CA Trust - Time to Change It?

Description: After catching up with just a bit of the past week's news, Steve and Leo explore the most mature possible replacement for the Internet's existing (and failing) "trust model," which has always been based upon the unequivocal trust of Certificate Authorities.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-319.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-319-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 319, recorded September 21, 2011: CA Trust.

It's time for Security Now!, the show that protects you online against all those sharks and meanies out there who are trying to get your credit card numbers, get your information. Steve Gibson is here. He is the host of our show and our anti-shark defender, I guess.

Steve Gibson: That works. I'd be happy to be the anti-shark defender during TWiT's...

Leo: Shark repellent.

Steve: ...Shark Week.

Leo: Yes, it's Shark Week on TWiT, by the way. We have a little toy shark that floats around the studios this week. I figure it will only last a week because it's inflatable. So we probably won't have it - it probably won't be around much longer. But it's a lot of fun.

Steve: There used to be, like, networked screensavers, didn't there, where something would wander around the screens...

Leo: Oh, what a good idea.

Steve: ...within a network? And so it was like, you know, Where's Waldo or whatever this thing was, like it turned all the screens into a single looping fish bowl. And so the fish, like, swims onto your screen and swims off of yours onto someone else's.

Leo: Let's do that. I've got to find those. I don't know if those are around anymore. Nobody uses screensavers anymore.

Steve: Yeah, true.

Leo: Used to be that was fun.

Steve: True.

Leo: You don't need them anymore. So, Steve, today we're going to talk about really kind of the offshoot of Comodo Hacker's attack on DigiNotar.

Steve: Right. There have been - for years it's been understood that the existing model that we have for SSL/TLS trust that is based on Certificate Authorities, which we unequivocally trust to always do the right thing, that model is getting a little creaky and having problems. And of course in the last couple of weeks, as you said, we've really seen this with DigiNotar, which by the way has declared bankruptcy, which is a...

Leo: What? I didn't hear that. Holy cow.

Steve: ...fabulous, yup, fabulous outcome. I'm hoping that - because basically they had really lame security. We saw, if we're to trust what was posted by the Comodo Hacker guy, a ridiculously insecure password protecting their stuff. And they were vulnerable to SQL injection attacks, which just can't be allowed to happen. So I'm hoping that other security authorities will look at that and say to their IT people, okay, I don't care what it costs, we can't have this happen to us. So we may have to lose a few along the way to get the rest to pay attention; but, hey, that's a price I'm willing to pay, so...

Leo: I hate it, though, that this kid, or whoever this Comodo Hacker is, is sitting there gleefully saying, "I brought 'em down, I brought 'em down." It just makes me - it's almost like he's won.

Steve: I don't mind. I don't feel the same way. I feel like, if a kid can do this, then other people can have also.

Leo: True. True.

Steve: Maybe other people were doing it more quietly. The problem is of this happening and us not finding out about it. Because this guy generated 500-plus certs, he wasn't

very stealthful. What we really don't want is this to be done without our knowledge. So this is, well, there are several things wrong with this whole model, and that's our topic for the day, is how was it broken, and what can we replace it with? Moxie Marlinspike, our old friend, has come up with something that he suggests, which is actually based on some research and prototypes that were put together by some researchers at Carnegie Mellon. And that's really going to be the model for what we talk about as a whole different approach toward establishing trust on the Internet.

Leo: Before we get into all of that, you appeared on Channel 7 in Los Angeles last night.

Steve: I did.

Leo: With consumer reporter Ric Romero, making a case for better passwords.

Steve: Well, yeah. They picked up on a mention in Time magazine a couple weeks before about the Passwords Haystack page and the concept, and they got it. They understood what it was about. And they said, "Wow, isn't this important?" And I said, "I think so."

Leo: Might be.

Steve: So they said, "Let's tell everybody."

Leo: Well, let's watch Steve Gibson on ABC7 in Los Angeles.

[Begin KABC7 interview]

RIC ROMERO: How safe are your computer passwords? Well, I'm going to show you two passwords, and you try to figure out which one is harder for a hacker to crack. Now, this first one is going to be a series of symbols, and we're going to also put in a few letters and a couple of numbers. All right?

Now, here's the second one. This one we're going to use the letter "D," a zero, a "g," and then just a bunch of dots after the "g." Well, believe it or not, this one right here is harder. And here's why.

Steve Gibson: Making the password longer slows down their ability to figure out what the password is, and length matters more than complexity.

ROMERO: Steven Gibson is a computer hacking expert. Because every password is like a needle in a haystack, he came up with something he calls a "Haystack Calculator." It's a website that can show you immediately how easy or how hard it is to crack your password.

Unfortunately, people like to use simple passwords so they can remember them, and "123456" is the most common password. And the word "password" is also near the top of

the list. And so is this phrase: "iloveyou." In fact, they are so common, hackers actually put them into a dictionary of passwords.

So Gibson says you need to make the haystack as big as possible to really hide the needle, and Gibson's calculator shows how longer passwords make the haystack bigger. Here he demonstrates by adding one character at a time.

GIBSON: We can see that the length of time required is increasing very quickly, to the point where now we're at, in the worst case, 38 centuries.

ROMERO: Before the password is hacked. In other words, you don't have to make the password so complicated you can't remember it. Just add more characters.

GIBSON: And it doesn't matter what they are. They could just be colons; or, you know, come up with something that is like your own personal secret. And you add that to your password, and it makes it vastly stronger.

ROMERO: But the safest passwords will have at least one letter in uppercase and another one in lowercase. There will be at least one digit and one symbol. 12 characters long is optimum.

