SQRL

Description: After catching up with the week's minimal security news, Steve and Tom take the wraps off of "SQRL" (pronounced "squirrel"), Steve's recent brainstorm to propose a truly practical replacement for always-troublesome website login usernames and passwords.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-424.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-424-lq.mp3

SHOW TEASE: Coming up on Security Now!, it's my last time filling in for Leo Laporte. We've got a new way to think about fingerprint security. We've got some good news about IE6. But Steve Gibson has come up with a way to virtually eliminate the need for a password to securely log into websites on the Internet. You've got to watch this episode, next.

TOM MERRITT: This is Security Now! with Steve Gibson, Episode 424, recorded October 2nd, 2013: SQRL.

Hey, everybody, it's time for Security Now!. I'm Tom Merritt. Sadly, for me, the last week that I'll be filling in for the vacationing Leo Laporte. And we have got an episode for you. Steve Gibson, the man from GRC.com, the man who may have just come up with a way to pretty much free us from passwords, joins us now. Steve, I'm really excited about today's topic.

Steve Gibson: Hey, Tom. It's great to be with you again. Well, this was supposed to be, in our alternating topical and question-and-answer podcasts, this was supposed to be a Q&A because we of course talked about fingerprint biometrics extensively last week. But the way the timing all came together with my getting to a position where I had enough worked out and documented of this idea that I've been teasing our listeners with now for, I don't know, five or six weeks, when it just hit me during breakfast one morning. I was sipping coffee, and it just was there. And I thought, wait a minute, does that work? Then I thought about it some more, and the coffee got cold. So I got more coffee because, you know, you need that.

TOM: Well, yeah, absolutely.

Steve: And then I was working on the tail end of weirdness of the new SpinRite code for dealing directly with hardware controllers on motherboards. And there were a couple people in the GRC newsgroup - we have a grc.spinrite.dev, which is where the development work goes on, and that gets fired up about every decade or so, when it's time to do a new SpinRite. And so there's been frantic participation in that newsgroup.

And a couple people had these just weird, oddball, old, but they had them, controllers that were just behaving weirdly. Most people would run all the test code, and everything was fine.

But anyway, I wanted to really wrestle this thing to the ground because I have people who are willing to test my code, and I don't want to let them go. So for a couple weeks my main focus was that; while pretty much like every shower, every time I was driving, I mean, every other time when I couldn't be working on SpinRite, I was thinking, okay, let me test this again. What has this da da da da. And it just kept looking like I actually had an idea. As I mentioned, it happened that I had one of my infrequent marathon phone conversations with Mark Thompson of AnalogX, who's a technical wizard and good friend. And so since he was on the phone, this was on my mind, and I completely trust him, I shared it with him. And he got it, to his credit, instantly. Actually, it's not that complicated. I mean...

TOM: No, that's the beauty of it, yeah.

Steve: You got it, too, because I shared this with you yesterday when I had the documentation finally ready so that I could, you know, I had something that conveyed it clearly. And anyway, so finally about I guess maybe two weeks it's been, maybe 10 days, when all, I mean, I finally said, okay, this phase of SpinRite work is done. Of course, this is an interruption, of course, to the work I'm doing on SpinRite. But everybody felt that it was important enough to suspend SpinRite just to get this published. I don't know where it's going to go. I mean, it's not something I own, except as being the father of it. But, I mean, it's - replacing usernames and passwords is bigger than me. It's like, this should just be done.

TOM: And it's saving the Internet from itself, essentially, yeah.

Steve: Right. So my hope is that I can be involved enough to work out other details. One thing we need is interoperability so that, if this thing happens - and it's just hard to see why it won't. It's such a low-friction solution. Anyway, I realize I'm sort of stepping on my own story here. But my point is I just want to sort of say, here this is. I did just this morning create a newsgroup at GRC, grc.sqrl, because that's the name of this thing, pronounced "squirrel," SQRL, where I'm sure there will be huge, interesting, fabulous discussions because we've got a whole bunch of really smart people, a lot of crypto people, and just - this thing's the kind of thing you need to have pounded on for a while.

So anyway, we didn't have much news of the week. So I guess it all works out. And what I think - what we may do is, because I know that questions are piling up, maybe when Leo gets back next week, maybe we'll make up for having skipped some Q&As by doing a couple in a row, based on how many questions people have, even about today's podcast.

TOM: All right. Let's get into the security news, starting talking a little bit about fingerprints as, well, are they usernames or passwords? That's what this story is about.

Steve: Well, yeah. Actually, a number of people brought this to my attention. I'm not sure how it got as much coverage as it did. But I got a bunch of tweets incoming, saying, hey, Steve, have you seen this? Some guy named Dustin Kirkland did a blog posting, and really the title of the posting says it all. And it's really - it's an interesting posting that's worthy of some discussion. And the title of his blog posting was "Fingerprints Are Usernames NOT Passwords." And I just thought that was an interesting position. And it certainly has some merit to it.

TOM: Sure.

Steve: His argument is - we talked about Touch ID and fingerprint biometrics at length last week. And the fact that, as we know, they are spoofable, specifically the Apple extra high-resolution reader, which required that the spoof be even higher resolution so that, if the fingerprint reader saw pixels, it would say, okay, people's fingers don't have pixels.

TOM: Not a finger, yeah.

Steve: So you need to raise the resolution of the image that you're creating the finger from to a substantially higher resolution than the resolution of the image capture scanner. So they did that, and then they were able to say, okay, look, we're still able to spoof fingers, even at this high resolution. And then of course the other arguments are that many people have against using biometrics is that they're not changeable, whereas you can change your password if it gets out of your control. If you're using it at a website, and the website's database gets breached, famously, how many people have received email, or has anyone not received email saying, oh, my god, you must change all your passwords immediately because we just lost control of them.

So the problem is, if you use a biometric and the website has that, well, you can't change your fingerprints. You can't change your iris print and so forth. So what I liked about Dustin's proposition is that a fingerprint is a name for you, like an alias, and your name doesn't change. So your eyeballs don't. Your fingers don't. So I just - I thought that properly couched sort of a statement of where biometrics deserves to be. So, yes, and actually this speaks to the notion that was raised by several people commenting on Apple's Touch ID, and that is it ought to be one factor of two, that is, yes, use your finger to unlock your phone. That's a perfect first step. But if you're then going to do something critical - and maybe that's enough for casual use of the phone. But if you're going to do something critical, then still need a password. Don't solely rely on the fingerprint as complete verification of who the user is. So anyway - go ahead.

TOM: Oh, yeah, yeah, it's interesting. It's not exactly the same as a username, obviously. Someone can more easily imitate your username than they can imitate your fingerprint, although both are possible.

Steve: Right. This lies somewhere on a spectrum between...

TOM: Yeah, right.

Steve: Yes.

TOM: I just thought of this just now. You can change your fingerprint. And I don't mean burning off your fingers or anything crazy like that, but you've got 10 fingers. You just can only change it 10 times, and then you're done.

Steve: Yes.

TOM: It's a limited amount of iterations there. But, yeah, I think it is much more useful to think about it like a username. That's a smart post from Dustin.

Steve: Well, yeah. And I did see people, I mean, there's been obviously a lot of discussion following the whole Touch ID, bringing this topic of biometrics and fingerprints back to the fore again. So people said, well, don't use the obvious finger. Don't use the thumb of your right hand or whatever you're expected to use. Use the pinky on your left

hand. And maybe don't let everyone know that that's what you're doing so that, if you're in a situation where the authorities are saying, okay, we need you to unlock your phone for us, you say, oh, okay, and without hesitation you use the thumb of your right hand, and then it doesn't work. You go, huh. You clean it off, wipe it, lick it, do whatever you do, then, like, try it again. And as we know, after what is it, five misfires, the system locks. And so you could easily say, oh, shoot, I forgot it was supposed to be my left thumb. Well, okay, do that a couple times, and you're pretty much down at the line. So...

TOM: I use the CLEAR service to get through the line at SFO when I fly from San Francisco these days because I live in L.A. now. And they use a fingerprint to identify you, along with a card. So it's something you have and something else you have, I guess. Not the right way to do two-factor. But anyway, that's how they identify you. It didn't work for me this weekend when I was flying back from TWiT. Thumb three times, and they're like, okay, try a different finger. So they had backed up all my other my fingers, and I was able to get in that way. Kind of a different biometric way of going about things. Thought that was interesting.

