KRACKing WiFi

Description: This week we examine ROCA's easily factorable public keys, the surprising prevalence of web-based cryptocurrency mining, some interesting work in iOS password dialog spoofing, Google's Advanced Protection Program, and some good loopback comments from our listeners; then we take a close look at KRACK, the Key Reinstallation AttaCK against ALL unpatched WiFi systems.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-633.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-633-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. This is, boy, this is a banner week for security flaws. Here we are Tuesday, and already there are two major flaws: ROCA, which affects public key crypto in a very serious way; and, of course, you've probably all heard about the WPA2 KRACK - K-R-A-C-K. Well, Steve's got reassuring news in both cases, and information on what you need to do to protect yourself, coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 633, recorded Tuesday, October 17th, 2017: KRACKing WiFi.

It's time for Security Now!, the show where we talk about your security and privacy. And, man, everybody is just dying to hear what Steve has to say this week. Hello, Steve Gibson.

Steve Gibson: Thank god this did not happen on Wednesday.

Leo: Yeah. We would have had to do a special redo.

Steve: Or later today, yes. I mean, it's wonderful when these things drop on Monday because then everyone's all revved up, and there's enough information for me to talk about.

Leo: Yeah, if it had happened Tuesday that would have been bad, too, really. Monday's the best day.

Steve: Yeah. Please, if we could have the major security problems hit on Monday. There was even a little bit of windup Sunday evening. Sort of the news of something about to happen the next day began to leak out on Sunday. And it's like, okay, this doesn't sound good.

Leo: Unh-unh.

Steve: So consequently, Security Now! Episode 633 for October 17th, 2017: KRACKing with a K, of course, WiFi, which will be our main topic. But there was another problem, also yesterday. Both of them happened. And the other one we're going to talk about first; and we'll deal with the KRACK crack, or the KRACK attack, or the release of the KRACKen or whatever you want to say, at the end of the podcast. But ROCA (R-O-C-A) is, I would argue, much worse.

Leo: Interesting.

Steve: But hasn't gotten nearly as much attention because it's not as sexy.

Leo: Right.

Steve: And doesn't have as good a name. You've got to give these things a good name, remember, like Heartbleed. Oh, my god. So we're going to talk about ROCA, which is a massive industry-wide failure of an embedded cryptographic library in five years of Infineon's chips, which are everywhere, which has until now been unknowingly producing weak public keys that can be factored, which is the whole thing you don't want in a public key. And they're everywhere, like BitLocker is not safe.

So this is, as I said, arguably a much bigger problem. The KRACK attack is not good, but we'll explain all about that. It definitely needs to be fixed, but it's not the end of the world. We don't need WPA3 or anything else. Also, I read an interesting piece by an adblocking company which started off talking about the surprising prevalence of web-based cryptocurrency mining, which we were talking about last week. And remember I said, I think it was during our closing-the-loop loopback with our comments from our listeners, that it's sort of annoying that your computer is being commandeered while you're visiting one of these websites, but it's not a security risk. They're just taking your processor time.

But these guys - and as I was reading this, I had a thought that hadn't occurred to me last week, and then they ended up presenting it. And I thought, oh, well, okay, good. So we'll talk about that. A really interesting, worrisome look at iOS dialog password spoofing. Also, Google has formally announced the Advanced Protection Program that we expected to come out and we talked about a couple weeks ago. We've got some comments from our listeners. Then we're going to take a deep dive into KRACK, K-R-A-C-K, which is an acronym for Key Reinstallation AttaCK - that's where they get the C-K from at the end of the attack - which is effective against all not-yet-patched systems, or those by Windows and iOS that never implemented it correctly in the first place and, as an interesting side effect, aren't subject to this main problem.

Leo: Isn't that funny. So that was what Rene was speculating is that maybe the reason that some of these devices aren't vulnerable is they don't do it the proper way. Which is crazy.

Steve: Correct. They screwed up. And it's like, oh. The attack doesn't work here.

Leo: Turns out it was a good idea.

Steve: But even so, Windows did quietly fix this officially in last Tuesday's Second Tuesday of the Month update. So they got it. And we know that Apple has betas in the works, and we expect to have probably 11.0.4, or maybe it'll be 11.1.

Leo: Yeah.

Steve: And of course we just hope they keep fixing iOS because, boy, it's still a catastrophe.

Leo: Yeah. It's 11.1, and I have 11.1. So everybody who has a beta, public or developer beta, should be up to date and secure. It's only devices. And then, you know what, eero's doing the same thing. I checked with our sponsor to see what they were going to do, and they're in beta. So I guess the mitigation isn't hard. But everybody reasonably wants to test it a little bit before they push it out to everybody.

Steve: Well, and what's interesting is that access points don't need to be updated. That's the other thing everybody has missed.

Leo: Oh.

Steve: So there's no need at all. It's the clients that are the problem.

Leo: Oh, okay.

Steve: Not the access points.

Leo: Oh.

Steve: Yes. But access points are still going to fix their protocol. So this is one nice way of measuring how you feel about your access point provider.

Leo: Yes, yes, yes.

Steve: I heard you mention, for example, that DD-WRT is already fixed. It is. And Ubuntu is, and a lot of them are. But I'll explain why this is the case. It's only if access points were connecting to each other because, in that case, one of them is a client of the other, and the attack is against the client, not against the access point.

Leo: I did not understand that. See, this is why we needed you. Fantastic.

Steve: Well, that's what I'm happy to provide. So I think we've got a great podcast, with lots of information.

Leo: That is real, boy, that's - okay. Man. As always, Steve's the best. And when things like this happen, invariably the first thing I hear is, okay, okay, Steve's going to talk about this; right? Yes, he is. And ROCA, too, which I'm happy to say all my keys are secure from ROCA because I checked them right away. Okay, Steve.

Steve: So we have a fun cartoon for the week that is about this problem. Unfortunately, it's incorrect, but it is fun nonetheless.

Leo: Oh, dear. Oh, dear.

Steve: So we've got two guys. One's got a cup of coffee in his hand. The other is eating a burger. They're in an office. And a third guy comes rushing in, swings the door open in a panic and says, "Oh, no. They found a vulnerability in WPA2. We can't trust any WiFi anymore. We need to change every router, starting with ours." And so the bearded guru who's sitting down with his burger says, "Calm down, guys. We'll just need to update the firmware, and everything will be fine." The guy who was initially panicked says, "Are you sure?" And the bearded guy says, "Yes. We just need to wait for the manufacturers to release a patch for all their router models, and then for all the sysadmins to flash all the new firmware." And then the fourth frame shows three skeletons.

Leo: They need spider webs.

Steve: Exactly. And in fact in the initial three there was a small flower in the plant. And in the final one there's, like, five flowers, and the vine has wandered off the desk because so much time has passed. So the point being, uh-huh, and that'll never happen. So the good news is, as we just said at the top, and I'll explain why, it doesn't matter.

So ROCA first. So both of these problems, the KRACK with the "K" and ROCA, are subjects of the upcoming ACM Conference on Computer and Communications Security known as the CCS, the ACM CCS, which is the major annual ACM Conference of the special interest group on security, which this year promises to be a barnburner. So not only will...

Leo: Only you would call a security conference a "barnburner." But I know what you mean. I know what you mean.

Steve: That's right. That's right. No barns were harmed during the production of this podcast. So not only will the presentation of the KRACK attack be there, which as I said we'll be talking about in detail at the end of the podcast; but so also will a, I would argue, even more unsettling discovery by a team of researchers in the Czech Republic, the U.K., and Italy from several universities. ROCA, R-O-C-A, as in almond, stands for Return of Coppersmith's Attack.

Leo: Wow.

Steve: Yes, R-O-C-A, Return of Coppersmith's Attack, a practical factorization of widely used RSA. So, okay. Coppersmith, Don Coppersmith is a well-known cryptographer with a long rsum of cryptographic accomplishments to his name. He developed one of the many improvements of factorization, of factoring, which of course is a concern everybody has because we rely on the intractability of performing a prime factorization of two very large primes after being multiplied. We're relying on the fact that it is computationally infeasible to demultiply them, to factor them, to break them back apart after the two large primes have been multiplied.

So it turns out that we're often talking about the importance of that problem, how crucial it is. And in fact the worry of quantum computing is that there will be a way to apply the power of quantum computing to this problem, which is why cryptographers and academics are already talking about a next generation of crypto that will be quantum proof because the worry is factorization may not longer be quantum proof, once quantum computers develop enough strength. So I'm sure in the future we'll be talking about lattice-based cryptography, which is already known to be quantum proof.

Okay. So one of the things that we also often talk about is the distinction between theory and practice. That is, it's important that you put into practice properly what the academics have figured out in theory because of course implementation is where the rubber hits the road, and that's what actually matters. What these guys uncovered was a doozy of a mistake, which exists in the embedded cryptographic library used by Infineon's products. Okay, now, I mean, Infineon is everywhere. They're one of the major embedded crypto people. Their stuff is in the Trusted Platform Modules, the TPM, especially v1.2, which is the one that's around that most of us have in all of our machines today.

Leo: Oh, well, that's a very big deal. I didn't realize that.

Steve: Yes. Yes. And BitLocker uses TPM for its key. And it turns out that, if you have BitLocker running on a machine with TPM 1.2, it almost certainly has Infineon in it, and the public key it would have generated can be factored.

