SECURITY NOW!

Transcript of Episode #506

Law Enforcement Backdoors

Description: Leo and I catch up with the past week's most interesting security events and cover some miscellaneous tidbits. We then examine the carefully written testimony of two leading computer scientists who argue against the feasibility of incorporating encryption backdoors into commercial mobile and other device technologies.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-506.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-506-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson's here. We've got lots of security news. And then Steve's going to comment on testimony in Congress about backdoors for law enforcement. We know it's a bad idea. Did Congress get the message? Stay tuned. Security Now! is next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 506, recorded Tuesday, May 5th, 2015: Law Enforcement Backdoors.

It's time for Security Now!, the show wherein which we cover your security and your privacy, and we get tortured grammar and all of that. Here he is, Security Now!'s professor at large, Mr. Steven "Tiberius" Gibson.

Steve Gibson: Yo.

Leo: Oh, that's the wrong button.

Steve: Great to be with you again, as always.

Leo: Thank you, Steve. Good to see you.

Steve: Yeah. Well, and I did want to mention, I tweeted out to the people who follow me and got a lot of positive response. But there are certainly those who listen to this podcast who don't. And so I wanted to give everyone a pointer to last Sunday's TWiT, which I thought was exceptional. You just had - it was a magical combination of personalities that worked well together, knew each other, and also brought a lot of information. I mean, Jason is, like, really wired into what's going on in Silicon Valley.

Leo: Yeah. You know, Jason hadn't been on for three years. We had a little falling out. And I decided, oh, life's too short, and I invited him. And it turned out the best possible week ever to bring back Jason Calacanis because he has Tesla No. 1. He knows Elon Musk. So he had a lot of insight on the Powerwall. It was just - I agree.

Steve: And the Hyperloop. He's, like, he's involved in...

Leo: Yeah, he's building the Hyperloop.

Steve: ...the two-mile test track. He won't say where. But, yeah, I mean, he was just - it was a great episode. So...

Leo: Thank you. Thank you.

Steve: ...I wanted to, for those listeners of this podcast who don't normally listen to TWiT, the last Sunday's TWiT, I think it was 508, was just - it was spectacular. And I noticed, after the formal podcast ended, it kept going on. And that was really good, too. And I think your producer mentioned, like, capturing that also. So is that around somewhere?

Leo: Yeah. Jason - it should be in the recording. Is it not?

Steve: I don't know. I didn't...

Leo: Oh, you watched live.

Steve: I didn't watch the recording because I watched live.

Leo: Yeah. So, and this, I don't know, in the early days of TWiT we did this all the time. I would record a lot more than we would use, and then we'd have some bits in at the end. After the theme song and the credits rolled, we'd kind of "Smokey and the Bandit" style come back and have a little, you know, stuff, not bloopers, necessarily. But in this case Jason Howell, our producer, suggested, and I think he was right - because we had a very kind of philosophical conversation. We had closed the show, of course, eulogizing a dear friend of Jason's, as it turned out...

Steve: Dave Goldberg.

Leo: Who was CEO of Survey Monkey and died, we've learned now, died in just a freak accident. He was...

Steve: He said like some sort of a concussion during exercise, or health-related.

Leo: He was on a treadmill.

Steve: Ah.

Leo: He was on a treadmill. And in fact the Washington Post used this as an occasion to do a long article on the dangers of treadmill because those are very powerful motors in there.

Steve: Yeah.

Leo: And people don't take it too seriously. And when treadmills started coming into the home, the accident rate during exercise went through the roof. So I think there was no one in the exercise area, so nobody knew exactly what happened. But he'd apparently fallen off the treadmill, injured his head, and bled out before anybody discovered him. So just a freak accident, a tragedy. Two young kids, I think 13 and 11.

Steve: And he was 47.

Leo: He was just 47. And that's one of the reasons I was so glad to have Calacanis there because Jason was his poker buddy, knew him well. And so he was really well equipped to talk about his - he called him Goldie - Goldie's contributions, what a great guy he was, and what a tragedy it was that this father and successful businessman - his wife, Sheryl Sandberg, runs Facebook.

Steve: Right.

Leo: Just a loss to the community. I did not know him. But I felt as if I did after Jason spoke about him. So that was also very timely. I was really glad to...

Steve: Well, and as you were saying, the dialogue after the formal podcast ended was just - it continued to be just worth, I mean, just good broadcasting.

Leo: Well, we were talking about, you know, life and death and why the good die young and, yeah, it was great. So we kept it. I think it's - I'm pretty sure it's at the end of the recorded episode as kind of outtakes, yeah.

Steve: Okay, good, good. So we don't have a huge news week. The main topic today is what I predicted or promised last week, which was the issue of law enforcement backdoors. This congressional testimony did occur the following day, that is, last Wednesday, following last week's podcast. And so I want to share the written testimony of Matt Blaze, who is a well respected doctor of computer science. He's the guy that found the problems with the Clipper Chip when the government originally tried to do this.

His oral testimony is not the same as the written. And I think the written is - because you can sit down and take the time to put it together carefully, and maybe you're in a little bit of less hurry. Whenever they're delivering testimony, it's like they leave out the punctuation. They just sort of rush through it, you know, during the actual committee meeting. But I want to share it because it brings up some really good points.

And then I found someone else who we've talked about before, who has come into the podcast, and that's Jonathan Mayer at Stanford, who's a computer scientist and lawyer. And he takes a very different heavy computer science approach, which is looking at Google and Android as an example, what are the actual practical problems with doing this? And he demonstrates that you can't, that it's just impossible. So I think that's going to be really interesting. So I want to share that, and then we'll discuss it. And of course at the top of the show we've got some interesting news to catch up on. So another great podcast.

Leo: As they say, busy busy busy.

Steve: So, okay. Our first story, news of the week, is - I don't know why it's called the "Pixie Dust" attack. But it's a new, well, it's a revelation about how to attack the WPS protocol on a number of WiFi routers. And we've covered the problems with WPS in years past. In fact, I'm not going to go back into excruciating detail of the protocol, only because we did that on Episode 337 of this podcast, Security Now!, which was January 25th of 2012, so a little more than three years ago. The title of that podcast was "WPS: A Troubled Protocol." And there we broke down in detail how it works and why it just was never really very solid. You may remember the name "Reaver." I remember when I saw that again, I thought, oh, yeah, I remember Reaver. Reaver was one of the tools that surfaced back then for cracking WPS. Some changes were made. Basically it turned out not to be nearly as strong as it was believed.

WPS essentially uses an eight-digit PIN, which is normally printed on the label, along with the serial number and the MAC address for the router. Right next to that it'll say "WPS PIN." And the idea is that, for example, if you have a WPS-aware client, like Windows, you can pair your Windows system to the WiFi router just by typing in this eight-digit PIN from the label of the router into a dialogue on the client, and you're sort of magically done. You don't have to worry about what the router's password has been configured to. So it makes that easier. And there's also a button you can also press. And it's actually one or the other. So some routers just have a button where it's like, okay, within a short window you can do pairing. Otherwise, type in this eight-digit PIN.

Well, eight digits seems like a lot to brute force because we know just eight digits of decimal is, what is that? I didn't think of it ahead. A billion? A million? Anyway, it's a lot. And it turns out, though, that there was a weakness in the protocol where, if you divided it up into two four-digit pieces, you could, like, do half of the protocol with only four digits, which dramatically reduced the total number you had to try because essentially you could do four digits at a time. You could do just half. And in fact the eighth digit is actually a checksum digit, so it doesn't really even count for the second four. It's only really three. So there's only a thousand of those, and 10,000 of the first.

So some adjustments were made to try to fix it. And, you know, it's sort of - it's limped along. Some routers have it. Higher end routers have removed it because it just sort of has made people feel uncomfortable. Well, so the news of this week, three years after all of this, is that a security researcher, Dominique Bongard, discovered, he looked closely at some actual implementations of the detailed protocol on some popular routers. And what he discovered was that it was much worse, of course, than we thought.