Steve: Yeah, I've run across people whose fingers just will not scan on the laptop-style, swipe your finger on the sensor. I don't know what it is. I mean, there was a friend I had at Starbucks who asked me to help her set up a new laptop. She had a Dell laptop that had the standard little strip scanner where you swipe your finger. And since I had set it up, I had registered mine and hers. And no matter what she did, she could not get it to recognize. And then I would do it, and it would work the first time for me. So it was like, okay, I don't understand what's going on here.

TOM: All right. We have IE6 news. But I think in this case we could almost say it's good IE6 news, which is kind of different.

Steve: Oh, it definitely is. One of the things that we've talked about on the podcast from time to time is that the orphan IE of Microsoft, Internet Explorer 6, which, while it was there, it was there for a long time for major, as a companion to major versions of Windows. And it was, for whatever reason, a large, huge-market-share browser. And of course it has the mixed blessing of working. And so when Microsoft did 7, and then 8, and then 9, there were a lot of people who just stayed with 6. And there may have been clear reasons why they couldn't upgrade. There may have been for me because Microsoft was changing the browser each time. So there may have been compatibility issues where the particular corporate infranet - wait, infranet? Intranet - Intranet only ran on IE6 and so people were stuck using IE6. And of course the problem is that it's now so old and unsupported that new problems that occur in the newer browsers, the new versions of Internet Explorer, are no longer being back-patched to IE6.

Yet the problem has been, I mean, and Microsoft has, like, launched campaigns to try to reduce the market share of Internet Explorer 6 because it was becoming a serious problem, an embarrassment for them that this old browser, they just couldn't kill it off, and that it had so many security problems, which kept getting found in the later browsers which were being fixed. Anyway, the news is that the global market share of Internet Explorer 6 has finally fallen below 5%.

TOM: [Whistling] Hooray.

Steve: Yay [laughing]. Yup.

TOM: Standing ovation.

Steve: Took a long time to happen.

TOM: How far does it have to go before we just say, okay, it's irrelevant? Because 5% still is too many, in my opinion.

Steve: Yeah, 5% is one in 20 people. So that's a lot of people still. I would say 1%, at that point you consider it - if you're down to 1%, then you have to ask, when the total percentage is that low, then you start asking, okay, who are they, and maybe they deserve to get the trouble that they're asking for by using any browser that is that far gone that only one in 100 people have it. And I wonder, actually, if we're not seeing this reduction in IE global market share, not because people are moving to newer versions of Internet Explorer, but because they are dropping IE altogether in favor probably of Chrome No. 1 and Firefox No. 2, since we know now that the Google Chrome browser market share is now the largest one in the world, with Firefox in second place. So it's not that people are like, okay, I guess I'll upgrade IE6 to IE9. They're probably on operating systems that can't run IE9, for one thing, because they're way back on Windows 2000 or maybe XP. But probably it's just that IE6's share is dropping because they've switched away from Internet Explorer altogether to one of the newer and better alternative browsers.

TOM: GeekCanuck is asking how many of these are behind NAT and invisible. Well, if they access the Internet, they're still visible. They still count as an instance of Internet Explorer. I guess there could be people using Internet Explorer on a LAN without accessing the Internet, they're just accessing a local Intranet. And those actually aren't problematic because they're not accessing the Internet. But most of them are going to be accessing both. So I think most of the usages are caught up in this number.

Steve: Right. And every single time, as we've discussed through the technology of the way this works through the years, every single query that a browser makes by default contains a so-called "user-agent header." It's the metadata, which of course has gotten a bad reputation post-NSA. But in this case metadata is the stuff that is not seen through the browser window, but it's the background management of the query and response. For example, cookies are metadata of queries. And so it's easy for somebody in a central position who is monitoring queries going over the Internet, looking at, for example, server logs. Anyone who has a very high-volume popular web server can log the user-agent headers of all the queries coming in and look at the distribution of them because the user-agent will say what the browser is. It'll explain its make and model and, typically, like a whole bunch of other extraneous data, like what versions of plugins it has and so forth.

TOM: All right. And our last bit of news here is the new BitTorrent Chat client, which I have signed up, haven't got access to it yet, but I'm looking forward to trying this thing out.

Steve: Well, what really frustrates me is lack of documentation because we can't - we're not - I mean, from our position, the Security Now! audience, all we care about is the crypto protocol. How does it work? For example, that's what I'm going to completely disclose today in this notion I have for replacing usernames and passwords. That's all that matters. It's like, here it is. What does everybody think? Let's go. But you can't do that with BitTorrent Sync, which they have not documented, or BitTorrent Chat. And so it's like, well, okay, there it is. I can't say anything about it because - I could say we hope it's good. But until they release what they're doing, it's useless. I mean, it's just - it's nothing.

So all I can say is - what I wanted to say was, to all of our listeners, in addition to

BitTorrent Sync, which there it is, and we don't know what it is, but they've got, oh, it's 256 bits of this and long bits of that, but no documentation other than that. It's like the people who say, oh, we've got military-grade encryption. It's like, okay, but what are you doing with it? Everybody's got that. So there is now something in alpha. This is alpha signup: labs.bittorrent.com/experiments/bittorrent-chat.html. That'll bring you to a signup page, which as you said, Tom, you have used - I'm in no hurry to, but I'm glad you're doing it - to get the alpha test. So they're very early. They have something. And obviously it's supposed to be secure chat. It's supposed to be no storage of your chats, no ability to be intercepted, I mean, we were talking about, what was it, I'm blanking now, the other chat system that was loosely based on Bitcoin's protocol, but not really.

TOM: Oh, right. I'm blanking on it, too, sorry.

Steve: But anyway, so the point is, naturally we're seeing, as we expected post-Edward Snowden and NSA revelations, and we're going to continue for some time, and at some point we'll start having good things. I mean, this might be good. But it's useless until we know exactly how it works, until...

TOM: Bitmessage. Sage got it in the chatroom, Bitmessage. Thank you, Sage.

Steve: Thank you, yes. Until they tell us, here's the protocol. And then smart, protocol-savvy people look at it and then can say, okay. I mean, that's what happened with LastPass. They were completely open kimono. I got all documentation from them, a complete explanation. They were even able to demonstrate that's what they were doing by creating a web page of JavaScript that was not obfuscated, that exactly duplicated the functionality so that, I mean, it was, again, complete disclosure. That's the way I was able to say I understand everything done here. I see no errors, no problems. I'm using it without hesitation. And there came my endorsement. In this case, no one can do that. No one can responsibly say use BitTorrent Chat or BitTorrent Sync until they tell us what they're doing. So it's good that they're doing it, but we just have to wait to get the details.

TOM: And it'd be one thing if they're saying, we're in alpha, we're only going to give that documentation to a restricted number of people. That would make sense. But that's not what they've done with Sync; right? They have Sync now available. I can download it right now. But you're saying they still don't have the proper documentation.

Steve: They've said they're open to disclosing it. Okay. And I have a relationship with their PR guy, and he keeps sending me marketing announcements. It's like, oh, look, we've got a pretty website design now. It's like, okay.

TOM: Great.

Steve: That's good for you. But I want the protocol. Really. Don't call me until you've got the protocol. That's what we need. So we just need that. And there just can't - they can't be serious without that.

TOM: We have one little bit of errata from last week's episode regarding a location...

Steve: It's funny because as I was saying it, I knew it was wrong; but I had already sort of started the sentence, and I was committed. But a number of people noted in listening to the podcast over the course of the last week that the NSA is not in Langley, because I was referring to the NSA as being in Langley, Virginia. That's where the CIA is. The NSA is in Fort Meade, Maryland. So a little errata just for keeping the record correct.

TOM: I wish I would have caught that for you, too, but I missed it, as well. And I should know better, too. Favorite tweet of the week? This is pretty funny. I like this one, too.

