Flip Feng Shui

Description: Leo and I discuss the continuing woes of WoSign. Autonomous micro-recon drones turn out to be real. A new crypto attack on short block ciphers prompts immediate changes in OpenVPN and OpenSSL. We introduce a new Security Now! Abbreviation, "YAWTTY," Yet Another Way To Track You. We continue with discouraging social engineering experiment, another clever USB attack, a bunch of fun miscellany, and a look at the weaponizing of Rowhammer with "Flip Feng Shui," the most incredibly righteous and sublime hack ever, ending with our follow-up to last week's Security Now! Puzzler.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-576.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-576-lq.mp3

SHOW TEASE: It's time for Security Now!. More troubles at WoSign. Mozilla's thinking about what they're going to do next. A microdrone reconnaissance tool. And how your smartphone's light sensor could help track you. Plus we'll take a look at something, that Rowhammer attack they're calling "Flip Feng Shui." How it all works, next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 576, recorded Tuesday, September 6, 2016: Flip Feng Shui.

It's time for Security Now!, the show where we cover the privacy and security of your systems online. It is a dangerous world out there; but, boy, thank goodness we have this guy on our side, Steve Gibson of GRC.com. Hey, good morning, or good afternoon, Steve. How are you?

Steve Gibson: Hey, Leo. Great to be with you again as you get yourself prepared for a trip. Do we have you next week, but then not for a couple weeks after that?

Leo: Yeah, let's see. Next week would be the 13th, but not - yeah. So one week. And then I think Father Robert, I'm not sure if - Father Robert, I think, or maybe - I'm not sure. We have to figure out which host. It's either Jason, Megan, or Robert. I'm not sure. I'm taking two weeks.

Steve: [Crosstalk].

Leo: Yeah, well, it has to be somebody who can read ads, frankly. They have to be approved for doing the ads because we don't ever want to make you do that. So on the 13th I'll be here. The 20th and the 27th I will not. And nor will I be back on October 4th because that's the day we're flying home.

Steve: Okay.

Leo: So I'll miss three episodes of this show.

Steve: Okay. And I am recording one of them with Father Robert. We've already arranged.

Leo: That's right, because he's not here for one of those weeks.

Steve: Right, right.

Leo: Okay. Yeah. Sorry. And then I'm thinking about doing something nutty. I don't know if you heard Windows Weekly earlier, but Mary Jo Foley convinced me maybe not to bring any technology with me on the trip.

Steve: Yeah, uh-huh.

Leo: You sound skeptical. Why ever so? Well, I have to bring a camera. I'm going to bring a digital camera.

Steve: You'll certainly have a camera.

Leo: So I don't mean that. But, I mean, not bring a laptop. Not bring a tablet.

Steve: But, I mean, even before the show you were talking about, well, maybe I'll need this phone as a backup for my main phone in case...

Leo: I have to have a phone.

Steve: ...that main phone does, you know. So it's like, okay, there's an escalation already beginning.

Leo: I'm already - I'm bargaining. It's called bargaining.

Steve: Right. Isn't that one of the stages of acceptance?

Leo: Yes, bargaining. But maybe if I just bring - no, I have to bring a phone for safety purposes. I'm going to bring a Google 5 phone because they have international calling.

Steve: Oh, that is a fabulous rationalization. It hadn't occurred to me before.

Leo: Well, yes.

Steve: You need it for safety.

Leo: Safety. But I don't plan to turn it on.

Steve: So you could hit somebody over the head with it? Because it'll bend, you know.

Leo: You know, actually I've used it. In Venice we got lost because it's a maze, and all the canals look the same. So then you fire up the phone, you fire up Google Maps, and you get your way, you make your way home. So for that, or if I'm, you know, I've fallen and I can't get up, I can call 911.

Steve: You could do that on your watch, though, couldn't you?

Leo: I'm not going to bring a smart watch, either.

Steve: Wow.

Leo: Bring one phone. The only reason the backup...

Steve: Now, are you a watch wearer? Will you bring a dumb watch or no watch?

Leo: No, I - oh, actually, boy, if I don't have a phone, how am I going to know what time it is? I might have to wear a watch.

Steve: The point of being on a vacation is it doesn't matter what time it is.

Leo: Well, it matters a little bit because there's excursions, and there's massages. You've got to keep track of the massages, dude.

Steve: You're highly scheduled on your vacations.

Leo: You know, the worst thing was reading materials. But I could bring a Kindle, or I can bring a - actually, I'm thinking of bringing a book, a physical book.

Steve: Love my Kindle. The Kindle is just...

Leo: Kindle's nice because it's small and light.

Steve: Yeah.

Leo: But what if I bought a really thick book?

Steve: Well, that you could hit people over the head with for security.

Leo: And then I was thinking, well, maybe I'll bring an iPod. You know, an old iPod doesn't have any - what I'm more worried about is, like, checking social media, sending pictures on Instagram, stuff like that. What if I didn't do any of that for two and a half, three weeks? Whew. I might not ever do it again.

Steve: Wow.

Leo: Just a thought.

Steve: Well, you know, there have been people, I mean, I don't understand this because I look around, and I see people who, with their family...

Leo: Live on their phone.

Steve: A family at dinner, all four of them have phones. And this is even post-Pokemon Go.

Leo: No, it's more common than not.

Steve: Yeah, it is. And it's just like, they're not having a conversation. They're all into their feeds of one sort or another, or [crosstalk] they're doing.

Leo: I thought it would be good for my marriage, good for my peace of mind. I could see the sights. I don't know. It's crazy talk, isn't it. What is Flip Feng Shui, and why

are we talking about it today?

Steve: Well, yes. "Flip Feng Shui" is the title of today's podcast. This is without...

Leo: People are, by the way, timing their SpinRite purchases to hit on Tuesday afternoon.

Steve: I always appreciate it.

Leo: Yeah, that's nice.

Steve: This is, without a doubt, the most righteous, sublime hack we have ever covered. It is multilayered, incredibly clever. And I wanted to do a Q&A this week, but this came along, and it's like, okay, stop the presses. This is just - this is too fun. And it's complicated. But I know that our listeners are going to get a kick out of it. Basically it is the weaponizing of Rowhammer, which we talked about, it was March 2015, so about a year and a half ago we covered Rowhammer. I will review what Rowhammer is and then explain how a group of researchers managed to use Rowhammer to exfiltrate the private key from other processes sharing the same cloud server as them. It's just amazing.

But we have a lot of other stuff to talk about. We've got the continuing woes of WoSign. It turns out that autonomous micro recon drones are real. A new crypto attack on short block ciphers has prompted an immediate change in OpenVPN and OpenSSL. We have a new Security Now! abbreviation. We're officially unveiling YAWTTY.

Leo: Carefully. What the hell?

Steve: Well, that's how you would pronounce it: Y-A-W-T-T-Y. YAWTTY. That's Yet Another Way To Track You.

Leo: Okay.

Steve: So we have the first officially designated YAWTTY. Then we have the results of a discouraging, or just the discouraging results of a social engineering experiment. Another clever way for USB to attack our systems. A bunch of fun miscellany. And I did get a couple questions that I had stumbled upon answered there. And then we're going to do the serious deep dive into Rowhammer.

And I almost forgot, of course we have the follow-up to last week's puzzler of what to do if an Internet appliance that does not allow itself to be modified in any way and only knows how to verify and approve certificates signed with SHA-1, what do you do in the future, or even soon, when any server certificate that it needs to access cannot be renewed as an SHA-1? So we will give some ideas about that, too. So, yes, all kinds of fun stuff.

Leo: It could be pronounced YAWTTY.

Steve: But that's not nearly as much fun.

Leo: Okay. Okay.

Steve: YAWTTY. Yeah, I guess it could, yeah.

Leo: YAWTTY.

Steve: Except that TTY is the official abbreviation for teletype. And we used to call those "titties."

Leo: Really?

Steve: Yeah, TTY, that was the official term.

Leo: I was hanging around with the wrong people. I had no idea, TTYs. I just call them TTYs because that's pretty much as quick as the other one.

Steve: Well, you know, we've got JIF and GIF, and so now we've got TTY.

Leo: YAWTTY. Oh, lord. Flip Feng Shui.

Steve: So we need to look a little bit, I think this will be interesting to people, at the sort of behind the scenes of the process that browser vendors go through in dealing with problems with certificate authorities. After, and we talked about this last week, the revelations that this Chinese CA WoSign had woefully poor signing security policies, a thread was created in the Mozilla Dev Security Policy group which had 155 posts from 34 people, many of them notable in the industry. And the initial posting by Mozilla, I think, really helps to clarify sort of where they stand.

And I know that I've had some feedback from our listeners since we talked about it who are taking the issue of having certificates trusted from authorities that they probably just have no need for. And that's the problem is we have this, as we've said, this blanket "trust everyone," well in excess of who we actually need to trust. And everyone knows that's the wrong way to do things. That's like having, in the old days, where you had a firewall that was open, except you would block the ports that people were using to attack you. It's like, no. Flip that around. Block everything, and then open the ports for the traffic that you know you need. That's the way to do it.

So we're now sort of still limping along with the reverse of that, sort of the equivalent of a trust everybody, respond to problems approach. And it'll be interesting to see when

that changes. I think it probably has to because there is a delicate line that a browser vendor is walking because certificate authority systems and solutions, companies, are commercial entities that are typically for-profit, Let's Encrypt being a notable exception. And Let's Encrypt has put pressure on other CAs to create, to arrange somehow their own free certificate solutions. So basically that lowest level of validation, the domain validation, where all you're doing is proving ownership of a domain, that's now no longer something that a certificate authority can expect to profit from.

