Listener Feedback #190

Description: Leo and I discuss the week's major security events and discuss questions and comments from listeners of previous episodes. We tie up loose ends, explore a wide range of topics that are too small to fill their own episode, clarify any confusion from previous installments, and present real world 'application notes' for any of the security technologies and issues we have previously discussed.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-461.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-461-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We've got a look at the latest security news, including BoringSSL, plus answers to 10 of your questions. Security Now! next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 461, recorded Tuesday, June 24, 2014: Your questions, Steve's answers, #190.

It's time for Security Now!, the show that covers your security and privacy online. Ladies and gentlemen, I give you the Explainer in Chief, Mr. Steven "Tiberius" Gibson.

Steve Gibson: Yo, Leo.

Leo: Yo, Steven.

Steve: You know that has filtered into Wikipedia now.

Leo: I know, I love it.

Steve: I'm not sure...

Leo: It's not his real middle name. But it could be.

Steve: Ah, but I'm quite happy, I'm quite happy with having danced with Captain Kirk's cardboard cutout for New Year's Eve. I think I probably earned that.

Leo: Yes. You're known for that now, yeah.

Steve: So we've got Episode 461, which is a Q&A, since we spun everybody's propellers up last week with our episode on authenticated encryption. And sort of an interesting week. We've got the EFF promoting the return of Open Wireless; an acronym, YAFOOSL, which stands for Yet Another Fork of Open SSL; Apple cautiously inching forward with Touch ID; guess who's going to start selling domain names. I'm sure you know already.

Leo: I do know. And I...

Steve: Some positive noises from the legislative front on the NSA and patents, miscellaneous hijinks and updates, and 10 questions from our terrific listeners. So lots of fun this week, I think.

Leo: As always, jammed pack with goodness. All right, Steve. Time for the security news.

Steve: So this is kind of weird. The Register, well, it got some good coverage because this was being promoted by our friends at the Electronic Frontier Foundation, the EFF. And of course TheRegister.co.uk covered it in their typically snarky fashion. The headline was "EFF wants you to open your WiFi to IMPROVE [all caps in their headline] privacy." And then their first little two sentences said, "The Electronic Frontier Foundation (EFF) wants Internet users to go back to the turn of the century" - and I thought, well, wait. Okay, yeah. We have had a century turn not that long ago, so that's appropriate. There was WiFi at the turn of the most recent century - "and open their wireless networks for anyone to connect, in order to enhance privacy. The EFF wants us all to use the Open Wireless initiative's free router firmware, which allows users to create open guest accounts that anyone in range can use."

So I saw these headlines, and I thought, okay, wait a minute. What? What? So, first of all, there is a site, OpenWireless.org. And it wasn't clear from looking at the EFF's site whether they were one of many sponsors. Because there are many sponsors shown at the bottom of the home page at OpenWireless.org. And the EFF is prominent, but there's a bunch. And so checked out the WHATIS listing, or the WHOIS, rather, entry in the Internet WHOIS database. And sure enough, in 2011, so three years ago, OpenWireless.org was registered by the EFF. There's an EFF domain name for its email. And the street address is that of the EFF in San Francisco. So this is really one of their domains.

And so here's their deal. And I'll explain it, and then we need to talk about it because it's sort of definitely has got pros and cons. So the way the EFF describes this, their philosophy is, they said: "Imagine being able to walk around any street in any city and never worrying about checking an email, downloading a map, making a video call, or streaming a song. EFF believes that open wireless networks greatly contribute to the public good. Computer users, worried about privacy or security risks, have largely taken the default route of closing down their networks. Though the willingness to operate in a

secure environment is understandable, the issue is that modern encryption systems make open sharing with friends, family, and passersby very difficult." Yeah. That's the point. Anyway, continuing, they said, "In order to..."

Leo: Isn't this what Comcast did, and we got all mad at them for?

Steve: Well, kind of. And we'll talk about that because...

Leo: Okay.

Steve: But Comcast lets you do this, but you still have to be a Comcast customer. So you have to use your Comcast login to log into somebody else's router, sort of without their knowledge. But, so, yeah. But we'll come back to that.

So they said, "In order to promote beneficial uses of the Internet in all walks of life, EFF and a coalition of organizations are launching the Open Wireless Movement. We are working on new technologies and best practices that will allow individuals, businesses, and community organizations to open up their wireless networks while not sacrificing privacy, security, and quality. Opening one's wireless is a neighborly act that should be supported by router manufacturers, Internet Service Providers, and legal systems. There have been cases where individuals running open networks were wrongfully raided [whoops] for one of their guests' wrong actions, but these cases are highly exceptional.

"We believe that individuals hosting networks should gain the same protections as service providers, especially since Open Wireless services are becoming ubiquitous. More and more cafs, airports, libraries, schools, and individuals happily share their networks with customers and passersby. We encourage Internet Service Providers to not have blanket prohibitions of Open Wireless in their terms of service. And we think business models could be built off of letting customers run open nodes.

"Open Wireless also helps conserve radio spectrum. It turns out that wireless networks, 802.11 [which is WiFi], operate much more effectively and efficiently than cell phone towers. Because wireless systems are connected to a much more distributed system of routers, many more devices can operate on the same frequency. If you wish to find out more about Open Wireless, go to OpenWireless.org. If you are a technologist or company that would like to get involved with the movement, email openwireless@eff.org."

So that's their position. Now, what is interesting is there's some upcoming event. And I didn't write it down, and now I can't remember what it was, but it's like a month away, where the EFF is going to be unveiling replacement firmware for a router, open-source replacement firmware which will have been security audited for a router they're not yet disclosing. So they haven't said what router. But, I mean, it's going to be probably Linksys, I would imagine, since that tends to be the one that most of these alternative firmware platforms run on most easily. And so this initiative is intending to promote the replacement of firmware.

And, I mean, certainly it's the case that we've seen a lot of exploits recently, backdoors discovered in commercial router firmware provided by companies other than the router manufacturer. They've purchased a license for that to install on their routers, and then it turns out there was a little more than we were bargaining for. And thus things like DDWRT and Tomato are popular open-source replacement firmwares for those routers.

So now the EFF is saying, look, let's actively promote this. We want to make this available. Now, what their firmware does is a couple extra things. It creates, now, it's not clear to me whether you can share a single router with a closed network, that is, whether it has a guest facility the way I think pretty much all...

Leo: Comcast does that. Yeah, and all routers, many routers, your modern routers have that, yeah.

Steve: Yeah, I think all modern routers do now. See, and the reason I'm not sure is that somewhere I read them saying get a cheap, get any cheap router and install this. So, as opposed to installing this on the one you already have. So, but I don't know either way. But there is a technology which I was surprised when I dug in that it was as widely available as it is. And that's EAP-TLS. EAP is Extensible Authentication Protocol, which it turns out is sort of hiding in all of our routers. It's really ubiquitous. But it's not sort of the default, de facto standard. In order to get WPA or WPA2 certification, you have to have it in the router. So it's in all of our routers.

And what that allows is, it allows a client certificate to be installed in your phone or your iPad or your laptop, whatever you're going to be connecting to, or your desktop, if you have a wireless connection to your desktop at home. And so what this does is EFF will be making this OpenWireless.org certificate, client certificate, freely available. So everybody installs this. And the matching certificate will be already part of this firmware, this OpenWireless.org firmware that they will be promoting that people install in their routers.

And what this means is that you still get encryption, even though your access point is open. So strangers can connect. There's no portal. They're anti-portal, and they talk about how frankly that's sort of hostile. In their image of this future, lots of people are running this replacement firmware. There's also bandwidth protections so that you're able to say, anybody using the OpenWireless.org - oh, and that's the SSID, by the way.

So, for example, when you're roaming you would - hopefully you'd bring up your list of available WiFi, and there would be a whole bunch of OpenWireless.org SSIDs, and you'd choose the one with the strongest signal strength, for example, and connect to it. No portal, no login terms of service or anything. They don't like that because they're liking the idea, for example, of people with ubiquitous sort of the "Internet of things" devices, watches and phones and so forth, that are just hopping from one of these to the next, using the client certificate to be encrypted. So we're not talking the traditional unencrypted, easily sniffable open hotspot. But that's their intent is to push this.

Oh, and the way they talk about enhancing privacy is they're - and of course we know who the EFF is. I'm glad they're there. They're very pro-privacy, anti-eavesdropping, and civil rights and so forth. Of course we talk about them, as a consequence, all the time. Their argument is we have to further back away from this idea that an IP represents a person. Their argument is IPs are not personally identifiable, and we need to further break that association. If you do have an encrypted access point, if your WiFi is encrypted, then you have a hard time making the argument that any misbehavior which is logged against that IP was not you because it can't have been anybody else.