Any of these sorts of protocols requires - and how many times have you heard me say this - a good source of random numbers. Cryptography, almost without fail, I'm not sure I can think of an area where there isn't at some point in a protocol the need for entropy because, even though we're protecting the protocol with crypto stuff, normally it's a secret that we're protecting. And it's not the same secret. Typically protocols grab a random number. And then, for example, we were talking about it last week, where if you wanted to communicate privately you'd pick a random number, use that as your key for bulk encryption. Then you'd use public key cryptography to encrypt the random number.

Well, obviously, if that random number wasn't random, you have absolutely no protection. None. So despite all the other mechanisms in place, if you don't actually have really unpredictable random numbers, you have no protection. And it turns out that, as a consequence of - in some cases it's got to be a bug, or just them not caring, apparently the implementers not caring very much. At least several routers that have been looked at are beyond bad.

And the examples of them being bad are what hooked me on this because, for example, Ralink access points end up using two nonces. Remember that "nonce" is the cryptographic term for a one-term random value. A number once, technically, is what nonce stands for. But the idea is that it can't be known, can't be predictable. It's got to be - oftentimes doesn't actually have to be random, but there has to be no way for an attacker to have an idea of what it could be, however that ends up being arranged. And sometimes just having a good source of true entropy is the way to do that. But in the protocol there are two 128-bit nonces called E-S1 and E-S2.

And again, back in that podcast 337, I go over this in detail. So anyone who wants more can get it there. They must be unpredictable for this handshake to work. There is also a publicly chosen and, like, visible random number which the two endpoints share. And that's called the "enrollee nonce." Okay. So this Ralink access point uses for E-S1 and E-S2, which have to be unpredictable, uses for both of them the value zero. Which is, you know, meets no criteria. It turns out that these nonces are mixed in with some other known values and hashed in order to produce intermediate results. And so no Ralink access point is secure if it has WPS enabled.

So now what we have is a known offline attack which can listen to any negotiation with WPS - even an attacker can initiate one - generate some protocol, and then just cut through it like butter, immediately determine what the access point's PIN is, and then associate with it, and you're on the network. So Ralink router owners, you absolutely disable, turn off WPS if you haven't already. And that was our recommendation three years ago because it just wasn't safe.

Okay, but there's two more examples that are better - well, it can't be worse than zero - but still demonstrate a failure to appreciate the cryptographic importance of entropy. And unfortunately, this one is Broadcom routers with eCos devices. Those two secrets, the E-S1 and E-S2 secrets, they're generated immediately after the enrollee nonce. And the function that is used is a simple, linear, congruential, pseudorandom number generator with no external entropy. So we've talked about that before. A linear congruential pseudorandom number generator basically takes a value and multiplies it by - it takes the current value, multiplies it by some unknown constant, adds another constant, and that's the next value. So what it is, is entirely deterministic. If you know one value, and you know those constants, you know the next value and the next value and the next value and so forth.

So here, in the Broadcom router, remember I said that the supposedly random nonces E-S1 and E-S2, are generated immediately after the enrollee nonce. Well, the enrollee nonce is public. It's published. It's in the air. So once again, this can be cut through instantly. Someone initiates a WPS handshake, gets the enrollee nonce, can immediately, from knowing what the PRNG in the Broadcom router is, determine what E-S1 and E-S2 are, and crack the PIN and acquire access to your network. So again, disable WPS.

And the last example is Realtek routers. And we don't have model numbers, so I don't want to over panic people who, like, have a Realtek access point. The chipsets may vary. The firmware versions may vary. But these three, instances of these three routers were used in the security presentation. The Realtek access point uses the time in seconds from January 30th, 1970. That is, so however, whatever duration it's been from January 30th, 1970, in seconds, until this protocol is exercised, it uses those to generate these three values. It uses the same generator to make the enrollee nonce, which thereby publishes the value of time that it thinks it now is, and uses this same time value, or the value from current time, for E-S1 and E-S2.

So if the entire exchange occurs within the same second, then E-S1 and E-S2, which are the - well, I'm sorry. E-S1 and E-S2 will be the same as the enrollee nonce, which is published in the air at the initial start of this negotiation. So again, no mystery there. And if it doesn't, if the second counter rolls over, then they are literally one greater in value than the enrollee nonce. So, again, no protection. The bad news is it's not like all routers have had this checked. These three routers were found to be ridiculously poor in their actual implementation of a protocol that was already very troublesome. So the lesson here is this is not something - WPS is really not something you want to leave enabled all the time. Yes, it's convenient. But we know what a problem convenience so often is. And this is another example of that. So classic security implementation problems in low-end consumer routers.

I did want to update everyone on some tweets that I received after last week or the week before, noting that RC4, this technically still secure, but people just aren't comfortable with it protocol, was disabled in Firefox v36. Apparently there's a whitelist in Firefox where Mozilla found some sites where they didn't want to cripple those sites because, for whatever reason, they only support RC4. So Firefox has a whitelist that still lets RC4 happen, if that's where you're going. But otherwise, by default, it is disabled. So it's no longer offered in the initial handshake of TLS, offering it to the server as an opportunity.

And I also wanted to mention that a number of people have tweeted, while we've been talking about this, that they've managed to turn it off in all their browsers. And there's ways to do it in Chrome, and ways to do it in IE. It's like, okay, good. It's not something I'm worried about. It's in the process of being phased out by the industry. There aren't actually any known vulnerabilities with it; whereas, for example, there are with the cipher block chaining ciphers, although there are clients, the browsers have mitigations in place now to solve those problems because those were sort of more theoretical, too. But we know that theoretical vulnerabilities become real very quickly.

And Mozilla has stepped into the game with Google of deciding they're going to put pressure for deprecating nonsecure HTTP connections. I guess really Google has been more in the we're going to make the HTTPS connections more secure by putting pressure on the security certificate side. But last Thursday, in the Mozilla security blog, the Firefox security lead Richard Barnes did a blog posting titled "Deprecating Non-Secure HTTP." And he said, just the beginning of his posting, he said: "Today we are announcing our intent to phase out nonsecure HTTP." Okay, now, understand what that means. He says we're phasing out the use of non-HTTPS, meaning we're, like, formally saying we're trying, we're going to do what we can to encourage people to no longer use non-secure web connections.

So he said: "After a robust discussion on our community mailing list, Mozilla is committing to focus new development efforts on the secure web" - their phrase - "and start removing capabilities from the non-secure web. There are two broad elements of this plan: setting a date after which all new features will be available only to secure websites, and gradually phasing out access to browser features for nonsecure websites, especially features that pose risks to users' security and privacy."

Now, what I thought was really interesting about this is they're putting pressure on a different aspect of the community. They're essentially putting pressure on the site developers, that is, they're saying - and his blog goes into more detail, and there is more detail in the discussion that he talks about being robust. I mean, it was a food fight because it's considered controversial. But what they're saying is they're going to, like in the future, only just by decision, for no security reason except they want to create a way of making it difficult not to use HTTPS, Firefox's forthcoming features simply won't be present. They won't work under HTTP. So a site and webmaster or site builder, the people who are responsible for content, if they insist on continuing to not offer HTTPS, that is, if they're serving content unsecured, Firefox will withhold features, like useful features...

Leo: Like what, though? Have they said?

Steve: Well, like HTML5 new features, like CSS new features. Actual, like, standards features that are continually being developed and being added to the browser, they're just going to say, no. If you're not secure, we're not going to let you use those. And so the notion is that content providers would be forced to work around that lack of features at some discomfiture such that is would be easier just to give up and switch to HTTPS. So...

Leo: Jesus.

Steve: I know. I know. It's like, wow. Okay.

Leo: We could be more bully-ish than Google if we try.

Steve: Yeah, and I have to say, anyone who's interested, it's blog.mozilla.org. The link is "Deprecating Non-Secure HTTP." Last time I looked, there were, like, 208 comments on the blog. And, oh, boy, I mean, it's crazy. I mean, because this is just regarded, you know, I mean, everybody's got their conspiracy theory. They're saying, oh, Firefox has thrown in with the certificate authorities, or they're in league with the HTTPS people or the LetsEncrypt.org. And it's like, what do you mean? That's a free certificate. I mean, it makes sense to be using it. And but I think what this speaks to is what we really see, which is look at IPv4 and IPv6. Rather than switching to 6, people are spending tens of millions of dollars on the vanishing number of IPv4 addresses. I mean, it really does take people, I mean, it'll take a lot of pressure to get people to move away from something that is working, if they really don't have to change.

