packets in a buffer and employs a Bloom filter to prevent duplicate sources from being stored in the buffer. Carousel "colors" the packets and only logs packets of one color at a time. Thus only certain packets are added to the Bloom filter, preventing it from becoming full. After some time, the Carousel "rotates" to log packets of a different color and clears the Bloom filter. Lam presented an adaptive method of determining the right number of colors. Carousel has three components: partitioning the sources into colors (using the lower *k* bits of a hash function), iterating through the partitions every *M/L* seconds, and then monitoring the Bloom filters to adapt *k*.

The authors implemented Carousel in Snort and performed simulations of worm outbreaks; Carousel is nearly 10 times better than the naive solution of logging all unique sources. Lam also discussed an implementation in hardware. Because Carousel is probabilistic, it may miss some sources, and it relies on the assumption that events repeat, so that a missed event will eventually recur and be logged later. However, Carousel is within an expected factor of 2 of an optimal solution.

What did Lam mean by a "full" Bloom filter? A Bloom filter is full if its false positive rate is above some threshold. Zhichun Li asked how Carousel could be used to log distinct events from the same source, and Lam referred him to the paper for details on such experiments. Aditya Akella asked whether they considered other forms of scaling, such as using multiple snort boxes in parallel. Lam said they wanted solutions that would be efficient in both software and hardware.

- ***SplitScreen: Enabling Efficient, Distributed Malware Detection***
  Sang Kil Cha, Iulian Moraru, Jiyong Jang, John Truelove, David Brumley, and David G. Andersen, Carnegie Mellon University

Sang Kil Cha presented SplitScreen, a system for efficiently scanning a collection of files for malware. Traditional malware detection uses exact signature-matching on files. However, the number of signatures is growing exponentially (ClamAV had over 500,000 in 2009), requiring more memory and time to scan, as well as increasing the rate of cache misses. SplitScreen's insight is that it is very likely that only a small portion of the signature database is necessary to scan a set of files; this is suggested by a study of CMU email traffic that shows that less than 1% of all the signatures are needed to find all malware in the messages (in fact, 80% of malware can be caught using only five signatures).

To take advantage of this assumption, SplitScreen uses a feed-forward Bloom filter to reduce both the number of files that need to be tested and the number of malware signatures to test against. The feed-forward Bloom filter is initialized by taking a fixed-length (*n*-byte) segment from each signature and placing it into the all-patterns Bloom filter (APBF). SplitScreen takes a rolling hash of each *n*-byte segment of the target files and tests to see if any are in the APBF. If so, the file is placed into the reduced file set, and the matched bits of the APBF are copied into the matched-patterns Bloom filter (MPBF), which is the same size as the APBF. After scanning the files, SplitScreen iterates through all patterns to see which are present in the MPBF; these patterns compose the reduced signature set. SplitScreen then does exact matching on the reduced file set and reduced signature set. SplitScreen's Bloom filters are split into cache-resident and non-cache-resident parts and only access the latter when there is a hit in the former, making them more cache-efficient than classic Bloom filters. Cha presented results that show that SplitScreen has higher throughput (2x–4x) and fewer cache misses than ClamAV.

Another benefit of SplitScreen is that it can be run on smaller devices.

SplitScreen's reduced memory usage helps in this regard, but in this scenario propagating the large signature database is expensive. Cha showed how SplitScreen can be used in an on-demand way to allow low-power devices to perform malware checks: a server creates the APBF and sends it to clients; this is much smaller than the entire signature database. Clients then send the MPBF back to the server, which produces the reduced signature set and sends it to the client for pattern matching.

Seung Yi asked why the authors assume that only a small number of filters are necessary to match against all malware that may exist in a collection of files. Cha responded with evidence collected from two studies: one of a university email trace, in which less than 1% of signatures was needed to match all malware present, and a trace of 300 GB of malware, which required 20% of the signatures to find all samples. Wyatt Lloyd asked if SplitScreen was robust to a SplitScreen-aware adversary which uses old malware that is not in the SplitScreen database. Cha responded by saying that even if an attacker creates malware that matches many signatures, SplitScreen outperforms ClamAV. Yan Chen asked how SplitScreen transforms a signature into an entry in the Bloom filter, and David Andersen, a co-author, said that SplitScreen picks a fixed-length fragment of each signature. Stefan Savage asked how SplitScreen and other lazy-evaluation virus checkers can obtain new signatures when they do not have a network connection. Cha answered that SplitScreen is efficient even when running on a single host. Aditya Akella asked if the authors had thought of any other applications of feed-forward Bloom filters and cache-efficient designs. Cha said that he had not. The SplitScreen source is available at security.ece.cmu.edu.

- ***Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces***
  Roberto Perdisci, Georgia Institute of Technology and Damballa, Inc.; Wenke Lee and Nick Feamster, Georgia Institute of Technology

Roberto Perdisci presented work on observing and characterizing the network-level behavior of malware. While malware authors can evade antivirus by using obfuscation techniques, network properties are more difficult to hide: bots still need to contact a command and control (C&C) server, send spam, exfiltrate information, or download binaries. This suggests that a network-based approach can distinguish between legitimate traffic and malware traffic and thus identify compromised hosts in a network. The authors focused on HTTP-based malware, since it is growing in popularity and can bypass firewalls. The authors aimed to generate malware detection signatures. To reduce the number of signatures, they first classified malware into families based on network behavior. Network behavior was observed from HTTP traces of a large collection of malware samples. Then, the authors used a three-step clustering process to classify malware into families. The first step uses statistical features, such as the number of GET and POST requests and URL lengths, to form coarse-grained clusters. The second step employs structural characteristics to determine the difference between malware traces and uses this distance to split coarse-grained clusters. In the final step, the authors merge close clusters to recover from possible mistakes made in previous steps. The final step finds the centroid of each cluster using a token subsequences algorithm and merges clusters whose centroids are close. Once clustering is complete, the authors generate signatures from each cluster using a modification of the token subsequences algorithm.

To evaluate their techniques, the authors collected 25,000 samples of malware over six months. Perdisci presented results of an evaluation on a one-month subset that contained nearly 5,000 samples and from which the authors' technique produced 234 malware families with two or more samples (they discarded singletons). From these families, the authors generated 446 signatures over the course of eight hours. These signatures were able to detect a significant percentage of malware in separate traces collected in later months. The signatures also detected half of the malware that host-based antivirus products missed and had no false positives. Perdisci showed that the same signatures perform significantly better than those generated from a single-step clustering process or the latest host-based clustering technique.

Zhichun Li asked if Perdisci had examined types of traffic other than HTTP. Perdisci said this is part of his future work, but that HTTP was the most important type of traffic in enterprise networks. Had a malware author obfuscated network behavior to interfere with the presented technique? This was a very hard problem to solve; it might help to combine their technique with traditional anomaly detection techniques or thorough binary analysis. Yinglian Xie asked to what degree signature generation depended on clustering accuracy. Perdisci replied that correct clustering is critical. He also mentioned a signature pruning process that removes signatures that are likely to produce false positives. Areej Al-Bataineh asked what safety measures were used when running malware and if they have statistics on specific families of malware. Perdisci said they put an IPS in front of the infected host and did not focus on a specific family.

NETWORK PERFORMANCE

Summarized by Amin Tootoonchian (amin@cs.toronto.edu)

- ***Glasnost: Enabling End Users to Detect Traffic Differentiation***
  Marcel Dischinger and Massimiliano Marcon, MPI-SWS; Saikat Guha, MPI-SWS and Microsoft Research; Krishna P. Gummadi, MPI-SWS; Ratul Mahajan and Stefan Saroiu, Microsoft Research

Marcel Dischinger presented Glasnost, a tool that enables network users to detect traffic shaping. So far, half a million users from different countries and telecommunication regulators have used Glasnost. Designing Glasnost was challeng-

ing, because tests must be easy to use, short, and accurate. Glasnost performs active measurements in a controlled fashion; it compares the performance of two flows, the first flow emulating a realistic application (e.g., BitTorrent) traffic and the second one just varying the payload. Glasnost is able to detect port-based and content-based traffic shaping.

Glasnost runs on 20 servers on nine sites worldwide and is a part of Measurement Lab. Results collected from tests run by users show that 10% of BitTorrent tests indicated rate limiting over the 18 months of deployment. They also show that: major ISPs do rate-limit BitTorrent traffic. Rate-limiting is more common in the upstream direction and is based on both packet port and content; some ISPs only rate-limit some users or at peak hours. Finally, Glasnost enables automatic test construction using traffic traces.

Ming Zhang from Microsoft Research (Redmond) asked how they calculated the number of false positives and false negatives. They are using information from regulators (especially from Canada) as ground truth. How can the design be augmented to have it working in the case that ISPs detect Glasnost tests (i.e., treat Glasnost servers differently)? Centralized design is prone to whitelisting by ISPs, but having a decentralized design for Glasnost was challenging. The authors tried to make it hard for ISPs to detect tests by making the code available (so that they could run it on their own servers) and enabling users to make their own tests. Finally, Dischinger said that ISPs may choose not to take such actions, however, because it is bad for their reputation if they are caught.

- ***EndRE: An End-System Redundancy Elimination Service for Enterprises***
  Bhavish Aggarwal, Microsoft Research India; Aditya Akella and Ashok Anand, University of Wisconsin—Madison; Athula Balachandran, Carnegie Mellon University; Pushkar Chitnis, Microsoft Research India; Chitra Muthukrishnan, University of Wisconsin—Madison; Ramachandran Ramjee, Microsoft Research India; George Varghese, University of California, San Diego

Ram Ramjee argued that the middlebox approach to redundancy elimination does not work well in some scenarios: e.g., where end-to-end encryption is used, or in scenarios where the bottleneck is not the WAN (e.g., wireless links). EndRE takes an end-to-end approach to redundancy elimination to address these issues. Additionally, an end-to-end approach helps to save energy and can operate above TCP, which translates into latency gains.

EndRE's design goal is to opportunistically use the limited end-host resources: it is lightweight, fast, and adaptive; reduces memory overhead; and simplifies client decoding. EndRE proposes a fingerprinting technique called SAMPLEBYTE which is both fast/adaptive and robust. EndRE's evaluation shows that SAMPLEBYTE is an effective technique. Furthermore, analysis of 11 enterprise traces suggests that the median memory requirement of a client over 44 days is 60 MB, and bandwidth savings over two weeks was 26–34%, which, in turn, translates into notable energy savings. Finally, Ramjee concluded that EndRE is a promising alternative to WAN optimizers.

There were no questions.

