SECURITY NOW!

Transcript of Episode #505

Listener Feedback #211

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-505.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-505-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We are going to talk. Lots of questions. Lots of answers. At the end, a little discussion of favorite programming languages. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 505, recorded Tuesday, April 28, 2015: Your questions, Steve's answers, #211.

It's time for Security Now!, the show where we cover your security and privacy online, with a guy who knows all, tells all, and he does it in plain English, which is kind of remarkable, Steve Gibson.

Steve Gibson: And I'm half behind the flowers on your other camera.

Leo: You are. ProFlowers. Well, you know, Mother's Day is next...

Steve: That's fine. We can see me. Yeah, that's all anybody needs.

Leo: Lisa sent me these. This is one of our sponsors, ProFlowers, and I thought I'd just kind of dress it up for you.

Steve: Oh, very cool. Oh, that's - create a little jungle scene. I like that.

Leo: Yeah. I actually just got a big box from them. I don't know what this is.

Steve: I am peering through the leaves.

Leo: These might be roses. Long, long-stem roses. Anyway...

Steve: We didn't talk about advertisers on the show. How many do we have?

Leo: We have the usual company of advertisers: PagerDuty, Dropbox, and Braintree.

Steve: Okay.

Leo: All the geek ones. Dropbox for Business actually is a new one for you.

Steve: Ah, yes. We have not had them on before. So this is a Q&A. And there's a lot of news this week. So we've got some fun - well, because it was the RSA Conference last week. And so, as often happens, we learn a lot that we didn't know before from the presentations at the annual RSA security conference. That of course is where I ran across Stina of Yubico and immediately understood what they were doing. So all kinds of good things always happen there. A lot of news from that. And then great questions and comments and stuff, fodder from our listening audience. So we've got a great podcast today.

Leo: I always love the Q&As. Just because they ask the questions I've really got in my mind. Oh, I'm glad somebody asked that one.

Steve: So I got a kick out of the little Image of the Week on the first page of the show notes.

Leo: Uh-huh. I see it. Let me open it up full screen so everybody else at home can see this cartoon. It says: "My biometrics authentication just told me I need a shave." Oh.

Steve: Ah, yes. And today we actually have a little bit of coverage of some interesting biometrics problems that also surfaced at RSA. But so this is a guy using his face to authenticate himself.

Leo: Well, and we are now, aren't we. Biometrics are big.

Steve: Yeah.

Leo: We use Touch ID on everything now.

Steve: Yeah. Okay. So the first story coming out, well, the first big one, was one of these where unfortunately all the headlines were inflammatory. I don't know if it was on purpose, or if the people, well, there have been instances, we know that oftentimes people who write articles are different from the people who put the headlines on them. I had the problem during my Tech Talk column is the headline would often say something way more over the top than I intended or than the content said. But that's the way publishing works, for whatever reason.

Leo: Yeah. And they're trying to get clicks.

Steve: Yes. So a group known as "Skycure" during the RSA Conference on Tuesday revealed a new flaw in iOS. The timing was such that it was essentially resolved around the time of the release. That is, it's fixed in 8.3, which we should all have had for like a week now. But the headlines said things like, "Security Flaw in iOS 8 Can Permanently Crash Your iPad or iPhone." Or another one, "iOS Bug Sends iPhones into Endless Crash Cycle When Exposed to Rogue WiFi." Or "Researchers Find Another Terrifying iOS Flaw." It's like, oh, my lord. And I had people, like, in a panic, reaching out to me, saying, "Steve, I love my iPad. I don't want it to be permanently crashed." It's like, okay, calm down.

So what this is, is this group discovered that there was a bug in the certificate parsing of iOS 8, or iOS before version 8.3, where it was fixed, that would cause iOS, upon contact with a WiFi hotspot, to crash. And then they noted that, for example, AT&T iOS devices, like AT&T-based phones or iPads with cellular service, automatically connect to any network named ATT WiFi without even asking.

Leo: See, promiscuous WiFi. It's always a problem.

Steve: Exactly. So their whole deal was, okay, they found a certificate parsing flaw, which is not good, and it's been fixed. And if you were in range of a rogue WiFi hotspot that was offering this certificate, your device would crash. Then it would reboot, reconnect, and crash; reboot, reconnect, and crash. And do that endlessly. All you had to do, of course, was turn it off or walk away, get out of range, and then turn off WiFi, and you'd be fine. Anyway, it's been resolved. And I think it falls far short of being a "terrifying iOS flaw." I mean, there's a...

Leo: And it's not permanent. I mean, as soon as you go somewhere else, it'll stop.

Steve: Right. Right. It was just while it was in - if you had WiFi on. And of course you're punished for trying to turn it off because, oh, location services work so much better with it on, as we've discussed. So anyway, that's all that was. It wasn't doing any permanent damage. It was a sort of a style of denial of service, inasmuch as, if you happened to be within range of a rogue WiFi that was getting its jollies briefly, until this was fixed, in crashing iOS devices, yeah, then I guess that was an Android-only hotspot for a while.

Also at the RSA Conference, this time on Friday, the guys at FireEye had some not good news for Samsung S5 Fingerprint Reader people. Basically, it's a security catastrophe. Their talk was titled, "To Swipe or Not to Swipe?" - and of course using "swipe" figuratively because we're not really doing that anymore, we're using static images - "A Challenge for Your Fingers." And so they start the talk by noting that, if a password is leaked, you can change it. But if a fingerprint image is leaked, that's a problem because we're unable to change our fingerprints.

So they articulated three classes of attack, something they called the "confused authorization attack," the "fingerprint database manipulation," and "fingerprint sensor spying attack." And we can imagine what those are. I'll just quote a little bit from this. It said: "Android phones typically store sensitive data such as fingerprint information in a walled-off area of memory known as the 'trusted zone.' However, FireEye researchers found it was possible to grab identification data before it is locked away in the secure area. This method of stealing data was available on all phones running under v5.0 or older of Android, provided the attacker got high-level access to the phone. They also found that on Samsung Galaxy S5 phones, attackers did not need deep access to a phone. Instead, just getting access to the gadget's memory could reveal finger scan data.

"Using this information, an attacker could make a fake lock screen that makes victims believe they are swiping to unlock a phone, when they're actually authorizing a payment. In addition, they found it was possible for attackers to upload their own fingerprints, then authenticate to the device, since the devices were not keeping good records of how many prints were being used on each device. The flaws they uncovered were widespread throughout handsets running Android 5.0 and below. Updating to the latest version of Android, v5.1.1, should eliminate the vulnerabilities."

So once again, this was coordinated and timed so that they were able to say, look, this is the important research we did. At the same time they were - we didn't have to wait long before we got these problems fixed. But of course we know this is why we inspected so carefully Apple's initial implementation of a fingerprint scanner, because, I mean, exactly these sorts of things. You really don't want your digital fingerprint to get loose. But then the flipside is, while fingerprint authentication is easy, we've also seen that all of these sensors can be spoofed. And now we're even seeing things like high-resolution photos of somebody waving is like - we have enough resolution in our cameras that you can capture a fingerprint from a photograph.

So what this really says is that we need to back off a little bit in thinking that our fingerprint is an unspoofable, super secure method of authenticating ourselves. It's got the problem that we can't change it, which is one problem. It's got the problem that law enforcement can force you to use it, which is another problem because it's not something you know, it's something that you can't be separated from, hopefully. So it's, yes, it is very convenient. But we need to stop thinking that it's absolute protection. There is a tradeoff that we're making. It's just the fact that we leave our fingerprints around all the time, you know, on glasses that we pick up and on the screen of the device which itself might be subject to attack. So we certainly want the underlying technology to be as secure as possible. But I would just caution everyone to remember that there are downsides to it. I love mine. In fact, it's funny because I have multiple - I have several generations. Well, I have every generation of iPad.

Leo: Of course you do.

Steve: Of course.

Leo: Some in the fridge, no doubt.

Steve: And the one in the car is the latest, with the fingerprint reader.

Leo: You have a car iPad?

Steve: Yeah.

Leo: Okay.

Steve: Yeah. It's out there charging.

Leo: Is that...

Steve: Yeah.

Leo: Okay. So you just always have it. So that way - that's for the coffee shop; right?

Steve: Exactly.

Leo: Yeah, okay.

Steve: That's for whenever I'm roaming, I've got an iPad with me. And it stays in the car and charges from the cigarette lighter. Anyway, so I've got one in the living room, one in the bedroom, one of course in the bathroom. I've got one next to me right here. They're all over. Anyway, only - we don't want any waste of time.

Leo: How do you keep them in sync? I mean, you have to install, I mean, aren't they all...

Steve: No, they all synchronize themselves through the cloud, magically. Anyway, my point is that I only have a fingerprint reader on the one that I'm going to be traveling with, the one in the car. And I get mad now at the other iPads.

Leo: Yup.

Steve: Because I'll put my thumb on the button, and they'll light up, but they don't do anything. And I go, "Oh, I have to swipe the screen. What an annoyance." So anyway, yes.

Leo: Or worse, enter your password...

Steve: Crazy world we're in.

Leo: ...when you want to buy stuff. It's so much nicer to have the fingerprint reader. And I feel like it's accurate. I know we've talked about this before. It seems like it's good. It's not - you're not going to get - despite this problem with the Galaxy S5. They didn't say it's the Note 4, which also has a fingerprint reader, or the new S6; right? Just the S5.

Steve: True. The sense I got was that there was something about the design of the Samsung S5 that allowed them easier access to it. And they pointed it out, and the architecture's been changed in order to sort of move things around a little bit better. You could actually get access to that authorized fingerprints database from user mode. And so that stuff should be really walled off better.

Leo: Also, by the way, the S6, the newest Galaxy, has a fingerprint reader that's very similar, in terms of usability and training, to Apple's.

Steve: Nice.

Leo: So I'm really pleased because it means it won't just be iOS devices that have this capability. And they're using a Synaptics, I found out, Synaptics fingerprint reader.

Steve: Yeah, they've been around forever.

Leo: Yeah. They do the track pads on most computers. So that means this is, I think, a technology you're going to see in a much more widespread fashion.

Steve: Yeah. My feeling is, it is a really good tradeoff. I like the idea of what Apple has done, that if you don't use your device for 48 hours, it says, you know, we want to make sure you're still you. And so you have to enter - or after you do a full system restart you have to give it your password to kind of get it up and going. But then, I mean, we don't want something so onerous that people will not use something secure. And arguably, the security versus hassle tradeoff of a fingerprint is a win, so long as it then performs perfectly, that is, it isn't leaking data. It doesn't false positive or false negative. If we take it from the concept of a perfect fingerprint reader, I think that's a perfect set of tradeoffs because people will tend to use it. And it's certainly better than having your device unlocked because you get - you're just annoyed having to type your password in every time you turn it on.

Leo: Yeah.

Steve: So one of the real questions we've had has been how are people getting infected by CryptoWall. And also at RSA, the guys at Malwarebytes revealed just what is frankly a terrifying truth. For two months, starting on December 10th of 2014 through February 2nd, when Adobe patched a zero-day flaw in Flash, CryptoWall, that we've talked about often, the essentially CryptoLocker lookalike, which encrypts all the files on someone's machine and then asks for ransom, it was able to install itself through malicious ads that were being injected through the number one advertising network. And they didn't reveal the name, but they just said it's the number one advertising network. You can figure that out for yourself.

And those ads were appearing on typically for around two days each on sites like - they were found on Dailymotion, Huffington Post, Answers.com, New York Daily News, HowToGeek.com, Tagged.com, and a handful of others. Because this was a Flash vulnerability, it was not necessary for the user to do anything. They don't have to - it's not - yeah, I mean, this is as bad as it gets. You don't have to be fooled into a phishing attack, clicking on a link that looks authentic in email. You simply go to the Huffington Post or Answers.com or HowToGeek.com with a browser that has the Adobe Flash plugin still enabled and willing to run, and that's all it takes. You receive, if you happen to, in ad rotation, one of these deliberately crafted malicious Flash ads. It leverages the vulnerability, escapes from the browser, downloads CryptoWall, and encrypts your system. So the good news is...

Leo: That's really bad. And it probably does it in seconds; right? I mean, it's probably very quick.