So they're saying let's use this approach of sharing our IP address to further weaken the association. And that's where this notion of Comcast comes in because the Comcast concept of, I think I read 50,000 routers in Texas, for example, have been replaced by Comcast, and Comcast customers are free to use anybody's Comcast bandwidth as long

as they're able to log into Comcast. Well, inherently that means that Comcast subscribers are no longer able to be held responsible for something happening to their IP.

And in fact the EFF has a PDF, I've got the link. You can track it down. Right now it's on their front page [eff.org/files/2014/05/28/open-wifi-copyright_0.pdf]. It's in the show notes this week, and I tweeted it earlier today, so there's a number of ways people can find it. Anyway, it's sort of their position on the issue of WiFi and copyright and how, in the same way that ISPs themselves can never be held liable for the actions of their users. Similarly, access point operators who are freely making bandwidth available can similarly, just like an ISP, they are essentially a service provider. They're harmless against the actions that people may take using the bandwidth which is streaming through them but was not actually their action.

So anyway, I don't know what'll come of it. But it was certainly an interesting, I mean, it got picked up in the news, and I'm sure in a couple weeks we'll be mentioning that, whoops, now we know what the firmware is for, and people can download it. And, I mean, I know that there are people who will just like the idea of running, you know, of offering bandwidth. And this arguably is somewhat more burdensome because of the need to use - I don't know whether it'll be open and available encrypted, that is, whether you'll be able to access it without the certificate. I would imagine you can. The bandwidth is both available without a client-side certificate. But if you want encryption, then you can use the client-side certificate.

However, this also means that, even though you're encrypted, you don't have super security. First of all, if you're using anybody else's hotspot, then you don't have super security because, even if they weren't doing a man-in-the-middle attack, which is possible even with the certificate because we're going to all know what the router certificate is because it's open source, it'll be sitting in an open-source repository in order to be built into the firmware. So that means that somebody else could set up a fake hotspot with the OpenWireless.org keying and still be able to intercept your bandwidth.

So anyone using anyone else's bandwidth needs to understand that they need to be protecting their traffic themselves with HTTPS tunneling within the connection. But at least its not wide open. So anyway, just sort of an interesting initiative. It will be fun to see where this goes. It's funny, I remember bringing up a list of, at the turn of the century, bringing up a list of wireless hotspots within range of where I am sitting right now. And they were all unencrypted back in the day. Now there isn't a single unencrypted wireless hotspot around within my residential community. They've all got the little padlock showing. And it is the case that you go to restaurants, and sometimes they're making WiFi available; or sometimes the restaurant will have wireless, but you need to ask them for what the password is. So what do you think?

Leo: Well, don't we - didn't we - part of the concern we had with Comcast was the security issue.

Steve: Yup.

Leo: That there would be some leakage or some unknown exploit that would cause your local area network to be visible to somebody as a guest. So that would be - that's the same; right?

Steve: Yeah. But absolutely you need to make sure that, for example, in the way that the guest systems are set up now in contemporary routers, you're able to say whether or not you want the guest account to have visibility into your LAN and/or to be able to bridge over to your other wireless side.

Leo: Well, that's another thing I didn't think of is the security of the person using the access point. You might be exposing your laptop to the owner; and, if he were nefarious, that would be bad. So I think this issue's there.

Steve: Yeah, yeah.

Leo: Otherwise I think it's a great idea. Fon, F-o-n, has been doing this for some time.

Steve: Yes, yes.

Leo: In fact, when they were first started as a company...

Steve: The Spanish telecommunications?

Leo: Yeah, yeah. And when the Fon routers first came out, the whole premise was to create a mesh of shared Internet access. And they, actually, no, I was thinking of Meraki. Fon does it. But the Meraki routers, that was the idea. Meraki ended up getting sold to Cisco and is not doing that anymore. They went enterprise. But I think that this is not a new idea. I like the idea. I just don't know how it's different. I guess the biggest difference from Comcast is you don't have to be a Comcast customer. It's just you're sharing it with the world.

Steve: I like the idea, too, that the EFF is behind it. They're clearly good guys. I mean, if there's anyone I'm going to trust, it's them. They're saying this will be fully security audited. So again, where we have people saying, well, you know, I don't really understand DDWRT or Tomato, or I'm not comfortable with it, well, here's something you could really, I think, be comfortable with. We'll have to take a look at it and see what the features are, how it operates, and really give it a good vetting. But I'm with you. I do, I really do like the idea of creating ubiquitous WiFi and a mesh, but in a way that does not compromise the security of the owners.

Now, one issue that you heard me mention in the EFF dialogue is that the terms, the current terms of service of many, in fact most big ones, they have a list on OpenWireless.org showing ISPs that don't do this, and they're not major ISPs. The typical default terms of service says you cannot knowingly, willingly share the bandwidth you're buying from us with other people. That is, it's for your use, your household's use. You can't make it broadly available. So that's where, I mean, the EFF understands that. And we're going to need some concession from ISPs backing down from that and saying, okay, fine, as long as you do it in a responsible way, we'll allow you to. So the point being that doing it deliberately right now would be a technical violation of terms of service.

Leo: That's what advantage Comcast has. They are the ISP.

Steve: Right.

Leo: So they set the terms of service, yeah. Right, good.

Steve: Yeah.

Leo: It's an admirable idea.

Steve: Yeah. And let's hope it happens. I mean, it would be, you know, I don't know how big of a movement it is at the moment. I think...

Leo: Well, you know, the whole point for Comcast is it's in effect their entry into wireless data; right?

Steve: Yes.

Leo: If they can blanket, and they have in many towns, blanket that town with Comcast access points, now they can do a phone service. There's all sorts of stuff they can do. And so there's a commercial incentive that the EFF lacks. Their only incentive is altruism.

Steve: Yeah, truly.

Leo: And I think usually greed trumps altruism. Sorry to say.

Steve: Yeah. So I did note that there was some news leak from China about the scale of Apple's orders for the Touch ID sensor, which has led those who track these things to predict that, when we get the next generation of Apple devices later here coming up in 2014, all of them are going to have Touch ID - the new iPhone 6's, the new iPad Air, and the iPad Mini. And someone who's working on a SQRL client took a look at the API that Apple just opened up because this was one of the things that we learned about during the iOS 8 was that the Touch ID API was going to be opened up.

And so what they did was very simple. It's just a binary go/no-go indication. Applications are able to request a real-time prompt from iOS to confirm the user's Touch ID identity. That is, that whoever's holding the phone has registered that fingerprint among those that are known by the phone, that are authorized to use the phone. So unfortunately, I don't think you get, like, who among the users. There's no, as it was described to me, it was just binary. It was yes or no. Touch ID just succeeded. So they're keeping it very simple and moving it forward slowly.

But, for example, in the case of at least this SQRL client, you'll be able to use that in order to reauthenticate that you're still the owner who wants to use SQRL to log into a website. So I'm glad for that level of functionality. And it's nice to see Apple creeping forward. I mean, they've proven it now with the 5s, and they're going to release it more broadly.

Well, Adam Langley has forked OpenSSL.

Leo: Oh, interesting.

Steve: Yeah. He posted...

Leo: And so has Google; right?

Steve: Yeah, well, I mean, yeah, Google. Adam Langley is Google.

Leo: Adam Langley at Google. But is he doing it as Adam Langley? He's doing it as Google.

Steve: Oh, yeah. He's doing it as Google, yeah, yeah, yeah.

Leo: Oh, okay.

Steve: Yeah, sorry.

Leo: It's more relevant to people if you say "Google has forked SSL" than Adam Langley has. So let's be clear what's going on here, yeah.

Steve: Yes, yes. Google's Adam Langley...

Leo: There you go. I like that.

Steve: Google's Adam Langley has forked OpenSSL. Of course, famously, SSL was forked, and we talked about that, by this LibreSSL group that have decided they're going to strip it out and clean it up and reduce its size. And there's been all kinds of noise spinning off of that. If you look at the link, Leo, at the bottom of that page, he's calling it - he Google - yeah, there's a link.

Leo: The source of the confusion is this is a personal blog called ImperialViolet. And I wish Google would, I don't know, do this more officially. You know what I'm saying?