- ***Cheap and Large CAMs for High Performance Data-Intensive Networked Systems***
  Ashok Anand, Chitra Muthukrishnan, Steven Kappes, and Aditya Akella, University of Wisconsin—Madison; Suman Nath, Microsoft Research

Ashok Anand argued that today's data-intensive networked systems require a cost-effective, cheap, and large hash table. He said using Flash SSDs is a good option because it is cheap and has a high read performance (10k random reads/sec), but is slow at writes. To deal with that they proposed a new data structure: BufferHash on flash which batches writes and performs sequential lookups (2x faster than random writes). Also, bloom filters are used to optimize lookups.

Their design has a two-level memory hierarchy, DRAM and Flash. To avoid unnecessarily going to flash during lookups, they use an in-memory bloom filter. To boost updates and insertions, they go to DRAM first and DRAM is flushed to flash eventually. Their evaluation suggests that BufferHash significantly outperforms traditional in-memory/on-disk hash tables, is best suited for FIFO eviction policy, and significantly improves the performance of WAN optimizers.

Jeff Mogul mentioned that phase change memory has different characteristics from Flash, and asked how much of their design would be useful with PCM. Anand replied that as long as there is asymmetry in read and write performance, their approach will work. Miguel Castro asked about a comparison to Berkeley DB that appeared in the paper, and how it was configured. Anand said it was configured the same way as CAM. Someone else commented that perhaps BDB needed to be rewritten for Flash.

LEET '10: 3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats

April 27, 2010
San Jose, CA

KEYNOTE ADDRESS

- ***Why Don't I (Still) Trust Anything?***
  Jeff Moss, Founder, Black Hat and DEF CON

  Summarized by Rik Farrow (rik@usenix.org)

Jeff Moss began by visually describing his presentation style. He showed slides that compared the approaches of Steve Jobs and Bill Gates when making presentations, with the Gates version a confusing (but symmetric) display of dialog boxes. By comparison, Jobs' slide was austere.

Moss was introduced to computers, and shortly thereafter, the world of dialup bulletin boards, in the '80s. He went on to found DEFCON, and later the very profitable Black Hat series of conferences. Currently, Moss is a member of the Homeland Security Advisory Council (HSAC), a volunteer position that he takes quite seriously.

As his first example of why he doesn't trust anything, Moss pointed out that we still don't have working email security. He displayed the headers from an email allegedly secured using TLS, but in reality, TLS was only used by the last forwarding server. PGP has proven to be too hard for most people to use, and also does not have perfect backward security: if you lose your secret key, all of the past email encrypted with that key can be decrypted as well.

Moss then mentioned that he had started keeping his passwords on a "secure" USB fob. Someone bet that he could recover the keys, and by carefully removing the layers from the chips in the fob, was indeed able to read out the bits in storage, despite the manufacturer's attempt at making this difficult to do.

Moss does like some Firefox add-ons, such as NoScript and Certificate Watcher, which led Niels Provos to ask, "Do you trust your add-ons and plug-ins?" Moss had no answer for that, and went on to point out that you have hundreds of certificate authorities but no way to add or delete them from your browser. Then he wondered why HTTPS is not used more, and Nick Weaver answered that HTTPS adds several round trips, each with delays of 300 ms or more.

After making several more points, Moss asked for questions. Fabian Monrose asked what Moss thought about DNSSEC; Moss thought it might be feasible in five years. Weaver commented that DNSSEC might be good for trust anchors, but has a much more limited chain of trust. Moss commented that you could serve your public key via DNSSEC, but soon DNS answers will grow larger than 8k. Michael Bailey asked what might be a solution. Moss answered that vendors have no reason to include good security in their products, with Google being somewhat of an exception. I asked about HSAC, and Moss said the council is supposed to guard against group think in the government. They meet and render opinions publicly.

BOTNETS

Summarized by Chris Kanich (ckanich@cs.ucsd.edu)

- ***Tumbling Down the Rabbit Hole: Exploring the Idiosyncrasies of Botmaster Systems in a Multi-Tier Botnet Infrastructure***
  Chris Nunnery, Greg Sinclair, and Brent ByungHoon Kang, University of North Carolina at Charlotte

Chris Nunnery and his co-authors acquired packet traces and disk images from machines among the two uppermost tiers of the Waledac botnet's command and control (C&C) infrastructure. While the infected hosts which make up the lower tiers of the Waledac botnet are currently well understood, until now the upper tiers of the botnet's management infrastructure were not.

The C&C infrastructure of the Waledac botnet consists of two tiers: an upper tier server (hereafter referred to as a UTS) and usually six second-level servers (referred to as TSLs). The infected hosts of the lower layer consist of spammer nodes which perform the grunt work of sending the botnet's spam payloads, and repeater nodes which forward communication between the upper layer and the spammer nodes. For quite some time, the TSLs were considered purely an obfuscation layer to hide the location and identity of the UTS, however, this work shows that while the TSLs do provide obfuscation for the UTS, the TSL servers provide several other services to the botnet and botmaster. The UTS performs all centralized, autonomous C&C functionality by hosting the zombie binaries and interacting with the spam template provider.

This work exposes several facets of the Waledac botnet's operation. First, an extensive auditing system exists whereby the UTS can audit the operation of the repeater nodes, either by testing the ability to serve a file or resolve a specially coded domain name to the address of another repeater node. In addition to this auditing system, the UTS keeps logs of commands issued through the repeaters as well as Fast Flux domain uptime; Nunnery remarked that the activity of researchers interacting with the botnet would likely have been recorded.

The researchers discovered that the Waledac developers compile the malware binaries using different affiliate IDs so that different groups can propagate and profit from installing Waledac on victim machines. The repacking of these binaries is outsourced to a site called j-roger.com, a process which takes about four seconds. Records recovered by the researchers saw 157 binaries repacked in a two-hour window.

The Waledac botnet employs a differential spamming platform, propagating both "high quality" (HQS) and "low quality" spam (LQS). High quality spam uses stolen SMTP authentication credentials and a SOCKS proxy, either rented from a third party or on a disjoint subset of compromised machines. The TSL machines send HQS: first they grab the spam target list and template from Spamit (a spam affiliate network) via the UTS, then send the email to both the intended recipients and test accounts at various free email providers in order to test the effectiveness of the spam run. In contrast, the LQS bulk email blasts direct from the Spammer nodes are not sent to test accounts, but successful delivery to the MTA is recorded.

Asked what packer Waledac uses, Nunnery said originally the UPX packer was used, but eventually they moved to a custom repacker. When asked about indications of accounting or cost differentiation/billing statistics, no direct evidence has been found, but there are some figures related

to price structure outlined in a file recovered from the file system image. What was the reliability of these machines, and could the botmaster move C&C to EC2? The machines were rented from LeaseWeb and had 99% uptime. Had the botmasters noticed the research group? They certainly had, and there was direct contact between the two parties.

- ***Insights from the Inside: A View of Botnet Management from Infiltration***
  Chia Yuan Cho, University of California, Berkeley; Juan Caballero, Carnegie Mellon University and University of California, Berkeley; Chris Grier, University of California, Berkeley; Vern Paxson, ICSI and University of California, Berkeley; Dawn Song, University of California, Berkeley

Chia Yuan Cho explained the spamming perpetrated by the MegaD botnet, which at its peak was responsible for one-third of all spam and currently sends 15% of worldwide spam. The botnet survived a takedown attempt in November of 2009. MegaD malware binaries have their command and control (C&C) master server address hardcoded, and upon bootup the master server will inform the bot of its template server from which to acquire spam templates, a drop server for binary updates, and an SMTP server for spam testing.

As different binaries connect to different master servers, this raises the question of finding binaries belonging to different master servers. The researchers used a Google hack to find different master servers and "milked" subsequent binary updates using C&C emulation. As the master servers use TCP ports 80 and 443 and imitate a Web server when responding to a simple Web client request, a Google query can be formed based on the C&C server's response in order to find new C&C servers. Once a new C&C server has been identified, knowledge of the MegaD C&C protocol allowed construction of benign programs which could request new versions of the MegaD malware binary.

After the November 2009 takedown, only one template server survived, and the templates it served stayed the same for a week. After that week, templates pointed to a new test SMTP server; 16 days after the takedown, spam was again being delivered at the full pre-takedown rate. Two possibilities exist as to how this was possible: either reanimation of servers redirected the currently active bots to new C&C servers, or fresh installs repopulated the spamming tier of the MegaD botnet. The former can be ruled out as FireEye (the company that initiated the takedown) reported that none of the C&C servers were still available post-takedown. The latter remains the sole likely possibility, especially as MegaD is known to use generic downloader malware to propagate via a Pay Per Install model.

Another large facet of this work identified two different master server groups within the greater MegaD population. The researchers' hypothesis that these groups are operated by two different botmasters is supported by many facets of the operation of these disjoint subsets. While each group contains several master servers, one group would use different subsets of the template elements available within the spamming engine; the templates themselves displayed different types of polymorphism and greatly differed in how often the template was updated on the server side. One group used only a Viagra-themed spam campaign, while the other group used a diverse set of spam campaigns not limited to pharmaceuticals. This work was supported by four months of infiltration of the botnet, consisting of milking both the C&C and template servers, Google hacking to find additional servers, and discovering the existence of two disjoint and differently managed botnets under the name MegaD.

Cho was asked why, since C&C servers were hardcoded into MegaD binaries, couldn't simple C&C server takedowns neuter the botnet. Vern Paxson responded that the botmaster's counter-solution is just to use the Pay Per Install model to repopulate his spamming botnet cheaply and effectively. Niels Provos asked what the minimum number of bots would be to run a successful spam campaign, and although there was no answer, the data point of 20,000 bots within the Storm botnet as an upper bound was brought up. How long does it take to build up a spamming botnet from scratch? Since 16 days was enough to return to full speed, how does one actually hurt the spammers' bottom lines? One suggestion was to clamp down on domain registration or find other ways to hurt them monetarily on the back end. Fabian Monrose asked if there are any positive takeaways from this work, and while no direct advantages could be enumerated, the suggestion was raised that concentrating on the economics of the spamming trade might prove fruitful. Can one use templates themselves to infer botnet membership? The answer was yes; work by Pitsillidis et al. on botnet Judo was brought up as an example of a similar strategy based on this intuition.

THREAT MEASUREMENT AND CHARACTERIZATION

Summarized by Srinivas Krishnan (krishnan@cs.unc.edu)

- ***The Nocebo Effect on the Web: An Analysis of Fake Anti-Virus Distribution***
  Moheeb Abu Rajab, Lucas Ballard, Panayiotis Mavrommatis, Niels Provos, and Xin Zhao, Google Inc.