But of course there are reasons to have much better validation. And I think that's only going to get stronger, that is, these reasons for that are going to get stronger in the future. So what that means is that a browser vendor has a great deal of power because, as we move from HTTP to HTTPS, having a certificate for a website is no longer optional, and the website needs to get it signed from a certificate authority that can vouch for the identity of the website.

So what that means is that it's no longer, well, the site works except you can't do something without a certificate that is recognized. Increasingly, it's nothing works if a server isn't issuing a certificate that is trusted. So what this means is that the web browsers have a huge amount of power. And they recognize they have an amazing amount of power essentially over the economic well-being of CAs because, if a browser decides it's going to pull its trust from a certificate authority for that browser, in a world where HTTPS is now de rigueur, that CA is in serious trouble.

So Mozilla wrote to sort of create a foundation for and to explicitly ask for comments. And what I liked about this was this is more information than we've had before. They wrote: "Several incidents have come to our attention involving the CA WoSign. Mozilla is considering what action it should take in response to these incidents." And notice it's plural. We actually have three, not just one.

"This email sets out our understanding of the situation. Before we begin, we should note that Section 1 of the Mozilla CA Certificate Enforcement Policy says: 'When a serious security concern is noticed, such as a major root compromise, it should be treated as a security-sensitive bug, and the Mozilla Policy for Handling Security Bugs should be followed.' It is clear to us," they write, "and appears to be clear to other CAs based on their actions, that misissuances where domain control checks have failed fall into the category of 'serious security concern.'"

So the first incident. And unfortunately they decided to number them from zero, which is just confusing. It's like, sometimes it's fun. Sometimes it's appropriate. Not here. So, okay. For example, we should have had a zero with the podcast. But at least now our podcast number is actually the podcast number.

Leo: We didn't really think about that, did we.

Steve: No. Anyway, so incident zero, which I wrote as "first incident" in my notes because that's annoying, said: "On or around April 23rd of 2015" - okay, so a year and a half ago - "WoSign's certificate issuance system for their free certificates allowed the applicant to choose any port for validation." Meaning, like, 8080 or 8090, that is, not forcing 443 for validation. "Once validation had been completed, WoSign would issue certificates for that domain. A researcher was able to obtain a certificate for a university by opening a high-numbered port" - in this case greater than 50,000 - "and getting WoSign to use that port for validation control. This problem was reported to Google, and thence" - and thence? Thence. Okay. Maybe they should number from zero after all - "to

WoSign and resolved. Mozilla only became aware of it recently."

So they said: "Before the recent passage" - and this gets into the CAB Forum proceedings - "passage of Ballot 169 in the CAB Forum, which limits the ports and paths which can be used, the Baseline Requirements said that one acceptable method of domain validation was 'having the applicant demonstrate practical control over the fully qualified domain name (FQDN) by making an agreed-upon change to information found on an online web page identified by a uniform resource identifier (URL) containing the fully qualified domain name." Which is to say, for example, I want to get a certificate from DigiCert, and so they give me something to put at GRC.com. Then I say, "What you gave me is there." Then they go fetch it from GRC.com and go, okay, yes, you're validated. And that is the protocol, for example, that Let's Encrypt uses for their automated system.

So they continue: "This method, therefore, did not violate the letter of the baseline requirements," that is, that WoSign was not doing any port restriction and allowed a high-numbered port to be used. "However, Mozilla considers the basic security knowledge that ports over 1024 are unprivileged" - remember that only processes running as root are able to open ports below 1024. Users can open ports above. So just any unprivileged app running on a server can set itself up as a web server on that machine, and that's what this university student did in order to demonstrate this problem.

So, however, "Mozilla considers the basic security knowledge that ports over 1024 are unprivileged should have led all CAs not to accept validations for domain control on such ports, even when not documented in the baseline requirements." So they're sort of saying, yeah, they should have known. But, okay, technically maybe not.

Second point: "The misissuance incident was not reported to Mozilla by WoSign as it should have been, and the misissuance incident did not turn up on WoSign's subsequent baseline requirement's audit," also as it should have. That's the first problem.

Second problem: In June of 2015, so a couple months after that, "an applicant found a problem with WoSign's free certificate service, which allowed them to get a certificate for the base domain if they were able to prove control of a subdomain." And that's what we talked about last week. "The reporter proved the problem in two ways. They accidentally discovered it when trying to get a certificate for med.ucf.edu and mistakenly also applied for www.ucf.edu, which was approved. They then confirmed the problem by using their control of theiraccount.github.com to get a cert for github.com, github.io, and www.github.io.

They reported this to WoSign, giving only the GitHub certificate as an example. That cert was revoked, and the vulnerability was fixed. However, recently they got in touch with Google" - and this is why this became in the news just a couple weeks ago - "to note that the ucf.edu cert still had not been revoked almost a year later."

So Mozilla notes three things: "The lack of revocation of the ucf.edu certificate, still unrevoked at time of writing, although it may have been by the time of this posting, strongly suggests that WoSign either did not or could not search their issuant databases for other occurrences of the same problem." Meaning that they should have been - a responsible CA in realizing they had a problem would have retroactively or retrospectively examined their entire past for other possible exploits of that vulnerability and then revoked those certificates.

Leo: Which means there may be many, many, many, many; right?

Steve: Yes, yes. "Mozilla considers such a search a basic part of the response to the disclosure of a vulnerability which causes misissuance, and expects CAs to keep records detailed enough to make it possible." I mean, that seems like a clear requirement for having as much responsibility as a CA does.

Second point: "This misissuance incident was not reported to Mozilla by WoSign as it should have been." And, finally, "This misissuance incident did not turn up on WoSign's subsequent baseline requirements audit."

And, finally, the third incident: In July of 2016, so last month, "it became clear that there was some problem with the StartEncrypt automatic issuance service recently deployed by the CA StartCom. As well as other problems it had" - and actually we discussed those about a month ago - "which are outside the scope of this discussion, changing a simple API parameter in the POST request" - oh, this is what we discussed - "in the POST request on the submission page changed the root certificate to which the resulting certificate chained up." Okay. This is different than what we talked about, but this is obviously related.

Leo: Geez.

Steve: But basically they were trusting any parameters that the page returned to them, which meant it was trivial for someone to generate fraudulent posts with parameters they specified, rather than that they had received in a page from the remote server, from the CA's server, and get up to all kinds of mischief. In this case, you could change the root certificate, which the resulting certificate was chained up to.

"The value '2' made a certificate signed by StartCom Class 1 DV [domain validation] Server CA, but a '1' selected WoSign CA Free SSL Certificate G2, and '0' selected CA [as the name], another root certificate owned by WoSign and trusted by Firefox." So, and what we now know as a result of additional information, is that WoSign quietly acquired StartCom some time ago, which explains this comingling.

Leo: Oh.

Steve: And so StartCom apparently, in some sort of merger of technology or who knows what, added these features. Or maybe it's WoSign's code which StartCom added their, you know, WoSign had zero and one, and StartCom now has two. So they may actually all be issuing, after this acquisition, through WoSign, yet having the surface on the face of StartCom, unless you change a parameter in the POST, in which case you can have your cert signed by, you know, name...

Leo: Anybody you want. Anybody.

Steve: I wonder what number three does. Anyway: "Using the value '1' [Mozilla writes] led to a certificate which had a notBefore date" - which is the usage start date - "of the

20th of December 2015." Okay, so December 20th, 2015. That's, what, 11 days before the end of SHA-1 could be generated. And, okay. Oh, and it was signed using SHA-1.

So Mozilla says: "The issuance of certificates using SHA-1 has been banned by the baseline requirements since January 1st, 2016. Browsers, including Firefox, planned to enforce this by not trusting certs with a notBefore date after that date." Now, see, because a notBefore date is the issuing date. And so the point is that, even now in 2016, if you fudged that POST parameter to a "1," so that you've got WoSign CA free SSL certificate, you would get a notBefore of December 20th, 2015, and SHA-1, clearly designed in order to fudge the system, that is, to allow, I mean, the only reason you would set a notBefore date like that is to allow for some calendar flop. So you don't want it to be like December 31st. You want to push it back a little further, just so that calendars and clocks and things being a little bit off don't cause a problem, but specifically so that you can issue valid SHA-1s in 2016, which is clearly what they were doing.

So Mozilla continues, saying: "But in the case of Firefox, the fix had to be backed out." Oh, Firefox planned to enforce this by not trusting certs with a notBefore date after January 1st, 2016, meaning any certs issued in 2016 or after. "But in the case of Firefox, the fix had to be backed out due to web compatibility issues." Which is interesting in itself. "However, we are considering how/when to reintroduce it, and CAs presumably know this."

Another point: "The issuance of backdated certificates is not forbidden, but is listed in Mozilla's list of problematic practices. It says, 'Minor tweaking for technical compatibility reasons is acceptable, but backdating certificates in order to avoid some deadline or code-enforced restriction is not.' WoSign deny that their code backdated the certificates in order to avoid browser-based restrictions. They say: 'This date is the day we stop to use this code.' If that is true," writes Mozilla, "it is not clear to us how StartCom came to deploy WoSign code that WoSign itself claims to have abandoned."

Then they say: "It seems clear from publicly available information that StartCom's issuance systems are linked to WoSign's issuance systems in some way. Nevertheless, it should not have been possible for an application for a cert from StartCom to produce a cert signed by WoSign." So this whole notion of cross-domain CAs makes everybody a little uncomfortable.

And, finally: "This misissuance incident was not reported to Mozilla by WoSign as it should have been. Taking into account," they conclude, "all these incidents and the actions of this CA, Mozilla is considering what action to take. Your input is welcomed."