Steve: Yeah. So here's what Adam said. He said, "Earlier this year, before Apple had too many goto fails and GnuTLS had too few, before everyone learnt [he wrote] that TLS heartbeat messages were a thing and that some bugs are really old, I started a tidy-up" - "I," says Adam Langley writing for Google, I mean, this is, you know, Adam Langley is Google, he's their security guy - "I started a tidy-up of the OpenSSL code that we use at Google. We have used a number of patches on top of OpenSSL for many years. Some of them have been accepted into the main OpenSSL repository, but many of them don't mesh with OpenSSL's guarantee of API and ABI stability."

The API is the Application Programming Interface, where you have ways of creating code that can talk to OpenSSL. ABI is the Application Binary interface where you would have binary interoperability, not needing to compile these things together.

So he says either API or ABI stability, "and many of them are a little too experimental. But as Android, Chrome, and other products have started to need some subset of these patches, things have grown very complex." And I can't even imagine how complex. "The effort involved in keeping all these patches, and there are more than 70 at the moment [seven zero], straight across multiple code bases is getting to be too much. So we're switching models to one where we import changes from OpenSSL rather than

rebasing on top of them." And I'll explain that in a second. "The result of that will start to appear in the Chromium repository soon; and, over time, we hope to use it in Android and internally within Google, too.

"There are no guarantees of API or ABI stability with this code. We are not aiming to replace OpenSSL as an open-source project. We will still be sending them bug fixes when we find them and will be importing changes from upstream. Also, we will still be funding the Core Infrastructure Initiative and the OpenBSD Foundation. But we'll also be able to import changes from LibreSSL, and they are welcome to take changes from us." And then he calls this "BoringSSL."

Leo: I love that.

Steve: And he said, "The name is aspirational and not yet a promise." Meaning he hopes it's boring. So essentially what this says is that they've been trying, they've sort of been holding on, trying to use OpenSSL, trying to continue using it to gain the benefit of the OpenSSL ongoing work; yet they've got 70 significant changes which their stuff needs, which really has been diverging the target of OpenSSL. And it's just gotten to be too much. So they're now - they've essentially taken a snapshot of the current OpenSSL codebase, and they have forked it. They've got their own set of files.

And this link, it's boringssl.googlesource.com, and what I have is /boringssl. This is the changelog. And, oh, my goodness. I mean, it does show this is where Adam is spending his time. It starts four days ago, on Friday when he made this posting. And it's hundreds of changes to the code. I mean, he's just going through it, fixing things and changing things. So I think they're being gentle in sort of saying they're going to feed things back. I mean, I'm sure there will continue to be cross-pollination. But basically what this means is Google has just decided to fork the project, to go their own way. Certainly they'll be interested to see what changes OpenSSL makes and then to import those changes into their code.

What they have been doing is this rebasing that he talks of, is maintaining 70 patches which they're continually reapplying whenever the OpenSSL code changes and they need

to, like, repatch it in order to create their own version of it. And they've just decided, no, okay, we're going to stop that. So that means that finally those 70 patches, I'm sure they've already been applied when the fork was made, now they've got their own stable codebase which will now diverge. I mean, we know about SPDY. We've talked about that. I've got on my list of things to talk about QUIC, Q-U-I-C, which is another one of these experiments which is really interesting looking. It is UDP secure HTTP protocol. Instead of HTTP being on top of TCP, this is on UDP, with the significant benefits in speed that that means. And that's what Adam talks about when he talks about how Google wants to be doing experimental things. I mean, I'm surprised they lasted this long, frankly.

So the world now has something called BoringSSL - open, public, and going to sort of be a development platform for Google's own moving forward security initiatives. And frankly, I wouldn't be surprised if this is the way Google ends up solving the problem that they've got with certificate revocation because what they took from OpenSSL has all of that in there. That's all that code is there. So this is probably their way of getting control of it and then doing with it from now on what they want. And so we now have OpenSSL, LibreSSL, and BoringSSL.

Leo: Good.

Steve: Which actually I don't think is going to be boring at all. Boring from a standpoint of hopefully we won't be melting down with security problems. But it's going to have, it's going to be the platform for all of these initiatives that Google will be experimenting with, which I think is neat.

Leo: I think it's a little odd that Adam Langley has such a cult of personality within Google. I mean, surely he's not the only guy writing security code at Google.

Steve: No.

Leo: It just bugs me a little bit.

Steve: I do understand, Leo. I do.

Leo: To have a cult of personality, it's like...

Steve: I do understand. So speaking of Google, domains.google.com. Google is going to become a domain registrar. They're in beta now. It's invitation only. You can request an invitation. And I don't know how many people have, or care. But you can get one. So in their /about page, they said, "A domain name, your address on the Internet, says a lot about who you are and what you do. New domain endings like .guru and .photography can help you find a meaningful address that stands out on the web. Every domain includes easy forwarding, branded email (you@your_company.com), simple management tools and other helpful features." So they've got a few interesting things. So no additional cost for private registration. That's the one that really perked my interest.

Leo: Hover does that. Our advertiser does that.

Steve: Oh, good. I, believe me...

Leo: I think that's de rigueur. The only people that don't do it, GoDaddy because they want to...

Steve: Yeah, is my registrar.

Leo: Who's your registrar?

Steve: I'm still at Network Solutions.

Leo: Oh, that's old school, man. All my original domains were there because that was the domain registrar; right?

Steve: Yup. They were.

Leo: They were the guys.

Steve: And they're just - it takes me maybe 10 times longer, I'm not exaggerating, 10 times longer to register a new domain name because I have to say no to so many things that they're trying...

Leo: Steve, Hover.com. Seriously. I moved everything over there.

Steve: Yeah, but I can't risk it being lost. I mean, I don't know why it would be. But I'm just, you know, I'm a Nervous Nellie about GRC.com ever, like, being stolen from me.

Leo: There's a pretty solid transfer process that is overseen by ICANN. I think it's safe.

Steve: Yeah.

Leo: There's a whole thing, you know, you have a code, there's a whole thing you have to do to transfer a domain name.

Steve: Well, but, like if someone broke into my Network Solutions account, or Network Solutions had a breach, then they could authorize the transfer of my domain, and that

would be bad. I don't want that.

Leo: Well, so, yeah. Oh, you do - I don't understand. That's how it is now; right?

Steve: Yes. And so I'm hoping that Network Solutions has really good, old-school, strong security.

Leo: Ah. I see. I see.

Steve: And believe me, I've got a password from hell, so no one's going to guess that. So, I mean, I've protected myself every way I can, with all kinds of verifications and email addresses that I never use for anything else and so forth. So I don't want...

Leo: There's a couple of questions about the Google registry, though.

Steve: Okay.

Leo: Well, for one thing, they've been doing this for years internally. So they have been a registrar, official registrar with ICANN for, I think, 10 years.

Steve: Oh, interesting. I didn't know that.

Leo: They're just now opening this to the public. But they have one thing that puzzles me. They limit you to a certain number of DNS requests per year. Now, I don't know if Network...

Steve: Whoa. No kidding.

Leo: It's a large number. I think it's a - I can't remember what it is, a million or something like that. But that is - I've never seen any registrar say that. Maybe that's the case. I don't know. That seems odd. Now, it's a large number, and remember DNS requests wouldn't - they'll be cached, so...

Steve: Yeah. So the only trouble, the only reason I could ever see that being a problem would be if you were deliberately using extremely short TTLs, Times To Live, on your DNS entries. And the only reason you would do that would be if you were needing for some reason to change your IP addresses frequently or be making changes that you didn't want to have cached for a long time. Back in the old days when we were being attacked, when I was with Verio, I had a relatively short TTL of a couple hours, which would force all the Internet server, all the other DNS servers to come back every couple hours and see whether GRC's IP address had changed. Things have been much quieter lately, and so we've gone back to a more normal TTL. So, I mean, I guess I can - I don't really see that being a problem. But that is an interesting caveat.

Leo: Kind of odd, yeah.

Steve: I guess maybe they're being more - they're just being honest. Instead of saying, oh, unlimited bandwidth, they're saying, well, we have a very high limit on the number of requests you make. But I do remember seeing other commercial providers of DNS doing that. So that's also saying they're not just a registrar, they're also serving the DNS. They're being the DNS server of the domains they register. For example, I use Network Solutions as my registrar, but I run my own DNS servers. So I'm not using Network Solutions as my DNS, only to register the domains.

Leo: You get a hundred email aliases. You get a customized - a hundred subdomains.

Steve: A hundred subdomains, yup.

Leo: That's like images.grc.com, those are the subdomains.

Steve: Yup, yup.

Leo: And they integrate with Squarespace, which we like. They're going to integrate with some of our partners.

Steve: Yup. But they say...

Leo: That limitation is the only thing that puzzled me, and I wanted to ask you about that.