Moheeb Rajab explained that the focus on Web malware at Google has been on delivery mechanisms. Since drive-by downloads have been getting more and more attention, malware distribution networks have been looking at other delivery mechanisms, with fake antivirus as an example of this evolution. The attack is simple: an HTML pop-up appears on the user's screen with an image of an antivirus engine scanning the user's system. The user is then informed he has malware on his machine and to remove it they need to download the "antivirus" software by clicking on a link. Once the user clicks on the link, the malware is downloaded and installed on the system. The malware then acts as an

annoyance, a keylogger or as ransomware, asking the user to pay for the removal of the malware from their system.

Rajab said that this attack plays on the user's fear and allows the attacker to directly monetize the malware. The statistics breakdown shows over 35 million downloads a month and Google's malware tracking system reveals a 3% increase over the past year. Furthermore, fake AV accounts for approximately 15% of Web malware. The authors used the anti-malware infrastructure they built at Google which analyzed 240 million URLs with a 20% sampling rate. The authors also reported that 11,500 domains were involved in the fake AV distribution over the past year, where the rate of increase grew from 90 to 600 domains per week.

The attackers attract users by commonly used techniques: Web and email spam, and exploit ads. There was also a new increase in event-driven exploits based on Google trend keywords; over 70% of malware domains that used the event-driven exploit were also fake AV distributors. The median lifetime of the domain decreased from 1000 hours to little less than an hour, in response to Google's better detection time.

Abhishek (McAfee) asked about how Google protects users: are the domain names reported back in search results? Rajab responded that the results are tagged; users are protected as Google's anti-malware system's detection time goes down. Vern Paxson (ICSI, UC Berkeley) wondered about the economic aspect of the fake AV attacks. Rajab said that they did not look at it, and co-author Niels Provos said that the money trail ends at the bank, so it's hard to get an actual idea of the scale of the fake AV economy. Roberto Perdisci (Georgia Tech) pointed out that fake AVs are persistent; what is the behavior after they are installed on the system? Rajab said they did not do the binary analysis. Jack Stokes (Microsoft Research) suggested that analysis might be improved by involving users. Rajab said that, currently, users can report suspicious URLs, but there is no direct way to tag a URL. Niels Provos pointed out that user input is noisy and poor, based on experience, but further investigation is warranted.

- ***Spying the World from Your Laptop: Identifying and Profiling Content Providers and Big Downloaders in BitTorrent***
  Stevens Le Blond, Arnaud Legout, Fabrice Lefessant, Walid Dabbous, and Mohamed Ali Kaafar, I.N.R.I.A, France

Stevens Le Blond presented the work they did on quantifying user contribution in BitTorrent and if it was possible to identify the content providers. The authors looked at injection and download rate at a large torrent index site (Pirate Bay). They would connect every minute and get the recent uploads and track the respective torrents. They also checked who the first few uploaders for each torrent were and collected their IDs. Using this method, they were able to identify 70% of the content providers. Furthermore, they studied 10,000 IP addresses and classified them as either middleboxes, spies, or monitors. The key insight was that most of the content in torrent networks is contributed by a few content providers and that there is a constant monitoring and spying presence on these networks.

Niels Provos asked if identification accuracy could be based on multiple parties using the same techniques to hide, leading to possible skewing of the results. Le Blond replied that cross-verification of the results shows their results to be 99.9% accurate. Eric Ziegast asked if they saw any differences between music, movies, and code? Le Blond responded that they saw no direct correlation between contents and size. The biggest provider was Easy TV, injecting six new TV shows every day, roughly 500 MBs per TV show. Those providers are using the same machine. We have looked at type of content to see what people were injecting. Ziegast then asked if they saw much malware, and Le Blond said no. Provos asked if Pirate Bay attempts to block peers that appear to be spying, and Le Blond said they do, but you can spy for 50 days before they notice. Provos said that is horrible detection performance.

- ***WebCop: Locating Neighborhoods of Malware on the Web***
  Jack W. Stokes, Microsoft Research; Reid Andersen, Christian Seifert, and Kumar Chellapilla, Microsoft Search

Jack Stokes summarized the work Microsoft has been doing in malicious page detection. The approach is different from other Web malware detectors, as the researchers use a bottom-up approach. The detection engine directly goes to the malware site and then uses a Web crawler to build a Web graph based on the links on the site. The Web graph is used to find the intersection between search results and the links on the malware site by overlaying the topologies. This approach is based on targeted detection of the malware found on a user's system, taking advantage of Microsoft's Anti-Malware (AM) tool.

Vern Paxson asked how you know if a URL is infected. Stokes responded that a human is analyzing and labeling it. Paxson asked how telemetry finds things that crawling does not. Crawling cannot tell if it's malware, but WebCop can tell. (There is room for improvement here, as WebCop does not download binaries and run instances in VM, so cannot tell if malware is real or not.) Fabian Monrose asked if every file that a user downloads is reported to Microsoft. Stokes replied, yes, if AM reports it as unsigned or it has a bad hash. Monrose said he was surprised, and he wondered if users know about this. Stokes said if it's free, you cannot opt out. If you pay money for it, you can opt out. It's in the privacy statement. By this point he was yelling and the room was getting worked up. Stefan Savage pointed out that this practice of collecting data from AV is truly industry-wide. Honeypots no longer work. The only way to deal with this is to use clients such as your sensornet. Monrose asked if we have a responsibility as a community to report this. Jeff Williams, the head of the WebCop group, said that they don't collect personally identifiable information or save the IP addresses.

INVITED TALK

- ***Naked Avatars and Other Cautionary Tales About MMORPG Password Stealers***
  Jeff Williams, Microsoft Malware Protection Center

  Summarized by Rik Farrow

Williams said that working at the Malware Protection Center provides a unique insight through protecting one hundred million Hotmail users. His group also pushes out the monthly update to anti-malware (AM). "We get to see where in the world things are, where they come from (Web, email, etc.)."

In this talk his focus was on password stealers. Williams pointed out that, contrary to what you might think, big targets are games and game discussion sites. A game password provides access to avatars as well as anything they have won or collected, and there is a thriving black market in gaming plunder. Also, law enforcement has little interest in protecting people from threats within games, even though the gaming market is on the verge of overtaking film in revenue.

When Williams mentioned that Brazil has a very high number of password stealers, Niels Provos asked why this might be so. Williams said there was no particular reason why, and Provos pointed out that there is no native AV industry in Brazil. Another person mentioned that liability laws are different there as well. Williams agreed, and quipped that Brazilians do more online banking than most other peoples—they just don't use their own accounts.

Williams described eight families of malware used to steal gaming passwords, some of which focus specifically on gaming kiosks. He mentioned the Dogrobot malware which led to kiosk owners wiping and re-installing machines every night to protect customers, and their business.

Williams then laid out the gaming black market:

Envelopes—stolen account info

Stalls—online space for collecting/selling information

Trojans—trojans and malware can be made to order, even online

Trojan generators—software that customizes trojans

Williams finally approached the title of his talk, showing a girl in a bathtub. Following a related link led to the installation of Agent.ABHN and Alureon.BJ. Williams said that in April 2009, there were 860,000 sites containing malicious scripts, leading to exploits against IE, Windows, and third-party ActiveX controls. Currently, Taterf is the most prevalent malware, designed to capture multiple game passwords. Taterf often includes other exploits and malware.

Protecting games is difficult because gamers focus on performance, and AV hurts performance. Williams does something like what Blizzard (World of Warcraft) has done: distributing and encouraging the use of hardware tokens for authentication.

Williams ended with a call to action: attackers have community, defenders should too to advance the state of community-based defense. He pointed to the Conficker working group as an example of a strong model, with cross-industry participation including AV, ISVs registrars, researchers, law enforcement, and many others. Williams said that software has bugs and traditional quality assurance doesn't find them, something I certainly have no trouble believing some eight years after Bill Gates announced "trustworthy computing."

Nick Weaver pointed out that hardware tokens don't work for banking (as malware can use a proxy to abuse authentication), but supposed that this wouldn't be true for gaming because it takes time to sell the loot. Williams countered that shorter timeslices have occurred in gaming crime as well. Stefan Savage wondered if they had dug deeply into the monetization side: where is the value coming from? Had they talked to game designers about reversal of loot loss? Williams said they hadn't, but it would be terrific if this could be done. Someone asked if they had a breakdown of the different types of accounts stolen, and Williams said that they didn't have hard statistics, but that it was not uncommon that a gaming password would work for banking as well.

THREAT DETECTION AND MITIGATION

Summarized by Chris Kanich (ckanich@cs.ucsd.edu)

- ***On the Potential of Proactive Domain Blacklisting***
  Mark Felegyhazi and Christian Kreibich, International Computer Science Institute; Vern Paxson, International Computer Science Institute and University of California, Berkeley

Mark Felegyhazi began his talk with a graph from previous work showing the time of register and time of first use for domain names used in botnet spam campaigns, noting that most spam domains were registered in large batches on the same day. Using the intuition that the scale of the spammers' operation requires batch registration of domain names, this work aims to proactively blacklist domains when they have likely been registered by a spammer before most of the batch has ever been seen in spam. The authors cluster using properties of the nameserver architecture and registration information from WHOIS records. They used two properties of the DNS architecture: whether a domain was self-resolving (i.e., the NS for example.com is hosted within *.example.com) and the freshness of the NS registration (in this case, whether the domain has been registered within the past year). Features including registration date were mined from WHOIS records.

To evaluate their strategy, the authors seeded a set of known "bad" domains from the joewein.net blacklist and tripled the number of known bad domain names via inference. Ap-

proximately 75% of the additional domain names eventually end up in various blacklists. Of the remaining unknown domains, visual inspection of many of the pages they host verify their content as spam or malicious, and some remain unknown, as they were not used at all over the course of this study.

What was the runtime of the clustering algorithm? It currently runs in about half an hour but has not been optimized for operational deployment. The clustering requires searching the zone file for domains whenever the blacklist is updated, and this would be very possible in practice. Yinglian Xie of Microsoft Research asked whether the domain names point at the same IP addresses and whether this might factor into clustering. Felegyhazi did not use the domain names' resolution in his clustering. Brent Hoon-Kang of UNC Charlotte asked whether NXDOMAIN results were part of the false positives and whether later domain testing might improve accuracy. Felegyhazi said that this might help, but the existing strategies identify domains sufficiently for blacklisting purposes. What were the characteristics of the unknown domains? They were mostly in the form "nounnoun.com" (as seen in many other spamming campaigns). Fang Yu of MSR asked about the breakdown of registrars for the bad domains. Felegyhazi remarked that this was an interesting point; a table exists in the paper showing a pronounced difference in distribution of registrars between all domain names and blacklisted domains.