Leo: Well, they're being economically rational. And I just feel like - I don't know.

Steve: I know. I'm with you. It's like, wow.

Leo: It seems a little totalitarian to say, well, no, you can't be economically rational. You have to do it this way. Fortunately, there's competition amongst browsers, and you have a choice.

Steve: Yeah. And that's what I found myself thinking was, wow, you know, I hope - I like Firefox. And but there's Chrome, and so far Google's not doing that. And so if that becomes a problem, I mean, although...

Leo: Well, Google's going to do it, don't you think? I mean, they're moving in that direction, too.

Steve: Yeah. And actually, this doesn't put pressure on the users, though. This really puts pressure...

Leo: Just websites.

Steve: ...on the websites. So users wouldn't see any difference unless the website started saying, please switch to Chrome.

Leo: Well, guess what.

Steve: Where we'll offer all, yeah, we'll offer more features.

Leo: Yeah, you want a good experience, use Chrome. That's not - people do that anyway.

Steve: Yeah, yeah.

Leo: I mean, I guess if the only possible reason that a site is not secure is out of stupidity or laziness, I guess you could make that argument. But I also feel like there might be legitimate reasons why a site is not using SSL. I don't know what they would be. I mean, we've had - it's going to cost me money.

Steve: Yes, yes.

Leo: I mean, I have to pay thousands of dollars to make sure that this works, and it's going to make it a less robust site.

Steve: I think there is a - say that you have a site that just wants to provide information.

Leo: Yeah.

Steve: It's just there to have people come and read. And, like, nothing else. I mean, and there's certainly a vast, a huge number of sites that fall into that category. It's like, okay. This just, I mean, there's no rationale, no - you can't see a reason for forcing it. Yeah.

Leo: Oh, yeah. I mean, I don't know enough about it, I guess. I just know that we are - we feel compelled to do it. And really you don't get the content from us. All you get is web pages from us. Our CDN, Cachefly, which does offer HTTPS, does it. But then I would have a mixed-mode message, though. You're going to get a warning that part of this page is encrypted, part of it's not encrypted. And you're going to get a warning. And that's not a good experience for users. So I've got to bite the bullet and just do it.

Steve: Yeah. And I really do think that, downstream, that's the future. At some point it'll be like, wait a minute, why isn't this site secure? And actually, two more stories, I've got a piece that's a little bit interesting because CloudFlare did a blog posting about sort of this new era of DDoS attacks. And it is the case that man-in-the-middle attacks are trivial to implement on nonsecure HTTP connections. And I don't know if that's justification. But it is the case that it really lowers the bar for attacks not to be served over secure connections.

Leo: And that would be a good argument for it. I mean, somebody'd come to TWiT, there'd be a man-in-the-middle, and then they could put malware on their computer, right, because they would think that it is a trusted connection to TWiT, when it's not.

Steve: Correct. And actually we can talk about it right now. I mean, the man-in-the-middle attack is almost impossible with HTTPS because you have to - the man in the middle...

Leo: Unless you're Superfish.

Steve: Well, yeah, has to present a certificate that the browser trusts. And now, so a state-sponsored man in the middle can be done because I don't think anybody believes now that a nation-state can't synthesize any certificates they want to on the fly. Just, you know, the NSA has to be able to make a certificate, if they want to. And we know that China can because they own CNNIC and can tell them exactly what they want. So, but still, that raises the bar from hackers to somebody who has the ability to generate certificates on the fly. And that's a heavy burden because the second you generate, as we've seen, a fake Google cert, and a Chrome browser touches it, all hell breaks loose, and the jig is up. So we have enough canaries around now that it's difficult to get away with that.

An unsecured page, though, is just - it's literally just plaintext moving by. And on the way to the browser, that is, as the page is being returned to the browser, simply inserting a JavaScript tag with a URL of some foreign server, when that gets to the browser, in parsing the page the browser will fetch and execute that JavaScript. It is that simple. And that, I mean, this is a variation on the Great Cannon that we talked about a couple episodes ago. That's what essentially China's Great Cannon was doing to DDoS GitHub. In that case it was substituting some JavaScript that was already being requested. But nothing prevents anyone from being anywhere in a connection and just adding a JavaScript tag. Browsers run JavaScript. They'll fetch the script, and they'll execute it and can get themselves in trouble. So I can see the logic, if we take that sort of, wow, you know, all is lost, the web is really under attack, then going to secure connections everywhere really does raise the bar.

And speaking of raising the bar, there is a crazy new anti-analysis malware which has been found. Craig Williams of Cisco, who's one of their security guys, has been looking at a new strain of spyware that they detected earlier this year, just a few instances. So it's believed to be only in use in targeted attacks. It logs keystrokes and steals data, but also has a destructive side which unleashes wiper capabilities if it detects it's being analyzed and audited. So it's not worried about end users because end users just run it. It's now gone to tremendous lengths to protect itself from being reverse engineered.

Quoting Craig, he said: "It sounds clich, but this really is a digital arms race, and we're seeing the next evolution of it here. Malware authors are no longer content with detect and shut down. Now, if malware realizes it's being audited, the binary will destroy the system."

Leo: Wow. That's the nuclear option.

Steve: Yes. "It's a simple case of attackers trying to dissuade researchers from going after a sample." And so just to give you a taste of this crazy thing, this thing contains 8,000 executable functions which are never used. It just piles them in there to water down the actual content hiding among 8,000 other functions that never get used. At one point, it writes a byte of data into memory 960 million times in order to break sandbox logging because one of the techniques that's used by sandboxes that are used to watch these things run is they log the activity of the malware. So this thing thumbs its nose at that and writes a byte 960 times. And it turns out that would produce a hundred-gig log file and take half an hour to write to the hard drive. So again, it's like, okay, this thing is really flailing around, refusing to be analyzed.

And the unpacking code that unpacks this into the system Cisco describes as "monstrous and has many times the complexity of the anti-analysis code." It contains dozens of functions overlapping each other and embedded with unnecessary jumps added only to increase the complexity of what's called the "call graph." He said: "The result is a nightmare of control flow graph with hundreds of nodes." Oh, and it continually hashes itself in order to detect if any change has been made. And if it finds any, it first tries to overwrite the user's master boot record. And if it can't verify that it did that successfully, it then randomly chooses RC4 keys and overwrites every file of the home folder with a different random key which it doesn't bother keeping track of.

Leo: Why should it, no.

Steve: So it's just...

Leo: Wow.

Steve: It's like, I can't think of the analogy of, like, some sci-fi where something, some alien has been trapped in a box, and it's just flailing itself around, like, wildly, and finally ends up destroying maybe its host or its container, at least.

Leo: Do you think this is written by bad guys? It sounds pretty sophisticated.

Steve: Wow. Well, that's a good question. I mean, unfortunately, we have to ask that question. And there was a lot of interesting, in this congressional testimony that occurred last Wednesday, I have to say I was more impressed than I expected to be with the government's, with Congress's position. There are, I think I read elsewhere, four people in Congress who have computer science degrees, which was four more than I expected.

Leo: Yes, really.

Steve: So, yeah. There seemed to be some hope there.

Leo: They know COBOL, boy. They know their COBOL.

Steve: So CloudFlare, they did a blog - basically they didn't name China. They didn't say "Great Cannon." But this was pointed straight at them because this was - CloudFlare's blog was titled, "An Introduction to JavaScript-based DDoS." And of course everyone who's been following this podcast knows that the way the Great Cannon works is it was intercepting unsecured connections to a Chinese search engine and replacing a bit of JavaScript which users' browsers all over the world retrieved and then executed. And so that is a JavaScript-based distributed denial of service.

And in the blog posting they sort of harkened back to the nice days, where it was just an NTP or a DNS amplification or reflection, you know, those simple days where you would be bouncing stuff off of some server, spoofing your source IP, and so it would reflect onto your target. Now, unfortunately, the DDoS guys have a new trick, which is JavaScript.