Leo: Cut them the heck off.

Steve: Yeah, exactly.

Leo: I mean...

Steve: And so the reason I gave that rather elaborate introduction is that this, you know, is Mozilla doesn't want to hurt this company, yet they want to protect their users. And so it's a balance; you know? WoSign is in business. And so what I think this really means is that this has to be done responsibly. So clearly, so you can see what they're

doing here is laying out their case to the public, saying here are all the problems. What do people think? And again, 155 posts from 34 different authors responding to what they think, so...

Leo: Does anybody say, "Oh, don't worry, just forget it?" Probably not, yeah.

Steve: So I got a kick out of this. We were talking about the autonomous drone horror scenario of millions of drones in a semi-tractor-trailer, or three of them, actually, being opened up on the shores of New York somewhere, and then flying into the city and wreaking havoc. It turns out that they actually exist. Not weaponized, but this was interesting. There's a company, a San Diego, California-based company called Shield AI - they're a tech firm specializing in mini reconnaissance quadcopters - which has just been awarded a million-dollar contract to provide drones that scour urban battlefields and beam back critical information to the U.S. Army. And I put in parentheses here in my notes, "Hopefully not U.S. domestic urban battlefields." Presumably they mean war zones, urban war zones in the Middle East.

Leo: I don't think they're making a distinction, really.

Steve: Yeah, I know, and that's of course a concern because these little things - well, so anyway, Shield AI's mission statement suggests their flying machines can "help solve the intelligence deficit..."

Leo: Oh, yeah.

Steve: Uh-huh, "that can often mean the difference between life and death for military personnel dropped into densely populated war zones. A notice on the U.S. business procurement website, FedBizOpps, reveals the company has recently been contracted by the U.S. Army and Naval Special Warfare Command to work on autonomous tactical airborne drones. There is little detail in the $1 million contract awarded by the Defense Innovation Unit Experimental" - so that's DIUx - "the new tech-focused outfit tasked with gaining the technological jump on America's enemies." This DIU is a new government group, sort of the equivalent of a DARPA back in the day, but one that seems a little more defense focused, even though the "D" in DARPA was defense. Anyway, so this is a "nine-month prototype project in the area of autonomous tactical airborne drones."

And so on the site there's a video which shows this thing buzzing around. And it sort of gives a sneak peek into what Shield AI has done and demonstrated in order to be awarded this contract. It shows a microdrone which is launched simply by someone tapping on the screen of a smartphone, which takes off and, with no remote control, no piloting, and full autonomy, maps the narrow corridors of a building. And to me it looked like, I'm sure it was, a storage rental facility because it was those long corridors of the rollup steel doors, you know, inside. And it produces a full map of the facility without any human assistance.

Leo: I mean, this is great. If you're, I mean, geez, if you're getting landed in an urban combat situation...

Steve: Yes, yes.

Leo: It would be - the fog of war is dangerous. It'd be great to have.

Steve: Yes. And imagine just like turning a bunch of these loose and having them fly out in different directions, especially if they're a grid, and they're able to talk to each other so that they're not bunched up.

Leo: It's an inevitable technology.

Steve: Yeah.

Leo: But the problem is, of course, then you contemplate, well, how might it be used domestically? And if you say, "Oh, they'd never," the city of Baltimore just got busted for secretly surveilling the entire city, but with Cessnas, not drones. But flying Cessnas and surveilling the entire city without disclosure. And, you know, this is going to happen.

Steve: Well, and we have covered stories, or, yes, the press has covered stories of our military industrial complex actively marketing that technology to municipalities. You know, drone technology, for example, in order for civilian law enforcement purposes, but still it has wandered off away from the battlefield, where it was presumably originally targeted.

So this is a good one. And this is an example of why we have to keep security researchers protected from retaliation for what they find. Given that the reality is, we were talking about at the top of the show, that security is not absolute, but to varying degrees our devices are resistant to attack, we need good attackers to poke at all of this technology so that problems could be identified. What's really interesting here is that this is not a bug that was found, but it was time to retire some tried and true crypto which is just no longer sufficiently secure, as a consequence of its fundamental design. And that's the block size.

New ciphers have a block size. For example, the Rijndael cipher, which was chosen as the AES cipher, is 128-bit block. And back on the podcast where we talked about how symmetric ciphers worked, I explained that you could think of a cipher, encryption, as a one-to-one reversible mapping of every combination - in the case of Rijndael, 128 bits into a different 128 bits. And since it's reversible, that means that every time you put the same 128 bits in, you get the same different 128 bits out under the influence of a key. And that's what makes this whole thing so elegant.

So think of the cipher as a black box: 128 bits go in, a different 128 bits, probably different, I mean, in theory, one of them could go right through. But maybe not. A different 128 bits comes out. And you can reverse it. That would be encryption. Decryption is just the reverse of that. And the point is that the key you use uniquely determines the mapping between the input 128 and the output 128.

Well, so clearly one of the important facets of the cipher is how big, how many bits, how long is the key, which tells us how many keys there are. And Rijndael can have 128-,

196-, or 256-bit keys. So its key size is scalable. The block size is fixed. Older ciphers have smaller block size. And I should say block size is that 128 bits because it's considered a block because, if this was data, a block of 128 bits of data would be encrypted as a whole, at once, into a different 128 bits.

So in older ciphers like DES, which was 56 bits, or Blowfish that was 64, those used 56- or 64-bit blocks. Now, they're still secure, but there is a problem. And they can succumb to the so-called "birthday attack." And this is an insidious problem. We've talked about sort of the whole - the birthday paradox in the past. You get a bunch of people together, and everyone tells the group their date of birth - not the year, the month and day. And there will be an unintuitively high probability that two people in a relatively small group have the same month and day of birthday because - and again, this is one of the places where people are just bad with probability. Our brains don't work in probabilities very well.

So the math holds. And the trick is that it's not - we're not asking does one person have the same, you know, does any person have this specific birthday. It's does any person have anyone else's birthday. And that's the trick behind the birthday attack. So it turns out that it is no longer considered safe, and has been proven to use these small block ciphers. There will be a paper presented at the end of next month, on October 23rd, at the ACM Conference on Computer and Communications. I don't even know which annual, like what number that is, but that's been going on forever.

So these guys in their abstract explain. They said: "Cryptographic protocols like TLS, SSH, IPSec, and OpenVPN commonly use block cipher algorithms such as AES, 3DES, and Blowfish to encrypt" - wait a minute. Maybe the key was, yes, I'm sorry, I'm just remembering that DES's key was 56, I think it's 56 bits because that's why you use 3DES. So I think DES is itself a 64-bit block, but it uses a short key. And that was the problem was that that key became too short, too soon. So 3DES serializes three DESes in a row, each with its own piece of whatever that key length is. I want to say 56 because I remember 112. I think it's 56 times three is the total key length in 3DES. But the problem is the block size, the fundamental block size is still 64 bits, which is a problem.

So these guys continue: "...to encrypt data between clients and servers. To use such algorithms, the data is broken into fixed-length chunks called 'blocks,' and each block is encrypted separately according to a mode of operation," which we've talked about extensively in the past, you know, CBC (cipher block chaining), codebook and so forth. "Older block ciphers, such as 3DES and Blowfish, use a block size of 64 bits; whereas AES uses a block size of 128 bits. It is well-known in the cryptographic community that a short block size makes a block cipher vulnerable to birthday attacks, even if there are no cryptographic attacks against the block cipher itself." In other words, nothing wrong with either of those ciphers. They've withstood the tests of time. The problem is they just don't have enough combinations of bits-in to bits-out in today's computational environment.

"We observe," they write, "that such attacks have now become practical" - well, and we'll see what that means. That's, you know, but again, stretching the word "practical," and we have to because we want to take these out of service before they become really a problem - "for the common usage of 64-bit block ciphers in popular protocols like TLS and OpenVPN. Still, such ciphers are widely enabled on the Internet. Blowfish is currently the default cipher in OpenVPN, and 3DES is supported by nearly all HTTPS web servers, and currently used for roughly 12% of HTTPS connections between mainstream browsers and web servers.

"We show," they write, "that a network attacker" - now, here's where we get to the

questionable practicality; but, still, something we need to understand. "We show that a network attacker who can monitor a long-lived 3DES HTTPS connection between a web browser and a web server can recover secure HTTP cookies by capturing around" - okay, sit down - "785GB of traffic." So, yes, three quarters of a terabyte.

So, "In our proof-of-concept demo," they write, "this attack currently takes less than two days" - actually, it's like 38 hours - "using malicious JavaScript to generate traffic. Keeping a web connection alive for two days may not seem very practical, but it worked easily in the lab. In terms of computational complexity, this attack is comparable to the recent attacks on RC4. We also demonstrate a similar attack on VPNs that use 64-bit ciphers, such as OpenVPN, where long-lived Blowfish-encrypted connections are the norm. Countermeasures," they write, "are currently being implemented by browser vendors, OpenSSL, and the OpenVPN team; and we advise users to update to the latest available versions."

So the attack requires that the attacker has the ability to monitor traffic passing between the end-user and a vulnerable website, "vulnerable" in this case only meaning that it is offering 3DES as an HTTPS TLS handshake available protocol; and that the attacker has managed to inject some exploit code, JavaScript exploit code, into a web page that the user is using because then that JavaScript is what burns up the wires generating the 785GB of traffic over a day and a half in order to, essentially, looking for this collision, this birthday attack collision. And that allows it, when it succeeds, allows it to decrypt the session cookie. So, yeah, 38 hours I had here in my notes that JavaScript spends generating 785GB worth of data to decrypt a cookie. OpenVPN required eight hours and, oh, a mere 705GB of data to recover a 16-byte authentication token.