Steve: Yeah. I don't see it as any problem for anyone. And also I'm curious, too. They said availability of new domain endings. The problem with things like .guru and .photography is that they need to be supported by the DNS server, like by the root DNS servers, or by servers that your server knows to query. And this is why, like, .com and .edu and .org and .net, they all are. But these sort of off-brand top-level domains just don't have wide coverage yet. So it's nice that you can buy them, but you need to be able to have them supported. Now, I guess the point is that Google would be doing that.

Leo: Right. You're using Google's DNS servers, so that's nice.

Steve: Right, and you get their infrastructure.

Leo: Yeah, I guess the other thing is do you want - how much of your eggs do you want to put in the Google basket; right?

Steve: Exactly, how much? Do we need Google to be doing anything more for us?

Leo: Yeah.

Steve: Spreadsheets and email and my handheld device and my cloud and my, like...

Leo: Google goes down, the world goes down.

Steve: Yeah. So, okay. Two legislative bits. The NSA was dealt a blow in the House of Representatives last week, last Thursday. They voted overwhelmingly, in a bipartisan vote, on both sides of the aisle, 293 in favor, 123 against, so way more than two thirds, to prevent the Defense Appropriations Bill from funding NSA backdoors and backdoor searches. Essentially, they passed an amendment to the Defense Appropriations Bill which cuts funding for NSA backdoors and backdoor searches.

As it is now, the NSA collects emails, browsing, and chat history, as we know, under Section 702 of FISA, which we talked about back when the whole Edward Snowden revelations began to happen. And then they are able to search through all of that aggregated communications of Americans without warrant. So this practice has become known as "backdoor searches." The amendment which passed would block the NSA from using any of its funding from this Defense Appropriations Bill to conduct those warrantless searches of the data that they collect.

Secondly, the amendment prohibits the NSA from using its budget to mandate or request that private companies and organizations add backdoors to the encryption standards that are meant to protect users' privacy on the web. Now, it is true that this controls the purse strings, and purse strings are important. But this is certainly different from legislation saying they are not able to do that. This says they can't use this money to fund that. So I'm not close enough to this to understand what real impact this has. It still, of course, has to get passed by the Senate, and then the President has to sign it into law. But given a vote this strong, I will be very surprised if this doesn't happen. So that was good.

And we got some nice motion, although less useful than many watchers of this were hoping, on the patent front. There was an argument, it's about a year old, by an Australia-based bank called Alice Corporation that sued a bank called CLS over CLS's use of a patent which Alice Corp. had on how to use a computer for escrowing. That is, essentially, how to use software to run a third-party escrow. And CLS said, "This patent is bogus." And Alice Corp. said, "No, it's not. We've got a patent, and we're enforcing it."

So what we got last Thursday was a ruling from the Supreme Court agreeing that simply running a concept through a computer doesn't itself merit a patent. That is, Alice Corp. didn't invent anything. They just said, okay, here's how you do escrowing with a computer. And so the people who are worried about the freedom with which patents are being granted for software were hoping for a much stronger statement from the Supreme Court. Unfortunately, we got something not very useful for anything else.

Clarence Thomas wrote the decision. It was a unanimous decision, so not even controversial on the Court's side. And he wrote, in his rendering of a judgment, he explicitly said that this didn't need to deal with the idea, the question of abstract ideas, which in his writing he called the "abstract-idea problem," because the facts of the case

were so obviously against the company with the bad patents.

So essentially this made a statement about software patents that are simply embodiments of real-world things implemented in the software, and so that's good, as far as it goes. To the degree that there are any other of those out there, they will now no longer stand. Anybody who's fighting what they feel is a patent in software that just simply states, oh, we did this with a computer, those patents will be easily overturned based on the rendering of this decision from the Supreme Court. But everyone who's watching patent law was hoping for more than this.

And then, okay. So I really got a kick out of this. I have a follower, @rothgar, who tweeted me. He said, "Steve," or he said, "@SGgrc A cool story about Google's use of the 95/5 rule." And he tweeted me a link which was to page 187 of Steven Levy's book, "In the Plex: How Google Thinks, Works, and Shapes Our Lives."

Now, remember that I explained two weeks ago, in answer to a Q&A, exactly how 95/5 operates, where over the course of a month, five-minute intervals are sampled. So all of the bytes that you use, like a server farm user uses, are totaled in five-minute intervals. And all of those five-minute intervals are then collected and sorted. So you'll end up with, naturally, a maximum to minimum sorting of five-minute intervals. Then, of however many there are, five-minute intervals in that billing period, the top 5% are discarded completely, and that next interval is regarded as the amount of bandwidth you were using all month for all five minutes of the month, and you're billed on that.

Well, if you think about it, and, I mean, all of we server users have thought about it, you're really kind of getting away with peaks in usage that don't show at all. So what Steven Levy wrote was: "Back in 2000" - so this is a time machine; this is 14 years ago when Google was way smaller - "Google wanted to get speedier by setting up data centers in locations closer to its users. Its first priority was getting servers on the East Coast of the United States." Meaning that this was in a time when Google was still just sitting there in Silicon Valley.

"By the spring of that year, Google was occupying space in a colo in Northern Virginia. The tricky part of setting up a new facility was loading all those thousands of servers with the indexes. That involved moving terabytes of data, which was potentially going to force Google to pay a huge amount of money to the bandwidth provider that owned the fiber. 'Networking was very expensive,'" quoting someone named Holz, or Holzle [Urs Hlzle], I guess, "'and our data push would take 20 hours at a gigabyte per second. That would cost us something like a quarter million dollars a month.'

"To save money, Google devised a trick that exploited a loophole in the billing system known as the 95th Percentile Rule," writes Steve Levy. "Over the period of a month, the provider would measure how much information was moving, automatically taking a measurement every five minutes. In order to discard unusual spikes in activity" - is the way Levy explains it, although of course we covered it differently two weeks ago - "when the billing rate was calculated, the provider would lop off the measurements in the top five percentiles and bill the customer at the rate of the 95th percentile.

"Google's exploitation of the rule was like the correct answer to a trick question in one of its hiring interviews. It decided to squeeze the movement of all of its information into those discounted spikes. 'We figured out that, if we used zero bandwidth all month, except for 30 hours once a month, we would be under that 5%,'" says someone named Reese. "For two nights a month, from 6:00 p.m. to 6:00 a.m. Pacific time, Google moved all the data in its indexes from West to East. 'We would push as fast as we could, and that would cause massive traffic to go across, but it was during the lull hours for them.

And of course the bill came out to zero,' says Reese, 'because they then lopped off the top 5% percent, and our remaining bandwidth was in fact zero because we didn't use any otherwise.' He says, 'I literally turned off the router ports for 28 or 29 days per month.'" So there's a hack for you. Google moved all of their data across the United States...

Leo: Without using any bandwidth.

Steve: In two 15-hour periods, essentially, to squeeze it into the 95th percentile and become completely transparent in terms of their usage. Wow.

Leo: Now, that's an optimization.

Steve: That's a hack. Oh, wow. So a piece of errata from Adam Ross. I was talking about GHash.io. And he sent me a tweet, so thank you, Adam, saying, "A small point which makes a big difference. GHash.io is a commercial entity that rents hash time on their equipment, not a pool." I was referring to it last week as a mining pool, which I though it was actually from things that I read on their own website, the way they were talking about measuring their size. But apparently they're just hardware. And so miners, when they said they had 55% of miners, well, I guess that's people who are renting time on their hardware. So it does really give them more central control to the degree that they're, like, over 51% of total hashing power of Bitcoin. That is, to my mind, a little more frightening, that this is one facility with all these incredible ASICs cranking away.

And I got a note from someone in Argentinia or, as they call it, Argentina. Federico Bett said, "Just writing to let you know that you have some listeners from Argentina. Keep up the good work."

Leo: Great.

Steve: And I tweeted a bit of news from TechCrunch that surprised my listeners. I had said to you, I guess off the air, Leo, that I am a big fan of Chelsea Handler, and I was sorry that her show was ending in August. But it turns out she's moving to Netflix. So TechCrunch provided the news. They said: "Netflix has just announced an exclusive deal with the beautiful and brilliant Chelsea Handler for an upcoming series of talk show programming," blah blah blah. So that's where she's going. I'm glad I'm not losing her. I just get a big kick out of her. I mean, I will say she is certainly unfiltered in the things she says.

Leo: Yeah. Yeah.