Steve: Yeah. Yeah, it's, I mean, all of this stuff is unfortunately at the speed of light. So, and then it begins to encrypt, and you're in trouble. So the Malwarebytes guys found that the hackers were using something called the HanJuan exploit kit, which was hosted on rotating domains to evade detection. And it's, interestingly, it was also - it wasn't a global attack, even though those websites are globally available. One of the things it did was it deliberately targeted U.S. consumers operating from residential IP addresses. We don't really know why, but that was just their target. And no action needed for the infection, just for Flash to run. So, boy, I tell you, I mean, if you didn't have enough reason to remove Flash from your browser before, the fact that we're seeing the ability to inject malicious ads using zero-day unpatched vulnerabilities - oh, and what's interesting is this date on February 2nd, the patch was released on the 2nd from Adobe, and the next day the attack campaign stopped. So...

Leo: Wow.

Steve: Which is really interesting. To me, that's...

Leo: Kind of proves it, doesn't it.

Steve: Yeah. Well, yeah. And it also says that there were a set of tradeoffs the criminals were balancing because they didn't want to give up their network, that is, they didn't want their network to be discovered. They didn't want themselves to be discovered in any way. So as long as there was a high probability, that is, a sufficiently high probability that they could get a percentage of victims that made it worthwhile, they were willing to expose themselves to the degree they were. And, for example, they did enough that Malwarebytes was able to find them and track them and build this complete profile.

But the moment that probability of infecting people dropped, they immediately pulled their system down, obviously to preserve it for the next opportunity. They didn't want it to be taken apart. As soon as it didn't look like it was going to pay off, it's better to protect it and keep it secret than it is to expose it as the number of people patch and the probability of getting CryptoWall infections drops. So I think that represents, among these facts, one of the most interesting is that the logic, the strategy is, as long as this is unpatched, we're going for it. But everything we've built in order to make this happen, the bad guys presumably think, only makes sense to keep deployed until the patch happens. So that also implies that they've seen rather rapid patch adoption. It's not like they waited a month and then they shut it down. The next day, gone. So...

Leo: Wow.

Steve: Yeah. Really interesting little peek behind the curtains. Okay. Now, last week we talked about this arguably, I mean, bad but not end-of-the-world problem in a very popular open source networking library that about five, sorry, 1,500 iOS applications were believed to have used. That was this AFNetworking Library. So it's very popular, free, open source. iOS apps drop it in, essentially use it when they want SSL/TLS connections. And I'm not an iOS developer, so I don't get why this isn't a system service. But apparently it isn't. Apparently, if you want whatever it is this AFNetworking Library does, you can't ask iOS for those services. Maybe iOS is too protective, or who knows. But so people are adding it to their own apps.

The problem we discussed last week that had a relatively short window was from 2.52, which was introduced in January, to 2.53, that was where this problem was corrected. So it was only about three-month window, not really a huge problem. Well, I guess people have been looking at it a little more closely because now an unbelievable problem has been found. And it's being misreported as bugs, and it's not a bug. It is an insane default setting. There is a setting in the configuration for this AFNetworking Library named "validatesDomainName." It defaults to "no," starting with v2. So, which is to say, that's telling the library not to check the name on the SSL certificate that the remote site has just given you.

And it turns out there are somewhere between 25,000 and 50,000 iOS apps that are believed to be at risk. We don't have an exact number because it's difficult to go in and figure out which version any app is using and so forth. But what this means is that there are a huge number of apps, including Bank of America, Wells Fargo, and JPMorgan Chase, which are using vulnerable versions of this AFNetworking Library, where the library verifies the validity of the certificate. It does all of its checks, verifies the signature, verifies the chain of trust back to a root authority, checks the date to make sure that it hasn't expired. Doesn't check the name. So that means that all a bad guy has to do in order to intercept and decrypt communications is present any valid certificate. They don't need a BofA cert. They can use a Joe's Hardware cert. And as long as it's valid and signed by one of the regular certificate authorities, the AFNetworking Library says, yeah, fine.

Leo: It's signed.

Steve: Yeah. It looks good. Not the right company, not the right domain, but we've been compiled with validatesDomainName set to no.

Leo: Wow.

Steve: So nobody can understand how this change got made or why.

Leo: Does this remind you of that programming bug that was in there? What did we call that, where they put in this conditional, and it just jumped around validation?

Steve: Yeah, goto, the...

Leo: Goto. That was in iOS, as well; wasn't it? The goto bug?

Steve: Yeah.

Leo: And it had the same upshot; right? Which is it ignored validation.

Steve: Yup. It just looked like it was...

Leo: It's goto fail. Goto fail.

Steve: Goto fail. Right, right, right.

Leo: Yeah, thank you, John.

Steve: And you could even stare at the code, and it's like, okay, looks fine. And this is the problem with code is you really have to stare at it in order, I mean, mostly that's where I use a debugger is I don't need it often because I write code incrementally. I'm a write/test, write/test, write/test. So I'm constantly checking everything about what I've just written. Then I move on. So it's only occasionally that I'm, like, staring at the code, which doesn't do what I expect, and I can't see it. And so what the debugger does, of course, is it just rubs your face in it. You go, you know, step by step by step by step. And it's like, oh. And then, you know, you realize what your mistake was. But sometimes that's the nature of this.

Leo: I'm going to - okay. Let's go conspiracy, all conspiracy on you. What if the same person that put goto fail in the code, which was this subtle bug that eliminated domain validation, also made this default no?

Steve: Yeah, it's an open source library. I mean, someone maybe, it would be interesting to see where the commit, who made the commit to...

Leo: And we never found out who did the goto fail commit, even though Apple must know that.

Steve: No. Yes, right. And so it's just - it's a little spooky that there could be this sort of longstanding problem. And, now, if people use the library knowledgeably, that is, if they went through the config and said, wait a minute, validatesDomainName is set to no? Does that mean what I think it means? They would set it to yes. So, but nobody can understand why it wouldn't be yes. There isn't a - except maybe for testing. In a testing mode, you might want to say, look, turn off validation for a minute, just because it's a hassle to get the right cert named. I mean, I'm making something up. I can't - I'm trying to come up with a justification for this. But on one believes the package should ever, ever ship this way. And the whole point of using a library is that you don't have to write it yourself.

Now, some people might argue, well, yes, but you should be a responsible user of it. Well, okay, but it ought to default to sane settings. And this defaults to don't bother checking the name on the certificate. It's incomprehensible. Now, the problem is a huge number of apps are using this very popular library, at least 25,000, maybe as many as 50. The company that has been on the front of this, this SourceDNA gang, they've got this search system. The problem is they can't publish the list because of course then that creates attack targets for the bad guys. So their sort of compromise is to create searchlight.sourcedna.com, where you can put the name of an app you are wondering about in, and they will tell you what they think about it.

So, I mean, which is really, if I look at the number of icons I've got on my pads, having been an early adopter and a frantic user of all this stuff ever since, I can't imagine checking them all. But maybe your more important ones - BofA, Wells Fargo, JPMorgan Chase, which unfortunately are all on the bad list - and look for updates. This is fixed in the latest release. But unfortunately, whereas this prior problem only had a window, because it was an actual bug that was introduced in 2.5.2, that lasted for three months till 2.5.3 that was about a week ago, this one was everything in v2. So until last week, everything compiled with v2, unless they either turned validatesDomainName to yes, or they could be using other techniques, for example, certificate pinning, where you actually check the serial number of the cert that cannot be spoofed. So it is possible that some of these apps, like for example BofA, might have left validatesDomainName set to no because they know, the app knows the serial number of the legitimate certificate and is verifying that, which is a possibility.

I mean, again, we're trying to make excuses. The only way I could think, well, I mean, I could think of ways to test it. You could play games with your hosts file on your own system in order to redirect a domain you wanted to check, like Wells Fargo, to Google.com, and see if the app, it's going to give you some sort of error, but see if it gives you a bad certificate error, or refuses to connect, or if it sort of fails in some other way, which would be a concern. I mean, unfortunately, none of these are good tests. We're sort of stuck. We need an incredible number of apps to be updated. And until that happens, we don't have the security that the application authors intended their apps to have because validation of domain names was set to no for some unbelievable reason, unfathomable reason.

Yubico got caught by a little open source software problem. They issued a security advisory on April 14th. It turns out there was a logical mistake in the open source code that Yubico inherited from the Java Card OpenPGP project. There were - and this is another example of code being hard to read. This really is hard. This one makes your eyes cross because - only because it's a double-negative with a conjunction. You can see the if statement here in my notes. The source code that they inherited contains a logical flaw that relates to the validation of the PIN that users have to enter which, if abused, allows an attacker with localhost privileges or physical proximity, like within NFC range, to perform security operations without knowledge of the user's PIN code. So it's a PIN bypass. And you can see, unfortunately this split across page boundaries, but they've got an if statement, and then open parens, and then "not pw1" is validated and "not pw1_modes" and then some mode function bit. And so they're...

Leo: They both need to be true.

Steve: They're inverting the sense and then ANDing them. And in that case...

Leo: Right. So they prove that to be false, then; right?

Steve: Correct.

Leo: Let's see, validated and this not satisfied. So they both have to be true. And what you really want is either one is true. That would...

Steve: And so that double ampersand should be a double vertical bar. That should be an OR conjunction, not an AND. And again, this is, like, this is the kind of thing that our human brain just doesn't process that well. In fact, in their security advisory, Yubico draws out a truth table of all four possibilities of both modes, their inversions, and their conjunction in order to say, look, this is wrong. Anyway, so the Java Card OpenPGP project has been notified. That same if statement occurred in three places in the code, all wrong. I mean, identical but wrong. So someone wrote it once and then copied and pasted it because they wanted the same function a couple more times. And Yubico, this ends up in the firmware of the NEO. So this only affects the YubiKey NEO. If you are using any version before 1.0.10, and you're using the OpenPGP applet, they will replace your NEO key at no charge. So you can go to...

Leo: Oh, that's nice.

Steve: ...yeah, yubi.co/support to learn how to log a support ticket and receive a replacement for free. They're taking responsibility for this.

Leo: I guess a firmware update is not possible. They have to give you a whole new hardware, yeah.

Steve: Precisely. It's deliberate, I mean, this protects it...

Leo: They don't want it writeable.

Steve: ...from the bad, exactly, from the BadUSB problem, is it's like, nope, this is factory only. Oh, god. And I just - I got a kick out of this story. This is sort of random. But this is from the "what could go wrong" department: "Amazon to start delivering packages direct to vehicle trunks."

Leo: I saw this. I love this.

Steve: Oh. I do, too. This is just - okay. So Amazon and Audi in Germany are conducting a pilot experiment to allow DHL delivery trucks to find and remotely open the trunks of Audi vehicles of owners participating in this project. And the story was interesting because it highlighted some problems I never really appreciated before. So I'm just going to share this. It says: "Together with delivery partner DHL, Amazon and Audi aim to ease a common frustration among commuters: never being at home when the delivery company brings goods that were purchased online."

Leo: I hate it when that happens.

Steve: Don't you. Of course, you've got all kinds of doorbell things that ring, so you know when the guy is there.

Leo: I can even let him in.

Steve: That's right. "Parcel-to-vehicle" is what they're calling this. "Parcel-to-vehicle delivery might help reduce the number of failed delivery attempts and temper a parcel logjam in large offices caused by employees who input their employer's address when ordering goods." So you can see what that's saying is that one solution people have found is, oh, deliver it to my office. Well, yeah, it's a 112-story building.

Leo: Yeah, where are you, yeah.

Steve: You're Mabel Appleby, yeah. So carmakers, meanwhile, "are seeking to offer," god help us, "to offer an array of add-on 'connected' services to retain tech-savvy customers and ensure the profits accrue to them and not to software companies. Audi said there were no vehicle insurance implications because the delivery agent will not be able to access the vehicle cabin." Which rules my car out because from the trunk you're able to pull some snaps and fold the back seats down in order to make room for bigger stuff and thus get into the inside of the car. But on the other hand, believe me, I'm not a candidate for this wacky thing.

Leo: I, by the way, I can't wait.

Steve: Oh, I know. "The carmaker said that in future customers would also be able" - oh, get this, it's going to be a two-way service - be able to send letters or parcels..."

Leo: Mm-hmm, mm-hmm, leave it in your trunk, yeah.