[End interview]

Leo: That's great.

Steve: Isn't that good?

Leo: I can't believe that ran on a mainstream nightly news telecast.

Steve: Yeah, I know. And I showed it to my baristas at Starbucks this morning, and they're like, wow, "I'm going to change my passwords. That's really good. I thought they had to be absolutely impossible to remember." I said, "Not so much. You want it not to be in a dictionary. But as soon as it's not in a dictionary, then length trumps complexity from there." So, yeah, I mean, it was at 5:40 p.m. in L.A. on - I guess they have the biggest market share down here, too. So a lot of people watch KABC.

Leo: Good. You did good. You did good, Steve. That's excellent.

Steve: That was very cool. Came out nicely.

Leo: Yeah. So I'll tell you what. Why don't we launch into our news and our hack updates and all that stuff, and I will get the commercial after we do those, before we talk about our conversation of the day, which is the future of Certificate Authorities.

Steve: Okay, well, we have very little, happily little security news.

Leo: Yay. I love it when that happens.

Steve: Yes. Especially since we have plenty to talk about, non-newsie stuff. I did note that DigiNotar has declared bankruptcy. Wired.com's Threat Level page covered the story, as did a number of other venues, and they said:

"A Dutch Certificate Authority that suffered a major hack attack this summer has been unable to recover from the blow and filed for bankruptcy this week. DigiNotar, which is owned by Illinois-based Vasco Data Security and was the primary provider of digital security certificates for domains owned by the Dutch government, was breached in early June due to lax security." So RIP, DigiNotar.

And I will say again, this is a fantastic outcome. It's like, yes, it's not convenient for all the people who purchased DigiNotar certificates. I don't know if there's any recourse for them except to write off the balance of amortized time and cost that they had on their certs before they expired and go buy replacements from someone else. But there's no way that a Certificate Authority can survive if none of the browsers are willing to trust them. And that's exactly what happened. As we've been covering over the last three weeks, all the browsers yanked their trust for DigiNotar in order to defend the security of the browser users. And that immediately puts the authority out of business. It's RIP time.

So, I mean, this is a great outcome because all other Certificate Authorities looking at this have to recognize - I mean, these are companies. There are hierarchies of management. Nobody ever has enough time. IT never has enough money. And so in a boardroom they're deciding who gets how much money. And the amount of money that the IT get for security is a function of the perceived need for it. So this kind of event filters down through corporate hierarchies and changes the priorities in a way that benefits us, the people who are trusting these Certificate Authorities that our browsers are trusting never to make this kind of mistake.

So as we know, security is not black and white. We need this kind of - we need the security of the remaining surviving Certificate Authorities to be maintained and kept as high as possible. So this was - I celebrate this. This is a great outcome. I'm glad that it happened as cleanly and as quickly as this. And really there was no other outcome. If browsers are not going to trust you, it's over. I mean, your entire business is issuing certs that are trusted. And if you can't do that, you're gone.

Leo: Yeah. I just dread another manifesto, another Pastebin manifesto from Comodo Hacker saying, "I'm the king of the world, yay."

Steve: I would say he earned it, so...

Leo: Okay. There you go.

Steve: I really, you know, it's like, if they're going to have something as lame as an SQL injection attack on their browser, boy - and then policies where, remember, they knew about this for weeks, many weeks.

Leo: Yes, no kidding.

Steve: And they kept it quiet. And it's their behavior as much as the breachiness of their network that is responsible for their downfall. Mozilla was livid that they were not notified. They saw indications that DigiNotar clearly recognized that they had had a breach, and they said nothing. So it's like, okay. It's one thing to have a breach. It's another thing to misbehave afterwards. So it's not only that their security was untrustworthy, but they themselves, their own response to that. Because, I mean, I would cut them slack. I would say this could happen to anybody, unfortunately. We don't want it to happen to anybody, but it can. The question then is how do you respond? And they did not respond properly.

Okay. Speaking of not responding properly, the, quote, "news," unquote, outlet that I love to hate, TheRegister.co.uk, ran the headline: "Hackers break SSL encryption used by millions of sites." This was endlessly tweeted to me over the last week. And so it does - something has happened, and it's interesting and significant, and it is the topic for Episode 321. But everybody, don't unplug your browsers. The end of the world is not nigh. We're going to be fine. They said, that is, TheRegister.co.uk said: "Researchers have discovered a serious weakness in virtually all websites protected by the secure sockets layer protocol that allows attackers to silently decrypt data that's passing between a web server and an end-user browser." And that's not true.

Later this week some researchers in, I think, Brazil, at a conference in Brazil, are going to demonstrate something that they call "BEAST," which is a cool acronym for Browser Exploit Against SSL/TLS. BEAST is described as a "cryptographic Trojan horse." And quoting from one of the articles that I had seen written up, they said, "An attacker slips a bit of JavaScript into your browser, and the JavaScript collaborates with a network sniffer to undermine your HTTPS connection," says Trevor Perrin, an independent security researcher. "If the attack works as quickly and widely as [the researchers] claim, it's a legitimate threat."

Okay. So I will, just so people understand what's going on a little bit before we cover this in really some fun detail - because this is an interesting hack. This is nothing new that was discovered. A vulnerability was understood about six years ago in our SSL protocol. The nature of the problem is that we've talked about encryption modes, modes of encryption where we take a symmetric cipher, and instead of encrypting each 128-bit block, for example, in the case of AES's Rijndael, which is 128-bit-at-a-time symmetric cipher, instead of encrypting each 128 bits, and then the next 128 bits, and the next 128 bits all independently, there is some weakness there because that would mean that anytime the same 128-bits was encountered as plain text, it would encrypt into the same 128 bits of cipher text if it was under the same key. So that's not a good thing.

