Two-Armed Bandits

Description: This week, Leo and I discuss printers around the world getting hacked! Vizio's TVs really were watching their watchers, Windows has a new zero-day problem, Android has an easy-to-hack pattern lock, and an arsonist's pacemaker rats him out. A survey finds that many iOS apps are not checking TLS certificates. Courts create continuing confusion over email search warrants.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-598.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-598-lq.mp3

In a blast from the past, SQL Slammer appears to return. Cellebrite's stolen cell phone cracking data begins to surface. We discuss some worrisome events in the Encrypted Web Extensions debate and that non-Windows 10 users are not alone. We answer a couple of questions, report on a terrific sci-fi series, offer a bit of other miscellany, and end with a fun story about one-armed bandits being hacked by two-armed bandits.

SHOW TEASE: It's time for Security Now!. Steve Gibson's here. Boy, does he have a lot to talk about. A lot of tech news, including that hack on the one-armed bandits, on the slot machines. What's going on with the iPhone apps? Thousands of them, apparently, or at least hundreds that are hackable. Steve will give you the story and the mitigation and of course all the security news. Coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 598, recorded Tuesday, February 7th, 2017: Two-Armed Bandits.

It's time for Security Now!, the show where we cover the latest security and privacy news with this guy right here, Steven Gibson of the Gibson Research Corporation. He's armed and ready with his super triple quadruple venti latte.

Steve Gibson: You betcha. Hey, Leo. Great to be with you again.

Leo: Good to see you.

Steve: Episode where we're closing in on number 600. This one is 598. And I named this one "Two-Armed Bandits." And in fact we're kicking off here on time, and I just got the -

I forgot even to set the intensity and camera settings. So people are going to be annoyed that I didn't tweet the show note link beforehand. If anyone is wondering where it is, it's the same format as always, but it's 598 rather than last week's 597. So I'm sorry for not putting it into my Twitter stream. I'll do that in two hours, once we're done. But anyway, there was a really fun story that was perfect for the podcast about a problem with one-armed bandits, which of course is the fun term for slot machines, not being random enough.

Leo: Uh-oh.

Steve: And the consequences of that, which is how we'll wrap this up. But we had a lot of news, as we have been for the last few weeks. Speaking of the devil, because of course last week was "Traitors in Our Midst," 150,000 printers around the world got hacked.

Leo: Oh, that wasn't what precipitated the show? That was after the show?

Steve: Yes.

Leo: Oh. Wow, you were prescient.

Steve: So a little bit of prediction there. We also have Vizio's TVs that have been - the corporation has been slapped by the Federal Trade Commission when it was discovered what they were doing, that they really were watching their watchers. Microsoft has an unpatched zero-day problem as a consequence of not responding to the security researcher who found it within 90 days. Hopefully - we're going to have to wait a week because this is one of those months where the second Tuesday, Patch Tuesday, is as late as it could possibly be because they missed the first Tuesday by one day. Last one, of course, was January 31st was our last Tuesday. So this is the first Tuesday of the month, meaning that maybe they're going to get this fixed.

And then I just loved this story. Some researchers examined camera footage of people using the Android pattern lock and realized that's really not hard to guess what their pattern is. So they demonstrated that in a research paper that we'll discuss. An arsonist's pacemaker ratted him out. We have a survey finding that many iOS apps are still not checking TLS certificates and the consequences of that. The courts have just made a ruling on Friday which has created new confusion in the email search warrant mess. I imagine you and Jeff may want to talk about this because it involves Google. We'll talk about that ourselves here in a second. We have a bizarre blast from the past that actually predates the podcast. There were actually security events, Leo, before the Security Now! podcast.

Leo: What? No.

Steve: Even though this seems like it's been going forever. Yes, this was the SQL Slammer worm has returned.

Leo: I remember Slammer. I remember Slammer, yeah.

Steve: Yeah, yeah. Cellebrite's stolen data has started to surface. We talked a couple weeks ago about how 900GB had been exfiltrated from them. Now we have some sense for what it is, and it's not actually very impressive. There's some worrisome movement in the encrypted web extensions debate, another topic that you and Jeff and company may want to discuss. I think it was Business Insider, no, no, shoot, Financial Times? I can't remember who. Anyway, we'll get to it when we get to it, about just reminding us that non-Windows 10 users are still not alone.

I'm going to answer a couple quick questions. I have a report and recommendation without reservation of a new sci-fi author and series that I finished. I was up till 2:30 a.m. and then 3:30 a.m. on the final two nights last week, Wednesday and Thursday. I just could not make myself stop reading it, and I ended up finishing the fourth book in the series, and it was really fun.

Leo: Wow.

Steve: A bit of miscellany, and then we'll talk about two-armed bandits ripping off one-armed bandits. So the Picture of the Week I got a kick out of. This was actually - this was taken off-angle from the large presentation screen in a security presentation. And just for the heck of it, I removed the perspective distortion, but it still has some parallelogram distortion that I didn't get rid of. It's like, okay, well, the message is coming through anyway. Anyway, so I got a kick out of it because this was in the context of the problem that we continue to have, I mean, arguably will always have, with well-meaning notifications that users are supposed to pay attention to because this is important.

I mean, of course, the classic example was remember how my own operations manager and bookkeeper and accountant and Girl Friday who's, like, made my life possible for the last - she's been with me now for 23 years - how I set her up with a mirrored RAID disk array, and it started complaining that one of the drives was dead, and press Enter to continue. And so she did, for about another couple years, until the second one finally died. And then when I said, "Well, Sue, the whole point of having two drives is that if one dies, then you can limp along on the other one."