- ***Detection of Spam Hosts and Spam Bots Using Network Flow Traffic Modeling***
  Willa K. Ehrlich, Anestis Karasaridis, Danielle Liu, and David Hoeflin, AT&T Labs

Anestis Karasaridis remotely presented this work on spambot detection. The authors' system mainly uses netflow data, along with some DNS data. The statistical properties of the spam at the network level are markedly different from legitimate email in terms of mean and variance of bytes per flow, which allows for network monitoring where the content of the message remains private. The authors identify both spamming hosts and spam command and control machines via their algorithm. While the netflow data is sufficient to classify spamming SMTP clients, the detection of controller machines is a two-stage process. Flow statistics such as fan-in and bytes-per-flow identify possible controllers, and then the second stage aggregates these records and ranks them by number of clients. DNS information is used to detect transient "Fast Flux" domains in order to improve controller detection. Direct connections within the netflow data without preceding DNS lookups also indicate compromised machine access. The authors found controllers for the Zeus, Ozdok, and Cutwail botnets via their approach.

Yinglian Xie asked whether attackers cognizant of these methods could introduce variance to hide from the algorithm, and Karasaridis noted that this was certainly possible, but they had not seen this happen in practice. Xie also asked what would make this spamming approach economically infeasible, because it is not very hard for a spammer to change hosts. The authors did not address economics in their research.

- ***Extending Black Domain Name List by Using Co-occurrence Relation between DNS Queries***
  Kazumichi Sato and Keisuke Ishibashi, NTT Information Sharing Platform Laboratories, NTT Corporation; Tsuyoshi Toyono, Internet Multifeed Co.; Nobuhisa Miyake, NTT Information Sharing Platform Laboratories, NTT Corporation

Kazumichi Sato presented his group's approach to extending blacklists with an aim very similar to that of Felegyhazi's work but with an orthogonal approach. The intuition underlying this work is that, as botnets must plan for resiliency against takedown, bots will often perform lookups for several command and control servers at the same time. The system assumes that if an individual host resolves two domain names at the same time frequently and one is bad, then the other is likely bad as well. A scoring system using a co-occurrence relation is used.

A difficulty with this approach is that popular, legitimate domain names are resolved by bots for either connectivity tests or simply by the legitimate user of a compromised machine. A larger population of non-infected hosts and their lookup statistics are used to weight the legitimate and popular domain names away from being labeled as bad. In addition, a host's overall lookup rate is used to weight queries so that individual heavy hitters do not bias the co-occurrence scoring. The dataset used includes one hour of DNS traffic from February of 2009. The authors created a blacklist using honeypots, including 270 domain names, and found 91% of bad domains within this trace were among the top 1% of all scores; among the top 100 scores, 39 were black, 4 were legitimate, and 56 unsure. The unsure domains included oddities such as lookups of the form <black domain name>.<legitimate domain name>, as well as DNS blacklist lookups.

Eric Ziegast asked whether this system had been deployed operationally since compiling the one hour of data used for the study? Sato answered no, stating that the timestamp mismatch between the blacklists and DNS data prevents deployment.

NEW THREATS AND CHALLENGES

Summarized by Rik Farrow (rik@usenix.org)

- ***Are Text-Only Data Formats Safe? Or, Use This LaTeX Class File to Pwn Your Computer***
  Stephen Checkoway and Hovav Shacham, University of California, San Diego; Eric Rescorla, RTFM, Inc.

Stephen Checkoway pointed out that most people consider text to be safe from infecting their computers. He presented a simple chart with two columns: unsafe and mostly trusted. Under "unsafe," he listed executables, Web apps,

media (Flash, JPG, etc.), and Office formats as examples. Under "mostly safe" he had ASCII text.

TeX and LaTeX have interpreters that create boxes with text or equations in them and then glue the boxes together, creating a laid-out page. By convention, the backslash is the escape character that tells the interpreter to do something special with the following text. You can also read and write files, although the *nix versions limit writing files to the current directory.

The authors wrote a virus, sploit.js, that finds .tex files and infects them. Interpreting infected files spreads the virus. They tested some online sites that offer TeX interpretation and found that they were potentially vulnerable to infection. And filtering out reads/writes is not trivial, as TeX includes macros that can obfuscate any virus, as well as a mechanism for changing the escape character, so potentially any string of text can be part of a virus.

Checkoway concluded by remarking that using binary vs text was not a good way to classify uploaded or downloaded files as safe. Niels Provos wondered how many academic sites were used to propagate sploit.js, and Checkoway assured us that they notified sites that provide LaTeX rendering services of the potential for remote code execution.

- ***DNS Prefetching and Its Privacy Implications: When Good Things Go Bad***
  Srinivas Krishnan and Fabian Monrose, University of North Carolina at Chapel Hill

Srinivas Krishnan pointed out that browser vendors poach market share by creating browsers that display pages more speedily. One of the tricks used involves prefetching DNS entries by looking up all the domain names found in links on the current page. The authors examined two ways of determining if a particular Web page had been viewed using two methods: probing open DNS servers and examining a DNS server dump.

They start by creating a profile of domain names that can be used to identify a particular Web page. Then they crawl a DNS server that allows queries from anywhere and serves the network of interest to infer cache hits using cache snooping. As the server replies, it will include TTL values for each domain, allowing the calculation of a confidence rating. This works because they already know the maximum TTL values by querying the domain itself, and if the domains were all fetched within a short time window on the crawled server, this indicates that they are related to a particular set of queries. They discovered that they could scan about every 30 minutes and get good results. Searching a DNS server dump provides a lot more accuracy.

Someone asked just how important this is. If you can dump someone's DNS server, you could likely just monitor the network and determine which URLs someone is fetching. Krishnan concurred, but pointed out that using logs works best and only works with targeted searches using profiles. Niels Provos wondered about the need for tokenizing, and Krishnan explained that tokenizing is only done when searching logs, not when trawling servers. Fabian Monrose pointed out that prefetching provides a lot more information. Roberto Perdisci asked if they made recursive requests, and Krishnan said they did not. Perdisci then wondered if this would indicate that something is awry. Krishnan responded that most open server operators don't check and don't care. Eric Ziegast asked if when doing online probes they can only infer that someone is doing a search, and Krishnan said this is correct. Only DNS server dumps reveal the source address of requestors. Monrose said that Chrome makes it easy to disable prefetching, and I discovered that it's not that hard to do in Firefox either.

- ***Honeybot, Your Man in the Middle for Automated Social Engineering***
  Tobias Lauinger, Veikko Pankakoski, Davide Balzarotti, and Engin Kirda, EURECOM, Sophia-Antipolis, France

Tobias Lauinger described a man-in-the-middle (MITM) attack that can improve upon IM spamming. IM spamming by bots is easy to detect, with 80% of users detecting a spambot within three messages (Huber et al.). The authors decided to take a different route by using a MITM approach to connect two real users and allowing them to chat until it was time to insert the spam.

Someone asked if they had asked their Institutional Review Board (IRB), and Lauinger said they don't have an IRB, so there is no one to say no. This provoked laughter from the audience. They did ask the University of Vienna and got approval from them.

They chose to use a dating channel and filtered out initial links. They also used a filter that translated male into female terms, and vice versa, and could keep people chatting up to five minutes. The MITM relay would start inserting URLs after ten exchanges. Using Tiny URLs worked best for this type of attack.

Vern Paxson asked how many sessions were shown in the graph (in the paper and as a slide), and Lauinger answered more than 1000 but not a million. Bill Simpson asked what name they chose, and Lauinger answered that they chose a name that could be male or female.

WORK-IN-PROGRESS REPORTS

The LEET '10 workshop ended with a short WiPs section.

Nick Weaver (ICSI) had a rant about making work harder for virus writers by suggesting that AV vendors should use polymorphic recovery tools, making them more difficult to disable or avoid.

Stefan Savage (UCSD) wants better ways of predicting the exploitability of things to help sysadmins choose which vulnerabilities are the most important ones to patch. The Common Vulnerability Scoring System (CVSS) is the industry standard today, and it uses a funky equation and several variables filled in by guesswork. So far, Savage's group has

used a metric that employs exploit data from OSDVB, measures the time from first announcement to first exploit, and has a more successful metric than CVSS.

Brent Hoon (Brent ByungHoon Kang, UNC at Charlotte) said that although Conficker had been contained (by buying up the domain names to be used in C&C) in 2009, Conficker has never disappeared. They are still seeing hits, and there appear to be about six million IP addresses that show signs of infection. Stefan Savage wondered how you could economically get patches to six million people. Someone else asked how Conficker was contained, and Hoon said that Microsoft had bought up 10,000 domain names, enough for three years. Hoon also said that Waledac, another contained botnet, has been decreasing over time. Savage commented that we have a low threshold for success.

Working with his advisor, Stefan Savage, Chris Kanich (UCSD) created a site that sat between those wanting CAPTCHAs solved and those willing to do so. They purchased and sold CAPTCHA solving, finding that CAPTCHAs cost .50 per 1000 solved, that it takes 8–12 seconds to solve each one, and where the solvers are. A paper related to this research will appear at USENIX Security '10. Niels Provos asked if there was an IRB review, and Savage answered that they can't identify any of the participants, so there was no review.

INM/WREN '10: 2010 Internet Network Management Workshop/Workshop on Research on Enterprise Networking

April 27, 2010
San Jose, CA

Introduction by Alva Couch (couch@eecs.tufts.edu)

INM/WREN '10 brought together academic and industry researchers to focus on the common goals of effectively managing the Internet and setting the future course of enterprise networking. This year's meeting involved many themes, including the rapidly evolving role of programmable networks and the OpenFlow interface to routing hardware, new monitoring and troubleshooting techniques, innovative uses of virtualization, and management challenges that arise from cloud computing.

OpenFlow is an open standard by which applications can interact directly with switches and routers via a portable interface. This exposes a usable set of switching controls at the application layer and works around the traditional roadblock of having to modify internal switch/router software in order to control switching. INM/WREN papers studied several uses of OpenFlow, including replacing traditional monitoring, engineering enterprise traffic, and distributing management.

Cloud computing was also a major theme. Papers not only studied how to manage clouds but also proposed several innovative techniques that show promise in enabling future cloud architectures. A multi-vendor panel exposed some of the more subtle challenges of cloud computing.

PROGRAMMABLE NETWORKS AND THEIR APPLICATIONS

Summarized by Alva Couch (couch@eecs.tufts.edu)

- ***Automated and Scalable QoS Control for Network Convergence***
  Wonho Kim, Princeton University; Puneet Sharma, Jeongkeun Lee, Sujata Banerjee, Jean Tourrilhes, Sungju Lee, and Praveen Yalagandula, HP Labs

Wonho Kim proposed a tiered system for QoS control using the OpenFlow architecture. Flows match flow specifications, which are grouped into slice specifications that have QoS objectives. Slices are mapped to hardware queues to determine priority. This mapping is performed independently for each switch. Multi-flow performance is managed by use of a Shortest-Span-First heuristic, in which the switch that is most performance-constrained is situated first in a required path. Small-scale experiments demonstrate substantive performance improvements.