Leo: Wow. That's really bad news.

Steve: It's really bad news. These guys scanned the Internet. They found thousands of public keys that can be factored. They looked through GitHub and found a ton of public keys on GitHub that were being used to protect things, but were not providing protection. So the problem is huge. This impacts one flavor of Yubico, and Yubico responded instantly. They knew about this a while ago. They've been safe, I think, since July.

The researchers discovered the problem in January of this year. January of this year; right? So 10 months ago. They informed Infineon in February and negotiated an eight-month silent period, knowing that they'd be presenting this at the ACM CCS meeting at the end of October. It's October 30th through the beginning of November, the first few days of November. So Infineon's had time. Word's gotten out quietly to those people who are being affected. And so there's been time for remediation. Yubico, if anyone is a Yubico customer and concerned, they've got - Yubico immediately put up a page to help identify which of their products might be affected. Most of them aren't. I think it's the YubiKey 4 when you use it to synthesize the public key.

Leo: That's what I thought. Okay, good. Because that is a feature of the YubiKey 4. You can generate private keys and keep your PGP key on there.

Steve: Correct.

Leo: And I never did that because I don't - for one thing, you can't put it anywhere else. It's on the key.

Steve: Right.

Leo: Okay, good. All right.

Steve: So it's only in the case that you do that.

Leo: Only if you use it to generate [crosstalk].

Steve: So it's a relatively small attack surface in that case. But they also found a ton of insecure PGP keys. So, I mean...

Leo: Really.

Steve: Oh, yeah. I mean, the problem, I mean, Infineon is just everywhere.

Leo: Yeah, see, I generate mine in software. So I'm okay.

Steve: Yes, and so you're okay, yes. So this got introduced in 2012, so for the last five years this has been a problem. Okay. So let's take a look at what this means. First of all,

I think, well, it's a mixed blessing. It's good news, unfortunately, for both people with keys and bad guys who want to attack them, that any public key can be almost instantly, like as in a millisecond, so effectively instantly, tested as a candidate for easy factorability. And there are online websites. There are online and offline tools. Everything is open source, MIT licensed so the code that has been produced can be put into other tools or products. And so there are a lot of resources to allow people to check their own public keys to see yay or nay whether that key would be vulnerable to this essentially factorability short-circuit.

So, okay. So to give us some context here, the worst cases for the factorization of 1024 or 2048-bit keys would be less than three CPU months for the 1024-bit, or 100 CPU years for the 2048-bit. Okay. So, first of all, that's interesting. And this is another important lesson for us is that doubling the bit length of RSA didn't double the difficulty. It went from, what was it, that would be 400 times harder, three CPU months, which is to say one quarter of a CPU year, compared to 100 CPU years. So 400 times harder to factor a 2048-bit key. And that's on a single-core recent CPU. So nothing GPU, nothing fancy, no hardware assistance, just a standard CPU, just to give us some sense for that.

But that's the worst case. The expected time is half that because you're doing a bunch of guessing, and you may just get lucky. So generally about half of that is what's expected. However, this factorization problem can be performed in parallel across multiple CPUs, allowing for practical factorization in hours or days. So, okay. And I should say this is the - if your key is weak, that is, if this test reveals that you have a weak key, this is how quick it can be done. It should be vastly harder.

A properly generated 2048-bit RSA public key should require several quadrillion years. So that's hundreds of thousands of times the age of the universe. We throw these large numbers around casually, but when you're at hundreds of thousands of times the age of the universe, you probably are safe, as long as nobody made a mistake. And the point was somebody did. The 2048-bit keys that this broken library has, which has been embedded into Infineon's hardware, can be broken in not several quadrillion years, but 100 simple CPU years.

Leo: Oh, that's a little different.

Steve: Yeah. Yeah, exactly. So to put some cost on this, in the case of the broken public key, the worst-case price of factorization on an Amazon AWS C4 computation instance for the 1024-bit key would be $76.

Leo: Wow.

Steve: Uh-huh. And the 2048-bit key costs more. That's 40,000. But obviously well within the price, well within the budget of a corporation, or certainly a state actor who's got essentially infinite money; and half that would be, again, typical. So about $20,000 worth of Amazon AWS C4 computation time, and you get to crack one of these weak keys.

Leo: So a scenario would be maybe you had a competitor's laptop, and it was an encrypted hard drive using TPM, and on that laptop were presumably secrets of

interest to you. You could, for 40 grand, unencrypt it, basically.

Steve: Yes.

Leo: Wow.

Steve: Yes, exactly, yes. So for a reasonable price in a short time, way down from several quadrillion years.

Leo: How much is that in Amazon compute - actually, really this may be more of a statement of how cheap it is to get compute time.

Steve: You know, come to think of it, it's free because you'd never have a chance to pay the bill.

Leo: The bill would never come.

Steve: It's like, no, we're still working.

Leo: I'll let my great-grandchildren pay that bill.

Steve: Right.

Leo: That's hysterical.

Steve: Okay. So they disclosed it, as I mentioned, to Infineon in February, so everybody's been scrambling.

Leo: Well, as an example, Yubico says every key we've sold since June is fixed.

Steve: Yes, exactly.

Leo: So they've known for a while.

Steve: Yes, exactly. So Microsoft, Google, HP, Lenovo, Fujitsu, they've all already released software updates. On the other hand, a laptop that doesn't get those isn't fixed. So very much as with the KRACK attack that we'll be wrapping up today with, this is an instance where you really do want to sort of ignore the advice of, which is typical for firmware, of if it's working fine, don't mess with it. In this case, actually it only appears

to be working fine. So you definitely want to make sure, especially if you have TPM v1.2, because that's where the problem is. Apparently, v1 didn't have this because it's older than this library, which only appeared in 2005. I mean, sorry, 2012.

Okay. So these guys, as I mentioned, scanned the 'Net, found thousands of keys, found PGP keys, found keys on GitHub. So apparently people have been using this Infineon library for producing keys, just figuring, oh, you know, hardware's better than software. Not in this case. They also looked at 41 laptop models that used TPM modules. They found vulnerable TPMs, Trusted Platform Modules, from Infineon in 10 of the 41.

And in their own disclosure they note that, they said: "The vulnerability is especially acute for TPM version 1.2 because the keys it uses to control Microsoft's BitLocker hard-disk encryption can be factored. So anyone who obtains a Windows machine with a BitLocker-encrypted drive on top of TPM 1.2 may not be secure." So what that says is you'll want to fix your hardware and then rekey your BitLocker using an updated key from the Trusted Platform Module. And I'm sure there will be remediation details available as soon as this all happens.

So both online and offline detection tools have been provided. They're open source and, as I mentioned, released under the MIT license so they can be put into other solutions. The best vulnerability test suite is keychest.net/roca, K-E-Y-C-H-E-S-T dot N-E-T /roca.

So the researchers wrote: "Our work highlights the dangers of keeping the design secret and the implementation closed-source." Oh, I forgot to mention that the way they found this was just by having some Infineon hardware generate a whole crapload of keys, of public keys, all presumably safe, and these guys studied the output and found the weakness. It would have been so much easier if, as academic crypto researchers, they had been able to look at the code. But it's closed. And so they had to go through the extra work of essentially looking at the output and reverse-engineering, discovering and reverse-engineering the problem from the result.

So they said: "Our work highlights the dangers of keeping the design secret and the implementation closed-source, even if both are thoroughly analyzed and certified by experts." They wrote: "The lack of public information causes a delay in the discovery of flaws and hinders the process of checking for them, thereby increasing the number of already deployed and affected devices at the time of detection." So of course Infineon's not happy. They have acted as responsibly as they can, but unfortunately this is firmware embedded into hardware, so getting to it is a little less easy than it is for OS updates and app updates and the sorts of things we're seeing all the time, and also probably requires more user interaction. And we'll be talking, of course, again, about how KRACK relates to this.

Yubico's page is Yubico.com/keycheck, and I've got a link in the show notes to the paper where they describe all this. But so basically there was, five years ago, a weak library or a flawed library which was supposed to be producing strong public keys, wasn't. And because it's from a well-known good manufacturer, I mean, I'm sure Infineon is not happy about this, these keys are well used and prevalent. So it is possible to check any, if you've got them, to quickly determine instantly whether it's vulnerable or not. And not all of them that are produced by this library are vulnerable. So it's worth checking, rather than immediately panicking. And if you're able to rekey once this is fixed, or just rekey from a software-based source, as you did, Leo, then that makes a lot of sense, too.

Leo: Yeah. I actually was looking into - because I thought, it's time. I read a really

good article saying don't keep your private keys on the Internet, which I don't do, or even on an Internet-accessible device. Like put them on a USB key and put it away. And so I was thinking, another place to put it would be of course a YubiKey. And then I noted that you could generate a PGP key on YubiKey, the YubiKey 4. It has enough horsepower to do that. But then you can't put it anywhere else. And if you lose that one, you're out of luck. So what I decided to do is, yes, generate a new key, put it on the YubiKey, but generate it in software using open GPG, and then put it on the YubiKey and put it on a separate USB key in a drawer somewhere so that the private key isn't publicly available. That's a little less convenient. But if it's on the YubiKey...

Steve: I think you mean so that the public key is not publicly available.