Leo: But fix it.

Steve: "But you should, like, let me know." And as it happens, that turned out to be a SpinRite recovery story I was able to share because, even though both drives had completely gone belly-up, I ran SpinRite on the most recently dead one because of course it had the most recent information, and SpinRite was able to recover it, and we lost nothing. But I said to Sue, "Okay, listen. Anytime something like this happens again, the point is there's an early warning system here. So don't just say, oh, look, I can get my work done if I just say okay."

So this dialogue, in the title it says: "Website Certified by an Unknown Authority." Okay, whoops, wait a minute. That would be an indication, our listeners know, of a man-in-the-middle attack. Something has intercepted a connection such that the identity of the other end was not verifiable. So of course what the user sees is the dialogue. It just reads - it's

got the yellow triangle warning - "Something happened and you need to click OK to get on with things." Which, you know, that's how everyone just sort of says, okay, how do I just get rid of this annoyance? And in the details it says: "Certificate mismatch" - oops, I just hit the spacebar and paged down by mistake. "Certificate mismatch security identification administration communication intercept liliputian snotweasel foxtrot omegaforce."

Leo: So this is a real error message?

Steve: No, no, no. They're just sort of saying nobody reads this crap.

Leo: Yeah, yeah, yeah. But, I mean, it was - some program does this.

Steve: Oh, I think somebody made it up for the presentation.

Leo: Oh, okay, yeah. My favorite part is the buttons.

Steve: Exactly. The button, and we've all seen this, this button is labeled "Technical Crap," dot dot dot.

Leo: So if you want that, you can get it, yeah.

Steve: So if you haven't had enough yet because "snotweasel foxtrot omegaforce" wasn't sufficiently confusing. And then there's three radio buttons: More technical crap, Hoyvin - what is this?

Leo: Hoyvin-Glayvin. It's Jerry Lewis. Hoyvin-Glayvin.

Steve: And then the third one is "Launch photon torpedoes." Meaning nobody reads this.

Leo: Hoyvin-Glayvin.

Steve: All they get is, wait a minute, I want - where's my document? What do I - you know, okay, fine. Do I push OK or Cancel? Just get this out of my way. And so we have - this is a sort of a fun reminder that we have a fundamental problem with the breach, the gap that exists between our systems, which are designed by people who know what is important. And basically we've made the dialogues look pretty because users want 3D and shadowing and so forth. But we still haven't managed to solve the underlying problem, which is essentially users don't care, and they're just going to click through.

And so ultimately this stuff has to be designed, I mean, if you literally take the word or the expression "foolproof," break it down into two words. Unfortunately, that's what we need is somehow to make this stuff just foolproof, that is, don't try to tell them. Don't

teach them. That was always my tech-y friend Bob's biggest complaint. Or actually it was the complaint of the people he was working with because he was always trying to teach them what this stuff means. And they're like, no, just what button do I push? Just I don't want to know this.

Leo: Don't explain it, just tell me what to do.

Steve: I don't want to go to college or school of Bob. I just want to know how to make the lights turn green. And he'd just shake his head and say, well, then, you're on your own.

Leo: I give up.

Steve: So last week the topic of the podcast was "Traitors in Our Midst." And independent of that, and it had to have been going on already, it couldn't have been inspired by the podcast because I looked at some of the work this person did, and it's some nice scripting. A security researcher who calls himself "stackoverflowin," with no "G," stackoverflowin, wrote a script to scan the Internet for printers publicly exposing their IPP, which we discussed that week, Internet Printing Protocol; LPD, the Line Printer Daemon port; or port 9100. Just scan the 'Net. Now, there was, again, misreporting as a consequence of people taking literally sort of his joke, like what he had printers print. But in the last week I've received a bunch of tweets from people. There's also a fun one, Leo. There's a link - I didn't turn it into a link. It's a Twitter picture below that first picture from Priscilla. She tweeted: "The printer for our POS" - and I'm assuming that in this case she means point of sale.

Leo: Yes, not piece of...

Steve: Yes, "systems at work got hacked." And she says, "LMAO," and then she posted a tweet which actually shows a strip of point-of-sale paper.

Leo: Yes, carbon paper, too. I mean, it's, yeah.

Steve: Yeah, with this thing. And so it says: "Read. Stackoverflowin the hacker god has returned. Your printer is part of a flaming botnet operating on Putin's forehead utilizing BTI's (break the Internet) complex infrastructure." Okay, none of that is true, but he was just having fun. And so one of them shows, I don't know, a terminal and then a computer. And the screen says "Hacked, hacked, LOL, just kidding." And that was one of the messages that he had 150,000 printers around the globe print. And then on the next page of the show notes is a different message, where he did an ASCII art bot and basically said the same sort of thing.

So, okay. So first of all, there was no botnet. Nothing was technically pwned or owned. But what was surprising, although it shouldn't have been from our reporting last week, is that 150,000 printers are publicly exposed, including this POS system. That's like, why? Why would you have your port exposed? And I thought about it, and I thought, okay, there's a takeaway for our listeners because our advice, Leo, yours and mine, for more

than a decade, we can say that now because we're in year 12 of the podcast: Disable Universal Plug and Play. That has been our mantra because that Universal Plug and Play is now a widely used interface on the LAN side of routers which is enabled by default because they want to be helpful, and they don't want tech support calls, and everybody else does it, so why shouldn't they? It allows any device on your LAN to make itself public. That's arguably what it was for.