So they take some residual from the prior block's encryption and mix it in with the next block's encryption to create a chain. And we talked about this, in fact, when we were talking about my Off The Grid cipher. I wanted to create the same kind of dependence between encryptions of individual domain name characters, and that's where you have this notion of sort of walking around this Latin Square where your position creates memory in the system. Well, similarly, the prior encryption block creates memory that feeds into the next encryption block.

Well, the way all encryption up to v1.1 of TLS operated was that the residual from the last block of the prior packet was used to initialize the first block of the next packet. But that's the nature of the vulnerability. Because there's a little residual crossing between

individual packets, that allows bad guys to get in there. And again, this has been known as a problem for six years, and it was fixed in v1.1 of TLS, and it's continued into v1.2 of TLS.

Anyway, it's an interesting hack. We're going to talk about it in two weeks. But it isn't the end of the world. Hackers didn't discover anything. And it's not even clear how exploitable it is because apparently you have to get JavaScript into your browser first. Well, the JavaScript, unless they've come up with a way of breaking the same origin policy, which no one has come up with a way of doing yet so presumably they haven't, then the JavaScript has to somehow come, the malicious JavaScript, in order to execute, from the site you're visiting. But if you have SSL up then, or even on an SSL page, then it's not clear how you get the bad JavaScript in in the first place in order to somehow collaborate with the sniffer.

Anyway, there's a lot we don't know. It looks like it's sort of a - it was a theoretical problem six years ago that is somewhat less theoretical today, but far from "Hackers have broken SSL for millions of websites that use it." So I wanted to let everybody know, we're going to go into it in detail in two weeks, but in the meantime don't worry about it because it looks like it really can't get anybody.

Leo: All right.

Steve: And finally, just a little miscellaneous comment. The boards for my Portable Sound Blaster have been bouncing in and out of stock. They get 50 at a time. They sold out again after last week. They're back in stock and currently have 36 in stock. I imagine this is probably - we have to be winding down. But I'm just tickled that so many of our listeners have been curious about this and are buying these little puppies.

Leo: We don't have to be. We might sell all 35 before the end of the show.

Steve: We may well. But I wanted to let people know, who may have been disappointed that it was sold out again, that it's back in stock at LPCTools.com. And you know that you're at the right page because they added a little blurb in red saying this is the one Steve Gibson was talking about on the podcast.

Leo: I love it that they did that and gave a link to the podcast. That's fantastic.

Steve: Yeah. And sort of an interesting note from Loncey Mills about SpinRite. And she said, "Dear Steve, I recently purchased my first copy of SpinRite to help with an issue I've been having with my laptop. Now, this isn't a story about how SpinRite saved me in the last second of the last minute of the last day before the big collapse. Neither is this a story about how running SpinRite on Level 5 recovered my data and found my neighbor's lost kitten in the process. No, this is something more sublime.

"Following other investigative procedures, I ran my new copy of SpinRite on Level 4 and was able to determine that my physical memory dump blue screen errors - Windows XP, folks - are not related to drive issues, but more likely due to excess heat. SpinRite told me about that. But finally, long story short" - and I think Loncey's probably wrong about this, I think it probably was a hard drive problem, as we'll see in a second.

But she says, "Finally, long story short, I ran my Windows defrag utility on my five-year-old laptop after using SpinRite and was then confronted with something I had never seen before: a 100-percent perfectly defragmented main drive. I was intrigued when I first saw the red stripes of unmovable regions that I had grown so used to beginning to disappear rapidly in the defrag window, and then I froze when they disappeared altogether and didn't come back. I had to take a screen shot to prove this to myself after the utility was done. Apparently I wasn't hallucinating. SpinRite has many uses, as I've heard on the Security Now! podcast. But this is a great side effect of the medicine that is SpinRite. Thank you. I feel better now. Sincerely, Loncey Mills, New York, NY."

Leo: I love the name "Loncey Mills," too.

Steve: Yeah.

Leo: Love it.

Steve: So thank you, Loncey, for sharing. And it's probably the case that there was something subtly not happy on the drive that was keeping Windows from being willing to relocate those areas, and SpinRite cruised across them. Even though it may have reported nothing clearly done, the drive is happier now. So probably the blue screens are gone. And certainly we got a complete defrag, which I have seen before also. So that's very cool.

Leo: All right, Steve. Let's get to work here.

Steve: I actually have two other little bits of completely random miscellany. I thought you'd get a kick out of knowing that I, Monday, made the mistake of trying an eight-shot latte.

Leo: [Laughing] No good?

Steve: Not.

Leo: Steve, you've got to build up slowly. You went from five to six.

Steve: Yeah, went five to six. I've been doing six for a while, and I thought...

Leo: Slowly.

Steve: I've been thinking, you know, six just doesn't really have much zing to it anymore. Well, boy, let me tell you. Eight shots.

Leo: Yeah, you felt it with eight, huh?

Steve: Oh, yeah.

Leo: What is that? Is that like - that's 1.6 grams of caffeine? Something like that. That's a significant amount of caffeine, of just pump right in, boom. You know, you could inject those, and you'd probably...

Steve: I enjoy my mornings on the patio, and I just don't feel cold, no matter what temperature it is out. So...

Leo: And you get a lot done.

Steve: Yeah, I really do.

Leo: So you're back to the six?

Steve: Yeah, I went back. I dropped back to six, and I'm happy there. I'm going to stay there for a while.