Steve: "...left in the trunk of their car."

Leo: I can't wait.

Steve: "And Amazon said it was working on a solution to allow goods to also be returned via the boot," as they call it. "When ordering, Amazon customers will indicate the rough location of the vehicle and desired delivery time. A DHL delivery agent will later be notified of the exact location via a smartphone app. The agent is granted one-time keyless access." And actually, circumventing this was the topic for next week, except that the congressional testimony of Matt Blaze, famous cryptographer, the guy who found the problem in - I'm blanking. What was the government key escrow nightmare?

Leo: Oh, yeah, yeah, the - yeah, mm-hmm. Clipper Chip.

Steve: Clipper Chip. He found the problem in Clipper Chip. Tomorrow he's testifying in a congressional committee hearing. I was hoping that C-SPAN was going to be covering it, but they're just not - no one's doing much tomorrow, unfortunately. But I do have his testimony that has a number of amazing things in it. So I'm going to share that with our listeners next week. But in three weeks from now I want to talk about bypassing keyless entry because I found the whitepaper about how to do that, you know, how these people walk up to the car, and it unlocks. And it's also really clever, in a different way, and chilling.

Anyway, so "The agent is granted one-time keyless access to the boot of the vehicle; and, when the boot is shut again, it locks automatically," like a good trunk should. "The customer must agree for their vehicles to be tracked for a specific timeframe and is notified via email upon successful delivery." And Leo, as soon as this service is available in Petaluma, we want your full report.

Leo: I'm going to totally do this.

Steve: I'm sure that your Watch will tap your wrist as soon as your package has been - as your mail has been taken out and your parcels have been delivered.

Leo: It's already in the trunk, and I can just drive home with it.

Steve: You could be having dinner out in the restaurant parking lot and, oh, look.

Leo: They deliver this to the front desk, I have to schlep it all the way back to the trunk of my car. I don't want to do that. Just put it in the trunk.

Steve: That's right.

Leo: What could possibly go wrong?

Steve: What could possibly go wrong.

Leo: Unless they put, like, I get a malicious DHL guy who puts a bomb in my trunk. But that could never happen.

Steve: This is why the podcast will never end, Leo.

Leo: [Laughing]

Steve: Now, this one is interesting. This sort of spun me in a circle because I started off - okay. So the title of this topic in my notes is "Why ad blocking is devastating to the sites you love." Which was the title of an editorial written by Ken Fisher, the founder and editor-in-chief of Ars Technica, back in early March of 2010. I found that when searching just sort of curiously for dialogue about the ethics and morality of ad blocking.

Leo: Thank you.

Steve: Because there was a project on GitHub called uBlock, which is there, uBlock, on GitHub, which compares itself to Adblock Plus, MultiBrowser, Safari, Chrome, Opera, Firefox. It is able to use the - it uses the Adblock Plus protocol, and syntax is really what I meant, so that things like EasyList can be subscribed to. And they go further, and they compare themselves showing much lower memory usage, much lower CPU usage, blah blah blah. So I thought, okay. And it'll do more things. It's sort of a general purpose blocker. So I just sort of thought, okay, what's the status of that? For a while I was thinking I want to launch this as a topic over in the GRC newsgroups, although we did do this, like, a decade ago, probably pre-podcast, had a really interesting discussion about the pros and cons.

But what Ken's article said that stopped me cold was, well, the article begins, the first line is "Did you know that blocking ads truly hurts the websites you visit? We recently learned that many of our readers did not know this, so I'm going to explain why." And the short version is that sites are paid for impressions, not only for click-throughs. In our common...

Leo: In our case it's impressions a hundred percent.

Steve: Right. And so the common misconception is that only if you click on an ad is any value accrued to the site.

Leo: We don't do that. We do impressions.

Steve: And I've heard people say, oh, click a few ads to pay them back. And we've heard, for example, of malicious scams where malware will bring up a site and click the ads in order to generate revenue. But as you affirm, Leo, it's impressions. So what that says is that the act of your browser taking the time to retrieve the ads from the ad network or wherever, that act generates revenue for the site. Which means by preventing your browser from doing that retrieval, I mean, the tough area is that you have control over your browser. And I'm the first one to say, I mean, I have a low tolerance for obnoxious ads. But I don't mind at all if there's ads on the page when I understand that the act of my browser retrieving those ads is providing revenue to that site.

And so for me, I mean, it's actually why I've liked what Adblocker has done because there's a huge list of permitted ads. They just want them not to be obnoxious, not to be jumping up and down and popping stuff up. I actually turned JavaScript off on Safari for a while on my iPad, my beloved iPad, because as it happened the sites I was going to had just horrible pop-ups coming onto the screen that were really obnoxious. And so I thought, I'm just going to turn off JavaScript so this stuff can't happen. I ended up having to turn it back on because of course sites don't function very well without it.

But anyway, so I wanted to make sure that our listeners know that pulling the ads is generating revenue for the sites they visit. And as long as it's not abusive, I mean, it's true that the advertising agencies could go too far, could flash things and have things jumping around and so forth, which I would regard as a problem. But short of that, given that ads are willing to sort of be compatible with our main focus, which is to get content from the site, I'm glad to know that it is the browser pulling the ad. Yes, it takes a little time. Yes, it takes some CPU resources. Yes, it does all those things that the uBlock folks are bragging that theirs doesn't. But is that so much to ask? You and I, Leo, have talked about how we will give a hundred bucks every year or so to Wikipedia because we're using it.

Leo: Yeah, I'm a monthly donor, yeah.

Steve: I mean, we're actively using it. Right. And I also, like the NoScript guy, I'll just send him a blob of money when I realize, you know, I haven't done that for a while, because I really want it to continue. Well, we really want websites to continue. And I just don't think having my browser pull the ads and, as long as they're innocuous enough, present them to me. I mean, I don't look at them. I don't click on them. But I think that's a completely acceptable tradeoff. And I don't have a single ad on my site. But I still, you know, I want the sites I care about to not go out of business. And we have seen some sites dying recently.

Leo: GigaOM.

Steve: Yes, that haven't been able to make ends meet. And anyway, the other point that Ken made was, unfortunately, because Ars Technica is a technically oriented site, they are...

Leo: That's the problem. And same with us. We have savvy viewers.

Steve: Disproportionately affected by ad blocking because you're not - it's not the common grandmother who's got a generic machine that she purchased and turned on.

Leo: Yeah. Doesn't bother Yahoo! at all.

Steve: Right.

Leo: It bothers Ars Technica. Now, I'm going to say a few things, if you don't mind, because...

Steve: Yeah, please, please.

Leo: A, we don't - I completely understand why people block ads. And somebody shut me down, I think it was probably on Twitter, when they said, "But Leo, do you skip ads on your TiVo?" Of course I do. We don't watch "American Idol" or "The Voice" live because there's so many ads in there.

Steve: It's impossible. You can't.

Leo: We just skip through them. And so we're doing - I do the same thing to television. So that's one thing. The other thing is our ads on our website are a very small amount of income for us. We use them, we do it for other reasons, the way we can get them - it's part of a package that we sell which includes the important ad, which is the ad on our show. The other thing that is really important, I think, is that of course people are blocking ads because they're annoying, but also they're malware. There's all sorts of issues with these banner ads. And I understand that.

And so to me the solution, really, long-term solution is for publishers, and I include podcasters as well as website publishers, to consider their audience, to make sure that, if they're doing advertising, they're doing it in an appropriate way with ads that they might be interested in. We're very picky. I turn down advertisers every day. And you ought to see how pissed off people get when I say I'm not going to take your ad. It drives them nuts. They say, "Why not? You think there's something wrong with us?" I say no, it's just not a good ad for our audience, or for whatever reason I don't want to advertise for it. And then I think the other thing is you have to have a connection with your audience. So we know that our ads work because our audience wants to support us; right? People love Steve. They buy SpinRite because they love you.

Steve: Right.

Leo: As much as because it's useful. So you do exactly the same thing. You make a useful product, you tell people about it every show, and they buy it to support you. And that's kind of the same thing we do. And that has worked for us. So it is possible. And I think you have to respect your audience. And I think the real problem is so many sites don't that the audience has been compelled to use ad blockers because they can't take it. But there's this ethical conundrum because you're saying I want the content, but I don't want to pay for it.

Steve: Yeah. I would, I guess, first of all, I'm also a commercial skipper. I cannot watch them. But I think that's more like the ads are on the page, and my eyes refuse to focus on them.

Leo: I do that, too, by the way. When they do an ad takeover that you can't read the content, I avert my eyes for the four seconds because I don't - and then I click "continue."

Steve: Yeah, I just, I mean, I'm averse to having stuff pushed in my face.

Leo: I agree. I agree.

Steve: And if TV commercials were engaging and, you know, like Super Bowl, look at Super Bowl commercials, my goodness. They're fantastic commercials.

Leo: You watch the Super Bowl to see the commercials.

Steve: Right. Oftentimes that's the case. So anyway...

Leo: So we do things because we understand that. We only do one ad per half hour ever. That's the maximum, compared to 15 minutes per half hour on network television. We only do ads for products we use and can endorse. We try to pick products we think our audience would be interested in. We try to make them relevant. And we don't spy on our audience to find out about that. I just use my instinct. I think we do the, I mean, we are ad supported, so this is something that's of interest to us.

Steve: Well, and our listeners don't know that I get email from your staff saying, "Hey, Steve, this is a company that wants to advertise. Do they look okay?" I mean, that happens all the time.

Leo: Yeah, we vet every single - and, you know, we're at the point, we've been around long enough now, that we get approached every day by people who want to buy ads. And they're just shocked that we will turn their money down. Just shocks people. But it's part of our deal with you, our audience. So, and yeah, we do, we vet every single advertiser. We make sure they're the real deal, and blah blah blah. It's an interesting thing, and I do still skip ads on TV all the time.

Steve: Yeah, I just...

Leo: I understand.

Steve: But on the other hand, they're there, and it's what pays for the content, such as it is. And there are a lot of people who, I mean, what, are you obligated not to go to the bathroom during the commercial break?

Leo: No, in fact...

Steve: What can you do? Anybody who's watching in real time may have other things to do. So, I mean, ads have always had sort of a rough go of it.

Leo: Yeah, yeah.

Steve: I think they have to survive on their own.

Leo: They do. If they weren't annoying, we wouldn't do it.

Steve: Right. And so I just - I wanted to make sure people understood. I thought that was an interesting point.

Leo: Thank you.

Steve: First, that impressions, your browser pulling the ad generates revenue for the site. And secondly, that also then, that note that we probably tend to - we, this podcast-listening audience, tend to more technical sites that may also have higher costs just because to have higher quality content, more technical content, is a little more expensive. And we're also the same people who would tend to be blockers. And so I would just say, you know, think about how you feel about it. I've seen other people's computers where there is no control, and it's like, oh, my god, this is for people...

Leo: It's pretty bad, isn't it.

Steve: Oh, lord.

Leo: It's worse than you knew.

Steve: How do you use this?

Leo: You know, it's funny, because Ars Technica, shortly after that column, started a premium service where you would pay and get some additional features. And I paid immediately to support them because I really love their content, and I don't want them to suffer. I want them to do well.

Steve: Yeah.

Leo: It's a great site.

Steve: So somebody posted on Stack Exchange a quick modification to - and I haven't got a link yet to my TrueCrypt page, but I will - to the installer script for TrueCrypt, which has had a problem with Yosemite when it went to 10.10 because the installer script wants to make sure that you're on Mac OS X v10.4 or later. So it was happy when you were at 10.9. But when you went to 10.10, unfortunately the comparison, it's probably a string compare or something, which sees the one as being lower than the four of 10.4 and refuses to install. So it's like five lines that you remove where it's checking the version of the install - it's checking the version of Mac OS 10, and then it installs just fine on Yosemite 10.10.

So this happened when we went to 10.10.everything because it's just doing a non-numerical compare. If you google "TrueCrypt requires Mac OS X or later," that will bring you to this page, I verified, since the link is crazy. I didn't want to create a bit.ly link for this. And because I will get this on my page soon. If you google "TrueCrypt requires Mac OS X or later," those are words from the crazy URL. And so Google found it. It'll take you to the page. And it's just a trivial change to one file in the installer that performs that check and normally rejects you. And then you're able to keep using TrueCrypt 7.1a for as long as you want.