Leo: No, the public key has to be publicly available, or it wouldn't be very useful. The private key is the one you use to authenticate the public key. So that you want to keep very close to your chest. You put the public key out.

Steve: And so that's the problem is it's the public key that gets factored.

Leo: Oh, screw that. But I'm going to make it with safe software.

Steve: Correct.

Leo: Right, of course. Yes, the public key is the weak link.

Steve: Right.

Leo: Private key wouldn't be because nobody has it.

Steve: And that's why it's private.

Leo: Right, exactly, yeah. But, no, I want to keep my private key private, not because of ROCA. But you're right. But I checked, immediately went to that ROCA tester, which by the way is down right now, the one that you...

Steve: It's probably just submerged.

Leo: Yeah. They say "KeyChest is upgrading our system." Which means, yeah, it means it got clobbered. But when I heard about this yesterday I ran all my keys through it, and all of them passed because they were all software generated.

Steve: Yes. Good, good.

Leo: Sorry, I didn't mean to interrupt.

Steve: Oh, no. No, that's a good segue. So the prevalence of web-based cryptocurrency mining. And this has an interesting twist to it. So there is a well-known popular advertising blocker, an adblocker called AdGuard. And they took it upon themselves to just sort of do a little research into the prevalence of cryptocurrency mining. We talked about it, I think it was just last week, or maybe the week before, about the fact that, while, yes, it might be annoying, if you go to an adult content site or a torrent or a pirate TV site or something, they tend to be the shadier sites on the 'Net. If your processor gets pinned because you've got JavaScript trying to do mining - and I mean, oh, my god, I can't think of anything less efficient than using JavaScript for mining. But as we said last week, it's not a security compromise. It's just an annoyance.

Leo: And it may even be a somewhat legitimate way of monetizing; right?

Steve: Well, that's where we're headed with this.

Leo: Ah, okay.

Steve: Is that wouldn't you rather, since it isn't a security problem, wouldn't you rather let the site you're visiting run mining on your computer rather than have you look at ads that are annoying?

Leo: Because it's not cycles you really care about.

Steve: Right. Unless you're mobile. If you're mobile, then maybe it does matter because your phone's going to heat up when you're there. But for a laptop...

Leo: They kill battery life, yeah.

Steve: Yes. For a laptop and a desktop, if I had a choice of letting a site I visit borrow my computer for the sake of monetizing my presence while I'm on the site, I mean; and, for example, if the page is in the foreground, not if I switch away to a different tab, then it's like, okay, sorry, I'm not looking at you anymore. So the idea is all of the proper influences align. The longer time you spend, the more frequently you go back, the longer you are there looking at their content, the greater the chance that lightning will strike, because that's about what it is these days on scoring a coin, and your computer will generate a bitcoin on behalf and for the benefit of that site.

And for large sites with enough users, the numbers begin to make sense. These guys note that, even though the incremental amount of revenue generated is not great, neither is it for ads. And so ads are also working on the large population model, and so does mining.

So we have an interesting picture here in the show notes from their blog posting, from the AdGuard.com blog post, where they said: Three weeks into crypto mining going viral, 220 of the top 100,000 websites are currently...

Leo: What?

Steve: Are offering crypto mining scripts. Yeah. And those sites, based on the popularity of those sites, 500 million users were doing mining on behalf of those sites during that period of time. And it's estimated that those sites earned about $43,000 in those three weeks.

Leo: This is actually great. Now, everybody in the chatroom's going, "I'm never going to do that, no way, I don't want to." But if it's got no impact on you...

Steve: Correct. It's not a security vulnerability. You're already running JavaScript like crazy. It's doing all kinds of tracking crap.

Leo: Do these sites do it explicitly? Are they, I mean, I've never seen this. Is somebody saying, hey, if...

Steve: No.

Leo: No. See, that's what they need to do.

Steve: Yes, exactly.

Leo: You have a choice.

Steve: Exactly.

Leo: Do you want ads, or do you want us to use crypto mining software?

Steve: Exactly. So the idea would be that, in the same way that our browsers had a DNT, a Do Not Track flag that they sent with every query, our browsers could have a "you may mine on me" flag. And so you'd be saying - and so when your browser says that, the site says, oh, good, we'd rather mine anyway than annoy you with ads. So you get an ad-free page, and your processor gets pinned, and there's some probability that your browser will succeed the bitcoin or the whatever cryptocurrency it is challenge, and the web server will score some cryptocurrency.

And it might even be possible to pool, that is, to use mining pooling to get incremental revenue just based on the total amount of time that a site is spent mining. I have to talk

to Mark Thompson, who really understands this stuff upside down and backwards. But I agree, Leo, I think it's a really interesting monetization solution, one that makes absolute sense.

Leo: It's going to be hard to convince people. But I think, man, if I thought I could get away with it, I'd do it. I mean, seems completely reasonable. I mean...

Steve: I agree. I can't see a downside.

Leo: And people are saying, well, what about the electrical cost? In aggregate, it will be high because that's the nature of bitcoin mining. But to any individual, it's not even going to be a penny.

Steve: Yes. It's so distributed. The whole thing is so distributed that I think it makes sense. And, I mean, if you put your hand next to your computer and looked at the heat that's being pumped out of the fan?

Leo: It's working. It's working anyway.

Steve: Yes.

Leo: People say how much bandwidth will it use? None.

Steve: None.

Leo: It's all local.

Steve: Yes.

Leo: I think we'd have to educate people if we were going to try this. I'd do it in a minute. The problem is I don't really - wouldn't it be great if I could figure out some way, as you're listening to a show, that it's bitcoin mining in the background. Two hours later we've paid for Security Now!. Costs us $1,000 an hour to do a show. Could you, if you had enough listeners or viewers, make - I bet you could make a thousand dollars an hour. But anyway, it's an interesting thought. I'm happy [crosstalk].

Steve: It really is.

Leo: Folks, I wouldn't do it without telling you, and I have no plans to do it. But I think it's a great idea.

Steve: I do, too. I just, I mean, and it hit me as I was reading along, thinking - and then it's the way they ended their blog post. I mean, and here's an adblocking company. Their whole business is to block this. And by the way, uBlock Origin already does, and AdBlock Pro or Plus, Adblock Plus does. So there's already been a response by the controllers in the industry to this. But I'd be inclined to say, yeah, let's allow mining if it means that I get a better, you know, if I could support the sites where I'm visiting. I mean, okay. Wikipedia. Fine. Mine on my machine while I'm browsing Wikipedia in order to send some money to them. Why not?

Leo: Wow. Oh, I would do that. I already send them, like, $100 a month. That would be a good thing for me. Wikipedia won't put ads on. Kudos to Jimmy Wales for not doing that.

Steve: Yup, yup.

Leo: I think that's a very responsible thing to do. But that means they have to beg.

Steve: And so the idea would be, in the Wikipedia model, they could present a banner if they see that your browser does not have a cookie explicitly allowing mining. And they say, "Hi. If you would like to support our site, please allow us to run bitcoin mining on your computer, in your browser, while you're on Wikipedia." And so you click yes, and that's - so that sets a static cookie. And from then on, every time you go to Wikipedia, the mining starts up. Maybe there's a little banner that says, "Hi. Thank you for your support. We are currently mining bitcoin on your computer."

Leo: Right. It'd scare people like crazy.

Steve: It could be totally done. Well...

Leo: Somebody said you've got to - it's all in the naming. Chatroom says call it "Ethical Mining."

Steve: Ooh, I like it.

Leo: Thank you, Dave Redekop. That's Nerds On Site. He says...

Steve: Ah, perfect, yes, David.

Leo: EMO, Ethical Mining Operation. I love it. It's one of those ideas that's brilliant, and I just feel like people are so scared of the whole idea they wouldn't - they don't like the idea of somebody running a program on their computer. We do SETI@home. It's like SETI@home.

Steve: [Crosstalk] when you go to a browser.

Leo: What are you doing? That's what you're doing.

Steve: I mean, and see, that's just it, is that JavaScript is going insane every time we go anywhere on the Internet.

Leo: Right.

Steve: I mean, and that's why I gave up. Remember NoScript. I was pushing NoScript forever until I finally said, okay, it just breaks everything.

Leo: I know what you call it.

Steve: The reason it breaks everything...

Leo: Clean Bitcoin. No.

Steve: Or CleanCoin.

Leo: CleanCoin. Ohhh. CleanCoin. We can set up a Dogecoin mining operation. You'd never make any money, but - all right. I'm not going to do it. Don't worry, folks.

Steve: So this was an interesting piece of work by a guy named Felix Krause. It got sent to me by one of our listeners, and I thought, oh, you know, this really does raise a very good point. And that is, it is easy to obtain someone's iOS password.

Leo: I've thought this for a long time, and I didn't want to say it.

Steve: Just ask.

Leo: Yup.

Steve: And, yes, on the show notes we have a side-by-side sample of something real from iOS, and then a sample pop-up generated by a phishing application.

Leo: Can I just say I feel like this is a problem across the board - OAuth notifications, when people ask me to sign into Google - because there's no certificate attached to these pop-ups.

Steve: Yes.

Leo: It's very hard to validate.

Steve: Yes.

Leo: What would you click on to know if this - it looks exactly the same.

Steve: Yes. And the problem is, as Felix notes, we have trained people to simply enter whatever is asked of them because there is, I mean, there's no model, no robust, no rigorous model for when we should receive one of these. These little sign-into-iTunes things pop up all the time.