Leo: Yeah, yeah.

Steve: The second thing is I have sworn off any more reading of David Weber until I'm on my stair climber. I slipped into book number two, which I finished already because I just cannot put it down. And I will tell people, and I know you, Leo, especially, anyone who is not nearly moved to tears or at least really choked up at the end of book two isn't human.

Leo: Wow. Wow.

Steve: Oh, my god, it's good. It is...

Leo: So basically I'm getting this picture of Steve Gibson, hopped up on eight shots of caffeine on his treadmill, bawling his eyes out.

Steve: Not on my treadmill. That's been the mistake - or the stair climber - is that I broke down and started reading on the Kindle, which I could do almost 24 hours a day.

Leo: Ah. Now you're eating it like candy.

Steve: I am not going - first of all, I'm not touching it until I finish FreedomTM. I'm about two thirds of the way through.

Leo: I'm surprised you were able to put that down because that's a pretty - at the end especially gets very gripping.

Steve: Well, believe me, Leo, after 60 minutes with your heart rate at about 155, and saturating a paper towel that you've got rubber-banded to your forehead, it's not about putting it down.

Leo: [Quietly hysterical]

Steve: It's about I'm rereading sentences because I have a hard time understanding them any longer. So, yeah.

Leo: Oh, well. I'm just glad - I can't wait to read these books, but I am holding off. I have too much to do to get hooked into a series. How many books are there, 20?

Steve: There's 11, and there's a 12th one coming. And, oh, they're just - it's so well done. The writing is spectacular, and the characterization, the development of the people. And he doesn't mind killing off important people. So it's like, ooh, I like that guy. Whoop, he's space dust now.

Leo: Space dust.

Steve: But, oh, boy, it's just - it's fantastic space opera. So I wanted to remind people again, David Weber is the artist - is the artist - is the author, and the series is Honor Harrington. The second book, well, the first one was "On Basilisk Station." The second one was "The Honor of the Queen." So, yeah, I just, oh, wow. But for those of you who are waiting for the completion of Off The Grid, I am not reading any more of this. I actually had someone tweet me, "Steve, just put the Kindle down. We want Off The Grid finished so we can start using it." I know there's a ton of people waiting for me. So know that I am back on it. I did lose two days cleaning up my office in order to have KABC come in and do their filming. That ended up being worthwhile. So I'm back to work now, and I'm going to get Off The Grid Finished.

Leo: Good for you.

Steve: Okay. We know that the - okay. I had a note to myself. Reality Check #1: Some problems are hard and do not have good solutions. This is one of them. Another one, for example, is terrorism. People don't want to acknowledge the nature, the fundamental nature of asymmetric warfare and that there just is not a way to defeat a tactic. You can't go to war against a tactic. And so, like terrorism, the problem of trust on the Internet is arguably an intractable problem. It doesn't have a good solution. There just isn't, like, some magic thing we haven't thought of yet.

And you could probably, if you were a student of philosophical logic, you could probably assemble a proof that there isn't a good solution, that you have to trust somebody, and if you're going to do that, then they have to be perfect. And we understand that perfection is something which is inherently probably impossible for our solutions today to provide. So we've had to settle for something less than perfect.

Well, we've just seen a beautiful example of the way the Certificate Authority hierarchy fails us, and that is, our browsers trust, unfortunately, a distressingly large array of Certificate Authorities, each of whom has the ability to sign any certificate for any server on the Internet. And because we have to trust in the perfect performance of every one of those Certificate Authorities, if any one of them screws up, then we're vulnerable to a certificate that they have signed which we inherently trust. So a lot of bright people have said, okay, this is getting kind of creaky because now the number of Certificate Authorities is up to 600, and we're beginning to see cracks in this infrastructure. What can we do different?

Now, the other problem that we've talked about in past podcasts - and those of our listeners who haven't gone way back to number - I wish we did start with number zero, Leo.

Leo: That would have been the right way, the right thing to do.

Steve: It would have been the right thing. The number one. We have talked, for example, about - and I've grumbled about that these things are not free, either, these certificates that I'm buying from - I have been buying them from VeriSign. VeriSign never gets any more of my money. As we know, I'm switching over to DigiCert toward the end of the year. When the first one of my VeriSign certs expires, I'm going to move them all en masse over to DigiCert because they look really good. They haven't made any mistakes. And they're a lot less expensive for the certs that I want. I'll be able to switch up to the extended validation certs, and they just look like a great company. So I'll be moving over.

But there are many websites where arguably we don't require the authentication which a real-world process, an inherently offline process requires for someone's identity to be asserted. But we would like to have an SSL connection for privacy's sake. Which is to say that - and I'm sure all of us, probably these listeners of a podcast like this more than the general public, and I know you have, Leo, will go to a site which is sort of off the main beaten path, and maybe it's run by some UNIX curmudgeon who refuses to pay anybody any money to have an SSL connection to his server. So what he does is he self-signs. He has a certificate that he made himself, and he signed it himself. So he's just saying, "I am me, darn it, take it or leave it."

Leo: Yeah, right. That gives you SSL; right? It's just that you can't verify identity.

Steve: Precisely, because he signed it himself. Which is to say anybody could sign a certificate.

Leo: I've done it. We've all done it.

Steve: Yup. In fact...

Leo: I'm a self-signer.

Steve: I have a server that, not surprisingly, is called "Steve," which is what I use for testing my stuff locally before I publish it up on GRC. So I have "www.steve" and "steve." And I've got self-signed certificates that last a hundred years, so I don't have to worry about them all the time. And so I've trained my browsers to stop bitching about them. Yes, I trust this strange person named Steve who apparently signed his certificate by himself, for himself. And that way I'm able to establish my own local SSL connections to test the SSL paths of things I'm doing. So it's very useful for me.