Steve: This one came in yesterday. In fact, I should have written down who sent it so I could give them credit. But so I got @SGgrc, someone tweeted: "I hope they shut down the government cleanly, or they may need a copy of SpinRite later." And then he said, "Take a check?"

TOM: Do you have a SpinRite that can handle that platform, though?

Steve: Let's restart the federal government. You never, you know, you hope it comes back online. Speaking of coming back online...

TOM: Yeah, sure, go ahead.

Steve: The other thing that was a hoot was - and someone sent me through Twitter a snapshot of his browser. If you can bring that link up, Tom, you'll get a kick out of it. And that is that the BarackObama.com website SSL certificate expired on October 1st, yesterday, at 7:28 a.m.

TOM: And guess who they can't pay to come in and fix it? Anyone.

Steve: [Laughing] Exactly. So presumably whatever it is, whatever IT system or who knows what the structure is for renewing that certificate, but that may be sitting there for a while, not something they can fix. And it's funny because years back when I was talking about SSL certificate expiration, and actually I was more annoyed by it back then, probably because I was still using VeriSign for my certificates, so I was - really an expiration was a painful event because it was so expensive. Now that I'm moved over to DigiCert I'm so much happier. It's like, oh, it's going to be fun to have it expire. I don't mind this at all. And not that expensive, was my point. But so I was grousing a little bit about the whole - how ridiculous it was that you were being asked to pay, like, seven or $800 for bits.

Now, first of all, that price has come way down, and I'm getting much more for my money thanks to using DigiCert. But I also really better appreciate the value in this rolling expiration system that the public key crypto system has. I mean, it is our way of solving a number of problems. If someone lost control, if a website lost control of their certificate, we know that that's a problem because that would allow others to potentially spoof secure connections to their domain, essentially using their certificate illegitimately. But we solve that problem by informing web browsers that that certificate is bad. I mean, certificates all have essentially a hash of their contents, which they cannot change without invalidating the certificate. So web browsers can be told, from now on, never trust a website that offers a certificate with this hash.

Well, the problem is, if certificates live forever, those lists would have to grow forever. And so thankfully certificates expire every two or three years. After we're sure the certificate will have expired based on its date, then the prohibition against accepting it based on its content we're able to prune from browsers so that doesn't have to grow forever. And so there are many benefits to it. So anyway, BarackObama.com's SSL certificate is expired as of yesterday, which I thought was sort of a bizarre event.

TOM: Interesting. All right.

Steve: And since we didn't have much news today, I was saving something for when Leo

got back, but I'll just sort of share it. It's just completely random. But my girlfriend just got a book published, which is actually No. 6 on Amazon in science fiction and fantasy for large print books. Jenny's book - and people have heard me talk about my girlfriend Jenny from time to time. It's comparative religion for children. The title is "Is God Real or Pretend?" And first of all, I loved the title. From the moment she told me the title a couple years ago, I thought, oh, that's just - I just love the title. And so I'll just read briefly, for our listeners who may be interested, the description that is there up on Amazon.

It says: "'Is God Real or Pretend?' is the story of young Franklin" - that's, by the way, Benjamin Franklin. Jen has, like, deeply studied history and biographies, and Benjamin Franklin is one of her favorite people from U.S. history, probably just any history. And so anyway, so she gave this child in the book Franklin's name for Ben. It's "the story of young Franklin's engaging and enlightening journey to answer this age-old question. Franklin's grandmother, Dr. Wendy Knowles, a professor of astronomy, first provides Franklin with the basic scientific means of determining what is real and what is not, and how science distinguishes questions it can answer and those it cannot.

"Franklin's mission of discovery continues as he meets a kindly professor of Greek mythology who offers a historical-cultural perspective on the question. Here Franklin meets the Greek gods and their timeless myths. Once armed with these new ideas, Franklin meets with representatives of the world's five major religions: Hindu, Buddhist, Jewish, Christian, and Muslim. These knowledgeable teachers from each of the great religions charm and delight as they shine positive lights on their religion. Franklin asks probing questions, while learning to appreciate and admire the diversity and beauty of these religious beliefs and traditions.

"Ultimately, Franklin's dynamic school report on the immensity and magnificence of the universe becomes the backdrop for thinking critically about religion and questions about God. This book is designed for anyone and everyone, young and old, religious or not, who wants to know more about these five great religions. It's the most unforgettable and exciting journey, one every thoughtful child and curious adults in their life will enjoy."

And I have to say from what I've heard from Jen that editors and people involved in the production of this who read the book were saying, wow, you've got to write one of these for adults. So it sounds like she did a pretty good job.

TOM: Yeah, it does. That's a really fascinating concept for a book, too. That just sounds - I want to read it.

Steve: Well, it's not long. It's 66 pages, large print, illustrated. And it's the sort of thing that a parent might read with their kids, as their kids start asking questions about religion and God. It's like, well, here's a context in which we can think about that and answer the question.

TOM: I know you don't have it in the lineup. Did you watch "Homeland"?

Steve: Oh, yes [chuckling]. I actually, I don't know if I should say this, I think I know what's going on. But I don't think I'm going to say.

TOM: Already, wow. Okay.

Steve: Yeah. I think, well, actually, I love "Homeland." And it just - I care about the characters, and I really enjoyed this first episode. And in the, I guess it was the - I don't

think it was a preview of this next week, but it was - what they're doing more now is "This season on 'Homeland...'" and so you're getting snippets from various, it's like, through the future of this season of the series. I think that gave me a sense for what's happening. And anyway, so, yeah. I'm really enjoying it.

And I have to say, I've mentioned this before, when I discovered this series, I was talking to Leo, and I said, "Leo, this is not a series I ever watch." People kept recommending it to me. I'm talking about "The Good Wife." And it's just - it's the title. It just doesn't sound like something I want. I mean, I want "Firefly," and I want "Attacking Demons from the Netherworld" or something. Not "The Good Wife." Anyway, I love the show. And it also began on Sunday. And it was wonderful. So it was. It's just really great television entertainment. And of course we all...

TOM: I may have to give "The Good Wife" a chance.

Steve: I would. I think it's really good television. And there's a huge backlog of, I don't know what, maybe four, five, six - it's been on now for many seasons. But compelling characters. The characters are really well crafted, and you care about them, and it's just - it's just an engaging series. And of course we also wrapped up "Breaking Bad," which - and everyone, I guess, I guess the final series conclusion, which was last Sunday, got a fabulous - was very well regarded and well reviewed.

TOM: Yeah, record-breaking for AMC.

Steve: Yeah, 10-point-something million viewers. And the creator said - he was interviewed on "Talking Bad" afterwards, and he said that the first episode, I guess there was a collision with some sports event that was also being televised, and it limped in at, like, a million or so. And of course it grew over time. But, yeah, it was a great wrap-up.

TOM: All right. Well, I can't hold back any longer. Let's talk about SQRL.

Steve: Okay. So what do we want in an ideal online authentication system? What do we want to replace usernames and passwords? As we've developed more maturity on the Internet, as the range of services has grown, I think that our interaction with the 'Net has expanded. There are places where we have a fiduciary relationship, like with our bank, or to some degree with Amazon, I mean, where they really do know who we are. They've got our name and address. We've got accounts. There's financial information shared, or there's products being shipped to our home or whatever. So there we're not anonymous, but we want security in being known.

But there's another whole aspect where we really do want anonymity. There may be places where we really need anonymity or just places where it's not important that we be known, like making a posting to some random blog. I mean, I know that sometimes I'll see someone's blog posting, and I'll note some errors that'll stimulate me to want to reply. And so I start to reply, and suddenly it's like, oh, you have to create an account. And it's like, oh, my goodness. Then they want my email address, and I have to send them - then I'm going to get a link and have to verify myself, and they're going to want this information. And I just say forget it. It's not worth the overhead of having to essentially decloak myself for this, just to make a posting to someone's blog.

TOM: Yeah, it's two things going on; right? You don't want to reveal your true identity, and it's not frictionless. You have to go through a bunch of trouble to do something you don't want to do.