Steve: So, yeah. "The Last Ship" aired on Sunday night. I mentioned it, I think, a week ago. And it was great. So I'm a sort of a sucker for the epidemic movies, "Outbreak" and so forth. And this is on TNT. So I recommend it without reservation. Hopefully they'll be able to keep it up. But the first episode was great. I tweeted out to my followers about 25 minutes in, it's like, oh, my god, if you missed recording this, record it because it was great. It's being re-aired tomorrow and Saturday, and even I think again on Sunday before the second episode on, again, later on Sunday. So I thought it was great.

I'm continuing to work on SQRL, moving forward. We have accumulated 405 translators and 57 languages. So as soon as I get the first release done, I will put up all of the text on the Crowdin.net site. Anybody who's interested in participating can go to translate.grc.com. That's just a redirect over to the Crowdin.net site, but it puts you right to the SQRL project, and add yourself to the community. So I'm excited to have that moving forward.

I did get - I didn't know what this meant when I saw the subject line. Sean Zicari wrote: "SpinRite success story - recovery is a dish best served cold." I thought, okay.

Leo: That's a takeoff on the line "Revenge is a dish best served cold."

Steve: Of course. I mean, very clever. "Recovery is a dish best served cold."

Leo: He's not going to put it in a freezer, is he?

Steve: Uh-huh. "Hi, Steve. I wanted" - good one, Leo. "I wanted to share a success story through unexpected means. First off, I've been listening to the podcast for a couple of years now and am a big fan, blah blah blah, in honor of how Leo normally reads those sentences. Seriously, though, I love the podcast and really appreciate your expertise.

"My friend called and asked advice about recovering data from the hard drive in his PowerMac G4. He's had the computer for years and has never performed maintenance on it to my knowledge. He said the computer would no longer boot, so he took it to a drive recovery specialist. After recovering from the quoted price, he called me for a second opinion. I thought this was the perfect chance to try SpinRite, so I said I would fix the drive for him for the cost of a SpinRite license - win-win. He agreed and mailed the drive to me.

"I ran SpinRite on Level 2 first. That completed successfully. I rebooted and attempted to read some data from the drive. I was now able to, but it took a long time to bring up files and folders. Specifically, the drive would make a repetitive scanning noise of some sort, not sure how to describe it, but guessing you know what I'm talking about." Oh, yeah.

Leo: [Mimicking drive].

Steve: Yes. Or sometimes it's [mimicking drive].

Leo: Yeah, that one sounds like something's wrong, yeah.

Steve: Yeah, that's not good. And that is the head going out and being unable to find the sector it's looking for, then thinking maybe it got lost and so retracting and then doing what's called a recalibrate and then going back out again. So...

Leo: Ooh, that's not good.

Steve: Not good. He said, "I decided to run a Level 4 scan next, but throughout the process the SpinRite UI was very slow to respond and froze frequently for long periods of time while the drive made that same repetitive scanning noise mentioned previously. At .12% the UI froze entirely, and the drive went silent." So everything just locked up. He said, "Reading through the SpinRite FAQ for ideas, and with a very pessimistic eye on the situation, I decided to try the last suggestion first. I put the drive in the fridge for an hour, popped it back in the computer, and started a Level 4 scan again. The scan process hit that same .12%, and the UI started to move sluggishly, but overall not as bad as before.

"I shut the monitor off and let it go. I was on the way out the door, anyway. When I came back four hours later, the scan process was half complete and moving along steadily. It finished late last night. I'm happy to say the drive is totally quiet again, and the data seems to be intact. There were a couple of I/O errors in my friend's Home folder, which I'm guessing may be related to trying to read an HFS+ volume in Linux, but it looked like all the important data was accessible. Thanks again for a fantastic product."

So this one really did, I mean, this was really way out on the verge of past its life expectancy. But thanks to cooling it off, which surprisingly can function, as he says, recovery is a dish best served cold. And he got his data back.

Leo: Send the IRS a copy of SpinRite. I think they're having a little trouble with hard drives.

Steve: I've had some people tweeting, has Darrell Issa been in touch with you?

Leo: All right, Steve. I've got questions. None of them come from me. They all come from your viewers, who go to GRC.com/feedback.

Steve: Feedback.

Leo: And people sometimes say, well, does Steve go into chat? Because I have to ask him a question. And I always point out that Steve likes to answer correctly. Steve takes great pains, in fact, to research his answers. So he doesn't want to answer off the cuff. So I don't want to put words in your mouth, but I feel like that's why you do it this way, so you can really get it down, get it right.

Steve: Well, yeah. And what I do also is I'll see many questions about the same thing, and I'll find, like, one that's the most representative. And so like I try to find things that are, like, in general, people are asking. I think that it's very easy to have somebody say - and I don't know how you manage to do this on your show, but I've watched you do it, Leo. "My Belkin RB27Q, I can't find the reset button." And you know where it is. But I don't know where it is. So...

Leo: But it's easy enough to find. I'm just - I can Google and talk at the same time. That's really how - that's my secret.

Steve: Yeah. And I need to stop talking when I walk, so, yeah.

Leo: Question 1 comes from Tennessee. Reid is on the line. He says: Steve, I work at a boarding school, and I've been setting up a transparent proxy server. It's a high school boarding students, so kind of important that I block questionable websites. I figured out how to block non-encrypted connections fairly easily, but I am not sure how to block the HTTPS connections without doing a man-in-the-middle attack. Now, of course HTTPS is designed to fight this, not to mention the ethical implications of intercepting such packets. I don't know where to go next. Do you have any suggestions?

Steve: So, yeah. I thought about this. And the only need, the only reason, from a technology standpoint, to intercept, to do a man-in-the-middle attack, to intercept HTTPS connections, is if you care about the contents of the traffic. The reason corporations do it, the reasons large facilities do it, is they're trying to protect their network from viruses which could get in over secure connections. So HTTP would normally erect an encrypted tunnel that would prevent someone from seeing inside.

But if what you really want to do is questionable websites, then DNS and IPs provide you all the protection you need. You don't need to see inside the traffic itself. You just need to say the following website domain names and/or IP addresses are blacklisted. So, and also, the only way to successfully filter HTTPS is to force everyone to have a certificate. And with that comes a lot of responsibility. So it's just sort of better to stay away from that and use DNS and IP blocking that can be done outside of HTTPS at the TCP level, just to keep people away from sites you don't want them to visit. For this kind of thing it just really seems like cracking into their connections is more than you want to do, and more than you need to do.

Leo: Because you're just blocking the site. You don't have to block the contents. Now, the one thing he's worried about is that somebody will use, say, an Open VPN, that you use proXPN, for instance. And then he can't see any of the traffic at all. But he could block proxies, I guess.

Steve: You know, there's just no way to, I mean, a VPN bypasses the block. So there's no way to get into a VPN connection. You'd have to, I mean, there are ISPs, for example, that prevent you from using Tor. And I've heard that there are some, for example, a gateway like Starbucks has might be VPN-hostile, although normally you can get around that by using a TCP VPN, you know, through HTTPS, and look like regular HTTP traffic, even though you're actually VPNing. So there are generally ways around that. But, yeah, you can do what you can. But there's a lot you can do without having to crack into the connection.

Leo: And just one other thing, Reid. To make it easy on you, a lot of schools use OpenDNS.com. They have a school system. Let them do it through DNS is often so

much easier and more effective than trying to do it all by yourself. It sounds like he's doing it by hand, which is a challenge.

Steve: Right. And then just block DNS except for OpenDNS.

Leo: Right, right.

Steve: That prevents anybody from using other DNS servers. And if you force them to use OpenDNS, then you get OpenDNS's protection. I think that's a great, great idea, Leo.

Leo: Yeah. Question 2 comes from Scott Doyle, Brisbane, Australia. He's been thinking about SQRL and just personal identity security in general: Steve and Leo, do you think it's possible to apply SQRL technology and methodologies to real-world personal identification data? We hear all the time these days of people's identities being stolen. Could we then simply "revoke" a stolen identity and get on with our lives? See, that's the problem with, you know, in the U.S., and I imagine every country does something like this, we have Social Security numbers which cannot be changed. They're yours for life. But a token that represents your SSN, now, that would be interesting. Long-time listener, since Day 1. First-time caller. Keep up the great work. The Sugar Hill podcasts were a revelation. The whole world should be made to listen to them.

Steve: So, okay. Just real quickly on this one, the way to think of SQRL, I think the easiest way to think of it is that it's a means of saying to a website, I'm back. That's really all it is. That's what it does. So it's anonymous. And when I'm writing documentation, I talk about an identity association, meaning that you associate your SQRL identity at a website. So that's what it is. I mean, it isn't more than that, but it isn't less than that. That is, it does that one thing, I think, perfectly. So that if you, like, create an identity like on a blogging site, then when you come back, the site knows it's you. You're the same entity that was there before. That's all SQRL does, is with cryptographic purity it says this same entity came back.

