An Interview with Ben Laurie

RIK FARROW

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

Ben Laurie is a software engineer in Google's security team and a visiting fellow at the Cambridge University Computer Lab. He wrote Apache-SSL, which powers more than half the world's secure Web sites, and he maintains OpenSSL, the world's most widely used open source cryptographic library. Currently he is trying to fix the Internet with Certificate Transparency (http://www.certificate-transparency.org/), among other things. ben@links.org

I had heard of Ben Laurie before, as he is the co-author of several papers I've read, plus open source software that millions of people use: Apache-SSL. And while I was at USENIX Security, Ben's name came up while I was talking about Bitcoin.

I had asked Ben to write in the past, but things never quite worked out, so it seemed natural that I try an interview instead, which did allow me to ask Ben about some things that I was curious about.

Rik: I read your Wikipedia page and learned that you've been working for Google for years. But you are also a Visiting Fellow at Cambridge University's Computer Laboratory, founding director for Apache, a core team member of OpenSSL, security director for The Bunker Secure Hosting, a committer for FreeBSD, trustee and founding member of FreeBMD, and an Advisory Board member of WikiLeaks.org. This seems like an awful lot to be doing while working for Google.

Ben: Indeed I do work for Google, and have done so now for seven years. My current main project is Certificate Transparency [1]. As you say, I am a Visiting Fellow at Cambridge, where I work with Robert Watson's team on a capability CPU, as well as Capsicum [2]. I don't really do much on Apache anymore, but I do work on OpenSSL. And I am a director of The Bunker. Google allows me a portion of my time for some of these things, particularly the work at Cambridge and on OpenSSL. The rest I fit into my copious spare time.

Rik: Reading through your blog, I noticed the post about the CRIME attack on TLS [3]. Was OpenSSL changed in any way to make this attack more difficult? I read in the ARS article that you mention that the popular desktop browsers either were patched or didn't support compression in the first place. Is DEFLATE still supported in OpenSSL? In Apache SSL?

Ben: OpenSSL was not changed nor was Apache-SSL, because Apache-SSL is mostly not used anymore, since mod_ssl (an Apache-SSL derivative) is now included directly in Apache.

Rik: There has been a lot of talk recently (September 2013) about the NSA being capable of "breaking" SSL, and otherwise weakening crypto to make recovering keys easier. With your OpenSSL hat on, does this mean doubling the key length, which will cost big providers a lot of compute time? Or is it really too soon to tell?

Ben: Actually, the CA/Browser Forum has recently doubled key length. From January 2014, SSL keys will have to be 2048 bits.

As for the claim, I don't really believe the NSA relies much on weakening crypto—why would they, when systems are so easy to break into? In my opinion, this whole crypto thing is a complete red herring. We should be focusing on operating system and application security.

Rik: And I have another question. Since I read your blog, I know you're working on Certificate Transparency (CT). Could you explain what the goal of that project is and how it would work?

Ben: As we know since the DigiNotar incident [4], it is possible for a CA to not only fail, but fail and be undetected for a considerable time. The goal of CT is to make it very hard to get a fake certificate and hide that fact. It works by creating a publicly verifiable log of all issued

certificates. The idea is that browsers will refuse connections that are not secured with a logged certificate, and domain owners (or their agents) will monitor the log to check that issued certificates [for their domains] are all legitimate.

You can read a lot more here: http://www.certificate-transparency.org/.

Rik: You are also known for the creation of a digital currency. Tell me about what lucre was, and what happened with it.

Ben: Lucre [5] was an implementation of an idea by David Wagner. Essentially, he observed that Chaum's blind signatures, which were patented at the time, were based on RSA, and there was a parallel idea based on DSA. At least, I think it was DSA; perhaps he said Diffie-Hellman.

I implemented this scheme mostly as a learning exercise, but also to explore the costs of issuing digital money. The answer was: very cheap, even back then—much cheaper now, of course. I didn't actually do much with it in practice, though I did implement the whole system as open source. The only serious system I am aware of that was built on it is this: https://github.com/FellowTraveler/Open-Transactions. But I'm not sure how far that got.

