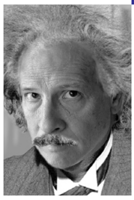
RIK FARROW

musings



rik@usenix.org

I FEEL LIKE I'VE FINALLY ENTERED THE future. I now have a microwave dish on my rooftop, watch television using a computer, listen to radio from online streams, and am even installing my own solar power plant. But when it comes to security, we are all still stuck in the Dark Ages.

In this issue you can read about a Web browser designed with security in mind. Don't think for a minute that current Web browsers include security in their design, because security is, at best, something added on as "desirable." Sure, Firefox, IE, Safari, and Opera are all carefully vetted for bugs and poor programming practices. And all also have security "features." But these browsers all share one thing, something that all browsers have done since Mosaic: They download and execute remote code in your own user context.

Just think about it for a moment. Would you routinely go off to strange Web sites, download code, and run it? Well, of course you would, because five nines (99.999%) of the people I know do exactly that. The 0.001% don't use Web browsers. And I am not kidding. I know one security geek who is still using nc for "browsing" the Web. That guy sure has to work hard to read Web pages (especially these days!), but he is secure, as the text he reads has been eviscerated of any danger because it is treated just as text.

Drive-by Downloads

I wrote about drive-by downloads exactly one year ago. Niels Provos and his team at Google [1] are still trolling caches of Web pages looking for pages that cause the downloading of first-stage exploits. And they are still finding them—thousands every day. The vast majority of these Web pages weren't made that way by their owners, but their Web server was exploited, and small changes were made not just to stored pages but also (in some cases) to dynamic content as well. All it takes is the insertion of an `iframe` or `script` tag to take control of almost any Web browser that visits the page.

Exploiting personal computers is big business and is completely untaxed and unregulated (a libertarian's dream, in that sense). Exploited systems are used in relaying spam, exploiting other systems, and to collect identity information. Oh, I forgot to mention DDoS, but when you control thousands of distributed systems all with Internet connections, that should be obvious. All of these activities are illegal (apparently with the exception of the U.S. record industry [2]), yet occur routinely.

The OP browser's design (page 14) does what all browsers should—isolate one site's content from another site's. With the OP browser, you don't get multiple sites' content all running within the same security context. Each site runs within its own page-rendering process. And actual display of contents is done by yet another process, also running within a secure environment. The current browser design runs everything within your standard security context, and that means that anything you can do, your browser can do too. Sure, browser designers do make honest attempts to limit what your browser can do to your system, but these limitations have failed over and over again. And they must continue to fail, because the basic browser design is so terribly flawed.

We run our desktop systems as if they were mainframes with multiple users. The operating systems were designed for multi-user systems (unless you are still running Windows 95 or Mac OS 9), yet you (and your exploiter du jour) are the only ones using your computer. Our processors are also designed like old time-sharing systems. Neither has kept up with the way computers are actually used today. The biggest reason for this is inertia, in that OS designers build familiar systems, and CPU designers take advantage of their decades of building the same architecture.

I don't want to suggest that starting over will be easy. We need a way of transitioning from our insecure desktops, laptops, and smart phones to systems designed with security, performance, and efficiency from the ground up. The OP browser provides a model for the transition. But without all the sexy features and performance we have grown so accustomed to, the OP browser will not replace Firefox or IE.

Our current situation reminds me of the bad guys in the movie *Who Killed the Electric Car*? [3]. In that movie, General Motors (GM) recovers the hundreds of electric vehicles that people had leased and has them crushed—even when the owners offer to pay $24,400 for each several-year-old car rather than allow them to be taken away and destroyed.

The villains in that movie were all part of the status quo: manufacturers who owned factories that build internal combustion engine vehicles, oil companies with vast refineries and distribution systems, and even vehicle maintenance suppliers (since the GM EV1 had three items that needed to be replaced: tires, brake pads, and windshield washer fluid). So instead of another ten years' experience with electric vehicles, we have $4/gallon gas (perhaps it will be $5 when you read this) and a glut of SUVs that have lost their appeal (and most of their resale value). Note that consumers were also found guilty in the film (as they were not willing to change).