Now, if while you're there you create an account that's got real-world data, then that will be associated with your SQRL identity. But that's not necessary. But it's likely, for example, as soon as I get it, like the moment Amazon supports SQRL, if they ever do, I will associate my SQRL identity at Amazon with my Amazon account. So I can then use SQRL to log into my Amazon account. And that then creates that association.

So it is certainly the case that, if SQRL caught on, there could be services, for example, that bind your real-world identity to your SQRL identity. And SQRL provides a mechanism for securely changing that, or getting it back if it's stolen. There's complete control over that. So that could be done as a service. But that's really sort of outside of SQRL's scope. All SQRL does is it's able to cryptographically anonymize you uniquely on every website and say, when you come back, it's me again.

Leo: I do think, though, that it would be nice to have some sort of authentication that could be revoked for your real identity, for your personal identity. Be an

interesting problem.

Steve: An identity that could be revoked.

Leo: See, that's the issue. You can't revoke your Social Security number. So...

Steve: Right. Well, and we have a problem with fingerprints, too. You can't revoke your fingerprints.

Leo: Your birth date's always your birth date. Oh, I guess you could change that. But if you created, if you had these identities that you could revoke, I don't know, it seems like, you know what, get to work on that, would you?

Question 3 from Dave Redekop. Hey, Dave. Nerds On Site, remember him?

Steve: Yeah.

Leo: From London, Ontario, Canada: Steve, your podcast has been instrumental in helping our thinking around what security should be. We're always grateful, and every week we look forward to your podcast.

You may be aware of the "Reset the Net" campaign, which has been about adopting more secure methods, protocols, and apps for our collective benefit, especially those of us who wish to preserve our remaining civil liberties. In our business - Nerds On Site is a support business. They're kind of a system for independent support professionals and IT professionals.

Steve: And very tech savvy.

Leo: Yeah. In our business we're constantly coming across new victims of cybercrime. And in thinking about it one day we realized that if we - the industry, really - simply followed your advice of HTTPS-only everywhere, we would have stopped every single financial cybercrime we've ever been called in for on a post-mortem. Unfortunately, we were never able to share much event detail with the public. Suffice to say, if secure HTTP had been the only allowed protocol, not a single one of the crimes would have been possible. So why not simply block all non-secure HTTP traffic, attempt to automatically redirect it to HTTPS to minimize inconvenience? I'm asking rhetorically, of course. Thanks to you and Leo for all your efforts each week to put together a world-class podcast. Thank you.

Steve: So this was a great question. And I see it in various forms in the Security Now! email all the time. So as I said, this is sort of my process of sorting through things and finding things that everyone is curious about, or at least more than one person. Certainly it's the case that more and more sites are supporting HTTPS. But unfortunately, even now, it is really the case that the majority of sites still don't. It's not free. You need to

have a security certificate on your server. It's a little tricky when you're in a multiple domain hosting environment because now clients support something called DNI, a Domain Name Indication, where they're able in the original connection to specify, in the initial HTTPS handshake, they can specify the domain name there that they're wishing to connect to. That's important. Otherwise, every single domain name has to have its own IP, which isn't available in a shared hosting environment where there's generally some IP constraint.

So, I mean, there are still too many barriers to HTTPS. It's getting better all the time. And one of the things that is nice is, when using HTTPS everywhere, you sort of opportunistically try to connect with HTTPS. And if it fails, you back down. You know, many sites do support HTTPS, but the URLs you click on, for example, in Google are not HTTPS. They're just HTTP. So you could be having a secure conversation. They've got the certificate. They've made all the services available. But normally they only switch you into that mode when you're going to log in or do something that is security critical, rather than just staying there all the time.

So this is a perfect example of slow evolution in our understanding of the importance of security on the Internet. I know we'll get there someday. I mean, there will always be sites that are HTTP only. But I think that'll end up at some point being the exception rather than the rule.

Leo: Well, I can't believe that a bank or financial institution wouldn't be. David Twiss in Adelaide, Australia. He isn't so sure about iOS 8 and randomizing MAC addresses: You mentioned that iOS 8 will randomize MAC addresses, said that's probably not going to break anything. I want to let you know there are some retail analytics services that will be broken by this change. Aerohive, Euclid, and RetailNext offer commercial services that rely on unique WiFi MAC addresses. These services use features of enterprise WLAN vendors including Aerohive, Aruba, Meraki, Ruckus, and Fortinet to sniff the MAC addresses of passing WiFi devices. Well, that's exactly what we're trying to block here.

Steve: Exactly.

Leo: You know, it doesn't spoof the MAC address when you log in, only if you're just passing through. There are also several museums that use homegrown systems based on this technology to track visitor volumes among their exhibits. The commercial services I have looked into hash the MAC addresses within each WLAN access point to help preserve some measure of user privacy, reporting the hashed MAC address to the cloud-based analytics service. In any event, the effectiveness of these services will - yes, yes, this is the point, David - will be negatively affected by iOS 8 randomizing MAC addresses. Keep up the great work. David Twiss, Adelaide, Australia.

Steve: So I just - I got a kick out of this because of course...

Leo: That's the point.

Steve: Yeah, I said - when I said they wouldn't break anything, I meant you'd still be

able to log into wherever you wanted to. But them randomizing MAC addresses is intended to break these retail analytics services. So anyway, I appreciated David's comment because it allowed me to make sure that I had made that clear. And I got a kick out of the fact that he even mentioned that he's looked into it, and they're hashing the MAC address within each WLAN access point to preserve a measure of privacy, which is an admission that there is a privacy problem. But the fact that they're using a hash, as we know, a hash is going to turn a MAC address into another unique token...

Leo: Yeah, big deal.

Steve: ...which still allows tracking.

Leo: Big deal.

Steve: So, whoops.

Leo: Big deal.

Steve: Or, as we say, that's not a bug, Leo.

Leo: It's a feature.

Steve: That's a feature.

Leo: I'm thinking he probably works for one of those companies. Doug Z. in Bethesda, Maryland knows a thing or two about the Windows Azure IP space problem: You mentioned the Azure IP space recently. I wasn't surprised. I've been using Azure - that's Microsoft's cloud service - for about a year now, primarily for testing. And one thing that struck me as very odd from the get-go is Microsoft does not allow you to allocate a standalone virtual machine that isn't publicly accessible. Part of the way they built Azure requires a public IP for every virtual machine created. What?

Steve: Yeah.

Leo: That seems odd. For my testing needs, I only need one publicly accessible machine, even though I frequently spin up to 100 VMs up. I need them all to be able to communicate with one another, but only one of them needs to be accessible from my personal computer. However, Azure has no ability, at least in this time of writing, to specify that a virtual machine is only internally accessible. My point, of course, is if I'm using Azure in this way, there must be others using it like this, too. That's why they need their IP space. It's being unnecessarily eaten up simply due to the lack of options that Azure provides to end-users when it comes to defining a VM as not

requiring a public address. In any event, I thought this was interesting and worth mentioning. Well, that explains why - what does Microsoft have, one or two of these, what is it, Class B addresses? That's a lot.

Steve: Yeah. And I really - I wanted to thank Doug for this. This was interesting. And you can imagine that somewhere right now somebody is working on adding that feature to the Azure system so that...

Leo: Well, but the problem is, since it didn't have it, so many people probably don't even worry about, and they just go, eh, there's plenty. Microsoft's got a huge store of these.

Steve: Yeah. And as we know from - we covered this, I guess it was last week, that they've run out of domestic IPs, and they're now assigning them from their other major geographic spaces and upsetting people who are using IP geolocation. And it makes their service look like it's in some foreign country. It's like, whoops. But, wow, yeah. Yeah, that's a problem.

Leo: Question 6, Chris Murphy, Denver, wonders about secure sites not showing they're secure: I got a new credit card, and I started going through the laborious task of updating all the sites that have my card on file for auto payment, or that I might shop with. I, like you, joined the Harry's shaving bandwagon after hearing you and Leo rave about them. Yes, a truly nice shave. I went to update my credit card information, but I noticed I did not see the telltale signs of a secure connection. No little lock, as you show. No way to see an HTTPS, nothing. I sent them a message saying, "Hey, guys, I heard of you on Security Now!. You'd better get up to speed." But then I hit Pandora, same thing. Using DigiCert, I found they're quite happy with both sites, so I switched browsers. My primary is Safari on the Mac, but I get mixed results with Firefox on the Mac, PC; IE on PC. Some sites showed different things, none consistent. How can we tell folks to don't do anything...