So anyway, the good news is the industry's response to the news of this was immediate. OpenVPN just released an update which actively discourages the use of 64-bit ciphers. The v2.3.12, which was just released, they write in their release notes, this release includes many small improvements and fixes. This is the first release that actively discourages the use of 64-bit block ciphers for security reasons. And OpenSSL maintainers plan to disable 3DES in their next release, v1.1.0; but they are deprecating the claimed security level of 3DES, which had been high. They're now deprecating it to medium, which will bias the crypto selection logic against choosing it. So it will tend to be less readily chosen over ciphers that have larger block lengths.

So this is a classic example of - there was a theoretical understanding of this potential problem, yet nobody moved until some researchers went to the trouble of demonstrating the viability of an attack against small bit block ciphers. And then, immediately, whoops, everybody fixes it. The good news is we've got large block ciphers. AES is fabulous. And so it was just compatibility and inertia. You know, I mean, the fact that 1-2% of existing HTTPS connections are using 3DES means that somebody with presumably a really old browser - because any server will have more recent ciphers. I mean, I'm trying to imagine a scenario where, as we understand the way SSL connections or TLS connections are negotiated, the browser offers the list of ciphers and suites, cryptographic suites that it understands. From those, the server chooses, like, the best one, whatever "best" means. And so that way they agree on a strong one.

There are various downgrade attacks that we've talked about in the past where an attacker gets in and, for example, trims the list of the ciphers that the browser says it knows, and the server sort of shrugs and goes, okay, and then uses a weak one, which then allows the attacker to attack the connection. But still it's odd to me that 3DES, I mean, for example, GRC supports it, but it's like at the bottom of the list because it's old. But again, I wouldn't want to refuse service to somebody because the only way to get to GRC is over an SSL connection.

So again, this is the way it should work is not bad guys exploiting this, but researchers having a great deal of fun writing a paper, presenting a paper, getting an advanced degree probably by doing this work, and then improving the security for everybody as we march forward. And this is something that, you know, you couldn't have done this 10 years ago. We didn't have the infrastructure, and so 64-bit ciphers made sense then. They really no longer do. And so it's worth noting that that's no longer the case.

Now, YAWTTY, Y-A-W-T-T-Y, Yet Another Way To Track You. Believe it or not, the color and amount of ambient light striking your phone.

Leo: Well, that's probably unique to the location; right?

Steve: Yeah, that's exactly the problem. And the web guys, they're just having so much fun adding new APIs. So this is the Ambient Light Sensor extension of the Generic Sensor API. First of all, there is a Generic Sensor API. Okay. Before long it'll be like if your hand that you're holding the phone has a tremor, it'll, oh, look, it'll pick that up on the accelerometer and then track you or maybe send you to the - nah, I don't want to say anything that's ageist.

But anyway, so everyone already understands what this means; right? It is now possible for a web page to determine the exact color and amount of light measured in red, green, and blue lux levels which your phone is exposed to. So if that's code running in an ad, and you then go from one site to another, although it cannot be used to uniquely identify you, it can be used to disambiguate you. That is, if that - and as we know, the Panopticlick site worked by aggregating a whole bunch of individually weak signals into something that was shockingly unique. When you took all of the aggregation of weakness, you ended up with something that actually did identify an individual.

And so, yes, light is subject to change. You change your position. You put your phone down. Anything happens, you're going to get a different amount of light. But in the interval between switching pages, the light level probably doesn't change. And so that's one more thing, one more parameter that someone could use to note that, oh, you know, we weren't sure if this was the same guy. But look, exactly the same amount of red, green, and blue? Probably is. So YAWTTY.

Leo: Interesting.

Steve: Yet Another Way To Track You.

Leo: Yeah.

Steve: Wow. You know? And this is the problem is that I'm not sure how useful this is for a website to know how much light is striking your phone. Maybe, what, it enlarges the font size in dim light because it knows that our human eyes have to iris open wider, which lowers our resolution, so we need a larger font and dimmer light? I mean, you know, you can make up use cases for this. But it just seems like now they're, like, they run out of really important things to do. And so they're like, well, let's...

Leo: We do it because we can.

Steve: That's right. Let's just, boy, what haven't we put in there yet? Let's see, how about the dripping water API? Because we need a kitchen sink somehow, so it's got to have the dripping water API.

Leo: The iPad Pro has this, too, and actually modifies the screen temperature based on ambient light to balance it. And the rumor is strong that so will the new iPhone announced tomorrow. So I don't...

Steve: Boy. They're not going to get - I don't think they're going to get any money from me.

Leo: For ambient light sensing? No. For the headphone jack.

Steve: No. I mean for the new phone. I think we've really hit that point where the phone I have is absolutely fine.

Leo: Yeah, there's nothing wrong with this one.

Steve: No. Great phone.

Leo: And it has a headphone jack, which is pretty awesome, yeah.

Steve: Yeah, it does everything I want. And maybe in another few years, when the battery gets tired, if that's a problem. But I don't - I'm never, like, off of a plug for more than a few hours, the way I operate. So anyway.

Okay. So some German researchers did a social engineering experiment. 1,700 university students, who all claimed to be aware of the risks of unknown links, were actually tested - without their knowledge, of course, because otherwise it wouldn't be a valid social engineering test. Email and Facebook accounts were set up with the 10 most common names among that group of targets. So contemporary university student names, 10 of them.

It was funny because I was talking to my sister Nancy. We both watch "Stranger Things." And of course two main characters in "Stranger Things" were Steve and Nancy. And I thought, you know, I mean, it's interesting that names do have an era associated with them. Like now we've got Derek and Tiffany these days. But back in the '70s and '80s, young adults were named Steve and Nancy. Not so much anymore. Anyway - I don't know a lot of other Nancys.

Anyway, so Facebook profiles were created having varying levels of publicly accessible profile and timeline data, some with public photos and profiles, others with minimal. Then these email messages were sent claiming the links were to photos taken at a New Year's

Eve party held a week before the study, so high relevance.

Two sets of messages were sent out. In the first, the targets were addressed by their first name? in the second, they were not addressed by name, but with just more general information about the event which was allegedly photographed. The links that were sent resolved to a webpage with the message "access denied," but the site logged the clicks by each student. So they were unique links tagged in the email to identify who clicked the link, or who was the recipient of that particular email and link, thus who went there.

The messages that addressed the targets by name scored clicks from 56% of email targets and 37% of Facebook message recipients. The less well targeted messages that did not address their target victim by name yielded only 20% results for email, so down from 56 if it said who they were, but scored higher on Facebook. 42% clicked via Facebook messages, probably because email feels like and our experience is it's a more personal medium, where we expect to be, you know, we expect people who know us to send us email. And our spam, our own human spam filters are on the lookout for nonsense not addressed to us. So it's like, okay, no. So I'm not going to click on something that doesn't know who I am, at least; whereas Facebook message traffic is much less so, less personally tied like that, the idea being it's just not email.

So the German security researchers who conducted the study said: "The overall results surprised us, as 78% of participants stated in the questionnaire that they were aware of the risks of unknown links, and only 20% from the first study and 16% from the second study confessed that they had clicked on the link." So that was interesting, too. They caught them clicking on the link. They logged it. But then in this questionnaire that followed up, there was a high level of denial of having done so, even though the evidence was there in the server log. And finally, among those claiming that they were security-savvy, they found that 45 and 25%, respectively, had clicked on the links.

So this is a problem. We've talked about it a lot. We think this is now the way targeted attacks occur. There are certainly, we were discussing last week, technologically targeted attacks where something is, like for example in the case of Pegasus & Trident, where the user didn't even have to know that their phone was attacked. In this case, the guy was smart enough not to click on a link that looked suspicious because he was aware of the risks.

But there's also the social engineering side. And what we've seen, for example, we believe that the whole Sony disaster was precipitated from one administrative assistant who - it only took one to click on a link. And that allowed this persistent threat to get inside of Sony and then have its way with the entire company's network.

Leo: Kind of amazing. Half of everybody. Half.

Steve: Yeah, yeah.

Leo: It's amazing.

Steve: You know, I think it's like we were saying last week. People will give away their passwords. They just don't actually care that much about security. Everyone says, "Oh, yeah, I'm security-conscious. But, wow, I want to see any pictures that occurred. Who knows what happened?"

Leo: Yeah. Well, to be fair, I mean, these guys are pretty good at pulling your strings and pushing your buttons. They always put some text in there that makes it very hard not to click; right?

Steve: Well, and that's the problem. And of course we've talked about at length the "Hi, I'm away on a trip, and I've lost my wallet. Can you please wire me some money?" And that was, until that got very widespread coverage, it was phenomenally successful in situations where people actually were away on a trip.

Leo: Well, and - excuse me. Don't look to me, I'm having a coughing fit.

Steve: I'll stop clapping behind you.

Leo: Alert, alert the media. I've been phished. I've clicked on a link. And the link that got me, I mean, I've only done it maybe twice in the last five years. But the one that most recently got me, Henry had just lost his phone. He was in Barcelona. And he just lost his phone. And I got a link that said, "This is Apple. We found your iPhone. Click this link to go to iCloud and run 'Find Your iPhone.'" And I clicked the link because that's kind of credible; right? It tallied in with events that had actually happened.

Steve: Yup.

Leo: In hindsight, of course, Apple's never going to send you that text. But I didn't know that. And then I got a site that looked just like iCloud and started logging in. Fortunately, my fingers froze about halfway through that, and I immediately went and changed my password anyway. But that was an eye-opener. I mean, you're right. We know better. But they manage to find things that are going to push your buttons.