Now, Microsoft gets away with this, this was their standard, because they've also got a firewall in the PC. So they're able to be a little more careful in using Universal Plug and Play. But so people understand, you can log into your router, and you can deliberately map ports inside. That is, you're able to essentially bypass the router's Network Address Translation stateful firewall. Meaning, and we've talked about this often, that unsolicited packets are ignored. It's only packets returning from an initial outbound connection which are automatically allowed back in and are then routed to the original sending machine.

But there are some things like Skype, for example, back in the peer-to-peer days of Skype, and gaming, where in order to be on a gaming network, they're greedy with the ports, saying there's like four or five ports you need to have opened. And so because that was too hard, I mean, and arguably, I'm not saying it isn't, it's a problem where convenience and security are colliding because on one hand your Xbox wants to have some ports exposed that we're trusting it to handle responsibly. So it uses Universal Plug and Play to open those ports behind our back. That is, the router has it. It's enabled by default. Any device, there's no security. That was the controversial feature of Universal Plug and Play because, again, they opted for ease. There is no security on it.

So a device on your network needs no permission to talk to your router and say - and in this case we're talking about a printer. This has to be how 150,000 ports are open. Printers are trying to be helpful. They're saying, oh, what if you're roaming, and you want to print to me from out on the public Internet? Well, if you're going to do that, I have to make a port open. So when you connect your printer in your LAN, it reaches out to the router and opens ports behind your back, exposing it to the Internet, which is where 150,000 of these ports came from, most of them probably unintended. So here's, again, so we've had an example of the number of printers that are probably exposed and the mechanism by which printers are able to make themselves public without their owners knowing it.

One thing you can do, because we know what these ports are, the port for the Internet Printing Protocol, IPP, is TCP/631. For LPD it is TCP/515. And for the RAW protocol it's port 9100. Now, because those are not - and as I'm saying this, I'm thinking, well, it's time maybe to revisit this. Because these are not your typical widely used service ports like FTP and SSH and web and so forth, they're not in GRC's ShieldsUP! small port list. They are on the service port list, except for 9100, because I go up to a little bit above 1024, but not to 9100. So if you did a full service port scan, look to see if 631 and 515 are showing as open, in which case something in your network, probably without your permission or knowledge, has opened those ports and is listening to - of course maybe you already got the robot ASCII art a couple days ago, and so you know that somebody somehow got into your printer.

The point is that this was just - they just used it as a printer this time. But this is a nice heads-up saying, okay, you've got your warning shot. Your printer is exposed. And we know from what we talked about last week from the research that was done that because, for example, PostScript is a full Turing-qualified language that is able to execute code, you just don't want your printer exposed. It creates an opportunity for someone to get into your network, onto your LAN, with a computer, because of course printers are

computers.

Leo: Is it sufficient to turn off UPnP?

Steve: I would say turn off UPnP and reboot. You could check from - the advantage that any port scanner like ShieldsUP! provides is it's got a public-facing view. You really can't see what's going on inside your network because there's just so much going on. What matters is what's exposed to the public. And so ShieldsUP! will, by looking back at your IP address, will give you a warning if anything that you are not expecting is exposed. And so it's worth doing a little ShieldsUP! scan every few months, just to make sure that none of the IoT devices has gotten up to any mischief.

But unfortunately, if we had a quiet LAN, if it was the way it once was, where all we had was some probably well-behaving PCs from conscientiously responsible security-oriented companies, then it's like, okay, well, that's probably okay. But now we've got Vizio TVs and light bulbs and the whole IoT thing. And we know that many of these companies are not being responsible about security. So if we give them access to Universal Plug and Play, it's just game over. They'll open ports for, they will argue, to provide functionality that their users want. But the problem is, in doing so, it creates entries into our network.

So the problem is, if you disable Universal Plug and Play, some stuff might break. Something might stop working. Now, that alone is a useful test to verify that that device was requiring Universal Plug and Play. Normally it's possible to create your own static mapping. That's, you know, the security-conscious gamers don't just turn on Universal Plug and Play in order to get their Xbox to be connected into the big network. They look at the five or six ports that are necessary, and they manually map those to their Xbox, and they get the same functionality without just letting it be open for the whole network.

And really, the fact that there's no security, that's the concern. But it was a choice, a tradeoff that was made. And arguably it was made, as I said, at a time when there was less potential vulnerability in the LAN because it was just going to be some Windows and Mac PCs, not refrigerators. Someone sent me a tweet of a Walmart slow cooker that had WiFi access.

Leo: Oh, geez.

Steve: And it's like...

Leo: I just got a WiFi-connected oven, you know.

Steve: Right. I think I watched you, yeah, you were talking to Stacey about that last Wednesday.

Leo: Yeah, the June Oven. And actually it's funny because I'm ready to cook last night and had salmon steaks all out, and the oven wasn't working. It was in a boot loop. So I went online, and they have chat support. And I said, "It's in a boot loop." They said, "Oh, yeah, you need to reset your oven. Unplug it, hold down the button,

and reboot." And I see the Android screen for refresh, you know, reloading the bootloader and refreshing the ROM. So I refreshed the ROM, and it all worked again. So we, I mean, we really - it's amazing, you know. And it won't work unless you connect it to the WiFi. It gets recipes and stuff.

Steve: Ah, what could possibly go wrong?

Leo: What could possibly go wrong?

Steve: So speaking of Vizio and IoT, the Federal Trade Commission has slapped them. I consider this a slap because it was $2.2 million. Actually, there were two types of...

Leo: FTC's action happened several years ago. Vizio settled today. That's the news story.

Steve: Okay. That was the news. Okay. Good.

Leo: So we've known about this for a couple of years.

Steve: Yes.