Now, the question is, how do you get JavaScript into browsers, which is what we were just talking about a minute ago. Now, of course, one way to do that is you infect a website so that the website itself is actually delivering malicious JavaScript. And unfortunately we have seen a variation of that where ad servers get malicious ads until they're discovered and taken down. And so a benign website has links to an ad service that's servicing ads, and that can also contain Flash, for example, and JavaScript and so forth. So, and obviously the ad suppliers are doing everything they can to prevent that from happening. But sometimes something squeaks through that's unseen before, or extra clever.

So one possibility is that a malicious page hosts JavaScript. The problem is that a malicious page will get known pretty quickly. And malicious pages, I mean, where the actual site has a page that is deliberately malicious, they don't have many visitors. So the strength of your denial of service attack is purely a function of how many people's browsers are running the malicious script you're offering. So you want to infect very popular, high-traffic sites, one way or the other. Get an ad onto them. Maybe do some SQL injection. If you can modify a page, somehow get a page to produce the script that you're trying to get the browsers to run.

Another possibility of attack is the JavaScript libraries themselves because what many web pages do, rather than hosting their own core library, like jQuery is a perfect example, 30 percent of websites in 2014 had pages that referred to jQuery. So they didn't grab a copy of it and then host it themselves. They just pointed to the jQuery library. And they like that because, as the jQuery library gets updated, their pages get the benefits of any bugs that are found and so forth. The flipside of that is, I mean, that represents a huge target for malicious exploitation, if somebody can get into one of these sites. And, for example, last year jQuery.com's website was compromised. So if they could get, like, turn the jQuery library malicious, that's a win. And Leo, didn't you have a problem a couple years ago with one of the libraries you were using?

Leo: Yeah. It wasn't jQuery. We use Drupal.

Steve: Yeah. So again, I mean, another...

Leo: And it was a Drupal library. It wasn't even a - so Drupal, and this is true of WordPress and a lot of content management systems, has publicly contributed libraries, or little code widgets. And so our developers had used one. And a vulnerability was found in it. Either we didn't patch it, or it hadn't been patched. Bad guy, you know...

Steve: Found it.

Leo: All you do is you google.

Steve: Yeah. Yup, yup.

Leo: And they're usually easy to find. And then you can use it. He used it to, I think it in effect gave him an open FTP folder that he could put stuff into. So he put some malicious script in there.

Steve: Like storing stuff on your server.

Leo: Yeah, and it was a malicious script that would get activated. But it took advantage of a flaw that had been patched in most operating systems. So to my knowledge nobody used - I don't know if our users got bit. And of course very quickly Google puts up that warning, and then everybody follows suit.

Steve: Right. Oh, in fact...

Steve: That's what happened.

Steve: Right, right, right.

Leo: Which we welcome, because that means people aren't going to go into our site and get infected. I don't want anybody to get infected.

Steve: Right, right.

Leo: We will be using Node.js in our site in future. So this jQuery thing is of interest because that uses - yeah.

Steve: Yeah, exactly. Same sort of solution. Now, there's an interesting proposal we've never talked about coming from the W3C, the World Wide Web Consortium. And this is a new option for - and it's called "Sub Resource Integrity." And this was mentioned in the CloudFlare blog. The idea is that normally your tag would be - it would say "script src=" and then in quotes would be a URL. And the browser would just go and fetch that. And, for example, it might be, and I have in the show notes an example, "jquery-1.10.2.min.js," "min" meaning it's been minified, JavaScript. And so your browser is saying this is the version of jQuery I want to pull.

Well, with the addition of Sub Resource Integrity, the script source tag, that tag can also have the optional word "integrity=" and then a hash specifier, like SHA-256 hyphen, and then the hash, the SHA-256 hash, the proper hash of that version of the library. So if a browser is aware of this, and the site has protected itself/its users by providing this, then the browser will see the script tag with this extra integrity value, fetch the resource, do an SHA-256 before executing it, verify the hash matches, and then, and only then, implement, you know, drop that script into the page at that point. So this is very cool. This does not protect us from injection, interception injection. But it would protect from anyone maliciously modifying the library or the resource that is being fetched.

So, again, it's not perfect, but it forecloses one of, well, actually, one of the more worrisome avenues would be that the original jQuery library would be infected, the version number not changed, the browser fetches it and runs it, and then every browser on the planet which is at that time fetching pages that are invoking jQuery, would all get this script. So that's breathtakingly awful. So it's neat that we'll have the ability - oh, and so what happened is, or the way it would work is that the webmaster would pull the script, generate the SHA-256 hash, and then put that in their page. The reason this is useful is that this still protects us after we have HTTPS.

Remember that the interception injection sort of relies on the man in the middle being able to inject a script tag into the page as it's going back to the browser. The bar for that is very low. It's trivial to do. But once we have SSL/TLS, HTTPS, then that becomes much more difficult to do. So random malicious weenies aren't going to be able to do that. But even with HTTPS, we could still be victim to the source of the library being compromised. So what I like about this Sub Resource Integrity is it blocks that one. If you get, as far as you know, a known good copy, make the hash, put the hash in your page, and then browsers that are aware of this won't act on it. And right now this is not well supported. But Chrome and Firefox both have it in the works. So there will be support for that coming soon.

And lastly, the last thing I had in my notes we sort of already covered, which is that I wanted to make sure people understood how trivial man-in-the-middle attacks to inject JavaScript is if we don't have secure pages because it's just plaintext going by. And you drop that little <script src= tag onto the page, and the browser, when it receives it, it'll just suck that right in from wherever. It doesn't have any idea, like what server is which. It'll just grab it. So, and especially if the page is not secure. There's going to be no mixed-content warnings or any other sort of problem. And content is being pulled from foreign locations all the time. That's the way the JavaScript ecosystem works.

So anyway, this is the evolution of distributed denial of service attacks, first really seen clearly by this Great Cannon, as it's been called, that China aimed at GitHub, which involved innocent users all over the world having JavaScript in their browsers attacking some site. And this is probably the next area where things are going to go.

Last Wednesday, as I said, there was a really interesting congressional hearing on - and I've got the whole title of it. What is it? I have it here in front of me: "U.S. House of Representatives Committee on Government Oversight and Reform, Information Technology Subcommittee, Encryption Technology and Possible U.S. Policy Responses."

Leo: Well, I hope those three computer science degree guys are on that committee.

Steve: Well, in fact one of them was.

Leo: Good.

Steve: And he was really good.

Leo: Good.

Steve: Now, I watched - I caught the last half of it live, and then I was able to get it off of the 'Net. It was two hours and 15 minutes. But there was an annoying blob of time in the front, and an annoying blob of time at the end, and an intermission because they had to do some - there was some congressional voting that had to happen.

Leo: You just described government. There's an annoying blob in the front, an annoying blob at the end, and intermission in the middle.

Steve: Anyway, I edited it down to an hour and 15 minutes.

Leo: Oh, great.

Steve: So basically I took an hour out of it. And then I thought, where am I going to put this? And then I remembered Google telling me some time ago that it had reserved some YouTube tags for me.

Leo: Nice.

Steve: And I thought, it's time for Steve to activate his YouTube account. So I now have a YouTube channel.

Leo: Wow.

Steve: I know, we're getting there, kicking and screaming.

Leo: Will wonders never cease.

Steve: YouTube.com/SGgrc.

Leo: That's easy to remember, good.

Steve: Yes. And so what's what everyone would expect. SGgrc is the name of my YouTube channel.

Leo: I like the picture of you on there. You're very relaxed.

Steve: Yeah, I'm trying to think what that was.

Leo: It's your Google account.

Steve: Oh, you know, that was a picture from New Year's Eve because I was wearing my...

Leo: Oh, yeah, there's the bricks.

Steve: I was wearing my "Born to Code" shirt, and that was taken there in your studio. And I kind of went through, I thought, I need another photo because, you know, a little thumbnail to have because what I had there was awful.

Leo: Good.

Steve: So, yeah. So there's two videos there. There is the edited down congressional hearing that I really want to encourage people to watch. It's hopeful more than anything else. And you can see it starts here. Now, that normally went on for, like, 25 minutes, I think, and I cut it...

Leo: Thank you for chopping that off.