Rik: I read "Decentralized Currencies" [6], your article criticizing Bitcoin, and understand your points about Bitcoin's problem with achieving consensus. In your article, you wrote that Bitcoin's proposed consensus group (for deciding where bitcoins reside) is based on "all the computing power in existence." And even though the Bitcoin "proof-of-work" requires more power than the value of the bitcoin produced, the fraction of computing involved in the "consensus group" is actually a tiny fraction of all computing power.

You posit that someone willing to spend a bit more power could produce a longer chain, and thus create a new proof of where bitcoins reside. It seems that you've demolished any logic behind the working of Bitcoin with these two statements, yet people still disagree with you. Could you possibly have this wrong? Is there something I am not understanding about your arguments?

Ben: People disagree with me for two reasons that I can figure out:

a) The amount of work required to make a longer chain is greater than the total work put into the existing chain; that is, to outpace the existing chain you need not just more power, but more power running for as long as the existing power has been. This isn't really an argument against me, so I don't know why people make it, but they do. I guess it's nitpicking at my claim that you need 50% of the total computing power. "Aha!" they say, "you actually need more than that, because we have a head start," which is technically correct, but doesn't invalidate the argument, and the amount you actually need is still far less than half the total, in practice.

b) They sometimes claim that there's no economic incentive to making an alternate chain. There are at least two responses to this: (1) then why are there alternative bitcoin-like currencies? (2) the attacker's incentives may not be economic or the economics may not be the simple economics of bitcoins.

Rik: Sarah Meiklejohn et al. have a paper in IMC 2013 [7] in which they state that 64% of all bitcoins are currently hoarded. In [6], you suggest that Bitcoin is not really a decentralized currency, as the people who created it might be the ones to benefit most from that. We can't tell who is hoarding bitcoins, but is it reasonable to conflate these two ideas?

Ben: Let's not mince words. Bitcoin is a Ponzi scheme. We all know who benefits from Ponzi schemes.

Rik: Since Bitcoin has a fixed cap on the number of bitcoins that can be "minted," doesn't this make Bitcoin a deflationary currency? For me, it recalls the struggle in England between the landed aristocracy, who wanted the currency pegged to land (their land), and those who wanted a currency that could grow as the economy expanded.

Ben: Clearly, the Bitcoin cap favors the Bitcoin aristocracy. As I point out in one of my papers, if you live in the fantasy universe where Bitcoin is decentralized, then there are fairer ways to distribute new, invented wealth (for example, randomly among all the decentralized participants [8]). But in the real world, the wealth of the incumbents rests entirely on the willingness of the next layer of the pyramid to continue to play the game. This is rather different from your historical example, though; there is no land to link the currency to.

Rik: With that out of the way, I have a question about Merkle trees, as that is related to the work you are currently doing. When I read about Merkle trees, the emphasis is on creating a tree of hashes; however, searching for a particular certificate in a tree of hashes boggles my mind, as it implies visiting every node until you find the one you are interested in. Is there a parallel data structure to the Merkle tree for certificates that points to the correct path for the certificate you are interested in? Or, put another way, I am interested in a better understanding of how Merkle trees are used practically, and it seems like you are in a great position to explain this practical use.

Ben: The purpose of the Merkle tree is to allow efficient proofs that

a) the log is append-only,

b) any particular item is in the log (given its location in the log), and

c) everyone sees the same log.

If you want to monitor the log for certificates of interest, you do indeed have to look at the entire log. For certs, that's not such a big deal; we've been up several months now and there's still only 2.5M certs in the log.

Rik: I've read that FreeBSD 9 has shipped with experimental support for Capsicum. I was excited when I first learned about Capsicum in 2010. I recently learned of a new application framework, called Casper, which manages the creation of sandboxes, making it easier for developers to start using Capsicum. What more can you tell us about where this project is going?