Leo: But what's interesting is Vizio agreed not to do it anymore. I wonder if they kept doing it until they settled. I don't know.

Steve: Yeah. I'm not sure about the timing here.

Leo: I'll tell you what, I'll look it up, and I'll give you a recap.

Steve: So the FTC said: "Consumers have bought more than 11 million Internet-connected Vizio televisions since 2010. But according to a complaint filed by the FTC and the New Jersey Attorney General, consumers didn't know that, while they were watching their TVs, Vizio was watching them. The lawsuit challenges the company's tracking practices and offers insights into how established consumer protection principles apply to smart technology."

So then it says: "Starting in 2014, Vizio made TVs that automatically tracked what consumers were watching and transmitted that data back to its servers." So I think you're right, Leo. It does mention that they retrofitted older models. But this sounds like probably shortly after that behavior was found, the FTC said, okay, hold on, folks.

Leo: Yeah. And they agreed to delete everything collected before March 1st of last

year.

Steve: Correct, correct. What I liked was a little bit of interesting information here, and that was that on a second-by-second basis, the Vizio TV was collecting a selection of pixels from the screen that it matched to a database of known television, movie, and commercial content.

Leo: Otherwise how would they know what you're watching?

Steve: Yeah. I thought that was just like, oh, wow.

Leo: Yeah.

Steve: So, well, so they went to some serious effort in order to track people. And of course the set itself might, if it was the tuner, it would know what you were tuned to. But if it was just the screen, like for example playing it...

Leo: Right, in most cases, right.

Steve: Right, exactly. So if you had another piece of equipment doing the channel selection and tuning, or a DVD player, it would be saying, okay, what movie is this? And so the idea of actually capturing video off of its own screen and then sending it back for identification is just like, okay. Well, again, they put some effort into this.

Leo: This is a longstanding practice, I've got to point out. Fifteen years ago...

Steve: This level of monitoring?

Leo: Yeah. Fifteen years ago we were offered at TechTV data from people's TiVos that had second-by-second viewing, graphs of what they'd rewound. And TiVo at the time, and I imagine still does, was selling that to networks for programming. It's like ratings.

Steve: Yeah.

Leo: And so that's - now, it's a little interesting that the monitor is doing that. And obviously they have to jump through more hoops. The TiVo knows exactly what you're watching; right?

Steve: Right.

Leo: I bet you anything Roku has that information. Apple TV has that information.

Steve: Yeah. I think they're monetizing it. You're right.

Leo: Of course they are, yeah. It's like ratings. It's better than ratings. It's gold.

Steve: Right. And the FTC's argument was that consumers didn't understand. And they called it, Vizio called it "Smart Interactivity" feature, although it didn't actually provide smart interactivity. And, I mean, we've been talking about recent FTC actions, for example, like in going after, who was it, the router maker, D-Link, I think it was. The FTC was stomping on D-Link. And I think that's going to be the arm of enforcement that we rely on for beginning to hold companies accountable for their behavior because that's what we've got here in the U.S.

Leo: You know, it's interesting that, as the FCC rolls back all of its regulations under the new administration, one of the things that they seem to want to do is to put all of the burden of this on the FTC, the Federal Trade Commission. And former Commissioner Tom Wheeler in a really good interview with Susan Crawford yesterday on Backchannel said that that's been the goal all along because the FTC is swamped because they have to do everything from dishwashers to microwave ovens. And they have an enforcement arm that's deceptive practices, but it's not going to be sufficient to protect Net Neutrality or any of these other things. But so all these companies like Comcast would just love the FTC to be in charge of enforcement because they know that they'd be - never hear from them.

Steve: Because they know nothing will be able to happen; right.

Leo: Yeah. Well, look at this action. This is stuff Vizio started doing in, what, 2014.

Steve: Yes. Yes.

Leo: So, yeah.

Steve: So a security researcher, Laurent Gaffie, gave Microsoft a 90-day heads-up to patch a flaw he found in - and this is just now - a Windows client processing of the Server Message Block, the SMB protocol, which is better known as Windows File and Printer Sharing. And of course this is a long, you know, this is a core Windows Microsoft protocol that's been around forever and has also been a constant source of problems. There were some stories a few years back of, because Internet Explorer would actually honor smb:// URLs, it was possible for websites that wanted to fingerprint their visitors to put an smb:// URL in the web page, and that would cause a Server Message Block connection from the user's computer back to the website. And through that you've got all kinds of information.

So anyway, so Laurent gave Microsoft 90 days, Microsoft did not respond in 90 days, and

so has just posted a Python proof-of-concept which allows a malicious server to crash a Windows client. And that's effective through Windows 10. I don't know how far back it goes. In the coverage I only saw Windows 10 and Windows 8.1. So I don't know one way or the other whether Windows 7 or, for example, XP would also be affected.

However, US-CERT has further warned that the vulnerability could do more than just crash a system, which of course is not surprising to us. As we know, these things start as crashes. And then when hackers look at them very carefully, they're able typically to extend that from just sending the program off into the wilderness - which is what we call a crash. It's extended into having the program execute code which they provide. And in this case US-CERT has warned that it can also execute arbitrary code with Windows kernel privileges. And the reason is the SMB protocol is deeply wired into the OS. So it is the kernel which is making this connection.

And so if you can do some sort of a buffer overrun-style vulnerability, then you're executing code with the permission of that service, which in this case is SMB. I'm hoping, because this, I mean, this is not showstopping, end of the world, because it does require that a client establish a connection to a server. The problem is there appear to be many ways to make that happen. Because it is the core protocol, the core networking and communications protocol in Windows, it's always there, and we have a long history of exploits against it.