Leo: Seemingly randomly. We've talked about this before. You say, in fact - I remember. I said, "Why, Steve, why?" And you said, "Well, I can only assume that Apple knows that there's a potential problem at this point, so they want to reauthenticate." But end-users don't know why. They just...

Steve: Nope.

Leo: And we're trained to say, okay, fine, another Apple password pop-up.

Steve: Yup. In fact, Felix says: "iOS asks the user for their iTunes password for many reasons. The most common ones are recently installed iOS operating system updates or iOS apps that are stuck during installation. As a result, users are trained to just enter their Apple ID password whenever iOS prompts you to do so. However, those pop-ups are not only shown on the lock screen and the home screen, but also inside random apps, for example, when they want to access iCloud, GameCenter or In-App-Purchases. This could easily," he writes, "be abused by an app, just by showing an UIAlertController" - which is the name of that in the developer docs - "that looks exactly like the system dialog. Even users who know a lot about technology have a hard time detecting that those alerts are phishing attacks." And looking at it, there's no way to tell.

Leo: You can't.

Steve: There's no way. Now, he went on further. I didn't show another sample. But in this instance he is showing your email. So the app would have to know that, which frankly is probably not hard or not impossible for it to know. But there are other iOS real dialogs that don't provide your Apple ID as a reminder. Again, a user - and even if there weren't, but there are, but even if there weren't, a user is still going to see that and go, okay, I don't know why, but my phone wants me to enter this. So they're going to.

Leo: Somebody in the chatroom, Don S., says, "When I see those, on the iOS,

anyway, I press the Home button, maybe even twice, because iOS will ignore those. It will sit on that screen until you sign it. But apps cannot."

Steve: Yes. And that is exactly what Felix...

Leo: That's a great trick.

Steve: That is what Felix said. That is his solution to this.

Leo: Ah, okay.

Steve: Is when you see that, press the Home button. It won't work if it's a real one.

Leo: Very interesting. Of course, it doesn't solve the Google OAuth pop-up problem or any of this.

Steve: No. And in fact my work in the last week on SQRL has been exactly this problem, that is, because what we were just talking about here with the iOS pop-ups, that's an instance of the bigger problem of spoofing. Spoofing is just - it's the intractable problem because it involves the human factor. And so, as you know, we're down very near the end of the SQRL project. And someone brought up the point that a web page could present a dialog that looked just like the standard SQRL login prompt, and how would anybody know? And so essentially I've duplicated the UAC look that Windows adopted for this reason, to create something that a web browser can't do because the web browser never has control outside of its own borders.

Leo: That's why Windows darkens everything and...

Steve: Yes.

Leo: And, yeah, that's interesting.

Steve: And I'm doing the same thing. I monochrome and darken the screen so the dialog stands out, and a browser can't do that. But unfortunately, I mean, in general, spoofing leverages the human factor. And there's just not a good solution to it.

Leo: It's another reason why you should be concerned about this feature in Android that applications can write over other applications.

Steve: Right.

Leo: A feature Google won't fix because it's used by Facebook for chat bubbles and other - yeah.

Steve: Exactly.

Leo: But that's exactly how you'd use that. You'd just write over another application.

Steve: Yeah.

Leo: Making it even more hard to detect.

Steve: Exactly. And you can't tell who you're talking to.

Leo: Who's writing that, yup.

Steve: Yup.

Leo: Wow.

Steve: So Google did follow through with what we were expecting. We talked about it, I think it was just last week, the so-called APT, the Advanced Protection Program. It has gone live. I've got a screenshot of Google's announcing it with their Get Started button. The screenshot reads: "Google's strongest security for those who need it most." And they're explicitly not suggesting this is for everyone because, as I mentioned last week, and now we have some additional details, there's a tradeoff. So it's intended to protect the accounts of "those most at risk of targeted attacks - like journalists, business leaders, and political campaign teams."

The main defense is a physical security key used for authentication, and it requires the use of Chrome, that is, their browser, because that's the other side of this authentication requirement. So you have to use Chrome, can't use a non-Chrome browser. And there are several tradeoffs. One is this you have to have a physical security key - so that's not free, I mean, there's some cost to that - which is used to guard against phishing because the physical security key will be required every time you log into a device. You can't even do, like, "remember me." So this will replace and disable all other forms of authentication, including SMS, thank goodness. But even the Google Authenticator app, sorry, physical key required. Also it limits data access and sharing. Third-party apps will no longer have access to Gmail or your Google Drive. And email is only available through Gmail or inbox clients.

So since iOS apps do not support security keys, Google has noted in their documentation that Apple mail, contacts, and calendar apps will not work, and users will be forwarded to standard apps on iOS. And Google services that require a sign-in, like Photos, will only be available through Chrome.

And then, finally, the last measure is that it's designed to encounter impersonators who claim to be locked out of their account. Again, we've talked about the weakest link. The weakest link is account recovery. It's like, oh, I forgot my password. And it's like, oh, fine. Here, we'll mail it to you, or we'll send you a link that, you know, blah blah blah. So Google notes that these extra steps, like reviews and requests for more details, will be there in place during account recovery process. So the recovery process will take a few days, they are saying.

Leo: Wow.

Steve: In other words, yes. So this is not free. A physical key is required, and with it comes substantial, I mean, but correct - they're doing the right thing. As we often talk about, there is a security versus convenience tradeoff. If you have convenience, you don't have security. If you have the convenience of one-click password recovery, there's no security there. So Google has decided they're going to get...

Leo: How do you get this?

Steve: ...serious. Good question.

Leo: I don't think they say. They probably approach celebrities and at-risk people.

Steve: What happens if - yeah, because, well, there is this dialog. What happens if you google Advanced Protection Program?

Leo: Yeah, because I would do this.

Steve: Yeah.

Leo: I'm actually doing, you can kind of do it anyway. What I do with - in fact, this came about, remember I was talking a couple weeks ago about how somebody had hacked Hover, my - yours, as well, my domain registrar.

Steve: Yes, registrar.

Leo: And added an email, because I use them for email redirection, and added another email account to the redirection so that they were getting recovery emails. So one way to mitigate that, I decided to stop using my main email address for all of this stuff, but to create a special address on a server no one knows about as my recovery address. That way you can't guess it, anyway, and you'd have to somehow get access to that server. It's interesting. With LastPass, which is of course the thing I secure most aggressively, it reencrypted all my keys when I did that. So I had to re-log into LastPass everywhere. I use a YubiKey for two-factor on LastPass which is

great because you can't - unlike Google, you can't revoke it or say, well, let me use some other method.

Steve: Right.

Leo: Once you turn on hardware two-factor on LastPass, you either have to have that or approve an email to the recovery address that says, okay, yeah, turn off the YubiKey. So I feel like that's pretty secure. Google unfortunately does, if you can't authenticate one way, Google gives you 20 ways you can authenticate, which I wish it didn't do.

Steve: Yes. And that is the problem. And so in this program they've said no. It's not free, but we're going to give people an option to have really strong security with a bunch of tradeoffs. That is, your Gmail is siloed, essentially, and cannot be accessed by other apps and other facilities. And account recovery, we're going to really make that difficult.

Leo: Looks like you can do it. Just go to - it's landing.google.com/advancedprotection. And then I see a Get Started button. So it says you'll need two security keys.

Steve: Yes. That's what we discussed last week.

Leo: Buy one Bluetooth key that will work on your phone with a cable. All right. I already have a Yubico FIDO compatible. And then they say buy a - wow. So I need to go out and buy a key. Wait for keys, and then you can turn on Advanced Protection. Wow.

Steve: And is there a cost? Is there a billing to it, or is it just...

Leo: Let me see. Let's see. Oh, now it's going to - see, there's that pop-up that I just - I don't know how to trust. I guess at this point I could look at a certificate because it's a page. But there's the login. First login. I'll figure out what it costs. I might do it. I'm a target; right? You are, for sure.

Steve: Yeah. Okay. So one note of miscellany, and that is I want to make peace with my listeners who are in love with "The Orville." Because I'm not.

Leo: You did try it; did you?

Steve: I tried it again. I watched the "Krill" episode. And when they finally went into the chapel for Krill services, I just though, this is just not serious enough for me. And that's it. Sci-fi for me is not humorous. And I understand. I've had some very good friends of mine recommend it, I mean, people I really respect, Trekkies saying, "Oh, Steve, make

sure you're seeing 'The Orville.' It's like, well, I did make sure. I saw the first episode. I got 10 minutes into that one, it was like, uh, no. This one I just kind of hung in there, but finally I thought, no, it's just, you know, it's not for me. But please, I get it that a lot of people like it. But we don't all have to like the same thing. So peace.

Leo: So one roadblock on this.

Steve: Yes?

Leo: The security key that Google recommends is currently unavailable.

Steve: Whoops.

Leo: So I guess why you need two keys, one is - it says one that you can work with a cable connected to a tablet or a phone. See, that's the problem with the YubiKey is there's no...

Steve: Ah, to power it, probably, in order for it to get power?

Leo: They recommend this Feitian MultiPass FIDO Security Key. I already have a YubiKey compatible with FIDO, which is my YubiKey 4. But I don't - I'm not sure what the requirements are of this Bluetooth key that'll work on your phone. I can't really use a YubiKey on my phone.