Steve: Or, yeah, or it's they've created a bar such that I'm just not going to bother. It's like, if you want me to do all that - I mean, and frankly, I see that when I'm purchasing stuff, too. I mean, it's a different case. But I'll go to somewhere, and I'm being asked to create an account, maybe not to purchase, but to do something. And it's like, eh, it's just not worth it. It's like, thanks anyway. And so the point is that, because that's the current model for identifying people, they're missing a lot of blog postings. Other sites are missing a lot of potential long-term visitors because what they're offering just isn't compelling enough. And we also know, oh, my god, now I need another password. I can't use the same password I always use because we know that's not safe. Well, in fact, I can't always use any same password. I have to have a different password for every site.

So there's, like, there's so much burden now to do something where - and I understand they want a relationship with me. But it's like, I don't want one with them. As they say, I'm just not that much into them. So, but still, they're missing something. So first of all, there's a range of sort of depth of identity and authentication that Internet services have reasonable expectation of, and Internet users vary in their willingness to disclose.

Also in this day and age we would like to be, arguably, anonymous, that is, untrackable. I would like an identity that I use with Amazon and my bank not to be obviously related to one that I arbitrarily use when I'm posting something somewhere entirely...

TOM: I think the worst example of that is Facebook; right? Like login with Facebook. Now Facebook knows everything about you on that site, and that site knows everything about you on Facebook.

Steve: Right. And remember also that Facebook knows that you logged in over there because the whole OAUTH - not OATH, OAUTH - technology has the site you're logging into going over to Facebook. Facebook sees that transaction and knows that that's where you're logging in through them.

TOM: They want this, yeah.

Steve: So all kinds of networking interaction there. So we'd like to break that. We'd like to have no obvious connection of our identity among different sites. It would also be nice, in a perfect world, to avoid keyboard interaction.

TOM: What?

Steve: Because how many times, I mean, it would be difficult for me - I don't think I could, most of our listeners probably couldn't, log in at a library computer, put in their username and password in a library, or in some sort of public access terminal. We've talked about this often. You'd be crazy. You have no idea what malware is in there. Many times there have been hardware keystroke loggers, little modules, little pods stuck in the cable in the keyboard connection going into the back of the machine, which just is logging into EPROM everything everyone types. And then that thing can be polled remotely and have its contents sucked down. So you'd be crazy to, like, do this in any sort of an insecure setting, to enter important credentials through the keyboard.

Now, the other thing we'd like to have is this notion of no shared secrets with websites. The whole shared secret thing is a problem. For example, a password. Our password at every website we use is a secret that we share with them. We know what it is. They know what it is. Problem is, they're proving themselves to be unable to keep our secrets.

TOM: Right. You hear about that all the time.

Steve: As I was saying before, yes, exactly, it's like, oh, my god, we just - such and such just lost a quarter million of their user accounts, including their - now, maybe they're hashed passwords. That's good. Except that, if they didn't do their security really correctly, as we've also seen, that, like, has my password escaped, there are sites you can go to that have already run the password hashes through high-speed cracking of password lists to see if they can crack your password. So in general a shared secret is a problem. Now, even more recent technology is the Google Authenticator-style approach.

TOM: Oh, yeah. No, I use that for the two-factor authentication; right? I've got to go to my phone and get that little code.

Steve: Yeah. Well, and I've got the original little PayPal football here that I'm holding up in front of the camera, which you press it, and it gives you a six-digit code. That's the OATH technology, which is in this case a time-based varying six digits. And the Google Authenticator is exactly that. But the reason you have a whole array of accounts is we're back to the shared secret problem.

TOM: Well, yeah.

Steve: It's like, yes, this is a second factor. And so what that protects you from is your credentials being captured and somebody reusing them. If you need to also know, not only something you know, but something you have, meaning this other factor of authentication, obviously you're more secure. The problem is it's a secret. That is, the way this works is your Authenticator has a secret which it shares with Google. And that secret allows them both to calculate what six-digit code should be shown within this 30-second window, which changes thus every 30 seconds. So the problem is you can't safely, I mean, technically you could force other sites to use the same shared secret. Then you wouldn't need a separate Google Authenticator account for every single site that you authenticate with.

The problem is, once again, if they lose control of their database, then it's not just their secret Authenticator data that gets leaked, but any that you've shared it with. So you're back to the same password problem of not being able to use the same password across all sites, which is very convenient, but we know is critically unsafe. So...

TOM: But there's also the waiter problem; right? Even if they're hashed and salted and secured in their database, every employee that's involved in their security firm can access my password; right?

Steve: Yes.

TOM: It's there on their internal server.

Steve: Yes. Very good point. And, again, in a perfect world, it might be nice if authentication was out of band, meaning that, if you've got a bad guy who has somehow hacked into your connection - now, arguably, if you're in that much trouble, then maybe authentication is the least of your worries, if you've got somebody who's, like, actively able to see your connection. On the other hand, you'd still like them not to be able to get it.

So if you're typing your username and password in, you might have malware on your computer, or you might have something, you might have a corporate proxy which is using a certificate installed on your computer - we've talked about this often - to decrypt all of your traffic in order to run it through, they say, antispam and malware filters. But

we also know that they're actually looking at the contents and doing keyword searches on it for acceptability and content protection and so forth. So there again, your credentials are in the clear there. Well, it would be nice if your authentication didn't go through the same channel that your main web experience was going through.

And the other biggie, post-Snowden, and now that we know the extent of the NSA's involvement in our privacy, is third-party involvement. That's the other big problem. The little football that I held up before, this is authentication that goes to VeriSign. So this is not authentication of a secret that I have with PayPal or with eBay. This is actually - this has a serial number that I registered with PayPal, but this is a service that VeriSign provides. So VeriSign is a third party involved in my authentication process. Unfortunately, we now know that it's just not safe to have third-party involvement with our identity. Many of these authentication, these, like, next-generation authentication systems, involve so-called "federation," where they want to federate authentication, where you authenticate to the third party, and the third party authenticates to the website.

Well, that might have sounded good last year, but that doesn't fly anymore because we now know that no organization can withstand a national security letter sent by the government saying we want you to turn over the information you have about this user. So I would argue that third-party authentication is dead. What we need is two-party, between you and the website, authentication that does not rely on the services of a third party because we just - they're not trustworthy, unfortunately.

And lastly, if we're talking about something new, it's got to be low friction. I mean, first of all, it can't be kind of better. It's got to be a lot better. It's got to be free. So there's no, I mean, one of the reasons people have moved away from this VeriSign model and eBay and PayPal football is it is really expensive. The reason Google's not using it, for example, they did their own, is that VeriSign gets a fee for every single authentication. So, first of all, the authentication instance has to be free. The apps or whatever it is you use has to be free. I mean, this is not something you could charge for. Maybe 10 years ago, but those days are gone.

Also, it has to be not overly complex because it wants adoption. We want people to be able to easily create whatever this is, both on the user's side and on the server side. And we see time and time again that really complex protocols - we were talking about just the other day that, with regard to the IPSEC security in IPv6, how it's now believed that the NSA influenced the design by helping to make it so complex that cryptographers were no longer able to even understand it. They, like, looked and said, okay, we can't say this is even secure because we don't understand it. So also it has to be simple and easy, feasible for one person, not big teams, not organizations, just one guy who says I want to implement this, for them to implement.

And in order to be adopted, it can't be jealous of the existing system. It has to be something which is feasible to have side by side, running next to existing authentication, as an alternative for people who would prefer to authenticate that way. And over time, if it were to succeed, it might end up being like your first choice, where it's like, oh, well, if you can't authenticate that way, then you go back to old-school username and password. And of course you'd always need some ultimate backup authentication for times when for whatever reason you can't authenticate this way. So that's my laundry list of ideal online authentication.

TOM: Now, what's crazy about this to me is, if I were to sit down today and go, okay, what should I fix about authentication, I wouldn't choose no keyboard interaction. I would immediately say, well, that's just silly. Of course we're going to need keyboard

interaction; right? Or sharing secrets with websites. Like maybe we could do without shared secrets, but I'm not sure how. You've gone through very meticulously and said, what would the perfect system involve here? That's why - some people are getting impatient in the chatroom. They're like, just tell us how it works. But it's important to go through and say all of these things need to happen because what's so impressive about SQRL is it addresses every one of these points that you talked about.