So I'm hoping that the 90 days, essentially that Microsoft needed 97 days, that is to say, that next Tuesday's Patch Tuesday will fix this because there is a - it's on GitHub now, Win10.py, a very nicely written Python script which pretends to be a Server Message Block server and is able to crash any Windows machine that reaches out to it. And as we know, and as US-CERT said, today it crashes it; tomorrow it could be executing code on it.

The question, I think the big question is, is there still a way to get a website to cause Windows to make an SMB connection? I immediately researched that when I saw this story, and it wasn't clear, that most of the coverage is many years old where this was happening, and I'm hoping that that's not the case. There is a mitigation. And that is we know what SMB protocol uses. It uses TCP protocol over ports 139, and 445 and UDP over ports 137 and 138. So if you're in an enterprise, and your responsibility is the security of your enterprise, and you don't have a reason for machines in your Intranet, in your LAN, to be connecting to remote SMB servers - and if you do, you probably need to rethink your architecture because that's just a scary thing because SMB has been such a trouble-prone protocol.

My point is, if you don't know you need to permit outgoing connections to remote shares, remote SMB protocol, then blocking TCP 139 and 445 and UDP 137 and 138, that would be prudent. And that would prevent this until Microsoft has it fixed. But again, it'd just be better not to permit that unless you have a definite use case because then you're ahead of the game, not waiting for vulnerabilities to surface, but being protected from them beforehand.

Leo: I think this is when we met was NETBIOS vulnerabilities.

Steve: Yes. Yes.

Leo: Fifteen years ago.

Steve: And those quaint email scripts. You know, it's like, oh, that's so cute, running scripting. And again, I was railing, I mean, I was a broken record at the time. Who needs scripting in email? Turn that off. And it took five years, and finally Microsoft said, oh, maybe that's not a good idea. It's like, oh, you think?

Okay. So I love this, Leo. You will, too. I remember the first time I saw Android's grid sequence lock. I thought, that's kind of cute. You know, you put your finger down, it's a 3x3 grid of dots, and you trace out some interconnection pattern among the dots. And it's like, oh, you know, that's kind of cool. That seems - it's different than, at the time, iOS's four-digit passcode, which they've extended. It's easier than typing in a long passphrase. You just sort of do a little swipe-y thing, and your phone is unlocked.

The problem is, I mean, the moment I saw this research, I thought, oh, that's just so cool. Because, if you think about it, looking at someone do it, even if you didn't see the screen, they're just - there aren't, there's not sufficient ambiguity in their actions. And it turns out that the more complicated the sequence is, the easier it is to crack. And what the research did was these guys showed that, from 30 feet away, just by looking at video in standard static security cameras in Starbucks or wherever, somebody who's holding the phone in front of them, where you're just able to see the motion of their hand, with 95% accuracy it's possible to deduce the sequence.

And of course the reason is that, as you swipe around, if you go up, then you know that you had to have been down in order to go up. And if you go up twice, then you know that the down where you were was all the way down because there's only three levels. And the same thing for left and right. So, for example, if someone did a little tiny square in one corner, well, okay, there are, what, four possible locations on a 3x3 grid where a 1x1 square could be drawn. And you typically have five chances before you're locked out from guessing wrong. So it's very clear that there just isn't enough ambiguity if anyone can see the relative motions of your hand.

So in the abstract of this research - and, by the way, the paper is on - I got the link in the show notes - is on Google Drive. It's just beautiful. They show a whole bunch of little 3x3 grids with lines drawn on them to demonstrate how unambiguous these things are. In the abstract they said: "Pattern lock is widely used as a mechanism for authentication and authorization on Android devices. This paper presents a novel video-based attack to reconstruct Android lock patterns from video footage filmed using a mobile phone camera. Unlike prior attacks on pattern lock, our approach does not require the video to capture any content displayed on the screen. Instead, we employ a computer vision algorithm to track the fingertip movements to infer the pattern.

"Using the geometry information extracted from the tracked fingerprint motions, our approach is able to accurately identify a small number, often one, candidate pattern to be tested by an adversary. We thoroughly evaluated our approach using 120 unique patterns collected from 215 independent users, by applying it to reconstruct patterns from video footage filmed using smartphone cameras. Experimental results show that our approach can break over 95% of the patterns in five attempts before the device is automatically locked by the Android OS.

"We discovered that, in contrast to many people's belief, complex patterns do not offer stronger protection under our attacking scenarios. This is demonstrated by the fact that we're able to break all but one complex pattern with a 97.5% success rate, as opposed to

60% of the simpler patterns in the first attempt. Since our threat model is common in day-to-day life, this paper calls for the community to revisit the risks of using Android pattern lock to protect sensitive information."

Anyway, I just thought that was, you know, it's the kind of thing where, again, in retrospect it's like, oh, yeah, that would not be that difficult to do. But it's sort of - so somebody who wasn't thinking about the attack angle, so to speak, said, oh, this is interesting and different and fun. Unfortunately, it's just not very secure. If anyone can see you even from a distance do it, there just isn't much guesswork involved in figuring out what you did.

Leo: They kind of acknowledge that. When you choose your screen lock, they rank them from least secure, which is the pattern lock, to the most secure, which is a strong password. So people do get a little bit of advice, anyway, from Android when they set that up.

Steve: And probably people, again, as we know, people are, A, not actually that concerned about their security and will opt for convenience.

Leo: They want something you can do with one hand. The nice thing about pattern lock, you do it with one hand.

Steve: Yup.