And so what happens is some people will go to a site, which is not a major site, which can arguably afford the every-two-year expense of having certificates reauthenticated and signed by a trusted authority, and they don't want to. They're just saying, look, if you want an SSL connection, here's my certificate. I signed it myself. Take it, and whoever I am, you can have privacy, but you don't get authenticity because I'm not willing to pay any money to a third party to tell me who I am. So, okay, fine. But the problem is - we have a new acronym, which I really love, and our listeners will, too, which is called TOFU, T-O-F-U.

Leo: So first you have PEE, then PIE, and now TOFU.

Steve: We have PEE; then we have PIE; now we have TOFU. TOFU stands for Trust On First Use, which is inherently what we're doing. We're going to somewhere where the site says, hey, I've got SSL, but I'm not having any third party vouch for me, so trust me. Now, theoretically, if at that instant somebody had a man-in-the-middle attack up against you, then you're accepting as that site whatever certificate they give you. And you're not having it verified with a third party. So you are trusting what you get on first use, that is, TOFU. And you're telling your browser, yes, I know, it's okay, trust this and don't bother me about it anymore.

So that's a worthwhile model, which is to say, I mean, in general it would be nice. We've talked about the overall problem that we're not using SSL all the time. It would be a much better Internet if we were. Yet, arguably, lots of servers don't feel the need and for whatever reason don't want to expend the cost of maintaining certificates. So as a consequence we don't have private connections to them. We've got completely sniffable, in any public WiFi or in many LAN situations, completely sniffable communications. So, and then there are other situations where, for example, we've got multiple web domains hosted on the same IP, and that causes problems with SSL certificates because servers need to bind to an IP because you establish your connection before you exchange SSL data, and so it's difficult to disambiguate multiple domains on the same IP, although that has been addressed in later versions of these protocols which are not yet prevalent.

So we've got a number of problems. So it would be nice if there was, first of all, potentially a way of doing something better than our current certificate hierarchy, where we've got our Certificate Authority, and we trust those guys to be absolutely perfect. And it would also be nice if we had something better than, maybe not perfect, but better than the TOFU model which allowed us, for example, imagine if it were possible to have sufficient trust in self-signed certificates so that users were not even bothered. If that were the case, then it would be much more likely that all websites, or a much greater

percentage, could say, hey, let's add self-signed certificates to our server because all the browsers now support this alternative model.

So Moxie Marlinspike, our old friend, has put together - he gave a Black Hat presentation about a system which he has put together called "Convergence." It's at the website Convergence.io. And he currently has a Firefox add-on which you can download and install, which essentially lifts you up off of the traditional CA, Certificate Authority, hierarchy into an entirely different model of certificate verification authenticity which is based on what I guess I would call a "federated trust" model. It's based on some work from about three years ago that was done by folks at Carnegie Mellon. And they even produced, these guys, the CMU folks produced a Firefox add-on called "Perspectives." Their project is at Perspectives-Project.org. And anybody who's curious can find a PDF, it's a 14-page PDF from the USENIX '08 conference where they presented this. Okay.

The idea is that you would change your browser or enhance your browser, or browsers, to trust a sort of a top-level notary organizer. The idea is there's a network of geographically distributed "notary servers," they call them, because the notaries, just as a real-world notary, witnesses you signing some document and then adds their own notarization to the document, saying we witnessed this. These notaries are sort of independent probes of the SSL/TLS certificate system. The idea of them being geographically spread is you would like them to be widely distributed across the Internet so that they have completely independent links into all of the different web servers that they're probing and completely different links into you.

So the idea is this geographic spread gives us network-level robustness against man-in-the-middle attacks where a bad guy is able to insert themselves somewhere in the network. So if immediately your traffic, as we the client probes these notaries, if it heads off in different directions, then any one link or set of routes that may be compromised still leaves out others that aren't. So this geographic spread is a good thing. And the same thing is true on the side of the notaries, which are checking in with the various servers.

So the way this works is that our browser holds the public key of one or more of these notary organizers. The notary organizer is sort of the master coordinator of a group of notaries under its management. So, for example, the EFF might be running a Perspectives set of notaries, spread all over the globe. And our client that uses this system would come probably with their public key built in. And maybe other organizations would run, like well-known trusted organizations would also be running their own notary networks. So the point is we somehow receive these notary coordinator public keys out of band, meaning not over the Internet or not through normal daily processes, but probably bound into, sort of like built right into the clients in the same way that right now we have Certificate Authorities whose public keys are bound into our browsers.

So again, you always have to have some beginning trust route that you work from. So the client asks the notary organizer for the list of notaries that it manages, and it provides a list by their IP address and their public key. So, note, not their DNS name, but their IP address. So this lifts us immediately away from a dependence on DNS. So that requires that these notaries be at fixed IPs, or at least that this list be updated if those IPs should change. But they're not expected to change often.

So now we have a trusted, signed list of notaries. And we're talking about, oh, maybe 20 that are scattered around the globe. Our client maintains, that is, a browser, or whatever utility we're using that wants to establish verifiable SSL connections, maintains its own local cache of the public keys it has seen before. Now, that's different than what we have

today. Remember that today, as we make an SSL connection, we receive the public key from the remote server, which is bound into a certificate, which has been signed by someone we trust. And on the spot we verify the timestamp, that the certificate is not expired, and also that this certificate was signed by someone that we currently trust. That happens every time we make a connection. So, and because it's all local, it's very low weight. It doesn't cost any time for us to do that right here.