Steve: Yes. That's - exactly. What I have is that. What I have is what I just described. It is no linkage among websites, anonymous, no keyboard interaction, no shared secrets, out-of-band authentication, no third-party involvement, low friction, easy. I mean, and the other thing is easy for the user to user. Maybe that's more important than anything else, is what is it that we - that annoys us so much is, I mean, I can't log on anymore without LastPass somewhere because it knows all of my different passwords, because we've all been driven all the way there, to the point where really it's just become that burdensome in order to be secure.

TOM: And LastPass, which doesn't solve most of these problems that you've mentioned, is too complicated for a lot of people.

Steve: Right.

TOM: Sadly, yeah.

Steve: Okay. So what is this?

TOM: How do we do this?

Steve: What is the user experience? What does a user do to get all of this with the what I call "SQRL login," "squirrel login"? They are anywhere - at home, at Starbucks, at a public kiosk, it really doesn't matter. And the login page presents them with the regular, probably, username and login because, I mean, that's going to be - that's probably never going to go away completely. But off to the side, next to it, is a QR code, one of those cute little square digital codes which we actually did a whole podcast on some time ago. I went through the exact, everything about the structure and function and operation of so-called QR Codes [SN-382]. And underneath it, it probably says SQRL, although - I don't know what else it could be, but just to label it for people, stands for Secure QR Login. So SQRL, Secure QR Login.

TOM: You don't have to be nuts to use SQRL.

Steve: [Laughing]

TOM: I don't know, I'm just testing out...

Steve: You don't have to be - I like that.

TOM: You'd have to be nuts NOT to use SQRL. There you go.

Steve: So all you do is you scan that QR code with your smartphone, and you're logged in.

TOM: Wait, that's it?

Steve: That's it.

TOM: And I know, because I read the documentation. But I still can't believe that that, I mean, there's a lot going on behind the scenes, but really that's all you do.

Steve: That's all you do.

TOM: You don't have to press a button or take a picture of your thumb or scan your eyeball, nothing.

Steve: No. Okay. So what does it do? What's going on? So even the crypto system - and remember that - our listeners will remember when I was first telling Leo that I thought I had come up with something, I said, I'm tempted to call this HIPS, H-I-P-S, as an acronym which stands for Hiding In Plain Sight. Because, I mean, I almost think that it was like, okay, why hasn't anyone done this? And our listeners are shortly going to have the same sense. It's like, okay, wait a minute, what's wrong with this? Why does this work?

So here's what is going on. Every time a site displays a login page to anyone, a QR code is generated which contains the URL of the site's SQRL authentication service. So maybe it's like sqrl.amazon.com or Amazon.com/sqrl?, I mean, whatever. It's the URL which your smartphone will use. So this is the website saying we want to receive SQRL logins at this URL. And then on the end of the URL, so it's just that, it's just the URL, the authentication service, and then a parameter in the URL tail, probably "?," and then gobbledygook, what we now know as pseudorandom junk. In crypto terms it's a nonce, n-o-n-c-e. It's a number used once. And so that's the end.

So every time a page gets displayed, the server uses its random number generator, creates a new nonce, which it just offers on the page. People who don't have SQRL ignore it. They figure out what their username and password is for this site, and maybe they can't, so they use LastPass or 1Pass or MyPass or YourPass or whatever, somebody's pass, and log in that way. But if you're SQRL-enabled, you just let your phone see that. Now, it might be that the URL would actually be sqrl://. Instead of https://, maybe sqrl, or maybe sqrls (for secure) //, although the connections to the server would be secure. Who knows. So maybe your phone automatically recognizes a QR code with sqrl:// and just does it. Apparently there are, you know, smartphones have the ability to have a URL registered that way. Or maybe you just tap your SQRL app which you've installed and let it see the code. What happens?

So the SQRL app sees this URL. It takes the domain in the URL and cryptographically hashes it with your master identity key. There's something called the identity master key we'll talk about more. But it's a 512-bit large, pseudorandom key that is, like, that's your identity. It is universal. You could have it your entire life. It never needs to change. You obviously want to protect it, and we'll talk about all of that in a second. But so the domain name cryptographically mixes, and we actually use a an HMAC function in order to do that, a Hashed Message Authentic Code function, which is secure, to combine the domain name with this master identity key. That produces a 512-bit private key in terms of public - an asymmetric key pair, private and public. From that private key, the matching public key is synthesized; and the entire URL, which is the whole domain name and the nonce, this one-time pseudorandom thing, is cryptographically signed by the private key.

So, and we've talked lots about cryptographic signatures. Basically a hash is made, and then that's encrypted under the private key. So we basically do a cryptographic signature using the private key of the entire URL contained in the QR code on the login page. The phone sends two pieces of information to that URL. That is, it generates a standard web

query, HTTPS web query, to the URL given; and it provides the public key, which was synthesized from the private key, and a signature. And that's it.

TOM: That's it.

Steve: Now, that's the key. And, yes, that's it. That's all there is to it. The server receives this query coming in which will be for a login page that it's displayed, but hasn't heard anything back yet from the user. So in comes a query. The query contains a public key and a signature. It uses the public key to verify the signature, which is how signatures work. That tells it that whoever it was who sent this little packet has the private key that matches because only the private key can sign that URL, which is unique in every instance. And so that proves the identity of the user. The identity of the user is just their public key. That's their identity. That's the token by which the website knows who's logging in. So the public key forms their identity token. The fact that the public key verifies the signature authenticates that they are the person who actually has that identity. And so this makes it proof against replay attacks and various spoofing attacks because it is a URL that was just generated that the website verifies.

And note that because the way the private key was generated was from hashing the user's master identity against the website, there's a different private key per site, and therefore a different public key matching per site. So the whole system is site-specific. Every time that user with that identity master key comes back to that site and scans the QR code with their phone, same hash is synthesized, same private key is made, same public key is made, new signature of the new URL presented this time on the login is signed. And the phone makes a query back to the server to say, hi, log me in. And that's it.

TOM: So if somebody were to spoof that page, they wouldn't have all the information they need to create a QR code that would actually fool your phone.

Steve: Okay. So there's the basis, sort of the underlying architecture, is that we have a - we synthesize a per-site private key from the domain name in the URL, which doesn't change for authentication on that web server, and the user's master identity key to create a private key. The private key never leaves. It's made on the fly, basically from the domain name and the user's identity, never leaves the phone. It signs the entire URL and is also used to synthesize the public key. And those two things, the public key and the signature, are sent back. So that's it.

Now, there are still some things we need. But as far as we know, that architecture is secure. There is some fancy crypto which actually I explain on the SQRL pages at GRC, which are now up, by the way, if you go - under GRC's main menu, under Research > Recent, the top item under Research and then Recent, is the introduction page to this SQRL system. And that is we need to prevent the phone itself from being abused. Now what we've done is we have securely bound the user's identity and authentication to their smartphone. Now we need to keep it from being hacked. We need to keep it from being, I mean, their authentication from being abused.

And so for that we need good, strong, local authentication. That is, that they need somehow to prove that they're the person holding the phone. Mom and Dad don't want Junior to go logging into their banking site and getting up to mischief behind their backs, or doing anything else. I mean, in general, people don't want anyone else to be able to authenticate as themselves by using their phone.

TOM: Yeah. When someone takes my phone, steals my phone, all that. All those questions.

Steve: Yeah. And even, like, I mean, I see people handing other people their phone. Oh, you need to make a phone call? Fine. Or people are generally sort of casual with the security of their phone. Now there's something we really need to protect. And unfortunately, today phones don't do that. I mean, we even saw the much-heralded Touch ID lasted less than a week. It was a couple days before the Chaos Computer Club in Germany had come up with a bypass, arguably in all perfect conditions. But still, the phone isn't protecting you from that kind of abuse. And if you loaned your phone to someone, someone who needed to make a phone call, or let me check a web page or something, well, you would unlock it for them, and then they're holding your phone. So it is...

TOM: Right. And the phone thinks they're you, yeah.

Steve: Right. It is still the case that the best means for an individual to authenticate themselves is a password. It is something only they know. It is a secret.

TOM: So we still need one password.

Steve: Right.

TOM: A single one.