- ***The Case for Fine-Grained Traffic Engineering in Data Centers***
  Theophilus Benson, Ashok Anand, and Aditya Akella, University of Wisconsin—Madison; Ming Zhang, Microsoft Research

Theo Benson proposed a new approach to datacenter traffic engineering called MicroTE. Datacenter traffic engineering involves balancing traffic within the center to avoid congestion and delays. But current techniques for traffic engineering, including equal cost, multi-path (ECMP) and spanning trees (STP), utilize single paths and ignore redundant paths when balancing. The MicroTE traffic engineering approach uses all paths, exploits short-term predictability for quick adaptation, and coordinates scheduling with a global view of traffic. The strategy was simulated using a trace of a cloud data center with about 1500 servers and about 80 switches. According to this simulation, MicroTE results in traffic patterns that are much closer to optimal than traditional approaches. An audience member asked how one defines optimality. It is estimated via linear programming with complete information. Another audience concern was whether the centralized approach advocated by MicroTE is scalable to large data centers.

- ***HyperFlow: A Distributed Control Plane for OpenFlow***
  Amin Tootoonchian and Yashar Ganjali, University of Toronto

One concern about OpenFlow is that its most common use case, involving centralized control, may not scale. Amin Tootoonchian presented HyperFlow, a distributed mechanism for managing OpenFlow switches, in which multiple switch groups are each managed by an independent manager. Managers do not know about one another and assume that they are alone in managing the network. Managers coordinate by distributing information on flows in adjacent switch groups, using the WheelFS file system to cache event streams. The authors conclude from performance limits for WheelFS that distributed consistency is sufficient for effective management if there are under 1000 updates/sec to the

WheelFS. The audience wondered how this approach can be combined with traffic engineering.

VIRTUALIZATION AND BEYOND

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

- ***The "Platform as a Service" Model for Networking***
  Eric Keller and Jennifer Rexford, Princeton University

Network virtualization is currently provided using the Infrastructure as a Service (IaaS) model, which requires customers to configure their virtual network equipment as if it were the physical hardware. In order to ease this configuration burden, Eric Keller described efforts to present virtual networks using the Platform as a Service (PaaS) model. In this model, customers would maintain the freedom to reconfigure their network without being confronted with the complexities of the physical infrastructure. Example applications include customer-controlled routing within a cloud platform (such as Amazon EC2) or customer-created multicast groups to avoid the need for overlay networks. Instead of configuration files, customers would provide executable scripts which dynamically adjust router functionality in response to network events. This effort is a work in progress; some of the challenges remaining are to identify the appropriate router abstraction for end users and to determine how to resolve conflicting updates from customers.

- ***vDC: Virtual Data Center Powered with AS Alliance for Enabling Cost-Effective Business Continuity and Coverage***
  Yuichiro Hei, KDDI R&D Laboratories, Inc.; Akihiro Nakao, The University of Tokyo; Tomohiko Ogishi and Toru Hasegawa, KDDI R&D Laboratories, Inc.; Shu Yamamoto, NICT

Yuichiro Hei presented a proposal for virtual data centers, defined as geographically distributed components presented as a single data center. Currently, providers must build multiple physical data centers in order to implement business continuity. However, fixed costs make this prohibitively expensive for small companies. By sharing physical data centers, small providers could maintain the scale and isolation of a single data center while achieving the reliability of multiple data centers. Cost-effective and reliable paths between the physical components of the virtual data center can be provided by an AS Alliance, a group of ASes which share information about unused routes and exchange traffic. In the event of link failure, an AS in the alliance can select an alternate route through the alliance in order to restore connectivity within the virtual data center.

- ***Europa: Efficient User Mode Packet Forwarding in Network Virtualization***
  Yong Liao, University of Massachusetts at Amherst; Dong Yin, Northwestern Polytech University, China; Lixin Gao, University of Massachusetts at Amherst

Lixin Gao presented a solution to the problem of achieving high performance in network virtualization platforms such as VINI. These platforms run in user mode and are slowed by the overhead of copying packets and handling system calls. Europa applies zero-copying schemes from the OS research community to platforms for virtual routing. The Europa kernel shares a packet buffer between the kernel and user space and avoids system calls by requiring that access to the buffer be asynchronous. An atomically updated state variable on each packet in the buffer serves as a mutex to mediate access. Experiments show that Europa-enabled virtual network platforms running in user mode are able to nearly match the performance of the Click virtual router when running in kernel mode.

MEASUREMENT AND MONITORING

Summarized by Alva Couch (couch@eecs.tufts.edu)

- ***A Preliminary Analysis of TCP Performance in an Enterprise Network***
  Boris Nechaev, Helsinki Institute for Information Technology; Mark Allman and Vern Paxson, International Computer Science Institute; Andrei Gurtov, Helsinki Institute for Information Technology

Surprisingly little study has been made of the actual performance of enterprise networks, perhaps because they are viewed as working "well enough." This study analyzes enterprise data from the Lawrence Berkeley National Lab from October 2005 to March 2006. Data was captured at switches for 351 hosts (4% of the network) and described 292 million intra-enterprise TCP packets. Bro 1.5.1 was used to associate packets with connections and analyze the nature of connections. The distribution of data within connections was exceedingly heavy-tailed: 90% of bytes appear in 160 connections out of about 363,000. Inter-subnet traffic was about 10 times intra-subnet traffic. Conversely, 57% of connections were short-lived and transmitted small amounts of data, while 37% of connections were long-lived and transmitted small amounts of data. The audience wondered whether there was any evidence of congestion or network problems, but in this data, there was no such evidence.

- ***Extensible and Scalable Network Monitoring Using OpenSAFE***
  Jeffrey R. Ballard, Ian Rae, and Aditya Akella, University of Wisconsin—Madison

Jeffrey Ballard presented OpenSAFE, a flow-monitoring layer based upon OpenFlow. OpenSAFE can selectively map flows to a designated port and direct exceptions to a software component. The configuration of OpenSAFE is specified via the ALARMS (A Language for Arbitrary Route Management for Security) language. Monitoring requests define input sources, data filters, and application processing. Line-speed monitoring is accomplished by tapping data from a port of interest to an otherwise unused dedicated monitoring port. OpenSAFE allows monitoring of waypoints, which are virtual points in a flow's path that may not correspond to hardware entities. This approach promises selective line-

speed monitoring, but there is a danger of data overrun for high traffic rates. The audience asked whether some of the software features could be implemented within OpenFlow itself rather than at the application layer. The authors responded that current OpenFlow implementations suffer from both implementation limitations and portability issues that might prevent implementation within OpenFlow.

- ***Beyond the Best: Real-Time Non-Invasive Collection of BGP Messages***
  Stefano Vissicchio, Luca Cittadini, Maurizio Pizzonia, Luca Vergantini, Valerio Mezzapesa, and Maria Luisa Papagni, Università degli Studi Roma Tre, Rome, Italy

In current BGP monitoring approaches, including BMP, data is gathered by polling BGP routers. Stefano Vissicchio proposed that data be collected by passive observation of traffic on BGP links. Links are tapped and the tap information is sent to a collector node. This is much less intrusive upon router performance than relying upon the router's management interface, and can be configured and extended without modifying BGP router software. Initial experimental results show promise for the technique.

NETWORK MANAGEMENT—EXPERIENCES

Summarized by Wonho Kim (wonhokim@princeton.edu)

- ***Experiences with Tracing Causality in Networked Services***
  Rodrigo Fonseca, Brown University; Michael J. Freedman, Princeton University; George Porter, University of California, San Diego

Rodrigo Fonseca presented their experience with integrating X-Trace into existing networked systems. As there is no standard way to associate software components in a networked system, it is difficult to find root causes and debug the systems via the existing device-centric approach. Rodrigo showed that the instrumentation can be easily done when X-Trace is integrated into 802.1X and that full "happen-before" events can be captured. When X-Trace is used in complicated large-scale distributed systems (e.g., CoralCDN), it gives previously unavailable information about existing performance problems such as long delays incurred for unknown reasons. X-Trace requires modifications to target systems, but it enables instrumentation with sampling and does not require support for time synchronization between multiple nodes. One interesting question was how kernels could support appropriate tracing. If a kernel can support recording system events (e.g., thread creation), this would be helpful for getting richer information about existing causality.

- ***Proactive Network Management of IPTV Networks***
  R.K. Sinha, K.K. Ramakrishnan, R. Doverspike, D. Xu, J. Pastor, A. Shaikh, S. Lee, and C. Chase, AT&T Labs—Research

Rakesh Sinha presented the architecture of the IPTV service network within AT&T. The IPTV service is inherently sensitive to delay and packet losses, and quality of service is directly visible to its 2 million-plus subscribers. Each channel has its own multicast tree, and to prevent service disruptions from link failures, Fast Reroute (FRR) is implemented using virtual links as backup links, so no OSPF convergence is required. However, multiple concurrent failures are common, and the failures can propagate, because FRR is not visible to IGP. To handle failures, one must provide network administrators with a more holistic view from multiple monitoring tools (e.g., NetDB, OFSPMon, MRTG). The authors developed Birdseye, a Web-based visualization tool, to provide a more comprehensive view of monitoring data from multiple sources and alert administrators to important events in the IPTV network.

PANEL

- ***What Do Clouds Mean for Network and Services Management?***
  Summarized by Eric Keller (eric.r.keller@gmail.com)

  Moderator: Dr. Anees Shaikh, IBM T.J. Watson Research Center
  Panelists: Adam Bechtel, Yahoo!; Tobias Ford, AT&T; Stephen Stuart, Google, Inc.; Chang Kim, Microsoft

Each of the panelists was given five minutes to summarize their position.

Chang Kim of Microsoft emphasized that there are several different management perspectives. Management may involve managing existing networks using the cloud, or building new networks and services on the cloud, or managing the cloud provider's network and services themselves. Each case presents great opportunity because there are many more affordable resources on demand. The downside, however, is that there is a lack of visibility, since cloud providers are less likely to expose individual components (failures, links, etc.), and lack of a standard management interface, especially because the interface is likely to be more service-centric than component-centric.

Tobias Ford from AT&T pointed out that the cloud is a convergence of several effects. Most importantly, the cloud should make the customers feel comfortable. SLAs explain availability to enterprises, while the provider must work closely with enterprises to explain the benefits, such as cost savings, of using the cloud. Ford believes the key to making the cloud work is automation. They are automating the creation of VPLS and VPN. The proper level of abstraction is crucial, and transparency makes customers more comfortable, so he advocates exposing the network elements to customers.