Steve: So if it's Bluetooth, it must have a battery in it. And so it uses BTLE...

Leo: Oh, I get it. I get it, yeah, BLE for compatible mobile devices. So there is an NFC. You know the Neo, YubiKey Neo has NFC in it. But I don't know if they make a - so I'm looking for a - I probably don't have to use this Feitian. I can use anything that has BLE.

Steve: Right.

Leo: This one has a rechargeable battery. Yeah, you're right. You nailed it. Okay. At least I know what I can look for here. All right. I distracted them from "The Orville." That was my plan. What? Orville? What?

Steve: Good job, Leo, good job. So I got a nice note from Kristopher Ting of Powell Technologies in Johnson City, Tennessee. This was yesterday, the 16th. And of course the subject was "Another SpinRite testimonial for you." He said: "Hey, Steve. I'm a longtime user of SpinRite and longtime listener of Security Now!. Wanted to send you a quick success story, in case you wanted one for an upcoming episode." Yeah, like the day after. Thank you very much, Kristopher.

He says: "My dad called me a few days ago, telling me his Windows PC was suddenly telling him his second hard drive (the one on which he stores all his important data) was not formatted. It was the dreaded 'This drive is not formatted. Would you like to format it? Yes/No' message." He said: "After telling him to immediately click 'No' and remove the drive, he brought the drive over to my house, and I mounted it into my PC. As I expected, my Windows PC had the same problem accessing any information on this drive."

So at this point we're talking a total loss of all of his dad's important data, which he explicitly and deliberately stores on this second drive. And he writes: "Saw it as RAW instead of NTFS." He says: "I immediately rebooted into SpinRite, ran a Level 2 on it. It was a standard 250GB SATA drive, so the estimated time to complete was under 40 minutes. At around the 95% mark, SpinRite detected a sector that needed deeper analysis and, after about three minutes, it was ultimately marked as unrecoverable. The rest of the drive checked out okay.

"Thinking that the one unrecoverable item was the cause of my dad's original issue, I felt certain that I would reboot only to find I had the same issue and that his data was officially toast. But to my pleasant surprise, the drive and its original partition were there and were now fully accessible. I immediately copied all his data onto my PC and will be replacing his with another in the near future. Dad's happy, and of course that makes me happy. Kudos to such a great product, Steve. Thanks, Kristopher Ting."

And I'll just note that, first of all, we've seen instance after instance where, even when SpinRite doesn't explicitly acknowledge that it fixed something, everything is fine afterwards. But the other thing that I wanted to remind Chris and others of is that "unrecoverable" only means not every single bit was recovered. But there are 4096 bits in a sector. And if the sector is part of the drive's directory system, the file system metadata, you don't necessarily need all the bits. And SpinRite does a partial reconstruction, even if it cannot do a full reconstruction.

So even if that sector was actually the problem, and we don't know whether or not it was, SpinRite will get you, for example, 4090 out of 4096 of the bits, and put them back and make the sector readable where it wasn't before. So in many instances, that's enough to allow the drive to then say, oh, I have a partition again, and allow access to it. So I often see a lot of people talking about how, oh, you know, it didn't perform full recovery, so it didn't do anything useful. It's like, uh, well, actually that's not the case. So we've talked about how large a gray area there is between everything working perfectly and nothing working at all, and how SpinRite is often able to pull a drive back out of the gray, back into the light. So it did so in this case. And Kristopher, thanks for sharing.

Leo: So I've done some research.

Steve: Ah, good.

Leo: Google, you're exactly right, Google's a little more clear about the two security keys that are needed for their Advanced Protection. And the reason you need to, if you didn't ever use anything with iOS, you'd be okay. Because NFC would be sufficient, USB and NFC would be sufficient for Android and most computers.

However, iPhone and iPad don't support NFC. They only support Bluetooth LE; right? So you need two keys, one like a YubiKey 4, which that's what I have already, and then another, as you said, with battery because to do BLE you've got to have battery. It has to support FIDO UTF, that's the weaker Google version of SQRL. And this does. This is from - I don't know a thing about the company. I've ordered one, and we'll see what happens, from VASCO Data Security. They call it their DIGIPASS.

Steve: VASCO are good people.

Leo: Okay. And so this is a - it has a USB dongle, which is kind of interesting, but it also supports BLE. So this will work with iOS, as well as all the other devices. So I'm going to get this to supplement my YubiKey because I'm going to do it as an experiment.

Steve: Good, good, good.

Leo: And I'll be your guinea pig on this.

Steve: And I thought that the newer, like from iPhone, was it 6? I thought we all had NFC now in our iPhones.

Leo: We do, but only Apple can use it.

Steve: Ah ha ha, okay, got it.

Leo: And it's only used for Apple Pay at this point.

Steve: Right, right.

Leo: So, yeah, and then I'll let you know. If I can get through all of this, I'll let you know what the cost is in a second.

Steve: Yeah, cool.

Leo: For Advanced Protection. But I think if I've got my Google and my LastPass really locked, securely locked down, I'm not worried so much about the rest of it; right?

Steve: Yes. I think you need an anchor which is secure, and then everything flows from that.

Leo: Got it. Thank god I've got you, Steve.

Steve: So closing...

Leo: Have you brushed your tongue lately? No, go ahead, keep going. I'll talk about that later.

Steve: Okay, one second. Closing the loop, we've got a couple nice bits of feedback from our listeners. I guess he would call himself @crackruckles. That's his Twitter handle, @crackruckles. He actually tweeted to both of us, said: "Just found a great site to show what info is leaked from your browser and torrent clients." And I - you, Leo, and our listeners, IPLeak.net. And it's a little sobering to see, just going to this site, you know, there have been many of these through the years that we've talked about. But this is pretty comprehensive: IPLeak.net.

Leo: It kind of looks like ShieldsUP!.

Steve: And it just populates, and it keeps on going and shows you all kinds of stuff that's...

Leo: Now, Father Robert said I shouldn't show my IP address on the air. But I don't think there's anything particularly secret about the TWiT IP address.

Steve: Yeah. The only thing I could see is that it would subject you to some DoS attacks, perhaps.

Leo: Oh, I'm not worried about that. We've got plenty of protection here.

Steve: Exactly.

Leo: This is good. So this actually would be useful when I use my Tiny Hardware Firewall with a VPN and Tor.

Steve: Yes. Yes, yes, yes.

Leo: Get some idea about what's going on.

Steve: Go to IPLeak.net and see what it shows you.

Leo: Very nice.

Steve: Steverino, whose handle is @DaMoisture...

Leo: Okay. Okay.

Steve: He said: "Security at USAA Bank" - okay, this is security at USAA Bank - "just tried to tell me 'they have done the research and found that shorter passwords are more secure than longer passwords' when I complained about their website's 12-character password limit." So Steverino, they're not used to talking to Security Now! listeners who know better. They're used to talking to people's, I don't know, cousins who have just, like, what, okay, really? Okay, well, that's good to know. I didn't realize that shorter passwords are better. Now I don't have to worry. That's right.

Ryan Scullen said: "How would DANE" - which is the technology we talked about a week or two ago where DNSSEC is being used to provide certificates. "How would DANE prevent man-in-the-middle attacks?" Well, and the answer is that one of the requirements for some man-in-the-middle attacks is DNS spoofing. And so it's really not DANE as much as it is DNSSEC which prevents various DNS attacks which essentially are carried out by somehow arranging for someone to get the wrong IP address for a domain they believe is legitimate and then going there for spoofing.

So it's not exactly tied in. But Eric Vollbrecht asked: "How is publishing a public key via DNSSEC any different than a self-signed TLS certificate, other than the method of distribution?" Okay, well, it's that last caveat. So Eric, you're right. It's not. You essentially are publishing a self-signed TLS certificate. It's TLSA is the protocol used. And but the key is the method of distribution. A self-signed certificate is less secure because you don't have a secure means of sharing it, essentially. If you get a self-signed certificate from a site you're visiting, and your browser says, oh, this site's certificate is self-signed, you want to trust it, well, you say yeah. But you could be at the wrong site, and now you're trusting the self-signed certificate from the wrong site, from a spoofed site.

So the beauty with DNSSEC is by securely and authenticatively obtaining the certificate for the connection, you solve the problem of distribution. So Eric, you kind of answered your question. Other than the method of distribution, there's no difference. But it is the securability of the distribution of the key which that's as useful as the key being signed by someone you trust, which is the current certificate authority model that we're all operating under today. So either you get a certificate from someone you hope is correct, but it's been signed by someone you trust, therefore you trust it, or you have a secure channel, a secure means of authenticating the source of the certificate, which is what DNSSEC and DANE under DNSSEC provides.

And Barbara asked, regarding the security.txt file that we discussed last week, remember, the one that will be coming soon to run alongside robots.txt, she asks: "What stops bad guys from spamming the contact email address?" That's exactly the thought that I had, and I forgot to mention it last week. So thank you, Barbara. One of the things you can put in the security.txt file is your email address. You can also provide a place to pick up your PGP key, to allow people to send you encrypted secure email that only you can decrypt, which is useful if they want to establish a secure dialog with you. But you can also give them a URL.