Steve: And we need it, not because the system needs it to be secure. The crypto appears to be bulletproof. We need it just to control access, access control to the system. Now, I talked about no third-party involvement, which I explained as an essential necessity in this day and age. Note that, however, what we've just described, what the SQRL system is, is a one-to-one authentication. No third-party involvement. You're simply saying to a website, hi, this is my token. And notice that this solves the problem of identifying yourself anonymously to some blogging site. You go to a blogging site, and you say, hey, I want to post a comment. Up comes the login to the site and a SQRL code next to it. You say, oh, snap that with your phone, and the site says, oh, okay, fine. Welcome back. They never know who you are, but it now has a fixed identity token to identify you. You can then, maybe you want to assign yourself a handle by which you are known for conversations there. That you can do.

The point is no one can ever impersonate you, and you can come back three years later, and the site will say, oh, welcome back, just using this and nothing more. But the fact that there's no third-party means the individual user is responsible for managing their identity. If there were a third party involved, if there were some big brother-ish organization, and, for example, you lost your phone, and you had SQRL installed there, well, you would want it disabled. So you could contact this third party and say, oh, my goodness, my phone's lost, deactivate my authentication that's associated with the phone, and they could do that for you. There is no "do that for you" here in this scenario.

TOM: Right, that's the flipside here; right? We don't want anyone else involved.

Steve: Right.

TOM: So how do we turn that off?

Steve: So, okay. So a couple ways. So the system provides very strong local password protection. There's the obvious user interface protection, meaning you try five times, and the user interface says, you know, you're not seeming like yourself today, or at all. So we can easily have user interface password lockout, which just after five tries it says,

sorry, you're going to have to reestablish your identity some other way. And the system provides for that.

The other type of attack is where something - malware or a hacker or the government or someone - somehow manages to get access to your phone's memory in an unencrypted state. We know, for example, that iPhones have long encrypted their memory under their user's password. You need to use that. You need to unlock the phone in order to decrypt the memory on the fly and so forth. But while apps are running, while the phone is unlocked, presumably that's available.

So we need protection underneath the user interface where, if someone got a hold of the meat of the application, the application's secrets, whatever those are, without the user interface able to block access, we still need the system to be secure against that. So that's one aspect of password use. And I'll come back to how we, in detail, protect that in a second because there's a second aspect that is related. And that is the other thing we want. Essentially, we have a smartphone with its 512-bit pseudorandom key that was made once and is potentially our online identity for the rest of our life. If we manage it correctly, it's huge enough and unique and secure enough, it never needs to change. Well, that means...

TOM: Now, is that a private or a public key?

Steve: Well, that is the secret key which is mixed with the domain name to produce site-specific private keys. So it is not a key that ever goes any further than that secure hash function. But the point is we don't want to lose it. No matter what happens. I mean, if the phone went through a trash compactor, or just the phone died, and completely dead, we absolutely - we need our identity. So that says we need to be able to export that key, that super-secret lifelong master key, in some safe fashion. Well, and say you're Leo, and you've got 27 phones. And you'd like to be able - and you're now using SQRL to authenticate yourself to all the sites you use, and you've got phone du jour that arrived during the podcast. So you need to be able to also transport this key between devices and between it and a safety deposit box, essentially. I mean, you want it somehow to be stored, storable, securely.

So the system provides the ability to export this super-secret master key as a QR code. So the user can say, I want to display on the screen my identity master key. Once it's up, then another phone can simply snap a picture of it in order to import your super-secret master identity into it. However, in order to verify the password that goes along with that, we need to provide in the super-secret master identity QR version of your key some password authentication information. And if we put password authentication information in there, then that exposes it itself to attack, meaning that, if some bad guy got a hold of it, then there is an encrypted, your encrypted master identity key, plus enough information to verify your password because, if you a year from now import that key into a new phone, you've got to provide the password. And if you transport that key, clone it to another phone, you've got to provide the password.

Well, that means that that little rectangular QR code has to be able to verify your password. And if it can verify your password, then that means it's potentially subject to a brute-force attack, meaning that there's all the information there it contains to tell an attacker all of the passwords they're guessing are wrong and to finally, ultimately, find the right one.

So how do we prevent that? We prevent that by making it ridiculously hard to check a password. We have technology now, we've talked about the Scrypt password-based key derivation function, PBKDF, Scrypt. Scrypt was designed by the guy who did the Tarsnap

system to be a memory-hard, password-based key derivation function. And I think we may have done a whole podcast on this [SN-388] because I remember talking about the way this generates a huge array of pseudorandom data, and then where every item in the array is modulus the size of the array, so that it is essentially a pointer back into another item in the array.

And so the idea is that you iterate through this, jumping through this array; and, at each location, that location tells you where to go next. So you go there in the array, and that location tells you where to go next. The point is, by doing this, there is no way to do this in the amount of memory that a field-programmable gate array or a GPU has. You could say we want a megabyte. And while there may be a megabyte in the system, there isn't a megabyte in the GPU's caching size memory or in a field-programmable gate array cell region. So this prevents acceleration by GPUs and FPGA technology.

So we use Scrypt technology to create a 60-second processing burden for the production of an exported QR code. If you tell your phone, I want you to show me my super-secret master key for whatever reason, because you want to email it, because you want to print it to make a paper copy, which all users would be advised to do, to stick in a safety deposit or to stick in somewhere safe, it deliberately is calibrated to take 60 seconds. It'll show you a countdown actually as a percentage from 100 because different phones will have different amounts of processing power. So it won't always be 60 seconds, for example, when you're importing that into a different phone, or a later phone. But it'll sit there for one minute, basically deeply - essentially doing the equivalent of deeply encrypting your super-secret, lifetime-long master key for an entire minute of your phone's maximum processing capability, and then it will show it to you.

What that means is that what it's showing you requires the same amount of processing in order to be decrypted. For a single guess. So once you export this, or, for example, say that you snapped it because you're Leo and you want to clone it to a second phone, you then have to put in the same password you had on the source phone. You enter that into the destination phone and wait. It's going to go through - it's got to perform the same operation, as burdensome, for a single attempt at the password. So if you use a strong password, it is impossible to guess this thing, to do a brute-force attack where you're talking a processing burden of a minute per single guess. I mean, you really don't want to even mistype your own password because you're going to have to wait another minute.

But my point is that exporting and importing these codes is done so infrequently, and we want so much security around them, that it makes sense to do a state-of-the-art, GPU-FPGA busting, memory-hard process so that it takes a minute. And what that means is obviously you want to keep this a secret. We're not saying you want to tattoo this on yourself.

TOM: No, please don't.

Steve: You want to keep this a secret. You want to print it out and stick it at the bottom of some drawer somewhere. But the point is, the way the system works, any and every exportation of that information is encrypted that deeply. It is 60 seconds of serious processing for a single attempt at seeing whether the password you used is correct. So, like, in the worst case, if it got loose, well, you're probably okay because you just can't perform - we talked about hundreds of millions of hashes per second. Well, now we're talking about one guess per minute. And so, yes, you still want to use a good password. You'd like it to be, if you'll pardon the choice of terms, squirrely. You'd like squirrely password. And I'm going to propose that any keyboard that displays the password prompt for defining a password doesn't make you use shifts in order to get to funny

characters, but lays them out there so that you're encouraged to use a password that's got all kinds of upper and lower and special characters and so forth so that it just won't be on anybody's password list.

TOM: Oh, that'd be a great Android feature, when you could put in a custom keyboard like that for the app that handles this. That's a great idea.

Steve: Right. Okay. So obviously you don't want a 60-second delay every time you use this on your phone. And so the idea is that whole 60-second process is only for performing the super-deep encryption of your master identity key when it's going to be displayed for whatever reason - to be put up on the display, to be copied to somebody else's phone, or to be printed out for long-term archiving and safekeeping.

TOM: How often do you think that that would need to be done? Not very often, it doesn't seem like.

Steve: No, I think like five times. Yeah, I mean, like almost never. So, and if people wanted even more, I've just made up a minute because that seems like a ridiculously slow rate at which any password can be guessed. And there's no way known to do this in parallel where you need a megabyte of memory statically available to this thing while it's running, so extremely acceleration-resistant. On the UI side, when SQRL says - when, for example, you want access to the application's settings, or when you want to authenticate, you need to give it that password. And there I propose that you're asked to wait for a second. It's 1,001, 1,002. That is still a substantial burden for any sort of local attack on a strong password, yet it's also short enough that it's not going to frustrate you.