Steve: Yeah. I think it's a weakness that any sober person needs to acknowledge. And we also often have bad days.

Leo: Right.

Steve: I mean, you're just - you're distracted. You're not...

Leo: You're not always alert, yeah.

Steve: Exactly. You're not focused on something. You're trying to listen to something on the radio at the same time. So you just go ahead and click something, trying to multitask, and pow.

Leo: And I don't think it's a bad thing. I think humans are trusting. I think that's a good thing. I think we generally think the best.

Steve: Right. It is an abuse of an inherent trust in our fellow man.

Leo: Good thing, yeah. We were watching CNN today, a really dramatic documentary on an L.A.-based cult. And I was watching it with Lisa. And we're both thinking, how could these people fall for this? But it's because you want to, you know, you have to be pushed pretty far to go, "Oh, wow, I'm being suckered now, aren't I." We want to believe the best. I don't think that's a bad thing about humans. But it does make us vulnerable.

Steve: Or we want to believe what we want to believe. Sometimes it's not the best.

Leo: Yeah, that's true, too, yeah.

Steve: So there is once again another USB problem. You know, we've discussed these for years. Turns out that this is another plug-and-play mechanism. As we know, there are USB network adapters. It turns out, if you stick a USB network adapter into any standard default configured machine, PC or Mac, the OS will go, ooh, here's a NIC, and immediately put it into the system and query it for its information. So if instead this is a hacking device, that is, if there's a little computer behind that USB faade, it pretends to be, it can pretend to be a server offering DHCP - I'm sorry. Yeah, it is DHCP. But in the notes it says DCHP. So I think I just copied and pasted, so there was a typo in the source material. But anyway, provides DHCP services.

One of the things that DHCP can do, because remember that it's not just here's your IP address. We know that DHCP can also say here's the DNS servers you could use. And then there's a whole bunch of other stuff that you can do, time and, you know, just pretty much everything. Well, one of the things you can do is you can say "This connection uses proxy auto-config," which is it offers a URL to what's called a WPAD, W-P-A-D, which is the proxy auto-config discovery file, which tells browsers what proxy to use in order to get out onto the Internet.

Well, it turns out DHCP overrides DNS. That is, it is used preferentially when both are available, has a higher priority. So some guys at SpiderLabs have created a tool in Python they call Responder, which pretends to be a NetBIOS, DNS, SMB, MySQL, FTP, LDAP, HTTP rogue authentication server supporting all those protocols. And so you stick this thing into a machine. It works even if it's locked because, again, hey, it's a network adapter. We're all friends here.

So that network adapter gets bound into the system, and a query is sent out for the connection details. If it's this malicious server, it's then able to take over the connections and cause traffic to be proxied through it, all without touching the keyboard, and all while the machine is locked, not even actively in use. Because all of our computers today could have all kinds of traffic trickling in and out of them pretty much all the time. And so it's able to piggyback and leverage that traffic in order to capture credentials, which is what they end up doing, and then run that through any of the well-known LAN manager credential hacks in order to obtain the password. And then you've got the user's

credentials on the network.

So again, a classic instance of convenience trumping security. A very nice feature. You plug this in. But, boy, you know, you ought to have to be logged in. You ought to have to acknowledge with a pop-up that a new network interface is being added to the system. Please confirm that you want to do that. But that would be so much work. And maybe users aren't going to click the right button, so we'll just make it automatic.

Leo: Wow.

Steve: Wow. Yeah. So now some fun miscellaneous stuff. That catches us up on the news of the week. Spaces or tabs? That's the question.

Leo: What do you use? You're an assembly language programmer.

Steve: Oh, I'm strict tabs, of course.

Leo: Yeah. But nobody else reads your code. I mean, I think one of the reasons people use spaces is because tabs can be defined to be different on different platforms. So if other people were using your code...

Steve: Correct. And it is the case that, you know, my code, as I move it from one browser or from one editor to another, sometimes the tabs won't work right.

Leo: Are eight; sometimes they're four. Right.

Steve: Yeah. And some editors do tab expansion, converting them to spaces, but they don't deconvert them, and blah blah blah. I mean, so I wouldn't argue that there are some advantages to spaces. But I like tabs. And I'm not changing.

Leo: No.

Steve: At this point.

Leo: And I like spaces, and I'm not changing.

Steve: So what we know is that a Googler analyzed - now, the headline said a billion, but it's only 400 million, so come on.

Leo: Oh, please.

Steve: Analyzed 400 million files to settle the programming dispute made famous by HBO's "Silicon Valley" on that "tabs versus spaces" question. And in the show notes here, I have the chart which resulted, which was from the top 400,000 GitHub repos, showing Java, H, JavaScript, C, PHP, HTML, CS, JSON, C++ Python, XML, Ruby, CC, and Go. It turned out...

Leo: No Python, which is odd because whitespace is very important in Python.

Steve: Yeah. P-Y, Python is there.

Leo: Oh, it is there, okay.

Steve: Fifth from the bottom.

Leo: Missed it.

Steve: And much stronger in spaces than in tabs.

Leo: Yeah, since whitespace is everything in Python.

Steve: So what we learn is spaces dominated except for C and Go. So in C the blue bar showing tabs outsizes the red bar showing spaces. But not hugely. But still it's more. And in Go, look at that, it's like no spaces.

Leo: I don't understand why Go, yeah.

Steve: Nobody ever hit the spacebar in Go.

Leo: My editor does it for me. So that's the thing. I mean, if you're using a good editor, indentation is handled by the editor. It's only when you create a new block. And then I use tabs, but I think I have it set so the tabs - so I get the best of both worlds - so the tabs are spaces.

Steve: I just like tabs [crosstalk].

Leo: What I don't want, and I don't think you should use, is a tab character. I don't mind you hitting the tab key.

Steve: Oh, no, I don't want all that cruft in my...

Leo: So you use spaces, in effect, because you have the tab key mapped to four spaces.

Steve: No, no. I have an actual tab character.

Leo: Oh, you put a tab character in there.

Steve: I thought you meant showing the tab character.

Leo: Well, you wouldn't want to show it.

Steve: No.

Leo: But, yeah, see, I don't - I will use the tab key, but it inserts four spaces.

Steve: Ah, yeah. And I use the tab character.

Leo: That's the best of both worlds.

Steve: Right. Okay. So get this. We all know John Carmack.

Leo: Yes.

Steve: John Carmack has entered all of our lives, I would imagine, all of the listeners of this podcast at one point. I was fascinated by Doom back in the day. Back then it stood out as an astonishing tour de force in essentially 3D environmental rendering. I just used to stare at the walls as I would, like, move around and watch the texture mapping occur. It was just - it was phenomenal.

So John Carmack of course is still around. He's the CTO of Oculus. But Wikipedia says: "John D. Carmack is an American game programmer, aerospace and virtual reality engineer. He cofounded ID Software. Carmack was the lead programmer of the ID videogames Commander Keen, Wolfenstein 3D, Doom, Quake, Rage, and their sequels." And then of course a couple years ago he became CTO of Oculus. I was talking to Mark Thompson, just I think yesterday or maybe Sunday, and somehow Carmack came up in our conversation because John is just, like, crazy about frame rate. And Mark explained that he really didn't get the whole frame rate thing until he experienced low frame rate VR.

Leo: Oh, yeah.

Steve: And then immediately had to sit down because it's such a disorienting problem. Carmack was willing to trade off resolution for frame rate, saying that matters more than resolution. And I have to say, you know, anyway, the guy, I mean, he's my kind of programmer. All of that stuff was written in assembly language because you couldn't even get the machine off the ground with the kind of performance it needed to do Doom if it wasn't in assembler. So, I mean, he is a programmer's programmer.

The point of all this is that someone sent me a tweet, knowing that John had just tweeted, and knowing me, John just a couple days ago tweeted, "Create and run an empty activity project in Android studio, and I get a 38MB folder with 1,175 files." Then he finishes, "We've just given up on elegance."

Leo: Oh, typical old-timer, complaining.

Steve: Yay, John.

Leo: That's just libraries. It's loading - it's a framework. You're loading a framework.

Steve: I'm working on this...

Leo: By the way, if you're writing in Java, yes, you've given up.

Steve: I'm working on SQRL's installer and remover because people, you know, it doesn't need one. It just runs. But unfortunately, it's not for us. It's for our nontechnical friends. And the browsers download it to the download directory. Okay, then what? Anyway, so the problem is I don't know if anyone has seen any of these commercial installers, but they would take my cute little 238K piece of art and turn it into a 3MB blob that basically does nothing. So essentially I'm binding installation and removal technology into SQRL so that you just run it, and it will notice that it hasn't been "installed," and then say, "Hi. I'm not installed right now. Do you want to install me?" And then, if you say no, it'll just let you run it without making any modifications to your machine.

If you say yes, then it will copy itself under the program files directory and register itself to pick up URLs and so forth. And register with the add/remove programs so that, similarly, if someone wants to remove it, they can, blah blah blah. And it's taking up no space. So it's like, yeah, there is a right way to do it. Nobody does. But that doesn't mean the right way doesn't exist. I'm happy to do it that way. Actually, I have no choice.

Leo: There's an excellent book, if you're interested in the Carmack and Romero story and the story about Doom and...

Steve: Oh, neat.

Leo: And it's called "Masters of Doom: How Two Guys Created an Empire and

Transformed Pop Culture." Not a lot of coding in there. But the story behind it is fascinating.

Steve: No, that sounds really interesting.