Steve: I cut it down to just long enough that you knew where you were. But I wanted to take a moment to mention a utility, well, not a utility, a little commercial tool that I've been impressed with for several years. It's called Video ReDo. And I know a lot about video editing because I know a lot about MPEG compression. It's just like something I focused on back when I learned years ago how to make DVDs. I was sort of curious about all that. And editing an MPEG compressed stream is tricky because everyone's familiar that video is a series of frames, a series of essentially still images that are played quickly. The way MPEG works, Motion PEG, essentially, is that you have...

Leo: You don't know what the PEG stands for, do you.

Steve: I forgot.

Leo: Photographic Experts Group.

Steve: Experts Group, that's right.

Leo: But Motion PEG is good.

Steve: Motion PEG.

Leo: Sounds like a pirate thing.

Steve: So the way it works is you have a so-called GOP, a Group Of Pictures, where the beginning is what's called an "I" frame, I for independent, meaning that it is a complete, sort of just a JPEG compression of a frame. Then you can have a mixture of what's called

"B" and "P" frames. P's are predictive because it contains changes. It only encodes the changes from the previous frame. And B frames, stands for "backwards" because it's able to look both upstream and downstream, and so it's able to incorporate the future. And so the idea is that you have this I frame, and then a group of, I think it's 14, P and B in various combinations. And so the idea is that you had, like, one index frame, one independent frame, and then you only encode the changes. And then you'll have another one because there's going to be some drift and some loss over time. So it's like, okay, time to, like, come back to normal.

And anybody who's, like, seen MPEG streams, like in the satellite TV days, they do, it's like, but you've seen, like, bizarre artifacts where something wrong will move. And it's because the motion is actually represented by vectors so that what's, I mean, it's an incredibly sophisticated encoding to represent the change between successive frames.

Okay. The point of this is that editing that is really tricky because what you want is frame-accurate editing. And the lazy way is to only allow people to make changes at the I frame, at the independent frame. But a lot of things can happen in between independent frames. So the best kind of compressed video editing is one which is able to allow you to make cuts anywhere you want with single-frame accuracy, and realize most of this didn't change, so let's not mess with it. But the boundaries of the cuts may need some fancy re-encoding in order to - because we're going to have a cut in the middle of one of these group of pictures and somehow need to represent that.

So Video ReDo does this, and it does it beautifully. It's got kind of a hokey-looking website. I mean, it doesn't look very impressive. But it is, of all the tools that I have found, this is what I've been using for years, and it has never let me down. Like sometimes I'll want to cut the commercials out of something that I've sucked out of my TiVo because I want to send it to Jenny to show her. And so it makes it simple to mark regions and then say, okay, re-render this. And it does it in seconds because it's smart enough to know that most of what it's doing it doesn't have to re-render, it just does a copy, and it only needs to re-render the edges of the cuts. And also it has a large palette of compressors so you can compress to MPEG-2, MPEG-4, AVC, it's really feature complete.

So I just wanted to give them a free shout-out because I used them in order to edit this congressional testimony down, and it just blew through it, where it could have taken, I mean, it was two hours and 15 minutes, and it did it in, like, 15 seconds, only because it is so smart about doing it. So for anyone who hasn't discovered it and who has a need to do mostly cuts of things, it's just - it's my favorite tool for that.

My quote of the week, Gene Hastings quoted something I just got a kick out of. He said: "You may think that IoT means Internet of Things. But really it means Internet of Targets." Which...

Leo: Oh, yeah.

Steve: ...I thought was really good.

Leo: It's very appropriate.

Steve: Because we're going to be in that era here before long. And I also got another tweet from someone named AskApache, who on the 30th tweeted: "Steve, SpinRite saved my drive. Even the BIOS wasn't recognizing it." And then he tweeted. In his tweet was a picture. And sure enough, you see the green R's for recovered right up at the front of the drive, where is so often the case. The system won't boot because the beginning of it has a problem because that's often where the heads end up spending a lot of time. And then there's one recovered sector out there in the middle somewhere. And then I actually had my own testimonial. From yesterday. Because SpinRite saved me many days of work.

Leo: What?

Steve: But also taught me an interesting lesson about end-user RAID protection. Sue's machine I last visited three years ago. I've mentioned her often. When anybody writes to sales@grc.com, you get Sue, who immediately handles any problems. People sometimes look at their credit card statement and say, what the hell is this? What is this Gibson Research Corporation? And so they call, and Sue says, "Uh, SpinRite?" And they go, oh, of course, fantastic, thank you so much, and sorry to bother you. So I need someone there just to answer those kinds of questions. I got a call from her yesterday morning that her monitor had died. And I said, okay, go get one, whatever you want, and plug it in. And so then I got a call an hour later. Now we're getting something, a Hal.dll not found. And I said, okay, I'll be right down.

So she was set up with a two-drive Promise RAID a couple years ago because I never wanted there to be a problem with her data being in danger. It's also backed up daily to the cloud. So, I mean, that is, all of the important files, the accounting files and her email folder and things that we really - we can't lose. Even the original FoxPro database from before we switched to web-based purchasing, so she can look up users who have SpinRite 1 and go, oh, yeah, here you are, we'll give you an upgrade. All of that is kept current.

I got down there. The system was in bad shape. And the first thing I see in turning it on is an error saying that the RAID is critical. It is in degraded shape. Press Escape to continue. Then she said, "Oh, yeah. Well, you know, it's been doing that for a while. And I just - I didn't want to bother you, and I figured I'd mention it next time we met."

Leo: It'll get better eventually.

Steve: Oh, oh, oh. So...

Leo: So she replaced one drive of a RAID array.

Steve: No, no, no. What happened was the first drive failed catastrophically some time ago. But because it let her press Escape to go on, she did.

Leo: Yup.

Steve: Until the second one died.

Leo: Yup. Was she in RAID 1?

Steve: She was in mirroring. I had her in full mirroring because I wanted full redundancy.

Leo: Sure. You had a backup.

Steve: Absolutely. What could go wrong? And it's like, "Oh, my god, Sue." And she says, "Well, yeah, but it let me go." I mean, you know, she pressed Escape, and it kept working. And so my takeaway, I'd never had this revelation before, was end-user RAID is different from server RAID. Because I've got all kinds of notifications, and I'll be alerted. And I actually, I set up email notification in case of a problem. But she's in Cox, and they don't - I was never - there's no, like, verify that the Promise's RAID management is able to send email. So I configured it so that it should work. I never got any email. And frankly, now I'm a little worried that maybe it had my email address from three years ago.

Leo: Uh-oh.

Steve: And not a static one. So that's a lesson to learn.

Leo: So you had a standalone RAID card. This was not BIOS software RAID. This is...

Steve: Correct. Correct.

Leo: You had a PROMISE card.

Steve: Yes, it's a Promise RAID card, a pair of drives. The first one died. It kept complaining every single day. But because it let her go on, she did. And my lesson from this is there ought to be a configuration option for, and we'll call it "end-user RAID," where when this thing is in trouble, it says no. It stops. Now, the point is, that's when you get the frantic call, yet you still have a fully functional system that you can recover from. But the end-user will not read the message. Doesn't matter. I mean, I told her, I remember so clearly saying, if there's any error messages from, like, the system, call me. I will be right there. No, no call.

Leo: Well, it wasn't really an error message. It was an advisory.

Steve: No, no. It was scary-looking. I saw it.

Leo: Really? Was it scary?

Steve: Critical RAID failure, system operating in degraded mode. But the point is, it gave her the option to bypass.

Leo: Play through. That's what everybody will do. I would have done it.

Steve: Yes. And it should, well, and, oh, look, it works. Well, I guess it's okay.

Leo: It works. Can't be too degraded.

Steve: And it's like, oh, Sue. So my lesson is, if there's - I'm going to look when I bring this back to her. Anyway, the second screenshot there in my show notes shows, "All data was completely recovered from this damaged sector. All data was completely recovered from this damaged sector," blah blah blah, on and on and on. And so SpinRite ran, and it brought that drive, the one that was barely hanging on...

Leo: Oh, man. Are you lucky. Oh, you are lucky. That's awesome, though. SpinRite saved your own bacon.

Steve: Saved me. I mean, like, days of, like, manually rebuilding her system. Because even though all of the vital data is saved, and I did have it imaged...