So we add something - so this system starts by adding something different. Now we have the notion of a cache of certificates that we have seen before and that have already qualified themselves as trustworthy. So in the typical case of, like, going back and reestablishing an SSL connection to Amazon or to MSN or to Facebook or Google, we will be receiving public keys which are already in our cache and have already passed muster. And we'll talk about what that process is next. But so in general we're seeing, oh, yeah, same key we've seen before, trust it, period. And so off we go. So there isn't a lot of overhead in the case of ever revisiting a site that we've seen before which has not changed its certificate and also which is not currently suffering some sort of an attack.

So in the case that we encounter a new certificate, the client looks and says, wait a minute, never seen this one before. Let's see what the Internet thinks. Essentially, let's see what our network of notaries thinks about it. So we use a very lightweight protocol, which - and we've talked about UDP before. There's no three-way handshake, send SYN, SYN/ACK, and ACK packets and all that. It's much more lightweight in the same way that DNS is. We randomly select maybe 10, for example, and in their paper they use a range 4 to 10, so a bunch of notaries from our trusted notary list, which we get from sort of this essential notary coordinator. And to those IPs we immediately, in parallel, send out, using a UDP protocol, so it's just a single packet, we send out a query for the most, well, for the history - and this is another important aspect - for the history of certificates that each of those geographically dispersed notary hosts has seen in the past.

They receive our single UDP query. And if they know something about that machine, they respond. They maintain a cache and a history of each domain's certificate past. So if they have not yet encountered, if they don't already have some history, then that stimulates them to go query that domain and initiate a history for their response, which won't be much of a history, but at least it's something. And again, even that is significant because we've asked the server, over a direct path from us to the server and back, for its certificate, and it gave us one. But in order to, even with no other history, in order to verify that through this system, we send out a call to the four corners using UDP, saying what does anybody know about this guy? So that goes out immediately in a whole bunch of different directions to servers which will then make a query to that domain name, if they haven't encountered it before, from a whole bunch of different directions. So immediately we get away from the notion of a single path being compromised.

Now, in the typical case, though, of a high-use website like Amazon or Google or Facebook, whatever, there will be an existing history. These notary servers typically probe all of the domains that they are maintaining a list of several times a day. And as long as they see the same public key come back as is the last one in their records, they simply update a time-last-seen timestamp. So this history of public keys, which they maintain, it has a time first seen, timestamp of last seen, and what the public key is. So those geographically distributed notaries send all this information back in responding UDP packets. So very quickly our client has, from the four corners, it has a blob of data.

Now, the last-seen timestamp will probably be different on all of them, as will the first-seen. But what happens is, if you can imagine that all of these notaries are independently probing this domain, poking at it a few times a day, asking for its public key, they're ending up with a growing level of certainty over what this domain's public key is. So if

the client receives the records from all these different notaries, which all agree about what this domain's public key is, and if there's large overlapping agreement in how long this key has been valid, and it matches the key which is pending for the conversation we're in the process of starting up - remember, we only have to do all of this if we haven't ever contacted this site before so that we don't already have its public key in our cache, or in the typically unlikely case of the key changing, which will happen, for example, every two years.

So this allows us, without a single point of trust, which we have always had so far with the CA-based technology, it allows us to establish a substantial level of trust which is also inexpensive, unlike the inherently sort of human overhead of verifying real-world identity, which is what I go through, for example, every couple of years with VeriSign, where they make me jump through some hoops. There's, like, telephone confirmation, and I've got to get Sue standing by GRC's phone line because that's the number that they've got, so she's got to be there to say, yes, this is me. And there's a lot of physical world hoops you jump through. That's never going to be inexpensive. So we would call that a way of binding the identity to the public key.

This system is fully automated and establishes sort of a geographically distributed and temporally distributed trust in the binding of a public key to a domain in a way that's automated, so it's fundamentally less expensive. And what it means, for example, I would argue that this probably doesn't replace BofA's certificate being signed by VeriSign. You still really maybe ultimately want authenticity, or authentication. And the only way you can really get that is if you really trust VeriSign to never make a mistake, and VeriSign is asserting this is really BofA.

But we've seen this break down. And what's arguably superior about this distributed system is it doesn't have a single point of failure. A bad guy that compromised a key would have some of these notaries not agreeing. If bad guys compromised one or two of these widely distributed notaries, then the notaries wouldn't agree. So we have redundancy among these notary servers. We have redundancy because we're not running all of our verification across a single link. It immediately spreads out as it leaves us, going in different directions to widely geographically distributed notaries, and their probings of the server comes into the server from their widely geographically distributed locations. So we get a lot of redundancy, a very lightweight UDP protocol, which we only need in the instance of having a non-expired certificate that we have never seen before, and one that we don't already trust.

So what the user's client also gets is something we've never had before, that is, the notion of security policies. At the moment, we've either got yes or no. Either the browser trusts the certificate because it was signed by someone it trusts, that is, in the existing model, or it doesn't. It's either good or bad. But now we have much more control over the nature of the way we want to trust. So, for example, in the clients, they're able to tell us what they have found out and bring it to us in a sort of a user-friendly way. They could say, for example, the key you've just received from this domain that you're wanting to connect to has been seen consistently across the notary network for X days. And that could be 572. And so if that's what you're told, if you're told from this widely distributed network that this is the same key that they've all been seeing for 572 days, that's better really than anything in terms of real reliability that the Certificate Authority system we have today is able to assert.

Leo: Sounds like a good way to do it. I like it.