And the last part of this that I haven't mentioned is that you do need to verify that the SQRL code domain that you're logging into, that is, the domain in the SQRL code, is the one you're logging into. So the other step here is you would see a page that wants your identity. And so you snap it with your phone. Your phone needs to show you the domain, the domain name, www.Amazon.com or Facebook.com, whatever, that you're going to authenticate to and get your permission because the one attack which some smart guys that I shared this with over the weekend came up with, I call it the "evil domain attack." You could be logging into an evil website, and behind your back it could go and get a login page, for example, for Facebook. And it shows you the Facebook SQRL code that it got from Facebook.

TOM: It's just taking Facebook and redisplaying it, so yeah, yeah.

Steve: Exactly. Basically it's a type of phishing attack. And so you don't want to blindly authenticate without verifying that you're authenticating the domain you think you are. And so the message, it would just come up and say you are about to provide your credentials for this domain. And then it would be, not in a little dialogue, but my intention is as large a print as will fit on the screen, it shows you. And so you just get in the habit of making sure that basically the credentials you're about to provide are for the site you expect because, if you didn't do that, if you were at evilwebsite.com, and it popped up, and it was showing you the SQRL code for Facebook.com, you would be sending your credentials to Facebook, but basically you'd be authenticating the web session that Evil Website had started. You'd be authenticating that login that Evil Website was doing behind your back. And so...

TOM: Basically riding in on your back, so to speak; right?

Steve: Yes. You would give it access to your - you'd basically be giving it, be allowing it to impersonate you to Facebook. So we can completely prevent that, just by making sure that we're providing the credentials to the site we think we are. But notice even in that case it doesn't get anything about us. It doesn't know who we are, doesn't get our ID, doesn't get - even the signatures, that all went to Facebook. All that happened is that it was spontaneously logged in. So, I mean, it's not a good thing, but it's still a limited impersonation for that session, which we can completely block just by saying these are the site's credentials you're about to share. And if you realize, wait a minute, I'm not at Facebook, I'm at Evil Website, it's like, no. This is a squirrely SQRL code.

TOM: Why does evilwebsite.com want my SQRL? Stay away from my SQRL, yeah.

Steve: Exactly. And that's the whole system. I mean, that's it.

TOM: We've got lots of questions from people. And I think you've touched on most of them. One of the ones that I'm not sure we covered directly was what if I want to have two people using the same device? I've got two people that use my phone. Is that possible?

Steve: Yes, yes. So there are a couple variations. First of all, I would propose that an application, an SQRL app, would have the notion of a user. And so your phone might just have you as the user. But there's absolutely nothing to prevent, for example, say for convenience, you and your wife, or your spouse, to be neutral, you and your spouse each have a smartphone. You each create your own identity in your own SQRL apps on your smartphone. And then you want to install that identity in each other's phone. So the idea is you create an additional "user," and we'll use that term carefully, a user, so that now the SQRL app has, like, the current user. It's either me or my spouse. And so there's nothing to prevent you from creating - from having the app essentially have a separate key and password information for each user. So that you could do. And, yes, so that works nicely. And you could freely delete them if you're no longer using them.

Your kids might want to install their identity in your phone, again for convenience sake. You would have no access to their identity because it would have the same level of protection as your own identity has against other people.

TOM: Now, the other question that of people had was what if I want to surf? What if I want to log in on the phone that I have the SQRL app?

Steve: Ah, yes.

TOM: How do I do that?

Steve: A couple ways. First of all, in the worst case, it's normally possible for you to copy an image on a web page. So you hold your finger down on the SQRL code, and up comes a little dialogue saying, "Copy"? And so you say yes, copy. Then you just simply go to the SQRL app and paste it into the SQRL app, and you're good to go. But the other cool possibility is, for mobile users, is that we could also do the same thing with the actual URL. Instead of showing the graphical SQRL code, it could just, if the site knows, if it sees from your user-agent that you're using a mobile device, and it's obvious that sites are becoming aware of mobile devices because I notice sites are recognizing that I'm on an iPad when I am, then it could still provide the SQRL code, and would, but might also just have a link, that is, there would be a link, sqrl://.

And so the point is it would be a little button underneath it saying "Logon Mobile." And so

you simply click the button. And so that executes the device that's registered for sqrl:-flavor URLs, which essentially fires up your SQRL logon, presents you with a screen saying is this the site you want to authenticate to, and you say yup, and off you go.

TOM: And back you are.

Steve: And then back you are, essentially where you were, but now securely authenticated.

TOM: In fact, QR code is really just a string of data represented as a QR code. So there's no reason you can't represent it as a string of ASCII characters; right?

Steve: Exactly. In fact, the QR code that we were talking about for your master key, that's all binary on a predefined format. But, yes, the QR code on a web page is just text. That's an existing standard. If you go to a website, or if you google "QR code generator," you'll get pages of QR code generators. There's a bunch of nice ones. And you can put in https://www.amazon.com/sqrl?, and then, like, a bunch of nonce sort of stuff, and see what a QR code looks like. It's just a standard URL. But the beauty of it being a QR code is that it essentially jumps off the page into your phone, optically, just that easily. Which was part of what hit me that morning during breakfast, was wait a minute, this seems like it could work.

And so imagine the experience. I mean, now, I mean, this is - I want one of this, one of these things. It's just it's my phone is able to assert my identity in a secure fashion to every site I visit. And what I would do is, over time, as sites began to support this - and, again, it's low friction. They leave their existing logon there. They simply add the QR code over time. Users add SQRL apps to their smartphone and begin using it. And so, for example, say the first time I went to Amazon, I might snap the QR code, which I notice has finally appeared on Amazon, and Amazon would take me to a page saying, hi there. We don't know who you are yet. If you are an existing Amazon user, please provide your traditional logon so we can associate your SQRL identity with your Amazon account. And so I do that one last time, and from now on, wherever I am, Amazon knows me.

TOM: Now, one of the reasons - and we keep saying "phone," but really we're just talking about a device with a camera and an Internet connection, essentially; right?

Steve: Yes, correct.

TOM: Because an iPod Touch would be able to do this if it's connected to WiFi.

Steve: An iPad.

TOM: Yeah, an iPad could do this. And I feel like one of the reasons you're saying have it on that device is that way you only have the one device that can authenticate you. And so you don't store it on a bunch of other devices unless you really want to. But you're sort of - you're distributing your risk; right?

Steve: Leo would have it on all of his phones. And it would be safe because it's deeply encrypted and protected. And so there really is, I mean, for convenience, I would think you'd want to have it on whatever you have with you. So, yeah, having it on one phone is good; but having it on all of your mobile devices, that could work, too.

TOM: Well, then a few people were saying, and curmudgeons, admittedly, probably a minority, but saying "I don't have a smartphone," "I don't want a phone with a camera,"

or even "My workplace won't allow me to have a phone with a camera," which still happens sometimes. And they're suggesting, can I have this app on my laptop and use it the way you're saying you could use it on the device itself? So the laptop is your device.

Steve: Yeah, I don't see any reason why not. This could be done as a plugin so that the plugin sees the SQRL code and performs the handshake for you and identifies you.

TOM: So it really has so many more uses, then, suddenly; right?

Steve: And actually the plugin has an advantage, too, that it could, for you, verify that the domain of the SQRL code matches the domain of the URL of the page. So it could eliminate that extra step that we have in the phone to verify that the QR code which we're not able to read ourselves, it looks like gibberish, so the phone decrypts that so we can see the URL to where we're going to send our authentication information. A browser plugin could do that for you.

TOM: That's incredibly useful. Now, your laptop doesn't even have to have a camera at that point, or at least the one that you're going to trust. Of course, don't lose that laptop, or you're going to have to go get your code out of the safety deposit box and go through that. I'm sure that you still want people to poke holes in this if you can. I know you've done a lot of bullet testing on it. What do you want people to do with this next?