Ben: Capsicum is a capabilities system layered on top of POSIX. In short, it adds fine-grained permissions to file descriptors, effectively turning them into capabilities, and a "capability mode." Once a process is in capability mode, which can be done for it by its parent process, it can no longer create file descriptors directly. It can only create copies of existing file descriptors with reduced (or equal) permissions, or acquire new ones from processes it can communicate with; however, it is fully backwards compatible: Capsicum-unaware software can communicate with sandboxed processes, and vice versa (if permitted). It is even possible to "transparently" Capsicumize existing software by replacing libc with a capability-aware version. It is thus possible to gradually migrate from the existing ACL-based world to a capabilities-based one without having to rewrite everything all at once.

FreeBSD 10 will have Capsicum enabled by default, and we're also working on porting Capsicum to Linux [9]. We're gradually Capsicumizing system utilities, but also going after bigger fish, such as SSH, Wireshark, and Chrome.

Casper is a framework and a utility that makes it easier to provide services, such as name resolution, restricted network connections, access to keyrings, and so forth, to sandboxed processes. We're still experimenting with exactly what kind of policies Casper should use, but one of the key aspects of the design is that it is flexible: if you don't like our version, you can replace it with one of your own, either system-wide or for particular purposes. Applications can even provide Casper services directly to each other, with appropriate configuration.

In order to extract maximum value from capabilities, it is necessary to decompose software into separate modules, which, in a POSIX world, necessarily corresponds to processes. We are working on utilities to help with that (http://www.cl.cam.ac.uk/research/security/ctsrd/soaap.html), but we're also looking at a novel CPU architecture to allow fine-grained sandboxing within processes (http://www.cl.cam.ac.uk/research/security/ctsrd/cheri.html). We already have FreeBSD running on this CPU, and a version of Clang that understands capabilities. This allows us to do some pretty cool things. For example, we can modify malloc to return a capability instead of a mere pointer. This completely eliminates heap overflow at a single stroke.

I could write a whole book about the possibilities and advantages, and I invite interested readers to get in touch with us, perhaps on the mailing list (https://lists.cam.ac.uk/mailman/listinfo/cl-capsicum-discuss).

Resources

[1] Certificate Transparency: http://www.certificate-transparency.org/.

[2] Robert N. M. Watson, Jonathan Anderson, Ben Laurie, and Kris Kennaway, "Capsicum: Practical Capabilities for UNIX," 19th USENIX Security Symposium, August 2010: http://www.usenix.org/event/sec10/tech/full_papers/Watson.pdf.

[3] Ben Laurie, "Compression Violates Semantic Security": http://www.links.org/?p=1277.

[4] DigiNotar: https://en.wikipedia.org/wiki/DigiNotar.