Okay, Leo, miscellany. Hook. The puzzle you hooked me on a week ago.

Leo: I can't believe you've finished it already. You were...

Steve: Oh, instantly. I finished it in a day and was very disappointed that, when I finished level 50, 50...

Leo: Oh. See, I'm almost there. I didn't realize that's all there were.

Steve: That's all there was. Now, I'm in dialogue with the author.

Leo: Oh.

Steve: Because this thing is too good for it not to continue. It is just - it's just wonderful.

Leo: It's so great that you love it so much. That's so great.

Steve: I'm just - it's the definition of a perfect relaxing puzzle. And that's what - the comments that people have had is generally that. Basically it's sort of a graphical kind of wiring combinatorial puzzle. You're playing the video on his site. And that's at PlayTheHook.com. So for people who don't know, it's 99 cents. It's iOS, Android, or Windows Phone. The guy first did an HTML5 version over at Kongregate. And in fact, if you go to PlayTheHook.com, Leo, yup, there it is. So there's his HTML5 version. And that is workable. It runs. You can click that and start playing. And it's interesting because this one has additional features, widgets, that his doesn't. And so I'm very much hoping that this is going to be - he's a neat kid. I can't remember...

Leo: He's a kid?

Steve: Yeah. He dropped out of college or university. And he has another one that is my worst nightmare kind of game, where it's twitch and reflex and...

Leo: Yeah, I like this because you have as long as you need to solve it.

Steve: Yes. There's no timing base. You can just sit there and sort of study it. And there are - you can solve it in a minimum number or just sort of be a little more relaxed.

Leo: Yeah, that's kind of fun is to kind of go back and see if you could do it in fewer moves.

Steve: Yes, because many of them you can have multiple things happening at once.

Leo: Right.

Steve: Which you can or cannot bother to do. So anyway, all I want from him is more because it's just wonderful. But so I wanted to let our listeners know, if you're not hooked on the idea to spend 99 cents, go to PlayTheHook.com, then follow the link to Kongregate, where you can actually play it, although you do have to turn every

JavaScript thing on you've ever seen.

Leo: It's not the Flash, though; right? It's JavaScript.

Steve: No, it's not. It's JavaScript. But it took me a half an hour of enabling NoScript to finally get all the pieces.

Leo: That's so funny. That's probably because of all the stuff around it, too; right?

Steve: Yeah, yeah, exactly.

Leo: A lot of ads on it and stuff.

Steve: Yeah. But if you go a little further, you might see something you've never seen before. He's got like a rotating circle with a chunk taken out of it. And I haven't gotten to it yet. Anyway, so he has some ideas that are a little bit off of the theme of there being no timing. He introduces that. I don't care what he does. I just want more because it was just - it was very enjoyable. So for anyone who likes the idea of a calm puzzle, I got a lot of positive feedback from our talking about it last week.

Leo: These look like all new levels. I don't recognize these levels.

Steve: Oh, yeah, these are not from his - basically this HTML5 version did so well, and it got such good, positive feedback, that he decided to take it to a full commercial app and implement it that way.

Leo: I hope he's doing well because it's nice. And it's original, which is kind of fun. You played Blek; right? We showed you Blek.

Steve: Yes.

Leo: This is like Blek.

Steve: And I did it a little bit, and then I thought I was well named.

Leo: Ah.

Steve: So, yeah, that one didn't grab me. This one, I like the idea of just sort of being able to stare at it. You sort of say, okay, now, that one's going to come out first, so follow that line around. And I've got to switch that one here, then I've got to press that. It's just - it is very enjoyable. So I wanted to make sure that people knew, just as a follow-up on our discovery of it last time. You keep doing that because I want you to find the cool round thing.

Leo: Okay. And it's on iOS, Android, and Windows Phone, which is great.

Steve: Yes. I did talk about the Apple Watch already. I mistakenly thought that the band, the $300 band could be too small for someone. But you explained that that was the woman's version, or at least the smaller person's version, and that the band for the regular watch, the larger watch, is proportionally larger.

Leo: And Apple tells you how big it is and has a little sizer. So you know ahead of time.

Steve: Yes. They've got a beautiful sizing chart and everything. And as you said, they're encouraging people to come in and not just press some buttons on the website and hope for the best.

Leo: Well, now you understand why; right? Yeah.

Steve: Yes. Now, everybody knows that I've got these blinking lights behind me. I'm trying to point to them, there. There they are. And that was from a relatively expensive, but also very nice, PDP-8 mini computer kit which I built years ago. That was based on an increasingly scarce chip, which was actually an integrated one-chip PDP-8 computer. They put the whole thing on a chip, which, you know, it sounds like a big deal, but obviously it's not because look what we put on chips now. Anyway, a beautiful, talented designer, I want to say Germany, has developed a PDP-8/I kit, where the engine behind it is a Raspberry PI.

Leo: It's probably faster than a PDP-8, too.

Steve: And, now, you need to bring this up. I created a bit.ly shortcut.

Leo: Okay.

Steve: Bit.ly/pdp8kit, all lowercase, pdp8kit.

Leo: And appropriately, it's from Obsolescence Guaranteed.

Steve: Yes. Bring up a picture of it, Leo. The guy has done an incredible job, and he's offering the kits for sale for less than $200.

Leo: What?

Steve: You have to provide your own Raspberry PI.

Leo: Well, that's 35 bucks, so that doesn't add a lot.

Steve: So maybe it's - I think the link's a little bit lower down, the information about it. Nope, not there. Like in that second group of links on the page, I think, right - nope. Yeah, that one.

Leo: Get one?

Steve: Below, that one.

Leo: Get one.

Steve: Nope.

Leo: I'll find it. I'll keep looking.

Steve: It's at the top of the - there you go, that's the one. No, no, no, lower. Okay. In that last group. The first button of the last group. There you go.

Leo: PIDP-8/I.

Steve: Oh, no? Shoot. Well, okay. Go ahead. We have to show it because it's intoxicating. Whereas the PDP-8 only had two rows of lights, this thing...

Leo: No, more?

Steve: ...shows all of the different registers.

Leo: There you go. Here's the front part, anyway. All the registers are lights on the screen, on the front.

Steve: Why, how are we not seeing it?

Leo: Oh, they've hidden it away somewhere. All right. Keep talking. I'll find it. Here's something. Is this it? That's it. It's the very first...

Steve: Yay, there it is. Oh, the very first link.

Leo: Yeah.

Steve: Look at it. I mean, it's just - it is gorgeous. So life-size, acrylic panel, painted switches, in a beautiful case, less than $200, because...

Leo: Is there a lot of soldering involved? No, because...

Steve: Oh, well, yeah, there's a lot of, like, individual little LEDs that need to be soldered into the circuit board. But one of your minions - you have minions, Leo.

Leo: Hey, Minion.

Steve: A minion. Get a minion.

Leo: Get a minion in here.

Steve: Anyway, less than $200. I think it's like 186. The more orders he gets, the lower his price will be. He knows all about me, and he spent a lot of time over on my PDP-8 pages, looking at my project. But, I mean, look at that front panel. Oh, my goodness. No one...

Leo: So are these silkscreened? Do we know how he's doing this?

Steve: Yes, silkscreen, multilayer silkscreen with a blackout mask behind.

Leo: Nice. And is that wood, I supply the wood case, or...

Steve: I don't know.

Leo: Yeah.

Steve: So our listeners need to - I'm sure that he's able to supply that. I don't know whether you have to pay more for that. Anyway, I wanted everyone - many people were interested, but they were put off by the high price of these guys that I did. You have to supply a Raspberry PI, but what's that, 40 bucks or something.

Leo: Thirty-five bucks, yeah.

Steve: Yeah. And so...

Leo: You know what's hysterical is - oh, look, he does have some prebuilt. So that's...

Steve: Well, I'm worried about that. He's offering them prebuilt. He's going to get a landslide of people. And he's just - I don't know how he's going to build them all. I don't think he understands what the demand will be. He and I are in a dialogue. I told him watch out. And when I tweeted it yesterday, he started getting, like, three per hour, where nothing was happening before. And I said, "Well, I'm going to talk about this on the podcast tomorrow." And I said, "I created a short link for you, bit.ly/pdp8kit." So I need three more of these because they go with the - because look how many more lights they have. I've just got to make all those lights blink, Leo.

Leo: That is sweet. Now, we should warn people, this doesn't do anything you want to do.

Steve: Well, it...

Leo: It's not like - it's just a pretty thing.

Steve: No, it will run the OS/8 operating system, which was written for the 8. And it's based on a famous emulator. There's a sim, I can't remember the name, there's a simulator for many of these old machines that has been written. This is based on that.

Leo: So you're really running a simulator on the PDP-8, an emulator.

Steve: So, for example, you can connect a terminal to it and write BASIC and write FOCAL, which was the DEC language, or run...

Leo: Wow, that's kind of cool.

Steve: ...the assembler and editor. And, I mean, this is where I cut my teeth was literally writing assembly language, machine language, for the PDP-8, with those tools.

Leo: Wow. Wow.

Steve: So anyway, this brings the price of doing that way down and creates like a little museum piece in the process. So I wanted everyone to know. Anyone who felt like they couldn't get one of these cool little guys that I've got blinking behind me. And I will, of course, write some software for the big one so that it does the same sort of thing.

Leo: The kit does come with two wooden mount blocks. So I think you get everything you need. You do have to supply...

Steve: I see the case above it, at the top of the picture.

Leo: Yeah. You do have to supply a Raspberry PI Model A+, B+, or 2. Although he says the $21 model A+ is recommended. And then you need an SD card of 4GB or larger. You could put a USB hub on it.

Steve: If you can get one that small.

Leo: And a micro USB WiFi adapter. What? Can I surf the 'Net on my PDP-8 using links? I can't wait.

Steve: So I just wanted to say that one of my very favorite shows finished its third season, and that's "The Americans" on FX.

Leo: I know you love this show.

Steve: Oh, my lord. I just, for what it's worth, if you get to a point in the upcoming summer that you're looking for something, it's the story of two Russians who come over and take up residence and raise a family in Washington, D.C., in the shadow of the Washington Monument, and get up to no good. And now their teenage daughter has begun to suspect that something fishy is going on because her parents are not acting, like, you know, the phone rings, and they leave at all hours. And so she begins to recognize - anyway, I just - I don't want to do a spoiler. But the way this ended, this third season, makes the whole thing worthwhile so far. So for what it's worth, people have appreciated my opinion on these things before, so I just want to share one because I got one. And it is that "The Americans" is really worthwhile. And what's her name, Felicity is how I always know her, Keri Russell...

Leo: Oh, I love Keri Russell, yeah.

Steve: Yeah, and she...

Leo: She's great in this, yeah.

Steve: The acting is phenomenal. I mean, this is one of those really good cableproduced shows. Top recommendation.

And I talked last week or the week before about how I could not find the screenshot that someone had sent me for SpinRite. And then I found it in my Twitter feed because he had tweeted it to me. And so I wanted to just show it. So what I liked about this was his tweet was - and this is Sean McCormack, who said, "Thank you @SGgrc!" with a big exclamation point. And all of those green R's mean "recovered." And in order for - and all the little blues mean there was nothing wrong. Which is to say that, wherever there's a green R - and there's how many, there's like 10 of them. Wherever there's a green R, and it's only about, what, maybe, not quite, it was not halfway through, maybe a little over a third through, that means that that was a sector where the software, any software said I want to read that, and the drive said sorry, can't give it to you.

Now, normally that's where it ends. That's it. Game over. Not if you're running SpinRite. So those green R's mean that when the drive refused to give up its data easily, SpinRite went to work and essentially bore down and did all kinds of tricks, playing with a set of commands that are not normal data transfer commands, but can be used for data recovery. And this was stuff I developed years ago and have been refining a little bit along the way such that, in order to earn a green R, one or more sectors that fit within that space on the map, sometimes there's multiple sectors because they kind of tend to die in groups, have had 100% data recovery, meaning that we got all of the data and confirmed it was the original data that was written. Maybe we rewrote it back to the sector and then verified that we could now read it properly, so that anybody can now. Or maybe, in the process, the drive said, wow, this sector's in bad shape, and put in a spare. And then we rewrote the final data back into the spare.