Stephen Stuart from Google considers the cloud to be a collection of different applications competing for resources. In this environment, things break all the time, but the network works (perhaps too hard) to hide broken parts from applications (e.g., TCP is good at hiding a broken network). Instead, things that break should be exposed to applications, which could behave differently if given better data—

they would have "actionable intelligence." To achieve this, we need programmable networks. Today, most networks are configured via a command line interface with layers of software that have to translate intent into vendor-specific configuration languages. Because of this, configuration is an act of faith. Instead, we need an RPC interface into the network element which allows application developers to "program the network." For example, when a rack switch fails, it is preferable for applications to react by throttling input to the rack or by moving the application off to another location, rather than continuing to bang on the network and let TCP handle it. This, of course, can be solved with OpenFlow.

Adam Bechtel of Yahoo! noted that things are much more complex than they used to be. Web pages are now dynamically prepared via a collection of services rather than statically served by a single host. There are no monopolies among the Internet/Cloud, so interaction among companies is more important. If external providers can do something better, customers will use them (e.g., CDNs). In networking, we have the appropriate abstraction for federation, which is BGP. It remains unclear what the proper abstraction level is for outsourcing services. Providing and consuming outsourcing comes with new problems. When you do something yourself, you have a lot of visibility into it, but if you outsource it, you lose that visibility. When you pay for something (e.g., outsourcing some service), you have an incentive to manage how applications use resources. How does the outsourcing provider manage the service? How does one kill the customer's misbehaving MapReduce? These remain open problems.

The floor was then opened for questions.

Alva Couch asked whether the application programmer will continue to have to understand the underlying architecture of the cloud, or whether it will be eventually abstracted away. Stephen Stuart responded that this depends upon one's performance requirements. Ford and Kim pointed out the need to interface with the networking team on the customer's side. Customer networks are not going away, so ideas on exposing APIs are important. Bechtel pointed out that for certain applications, one does need to know internals (e.g., Hadoop), while for others (e.g., search), internals can remain hidden. Stephen Stuart then reversed the question: do you expect legacy applications, which use particular features, to still be supported (e.g., Ethernet multi-cast)? Ford said that for new applications, control over things like multi-cast is essential, so we need to figure out how to expose them meaningfully.

Nick Feamster then asked what kind of interface (e.g., command line, scripts, database configuration sent to routers, RPC, programmable hardware) administrators and software should have in order to manage network elements. Stephen Stuart pointed out that OpenFlow allows him to pull the control plane out of switches. The application will not know what "my load will be X" means, but a controller can and will be ahead of the game rather than reactive. Nick Feamster later followed up, pointing out that you can do that with routers today—pushing configurations rather than an RPC interface. Stuart clarified that OpenFlow gives more control than using intermediate software layer interpretations. Bechtel mentioned that AT&T is aiming for using configuration stored in databases and sent to switches. In his experience, managing ACLs took the most amount of time, so they built a system for automating configurations; the next target is VIPs (virtual IP) and sending that to load balancers.

Kobus van der Merwe then asked whether one should think about end-to-end or localized performance in the data center. Stephen Stuart said that in his wackier thoughts, he does not need a network. The applications do not mind having a network, as long as the network does what the applications want. Ford believes in simplifying APIs so they apply across many domains.

Someone then asked how the panelists are dealing with the multi-level security requirement of the government. Ford pointed out that this is consistent with their view that an enterprise wants to know where its data is. By exposing APIs which controls those policies (e.g., where data flows), this would bring together the security concerns.

Sanjay Rao brought up the cloud provider lock-in problem and asked whether there would be an API that enables switching providers. Kim believes that providers will come up with their different APIs, much as network management APIs are not well standardized today. Bechtel believes the APIs will be a source of differentiation. Ford disagrees. He instead feels that the providers must work with partners, have some form of federation, and use open protocols and APIs.

Jeff Mogul questioned what the incentive is for providers to have provider lock-in. Aditya Akella suggested that there is a role for a cloud broker that leases from multiple providers and handles the differences. Stuart pointed out the success of Gmail's IMAP interface. Because customers can download their email and escape the cloud features when needed, they feel more comfortable with adopting them. Chang noted that IaaS is working because customers are free to bring whatever software they want and can move it if they want.

Amin Tootoonchian asked about challenges facing the deployment of OpenFlow. Stuart stated that before deployment is implementation and before implementation is design. Right now Google is working hard on the design of OpenFlow.

Kobus van der Merwe asked how one deals with operational complexity. Dave Maltz answered that the nice thing about the cloud is it forces you to do things that can scale.

IPTPS '10: 9th International Workshop on Peer-to-Peer Systems

April 27, 2010
San Jose, CA

ENTERPRISE P2P

Summarized by David Choffnes (drchoffnes@eecs.northwestern.edu)

- ***Blindfold: A System to "See No Evil" in Content Discovery***
  Ryan S. Peterson, Bernard Wong, and Emin Gün Sirer, Cornell University and United Networks, L.L.C.

We all rely on easy access to content in the Internet, but a major point of friction in making the content available is that content providers have a responsibility to enforce demands of copyright holders, even if the content was made available by a third party (e.g., users). By making the content easily accessible to users through searches, these providers also expose themselves to potentially costly copyright enforcement. This problem affects a wide range of popular services, from search engines like Google to BitTorrent aggregators and video services such as YouTube.

In this work, Bernard Wong proposed addressing the issue with Blindfold, a system for making providers blind to details of the content that is stored and served while still providing the necessary features of keyword searching that makes content so easily accessible. At a high level, the system divides content discovery into independent index and content services, then uses encryption to hide the search terms and plaintext content. The content discovery protocol further relies on CAPTCHAs to prevent automated recovery of either search terms or the content. In the end, the authors argue that their approach effectively provides plausible deniability for content providers.

The presentation was followed by a lively Q&A session. Predictably, someone asked about legal issues and exactly how far "plausible deniability" goes (e.g., in court). Wong reminded the audience that he is not a lawyer, but the protection could be sufficient for users. The next person wondered whether the system supports anything other than key/value pairs for searching. The answer was no, only key/value for now. The last question returned to legal issues, specifically how to protect index service providers. Wong suggested using cover traffic in the key/value store (i.e., hide real search terms in a sea of fake terms)—for example, using a dictionary to supply them. He also suggested a fully distributed index service, such as the Vuze DHT.

- ***AmazingStore: Available, Low-cost Online Storage Service Using Cloudlets***
  Zhi Yang, Peking University, Beijing; Ben Y. Zhao, University of California, Santa Barbara; Yuanjian Xing, Song Ding, Feng Xiao, and Yafei Dai, Peking University, Beijing

Online storage services are becoming incredibly popular (e.g., through S3 and Mozy), but their reliance on data centers introduces a small number of points of failure that can significantly reduce content availability. Distributed approaches to storage services such as P2P can eliminate this risk of small numbers of points of failure, but the high churn rate of average users in such systems generally reduces availability. Zhi Yang proposed combining these two complementary models to reap the best of both worlds through a system modestly named AmazingStore.

The system is designed using a DHT to locate and store content, with data centers as primary storage and peers as backups. Zhi Yang explained a number of trade-offs in this system, including the level of replication (to ensure a target availability) and the various costs of serving content—free from peers but potentially expensive from data centers. He then discussed how master servers in AmazingStore monitor peer liveness with heartbeat messages to estimate the probability of permanent outages and to replicate data to ensure its availability. Finally, Zhi Yang showed results from a live deployment of their service in China containing 12,000 users and 52,000 objects. The key take-aways were that content availability increased with peer assistance and peers alone cannot provide sufficient availability, so data centers are an integral part of this system.

One important point of confusion about the presentation was whether the system is a peer-assisted data center or a datacenter-assisted P2P storage service. Zhi Yang replied that the system is designed both to improve availability and to offload datacenter costs onto peers. Do users expect the system to provide primarily reliable storage or highly available storage? About 10% of users are storing backups and the remaining portion is for file sharing.

P2P SEARCH

Summarized by Michael Chow (mcc59@cornell.edu)

- ***Estimating Peer Similarity using Distance of Shared Files***
  Yuval Shavitt, Ela Weinsberg, and Udi Weinsberg, Tel-Aviv University, Israel

Udi Weinsberg listed three reasons for not being able to find useful content in estimating peer similarity: not enough metadata, the searches take place in extreme dimensions, and sparseness. The goal, then, is to come up with a new metric for peer similarity.

Their work was done on an active crawl of Gnutella. It looked at a sample of 530,000 distinct songs and 100,000 peers. From the data, they found that 98% of peers share less than 50 songs. They then created a file similarity graph where the files were vertices and the link weights were the number of peers sharing both. The link weights were normalized with popularity and only the top 40% were kept. The filtering process was for filtering out weird tastes. A bipartite graph between every two peers was then created by doing a shortest path walk on the file similarity graph. A maximal weighted matching process was then run on the

bipartite graph. It finds the best matching link between files and then normalizes between peers. This serves to eliminate spareness.

Weinsberg then pointed out some issues with this process. He discussed the possibility of the similarity graph not being connected, but this is not really an issue, since this shows dissimilar tastes. Next he talked about the costliness of finding the shortest path on the file similarity graph. In order to reduce the cost of running the shortest path walk, they looked at only the top *N* nearest neighboring files, and they limited the search depth to *K* times the distance of the first finding. They found that 1.5 times the distance of the first finding was sufficient. Weinsberg next briefly discussed the results of their work. They found that there was a correlation between the similarity of artists but no direct correlation between the similarity of geography (physical closeness).

Emin Gün Sirer, Cornell University, pointed out that the voting patterns on Gnutella could be different from sharing patterns and asked how they decoupled this. Weinsberg said that they filtered implicit voting, e.g., user downloaded file, and immediately deleted it, and they also filtered out strange behavior. John R. Douceur, Microsoft Research, asked whether there was a way to make this distributed, because constructing the file similarity graph requires complete knowledge. Weinsberg said it's a problem they are currently working on and would require some sort of approximation of the file similarity graph.

- ***Don't Love Thy Nearest Neighbor***
  Cristian Lumezanu, Georgia Institute of Technology; Dave Levin, Bo Han, Neil Spring, and Bobby Bhattacharjee, University of Maryland

Christian Lumezanu talked about the need for applications to select nodes based on latency constraints. He outlined a scenario involving a multiplayer game where a bunch of players want to find a server that minimizes average latency to players. He described a naive approach which involves polling the latency of servers and exchanging values with each player. The problem with this approach is that not all players may know all of the game servers. He proposed a set of network coordinates that each node has. Latency is measured by the distance between network coordinates. Each node can calculate the theoretical optimum based on the cost function and find the node that minimizes the cost function. The nearest neighbor may not necessarily be enough for complex cost functions.