Leo: Will Wheaton reads the audiobook, and right now it's on sale at Audible for four bucks, so it's a good little acquisition, "Masters of Doom."

Steve: Yeah. I like history, as we know. I've got PDP-8s blinking behind me.

Leo: Well, and Doom really was history, wasn't it? I mean...

Steve: Oh, Leo.

Leo: Yeah, it changed everything.

Steve: It stunned us.

Leo: It was the first premium product, too; right? First three levels were free, so it was a great way to get you into that. I mean, he revolutionized gaming.

Steve: Yeah. And, see, back then you had to be good to do it. He did it because he was, I mean, that was some serious code, very much like the early Mac programmers, who managed to get a 68000 to do things no one thought it was powerful enough to do. But they did.

Leo: Pretty cool.

Steve: I miss those days. Anyway...

Leo: Because back then they used tabs. No frameworks.

Steve: That's right.

Leo: Did it by hand.

Steve: There you had file size problems. Tabs were going to save you some space.

Leo: That's true.

Steve: That's right, baby.

Leo: That's true.

Steve: Andrew Hutcheson tweeted something. I don't know where he got it. Actually, he tweeted both you and me. And I just love this. Somebody, maybe he, originally wrote: "I'm not scared of a computer passing the Turing test. I'm terrified of one that intentionally fails it."

Leo: That's good. I like that. Yikes.

Steve: Love that. Thank you, Andrew.

Leo: Yikes, yikes, yikes.

Steve: Okay. And a couple little quickies. Adam Stearn, but may others have asked, he said: "What was that mail archiving application you mentioned weeks ago?" And I guess maybe I re-mentioned it weeks ago, but I initially mentioned it years ago. I just wanted to refresh everybody because I'm still using it, and it rocks. It's called MailStore Home. So MailStore is the company and the product, and I use the Home edition, which is free for personal use. I'm using v8, of course, but they're now at 9, which I noticed when I went to go check them out when someone asked me a few weeks ago.

Anyway, it's a great tool. I use it continually. Basically I have it set up so that mail that I receive is copied into an archive, and then it pulls from the archive and deletes the archive. So essentially I have - it's just the way it was convenient for me to set it up that way. So I have, it's now 2.7GB of indexed, instantly searchable, everything I've received from the beginning of time email. And it comes in handy all the time.

Leo: Does that require Outlook, or what is...

Steve: No, it doesn't. It supports about seven or eight different formats.

Leo: pgMail is supported.

Steve: It does support Outlook and PST files. It supports MBX files. It had no problem sucking in Eudora's - I had all of my stuff in Eudora, which is just a flat...

Leo: That's MBX; right?

Steve: A flat text file.

Leo: Nice.

Steve: Anyway, it really - it does the job. I'm so impressed. And again, free for personal use. Free for home users.

Leo: Nice.

Steve: Oh, look, it's at 9.8 now. I might want to wait till it goes to 10. Actually, my v8's working just fine. Anyway, I do recommend it.

Leo: That's good.

Steve: And someone said, given that Apple refuses to allow outside security audits, what's the basis for saying iOS is the most secure mobile platform? And so I thought about that, and I wrote back to this person. I said: "As we know, Apple exercises extreme control over their closed iDevice ecosystem. Google currently has much less control. This gives Android users more freedom, but at a cost in device security. Though Google is much better with their own devices, the vast majority of Android smartphones worldwide are either never patched or are patched partially and/or very late."

So we covered just last week Apple's immediate response to the zero-day that was found, and that within 10 days everybody had updates available, although I'm still puzzled by the fact that it seems to take Apple a while to get them pushed out. I just now, as we were doing the podcast, one of my iPads said, oh, there's an update. It's like, yeah. And so it's taking care of itself. But still, it's very quick.

And so, again, given that our model, the proper model is systems that are resistant to attack, clearly being able to respond to known attacks in a timely fashion is required because it's the known attacks that then get you. And that's a glaring weakness. We know Google is working because they're certainly concerned about security. They're working to fix that. But the way Android got started and the way it got picked up and sort of diversified into so many different products, many where it's just get the consumer's money and then good luck to you. Unfortunately, it's a computer with lots of known problems that aren't getting fixed.

And then, finally, Tim Stewart tweeted a question, something we had not discussed. He said: "Is loading a form over HTTP, but posting to HTTPS, a security risk?" And in the past, in like the olden days, back in the early 21st Century, I would have probably said yes, we've talked about this, that it's disconcerting to see a form delivered over HTTP. But technically the secure information, like if it's your credit card information and so forth that you're filling out, when you submit it, if it's an HTTPS GET or POST, then that's going to be protected. The server will be authenticated over TLS, and you'll get privacy from encryption. Everything should be good; right? Well, no.

The problem is, if you got the form over HTTP, anything could have been done to it. Any modifications could have been made. So, for example, a little bit of JavaScript added to it could, upon you clicking the Submit button, not only submit it to where you think, but

bundle up all those parameters in a query tail to a GET, which the browser is able to issue to a different domain and thus exfiltrate your form, your private confidential form data to some other domain without raising any alarms or concerns. So it is actually the case that you need the form delivered to you over HTTP, and it's submitted over, I'm sorry, over HTTPS - I was reading my notes where I said HTTP - over HTTPS and posted as HTTPS. Otherwise, there's just too much mischief that a bad guy can get up to.

And, finally, I didn't really have any - I didn't have time, actually, to go searching through my notes for a fun SpinRite anecdote. But something came up from my discussion last week that I didn't know. Mark Thompson didn't know it. Nobody I know knows this, or nobody I know who I would expect to has ever heard of this before. Which, I mean, it's right in the middle of my bailiwick. I was talking last week about how SpinRite is itself the DOS stub of a Windows app. And that way, when you run it under Windows, you get the little UI, the Windows UI that then allows you to install it, which mostly just means formatting a boot device for bootability. And then it copies itself to that. And when you run it, the same EXE, from the device, then it runs SpinRite, the DOS stub.

Someone tweeted me, @sjh_canada. I never knew what EXE and COM stood for. I just, you know, I sort of took them for granted. And I mentioned last week how a COM program is like, remember command.com or msdos, well, actually that was msdos.sys. But frequently the kernels, certainly in FreeDOS, it's kernel dot - maybe it is kernel.sys. But anyway, even though it has a SYS extension, it is actually a COM file. It's just a memory image, that is, and so it's limited to 64K, that is, one so-called segment of memory, because without anything fancy, that's all you're able to write in an 8086 class machine because the maximum size of addressing was 64K blocks.

Anyway, COM is short for "compile time binding," and EXE is short for "execution time binding." I never heard that before. Never knew it. And that exactly fits what's going on because by "binding" they mean essentially establishing the running image, in this instance of the use of binding. So when producing a COM file, the compiler binds the result into the memory image, and you just drop that image into memory, and you run it. Nothing has to be done. And the reason you used those back then is you didn't have to have a sophisticated loader. All of that work was done by the compiler, that is, the compile time binding. And so you just drop it into memory and just jump to it. The executable format is much more sophisticated and has to have a much more complex, thus bigger, loader.

So in a classic case of bootstrapping, where you're actually using the term "bootstrapping" as in one stage being used to get the next stage in, the COM loads easily with just a simple copy of the file into memory, and then runs. It contains the sophisticated loader that is then able to load the EXE, which is the execution time binding, where the binding into the RAM image occurs at execution time, as it's being loaded into memory, rather than by the compiler. So very cool. Never knew that. Thank you for sharing that.

Leo: I didn't either. I mean, I knew COM files were small, but I didn't know why.

Steve: I know. Nobody I know ever knew what EXE and COM stood for.

Leo: Yeah.

Steve: Turns out they do stand for something.

Leo: Yeah. Hey, a couple of little things. First of all, don't forget your mind teaser, your brain teaser from last week.

Steve: Nope. And that's at the end of our notes.

Leo: Okay. Don't want you to forget that.

Steve: Yup.

Leo: And somebody's asking in the chatroom, and I think this is very cool, Jonathan Zdziarski, who is, of course, we talk about him all the time, the Mac security guy, has written a little utility that he's giving away called Little Flocker. It stands for F-Lock. It's the old F-Lock. And it's a program for Macs that restricts file access. You have to give permission for an application to access your files. And, now, that could be annoying. It's kind of like Little Snitch, which is blocking outbound network connections, which is another program, very useful in preventing malware, also very annoying.

Steve: Yeah. All of these things are [crosstalk].

Leo: Pretty convenient.

Steve: Ask for permission, it's like, well, yeah.

Leo: But I think it's kind of intriguing. It's a kernel extension, which is always a nerve-wracking thing because that modifies the operating system at a very low level. But Zdziarski, on the other hand, seems to be a very benign fellow. Anyway...

Steve: And he's been doing a lot of pen testing. I've been following his tweets as he's been developing this, and it's been moving along now for a few weeks.

Leo: So at some point, just curious. I know you don't use Macs particularly. But this seems like a good idea. I'll run it and let you know how horrible it is, yeah.

Steve: Do. Because we know, if you can tolerate it, Leo...

Leo: Well, Little Snitch is one of those things where you make rule sets a you go. So every time...

Steve: Yeah. But so was NoScript, and NoScript drove you out of your mind.

Leo: I couldn't do it, yeah. So this will be like that. Every time you want to save a file, the first time you do it with a program it's going to say, now, "Microsoft Word wants to save a file. Is that okay?" And you'll say yes, from now on you let Word do whatever it wants or whatever. But security and convenience.

Steve: And the fact is, that's a perfect example of the firewall rules being set correctly.

Leo: Right.