Steve: Yeah. Or it could say we have a suspected attack. The offered key is not consistent. Only X out of Y notaries currently see it. So that would raise some alarm. Or you could get a warning: The server key has only been consistently seen for the past X days. So, for example, maybe two. Or the offered key conflicts with the cached key, but has been consistently seen for X number of days. Now, that's what you would expect to see if somebody had just updated an expiring key with a new one. You would have the old one in the cache, and this would be changed. But the notary network that would have already been probed, it would be probing multiple times a day.

And probably other people would have certainly driven that target domain's current key into their new list. You would be able to see, oh, your old key was in the list, and it's been replaced by the new key. So you could see that there was an old key that they all had, which is what you still have in your cache, but now everybody for the last five days has got this new key, and that's what you've just received. So it allows you to basically track major changes and really not fall prey to the kind of attacks that we are seeing before.

So on Moxie's site, Convergence.io, what he's basically done is implemented everything I've just said in a Firefox plug-in. And he's got a page of, like, benefits. And I'm just going to run through them, now that we've got an understanding of what this architecture is, I want to run through the things that he's claiming, and we'll just discuss it in that context. So he says his Convergence.io system is secure.

He says, "Convergence is a secure replacement for the Certificate Authority System. Rather than employing a traditionally hard-coded list of immutable CAs, Convergence allows you to configure a dynamic set of notaries which use network perspective to validate your communication." And that's exactly what we've just heard me describe.

He says, "Trust Agility. Convergence allows you to choose who you want to trust, rather than having someone else's decision forced on you. You can revise your trust decisions at any time, so that you're not locked into trusting anyone for longer than you want."

He says, "Distributed. Convergence makes it easy for anyone to run their own trust notary." Which is a good point that I didn't bring up. Individuals can run their own, if you chose. Or your organization could run its own, if you were, like, a big company like IBM with offices spread all over the place, set up trust notaries, manage your own central organizer, and then have your own IBM client browsers use this system as opposed to the centralized Certificate Authority system. So that's entirely possible within this same scenario.

He says, continuing on this topic of running your own trust notary, "Each notary can only make security decisions for the clients that have chosen to trust it. So the security, integrity, or accuracy of a notary does not affect those who have not selected it" for trust.

"Robust. Convergence can be configured to require trust consensus amongst multiple notaries, preventing any single notary from having the ability to compromise security." Which is another good point. Right now we rely entirely on one single Certificate Authority operation to sign a certificate. Here we're inherently not relying even on a single notary because, if we send out probes to 10, and a couple of them have been compromised, or a couple of the links to or from them have been compromised, they won't agree with the rest. So we can go, okay, something's a little flaky here; but the majority of them agree, so it looks like this is good to go. Or we could be really super cautious and say, uh, we're going to decide we don't want to do any trusting right now, until we figure out what this means.

And he says, "Simple. Convergence is fully backward-compatible with the existing deployment of certificates and does not require website operators to change anything." Oh, I forgot one point, and that is that, if a site wanted to make sure it's Convergence friendly, or Perspectives friendly, all - for example, say that I wanted to make sure that I was listed. All I would have to do as the site owner would be to send a request to these various notaries. If they didn't already have me in their list, they would treat me like a client. And so they would request GRC's public key on my own behalf. And so that's a way I have of making sure that they're able to respond to queries about me quickly. So that's another sort of a cool thing.

And he says, "Just install the Firefox add-on, select whom you trust, and be done with Certificate Authorities forever. Everything will look exactly the same, and you'll never get a self-signed certificate warning again." Which is one of the very cool things, which is, as we were saying at the top of this topic, going to a site which has a self-signed certificate has had the downside that people would get warnings, and I'm sure many people go, "I don't know what that means. I'm just going to - I'm going to click away from this." Now this system beautifully solves the self-signed certificate problem in a way that doesn't cause any trouble.

So I wouldn't - while this could be used as a replacement for the Certificate Authority system, I see this as really nicely augmentative, and potentially allows lots of sites that have not, before now, been able to justify the loss in traffic of a self-signed certificate to say, yeah, this is not going to raise alarms. And now we're going to be able to give people privacy, and at zero cost.

Ah. And he also says, "Anonymous. Convergence caches trust information locally, and has a mode to shield your IP address from notaries when communicating with them, so that you never leak your browsing history to anyone else." Now, that is a downside of this system which is worth mentioning. And that is that, if we don't have a domain in our cache, we are asking a third party for verification of that site's specific public key. Which means - and we're sending UDP packets, which means the notary can, if it wished to, monitor where we're going.

Now, that is, I mean, like I said, there aren't any free lunches here. The beauty of the existing system is that it is fully anonymous. We trust Certificate Authorities. There's no traffic from our browsers when we go to a site we've never seen before because it will be signed by someone we trust; therefore we trust the certificate implicitly. Well, we don't have that. If we use this notary-based system, we are leaking out to this network of notaries where we're going.

Now, I don't know how Moxie has gotten around that. Maybe, well, I just don't know. It's hard for me to imagine, and I haven't looked into it. But it is worth noting that in this case, in the case of at least the Perspective systems, we are leaking our browsing history. So that's reason, for example, to use a network like the EFF, where it's really clear that these guys are on the side of privacy and on our side, so that they're not logging and not tracking us and doing anything else nefarious with this information. But that is a downside of an active traffic-based approach.

At the same time, we were talking about revocation the other day and the OCSP problem. The Online Certificate Status Protocol has the same problem, that is, for our browser to be told to check for a certificate being revoked means that we're sending that off to a third-party server to ask if the certificate is still okay, and it's able to see who we are by our IP address, and also what site we're asking it about. So again, some of these problems do not have really good solutions.