Anyway, many people haven't actually seen what this looks like, but this was a beautiful case where this drive was in bad shape, and SpinRite has 100% recovery of many, at least 10, different spots where it was completely unreadable. It was just, sorry, error. You can't have your data there. That's SpinRite.

Leo: Pretty good. All right. I've got questions for you, Steven. All right. Let me open up the questions here because we've got some for you. And I'm ready to read them if you're ready to answer them.

Steve: Yup.

Leo: Numero uno, Bobby in Idaho. He appears to be a troublemaker. He wants more on, get this, brute-forcing encryption. Sure, Bobby. We'll tell you everything. Steve's glad to help out. In Episode 501 you explained how you know when you've successful brute-force broken encryption. Because it's actually checking against the authentication. So if encryption doesn't authenticate, how would you brute force it? Hmm, hmm, hmm?

Steve: Okay. So I answered a listener's question a couple weeks ago on this question. And I explained that proper encryption is always then wrapped with a layer of authentication because there are many ways to mess with encryption if it is not authenticated, that is, if the attacker is allowed to make a change, even to the encrypted data, there are well-known available attacks that can produce trouble. So you authenticate what looks like noise because it's encrypted. It'll be maximum entropy. You authenticate it separately to prevent from ever considering it valid if it's been changed.

Thus you know when you're brute forcing, you know when authentication works that you have found the right password.

So Bobby's saying, okay, if you don't have that outer authentication wrapper, then what? Okay? And the answer is heuristics. You literally look at what the output is from a test key and see whether it could possibly be correct. It's a little bit like what Turing did with the Bombe and decrypting the Enigma machine because it was looking for possible solutions. So the machine would stop when the constraints that they had imposed on it were met, and then they would look.

So, for example, say that this is, again, if you're decrypting something, you typically know what it is. For example, say it's email. Okay, email in a language, and you know what the language is going to be, is going to have, when it's decrypted, some very definable characteristics. For example, ASCII is a seven-bit code in an eight-bit byte. So the high bits will all be off. Now, the chance of using the wrong key and misdecrypting something such that a chunk of it comes out wrong, but with all the high bits off, is vanishingly small.

So a perfect example is you simply decrypt the first block of the cipher text using every possible key, checking just to see, given that you know that it's ASCII, given, you know, checking to see that all the high bits are off in your decrypted result. If not, you know it cannot be the right key. If they are off, then you can be a little more sure by checking the next block using that same key, making sure that they, too, are all off. At that point you pretty much know that you've got it nailed. Then, of course, look at it and see if it makes sense.

There's some vanishingly small chance that you would get the wrong key that would produce ASCII gibberish, but with all the high bits off. Probably won't happen. But if that's the case, you say, oh, that's like the Bombe stopping at a code that could have been possible, but then they tried to use those settings in an actual Enigma machine with a different message from the same day to see whether it would decrypt properly. If it did, they had it solved. If not, they pressed "keep going" on the Bombe, and it picked up where it left off. So basically, brute force without authentication means you just have to brute force. Which means you look at each output and apply some heuristic to say, could this be possible? And if it could be, then it stops, and then someone human looks at it and goes, uh, no, keep going; or, yeah.

Leo: You'll know.

Steve: Yeah, exactly. You'll know.

Leo: You'll know.

Steve: You'll know when you get it right.

Leo: I love that seven-bit in an eight-bit byte because, I mean, that's great. That could be done by a computer very quickly.

Steve: Yup.

Leo: And do we see any eight-bit set? No? Okay.

Steve: Got it.

Leo: Question 2, Brian Tillman in Wyoming, Michigan - by the way, that'd be a fun exercise, is what would be a quick and simple effective test against various kinds of data to know you've done the brute force correctly. Wouldn't that be fun?

Steve: Yes.

Leo: Kind of like a little puzzle? Brian Tillman in Wyoming, Michigan is haunted - haunted, I tell you - by the idea of malware in disk drives: Back in Episode 496 or '7, you related evidence that had been found outside of disk drive firmware of malware that deliberately alters said firmware to malicious ends. Following up in the Q&A later you elaborated how it was not practically possible to know whether a drive's firmware had been infected so.

But drives are usually made in countries whose governments are extremely interested in learning U.S. business secrets. I see these devices - hard drives, flash drives, memory cards - as a great way to implement industrial espionage. Who knows what might be embedded in these drives, and how would we even know? You said we wouldn't. No one reverse-engineers the majority of these devices. Their designs are considered closed and proprietary. They could be collecting data or doing anything. We wouldn't even know. I'm haunted, too, now. What is your answer, Steven?

Steve: We're in trouble. No, you know, this is - through this podcast I have often commented how odd it is to me that, for example, the Chinese government and populace use Windows that comes from a U.S. company. And we've talked about problems where routers were being infected, either by unknown agents or, unfortunately, by our own NSA, in transit, in order to have them modified. I guess I'm sort of bemused.

You know, right now at this point in time we're all - the whole Trans-Pacific trade agreement is in the news, where the President is negotiating to try to figure out how we extend trade and increase opportunities. And some of the people that we're trading with, as Brian notes, are diplomatically, politically hostile. Yet one of the things we're doing is we're swapping hardware that we cannot see inside of, that we can't see into. There was an article I ran across about a hand scanner that was coming from China that had malware embedded in it such that it waited until it got plugged into a certain type of system, and then infiltrated the network. And, I mean, so this is not science fiction. This stuff is being found. So the problem with drives is that their firmware is not readable. So we don't know what's in there. And the drive manufacturers consider their firmware proprietary.

Now, maybe at some point in the future the technology will evolve to a point where we'll end up with something like open source drive firmware, the way we sort of had open source everything else occurring over time. It used to be OSes were all closed source. Now very good ones are open source, and so forth. So, I mean, and that might happen because, if this becomes a big enough problem that people say, look, we're not going to buy a drive whose firmware we can't have vetted by a third party because there is just too much opportunity for real mischief.

But the fact is we're in this phase now where the value to the attacker is rapidly increasing, and our true responsible management and oversight is hugely lacking. It's, I mean, we're taking so much on faith that this stuff is what it says it is and isn't more, and we have no way of verifying it. I mean, I can't - of course the famous expression is "trust and verify." Well, all we can do is one of those two. And we really are just hoping for the best.

Leo: There's trust in everything. We've talked about this before. You can't drive down the street without trusting that the other guy's not going to cross the line and slam into you.

Steve: Yup. Yup.

Leo: The problem is we're trusting people we don't know and don't trust.

Steve: Yes. And at a scale. That's the other thing, too, is if someone is suicidal and decides to take themselves out with your car, well...

Leo: It's limited in its impact; right.

Steve: They've taken you out, too. But now they're done. Whereas, if a nation-state is able to infiltrate a vast number of networks, I mean, the problem is these things really scale, these sorts of attacks.

Leo: Michael S. McElrath in Flint, Texas - sounds like a flinty place - worries about SSD warranty returns. I have an issue most users may not consider when returning warranteed solid state drives: They are prone to just quitting without any notice. No further access is possible. Bam. Gone. My dead SSD has my personal/company tax filings, my personal files including bank and credit card information and my customer files; facility layouts, production data. In other words, crucially privacy-sensitive content. Now it's dead. For the $200 cost of the drive, would you return it on the assurances that they will erase any information they find? TNO. SSDs must be encrypted at all times, especially during the warranty period. Thank you. Michael in Texas.

Steve: So it has been my experience, and I know it's been many people's experience, that SSDs, very much like hard drives, can fail in either of two ways. They can fail slowly, or they can just turn into a doorstop. I have seen SSDs that just, as Michael says, just bang, they're just gone. It stops being an SSD. Hard drives, notoriously, if they have a head crash or the servoing mechanism dies, I mean, there are catastrophic ways for a hard drive - did I say the spindle motor just doesn't spin anymore, or the heads stick on the platter and refuse, with what's called "stiction," refuse to allow the drive to spin up. Again, no more data coming off of that drive unless you can break that contact with the heads and the surface.

So Michael's point is, and it's a good one, is that something to consider is, if you intend to avail yourself of an SSD's warranty, there's all the more reason to apply external encryption. He didn't mention this, but drives today will have the "security feature set," as it's called, which allows you, like typically the BIOS, to give the drive a password which will then encrypt the contents of the drive. The problem is that can be removed at the factory. So it's a weak solution, one that specifically, if you're worried about the factory warrantying the drive and not poking around in it, you can't use. You need TrueCrypt or a similar add-on whole-drive encryption solution.

And so it's just something to consider, that, again, if you want to take advantage of the warranty, if you're worried that a dead drive - which of course it's not really dead. They can certainly look at the chips and get the data off it, exactly as Michael suggests. It's just dead to the interface for whatever reason, who knows why. You might consider that it's worth putting your own external encryption on that drive, whole drive encryption that you add, because then it's just a drive full of noise.

Leo: You know, people may say, well, I'm not going to worry about warranty returns on my SSD. But you know where this really hits is on your smartphone. Because how often, I mean, it's not unusual at all that your smartphone dies, and you bring it back to the store, and they say, "Oh, yeah, sorry, here's a new one." Well, all your stuff's on there.

Steve: Right.

Leo: So if you're using an iPhone, you're okay, right, because Apple encrypts.

Steve: Right, yup.

Leo: But on Android devices, not all of them encrypt by default. Might be good to turn that on. Nexus 6 does, but others may not.

Steve: Yup, that's exactly right.

Leo: And it's the same issue, exactly the same issue.

Steve: Yes, yes.

Leo: Maybe more so.

Steve: And unfortunately there are a lot of immature people who would get their jollies from poking around in someone else's business. I take great pride in the fact that I, back in the day, the early days of SpinRite, I would sometimes receive customers' drives who had problems beyond what SpinRite could do. Their file system was messed up, and they absolutely had to have something. I never looked in. It just doesn't interest me, as a point of pride. But I know that there are people who have a different approach, unfortunately, and they just get a big kick out of that.

Leo: Question 4 comes from Andy Martin in Los Angeles. He wants a quick message encryption tutorial: Steve, I've been listening to Security Now! for two years. Oh, there's your problem, Andy. You need at least eight more years. You understand encryption, but I'm more interested in the programming behind it. Have you ever written a tutorial on how to implement secure messaging? It seems so easy to do right, but it appears no one wants to take the time to do it right. Surely there is an open source library out there that makes this easy.

This is the flow I would want: Imagine that I manage the server, and I want no way to decrypt any messages. Every client generates their own private key and public key. They push the public key to the server. Now Client 1 wants to send a message to Client 2. The message has to be authenticated, then encrypted; right? How can the client encrypt it so that the other client can decrypt it? It seems that just knowing the public key is not enough because then anyone with the public key can decrypt it. Mmm, no. And frankly, I don't know what "encrypt with public key" even really means at the algorithmic level. I think you need a few more years, Andy.

Anyway, then the client sends the message to the server, which routes it to the other client, who could then decrypt and magically authenticate it. Help! If the client ever lost their private key, they would just need to re-make a public key/private key pair and push it to the server again, losing any messages that might come while they had not yet posted their new public key. I think he needs some help, Steven.

Steve: So, okay. We have discussed this at length, and I don't want to go over the whole thing again. I will just say that the server, as Andy notes, holds all the public keys. And so, and the server then is, as it's called, a "key server." So if Client 1 wants to encrypt something from Client 2, you first choose a random number. So you need a good source of randomness, "entropy" as we call it. And you use that as the symmetric key for encrypting the bulk because public key crypto is so slow that it's never practical to do bulk encryption with it. Instead, you use the public key encryption only to encrypt this random key that you generated just out of thin air and used to encrypt the bulk.

So the idea is that you have, from the key server, you have the target's public key, which means that something you encrypt with their public key can only be decrypted with their private key. So you take their public key, which you receive from the key server, use it to encrypt the key, the symmetric key that encrypted the bulk of your message. And you could also, for example, use your private key to sign the result. Then you send it either to the server to relay or directly to Client 2, whichever makes more sense. Now Client 2 has it. They get your public key from the key server, which allows them to verify the signature on it, which could only have been made by somebody having Client 1's private key. And so they verify that. Then they've got...