Lumezanu then introduced Sherpa, an overlay system that finds the lowest cost overlay. Sherpa makes use of Voronoi regions or a set of points in space closer to that node than any other node in that space. Sherpa makes use of compass routing and gradient descent to find the lowest cost. Compass routing is a greedy geometric algorithm that finds the nearest neighbor to the optimum. It selects the node with the lowest geometric angle and stops when the optimal point is in the Voronoi region. Gradient descent is then performed, exploring adjacent Voronoi regions with lower costs. It then checks the nodes in that region. Since there are slight inaccuracies, latency probes are performed for inapproximate mappings to the network space. In evaluating Sherpa, they found that 80% of the time Sherpa selected a cheaper node than the one found by the nearest neighbor. In comparison to an all-knowing oracle, in 65% of the queries, the node chosen by Sherpa is in the lowest 10% of nodes.

Emin Gün Sirer, Cornell, asked if they had looked into ways of minimizing inaccuracies from embedding to the network coordinates. Lumezanu said that these inaccuracies are reduced by using latency probes. John R. Douceur, Microsoft, asked why the nearest neighbor is better than Sherpa 20% of the time. This was mainly due to embedding error.

- ***SplitQuest: Controlled and Exhaustive Search in Peer-to-Peer Networks***
  Pericles Lopes and Ronaldo A. Ferreira, Federal University of Mato Grosso do Sul, Brazil

Ronaldo A. Ferreira addressed the problem of complex queries in P2P networks, which are still an open problem. Existing solutions are based on random walks on unstructured networks, hence incur high traffic on the network due to replication. Moreover, they cannot guarantee that content is found, even if it exists in the network, due to the random nature of the P2P network and lack of complete network coverage. The goal of this work is to have complete coverage of the network when issuing queries by grouping peers and connecting the groups. A query is efficiently propagated among groups until all groups are covered or the answer is found, hence complete coverage is guaranteed.

More specifically, Ferreira suggested grouping peers in such a way that each peer has complete knowledge of the content that exists in each of the other peers in the group (achieved using replications within the group). The groups are placed on a virtual ring, such that each peer in a given group has links to the previous and the next groups in the ring and also to other random groups on the network. When a query arrives in a given peer, the peer checks whether it can be answered with content from its group. Otherwise, the peer sends the query to the connected groups and notifies them which other groups they need to propagate the query to (in case they cannot resolve it), by partitioning the space into non-overlapping intervals.

The size of each group is selected so that it minimizes the index replication and search hops in order to cover the complete set of groups, and was shown to be proportional to the square root of the number of peers. Groups are visited only once, basically performing a broadcast in a randomly constructed tree, determined by the order of groups within the ring. The depth of the tree has a theoretical limit and was shown empirically. Preliminary results show improvement over the previous work BubbleStorm [SigComm 2007].

Overall SplitQuest appears promising—fast, complex queries and less traffic on the network.

Someone pointed out that prior work relies on random search, hence the success rate is smaller than 100%. Why is it that the proposed technique also has less than a perfect success ratio? This is due to peer churn. The results for static scenarios yield a 100% success rate. Someone else said that large networks can result in huge groups. This is a problem both in peer churn, since the groups can change significantly, and in replicating content. Ferreira answered that you don't really share the file but, rather, some metadata about where to find the file in the peers within the same group. You can create smaller groups, but you will have more groups and consequently more messages will be propagated to resolve a query. However, huge groups are indeed a problem and will be considered in future work.

GOSSIPING SYSTEMS

- ***Balancing Gossip Exchanges in Networks with Firewalls***
  João Leitão, INESC-ID/IST; Robbert van Renesse, Cornell University; Luís Rodrigues, INESC-ID/IST
- ***A Middleware for Gossip Protocols***
  Michael Chow and Robbert van Renesse, Cornell University
- ***StAN: Exploiting Shared Interests without Disclosing Them in Gossip-based Publish/Subscribe***
  Miguel Matos, Ana Nunes, Rui Oliveira, and José Pereira, Universidade do Minho, Portugal

No reports are available for this session.

BITTORRENT

- ***Public and Private BitTorrent Communities: A Measurement Study***
  M. Meulpolder, L. D'Acunto, M. Capotă, M. Wojciechowski, J.A. Pouwelse, D.H.J. Epema, and H.J. Sips, Delft University of Technology, The Netherlands

M. Capotă introduced a study on the differences between two fundamental BitTorrent community designs: public (open to anyone) and private (requiring user accounts). Capotă focused on several measurements regarding performance and operational aspects of these communities, namely, download speed, connectability, seeder/leecher ratio, and seeding duration. Not surprisingly, private communities perform better.

The study focused on two public and four private communities, in which the private communities had different sharing enforcement policies. The study was performed using a simplified BitTorrent client that monitored real torrents provided by these different communities. The study showed that private communities have 3 to 5 times faster download speeds; 50% better connectivity (direct reachability of peers in the swarm) on average; at least 10 times higher seeder/leecher ratio at the torrent level; and a much longer average seeding duration. The presenter concluded that tit-for-tat mechanisms to promote collaboration are virtually irrelevant, as private communities appear to be highly successful without requiring such mechanisms.

Was any selfish behavior identified in the study? The authors did not try to identify such behavior, as it was not the main goal of the study. What caused the better connectivity in private communities? This was probably related to more skill on average among the private communities' users. Another question concerned the evolution of swarms for highly seeded torrents. Capotă clarified that this was a subject for future work. Did torrent content affect the study? There was no bias in the study; the studied communities were focused on the same type of content, and thus the comparison is valid.

- ***Comparing BitTorrent Clients in the Wild: The Case of Download Speed***
  Marios Iliofotou, University of California, Riverside; Georgos Siganos, Xiaoyuan Yang, and Pablo Rodriguez, Telefonica Research, Barcelona

Marios Iliofotou reported results on a study to determine if there are more differences between BitTorrent clients than meet the eye. The authors conducted a large-scale study focused on the two most popular BitTorrent clients, uTorrent and Vuze. The study covered 10,000,000 users and 6000 different ISPs during one month last summer.

On average, uTorrent is able to achieve 16% higher download speeds than Vuze, in a consistent way both across time and ISPs. In order to extract some additional clues on the implementation details that could explain this difference, the authors resorted to a controlled setting in order to minimize the impact of hidden variables. The authors were able to identify four main implementation differences, of which two were presented: neighborhood management and upload bandwidth distribution. Concerning the first, Vuze has more ephemeral connection (lasting less than 5 minutes); concerning upload management, uTorrent simultaneously uploads to more peers. Iliofotou concluded with the observation that some design choices have a significant effect on performance and should be carefully evaluated.

Were the authors able to identify traffic shaping in some particular ISPs? This was not the goal of the study, so they did not check for this. Had some clients used more aggressive seeding strategies? Again, this was not an aspect studied in this work. Was peer connectivity taken into account, since different clients could use different tricks to go around firewalls and NAT boxes? They did not check this since their main concern was to avoid other bias sources in the study.

- ***Power-law Revisited: A Large Scale Measurement Study of P2P Content Popularity***
  György Dán, KTH, Royal Institute of Technology, Stockholm; Niklas Carlsson, University of Calgary, Canada

György Dán introduced a study whose goal was to characterize the distribution of content-popularity metrics, in particular instantaneous and the download popularity in BitTorrent systems. To this end the authors conducted a practical study in which they used a Mininova.org screen scrape and a scrape of 721 of BitTorrent trackers.

The study showed that it is hard to fit both popularities in power-law distributions, as previous studies had indicated. Instead, Dán argued in favor of distributions with the following characteristics. For instantaneous popularity, one could perhaps consider a power-law distribution head, a distinct power-law trunk, and possibly (although not obviously) a third power-law distribution tail. For the download popularity, Dán argued in favor of a distribution composed of a flat head, a power-law trunk, and concluded that the tail may exhibit a power law distribution for short periods of time, but the power-law would certainly not hold for long periods.

There were several reactions to the paper. Initially, a participant commented that having more accurate distributions was good, but it was also important to understand their implications in peer-to-peer systems. Dán commented that indeed these distributions were really complex, as they are influenced by several aspects. What changes should be made to peer-to-peer systems given the new, more accurate distributions? This is still an open question, but results show that popularity does not follow power-law distributions, and so systems should clearly not be designed with this in mind. What were the reasons behind the absence of a long tail? This is probably related to content aging, but there were probably other factors. Finally, another participant was curious about the behavior of individual swarm participants. Dán clarified that participants' IDs were not logged in their study, so there was no way for them to know.

- ***Strange Bedfellows: Community Identification in BitTorrent***
  David Choffnes, Jordi Duch, Dean Malmgren, Roger Guiermà, Fabián Bustamante, and Luís A. Nunes Amaral, Northwestern University

Privacy in BitTorrent communities has become an increasing concern. David Choffnes demonstrated this, showing how the strong global connection structure in this P2P system allows an external observer to infer the existence of communities with common interests. This enables a guilt by association attack, where a third party can derive the content that is shared by users of an entire community, using information only from a single member of this community.

In order to evaluate this, Choffnes described a study where the authors took one month of data related to an average of 3000 users per day that established more than 10,000 connections among themselves per day. Using these traces they created a graph where links had weights that were proportional to the number of times they were registered in the trace. Using heuristic-based techniques, the authors were able to identify nine distinct communities with variable sizes. This information allowed the author to infer that information from 1% of nodes had the potential to reveal 80% of the network for direct and one level of indirect observation, and that one host has a potential to reveal 80% of the network for double indirect observation.

Did the authors have any idea concerning the nature of the identified communities? No, but they were looking for explanations for this, such as common geography and content types. To further clarify, they were unable to identify the nature of these communities because the study intentionally did not record the nature of the data being downloaded by users. How could the authors derive the density of the relations between nodes in the same community? This was derived empirically from the data itself. Had the authors tried other techniques, such as clustering, to identify communities? They used simulated annealing to maximize modularity, which is commonly used to identify communities. Did the results show that there was no sharing between members of different communities? Indeed, such sharing activities existed but were not common compared with sharing within communities.

BSDCan 2010: The Technical BSD Conference

May 13–14, 2010
Ottawa, Canada

- ***Security Implications of the Internet Protocol version 6 (IPv6)***
  Fernando Gont, Universidad Tecnológica Nacional/Facultad Regional Haedo (Argentina) and United Kingdom's Centre for the Protection of National Infrastructure

  Summarized by Alan Morewood (morewood@computer.org)

  Editor's Note: As a Gold Sponsor of BSDCan 2010, USENIX invites attendees to submit reports for publication in ;login:. *We received this very timely summary about issues with implementing IPv6 in production networks.*

Although there where frequent references to the sysctl parameters that allow BSD to tweak various kernel settings, Fernando's talk was focused at a higher level, explaining the fundamental concerns uncovered during his ongoing research with the UK's Centre for the Protection of National Infrastructure (CPNI).