Steve: Yeah, sorry about that. I didn't edit that.

Leo: How can we tell folks not to do anything until they see the lock and/or HTTPS if it isn't constantly displayed? How can I, as a somewhat savvy computer user, trust a site myself? Should the browsers stay somewhat consistent in showing what's secure and what's not? Thanks in advance for any advice. This is a common question.

Steve: Yes, and we haven't talked about this for a long time, so I thought it was worth sort of coming back to it. And I had something to add to it, also. So the problem, Chris, is that it's very much like I was just talking about in response to the question about HTTPS. GRC went to only HTTPS. So even when you're doing nothing, I mean, when you're going, just like browsing around GRC, we're secure. We're HTTPS everywhere, all the time. And we even have, we even support the HSTS, which is the secure transport flag, where we're telling servers, only connect to us securely. And we even built that into Chrome. That was Adam Langley I had email exchange with, and I said, hey, could you add me to your list of domains that you only connect to securely?

So it is possible to always be secure. But it's not necessary. And the confounding thing is that you could be sitting at a page on the site, asking for your credit card information, where the page itself is not secure. The page was delivered insecurely. But the form, when you actually click the button, it will be a secure submission. Now, that's really, though, sort of retrograde. The reason is, if the form is delivered insecurely, it could have been modified.

And so once upon a time we were saying, well, you know, that's kind of okay, as long as when you click the button, the submission itself is over a secure channel. The problem is you can't see that. When you click the button, the confirmation page will come up in response to your secure connection and submission. And it'll say thanks very much. That'll be secure. But it gives people a much better sense of security if the form itself is delivered securely.

But it's also only the safe way to do it. If the form is not delivered securely, it's completely susceptible to somebody altering it, changing, for example, I mean, changing the URL that the form submits to. So if you get a form from your bank not delivered to - where the form itself, the fields you're filling out, is not secure, you have no idea, no assurance that, when you click the button, it's going to go back to the bank because their insecure delivery of the form means that anybody could have altered the URL in the form's Submit URL to send your credit card information elsewhere.

So I remember years ago saying, yeah, that's really not such a big deal. Of course the world has changed a lot since then. And I would argue now, wow, you don't want - you want to get to a secure form, and that's what you fill out, because then you know the form came from the institution, and nobody could have gotten into it and changed it.

Now, there's an option in NoScript which is nice, which is off by default, but I've got mine on, and rarely do I see a problem. Only when I'm not expecting the form to be secure. This switch in - wait, I said "NoScript." I meant "LastPass." It's a LastPass feature. So everyone using LastPass can turn this on, although it's not on by default, which warns of an insecure form submission. So LastPass is in there, looking at all of the code coming into your browser. And it will alert you, if you have a page, if you're received a page that's got form stuff on it, and the URL to submit it is not secure. So that's a nice feature to have turned on. And I would recommend everyone do that.

And as I said, sometimes you'll come to a form that just doesn't matter, you know, it's a questionnaire or something, and who cares if it's not secure. And LastPass, sure enough, pops up a notice saying, just so you know, this is not a secure submission. It's like, okay, good. Thanks for letting me know. I'm glad that normally I don't ever see that, when I care about what I'm submitting, the security of that.

Leo: I'm looking for that in LastPass. Where is that?

Steve: It's down deep. I think it's under - on the Advanced tab.

Leo: Yeah, because I did not know that that was available. That's a nice...

Steve: Yes, a nice feature. Let's see. Preferences, Advanced, "Warn before filling insecure forms."

Leo: Good. Now, do sites, is that still - it used to be very common that you'd have mixed pages.

Steve: Yes, I know. Very common.

Leo: Is it less so now?

Steve: Unfortunately, not. I mean...

Leo: Seems there must be a reason for that; right? Why would you do that?

Steve: Technically, basically, it is now, I would say, it's security sloppiness. The web developers are figuring all that needs to be protected is the transmission of the user's credit card information over HTTPS. But they're forgetting that the point is the reason you need to protect anything is interception. Well, and if you've got interception, then the form being filled out could be intercepted, and that URL could be changed. So it is really no longer the case, or technically it never was, but in this day and age where technology is doing everything, I mean, the bad guys are so proactive, I don't think you should fill out a form that was not delivered securely. Yet, exactly as Chris says, he encountered it many times.

Leo: Very interesting. We get that, we've always gotten that question. And I agree. You should, because we are pretty clear that you should be using, if you're giving a password, it should be secure, blah blah blah. If there's no way to see that, that kind of makes the advice useless.

Steve: Right.

Leo: Benjamin in Austin, Texas wonders about timing attacks against crypto: Steve, I'm a Java dev by day, but I've been closely following the LibreSSL developments as a means to both re-familiarize myself with C and learn about some common security pitfalls. All in all, it's been an enjoyable enterprise. One thing I'd appreciate your input on, though: LibreSSL folks have a major beef with the cavalier way other OSes treat encryption calls that are susceptible to timing attacks. While I understand the viewpoint of "defense in depth," information leakage and the like, aren't timing attacks for a nontrivial, frequently performed operation really in the realm of "potential" exploits? I guess what I'm thinking here is, if my focus were securing software, my time would predominantly be spent examining stack corruption and use-after-free over things like timing attacks. Use-after-free over things like timing attacks. Any thoughts on the matter?

Steve: Right. So it's a great question. And it's something I didn't mention last week. And remember I talked about authenticated encryption. I saw something, I just sort of closed my eyes when I saw it. I did a survey when I was looking around to see if I could find any public domain, like open licensed AES GCM code so that I wouldn't have to write my

own. I saw the source for NetBSD. And there in the GCM source, where they're checking to see if the authentication tag is correct, they do the standard memcmp, m-e-m-c-m-p, call, which is not time safe.

What happens is when you're in the computer, you're comparing two strings, or in this case they're, like, depending upon the length of the tag, they might be 16 bytes. You look at the first byte of each and see if they're the same. If they are, you look at the second pair. If so, you look at the third pair, and then the fourth pair, and the fifth pair. When you find a mistake, you technically or typically abort your comparison. You immediately return, saying, whoops, they're different. And only if you get all the way to the end, then do you know that that the two strings are identical.

And in the NetBSD crypto code, exactly as the LibreSSL folks are shaking their heads about, is an insecure timing attack, meaning that, if the tag doesn't match, the response from that will be faster than if it does match. And if you looked closely enough, you could figure out how much of it matched because the more that matches, the longer the string comparison will go before it finds the mismatch.

In my code, for example, and in secure code, the way you solve this is you exclusive OR the two things. And then you OR that into sort of a running collector of wrong bits. And you always do that across the entire string so that the length of time you take is always the same. And then only at the end do you look to see whether any of those exclusive ORs, which are essentially comparisons, ever caused a bit to be set in your sum, which is essentially summing all of, you know, like an OR to collect all of the different bits. Then you reply with the answer. And so that's the right way to do it.

But it certainly is the case, I mean, right there in NetBSD is timing attack-prone crypto code. So what's really sobering is the phenomenal results that attackers get with timing attacks. We've talked about it on the podcast before. It is surprising how potent these are. You can do them, for example, in a shared hosting environment, even when you're, like, in an Azure or an Amazon AWS server, where you're in different virtual machines on the same physical machine. All a bad guy needs to do is arrange to be running on an adjacent virtual machine. And they then, from outside, exercise your secure code, giving it lots of things to do. And not even being in the same VM, but being on the same physical chipset, they can crack your private keys. It's just - it's amazing.

So it's absolutely the case that timing attacks are real. They're so-called "side-channel attacks" because they're using something, there's a side channel timing or energy consumption or radio frequency emissions, something other than the regular data channel that is causing leakage of what's going on in the data channel. And, boy, all it takes is a few real-world examples of that, and you become a believer. It's absolutely worth doing, as I've done with SQRL; as I'm glad to know the LibreSSL guys are doing; and as unfortunately, the NetBSD guys haven't. And I was hearing that was such a secure package, too.

Leo: That's too bad. Oh, well.

Steve: Yeah.

Leo: Bill in Miami, Florida, a question about password reuse: I've been wondering if using the same complex and very, very long password string for all my four

computers' WDE...

Steve: Whole-drive encryption.

Leo: Oh, whole-drive encryption, introduced any serous added weaknesses if someone - the NSA? - is able to mount a full-blown attack on all four computer drives. By the way, I've loved SpinRite from the DOS days, cannot count the times over the years/decades where it has been of great help in getting myself or my family members' computers up and running. In fact, thanks to SpinRite, I must have the oldest continuous working hard drive MP3 player in the world. I love software that just does what it claimed to do time after time.