Leo: So key understanding, fundamental concept, the public key, which can be distributed widely, freely, publicly, can be used to do two things: to encrypt and to verify. Right?

Steve: And to decrypt.

Leo: Public key?

Steve: Yes, yes. Because...

Leo: Don't you have to have...

Steve: Oh, no, I'm sorry, to...

Leo: Authenticate and encrypt.

Steve: ...encrypt and verify the signatures, right.

Leo: Yeah. That's important. The decryption requires the private, closely held, only I have it key.

Steve: Exactly.

Leo: You don't let that one out of your sight.

Steve: Exactly. And so that's what Client 2 does after verifying using Client 1's public key that it actually is coming from Client 1 and has not been modified because authentication will verify that. It'll both verify who signed it and that it has not been modified. And then Client 2 uses their private key in order to decrypt that symmetric key, which is that random number, that big random key that Client 1 generated. And then that they use to decrypt the message. So that's the whole flow.

Now, because Andy's right, there's a need for a library, it has been created. Dan Bernstein, world-famous cryptographer, has something called NaCl, which is a library. Unfortunately, it is not broadly cross-platform, but it has been taken and extended and made absolutely cross-platform. And that's called "libsodium." So GitHub has libsodium, L-I-B-S-O-D-I-U-M, which is basically an API-compatible recoding to be cross-platform of Dan Bernstein's NaCl. It has a simple API that does everything anyone wants. It uses the state-of-the-art, efficient, elliptic curve crypto. And it's got functions like encapsulate this message where here's the public key of the recipient, the private key of the sender, and I want you to encrypt it and authenticate it and give me the result. It does all the work for you. It's been heavily scrutinized. I'm using several of the functions for SQRL, specifically the digital signature stuff because that's what we need.

Anyway, that's what you want to use, libsodium from GitHub. And there's something also I just discovered called GitBook, which is for eBooks. And there is a beautiful libsodium book on how to use libsodium in the GitBook.com repository. So, and it's available in all kinds of eBook formats. So there's no excuse. Use that library, read that book, you'll have everything you need, Andy.

Leo: What about OTR, Off The Record? Because that's used often for messaging.

Steve: Right. So there the requirements are different because that's a real-time protocol.

Leo: Ah.

Steve: And you need to have a real-time interchange between the endpoints.

Leo: Right.

Steve: This system that we just laid out is completely static. I mean, it's very much like the original PGP model. This approach has been around now for years.

Leo: Right, right. And it solves all the criteria that Andy laid out for what he wants.

Steve: Yup.

Leo: And the thing is, if you share your public key, as I do, and I don't know, do you share your public key? You should.

Steve: Don't have one.

Leo: You should make one. I'm going to send you an invite to this place, Keybase.io. It's JavaScript based, although if you prefer you could do it at the command line. That's what I do. It is a way to - you know what? It solves the problem of the key signing. Because one of the problems with PGP is it's a self-generated key. So you want to create a web of trust of people who've said, yes, that's Leo's key, I know that. This kind of solves that problem. You don't have to go to a key-signing party. But you use your other things, like your GitHub or your Twitter or your Reddit account or your website, to validate that, yes, that's me, THE Leo Laporte. And then on this site I've got my key. So you can get, from this, you can actually get my public key right there and add it to your keychain. And that way you can do two things. Again, you can encrypt to me, not decrypt, but encrypt to me, and authenticate messages from me and say, oh, yeah, Leo sent it. It's unchanged."

Steve: Yeah, I guess I just don't have the need. I mean, I've never...

Leo: Well, I'll tell you why you might. Well, we talked about miniLock. I guess you could use that.

Steve: Right.

Leo: The idea that maybe somebody would want to send you a private message. And since you use Twitter, a public messaging system, they could use encryption to do that over Twitter.

Steve: Yeah. But again, I mean, no one ever has.

Leo: So there you go. If you don't need it...

Steve: I just don't have the problem.

Leo: ...you don't need it.

Steve: No.

Leo: That solves that. Thank you, Andy. Jim in Grand Rapids coming up with an update on PCIE SSDs and AHCI/NVMe. Whoa, well. It's about time. Actually, let's do it now. I can't stand the suspense.

Steve: It's a quickie.

Leo: It's a quickie. Two weeks ago you had a listener question about SpinRite and SSDs on the PCI Express bus. I know I'm the 8,000th person to point this out, but while most PCIe SSDs on the market now use the AHCI interface, the newest drives are using something called NVMe, or Non-Volatile Memory express interfaces. Just a heads up.

Steve: So, yes. Apple, in fact, is using the new NVMe on this latest MacBook because it turns out that AHCI, powerful and fast as it is, is not as fast as you can make an SSD. That is, if you squeeze an SSD through a SATA interface, an S-A-T-A interface, they can go, what, is it 6Gbps is as fast as SATA3 will go. But an SSD, I mean, we're inherently kind of coming from a spinning media mindset, where the SSD, solid-state disk, I mean, it's even got "disk" in its name, it sort of says, okay, there's all these disks in the world with this SATA interface. I want to be plug-compatible with the SATA interface. So that's the way SSDs got into the market. And they were faster than spinning disks, but they still had this serial notion, that is, that the data would be transferred serially. So even at 6Gbps, it turns out there's no reason for it to be serial. Essentially, it's just RAM. You can have it all, right now.

So what the designers - and this has been a few years in coming. There's an Intel spec, this NVMe interface. Essentially, whereas AHCI, the Advanced Host Controller Interface, is serial, NVMe is parallel. You can have multiple queues. You can ask the - I keep saying "SSD." You can ask the NVMe interface storage device to just sort of say, you know, give it everything it has. And it's just, blam, there you go.

So people have asked, "What about SpinRite?" And my answer is that the way I have already designed the 6.1 that I've got running is that it is multi-interface. And it enumerates the PCI bus to determine what's there and then uses the driver it needs. I don't know exactly where in the cycle I'm going to do this. 6.1 adds many features. But for compatibility, AHCI, it doesn't use the - I'm trying to think what - the Mac. I've already got it running on the Mac keyboards, which was a problem for SpinRite. And it will understand the GPT partition table. I don't want to slow 6.1 down to add anything else.

So I'm committed to getting 6.1 out ASAP. But the whole point of the 6 series is to create a foundation for 7. So I also need to support USB natively, rather than through the BIOS. So that has been slated for 6.2. So I don't know where NVMe will fit. Maybe it'll be 6.2 instead, or maybe it'll be 6.3. But I will not stop working on SpinRite, believe me, because my goal is to catch up and have it running perfectly screaming speed on everything. So the 6 series, which will follow one after the other, will support NVMe. And it turns out I've already dug into it, and it's not difficult to do. So SpinRite'll have it.

Leo: M. Weber, who is on the move, wonders about the WiFi location confusion: My plane from Orange County - oh, I've had this happen - landed in Dallas. And while we were still in taxi to the gate, I pulled up a map on my Android phone to get directions from the rental car facility to my client's office. To my shock, to my horror, the driving directions told me it would take more than 20 hours to drive there. Once I got over my shock, I realized the map was showing my starting point in San Jose. Then I remembered your discussion from a few weeks ago of how the location service uses the available WiFi devices to establish coordinates. I turned off WiFi, ignored the location service's begging me to turn WiFi back on "Because it's so much more accurate with WiFi." Hah. And problem solved. I guess someone's portable WiFi hotspot had been pegged as stationary. Seems like a pretty big flaw in a system that we are more reliant on. Thanks, Steve, for a timely and practical show. I have an alternative explanation of that, by the way.

Steve: Which is that it knew where he was?

Leo: No, that he was on the WiFi of the plane. And often the case, especially with commercial WiFi providers, I've had this happen on cruise ships, it identifies its location as the home office.

Steve: Interesting.

Leo: So he was using Boingo on the plane, Boingo WiFi, which is in San Jose. And so it says that's where we are. On the cruise ship, I kept going back to Venice. I'd be in Croatia. It was amazing. In, like, 30 seconds I could go back and forth and back and forth. So that's kind of not really...

Steve: Well, and the point I wanted to make was that this is an imperfect system.

Leo: Yeah.

Steve: GPS is perfect, but we don't always have it. And cellular...

Leo: Yeah. You're in a plane, dude.

Steve: Right. And cellular is another means, is a more reliable means, but doesn't provide the granularity that we would like. So what's happened is, in a classic sort of let's use a heuristic, we are doing something which could fail, much like the heuristic I suggested for brute-force cracking encryption. It could give you a false positive. It could be wrong. You could get the wrong key and still by some miracle have all the high bits off. Very unlikely, but possible. Similarly, hotspots do move around. So, you know, it's probably better than not using it. But you would hope, for example, that the software would look at all the hotspots in the area and see whether, first of all, they make sense. Like does it make sense that all of these are in the same relative proximity to each other, and then reject the one which is clearly some sort of a crazy false positive, and then go from there. So maybe the software could have been smarter. But it's a classic heuristic.

Leo: Well, if you think about it, if you're in a plane, and you're enclosed in metal, you probably don't have access to other WiFi, just the plane's WiFi.

Steve: Right.

Leo: GPS may not be working because you don't have line of sight to the sky. So it's going to use what it's got, which is the location of the WiFi router, which is back in San Jose. It doesn't - the router in the plane doesn't identify where it is. It identifies where the Internet's coming from or whatever, where the company was. And Boingo, of course, is in San Jose.

Steve: Right.

Leo: So I think that that's probably what it was.

Steve: Yup, makes sense.

Leo: It's a great conundrum. And I bet you, as soon as it got a GPS signal, it would reject the spurious WiFi signal.

Steve: Yeah, yeah.

Leo: And certainly by turning off WiFi you did that. You said no, no, no, no. Look and see where you are.

Steve: Bad information coming in here.

Leo: Right, right. Let's take a break, come back with more. You have already agreed, by the way, I hope, to supply some cute little tips for The New Screen Savers?

Steve: Absolutely, I have.

Leo: We can record those after the show some week and just, you know, they can be little things like, you know, turn off your whatever, WiFi, or don't, I don't know, whatever it is that you think would be important.

Steve: Yeah. I'm chatting with...

Leo: Jerry or Karsten or Lisa or...

Steve: A gal. I'm blanking on her name.

Leo: Oh, Tonya. We have so many producers on the show.

Steve: I'm chatting with Tonya tomorrow morning about doing that.

Leo: Excellent. And then after a show some week we'll record those. The New Screen Savers launch is May 2nd, right after the radio show, about 3:00 p.m. Pacific, 6:00 p.m. Eastern time.

Steve: Talk about a lot of buzz. People are very, very excited.

Leo: They're very happy. Lot of buzz going on in the Comcast headquarters, too, apparently, but that's another story. If you have an idea for the show, or we are actually looking for questions for our Help Me! segment, you can email newscreensavers@twit.tv. And don't forget, we're going to be live, 3:00 p.m. Pacific, 6:00 p.m. Eastern time, 2200 UTC, Saturday, May 2nd. And we decided - this is a great opportunity. One of the reasons we're doing it, we wanted to do a variety show so we could have little tips and stuff and bits from everybody. Our experience on New Year's Eve was what told us that.

Steve: How did the dry run go on Sunday?

Leo: Great. It's going to be a great show.

Steve: Yeah.

Leo: You know, it's going to be so much fun. Everybody's, you know, people have wanted me to do this for 10 years. And but now we have the horsepower to do it, I think. So I'm excited. New Screen Savers at TWiT. Actually, you know, it says "newscreensavers," but I think screensavers@twit.tv is actually the email. I think this is wrong. Do screensavers@twit.tv. Unless we've got two addresses.

All right. More questions for Steverino, starting with Patrick in Central Minnesota. He wants to remove RC4. I want it gone. What? In the last two podcasts, you've described the unfathomable realization that many banking sites, like Bank of America, are using the bad RC4 cipher as their main communication medium. You also described how a browser offers a list of ciphers to the bank's server, and then it chooses one, in theory the best of the bunch, but apparently sometimes the bad one, RC4. So can I just remove that from the list? In fact, let's take all the weak ciphers out of the list so my browser just says, hey, I don't do that. And then Bank of America will be forced to choose a better one. If I can't get rid of RC4, can I configure my browser to not even mention it in the cipher list? Seems like this would solve a lot of problems. And if a site only accepts RC4, well, I don't want to talk to it anyway.