And he says it's fast. "Convergence is so lightweight you won't even know it's there." And proven. And he says, "Convergence is based on the ideas originally developed by the Perspectives Project at Carnegie Mellon University." So that is arguably the alternative architecture that has, I think, a very good chance of gaining traction. We've got a plug-in now for Firefox. If this begins to gain traction, I could see other browsers supporting it, at least as an option, as a lightweight solution for maybe not existing signed certificates, but certainly certificates that are self-signed, if we want to say, okay, I really don't want to see these as long as a bunch of notaries whom I trust are all saying, yep, this is the one we've seen this site providing. This is probably really theirs.

So it's a neat system. And by getting around the need for real-world verification of identity by binding it through an automated system rather than a physical system, we end up being able to do it at no cost and very little overhead. So that's the alternative. And no one's come up with anything better.

Leo: I like it. And thank you, Moxie Marlinspike, not his real name...

Steve: Yeah.

Leo: ...for the plug-in. And the idea. Do you think there's - okay, let's be honest. Is there a chance in hell this'll happen?

Steve: No.

Leo: [Laughing] We have such an entrenched system. Although the notary addition doesn't seem like such a big deal.

Steve: Yes. And, I mean, if we - okay. So there's really no way to bring up ubiquitous SSL except to solve the problem of self-signed certificates annoying users. This really does that. This really, if all the browsers had this system - and if they did, of course we would then add nice, simple-to-use utilities that all webmasters could use to generate self-signed certificates - then this would be really cool. And, in fact, you could imagine something nice where a header in non-SSL communications could mention that, in the same way that we have a do-not-track header, there could be a header saying "I support a self-signed cert." Then that would tell your browser, oh, let's bounce back to this guy over SSL rather than over not, knowing that we've got a self-signed cert, and that would allow us to have privacy. So you can see that it could be implemented in a safe and clean fashion that would allow people to switch over to privacy by default, rather than no privacy by default, which would certainly be an improvement for the Internet. That I could easily allow this, I could easily imagine this system doing.

Leo: It's the kind of thing that - I'm just trying to think of who would, who could use, who could strong-arm this kind of system. Is it ICANN that could kind of push this kind of system in? Who...

Steve: Well, actually ICANN, or the IETF...

Leo: IETF, that's who could do it.

Steve: Yeah, they are looking at...

Leo: Because it's their standard.

Steve: Well, they're looking at a DNSSEC-based approach, where secure DNS would be used to independently offer certificates instead of the Certificate Authority. So the idea would be we would use the next-generation signed DNS records in order to say, okay, this is a way of now we're getting the IP address in a way that we absolutely can trust, and we're getting the domain's public key, similarly, in a way that we absolutely can trust, not with a Certificate Authority. So that's the direction the IETF is approaching, or pursuing. And the problem is we're still a long way away from DNSSEC being ubiquitously deployed. But that could happen, too.

Leo: Steve, hope springs eternal in the secure mind.

Steve: And so does Honor.

Leo: And Honor, yes, Honor Black - what is her name, last name?

Steve: Honor Harrington.

Leo: Harrington. She didn't get killed. You couldn't kill her.

Steve: Oh, no. She got...

Leo: She's the star of our show.

Steve: She got some serious commendations.

Leo: I can't wait.

Steve: Oh, Leo, you'll be choked up.

Leo: You can just see Steve wanting to go back. I can't wait to read it.

Steve: Oh, I desperately do. I'm going to be in such good shape because I'm just going to have to be on the stair climber in order to read, start into book three. But I'm going to

get Off The Grid finished first.

Leo: That's a brilliant way to do it. Say I'm only going to do this when I'm exercising.

Steve: Exactly.

Leo: Steve is at GRC.com. That's his website. That's where you'll find SpinRite, his bread and butter and the world's finest hard drive maintenance utility. He also has a lot of great stuff there for free, including Off The Grid and...

Steve: Almost.

Leo: ...Password Haystacks. Well, you've got the Off The Grid concept there.

Steve: The concept's in place. I've just got to get it nailed down so people can actually use it. And I'm really pleased that so many people are, like, panting. It's like, hey, I want to use this thing. So it's like, oh, cool.

Leo: Yeah, Steve gets a lot of thrill out of this. He's definitely an enthusiast, and that's what makes it so...

Steve: Why I do it.

Leo: Yup. You'll find also 64 and 16KB versions of the audio of the show there, as well as transcripts, at the Security Now! page on GRC.com. We have audio and video on our page at TWiT.tv. And of course you can watch us do the show Wednesdays at 11:00 a.m. Pacific, 2:00 p.m. Eastern, 1800 UTC at live.twit.tv or TWiT.tv. So you can watch live or get it after the fact. A lot of times people watch live and then get the show and listen again, then get the transcript and read that. And by the third or fourth time through, they understand it.

Steve: Oh, boy.

Leo: [Laughing] Hey, there's only 319 of them to do that with. I was talking to somebody who's a professor, uses it. I think a lot of college professors use it in computer science classes and security classes.

Steve: Yes. We have heard a lot of feedback from people saying, hey, this has been assigned to me in my class. So...

Leo: Steve, that concludes this edition of Security Now!.

Steve: It does. Next week we'll do a Q&A. So anybody who's got any questions raised by this, I imagine there will be some, by all means, GRC.com/feedback. Drop your note to me there, and I'll see them, and we'll address them next week.

Leo: Excellent. We'll see you next time on Security Now!.