And so what I will do, because naturally bad bots, spam bots are going to be scrubbing the 'Net looking for security.txt files, I won't put an email address there. I will put a link to a form which has "I'm not a robot" CAPTCHA stuff in order to prevent spamming. So the solution is you can do many things, you can put many things in the security.txt file, one being not an email address, but rather a link to a web page where you can then do anything you want to. So you are able to take it out of the email channel, if you choose.

Richard Petrie said: "You asked on last week's episode why people were averse to printing password recovery tokens." That is to say, actually talking about why they're averse to printing QR codes. And he responded: "I have no physical security." And that's of course an absolutely valid point. I do. I'm able to, you know, my environment is just me. And so I'm able to provide adequate physical security for the slips of paper where I have my QR codes printed. But I completely get it that, if your environment has inherently got a lot of people coming and going, and you don't have control over it, then yes, I would absolutely agree that without physical security you would be better off printing them to a PDF or taking screenshots and then storing them in a safe fashion online, where you do have security over access.

And, finally, Brad Dux said: "Regarding people wanting no password for SQRL." He says: "The protocol is open. So do you not expect people to release implementations without a password?" That's a very good point. And the password requirement is for the client, that is, not only is the protocol open, the protocol never discusses passwords. The passwords are not even in the SQRL protocol because, for example, biometrics, a fingerprint or facial recognition or whatever you want to do could be used. It could be the physical proximity of, Leo, the Bluetooth key that you'll be getting. Anything could be used as the additional factors.

The reason we need some control is that we are turning our authentication over to SQRL as a third party. So we want to make sure that's not abused, that there's some interaction of some kind that permits SQRL to authenticate on our behalf. Nothing in the protocol requires it. And in fact even my client allows you to reduce it to a single keystroke, if you want to, after you provide the long secure password once. So, I mean, and I think users are going to actually see that it is just not burdensome. Normally, we call it the "quick pass," is the first four characters of your much longer password. You have to provide that once. And then, from then on, per authentication, it's just one, two, three, four, just you're able to just type the first four characters. And you don't even have to hit Enter because the system knows that it's four characters.

So, I mean, we've really streamlined it. I think people are going to be - won't feel that it's any barrier. But technically, Brad, you're right. Not only is there nothing to prevent people from doing non-password-based SQRL clients, I mean, I don't think anyone would want to use one, once they understood that anybody could come along and click on the QR code on a page and be logged in as them. They would want some protection. So it's available.

Leo: Nice. I'm using my Tiny Hardware Firewall, VPN, and BitTorrent. And my IPLeak thinks I'm in Hungary.

Steve: Nice.

Leo: I like that. And it says it can't figure out what my IP address is at all, let alone leaking DNS requests or WebRTC requests. It's just confused as all get out.

Steve: Perfect.

Leo: Yeah. It does, it's really amazing how much information the browser leaks, however. That's really frustrating.

Steve: Oh, my goodness, I know. It really is. And they're just pouring out crap. I mean, like which extensions you've got installed.

Leo: Yeah, right. And of course that's a fingerprint technique. There's probably enough unique information in its configuration here to fingerprint me.

Steve: To still track you, even though you're coming out of Hungary.

Leo: Yeah. What's Leo doing in Budapest?

Steve: Well, he's been traveling a lot lately, you know. He's liking those vacations.

Leo: Why is his Surface laptop in Budapest? I do, I have to say, at some point I'd love for you to just certify, if you can, WiFiConsulting and their Black Hole Cloud VPN because one thing we're learning, especially in light of KRACK, is that VPNs are not all made the same, and some of them are actually horrible.

Steve: Right. In fact, there was a story that I let go by last week about VPN logging of everything that their own clients and their own customers were doing.

Leo: Right, right. I'm pretty sure these guys are great, and I've been using them for years. And I love the fact that I'm going through hardware. This thing appears to be an Ethernet dongle; right?

Steve: Nice.

Leo: But then it has two WiFi radios in it. It's just very clever, I think. Anyway, it's also pretty fast, which is nice. Continuing on, let's talk KRACK.

Steve: So, okay. We've discussed over the years WiFi details, WiFi crypto, extensively. But I need to do a little bit of a review because there haven't been any, like, horrific problems. I mean, remember, Leo, back in the beginning of the podcast, I mean, there was stuff happening.

Leo: WEP. WEP.

Steve: WEP and oh, my goodness, and TKIP and RC4 and all these problems.

Leo: That's what's so terrifying about this, because we had settled for five or six years now on the simple recommendation, you don't need to do anything else, just use WPA2, PSK, and your network's secure.

Steve: Yes, actually since - it's been 12 years, since 2005.

Leo: That long, wow, okay.

Steve: Yeah, we've just all gotten kind of happy. So, okay. So I have to lay a little bit of foundation for people to understand the nature of why what's happened is a problem. In cryptography there are two broad categories of ciphers. There are block ciphers and stream ciphers. In a block cipher, which is, for example, what you use if you're going to encrypt your hard drive, the cipher algorithm itself, like AES, is itself a block cipher. And in the case of AES it uses 128 bits, which is 16 bytes. And so that divides evenly into the size of a sector, and everybody's happy because everything kind of works right.

Communications like WiFi use stream ciphers. And stream ciphers and the ones that are used with WiFi use this magic XOR which is really kind of cool the way it works. And we've often talked about both the power and the danger of exclusive OR. The way to think of XOR is that it's the difference between two bits. So if the two bits are zero, well, they're the same, so the XOR is zero. If they're both ones, again they're the same, so the XOR is zero. But if they're different, if the two bits are different, if it's zero and one or one and zero, then the XOR of those two bits is one. So think of XOR as the difference between the two inputs.

And I'll just say briefly that that's the whole secret of RAID 5. That's the way RAID 5 works. If you have two drives, and you want to be protected against either of them failing, you add a third drive which is the XOR of the first two. That is, it's the difference between those first two drives. And so as data is written to either of these two drives, the third drive is kept synchronized. It's always kept updated to be the difference between the first two drives. Now, if the third drive, that one were to die, well, who cares? It's like, okay, go get another one. You just lost your difference drive, but it doesn't matter because you've still got your first two.

But if either one of the first two dies, think about it. You can figure out what the other drive of that first pair was by comparing the surviving drive to the difference drive, to the XOR drive. And that allows you to compute the entire contents of the drive that died. That's how RAID 5 works. So anyway, just something as simple as this XOR operation is very powerful. And the other thing that we've talked about in the past, when we've talked about stream ciphers, is something very counterintuitive. If you took plaintext, just a plaintext stream of bits that is readable, and it's clear for everybody to see, and you XOR that stream with noise, with randomness, the output is the best encryption in the world. Which is, like, what? It's that simple? Yes. Because XORing with random noise, it randomly inverts bits of the stream. And there's no way to know from the output which bits were inverted.

And so that's what's sort of difficult conceptually to, like, believe. It's like, well, if you just flipped the right bits back, then you'd have the original plaintext. It's like, yes, but you don't know which bits. And so it's weird that taking plaintext and just XORing it with a random stream of bits is unbreakable. It's really strong crypto. And here's the key when it comes to WiFi, because that's what WiFi does, is it simply generates a pseudorandom bitstream and XORs the data that's being sent. And then the other end generates the identical pseudorandom bitstream and XORs what's received, which recovers the plaintext. And that works.

Now, there are some requirements, though, for security because, powerful as this XORing plaintext with something pseudorandom is, you can kind of get a sense for the danger. In the same way that the RAID drive was like a lost drive was recoverable, there is a danger, if you know what the plaintext was, and you can see what the encrypted data is, that reveals the stream. If you XOR the plaintext and the ciphertext, you get the bitstream. So this has to be used very carefully. It's very fast, very lightweight, very secure. We've all been happily using it for 12 years. Well, and stream ciphers for even longer. As long as your source of the pseudorandom data is good, you're okay.

Now, the problem is that we need to use a static key, that is, the key needs to be agreed upon by the endpoints and not change. But the danger with a stream cipher is ever reusing the same stream. That's the problem. You must provide a mechanism from ever allowing the same pseudorandom stream to be reused against the plaintext because, if you do that, turns out it's trivial to crack it. So to solve that problem, that is, not needing every packet to have its own key, but have long-lived keys, they introduced the notion of - and this is something we've talked about in the past - an initialization vector, an IV. And the concept there is that it can be known, but it must never be duplicated.

So the idea is that, when you're starting to encrypt a packet, you increment this nonce, which can just be a counter. And that's the initialization vector. It mixes in with the key to produce a unique pseudorandom bitstream for that packet. The other side has a synchronized nonce, and it's got the same key. So it's able to increment the nonce, packet by packet, and reconstruct the same pseudorandom bitstream to get the data, to decrypt the data by re-XORing it against the same pseudorandom bitstream, and out comes the plaintext.

Okay. So now that means never repeating the bitstream used to XOR the plaintext is crucial. It turns out, not surprisingly, that's not guaranteed. What these researchers, two guys from Belgium, figured out was there was a way with a lot of preconditions to force a reuse of the nonce. Which, as I said, is absolutely disallowed. So when a client - meaning iOS or iPad or Android device or a Mac or a PC, that is, anything that's not an access point. That is, anything that wants to connect to an access point, that's the client, also known in the terminology as the "supplicant," as opposed to the authenticator.