Leo: So if somebody is in the situation, they're a strap hanger, whatever, and they need to - on the subway, you know, and they need to use one hand, they almost always will choose pattern lock. And usually they'll choose very easy-to-deduce patterns. John C. Dvorak was always trying to figure it out from the grease on my phone. Because if you do it enough, it will kind of form a pattern on the phone.

Steve: Yeah, yeah. So Network World had some reporting on this. And I paraphrased it a bit. And their title was "Cops use pacemaker data to charge homeowner with arson and insurance fraud." They said: "If you're dependent upon an embedded medical device, should the device that helps keep you alive also be allowed to incriminate you in a crime? After all, the Fifth Amendment of the U.S. Constitution protects a person from being forced to incriminate themselves. Nonetheless, that's what happened after a house fire in Middletown, Ohio.

"WCPO Cincinnati" - which is a TV station - "caught video of the actual fire, as well as delivered news that the owner's cat died in the fire." And then they editorialize. "As a pet owner, it would be hard to believe that a person would set a fire and leave their pet to perish that fire. The fire in question occurred in September 2016," so a few months ago. "The fire department was just starting an investigation to determine the cause of the blaze. A month later, 59-year-old homeowner Ross Compton was arrested and charged with felony aggravated arson and insurance fraud. The cause of the fire was still undetermined, but it had resulted in $400,000 in damages to the house and contents of the 2,000-square-foot home.

"Fire investigators knew there had been," quote, and this was suspicious - "'multiple

points of origin of the fire from the outside of the residence.' At the time, the police cited inconsistencies in Compton's statements when compared with the evidence from the fire. There were additional conflicting statements given to the 911 operator. Compton had said 'everyone' was out of the house, yet the 911 operator also heard him tell someone to 'get out of here now.' In the 911 call published by WLWT5, an out-of-breath Compton claimed he had 'grabbed a bunch of stuff, threw it out the window.' He claimed to have packed his suitcases, broken the glass out of a bedroom window with his walking stick, and tossed the suitcases outside. Compton also told the dispatcher he had 'an artificial heart,'" which he didn't have.

"After this, things really get interesting because police investigators used data from Compton's" - well, maybe it was artificial because I had - well, no, the term is "pacemaker" elsewhere. So "...Compton's electronic heart device against him." And the question is, "Isn't that self-incrimination? Can a person plead the Fifth when it comes to self-incriminating data collected from their medical device?"

Leo: I don't think so. I'm no judge, but...

Steve: I agree.

Leo: They can collect your fingerprint. I don't know why they couldn't collect pacemaker information.

Steve: Yeah, I agree. "Police set out to disprove Compton's story about the fire by obtaining a search warrant to collect data from Compton's pacemaker. WLWT5 reported that the cops wanted to know 'Compton's heart rate, pacer demand and cardiac rhythms before, during, and after the fire.'" Then on January 27, just two weeks ago, the Journal-News reported that court documents stated: "A cardiologist who reviewed the data determined 'it is highly improbable Mr. Compton would have been able to collect, pack, and remove the number of items from the house, exit his bedroom window, and carry numerous large and heavy items to the front of his residence during the short period of time he has indicated due to his medical conditions.'"

"Middletown Police said this was the first time it had used data from a heart device to make an arrest, but the pacemaker data proved to be an 'excellent investigative tool.' The data from the pacemaker didn't correspond with Compton's version of what happened. The retrieved data therefore helped to make the indictment." So, yes, another instance of, I mean, we've talked about malicious attacks against things like insulin pumps and pacemakers in the past. And then we were just recently talking about somebody, there was a murder in a hot tub, and between 1:00 a.m. and 3:00 a.m. some huge amount of water was monitored and recorded from the person's IoT water meter. And so we're seeing increasing instances where police are being creative in gathering forensic evidence from all of the electronic things that surround us. Yikes.

So Will Strafach, S-T-R-A-F-A-C-H, Strafach, maybe?

Leo: I guess. I was talking about him, too, earlier, yeah.

Steve: Yeah. The president of the Sudo, I love that, the Sudo Security Group, as in S-U-

D-O...

Leo: And by the way, Rene told us on MacBreak Weekly this morning, well-known guy, highly respected. So listen. What he's saying is right on.

Steve: Yup. He posted, well, and he's got an interesting service that we'll talk about that they're just in the process of launching. So he posted yesterday on Medium, he said: "During the development of our web-based mobile app analysis service" - and that's what we'll talk about in a second, it's called verify.ly, V-E-R-I-F-Y dot L-Y, so verify.ly, which now it's got a page out so listeners can go check it - "verify.ly," he says, "it was essential to have a clear understanding of the most common security issues which plague mobile applications today. Automatically scanning the binary code of applications within the Apple App Store en masse allowed us to get a vast amount of information about these security issues."

Okay. So what they've got is they've got a static binary analyzer that scans Apple iOS applications, looking for problems that it's able to identify just by doing like a first-degree automated reverse-engineering of what the code apparently is. He says: "I will present some findings within this post which I believe to be in the public interest, related specifically to iOS applications which are vulnerable to silent interception of normally TLS-protected data while in use. Our system flagged hundreds of applications as having a high likelihood of vulnerability to data interception, but at this time I will be posting details of the connections and data which I was able to fully confirm as vulnerable using a live iPhone running iOS 10" - and he has in quotes "malicious," meaning his own test proxy server - "to insert an invalid TLS certificate into the connection for testing."