Steve: Well, okay. So I just had to tell the world. We have a lot of listeners all over the place. Everybody knows about it now. There are pages up where I've got block diagrams and careful descriptions. I think I've got three and, like, a quarter of maybe 10 pages finished. So the core pages - basically, Tom, what I showed you yesterday, so that you would know what this podcast was about, that's all done. Anyone who reads those three pages will completely get it. I'm working on the attacks page, where I want to document all the different attacks that we know of, like brute-force password attacks and, like, DNS spoofing attacks, is there any vulnerability there and so forth. I also want an implementation page.

I mean, as far as I'm concerned, I'd love to get back to SpinRite, frankly. I've done my job here. But if there's anything I can do by being some focus for this, I'm certainly willing to do it. If people like the idea, then we'd like to have SQRL apps created. But we'd also want to guarantee interoperability. So that means that we need to define the protocol by which the SQRL app contacts the website. So we need interoperability, essentially standards. We need some standards to be created around the concept so we don't end up being fragmented because that would kill it quickly.

And I did just create a new newsgroup at GRC. I do it very infrequently; it's been years. But there is now grc.sqrl is the newsgroup on GRC's NNTP newsgroup server. I haven't looked there, but I'll bet it's already active, knowing the people that are hanging out there. So that's really going to be our central focus. I do have a feedback page among those SQRL pages on GRC. Essentially, if you go under Research and then Recent, and then you'll see SQRL Secure QR Login. That'll take you to the introduction page. At the bottom is a block of links. And I do have a feedback page, as I always do for these things, where anybody can essentially send feedback directly to me. I'm interested in what anyone has to say. But for extensive dialogue and conversation it would be much better to participate in GRC's newsgroups, where we've got a bunch of really great people working on this stuff.

TOM: And that's just grc.sqrl, right, for the newsgroup?

Steve: Yes, is the newsgroup, grc.sqrl. And I actually have a link to that on the feedback page because we have a web browser interface, which is read-only, to the newsgroup so you could just see what's going on easily. And we're old-school NNTP news server.

TOM: Yeah.

Steve: We don't have a web forum. So you need to use something like Thunderbird, or I'm still using Gravity on Windows, which is my favorite.

TOM: Nice.

Steve: There's one called NewsTap for iOS that I like and I use on my iPad. That works great. And then, yeah, we have a really, really great group of people who are working. And of course I'm sure that I'll talk about this next week with Leo. We'll see what's happened in a week. I'm going to try to get these pages finished. And then we'll sort of see how it goes. But that's what this is. So far, after a bunch of crypto-savvy guys have looked at this over the weekend, we made some improvements and changes to my original concept, better understood the nature of attacks, and it's still standing. And I wish this existed. This is how I would log in and use my authentication across the Internet. It's just got a huge number of benefits.

TOM: Absolutely. I mean, it basically makes what I do now a whole lot simpler. So if I can have an extension on the laptop that I use and trust, right, that's better than LastPass.

Steve: Yes.

TOM: It's the same as LastPass. It's an extension. But it has so much more to it. And then I've got an app on my phone for when I'm on a computer that I don't trust. And that's the same as having that Google Authenticator app that I have except, again, it does so much more and makes it so much easier. I'm really pleased with this. The other reason that I'm pleased with this and excited about it, Steve, is the chatroom, as they always do, is going to take potshots at this all the way through. And between me and Sparkyman was in there answering questions as well...

Steve: Oh, good.

TOM: ...I don't think there was a single objection that was brought up that we didn't have, yes, he's thought of that; yes, he's getting to that; it's at this part in the documentation.

Steve: Yeah. Yeah, I know. I think it works. I think people are going to be going, oh, my goodness, I mean, this works.

TOM: The only surprise I had was two people who said, "I had a very similar idea that I pitched to X company, and they said, sounds great, but it's not in our strategy right now."

Steve: Yeah, well, see, and that's just it. This is not something someone can own. One of the items I have on the first page down low, down toward the end, is did I invent something? And I answer my own question, and I say, I don't care. I mean, arguably, I'm sure there are people who say, oh, you know, you need to get intellectual property protection on this. I have formally and officially said, here. This is an idea. It was a good cup of coffee that I had that morning.

TOM: I want that cup of coffee.

Steve: You know, let's see if the world wants to run with it.

TOM: Yeah, absolutely. No, I think this is great. And like you say, even if similar ideas have been floated before, they don't exist. And what you're pushing for is let's make this happen, let's make this exist. Here's an architecture. And you've bullet tested it. That doesn't mean it's done, but...

Steve: It's why it being so low friction matters.

TOM: Yes.

Steve: When I say "low friction," I mean that it is so much better from the user's experience that there will be push for it. It is so simple architecturally that a single guy who writes mobile web apps or mobile apps can implement it. I didn't talk about the crypto, but all the crypto is there. I use Dan Bernstein's, it's called Curve25519. It's an elliptic curve crypto. Notice that one of the requirements of this is that you be able to have a pseudorandom-generated private key. That's different than RSA, where you use a pair of primes, and you use the product of a pair of primes. There's no way to deterministically arrive at those from the hash function.

So one of the enabling features that I didn't talk about specifically, it's all documented already on those three pages, is that the output of that HMAC secure hash function is this 512-bit pseudorandom number. That's your private key. So we need cryptography which is able to accept any pseudorandom number as a private key and compute the public key and then sign something under the private key, and all of that's been worked out, too. There's existing code, all public domain in source, for every single piece of this. So basically all of this is already open source, public domain technology. We've just got to glue it together.

TOM: Well, this is fantastic. And folks should go check this out at GRC.com/sqrl, if you're interested, as you put up all the pages.

Steve: Actually two SQRLs, /sqrl/sqrl.htm. I wanted to give it its own directory to live in.

TOM: Gotcha. So sqrl/sqrl.htm. You're right. Of course you're right. But I'm just like, I just tried it and got a "page not found" the way I did it. So, yes, GRC.com/sqrl/sqrl.htm, no "l," and take a look at it. Go to that feedback page if you want information on how you can further investigate this, implement this. There's certainly lots of other stuff going on with QR code. Somebody just put a link to an old Verge article from this summer about Google messing around with QR codes for security. As Steve said, this is not - that's not the point. The point of this is all of this stuff, as Steve just mentioned, is already available, free and open source. It's just putting it together and making it work. And hopefully some patent squatter doesn't try to come along and claim they invented it. But that's always a risk with anything you do on the Internet.

Steve: I did look at what Google had done, because of course when I came up with this I thought, wait a minute, how can nobody have thought of this before?

TOM: Right, right, uh-huh.

Steve: And so I spent a couple days really looking hard. What Google did was they had

an interesting idea, not this one at all. Their idea was they would provide a QR code. You would then snap that with your phone, and the login would jump off, essentially, off the page onto your phone. So it was a way of, like, transporting the standard login over to your phone, and then you'd do the same thing you normally did. It's like, okay, I'm not sure why that's better. And of course it wasn't, and it died.

TOM: That's just borrowing the system they use for Chromecast to send video links, which don't necessarily need to be secure, back and forth. And just sending the login URL, that's different. GRC.com also for SpinRite, for all the other freeware and services and research and everything that you do. I mean, this is the thing that you've been up to. Is there anything else to mention before we take off?

Steve: No, I'm sure I'll be focused on this until I get back to working on the next version of SpinRite. And I should mention that, since SQRL is not an often-occurring string, I would bet that a week from now you just put it into Google, and you'll probably be able to find GRC's pages.

TOM: Okay, that's a good point.

Steve: So I think it'll take you right there.

TOM: Excellent. Thank you, Steve, as always, for the show and for the great information, and for doing this. I think this is fantastic.

Steve: Well, let's hope it happens. I want it for myself. I think it would be a great step forward.

TOM: TWiT.tv/sn if you want to get the show notes and all of that stuff, as well; and GRC.com for all the great work Steve does. Thanks, everybody. And Steve, once again, thanks for allowing me to fill in for Leo. It's been great fun these past three weeks. Really cool.

Steve: Has been. And it worked so well that I'm sure Leo will feel free to take a vacation again.

TOM: I'm sure he will. Thanks, everybody. We'll see you later.