Leo: It was only four sectors that were damaged.

Steve: Yeah, actually those blocks can represent multiple sectors.

Leo: Right, right. Or areas.

Steve: It only had, yeah, four regions that had problems. But it was enough that it was like, eh, Hal.dll not found, we're going no further. So anyway, she understands that bypassing this was not the right thing to do. But, boy, I just - if anyone is within earshot who manages RAID software or can change RAID feature sets, I'm not kidding you, I don't think I've ever seen an option of refuse to proceed on a degraded RAID, but I want it. That is the right option, I've just learned, for many situations where you're, like, trying to protect someone who is just going to ignore it because, as you said, Leo, and we know, users will. They will just click right through.

Leo: Yup, okay.

Steve: It's just like they're in a hurry. They're busy. And it's like, no. This is when you stop. At a point when you've full recovery, and you don't need SpinRite, and you can still...

Leo: Before you need SpinRite.

Steve: Exactly, before you need SpinRite. But if it goes too far, of course, you always have SpinRite.

Leo: How long did it take to recover? Amun-Ra is asking in our chatroom.

Steve: It was, like, three hours.

Leo: Not bad.

Steve: Yeah.

Leo: Go get a cup of coffee.

Steve: Yeah.

Leo: Or two.

Steve: And I kind of went in and looked at it, and I - it's funny, too, because sometimes I hear people saying, is there any way to make it quiet? And I'm thinking, it doesn't make that much noise. But it kind of goes, you know...

Leo: It's reading, writing, reading, writing, er er er er er er, er er er er er er.

Steve: It, like, chortles to itself. No, I mean, I have it making - it's like I use the speaker to sort of talk to you.

Leo: Oh, you do stuff.

Steve: Oh, there's like a little R2-D2 thing going on. So it's like, yeah, okay, I think we'll make that an option in 6.1.

Leo: Silent mode, yeah.

Steve: Yeah. Okay. So law enforcement backdoors.

Leo: Yes.

Steve: This is really good. And so I'm going to take this, this is only a couple of minutes for me to go through each of these. But so many good points are made, and Matt and Jonathan take such different positions. So this is, okay, so - I just closed my notes because I had - I'm going to open them again real quick. I had a little background.

We've not talked about Matt Blaze much, more because he's not as much of an activist. But he got his bachelor's in computer science in '86 at City University of New York, his master of science in CS in '88 at Columbia, his master of arts in '89 from Princeton, his Ph.D. in '93 from Princeton. And he did his Ph.D., which always represents some pioneering new work, in - his thesis was "Caching in Large-Scale Distributed File Systems." So he's got all the credentials that a congressional committee wants to see. He's Dr. Matt Blaze, great degrees, and he is of course known as being the guy who found the problems with the government's previous attempt to propose a backdoor in cryptography.

So Matt writes: "Thank you for the opportunity to offer my testimony on the important public policy issues raised by cryptography and other security technologies. Since the early 1990s, my research has focused on cryptography and its applications for securing computing and communications systems, especially as we rely for increasingly critical applications on relatively insecure platforms such as the Internet. My work has focused particularly on the intersection of this technology and public policy issues. For example, in 1994 I discovered some fundamental technical flaws with the ill-fated "Clipper Chip," an encryption system designed by the National Security Agency, intended to provide a government backdoor to encrypted communications.

"I'm currently a professor in the computer science department at the University of Pennsylvania. From '92 until I joined Penn in 2004, I was a research scientist at AT&T Bell Labs. However, this testimony is not offered on behalf of any organization or agency."

So, Part I: "Robust Digital Security Technologies Are Vital to Protecting Our National and Critical Infrastructure." Matt writes: "It's difficult to overstate the importance of robust and reliable computing and communications to our personal, commercial, and national security today. Virtually every aspect of our lives, from our health records to the critical infrastructure that keeps our society and economy running, is reflected in or supported in some way by increasingly connected digital technology. The influx of new communications and computing devices and software over the last few decades has yielded enormous benefit to our economy, as well as to our ability to connect with one another.

"This trend toward digital systems, and the benefits we reap from them, will only accelerate as technology continues to improve. Preventing attacks against our digital infrastructure by criminals and other malicious actors is thus now an essential part of protecting our society itself.

"Unfortunately, modern computing and communications technologies, for all their benefits, are notoriously vulnerable to attack by criminals and hostile state actors. And just as the benefits of increased connectivity and more pervasive computing will continue to increase as technology advances, so too will the costs and risks we bear when this technology is maliciously compromised. It is a regrettable and yet time-tested paradox that our digital systems have largely become more vulnerable over time, even as almost every other aspect of the technology has, often wildly, improved.

"New and more efficient communication technologies often have less intrinsic security than the systems they replaced, just as the latest computers and other devices regularly suffer from unexpected vulnerabilities that are exploited remotely by malicious attackers. Large-scale data breaches and similar security failures have become so commonplace that they now only make the news when their consequences are particularly dramatic. Serious security failures are literally a daily occurrence, and it is not an exaggeration to characterize this situation as an emerging national crisis.

"Modern digital systems are so vulnerable for a simple reason: Computer science does not yet know how to build complex, large-scale software that has reliably correct behavior. This problem has been known and has been a central focus of computing research since the dawn of programmable computing. As new technology allows us to build larger and more complex systems, and to connect them together over the Internet, the problem of software correctness becomes exponentially more difficult. As this insecure technology becomes more integrated into the systems and relationships upon which society depends, the consequences become increasingly dire.

"While a general solution to the problem of software reliability and correctness has eluded us, and will continue to do so absent some remarkable, unexpected breakthrough, there are two tried-and-true techniques that can, to some extent, ameliorate the inherent vulnerability of software-based systems. One is the use of encryption to protect data stored on or transmitted over insecure media. The other is to design systems to be as simple as possible, with only those features needed to support the application. The aim is to minimize the 'attack surface' that any software vulnerabilities would expose.

"Neither the use of encryption nor designing systems to be small and simple are perfect solutions to the software security problem. Even carefully designed, single-purpose software that encrypts data wherever possible can still harbor hidden, exploitable vulnerabilities, especially when it is connected to the Internet. For this reason, software systems must be exposed to continual and resource-intensive scrutiny throughout their lifecycle to discover and fix flaws before attackers find and exploit them. But these approaches, imperfect and fragile as they might be, represent essentially the only proven defenses we have."

I'll just break there for a minute and say, you know, what I love about that is that it's sort of an in-your-face, like an in-your-face confession of a serious computer scientist saying, look. I mean, it's like he's encapsulating the drama that is this podcast, for example, every week.

Leo: The drama that is this podcast.

Steve: And we continue to sort of want to deny that, like, we're not almost there, that, like, oh, this'll be the last problem in OpenSSL. Oh, we've found all the problems in TrueCrypt. Oh, you know, this was a particularly bad Patch Tuesday, but at least the next one will probably be better because they seem to alternate. I mean, it's like we keep looking at these sort of as, like, anecdotally. But when you pull back, you realize the truth of what he has written, which is we are making systems more complex. And unfortunately they are becoming less secure. Today's automobile is provably less secure than yesterday's. The more features we add to our systems, and we seem unable to prevent that because users want them, the more opportunities there are for interaction.

So, I don't know, I just - this is, to me, sort of sobering because - and it's a perfect place for him to start because it does, sadly, it reflects the truth of our own experience here on the podcast for a decade. It's like, wow. We don't seem to be running out of problems to talk about. And his argument is computer science - and when you think about that, it's just math. Chips typically, I mean, chip errors are diminishingly rare. There've been like a division overflow in the Pentium and, you know, sometimes...

Leo: Like that's the one. I mean, you can think of one, really. Right?

Steve: Yeah, exactly. Right, right.

Leo: That's how rare it is.

Steve: Right. And beyond that, there is no excuse for software not working perfectly.

Leo: No.

Steve: Except the problem is the cost.

Leo: It's humans. Oh, okay.

Steve: It is too - yes. It's too expensive.

Leo: Right.

Steve: Yeah.

Leo: We should use LISP.