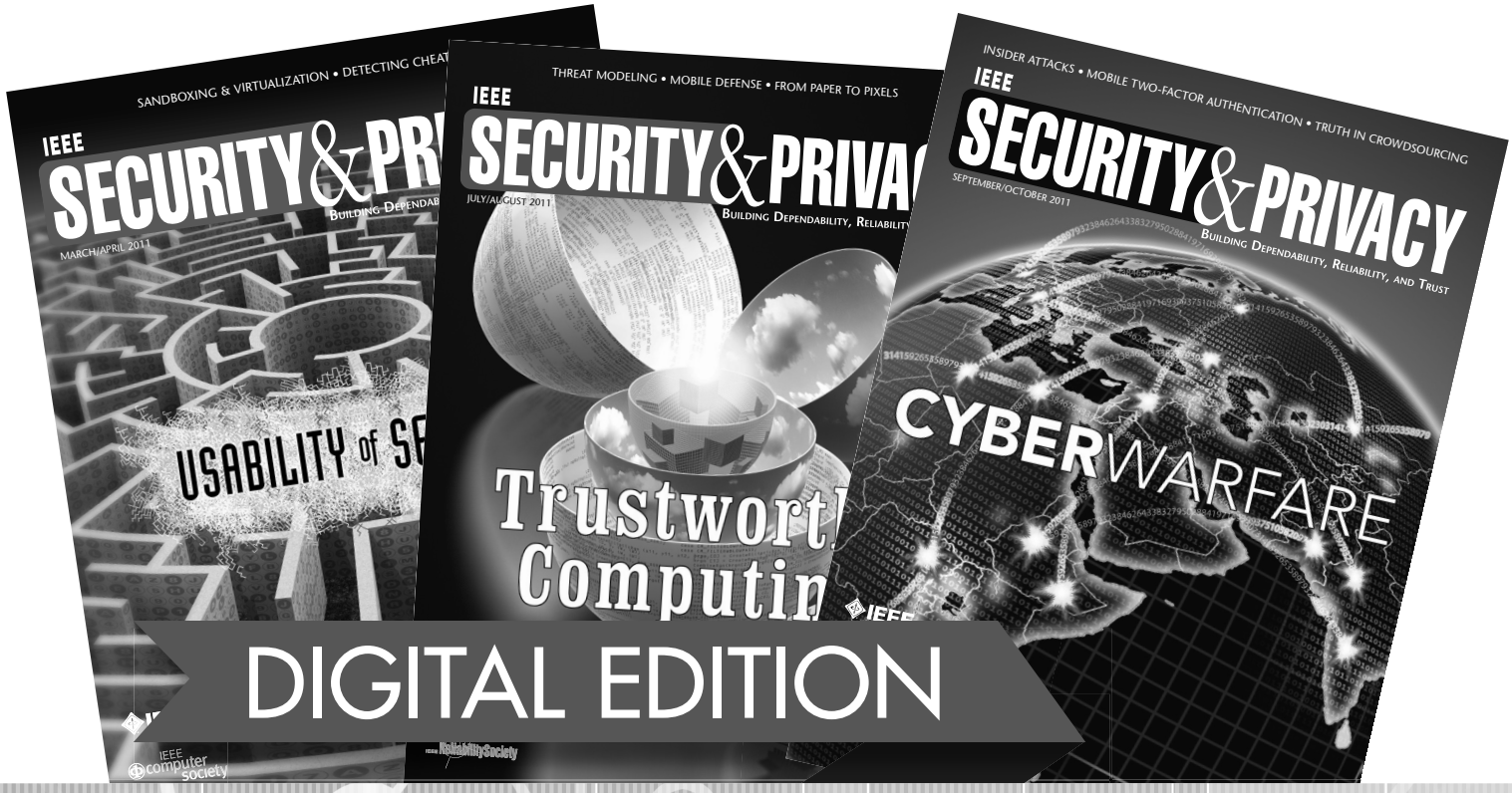
[5] Lucre: http://anoncvs.aldigital.co.uk/lucre/; available here: https://github.com/benlaurie/lucre.

[6] Ben Laurie, "Decentralized Currencies Are Probably Impossible: But Let's At Least Make Them Efficient" (critique of Bitcoin): http://www.links.org/files/decentralised-currencies.pdf.

[7] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage, "A Fistful of Bitcoins: Characterizing Payments Among Men with No Names," IMC 2013: http://cseweb.ucsd.edu/~smeiklejohn/files/imc13.pdf.

[8] Ben Laurie, "An Efficient Distributed Currency": http://www.links.org/files/distributed-currency.pdf.

[9] Capsicum code for Linux: https://github.com/google/capsicum-linux.



DIGITAL EDITION

IEEE

SECURITY & PRIVACY

SUBSCRIBE FOR $19.95

- **Protect Your Network**
- **Further your knowledge with in-depth interviews with thought leaders**
- **Access the latest trends and peer-reviewed research anywhere, anytime**

"*IEEE Security & Privacy* is the publication of choice for great security ideas that you can put into practice immediately. No vendor nonsense, just real science made practical."

—Gary McGraw,
CTO, Cigital, and author of *Software Security* and *Exploiting Software*

www.qmags.com/SNP