Have we reached the $4/gallon point in terms of desktop insecurity? I really don't know, but I do make Linux Live CDs for my friends who need to manage some of their finances online. Today, I wouldn't type anything on a Windows system that I wanted to keep private, and I almost feel the same about Macs and even Linux systems. They all share similar design flaws, in that one user runs remote code via the Web browser and mailtool as well. Linux, BSD, Mac OS X, and other UNIX-like systems at least separate the user from the administrator, making kernel-level exploits more difficult. But the installation of code within the browser will capture all of your keystrokes, and tools have existed for some time that selectively filter out those keystrokes that look like credit card numbers or what you have typed during a visit to a large number of domains, all related to finance. These captured keystrokes then get posted to a Web server under the control of the dark industry.

The Lineup

I really hate being so negative, but I do prefer to be brutally honest, even at the risk of sounding like a broken record. The Grier article about the OP browser gave me the perfect opportunity to sound off about this issue again. But all is not dark and dreary.

In this issue, we lead off with an article from Brandon Enright and some researchers from UCSD. While building an internal tool to track Storm botnet infections within the campus network, the team developed Stormdrain, an efficient means for discovering Storm bots wherever they are. You can read their LEET '08 paper [4], but I asked them to write about something a bit different: their experiences when they discovered that there were whole families of Storm-bot pretenders to uncover as well. Some of these look-alikes were created by other Storm researchers, and others represent attacks against Storm by other botnet owners. During the LEET workshop presentations related to Storm, it was common to hear comments like "That was you?" or "I owe you a beer" from researchers who had collided with them in the Storm network.

Colin Dixon and his co-authors wrote an article about their proposed DDoS defense for sites that generate dynamic content. DDoS is still a real issue eight years after these attacks first made headlines, and ways of providing access to servers with dynamic content while under attack—and that don't involve massive changes to the Internet's infrastructure—are rare. I liked what I heard during their paper presentation during NSDI '08 and asked them to write this article describing their very interesting approach.

Diana Smetters and her co-researchers had presented a paper at the UPSEC workshop that dealt with issues I felt we all face routinely. In the course of our work, we often need to share information electronically, yet doing so without running foul of security policies is just about impossible. Smetters and her co-authors interviewed people who needed to share data as part of their jobs, and they report just how these people actually handled sharing. I think there are lessons in this article for all of us, whether we write the security policies or violate them with file-sharing practices.

Rohan Murty and his co-authors present a different approach to providing better Wi-Fi bandwidth. In this article, related to their NSDI paper, Murty explains how they went about using existing infrastructure (plus some Wi-Fi cards) to control how wireless clients associate with access points, without modifying the clients.

Alva Couch thoughtfully provides us with his own viewpoint about why we need real standardization in sysadmin. Couch cogently explains not just why he thinks we need standards but what these might look like, and he provides real-world examples.

David Blank-Edelman presents us with cool Perl debugging techniques. Peter Galvin offers his favorite list of procedures for analyzing system problems, as well as opening this topic up for discussion in a wiki. Dave Josephsen demonstrates relatively unknown tricks that you can do with RRDtool, as long as you can grok RPN (Reverse Polish Notation). Finally, Robert Ferrell revels in cajoling us into considering using memes as a method of sharing encryption keys.

We have five summaries, including two from the recent BSDCan '08 conference in Ottawa. We also have the NSDI and LEET summaries. Finally, we have the short version of the summaries of WOWCS, the Workshop on Organizing Workshops, Conferences, and Symposia for Computer Systems, a meta-workshop. Given the importance to researchers of getting their work published, the WOWCS workshop summaries should be required reading. WOWCS discussion covered many of the issues involving Program Committees, so if you plan to participate in a PC or submit a paper to one, I suggest you read this summary (or the longer version found at [5]).

As our personal computing devices get more powerful, and even as our cell phones may soon take over for our laptops, we need real security. What's the point of having computers if they can't be trusted? Our security model, our OS model, and even our CPU/system architecture were designed for a long-gone time. It is time for us to prepare for a secure future.

REFERENCES

[1] Ghost turns Zombie: Exploring the Life Cycle of Web-based Malware: http://www.usenix.org/events/leet08/tech/full_papers/polychronakis/polychronakis_html/.

[2] MediaDefender shuts down legitimate BitTorrent tracker with DDoS attack: http://blog.wired.com/27bstroke6/2008/05/mediadefender-d.html?cid=117123750.

[3] http://en.wikipedia.org/wiki/Who_Killed_the_Electric_Car?.

[4] The Heisenbot Uncertainty Problem: Challenges in Separating Bots from Chaff: http://www.usenix.org/events/leet08/tech/full_papers/kanich/kanich_html/.

[5] WOWCS scribe notes: http://www.usenix.org/events/wowcs08/tech/WOWCSnotes.pdf.