Steve: So this was an interesting question. Of course I cannot speak to all whole-drive encryption. But I know how TrueCrypt works. And the answer is there is nothing wrong with sharing the same complex, very, very long pass string across multiple machines, if the encryption is done right, as it was done in TrueCrypt. What TrueCrypt does is it creates an absolutely random key. And that's what's used to encrypt the drive. Then the passphrase is used to encrypt the key. So on four different drives, independently set up, they will each use a very different random key. And then the same passphrase will be encrypting that very different random key.

And I don't see any, I mean, it is the way TrueCrypt did this; they did it correctly with a random IV, a random key, just the same passphrase. And so it is the case - the weakness is if one of them got cracked through a brute-force attack, which is as far as we know the only known way of cracking that key. And that's why it's got to be complex and very, very long. But a brute-force attack on one would have allowed them to discover that passphrase used on the other three. But the point is I think what Bill's asking is, is there any weakness inherently, like intrinsically introduced from passphrase reuse on different drives. And again, no, because it was done correctly in TrueCrypt, and presumably in other whole-drive encryption systems, too. There should not be a problem. But with the understandable and obvious problem that, if one got cracked, then the other ones have been, too.

Leo: No update on TrueCrypt, is there?

Steve: No, nothing. Haven't heard anything.

Leo: Isn't that weird? It's so weird.

Steve: I don't think, yeah, I don't think we'll hear anything until, I think - actually there was something I saw. It was on Pastebin, though. And I didn't track down the source. It looked like a dialogue that Matthew Green was having with the TrueCrypt - with a TrueCrypt developer, where he was asking the developer to consider relicensing some portions of the code in a way that would allow them to fork it and to reuse it. And again the developer was really resistant to that. He just wanted it not to be used. Use it as a reference, but do your own.

Leo: Well, that's interesting, I mean, the fact that the developer is speaking is interesting, if it's the actual developer.

Steve: Yeah.

Leo: Tim in Southampton, England shares a frightening personal story: Only a few hours after sharing a Google sheet with my wife and unknowingly setting the sharing option to "Anyone who has the link," I was surprised

to see at least two "Anonymous" users viewing the sheet. The link was emailed to my wife, not distributed in any other way. I can only think our Gmail accounts have been compromised, our iPads have been compromised, perhaps Google has been hacked, or more likely someone reverse-engineered the randomly generated URL. That's a pretty long URL. That seems unlikely. I implore your listeners to be extra vigilant when sharing any files within Google Drive. Luckily for me, I don't keep any confidential items in the cloud. The only thing these anonymous users would have viewed is a list of tasks I need to do around the house [sigh]. I've contacted Google to investigate and will keep you updated with any response that I get.

Steve: So it was funny because this triggered a funny anecdote from the very first release of SpinRite 6. I had a technology where my original concept for distributing SpinRite was people could buy it, and I would give them a link with this just bizarre, long, cryptographically-derived string. And this would be - they would receive it actually on the web page. I don't remember whether I was emailing that to them. It might have just been on the web page. They would click it in order to download their copy of SpinRite. And essentially that link was authorized for their use. And I remember saying, print this out. This is your key for access to SpinRite.

Now, I guess I've grown a lot in the last 10 years because I just wouldn't do that now. But what we discovered was that people were downloading other people's copies of SpinRite from, like, archive sites. It's like, what? How did that happen? Turns out that the download accelerators of the era, and probably still today, they were aggregating the links that anyone clicked who was using those tools. So those download accelerators would, like, open up multiple simultaneous connections, like five or six connections, in order to ostensibly get the file quicker. Every link that anyone clicked who was using those was sharing that link with the cloud. So, boy, was it not secure.

Now, you know, I immediately revised the technology and canceled the licenses and reissued licenses for those people whose copies escaped them through no fault of their own, other than using a horrible download accelerator. Now the links are good for only one time, and they expire after five minutes or something like that. So we've never had a problem since because, even if you shared the link, the act of using it, which was the act of sharing it, caused it to no longer be active. So that ended the problem.

But I thought this was interesting. I can't imagine how Tim's URLs to his Google sheet are getting loose. As you said, Leo, those are really long. You're never going to guess these sorts of URLs. It might be that somebody's just looking for them and poking around in other people's business.

Leo: It's weird.

Steve: Yeah.

Leo: Doesn't seem to bode well, that's for sure. Finally, Question 10 from Corby, Reno, Nevada. He wonders about cipher block chaining and data corruption: I've always wondered, Steve, how CBC, Cipher Block Chaining, deals with data corruption. If each block in a chain is affected by the previous block, what happens if just a single bit becomes corrupt in a early block? Surely CBC needs to be more resilient to having one bit corrupt the rest of the data. I'm thinking particularly of whole-disk encryption. Perhaps after a certain number of blocks the chaining process starts over?

Steve: So that was a great question. I was talking also last week, when we were talking about authenticated encryption, about how cipher block chaining, you start with a so-called IV, an Initialization Vector, and XOR that with the plaintext, which has the effect of inverting all of the one bits from the initialization vector of the plaintext. Then that you encrypt to get the ciphertext. Then you take that sort of as the initialization vector of the next block. That is to say, you take that resulting ciphertext and use it to XOR the next block's plaintext, which you then encipher, and get the second block ciphertext, and so forth. So this forms a chain all the way down.

So, yes, one corrupt bit will affect everything else downstream. How is that feasible? Well, first of all, whole-disk encryption doesn't use a single chain over the entire disk. It actually uses a different technology called XTC, like the most recent one, that is, has come into vogue now. But even CBC being used on the 'Net, it is cipher block chaining only chains a block of data, for example, a packet's worth. So a packet might be maybe 16K, for example. And the individual blocks are 16-byte blocks. So a thousand, or a "K," you know, 1,024, chain or block ciphers in a row. And it's true. If one bit was corrupt, then it wrecks the rest. But only for that packet. Each packet starts over, starts its own chain, and it has its own initialization vector and its own sort of a local packet-length chain.

And with communication, if you get a bad packet decryption on the other end, then you say, oops, we got a problem. Please retransmit. So in a communication scenario, you always have the ability on the Internet to say "please retransmit." You don't have that ability on a hard drive. And that's why we have error correction code to fix errors when we're trying to read them back because whoever it was who wrote that data, that might be months ago. They're long since gone. So because you can't ask for it to be rewritten on a hard drive, you need ECC. You don't need that in communications because you're able to say, hey, send it to me again. And the answer really to the question of cipher block chaining is that just small blocks are chained, and the chains always start fresh at the beginning of a new packet. And it works great.

Leo: You know what else works great? Security Now!. And SpinRite, the world's best hard drive and recovery utility. And the two merge together in one place, GRC.com. That's Steve's website. You can go there right now and see all the great free stuff he's offered. Buy a copy of SpinRite. You know, all that kind of stuff. You can also follow Steve on Twitter @SGgrc.

Steve has 16Kb audio of the show for those of you who don't want to spend any bandwidth to get it. He also has great transcriptions. We'd like to start hosting those here, too, if that's all right with you.

Steve: Yeah.

Leo: Somebody will send you a note. Because we do have transcriptions for the other shows, we thought, well, we might as well have a complete set. You can also get full-quality audio and video on our site, TWiT.tv/sn, and wherever finer podcasts are stored, like iTunes and places like that, or get the TWiT app so you can watch live or download later. We do this show live Tuesdays, 1:00 p.m. Pacific, 4:00 p.m. Eastern time, 20:00 UTC. Love it if you'd stop by. And if you can't, well, then just download and listen. That's the most important thing. Steve, we'll see you next week. What are we going to talk about? Do we know?

Steve: No idea. We will let - maybe Google's QUIC protocol. Or maybe whatever the Internet brings us in the meantime. We certainly don't seem to be running out of things to talk about. I've got a huge list moving out into the future.

Leo: Well, I look forward to it. Thanks, Steve.

Steve: Wait, and we're not going to have you next week, are we.

Leo: I don't know who we're going to have next week. It won't be me. I'll be in Hawaii, in Maui.

Steve: Okay.

Leo: Snorkeling and scuba-ing and enjoying my life.

Steve: Cool. Well, I will enjoy talking to whomever you have from your staff.

Leo: It might be Mike. It might be Father Robert. I'm not sure.

Steve: Okay, cool.

Leo: I'm sure we already know, and I just haven't been informed. We'll make sure somebody lets you know.

Steve: That's what it's like to have a big operation.

Leo: Yeah. Hey, thanks, Steve. We'll see you next time on Security Now!.

Steve: Thanks, Leo.