As a side note, I've been a listener of Leo's shows since day one. I got hooked on Leo from Tech TV until that channel went belly-up without any explanation as to why it was suddenly gone. Oh, I could tell you some stories, my friend. Every so often I'd google him to see what he was up to, and then one day he made his first podcast with some of the old crew. Those first ones were kind of crazy, but his personality always carried a lot of weight on the show. Thank you. Wow, he's come a long way. I'm sure glad you two found each other. Thank you for many, many years of "netcasts," with a wink to Leo.

Steve: So, okay. So first of all, I realized why banks had RC4.

Leo: Oh, why?

Steve: First in the list. Not quite as unfathomable as I had been saying.

Leo: Probably has something to do with IE6 or something.

Steve: Actually, it has to do with some of the attacks that we found, like BEAST.

Leo: Ah.

Steve: Which were attacks on the block ciphers. Block ciphers like, well, any of the block ciphers, like AES, use some sort of a mode like cipher block chaining mode. And as we've covered on the podcast, various little nicks have appeared in those, where if the attacker has access to the communications and can generate lots of bandwidth, there are games that they can play that the block cipher modes are specifically vulnerable to, BEAST being the first one. And our listeners who've been with us the longest will remember when BEAST was revealed, the recommendation was move RC4 to the top of the list.

Leo: Oh.

Steve: Because RC4, while it's an old creaky cipher, oddly enough, there's no known attacks against it. There was a paper maybe six months ago where the keying of RC4 was further brutalized. But there are actually no known attacks against RC4. Whereas now, with BEAST and then Lucky 13 and then POODLE, all of these are attacks against the block cipher. So this sort of leaves us in a conundrum because people feel nervous about RC4. People just don't like it because it's so simple. It's actually one of the reasons I do like it. And if you just warm it up further, then it scrambles up its starting state. And then it's a fantastic, very fast cipher. In fact, some of the original designers of RC4 have done a - have, like, fixed it. By just making a very few changes to it, they have strengthened it. But people are just not liking a stream cipher. What's different is that it is a stream cipher. It produces a pseudorandom bitstream which you XOR your plaintext with to get the ciphertext, which is different from a block cipher that takes your plaintext in chunks of bits, in blocks, and then mutates that block into a different block, and then does some fancy interblock linking in order to create the so-called "chain."

So, I mean, the dilemma we have is that we want to promote state-of-the-art stronger ciphers. The later versions of TLS have mitigations against these block ciphers. So people who have TLS 1.2, for example, are going to be okay. But people who don't then have a problem. At the same time, all the browsers quickly added mitigations, you know, prevention for like the BEAST attack. But not everyone is known to be using the latest browser. So what's someone to do? Weirdly enough, there is no good solution.

Ivan Ristic over at SSL Labs is right in the middle of this because he's trying to give people a letter grade, A through F. And he's decided that, if you use RC4, if RC4 is present, you cannot get better than a B because he wants to encourage people to remove RC4. GRC has an A now, and we don't support the RC4 cipher. There just is no reason to. So what banks should do is remove it, except then the concern is that there may be some creaky old person browser something somewhere that still needs it. I doubt that's true. The problem is, as long as everybody is worried that removing it might break something, nobody at either end will remove it. And then, if the bank puts it first in the list, that's what you're going to use. At the same time, there's actually nothing wrong with using it. There are no known attacks against it, against the cipher itself. And the browsers have solved problems where the browser, where the client has been able to.

So we're sort of in this weird place. For what it's worth, it looks like Windows is the only operating system that through some registry manipulation will allow you to turn off RC4. I haven't bothered to do it, but I did some digging around. And if you just google, like, "removing cipher suites from Windows," that will take you to a page where Microsoft explains it. You go into the registry, you flip some bits, and your system will no longer run the RC4 cipher. It'll just remove it from the list. Now, Firefox won't get the benefit of that because it's got its own security suite. Once upon a time it had a way of doing that, and they removed that. So there's nowhere, there's no way now to take it out in Firefox.

Leo: Ah, not so fast.

Steve: Oh, yeah?

Leo: It looks like Aurora, the development edition, has added that. In fact, "disables use of RC4 except for temporarily whitelisted hosts."

Steve: Nice. Nice, nice, nice.

Leo: So it's back, baby. This is Aurora, is, what, the developer edition; right?

Steve: Nice. So we're right on the edge of making this transition.

Leo: If you use Firefox or Windows.

Steve: Yeah.

Leo: I bet you Google and others will follow suit; right?

Steve: Yeah. I think people are just getting to the point where it's like, okay, it's just time to stop using it. And what Ivan is going to do at SSL Labs is, I think by September of this year, if you still offer it on your server, you get an F.

Leo: Good.

Steve: He's going to deprecate servers this summer to a D. And if it's still there in September, F. It's just time to yank it because what you really want to do is just stop using it. Just let's not use it. Then we don't have that whole issue.

Leo: Chatroom has also given me a link to this Microsoft security advisory update for disabling RC4.

Steve: Nice.

Leo: Not sure when this went out, but it was for all versions of Windows from 7 and up. And it gives you some information, as you said, for what to modify in the registry to completely disable RC4.

Steve: Yup.

Leo: So you can do that, as you said.

Steve: So it can be done.

Leo: Yup. Good news. Actually this - I'm really glad you had that question. Very interesting. Tim Trott, Marianna, Florida, continues the conversation, wonders about SSL and a shared server: SSL requires a dedicated IP address. I run a server at Cyberchute.com where all except e-commerce sites are on a shared IP address. I've been assigned 16 IPs for a total of 185-plus websites. Wow. Will I be forced to obtain IPv6 assignments, which cost money at Rackspace, for my remaining shared IP hosted sites in order to give them each their own IP address? Some SSL certs cost more than the retail price for hosting, so I will welcome Let's Encrypt. That's the free thing we were talking about.

Steve: Yup.

Leo: And will that require a dedicated IP? And what will happen to self-signed certs, which I of course have never used, he says.

Steve: Okay. So here is the story. It is no longer the case that SSL requires a dedicated IP. It used to be true because the negotiation with the certificate took place before the encrypted TCP tunnel was first used. So what you had was an IP-to-IP connection where you were negotiating SSL and needed to verify the server's domain name. That's why you had to have - that's why the server had to assert its certificate, its domain name, based on the IP that you were connecting to, since it wasn't until that was done that the web server could then make a query with a hosts header and say, hi, I'm hoping that I'm going to www.google.com. So consequently, the IP address had to be bound to the certificate. That changed with TLS.

There's something called SNI, I want to say server name - I'm blanking on the acronym, SNI, Server Name Indication. And that is an option which has always been available, starting with TLS, which solves this problem completely. And it was somewhat worrisome maybe five years ago. But everybody now supports at least TLS 1.0, which is essentially SSL 3.0, but, again, has additional features. TLS 1.0 and on allows the initial handshake from the client to the server to have an extension field saying I'm going to be connecting to www.google.com. That is the domain name for the first time ever. Not just the IP is in the handshake. Which allows a multi-homed server hosting many different websites to recognize that extension field and then supply from an array of certificates that it has. Go look up "server name indication" on Wikipedia, and you'll see a list of all the browsers and all the servers that support it at each end. Basically, everybody supports it at each end. So it is now something, Tim, that you are safe to use. So you could put all your commercial sites with free certificates from Let's Encrypt behind one IP if you wanted to.

Leo: Yay. And an update, we were talking a little bit about the challenge we were having going HTTPS on TWiT.tv because we use so many different...

Steve: Caching.

Leo: Caching, well, it's not just caching. You know, I totaled it up. We have a CDN, that's Cachefly. And of course we will be pulling from that if you want to watch the video or listen to the audio on the website. We have an API server, Apiary. The website is served from a Node.js hosting service called Heroku. And the API itself is served from a Drupal host called Acquia. So at least four. There may be others involved. However, thank you for bringing up the issue because I did - at our scrum last week. The next day I said to the developers...

Steve: What's the story?

Leo: Yeah, hey, you think we could go HTTPS? And they very strongly said yes, we should do that. Matt, who's great, a really a good programmer, said, "Yeah, I strongly encourage you to do that." And they came back, and they said, "We can do this. It's not as hard as we thought." You know, he said, "Let me look into it because, yeah, you're right, it raises some issues." Not as hard as we thought. There's some chaining or something. And it's going to cost us another $2,000 for the time involved in doing that, which is nothing, as far as I'm concerned, 1% of the total cost, less than, and well worth it.

Steve: Nice, nice.

Leo: So we will be HTTPS everywhere.

Steve: Yay. Using DigiCert EV certificates.

Leo: With a DigiCert EV, which is really neat.

Steve: Wonderful. Wonderful.

Leo: Yes, and we'll be green. We won't just be HTTPS, we'll be green, baby.

Steve: Yup, baby.

Leo: And proud of that. So it was great. The developers said "Yes, thank you for asking, we will do this." Yeah, so that was good. Joe Laba, Question 9, in Metropolitan Detroit

says, "But, but, but..." of TrueCrypt: Steve, apparently I missed something somewhere. I remember you saying that there was no legal way for anyone to fork the TrueCrypt source code. But it sounds like someone - not just one, many - are going to be doing just that. I think it would be great, but what happened to make this possible? It was - TrueCrypt, we did point out, is not actually open source.

Steve: No, it's not. It's open license. And people are just doing it anyway because...

Leo: What are they going to do?

Steve: It's there, yeah.

Leo: Come out of the dark and sue you?

Steve: Yeah, I mean, it is absolutely true, the letter of the law of the TrueCrypt license says this is yours to inspect, but not to modify. And so they were making it available in exactly in the open source spirit that you've often talked about Leo, of we need to be able to inspect it. And doubtless they were helped by people over the years finding problems in the source. And look at the audit. It found some problems. Nothing major, but better to have found them than not, only because it didn't need to be reverse-engineered. It was open source. The license says you can't change it. Well, the developers also said, and we're going away, and we're going to remain anonymous, and you're never going to find out who we are, and we're not supporting it any longer. So people are like, well, fine, we're going to take it and keep it alive because...

Leo: And my suspicion is the existing developers would embrace that.

Steve: Yes. I think, again, there's their officially stated policy, and there's their, eh, fine, you know, we made it very clear we are disassociated with it. Nobody come crying to us.

Leo: They're done.

Steve: Yup.

Leo: But that's a good question because we did say that. We were talking about that.

Steve: Yeah.

Leo: Last question.

Steve: And it's been reaffirmed.

Leo: Robert Lowery of Kansas: Steve and Leo, thanks for the great podcast. I've been a listener for blah, blah, blah. SpinRite saved my bacon, blah, blah, blah. Fan, blah, blah, blah. I want to turn the tables a bit and ask Leo a question: You occasionally mention that you enjoy programming, and you're obviously able to converse with Steve about some fairly in-depth programming topics. I'm curious what you use your programming skills to create? What languages are you most comfortable using? If you're going to get up to speed on a new language, how do you approach learning? Have a great day. Get out and grill. Believe me, Robert, I am. Robert in Kansas. He's obviously a beef farmer. Well, we know you love assembly language.

Steve: Yup, that's my language.

Leo: But you're a professional programmer. I am the farthest thing from a professional programmer. I'm a hobbyist. I love it. I did write some software that was relatively widely used, but it's been many moons, 20, more than 20, almost 30 years ago, in assembly language, for the Macintosh, for the early Macintosh.

Steve: That's right, 68000.

Leo: I was running a BBS, and one of the first - I wrote two things that I released to the world, open source, by the way, before there was even open source. I just made it public domain and published the source code. Yeah, 68000 assembler is beautiful. But I love C. I learned C. And somewhere, yeah, I have my Kernighan & Ritchie right here. This is actually - this is not the original. The original is so beaten up that I actually bought a second copy of this. So C was my first - BASIC was my first language on Atari. I learned assembly, which assembly's beautiful. I agree with you. I love assembly. And there's something magical about getting down to the how the machine works.

Steve: There's just nothing below it.

Leo: Yeah. And it's useful, very useful, in understanding that. And then C I learned. I love Forth, believe it or not, but that's when it started becoming a hobby; right? That's when it was, like, I just collect these. And I do collect languages. And I have books on every possible language. You know, once you learn C, you could pretty much learn anything.