Steve: That it's block by default, and then allow, you know, permit selectively.

Leo: Right. Not always fun. But really safe.

Steve: No, I would say never fun.

Leo: Never fun. But always safe.

Steve: It's a problem, but we don't have any better technology today.

Leo: Yeah. Time for Flip Feng Shui, Steve.

Steve: Which is their name for this. I take no credit for it. It's a great name. Okay. So a year and a half ago it came to light that DRAM, which is where the bulk of the data in use by the computer is of course resident in DRAM, was subject to deliberate manipulation, a so-called "bit-flipping attack." And what was discovered was that pounding, that is to say hammering, on one row of DRAM was shown to have the ability to influence one or more bits in the adjacent row. And the stat I remember, I don't have it in my notes, but I remember it from bringing myself back up to speed, is one out of 1,700 bits is flippable. So one in 1,700, that's a lot of bits when you consider how many bits there are. So a relatively good chance of Rowhammer attack working.

And then the double Rowhammer is hammering the rows on both sides of the target row. And essentially what's happening is we're inducing noise. The DRAM is another instance, just like hard drives and SD, where the commercial pressures to increase density have forced the engineers, who are super clever, to cut the margins, cut the signal-to-noise ratio down so that it works, but you just don't have as much margin as you'd like to. And that's the whole, you know, that's one of the keys of SpinRite is that things have always been that way. The technology keeps advancing, so the amount of data we can squeeze is always more. But it's still always pushing the limit. And so that creates a gray zone. And as I mentioned in the past, SpinRite goes into the gray area and pulls your data out of it.

So, okay. So we had a year and a half ago a problem, a recognized problem. This was

bad, that something could hammer on DRAM and cause an error. Well, okay. That's not good. But at this point it sort of - it shows the computer is not working the way it should, but it's not clear how you exploit it. Well, in the last year and a half, the last 18 months, there have been some special case exploits, where for example there was a privilege escalation exploit that had some probability of flipping a bit that was critical to maintaining and managing the privilege of a process, and that it could change its privilege. Basically you're doing something out of band. If you tried to write what you wanted to that memory, the system would slap you and say, that's a memory violation. You have no access there. But if you pound right next to it, and the bit that you wanted to flip flips anyway, then, hey, you didn't do it. Actually you did, but you did it in a way that the system didn't see.

So that's Rowhammer. And since then there has been effort to tighten up the hardware. One of the things that could be done is that the RAM can be refreshed more often. Typical RAM refresh is about every 64 milliseconds. And so what's happening is, as we know, DRAM uses the fewest parts possible per bit in order to get as many bits as possible per unit area. So the capacitor where the charge is stored is made as small as possible, and that's a problem because, well, the good news is it lowers the power consumption because you're having to force less, you're having to fill less in order to get a voltage change on a smaller capacitor. But it also means that it's the signal in the capacitor is increasingly small relative to the noise in the environment. And Rowhammering is about environmental noise.

So if you revisit the RAM more often, the whole refreshing process is one of reading a row of RAM before any of the bits have had a chance to change. They want to because the electrons are leaking away. And so, if you come back within every 64 thousandths of a second, read what's there, and then, like, reinforce it, just write back what you just read, then that's what DRAM refreshing is. So by doing it twice as often, by shortening the refresh interval to 32 milliseconds, that's been shown to provide very, you know, much stronger protection from Rowhammer.

And DDR3 memory was the first place this became prevalent because of the density of the cells. It was just so aggressive. DDR4 is even higher density. It would tend to be more susceptible, except it was developed with a Rowhammer awareness. And there's actually on the DRAM controller is something counting the row accesses so that it's able to catch, it may not be a deliberate Rowhammer attack. It could just be - it could be serendipitous, although the reason that would be suspicious is that the other thing Rowhammer has to do in order to work is to flush the cache because remember, as we talked about last week, our systems have multiple layers of caching, typically three - Layer 1, Layer 2, and Layer 3 - before you get to the DRAM. So the processor has to read the data that occupies one of the rows and then explicitly flush the cache. Otherwise, if it tries to read it again, the cache will do its job and provide the data.

So Rowhammer functions by - I'm sorry. So DDR4 memory has been strengthened by having a controller that looks for high-frequency usage of specific rows and then goes right to them, and their neighbors, and refreshes the memory on a spot basis, in addition to the ongoing background refresh. And that looks like it's gotten better. The problem is there's still lots of DDR3 memory, and it's only recently that chipsets have increased their refresh rate. There are apparently some microcode updates available for motherboards to increase the refresh frequency and decrease the interval from 64 to 32 milliseconds. And there's about a 1-2% overhead that results from doing that.

Okay. So we have a technology, a potential exploit technology, which unfortunately today works. That is, one in 1,700 bits of DRAM on servers in the world can be flipped. And these are called "memory disturbance errors," technically. Okay. Now we've got to

step back and look at how memory is managed. And I was trying to think of an analogy for memory management that would make sense. So imagine a deck of cards. Everyone knows about a card deck that has 52 cards. Imagine that being well shuffled, so it's completely randomized, and those 52 cards are dealt out to four players around the table so that each player gets 13 cards, facedown. Now, imagine that that memory, that is, the original deck of cards is all of the memory in the system.

Now, what the four players have been dealt is essentially their view of memory. So each player has 13 cards. Say that each card represents a meg of memory. So they each get 13 megs of memory. Each player sees the memory as one through 13 megs. But the fact is those have been randomly assigned out of the pool, out of the actual physical pool. That's hardware mapping. That's hardware memory management. The idea being that what the process sees is an abstraction. It sees a linear range of memory like those one through 13 cards, but the actual physical location could be anywhere in the system. But one of the keys is, notice, no player has the card of somebody else. That is, the way they were dealt out, everyone has disjoint cards, that is, their own separate 13. No one is sharing cards. That way they all have their own memory. And I'll explain, and you'll see in a second how this becomes relevant to this interesting hack.

But the virtual memory manager and/or the operating system manages the pool of physical memory using memory mapping hardware, which imposes itself, interposes itself, between the program running and the hardware in the machine. And so that when - so the program sees a region of memory which is maybe not the actual physical memory. Almost never is the actual physical memory location. That's abstracted for it. Which allows each program to sort of operate independently and with no collision.

Okay. Now. People running cloud servers, they noticed something. They noticed that, in a big cloud server, there might be a hundred virtual machines operating. And they're probably running the same version of something, some OS, whether it's Windows or Linux or a Unix. And if that's the case, they share a lot of memory. That is to say, the same operating system in two separate virtual machines is going to have the same data stored in a lot of its pages. And so there's something known as Kernel Same-Page Merging, KSM, Kernel Same-Page Merging. To optimize a large system's memory, the virtual machine manager is continuously scanning memory, searching for identical pages. And it turns out that is such an effective strategy for conserving memory that it's now the way it's done.

So think about it. Separate virtual machines loaded with the same stuff, whatever it is, maybe the same version of OpenSSL, or the same OS, they're going to have a lot of code similar. And these things always load starting at a page boundary. So the memory page alignment will be the same. There will be many identical pages. So when the virtual machine manager notices two pages that are the same, which are typically occurring in different virtual machines, it merges them. What it does is it marks them as read-only so that any attempt to change them will be brought to its attention. And it then points one of the virtual machines to the other virtual machine's memory. That is, it breaks that one-to-one relationship between apparent memory and physical memory, creating a many-to-one relationship. So that suddenly many of these virtual machines will have a mapping table that maps all of what looks like private and separate memory to a single page of memory.

So you can kind of see where this is going. This starts to get a little dicey here because now we've got, I mean, technically it's brilliant, I mean, and it's gutsy. And it works. And it's what everyone is doing. If any machine attempts to change a byte of memory which it is sharing with others without its knowledge, the virtual machine manager steps in, copies that to its own page, makes the change, and gives that process back its own

private copy because it's no longer identical to the others. So it works. It hugely reduces the memory footprint and allows the same hardware to simultaneously hold vastly more virtual memory or virtual machines because there is this high incidence of memory collision among individual virtual machines.

Now, what these guys figured out how to do is how to weaponize this. And they've written the code. They're presenting the paper. Did they present it? Or are they? I had it in my notes here. I don't see it. Oh, yeah. At the USENIX security symposium a few weeks ago they presented this. So, first of all, they scanned their own memory, they Rowhammered themselves, looking for an identifying flip, all of the flippable bits in their own memory area. And they need a bit at a certain range of offsets within a block because of the strategy, which I'll explain next.

So they first audit - they do a self-Rowhammer audit to find vulnerable bits in their own space, which they're able to do. The operating system says, "What the hell's going on here?" But it just lets them. So then, once they have found a bit with the proper offset in the page, they know that one of the other processes in the system is running OpenSSH. And the way OpenSSH is configured, there'll be an authorized keys file which is the public key that matches the remote user's private key. So the public key is in the server, as we would want. The user maintains their private key. And when you connect, when you attempt to connect with OpenSSH into the server, the server's authorized key is used to verify the incoming connection's private key. So that all makes sense.

But because this is open source software, because it's known where it's going to be, they're able to - oh, and the key is public. What they're able to do is duplicate a page which they know resides in a different machine, and where the offset of the bit in the page is in the private key where it's being stored in RAM in OpenSSH. As soon as they explicitly deliberately duplicate a page that exists in the other virtual machine, they wait a bit. The virtual machine manager scans around, sees, oh, look, here's a page that's the same as it is over there. I'm going to merge those. Now they merge them.