So to take a break here for a second, so what he's talking about is that, as I have been, it's like been my most oft-repeated fundamental, is that without authentication all you have is the illusion of security. If you do not know to whom you are speaking, then the fact that you're encrypting your data means nothing because modern encryption negotiates keys on the fly for security. But if the person you're negotiating that on-the-fly key with is a bad guy, and you don't know it, then it's self-defeating. So what this static analysis found was 76 popular iOS applications that do not verify the identity of the certificate to which they are connecting. Which means an illusion of security, but no actual provision of security. So he lists some highlights from what he found.

"During the testing process," he writes, "I was able to confirm 76 popular iOS applications allow a silent man-in-the-middle attack to be performed on connections which should be protected by TLS" - and in this case HTTPS - "allowing interception and/or manipulation of data [in flight] in motion." According to Apptopia estimates: "There has been a combined total of more than 18 million downloads of app versions of those iOS applications which are confirmed to be affected by this vulnerability."

For 33 of the iOS applications, this vulnerability was deemed to be low risk, meaning that all data confirmed to be vulnerable to interception was only partially sensitive analytics data about the device. Maybe partially sensitive personal data such as email addresses and/or login credentials, which would only be entered on a non-hostile network. So about half of them, a little less than half, while they weren't vulnerable from not verifying the identity of the certificate, it wasn't a horrible problem.

For 24 other of the iOS applications, this vulnerability was deemed to be medium risk so that they were able to confirm the ability to intercept service login credentials and/or session authentication tokens for logged in users, which of course we know would allow

an impersonation attack, very much like in the old days of Firesheep.

For 19 of the iOS applications, this vulnerability was deemed to be high risk, where they confirmed the ability to intercept financial or medical service login credentials and/or session authentication tokens for logged-in users. And we've been speaking the last couple weeks about Apple's App Transport Security feature, ATS. It turns out that that does not contain and help block this vulnerability from working.

He posted in detail, and I'm not repeating here, the low risk, those 33 low-risk applications; but he's withholding the names of the 24 medium and 19 high-risk applications for a responsible disclosure process. So the publishers of those applications have been notified and are being given time to fix this discovered problem before that's made public.

Leo: And they're only low risk because they don't hand over important information. It's the same mechanism all around; right?

Steve: Correct. Correct.

Leo: It's [crosstalk] information can be extracted.

Steve: Right. So in digging into this, I was wondering, you know, why can't Apple and their App Transport Security, ATS, and iOS simply enforce certificates? For example, we're used to platforms that do that for us. Like down at the OS level, if you have said to the OS, make me a connection over to here, then that comes back failed. If the handshake and the verification doesn't work, you get back an error saying "invalid certificate," and you're not talking to those people.

So he explains, he says: "This class of vulnerability poses a complex problem, as application developers are the only ones" - this is the key. "Application developers are the only ones who can fully mitigate it. It's derived from networking-related code within iOS applications being misconfigured in a," as he puts it, "a highly unfortunate manner. Due to this, Apple's 'App Transport Security' mechanism will see the connection as a valid TLS connection, as it must allow the application to judge the certificate validity if it chooses to do so.

"There's no possible fix to be made on Apple's side because, if they were to override this functionality" - that is, if the OS were to enforce it, as I was suggesting - "in attempt to block this security issue, it would actually make some iOS applications less secure as they would not be able to utilize certificate pinning for their connections," meaning that certificate pinning allows an application the freedom to obtain the certificate and get the fingerprint of the certificate, which is unforgeable, so that they're able to lock themselves to a given certificate remotely.

He says: "And they could not trust otherwise untrusted certificates which may be required for Intranet connections within an enterprise using an in-house PKI [Public Key Infrastructure]. Therefore, the onus rests solely on app developers themselves to ensure their apps are not vulnerable." So there's, like, no solution except what, I mean, and this is why what's being done here by the Sudo Group is so good is they've proactively scanned apps, developed candidate potential vulnerabilities, then verified them manually. So their automated system can raise red flags. But then they need to verify that it wasn't

a false positive. They do that, then they notify the application developers and say, look, you need to lock this down. You've got X amount of time to do so because we need to, as a responsible security research firm who's discovered this, we need to let people know if you don't get it fixed.

So the only workaround he offered in the interim, and of course we don't know which applications are vulnerable at the medium or the high level of vulnerability, so we don't really know what we should be protecting against. But he said, as a workaround, disable WiFi during sensitive transactions. He said: "There's a short-term trick which can be used to mitigate this type of vulnerability. The vulnerability is very likely only to be exploited if your connection is flowing over WiFi, whether you've joined a public WiFi network or a determined attacker has force-joined your mobile device to a rogue network without your knowledge.

"Therefore, if you are in a public location and need to perform a sensitive action on your mobile device, such as opening your bank app and checking your account balance, you can work around the issue by essentially opening Settings and disabling WiFi prior to the sensitive action. While on a cellular connection the vulnerability does still exist, cellular interception is much more difficult, requires expensive hardware, is far more noticeable, and is quite illegal within the United States. Therefore, it is much less plausible for a casual attacker to risk attempting to intercept a cellular connection."

So, and then to companies he addressed, he says: "If you offer an application in the iOS App Store, consider analyzing builds prior to App Store submission using our verify.ly service. This class of vulnerability and all other possible low-hanging fruits, meaning vulnerabilities discoverable to a determined attacker who commits 24 hours of total analysis time" - not deep analysis - "can be fully detected by performing an automated scan of the binary code and giving you an easy-to-read report outlining any and all flagged issues, ensuring your customer data is safe."