Steve: Yeah. Although Forth, Forth is a write-only language.

Leo: Wrong. Forth can be written so that it looks like English prose.

Steve: Okay.

Leo: Because the essence of Forth is you create your own primitives out of Forth primitives.

Steve: True, true.

Leo: And you can write a sentence in Forth that actually...

Steve: That is true.

Leo: It was created by Charles Moore for controlling telescopes. And I think it's still used...

Steve: Yeah, astronomy.

Leo: Yeah. But I loved it. It was a stack-based language. It was just wild architecture.

Steve: And there's, like, weird - it pops up. Like there's something to do with a UEFI has like a Forth interpreter in it because...

Leo: Yeah, well, Forth is good in embedded because it's compact.

Steve: Yes, yes.

Leo: Very compact. So most languages are like C, they're imperative languages. But you go back to Forth and even earlier, you go back to Lisp, these are languages that are very different. And so I've, believe it or not, I've been learning Lisp, as my new thing is Common Lisp. I decided I wanted to go back to the beginning, a language as old as I am.

Steve: Yeah, well, and in fact Lisp in particular has a really rich heritage. I mean, it is a...

Leo: Oh, it's amazing.

Steve: And for it to have continued the way it has for so many years, I mean, because it is in a class of its own.

Leo: And it's interesting because it actually has started to influence modern languages. You know, everybody followed the C root. But as time has gone by, many elements of Lisp, and that style of programming...

Steve: And as computers get more powerful, I mean, one of the reasons we have C is that it was written on one of the PDPs, it was written on the PDP-11, when you needed -

really the whole concept was the smallest increment above assembly language that was machine independent. So for me, that's why it's my second language is it is, well, and in fact I wrote a big chunk of SQRL in C because it needed to be cross-platform. I implemented the GCM authenticated encryption technology. And because there wasn't a public domain library and open source, I created one and made it available to everyone because you need that as part of SQRL. And C was my choice for implementing it because it needed to be - I needed to offer the source. And for it to be able to be compiled for iOS or Android, you know, the ARM platform or whatever.

But what I loved about C was these guys, they first wrote the OS in assembly language, PDP-11 assembly language. And then they said let's recode this in something that is really, I mean, just like the smallest step away that gives us more expressability. We can do expressions. We can do the sorts of things, flow control that is more elegant. And that's C. But then again, that was because they were still dealing with very small, very low-power mini computers. Today, we just have so much more power to do much more powerful languages, dynamic languages.

Leo: I think, though, the point, really, is that the language - so all languages are what we call Turing complete. They can all be made to do the same thing. But there are some languages, for whatever reason, maybe my personality, yours, or whatever, that we just kind of get better, and we're more fluent and expressive in. And that's what you're kind of looking for, if you're a programmer. And I have to say Python, I love Python. I used Perl for a long time, wrote a lot of little bits of utilities and grep stuff in Perl. I love Ruby. Ruby's gorgeous. It's kind of, after Python, Ruby was the next logical step. There's more modern languages. Go from Google is really great for concurrency. Each has its kind of merits. And Haskell, there's a great book called "[Learn You a Haskell for Great Good!]"

Steve: That's a wacky language.

Leo: But it's really - but it's, by the way, it's going back to Lisp. That's what's so interesting. So I finally said, you know what, I learned Scheme, which is a derivative in Racket. But I want to go back and go to Common Lisp. And, by the way, it's really fun. And it's actually very easy to learn.

Steve: And it's available everywhere.

Leo: It's free. In fact, you know what, I'll show you, I'm using Emacs to do it because Emacs is written in Lisp. Emacs is the programming language - or actually it's really a lifestyle more than an editor. Richard Stallman wrote it, and he wrote it in Lisp. But I'm using a thing called Slime, which is a mode for Lisp programming, makes it very easy. So I'm actually in what's called a REPL, a Read Evaluate Print Loop, that lets you enter in code and execute it immediately. In fact, you see I had a typo in here. It dumped me out into the debugger. This is the debugger. You can't see it because the color contrast is bad. But it looks fine on my screen. And it tells me, oh, you've got a typo. So let me go back here. Let me go back up in my Emacs. And the problem is this parenthesis. So I've already entered it in, but I'm going to reenter it. I'm going to make that a brace. Hit return, it's going to put it back down there, it's going to execute it. It warned me, it says you've redefined this function.

That's all right. I'm in Emacs, executing Lisp, which is awesome.

Steve: Nice, yup.

Leo: It's awesome. And so this is an IDE, in a way, like a modern IDE, in a very old-fashioned form, in essentially command-line. So, now, he said how do you learn? There's some really great - the web is wonderful now. Look for - there are series of books that teach a variety of languages. "How to Think Like a Computer Scientist" is one, and they have every language, although it was, I think, originally Python. And you'll find these online. If you look at my programming folder, I have a lot of links to various places and things and ways to learn. You should also look at "The Hard Way," "[Learn] X the Hard Way." This is a style of teaching, and many different languages have "[Learn] X the Hard Way" books. It's really fun. There's a lot of - if you want to learn, of course there's Code Academy. They teach JavaScript, which is not a bad language to learn. You taught yourself JavaScript, Steve. Was it hard?

Steve: Yup, yup. No, it was, well...

Leo: It's very C-like; isn't it?

Steve: Yeah, I've been programming forever, but, yeah.

Leo: Yeah, but it's like C. Well, you know what the rule about programming is, you've got to do it every day. At least an hour or two every day. Because then, if you don't, you have to relearn the language. Randal Schwartz told me this. He's a Perl guru. Randal is such a guru...

Steve: That's especially true for Perl.

Leo: Yeah. Because you forget. You don't do it for four days...

Steve: Oh, lord.

Leo: You've got to get the book out.

Steve: Yeah. I've got the whole - GRC's news server uses a Perl frontend for adding a whole bunch of features. I mean, that I wrote.

Leo: You wrote it?

Steve: I wrote a Perl wrapper around the news server.

Leo: Oh, respect.

Steve: And it does all kinds of extra stuff. But, boy, I have to go look at my source, and I go, you know, I sort of like relearn Perl from looking at what I wrote before and go, oh, yeah, that's the way I do that. What I would say, answering the question of how you learn a language, is start with a book and read it until you start getting antsy. At some point you just kind of like, you start feeling like, come on, I want to get going, I want to - and then solve a problem.

Leo: Yeah.

Steve: That's the key, is solve a problem. Think of something that you want to do in that language, and put yourself about that task. Because it is by solving a specific problem that you will then realize, oh, I'm not quite as ready as I thought. And so basically it slaps you down a little bit. It'll put your antsiness back in its place. And then you'll go about finding the answers that you need for how to solve the problem. But that's - I think that's the key. You just can't, like, do nothing because it's just, at some point...

Leo: No, you have to write something.

Steve: Yeah, exactly, you've got to create something.

Leo: That's actually, for me, that's the biggest challenge is, oh, well, what problem should I solve? So it's nice to have a problem to solve. Somewhere, and I'll find it, there is a document I found once, a guy who said, "I learn a lot of languages as a professional programmer. I have 10 things, if I want to learn a language, I have to solve these 10 problems. And by the time I've solved all 10 in that language, that dialect, I'm fluent. I wish I could find it. I thought I had it in my bookmarks here, but...

Steve: I've also seen that. I'm kind of like, I know...

Leo: It was like on a news - it was a news server somewhere I saw years ago, and I copied it. I have a PDF of it. The other thing I'd recommend, I really recommend, and we've mentioned this before, is HTDP.org, which is designed to teach people to think about programming in a kind of a - more than just kind of get out there and write code way, kind of - this is from MIT Press. It's free and it's online, HTDP.org. And this actually uses a Lisp dialect called Scheme that's free and easily available. It's a good teaching language.

Steve: Nice.

Leo: But it's fun for me. I'm not a pro; I'm too old to be a professional programmer.

I wish I had been in my youth because I love it. But it's kind of like doing crossword puzzles.

Steve: What was the kid's programming language that Alan Kay did? There was a...

Leo: Well, he did Logo, Turtle Graphics. But I think you're thinking of Smalltalk. And then Scratch.

Steve: No, Scratch, Scratch is what I was thinking of, yes. Logo...

Leo: Yeah. Scratch is a Smalltalk, yeah.

Steve: Logo, then Smalltalk, then Scratch was a Smalltalk variant. And that's another...

Leo: Scratch is still there. A great place to go is Scratch.MIT.edu. And Scratch is very much like that - we were talking about that Android app inventor from MIT. It's the same exact idea where you click blocks together to make it do things. Yeah, this is Alan Kay's, still alive, this thing. In fact, Scratch, I think the one laptop per child is - much of the UI is written in Scratch.

Steve: Nice.

Leo: Smalltalk is a great language to learn, by the way.

Steve: Yup, yup.

Leo: What a good language. And that's test-driven programming, really some really good disciplines built into Smalltalk. And it's the original object-oriented language. It's, see, it's fun. You can just go on and on.

Steve: Yeah, there's been so much, there's so much depth and history here.

Leo: I love it.

Steve: And I've often thought I would do someday what you are doing, which is just sort of decide I'm going to learn another language and pick it up.

Leo: Really fun. Really, really fun. And I wanted to start, I wanted to kind of do foundational work. And so for me, going back to Common Lisp and starting there...

Steve: I think that's very neat.

Leo: ...is like tearing everything down and starting over.

Steve: Yup.

Leo: It's actually quite simple.

Steve: And look how much fun you're having. I mean, that's the whole point. Have fun.

Leo: So much fun. Steve, we are, speaking of fun, we are done. But always fun to do this show, learn so much from it. And I hope you all enjoy it as much as I do. We do Security Now! every Tuesday, right after MacBreak Weekly, about 1:30 Pacific, 4:30 Eastern time, 2030 UTC at live.twit.tv. Please tune in. Love it when you're in the chatroom. It really adds to the show for me. And then of course you can participate in other ways. Steve is on the Twitter, @SGgrc. You can send him questions there, or comments, or suggestions. He reads those. If you have questions for the show itself, you can go to GRC.com/feedback and fill out the form there. That's the best way. Steve also has lots of other stuff at GRC, including SpinRite, the world's best hard drive maintenance utility, a must-have.

Steve: It works.

Leo: If you've got a hard drive, you need SpinRite. It works. You might also be interested in all the other freebies he's got there. See, you only, really, you only pay for one thing. Everything else is free at GRC.com, including 16Kb audio versions and the transcriptions of this show. We offer full fidelity audio, soon to be stereo, soon to be joint stereo versions of this show.

Steve: Ah.

Leo: I don't know why, but - oh, I do know, actually I do know why, and I can't say.

Steve: Okay.

Leo: But a partner wants them in stereo, let's put it that way. So we thought, well, we'll just go stereo. So Steve will be slightly left, I'll be slightly right. Or something.

Steve: Interesting. Interesting.

Leo: Not so much that if you were listening in one ear you wouldn't understand it.

It's just slight stereo. A little fullness to it. We also have video, if you want to watch. It's a fascinating thing to watch. You could see Steve's blinking lights from his PDP-8. Those are all at TWiT.tv/sn, or look for Security Now! on all your favorite podcatchers and the TWiT apps and all that stuff. Steve, always a pleasure.

Steve: Well, and next week the - I have read Matt Blaze's testimony that he'll be giving to Congress tomorrow. And this is the whole issue of government's backdoor or front door or golden key or whatever. And there are some - there's some subtlety to it, but some really good points that he makes. And so it's my intention to share his testimony with our audience for next week's podcast. I think everyone will find it really interesting. And so we'll do that, and then we'll spend the rest of the podcast talking about it.

Leo: Oh, I can't wait.

Steve: Yeah. Because, I mean, this is the big, this is really the big question. Is our legal system going to force people, force encryption technologies to have some way for law enforcement to decrypt? The government and law enforcement desperately want it. And everybody who understands why it's a bad idea, even those who aren't concerned about privacy, but actually understand why it's a bad idea, Matt understands it. And he raises some points that I had never considered. And so I think it's going to be a fascinating podcast.

Leo: Oh, I can't wait. Next week.

Steve: Yup.

Leo: See you later, Steve.

Steve: Thanks, my friend.