Okay. Once that's happened, these guys have already verified that they can flip the bit where they want to. So they hammer on their own copy, which is actually now a shared copy. They reflip the bit that they verified they could flip, which flips the bit in the RAM instance of the other virtual machine. Now, this also factors in, if you'll pardon the pun, to the whole issue of primes. We were discussing last week that what it is that is hard about prime factorization is the factoring of two really large primes. The public key has this merge, you know, is the multiplied primes, is in the public key. It turns out, if you flip one bit of a public key, you make it incredibly easier to factor because it's no longer the multiplication of primes.

Leo: Ah.

Steve: It's just like we were saying last week. So that flipped bit in the public key then allows them to perform a factorization. That allows them to obtain the private key, and they can then remotely log in over SSH into the machine that they have compromised. Unbelievable.

Leo: Wow.

Steve: And it works. It works. This is like, oh, wow.

Leo: That's amazing.

Steve: Isn't that incredible? Yes.

Leo: So the puzzler from last week helps us understand that, yeah.

Steve: Yeah, yeah. And how you can take something which is, oh, look, we found a problem with DRAM. If you pound on this row, something flips next door. It's like, oh, that's bad because then the program will crash. Yes. But also, if you're really, really good, you can turn that into an exploit.

Leo: Mm-hmm. Crashes are good. Yup.

Steve: I mean, yes, it's just a perfect example of the way virtually any kind of defect can be leveraged.

Leo: Nice. Very nice.

Steve: So, wow. Sublime and righteous hack.

Leo: And well named, I might add.

Steve: Yes. The Flip Feng Shui.

Leo: Feng shui.

Steve: Okay. And we'll wrap up with a discussion of last week's puzzler. Everyone will remember that someone tweeted me with a problem. And in fact I haven't had a chance yet to get back to him my probably favored solution to it, which is really pretty cool, and something we even talked about some time ago. And that is, imagine you have an appliance whose firmware is old and only knows about SHA-1. Very much like XP before SP3. XP up through SP2 didn't know about SHA-256 at all. So it couldn't verify that cert or any cert in the chain or - and, see, the roots are still signed with SHA-1 because they don't have the problem of, well, because they're self-signed, essentially. So I just lost my train of thought.

Okay. So we have this device whose firmware could absolutely not be changed. The problem is, as we all know, SHA-1 certs, unless you go to WoSign and change the POST parameter to a "1," in which case, although maybe they fixed it by now, you could get yourself a cert good for a couple years. The problem is any reputable CA will not, after January 1st of 2016, on and after, will not give you an SHA-1 cert. So one of the hacks, the first one was one you did mention last week, Leo, that I think you probably saw Bill in Michigan note, or maybe you guys came up with it independently.

Leo: No, that just was off the top of my head. In fact, I'm shocked that it was a good idea.

Steve: Very good. And that is to use the clock. The only reason that appliance would balk at the SHA-1 certificate, at an old SHA-1 certificate, is if it thought it was expired. So if it's using it own internal clock, set it back a year. Now, remember, certificates do have a not-valid-before date, so you can't go back a decade and get 10 years of life. You have to go back a year. And basically, for the rest of time, or until these devices die or get retired or finally get updated, you'll have to annually move it back to the same year to keep it within that period of time. So the remote server would be deliberately issuing an expired certificate. Servers don't care. We see it happen all the time. Sites are being protected with expired certificates. Our browsers are the ones that say, hey you're trying to pawn off a bad cert on me. And alarms go off, or if you're Google your system melts down.

But anyway, in this case the first possibility is simply not let the device know what day it is. Keep it within the valid date. Servers don't care if they're issuing an invalid cert. You'd have to arrange not to have anybody else obtain that cert. So hopefully the fully qualified domain name of the device, or maybe the port it's connecting to, hopefully there's something that distinguishes incoming connections from it that would allow the server to have on that domain and/or that port, have that bound to this increasingly creaky and old, but still useful, SHA-1 cert. And then you would get it, you would arrange to make it okay by keeping the device's clock there.

Now, if it's not an internal clock, it might be that the device is a little bit smarter, and it's making NTP, Network Time Protocol queries to the National Bureau of Standards or the Navy or wherever. There's a whole bunch of different NTP servers around. In fact, Microsoft has time.windows.com. So if that's the case, you would want to, first of all, learn about what the device is doing to obtain time by capturing its traffic. NTP is not a secured, encrypted protocol, so you can see - and it's typically UDP. So you can see those easily go out and come back. And you'll see it doing a domain name lookup at wherever it's using for NTP. So then you arrange for its DNS server, whatever it is that's serving DNS, to serve it a spoofed IP. That is, you change the IP of the fully qualified domain name it's looking up to your own NTP server, where you keep that date within the sweet range, with typically two or three years for an SHA-1 cert, so that the devices are all happy.

So once again, it's a network time protocol-based approach of fooling the appliance into what day it is. And, if for some reason, neither of those two approaches work, there is another one which is very cool. And that is leveraging CloudFlare's so-called "No Browser Left Behind" solution. It is the Picture of the Week that I deliberately didn't point out at the beginning of the podcast because I didn't want to give this away. But that picture on the first page of the show notes shows the - it's a flowchart of CloudFlare's logic for their dynamic certificate issuance. And we've talked about how they work before, and we did talk about no browser left behind quite a while ago.

Their concern was that, okay, yeah, SHA-1 is something that doesn't provide the security we want, and it's not necessarily future-proof. But there are devices, there are browsers that are not able to do SHA-256. That is to say, we don't want to leave any browsers behind. Anybody on XP with SP2 - and in fact CloudFlare has stats on the number of queries they service that are happening right now from systems that cannot do SHA-256.

So what they do is, and this is the brilliance of their solution, because you don't have to

statically issue the same cert to everyone. The TLS handshake gives you clues about how advanced the TLS cryptographic stack is in the client that is requesting the connection. And if it's TLS 1.2, then you're able to check whether it understands elliptic curve or not. If so, you give it that. If it's TLS 1.1, then it probably knows SHA-256. You give it that. If it's TLS 1.0, which was SSL 3.0, which, for example, these appliances certainly would be, you give it - and CloudFlare does - an SHA-1 cert. So all you would have to do would be to have CloudFlare handle that domain which these appliances are connecting to, CloudFlare essentially being the TLS or HTTP or connection proxy for those devices. And they would keep working as long as CloudFlare continues to offer the "No Browser Left Behind" service.

Leo: Clever.

Steve: Very, very cool solution.

Leo: That's a good way to do it. And we've mentioned CloudFlare before. And the reason you don't use it to protect yourself against DDoS attacks is this cert replacement thing.

Steve: Yeah, because, well, yes, correct. I don't like the idea of anybody else having GRC.com certificates.

Leo: Right, right.

Steve: That's just like - that goes against my...

Leo: Not worth it.

Steve: ...grain, you know, [crosstalk].

Leo: But there are a lot of people wouldn't care about that.

Steve: Correct. Oh, yeah, yeah. I mean, and a lot of major people use them. But they're not GRC. And but the other problem is that all of the extra goodies that GRC offers, I didn't realize how many people were using the DNS spoofability test until it was down because I broke it. And it's fixed since then. But I got all these complaints from people. It's like, god, I really want to use that. And, like, so all - and, like, ShieldsUP, spoofability, there's a whole bunch of the fancy network stuff that I've done that cannot be behind a proxy. It just won't work behind a proxy. It has to be able to accept unsolicited incoming packets. And if it can do that, then it is open to attack.

Leo: Steve Gibson, he's at GRC.com. Is it up?

Steve: Yeah. We're back.

Leo: All you've got to do is visit GRC.com and revel, revel in the freebies, all the great stuff that he does for us, like SQRL, ShieldsUP, the DNS tool, which is actually...

Steve: I have several big announcements coming, too.

Leo: And good stuff coming. I use the DNS Benchmark all the time. That's really useful. Download that. And then there's the one thing, the one and only thing he asks you to pay for. It's his bread and butter. It's his daily living. It is, of course, SpinRite, the world's best hard drive maintenance and recovery utility, and well worth every penny. And if you go to GRC.com, make sure you pick up a copy. You can also pick up a copy of this show. And not only the audio, the MP3, but a written transcript of it, as well, GRC.com. You can leave your questions there, GRC.com/feedback. Or tweet him. He's @SGgrc and accepts direct messages as well as tweets. So it's a great place to go to ask a question. And maybe we'll do a question-and-answer version next week.

Steve: I hope so. Although this was fun.

Leo: Really great. Really. Always a pleasure. You're fabulous. We love doing this show. And I know there are almost 100,000 people every week who devotedly listen. So many people tell me either that they use it in curricula at colleges for computer science and security, or that they use it to get jobs or get A's. One guy told me, because he'd listened growing up his whole life to Security Now!, he was able just to take and pass a number of computer science classes without even going to the class because he already had taken those classes right here. So thank you for the - you really do something pretty amazing here. And we'll do it again next Tuesday, about 1:30 Pacific, that's 4:30 Eastern time, 20:30 UTC. I will be here one last time next week.

Steve: And I was feeling so glad, Leo, that the podcast is not on Monday.

Leo: Why is that?

Steve: I'm glad it's on Tuesday.

Leo: Oh.

Steve: Well, because Monday is too often a three-day weekend.

Leo: Well, that's true. That's true.

Steve: And I don't want those days off.

Leo: You'd miss a show. And you never miss a show.

Steve: Oh, that would not be good.

Leo: Now in its 11th year of serving the masses fabulous information about security.

Steve: Twelfth year, actually.

Leo: Twelfth. Finished its 11th. Now in its 12th.

Steve: Yes. In year 12.

Leo: Thank you, Steve. We'll see you next time.

Steve: Okay, my friend. Talk to you next week.