When the client wants to connect to a WiFi access point, it sends an authentication request. The access point sends an authentication response, that is, an authentication response back to the client. The client sends an association request, and the access point sends an association response. So that's the first back and forth. That sort of just gets things rolling between the two. That establishes their agreement to then negotiate secrets. So that is just authentication and association to sort of establish the dialog between them, but no secrets yet.

So then they do what's known as the four-way handshake, that is, after those first, then they do the four-way handshake. The access point sends the first of the four messages to the client containing its nonce. So it makes up a random value, sends it to the client. The client then uses its own nonce with the one it received from the access point to derive

what's called the PTK, which is one of the keys used for WiFi. And then it sends its nonce back to the access point. That's the second of the four packets in the four-way handshake.

Now the access point has the nonce it sent and the nonce it received, so it's able to derive the identical PTK key. So now both sides, having exchanged nonces, have this PTK key after two packets, the first one from the authenticator, from the access point; the second one from the client. Now the access point sends message number three of the four, containing an additional key, and the client finally sends the fourth packet of this four-way handshake back to the authenticator. And so that's the four-way handshake. That's the way things normally go.

What these guys discovered, essentially, was there had been a mistake made from the beginning, from the first moment that this technology was designed, which arises if the client's fourth packet, that fourth phase of the handshake, it sending back essentially a confirmation. If that never arrives, the access point will resend message three, thinking that, you know, not knowing whether message four got lost or dropped or collided with or whatever in transit; not knowing that the client didn't get message three. All the access point knows is that it didn't get message four, which is the result of the client receiving message three. So it resends three.

It turns out that the protocol has an error such that, when any client receives message three, it resets the nonce. And that's the vulnerability. As I said, we cannot ever allow any encrypted data to be reencrypted under the same bitstream. While the keys are not going to change, we're relying on the nonces to change. So if we re-zero the nonce, we're going to reuse the same bitstream, and that collapses our security. So this, it turns out, is difficult to implement in practice.

Okay. So assuming that you don't have - no one assumes we have an evil access point because, after all, it's the access point that you're trying to talk to. There's no reason to have an evil access point that is maliciously resending that third message in order to cause the client to reset its nonce because the access point knows what the client's saying anyway. So one of the things that was not well covered is this requires a man in the middle. That's one of the first things that has to happen. This is man-in-the-middle attack, meaning that a third party has to somehow arrange to intercept the communications between the access point and the client.

Well, in a wireless mode with radio, that's not super difficult except that it's a little complicated because the MAC address of both of the nodes, both of these endpoints, is mixed into the crypto material. So a man in the middle can't have a different MAC address if it's pretending to be the access point for the client, or none of the crypto will work. It'll all break. So the man in the middle has to have the same MAC address. Except you can't have two of the same MAC addresses on the same Ethernet. And remember that WiFi is just a version of Ethernet. So the way these guys very cleverly solved the problem is they run their attacking man-in-the-middle radio on a different channel, with the same MAC address as the access point. That way the crypto doesn't break, and there's no inter-MAC collision because they're deliberately on different radio channels.

So now this rather sophisticated man in the middle has to be within radio range of everybody and has to be quick on the draw, able to intercept, and essentially it's impersonating the access point to the client, and it's impersonating the client to the access point. And it uses its man-in-the-middle position to maliciously resend that third message to the client in order to force the client to reset its nonce because of a protocol documentation error that's always been there. And just that crashes the rest of the link security. Now, it's true...

Leo: Wow.

Steve: Yes, yes. So, okay. So several things. First of all, notice that all of this is the client, is on the client side. That's why I said it's not technically necessary for any of us to worry about our access points. It's a malicious access point in the middle that is the problem, leveraging the client's logic on the client side. So that's really what needs to get fixed. It'll be good for us to update our access points because access points can behave as clients if they're connecting to another access point. That is, when you want to connect like a mesh, if you have any kind of a mesh system, per the WiFi protocol, somebody is the access point; somebody is the client. Or in WiFi terminology, one is the authenticator; the other is the supplicant. So it's the supplicant side, the client side, where the vulnerability exists.

So it'll be good for us to update our firmware, but it's not crucial. What's crucial is that the things that connect to the access points, the clients, that's what needs to get updated. But also understand, I mean, somebody has to want to do this. I mean, it's going to require - and there are no tools yet available for exploiting this. It'll be nice to see whether there are some tests that become publicly available, and I imagine there will be some.

Leo: He has a Python script he demonstrated, and he said as soon as it's mitigated, after a reasonable amount of time, he'll release the Python script. But I wonder now, does he have to have some specialized hardware? Or could he just use the laptop WiFi radio to do all this?

Steve: Well, for example, what we could do, you could put this as a - someone could build a test in an access point because it's the access point abusing the protocol to the client that is the problem. But, for example, anything can be an access point. So you could, for example, have a laptop...

Leo: Do an ad hoc WiFi network, yeah.

Steve: Yes, exactly, a laptop running this Python script, which appears as an access point. Then you would have your other devices, your tablets and your Android and your iOS client...

Leo: Simulate the supplicant.

Steve: ...connecting to it as the WiFi radio and see whether they are robust against this attack. So it turns out...

Leo: But could you perpetrate the attack from a laptop, is what I'm...

Steve: Yes, you could.

Leo: You could, okay.

Steve: Yes. Yes. So that's the way it would be done is somebody sitting at Starbucks, for example, to use my often-used...

Leo: You could put it in a Pineapple, probably; right?

Steve: Correct, correct. Or a Raspberry Pi. Anything that had a WiFi radio would be able to do that, yes. So by weird coincidence, Microsoft and Apple misimplemented the original insecure specification in a way that made them invulnerable to this. The specs showed that - and the spec is a little unclear, too. Matthew Green had an interesting statement about how this happened, saying that really, you know, the IEEE process - the IEEE is the organization which is responsible for the Wi-Fi Alliance and for managing the spec. As Matt said, look at anything the IETF has done, and you'll see RFCs and everything in the open for everyone to see and inspect. You can't get the specifications from the IEEE. You have to be a member. You have to pay dues. I mean, I've often been thwarted by wanting to look at the WiFi spec. It's not public, which is just crazy.

Leo: That's weird. That's ridiculous.

Steve: It's crazy. It's absolutely wrong that we're all using a non-public, non-published specification. It's just wrong. But that's the game that they've always been playing. Maybe something will change.

Now, on Android, from 6.0 on, it's actually the worst of all because there's a reading - oh, I'm sorry. A particular implementation which appeared in one of the code bases that Android borrowed from, one of the open-source code bases, didn't only re-zero the nonce, it re-zeroed the key. So when you do this to an Android 6.0 and later device, it's like worse than anything. It actually switches to a null key, which of course everybody knows is all zeroes. And so it instantly allows you to access the communications of the device. So not good.

I'm just trying to think if there's anything else. Oh, yes, remediation. All that is necessary to fix this is that a Boolean flag be added to the software. And I've already looked over some of the diff files of the source code which has been updated. And it's simple to fix. You just do a little bit of logic to say, if the key has already been established, and I'm getting another message three from the access point, don't clear the nonce. Thank you very much. That's all it is. That's all you have to do. Just don't do that. And had that always been there, we would have never had this problem. So it is...

Leo: And that's on the client side, not the access point side.

Steve: Yes, correct.

Leo: That's really critical, yeah.

Steve: Yes, yes. So there need be no panic at all about access points that are not connecting to other access points.

Leo: Although your mesh router does do that, in effect.

Steve: Correct, yeah. And some of the routers actually have a checkbox where you can disable it to function as a client. And so if you wanted to be safe until your firmware is updated, if you had the option in your access point, in your WiFi router, to disable behaving as a client, turn that off, and that will make you completely safe. Then it can't be used as a client in order to connect. In which case, I mean, even then the danger is minimal.

Leo: But if you had an Android phone or an unpatched Android phone or unpatched iOS phone, it's a client, joining a WiFi network.

Steve: Except iOS was never vulnerable.

Leo: Right, because it misimplemented it; right.

Steve: Exactly.

Leo: But if you have an Android phone that properly implemented it, and now it's a client, what do you do? You can watch the traffic between the two because it's unencrypted now? Is that what's going on?

Steve: Oh, so, yes. And very good point, thank you. I tweeted this this morning, and I wanted to remind people that, okay. So what this would allow after an active attack, it would allow passive eavesdropping of some of the traffic. But remember, a VPN, TLS, and HTTPS are all going to be encrypting the traffic anyway. So WiFi encryption is an outer wrapping of encryption on top of what goes through it. And now, increasingly, what goes through it is encrypted anyway.

Leo: Although he seems to show on his - and I'm watching the video right now. He seems to show that you can now do a man-in-the-middle on the certificate, as well.

Steve: Well, yes.

Leo: And crack into TLS traffic.

Steve: You may be able to do that. What initially they were saying, that if you captured the TCP SYN packet, that you could get the sequence numbers and intercept and essentially hijack the TCP connection. I mean, so the idea is they're still sort of fleshing out how you leverage the ability to crack this. And you're right, I guess if you were able

to get the beginning of the...

Leo: Oh, I see. He's intercepted the OAuth is what he's intercepted.

Steve: Ah.

Leo: And that's in plaintext because you've bypassed the encryption.

Steve: Correct.

Leo: But a VPN protects you. So that's really the real lesson here, yes.