The three most important messages from the presentation may be: to train design and operations staff on IPv6 before deployment; that there are significant similarities and differences between IPv4 and IPv6 but that myths and marketing are unreliable sources to distinguish the differences; and, finally, for developers to always provide a limit to functionality which uses kernel resources.

To have features similar to an organization's existing IPv4 firewall, one needs to have similar policy enforcement mechanisms available, so careful evaluation of vendor's IPv6 equipment is necessary. Some settings, such as ICMPv6 redirect, are not entirely necessary for operation according to Gont, only for optimization. A strong knowledge of architecture will mean that redirects are not necessary for operation or optimization. Education and training would allow for the right configuration; while knowledge of IPv4 and related utilities is useful, there are some differences in the details.

Know the details, not the hype, suggested Gont. Early in the presentation, Gont made reference to a myth that, due to the large IPv6 address space, scanning for valid addresses will be infeasible. This presumption is predicated on the idea that the space is used in a random and uniformly sparse manner, but studies have shown that deployment methods actually used have many factors which lead to significant predictability. These include sequential manual address assignment, sequential MAC address assignments typical of an enterprise environment, and addresses based on IPv4, all found within the Host ID field. It was noted that with the default MAC-based HostID assignment, individual hosts can be tracked when connected to different networks, causing possible privacy concerns.

Gont pointed out that while OpenBSD had many of the tweaks necessary to safeguard resource usage and limit other IPv6 vulnerabilities, there were still areas of concern not covered and that the default settings were not the most conservative, allowing for full functionality. Some BSD implementations did not enforce minimum fragment or packet sizes, none filtered MAC addresses reserved for broadcast and multicast, and some did not have tweaks to disallow autoconfiguration from manipulating routing configuration, all facilitating various abuses.

An example of resource utilization issues is that a device can receive multiple network addresses, at least one per valid network prefix in operation on a link. Without a limit, an attacker may impose thousands of addresses on a host's network interface via the autoconfiguration features, causing excessive resource use on the host. Similar resource issues were brought forward for fragment processing, link layer address cache sizes, and many other important functions. An audience member noted that putting limits on these parameters could prevent correct operation while under attack, which was acknowledged, but all agreed that having an operational kernel facilitated corrective action.

Many more concerns were presented, such as the use of fragmentation which, despite the new restrictions on overlapping fragments (RFC5722), may still allow firewall rule bypassing. More details on this and other issues are still not public, for security reasons. They are working with vendors and other relevant parties to correct protocol and implementation issues.

writing for *;login:*

Writing is not easy for most of us. The way to get your articles published in *;login:*, with the least effort on your part and on the part of the staff of *;login:*, is to submit a proposal to login@usenix.org.

PROPOSALS

;login: proposals are not like paper submission abstracts. We are not asking you to write a draft of the article as the proposal, but instead to describe the article you wish to write. Some elements are essential in any proposal:

- The topic of the article
- The type of article (e.g., case study, tutorial, editorial, mini-paper)
- The intended audience (e.g., sysadmins, programmers, security wonks, network admins)
- Why this article is useful
- List of any non-text elements (e.g., illustrations, code, diagrams)
- Approximate length of the article

We suggest that you try to keep your article between two and five pages, as this matches the attention span of many people.

The answer to the question about why the article needs to be read is the place for your most eloquent explanation of why this article would be important to the members of USENIX.

UNACCEPTABLE ARTICLES

;login: will not publish certain articles. These include but are not limited to:

- Previously published articles. A piece that has appeared on your own Web server but not been posted to USENET or slashdot is not considered to have been published.
- Marketing pieces of any type. We don't accept articles about products. "Marketing" does not include being enthusiastic about a new tool or software that you can download for free, and you are encouraged to write case studies of hardware or software you helped install and configure, as long as you are not affiliated with or paid by the company you are writing about.
- Personal attacks

FORMAT

Please send us plain-text proposals: simple email is fine. Send proposals to login@usenix.org.

DEADLINES

For our publishing deadlines, including the time you can expect to be asked to read proofs of your article, see the online schedule at http://www.usenix.org/publications/login/sched.html.

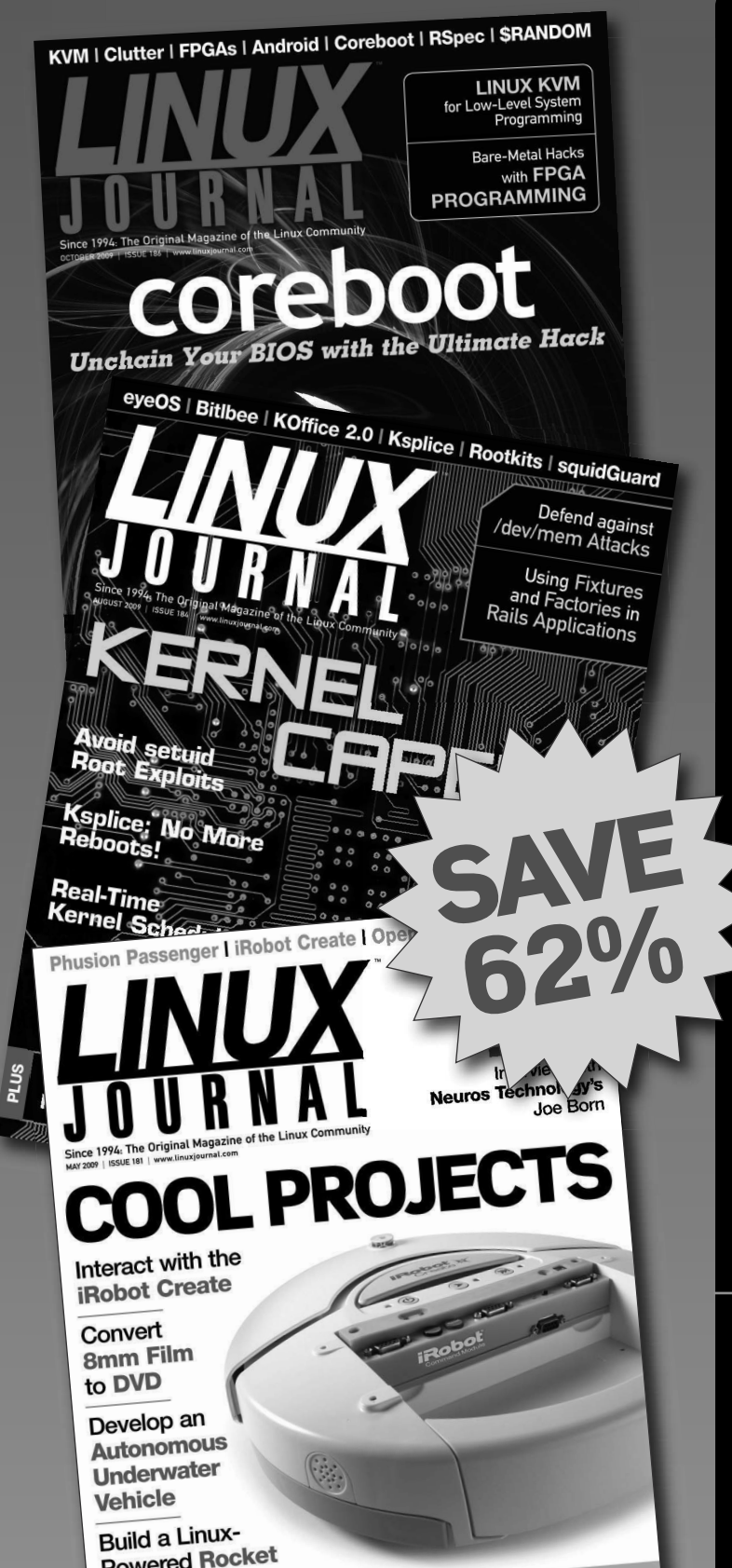
COPYRIGHT

You own the copyright to your work and grant USENIX permission to publish it in *;login:* and on the Web. USENIX owns the copyright on the collection that is each issue of *;login:*. You have control over who may reprint your text; financial negotiations are a private matter between you and any reprinter.

FOCUS ISSUES

Each issue may have one or more suggested focuses, tied either to events that will happen soon after *;login:* has been delivered or events that are summarized in that edition. See http://www.usenix.org/publications/login/sched.html for the proposed topics for upcoming issues.

If You Use Linux, You Should Be Reading *LINUX JOURNAL*™



SAVE 62%

- In-depth information providing a full 360-degree look at featured topics relating to Linux
- Tools, tips and tricks you will use today as well as relevant information for the future
- Advice and inspiration for getting the most out of your Linux system
- Instructional how-tos will save you time and money

Get *Linux Journal* delivered to your door monthly for 1 year for only $29.50! Plus, you will receive a free gift with your subscription.

SUBSCRIBE NOW AT:

WWW.LINUXJOURNAL.COM/SUBSCRIBE

Offer valid in US only. Newsstand price per issue is $5.99 USD; Canada/Mexico annual price is $39.50 USD; International annual price is $69.50. Free gift valued at $5.99. Prepaid in US funds. First issue will arrive in 4-6 weeks. Sign up for, renew, or manage your subscription on-line, www.linuxjournal.com/subscribe.

NOV. 7-12
2010

24TH LARGE INSTALLATION
SYSTEM ADMINISTRATION CONFERENCE

SAN JOSE
CALIFORNIA

LISA'10

UNCOVERING THE SECRETS OF SYSTEM ADMINISTRATION

PROGRAM INCLUDES:
UNRAVELING THE MYSTERIES OF TWITTER INFRASTRUCTURE, LEGAL ISSUES IN THE CLOUD, AND HUGE NFS AT DREAMWORKS

STAPLE HERE

INSERT

SPONSORED BY
USENIX in cooperation with LOPSA & SNIA

KEYNOTE ADDRESS BY
Tony Cass, *CERN*

**JOIN US FOR 6 DAYS OF PRACTICAL TRAINING ON TOPICS INCLUDING:

6-day Virtualization Track by instructors including John Arrasjid and Richard McDougall
Advanced Time Management: Team Efficiency by Tom Limoncelli
Dovecot and Postfix by Patrick Ben Koetter and Ralf Hildebrandt
5-day Linux Security and Administration Track

**PLUS A 3-DAY TECHNICAL PROGRAM

Invited Talks
Refereed Papers
Workshops
Vendor Exhibition

REGISTRATION OPENS IN LATE AUGUST
www.usenix.org/lisa10/lg

USENIX
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