Steve: So, Part II. Oh, yeah, that'll help. Count your parens. So then his next section is "Law Enforcement Access Requirements Carry Great Risks." He writes: "U.S. law enforcement agencies have for the last two decades been warning that wiretaps and other forms of electronic evidence gathering are on the cusp of 'going dark.' These fears have been focused chiefly on the potential for criminal use of encryption - which, properly used, can prevent eavesdroppers from recovering communications content - as well as emerging decentralized communications paradigms, such as peer-to-peer communication, that are not easily intercepted with the same techniques that were used to wiretap traditional telephone calls. They call" - that is, law enforcement - "call for developers to incorporate 'lawful access' features into products and services that facilitate wiretapping.

"At first blush, a 'lawful access only' mechanism that could be incorporated into the communications systems used by criminal suspects might seem like an ideal technical solution to a difficult policy problem. Unfortunately, harsh technical realities make such an ideal solution effectively impossible, and attempts to mandate one would do enormous harm to the security and reliability of our nation's infrastructure, the future of our innovation economy, and our national security."

And he explains: "Access Requirements Make Encryption Vulnerable and Expensive." He says: "Let us consider first the relatively narrow problem of ensuring law enforcement access to encrypted communication. This is perhaps the simplest part of the law enforcement access problem, but it is dauntingly - and fundamentally - difficult to solve in practice without creating significant risk.

"Encryption systems encode messages in a way that prevents their decryption without knowledge of a secret, called a key. Ordinarily, only the parties to the communication know the key, which can be destroyed and forgotten as soon as the communication has ended and need never be sent to anyone else. In most well-designed encryption systems, third parties - including the developer of the software used to perform the encryption and the service providers who operate the infrastructure through which it traverses - do not know or have copies of these keys.

"The encryption is said to be 'end to end,' meaning it is conducted entirely between the communicating parties. End-to-end encryption is an important simplifying principle that allows for secure communication even over insecure media. It means that only the endpoints - the computers or devices being directly used by the parties - need to have access to and protect the keys, and the compromise of any other part of the system has no effect on the security of the messages. Securing the endpoints can sometimes be perilously difficult in practice, but it is a much simpler problem than securing the entire path over which messages are transmitted.

"Any law enforcement access scheme of the kind apparently envisioned by the FBI would necessarily involve a mechanism for the transmission and storage of sensitive secret keys to a third party, whether the government or some other entity that holds it. This approach is sometimes called 'key escrow,' 'key recovery,' or 'trusted third-party encryption.' The secret is held 'in escrow' by a third party. Key escrow was the widely criticized approach incorporated into the Clipper Chip in the early 1990s. It destroys the end-to-end design of robust encryption systems without any benefit to the application.

"There are several fundamental problems with such schemes. The most basic problem with law enforcement access cryptography is simply that we do not fully understand how to design them, even at an abstract, theoretical level. Any key escrow or lawful access cryptography system, by its very nature, increases its number of points of failure. Unfortunately, we do not understand the problem well enough to even quantify how much this reduces security, let alone identify a safe level of this reduction. The design and implementation of even the simplest encryption systems is an extraordinarily difficult and fragile process."

And, you know, doesn't all of our experience demonstrate that, Leo? I mean, we talk about how the brightest minds, the most theory, academicians looking at this stuff. And we're always finding problems in encryption systems.

So he continues: "Very small changes frequently introduce fatal security flaws. Ordinary, end-to-end, non-escrowed encryption systems have conceptually rather simple requirements; and yet, because there's no general theory for designing them, we still often discover exploitable flaws in fielded systems. Adding key escrow renders even the specification of the protocol itself far more complex, making it virtually impossible to assure that any systems using it will actually have the security properties that these systems are intended to have. It is possible, even likely, that lurking in any key escrow system will be one or more design weaknesses that allow recovery of data by unauthorized parties. The commercial and academic world simply does not have the tools to analyze or design the complex systems that arise from key recovery.

"This is not simply an abstract concern. Virtually all law enforcement key recovery or key escrow proposals made to date, including those designed by the National Security Agency - the Clipper Chip - have had unanticipated design weaknesses discovered after the fact." So he's noting that this is just not academic curiosity or theory. In fact, problems have been found in all of them.

"Frequently, subtle but devastating weaknesses in cryptographic systems and protocols are only discovered long after they're deployed in products and services, which means that sensitive" - can you say Heartbleed? There was many years of that bug that went undiscovered. So, I mean, what he's saying is exactly the rather embarrassing truth of this industry that we've been covering - "...long after they're deployed in products and services, which means that sensitive data was at risk from their very first day of use. Law enforcement access requirements make such hidden flaws far more likely to exist. Aside from cryptographic weaknesses, there are significant operational security issues. Third-party access, by its nature, makes encrypted data less secure because the third party itself creates a new target for attack.

"The FBI has not stated whether the cryptographic access mechanisms they desire would be operated centrally or by the vendors of individual products. Either approach creates its own inherent risks and costs. A centralized system becomes a large and highly attractive target, while leaving the task to individual product vendors introduces the likelihood that some vendors will lack the resources to securely manage the keys for their customers or will be specifically targeted for attack by national adversaries.

"Importantly from a business perspective, the infrastructure to properly support any scheme of this kind would be very expensive to operate. Further risks arise from the operational complexity of managing access to the secret keys. Key access centers must presumably be prepared to respond to law enforcement requests for key data 24 hours a day, completing transactions within a short time of receiving each request and in complete secrecy from the target of the investigation. There are thousands of law enforcement agencies in the United States authorized to perform electronic surveillance. The escrow centers must be prepared to identify, authenticate, and respond to any of them within a short timeframe.

"Even if we imagine relaxing these requirements considerably, for example, one day or even one week response time, there are few existing secure systems that operate effectively and economically on such a scale and under such tightly constrained conditions. It is simply inevitable that lawful access systems that meet government requirements will make mistakes in giving out the wrong keys from time to time or will be vulnerable to unauthorized key requests. Nation-state adversaries could be expected to be particularly interested in, and adept at, fraudulent access to our law enforcement access services."

And I've read enough. He goes on a bit, but that was the meat of his point.

Leo: We get the idea, yeah.

Steve: Yeah. And so I guess the thing that I liked about this was this was the testimony that was entered into the record that I'm just praying will put the brakes on any further entertainment of this. It was interesting, in the testimony itself, after - there were, like, maybe five people on the panel. There was the D.A. of, I can't remember his name now, like New York or New Jersey. And, I mean, went as far as to accuse the commercial interests of aiding and abetting, and we know what's going to come next, you know, the terrorists and the child pornographers and all that, and at that point drew some real ire from some people on the committee.

Leo: Good, good.

Steve: And in the testimony, one of the things that I thought was most interesting was that there was a gal, there was a woman there representing the FBI. And she basically explained that they have to have access to this information. And what was interesting was that, when she was pushed on it, when they said, well, you know, like, okay, you've heard the testimony from the cryptographers, from the experts, who say that it's not possible. So what do you want? You're asking for a backdoor into mobile device technology. And her response was, no, no, no. No, we're not. We're merely asking that, if we need to have access to communications, that there be a means for providing it.

Leo: Yeah. That's not a backdoor. That's just...

Steve: It's like...

Leo: ...a portal.

Steve: Okay? What? It's many, it's golden keys or some nonsense. And it was so funny because basically the sense from those on the law enforcement side was, first of all, there was a sense that the commercial interests were providing encryption and just thumbing their nose at law enforcement. And to which the - I think it was, is it Jack Lew? I think he said he was a recovering computer scientist, or computer something. He said, look. He said what's happening is the entirely expected response to massive overreach, and arguably ignoring the Constitution, of law enforcement as has been revealed over the last few years. What you're seeing is the public responding, wanting a non-surveillance state. And so the commercial interests are providing what the customers want. Customers want good, strong encryption. It's being provided. And guess what, the technology is there to do it.

So it was interesting to watch the struggle because the FBI was just, I mean, like was adamant in saying we must have access to communications where we're able to produce a lawful court order to compel somehow, that we somehow get access. And then they're saying, well, then you're talking about an encryption backdoor. And it's funny because the implication was these guys are clever. The technical people, they'll figure a way. We want you to make it a law, and then they'll figure out how to do it. And so that's sort of where this thing got left was we don't believe there isn't a way. Silicon Valley is full of smart people who, like, solve any problem that they need to solve. If we make them need to solve this, they will solve it. And so...