Steve: Yes, yes. A VPN protects you, and TLS and HTTPS protect you. So most of our communications today are encrypted even without WiFi encryption. For example, again, Starbucks. That's a nonencrypted connection. All everybody doing anything in an open WiFi access point is, you know, there's no password there. It's all in the clear. So we're relying on our own encryption, whether by VPN or HTTPS, to protect the data from other people who would otherwise be trying to spy on us.

So, I mean, this is going to get fixed. The good news is Windows did patch it officially last Tuesday, even though it never really was a big problem. There are, like, three different types of sort of subsets of this problem, and Windows was technically vulnerable to one of them, but not the main one. And the same thing is the case of iOS. As you said, 11.1 is now at beta, and we should have that soon. And after reading this, I wanted to make absolutely sure I was right because I couldn't see how an access point needed to be patched.

So I checked their FAQ, and they say in their own - the authors' FAQ says, "What if there are no security updates for my router?" And they write: "Our main attack is against the four-way handshake and does not exploit access points, but instead targets clients. So it might be that your router does not require security updates. We strongly advise you to contact your vendor for more details. In general, though, you can try to mitigate attacks against routers and access points by disabling client functionality, which is what I was talking about before, which is, for example, used in repeater modes, or mesh, and disabling 802.11r fast roaming. For ordinary home users, your priority should be updating clients such as laptops and smartphones."

And so that's from these guys who, throughout their 16-page paper that I read in order to exactly understand what was going on, I mean, they're finding any possible little nook and cranny, any little kink anywhere. So for them to say, yeah, you're probably fine with your router, I mean, that means you really are fine with your router. But we do want to get our clients updated. And as I said also at the top of the show, the response that our router vendors generate, even though we don't need to worry about it, it'll give us some nice feedback about how security-conscious they are and how responsive they are to these kinds of problems.

Ubuntu has been updated. I've got a bunch of - DD-WRT is already updated. OpenBSD is updated. BleepingComputer - I've got a couple links here at the end of the show notes. BleepingComputer has a list of the firmware and driver updates as they're happening,

and there's another link for a page maintaining a list of who's done things and who hasn't. So again, we're a little overheated on the router side because it's not a vulnerability against the router. It's against the client. And I think we're going to have those updated. And this, of course, this is the challenge once again with Android is there are an awful lot of Android devices. Literally anything that isn't updated for this has probably been vulnerable since 2005. So all Android ever.

Leo: As long as nobody knew about it, that's okay. We just don't know.

Steve: Exactly. Now we know.

Leo: So just to be clear, and I know you've said this, and I'm reiterating, updating your access point is not a mitigation.

Steve: Correct.

Leo: And that's the real problem is that we all have bunches of IoT devices whose companies are long gone or don't care and probably won't be updated.

Steve: Okay, now, the good news is this is the link key, that is, it is the key that's negotiated on the fly from point to point. It does not reveal your WiFi password. There's no...

Leo: But let's say they hacked a camera. They'd be able to see the camera traffic; wouldn't they?

Steve: Correct, yes. So, yes. So it would be the traffic to the device could be exposed.

Leo: Whatever that IoT is sending to the access point is now unencrypted and visible.

Steve: Yeah.

Leo: Unless it uses TLS or a VPN or something like that.

Steve: Correct.

Leo: Okay. Well, you said in your tweet yesterday don't worry, and I think you've helped people a little bit. And, frankly, if your IoT device is - I'm trying to think of - doesn't send video and doesn't send - if it's just, if it's like a Harmony Hub or something, who cares?

Steve: Yeah. Well, and the good IoT devices like the Amazon...

Leo: Like the Ring and the Echo, yeah.

Steve: They're all going to be over TLS.

Leo: Yeah. Oh, that's a good point, too, okay.

Steve: Yeah. All the good ones, the real heavy-duty IoT devices, are going to be encrypted separately.

Leo: So when I looked at the video again, I realized what he was showing was he could see part of the OAuth process. But if you've already got an account with this TLS-based website, that's encrypted, and there's no way you can kind of bypass that.

Steve: Yes. Correct, correct. So the only danger would be non-HTTPS, where things like passwords are going in the clear. Exactly like back in the day with Firesheep, where we were able to see that stuff in the clear. This crack breaks the radio encryption in situations where sensitive data is relying on the radio encryption from point to point, which we're actually relying on a lot less today than we have historically.

Leo: Oh, boy. I'm so glad you came on, as you do every Tuesday, and reassured us on this one. This is good stuff. I will mention that I have ordered the dual-dongle setup for Google Advanced Protection.

Steve: You've got dual dongles.

Leo: They still haven't told me how much it would cost. You have to register your dual dongles. So I have to wait till I get the dual dongles. I have one.

Steve: Next week. Next week; right?

Leo: But next Tuesday I will be able to tell you. I'll get them in Friday. Amazon says I should have it by Friday. I already, as I said, I already have an appropriate YubiKey. I just need that BLE device that you charge and all that. And I guess that means from now on I have to carry those around with me; right?

Steve: Yes, Honey.

Leo: But that's a vulnerability, too, because if somebody looks and says, oh, he's got

a YubiKey and that other thing, well, he must be doing Google there. They could try it; right? Well, we'll see what Google does.

Steve: Well, and again, the reason - what we're trying to protect from is Russia.

Leo: Yes. So not somebody mugging me.

Steve: So they're over there. Even if you're in Hungary still.

Leo: I should be okay.

Steve: Yes.

Leo: I can't wait, actually. I think that - I don't care about my Twitter, but I'm going to lock - Google, that's my Gmail. That's all my email. I want to lock that down. I want to lock down LastPass. Those things really are...

Steve: And the idea that when you switch into this mode with Google, they silo your Gmail so nobody else can get to it. Apps can't connect, I mean, everything else is like it's firewalled.

Leo: Good. That sounds pretty good to me.

Steve: Yup.

Leo: We'll see. I'll give you a report next week.

Steve: Yeah, now, I do remember, Leo, when you started using your YubiKey, that it was like never around when you needed it.

Leo: Yeah. Well, I've put it now on my car key. It's on my car key keychain. And so it's always in here. It's always here. But it's sometimes not in the same room. And, yeah, you have to get up and go get it and put it in. But LastPass only requires you authenticate once every 30 days.

Steve: Right.

Leo: So that's not the end of the world. And you just kind of get used to it. And if you are the kind of person who carries your keys, you know, yeah. Or maybe I'll just

make a little broach, a dongle broach, something that I always have it with me.

Steve: That's right. We want you wearing them around your neck.

Leo: I bet you people - I bet you sysadmins do that; right? They wear...

Steve: Oh, I guarantee.

Leo: Yeah, a little clip on your belt or a tie clip. Oh, yeah.

Steve: Remember when people were wearing thumb drives. Remember when people were wearing thumb drives on a lanyard.

Leo: I remember Woz coming into The Screen Savers, saying - he had it on a lanyard. He had a USB key, and he said, "That's 2GB on that thing." And we all went, "Whoa."

Steve: You must be rich.

Leo: Whoa, 2GB.

Steve: You're never going to fill that up.

Leo: Never fill that up. Steve Gibson's the greatest. GRC.com is his website, the Gibson Research Corporation. That's where you go for SpinRite, the world's best hard drive maintenance and recovery utility. You must have it if you've got hard drives. Even SSDs, you've got to have it.

Steve: Fixes them right up.

Leo: Why don't we make that the new SpinRite logo? Live long and SpinRite.

Steve: Ooh, I like that.

Leo: Live long and SpinRite. Or SpinRite and live long. He also has lots of free stuff there. You can find out more about SQRL, of course Perfect Paper Passwords and Password Haystacks. And people use his site to generate 64-byte totally random passwords. They use his site to validate the difficulty of unencrypting a password. It's all sorts of great stuff. ShieldsUP! to test your router. GRC.com. You can also go

there and leave feedback or questions: GRC.com/feedback. But the easiest place maybe to reach Steve for that is on his Twitter feed. He allows DMs: @SGgrc. Do you have 280 characters yet, Steve?

Steve: I don't because I'm using TweetDeck.

Leo: Me neither. I think it doesn't matter. I think it will automatically give you 280 when you can get 280. Maybe not. I don't know. But nobody that I know of has it except for Julian Assange for some reason. So @SGgrc. Keep it brief, I guess is the answer, although your DMs can be as long as they want. Now, we also have the podcast on our site. He has it at his site, and not just audio, but also transcripts. We have it on our site, TWiT.tv/sn. We have video, as well. And sometimes the video is worth watching because like the Image of the Day and stuff like that, you can see what Steve's talking about, although he does a great job of describing it.

We also have a subscription form there so you can find a way to subscribe on iTunes, Pocket Casts, Overcast. So that way you get every episode. This is really the one show on our network that doesn't turn to fish wrap by the next week. It's something you want to keep on hand. There's always lots of very useful stuff. Mr. Gibson, we'll be back next Tuesday, 1:30 Pacific, 4:30 Eastern time, 20:30 UTC for Security Now!, and I'll see you then.

Steve: Till then, my friend. Bye.

Copyright (c) 2014 by Steve Gibson and Leo Laporte. SOME RIGHTS RESERVED

This work is licensed for the good of the Internet Community under the Creative Commons License v2.5. See the following Web page for details: http://creativecommons.org/licenses/by-nc-sa/2.5/