Leo: So that's kind of the typical ignorant - it's all magic, so just make some more of it.

Steve: Yeah. We don't understand how this works.

Leo: We don't understand it, so...

Steve: You know, look what we're holding in our hand now.

Leo: It's magic. You guys are magicians.

Steve: Add some more magic. Just add...

Leo: Why don't you make some more magic? You know you can.

Steve: Yeah. And I won't go through Jonathan Mayer's note, but I do have a link to it in the show notes, if anyone's interested. Basically he talked about the practical problems of securing a platform, like how would this work? Say that law enforcement required - and he took Google and Android as an example. Law enforcement did require Google to be able to decrypt the storage on an Android device. Okay. So now Android devices can be, under a law enforcement order, decrypted. But what's to prevent an application running on Android from doing its own encryption, like TrueCrypt does its own encryption, even though the drive can do it, too. We know you can always add your own encryption to your own data. And so now we have a problem with, like, how are we going to prevent applications from doing their own local encryption?

Well, okay, now we need a way of finding out whether applications are able to do their own encryption and mandating that they use a common cryptographic library that has a backdoor in it so that it's possible to get into sort of third-party-encrypted solutions. But this is an open platform. And the Google Play store is not the only place you can get Google apps. You can side load them from developers, like from the original sourcing websites. And you can get them from other app stores on the 'Net. And this is not only in the U.S. It's a global platform. And so there could be other sources of these.

So how do you go about actually, I mean, in fact, enforcing this? And it's funny, as I was reading this, what I was put in mind of was the arguably failed War on Drugs, is you make something illegal. You outlaw something that you cannot, you fundamentally cannot enforce. And what you do is you just create a black market. You drive it underground, active and thriving and operating, yet against the law, because it is unenforceable. And so when you come down to it, what the government wants is unenforceable. And Jonathan, who is a computer scientist in addition to being an attorney, he marches through this, how one layer after another, it's like the cat is out of the bag. Strong crypto exists. You cannot de-create. You cannot roll back time and no longer have strong crypto. It exists. It's math.

And so he drew the point that, even if you went to the point, say that you somehow came up with some way of controlling the libraries and controlling apps. You built a kill switch. First there was a problem of app IDs and the Google kill switch, but then apps could have random IDs if they wanted to avoid the kill switch. So then Google would have to have a more robust scheme for, like, searching out strong encryption in the same way that it looks for viruses. And that's - we don't know how to do that. That's an unsolved problem.

But even if all of that was solved, you could still bring up a web page, where the server itself is serving web content, which we now know can issue JavaScript where you perform script locally on the page and then send the message somewhere. So the only way then to control that is to censor the Internet, to have the U.S. government pulling down websites that are delivering secure crypto. And the point being made that, when you actually say, how do we pull this off, the answer is you can't. You cannot. You cannot kill secure crypto. It now exists. And it's creepy to just sort of - to think that it would be against the law to strongly encrypt your own file, like to upload to the cloud. That, like, you'd have to use a government-approved crypto so that they have some way to get the key. I mean, it just - it's just not practical. It can't be done.

So anyway, I hope this is - it'll be interesting to see how this goes. But I don't know how this tension gets solved because I don't see law enforcement giving up. They're just going to keep saying we need this, we need this, we need this. And one of the things that comes up in the testimony is, has there yet been a situation where a crime was not solved that would have been if you had access to the contents of a mobile phone? And [Daniel] Conley, who was the D.A. guy [Suffolk County, MA], I think, and he, like, hemmed and hawed and didn't really know of one. So it's like, uh-huh, yeah, well, you guys want everything, but I'm afraid you're not going to get this.

Leo: Well, we want hyperspace travel. Doesn't mean we're going to get it. Okay. Good stuff, Steve.

Steve: So, interesting testimony. At my YouTube, my new YouTube channel...

Leo: Love it.

Steve: YouTube.com/SGgrc, it's an hour and 15 minutes. Really, nothing I just shared with you is repetitious because what Matt did there was different. I verified that he had a much shorter statement, like five minutes or less. But some really good tension between the guys on the panel that did know their computer science. Oh, and there was one point, apparently the FBI had a recommended security practices, sort of like for Mom and Pop. And one of them was put in, you know, encrypt your phone with a strong passphrase was in a bulleted list. It had been quietly removed from the FBI website some time before. And one of the guys on the panel caught that. It's like, oh, that's interesting. You're no longer...

Leo: We don't want you to do that, yeah.

Steve: ...suggesting that people encrypt their phones.

Leo: Yeah, stop that.

Steve: With this passcode.

Leo: Knock it off.

Steve: Use 1234, please.

Leo: Yes, thank you. Hey, next week, questions and answers; right?

Steve: Yup, yup.

Leo: So if people have some questions about this topic or any other, go to GRC.com/feedback. That's where Steve takes questions. Or you could tweet him, @SGgrc. And we'll get to 10 or more questions next week.

Steve: Yup. And whatever news happens in the meantime.

Leo: Yeah. While you're at GRC.com, don't forget SpinRite, the world's best hard drive maintenance and recovery utility.

Steve: Boy, and, you know, I'm now...

Leo: And it works for Steve.

Steve: But this one pulled it back from the hairy edge.

Leo: You're a believer, yeah.

Steve: I'm now - I'm going to make an annual commitment to myself with Sue, because this went three years, just to go down...

Leo: Wait a minute, she was hitting Okay for three years?

Steve: No, I don't know. She didn't tell me. She was, I mean, she knew she'd, like, it was bad. She should have told me. And I didn't push. But I just made her promise never to do this again. But, no, but my point was preventative maintenance because that's the flipside is SpinRite fixes this stuff before it can get bad. And so more than once a year is better if it's convenient. But she's down there, I've got to pull her whole system apart, and it's covered with dust bunnies and gross, and then I have to take it apart and, you know, all of that.

Leo: Eww.

Steve: So it's like, okay, at least once a year. I'm committed. I'm running SpinRite on my own drives because I have to. Because, boy, whew.

Leo: I won't say anything about backup. You had other backup; right?

Steve: Oh, yeah. Actually, I'm grandfathered into Jungle Disk, so I've still got the free...

Leo: Oh, okay. So you wouldn't have lost anything.

Steve: Oh, no, no, no. I've got images and - but it's so much easier not to have to rebuild the system and then, you know, copy the files back and everything. Just, and everybody knows, I mean, some people say, hey, SpinRite's $89, but drives are, like, nothing now. And I go, yeah, and what's your time worth? You know? It's like, SpinRite will get the drive back, as we keep hearing, and prevent it from dying in the first place. So, and you get to use it forever. People are using it from 25 years ago. And as soon as I get SQRL finished, I'm back to developing the next version, which everyone who has it will be able to upgrade to for free. So I don't know where you can get a better deal than that.

Leo: Level 2 for the regular maintenance; right?

Steve: Yes.

Leo: While you're at GRC.com, you can also find lots of free stuff. He's very generous with his time, including SQRL, which is going to save the world.

Steve: It's coming soon.

Leo: All that other stuff. GRC.com. And 16Kb audio and handwritten Elaine Farris-style transcriptions of each and every show.

Steve: Oh, yeah, the Word document was 100 pages last week.

Leo: Golly. It was a long show.

Steve: She broke a record.

Leo: We're writing a novel every week here now. Or at least a novella. That's amazing. You can also get full quality audio and video versions from our site, TWiT.tv/sn for Security Now!. And you can subscribe, and you can get it almost everywhere. It's on all the podcast clients. We have our own TWiT clients on iOS, Android, Windows Phone, Roku, everywhere you want to be, Windows itself. And, oh, and if you want to be here live, we have three nice people in the studio. A couple of honeymooners. Yup. Which is weird, but I'm not going to - I don't want to, you know, say anything or anything. You just email tickets@twit.tv, we'll make space for you. You can also watch live. We do the show every Tuesday afternoon, 1:30 Pacific, 4:30 Eastern time, 2030 UTC, at live.twit.tv. Steve, thank you so much. We'll see you next time on Security Now!.

Steve: Thanks, Leo.