And then at the end of his post he talks a little bit about verify.ly. In fact, I went over to see what this was all about. And their site explains that verify.ly allows you - and by "you" they mean an app developer, this is not for the end-user - to scan the binary code of an iOS application to produce a human-readable report detailing all detected common security issues and a breakdown of all useful security-related information pertaining to the app. The app scan is performed in seconds using our proprietary automated static analysis engine, yielding actionable information regarding the security of the scanned mobile application. No source code is required.

And then, for example, under features, they have: Detects all common CWEs and mobile OWASP Top 10 issues within an app. Detects hardcoded sensitive content that is easy to reconstruct - API keys and secrets, encryption keys, passwords, and more. Detects any use of malicious, private, or risky APIs. Detects issues related to SSL and TLS validation, like was the subject of this posting. Detects issues with sensitive data at rest stored non-securely. Detects code hashes against databases of known malicious code. And generates easy-to-read report explaining issues and displaying relevant data in a high-level manner to ensure readability by the widest and potentially nontechnical audience.

And it sounds like it's in the process of coming online. They offer a professional plan for $100 a month, which is a single user account. A small business plan for $500 a month is coming that allows 10 users per account. And then for an enterprise class plan they say contact them. So a nice-looking service. And this is - I'm glad to see this, Leo. It's the kind of broad-based, automatable, binary-based testing that allows a company like this to provide a service of saying, you know, your app doesn't look to us like it's as secure as it needs to be.

Leo: Yeah. I like the name, too, verify.ly.

Steve: Yeah. So again, somewhere, maybe it was there, I saw something about it being a problem that, like one of the ways this is happening to developers is there is this drag-and-drop approach which is becoming increasingly prevalent, unfortunately. And that is that developers who are at deadline, under the gun, in a hurry, they will grab a blob of code off of GitHub or somewhere that has some functionality that they don't fully understand. They didn't write it.

And that's the point is they just grabbed it because they needed this chunk of function. And they drop it into their app, just assuming things about it which is not in evidence. And they get themselves in trouble as a consequence of code having behavior different than they expect. For example, not verifying certificates. They just assume it's being done and don't know one way or the other. So they get themselves and their app in trouble.

Leo: This is really common. Remember the flaws that were exposed, I can't remember, was it OpenSSL where they cut and paste Intel reference code with baseband radios because why rewrite it? Somebody wrote it. There's the code. I can get it from Stack Overflow or somewhere. Why would you need it?

Steve: Yeah. I think it might have been a whole bunch of routers that all had...

Leo: That was it, the routers, yeah.

Steve: They just took a chunk of code that was clearly marked by Intel as "this is just to demonstrate how to talk to the registers of our hardware chip. This is not for production."

Leo: Don't do this.

Steve: But it was like, oh, it works. We compiled it, and it worked, so we shipped it.

Leo: We shipped it.

Steve: Yeah. So last Friday, what, five days ago, confusing things further, muddying the waters, a U.S. judge ruled that Google, unlike the decision that had been reached for Microsoft, must turn over foreign emails. And this is from - I edited this down from Reuters reporting. And, boy, the logic is tortured here.

A U.S. judge has ordered Google to comply with search warrants seeking customer emails stored outside the United States, diverging from a federal appeals court that reached the opposite conclusion in a similar case involving Microsoft, that of course we all discussed at the time. U.S. Magistrate Judge Thomas - and his name is Rueter, R-U-E-T-E-R - in Philadelphia ruled on Friday that transferring emails from a foreign server so - here's the tortured logic - from a foreign server so FBI agents could review them locally

as part of a domestic fraud probe did not qualify as a seizure.

The judge said this was because there was "no meaningful interference" with the accountholder's "possessory interest" in the data sought. "Though the retrieval of the electronic data by Google from its multiple data centers abroad has the potential for an invasion of privacy, the actual infringement of privacy occurs at the time of disclosure in the United States," this judge wrote. Google replied in a statement Saturday, that is, over the weekend: "The magistrate in this case departed from precedent, and we plan to appeal the decision. We'll continue to push back on overbroad warrants."

And this ruling came less than seven months after the Second U.S. Circuit Court of Appeals in New York, which is what we discussed at the time, said Microsoft could not be forced to turn over emails stored on a server in Dublin, Ireland that U.S. investigators sought in a narcotics case. That decision last July 14 was welcomed by dozens of technology and media companies, privacy advocates, and both the ACLU and U.S. Chamber of Commerce.

A few weeks ago, on January 24, 2017, the same appeals court voted not to revisit the decision. So they were staying by it, and it had already gone to appeal; and they said, no, we're sticking by it. The four dissenting judges, however, called on the U.S. Supreme Court or Congress to reverse it, saying the decision hurt law enforcement and raised national security concerns. Both cases involved warrants issued under the Stored Communications Act, a 1986 federal law that many technology companies and privacy advocates consider outdated.

In court papers, Google said it sometimes breaks up emails into pieces to improve its network's performance, and did not necessarily know where particular emails might be stored. That's going to be a hard one to substantiate, of course, because if you want to retrieve your email, you can. So obviously Google knows how to find it. Relying on the Microsoft decision, Google said it believed it had complied with the warrants it received by turning over data it knew were stored in the United States. And in background, Google receives more than 25,000 requests per year from U.S. authorities for disclosures of user data in criminal matters, according to the judge's, to Judge Rueter's ruling.

And the takeaway for our listeners is email is not secure and arguably not securable. If you want to exchange messages securely, then TNO is the only solution. Write the message in a simple editor. Encrypt them yourself using a high-entropy key that you arrange to share with the other party out of band, and then send that encrypted blob however you want. If you write your message, you encrypt it yourself, then it doesn't matter what you do with it. You could send it by Pony Express, by donkey, by unencrypted email. It's just a blob of binary that nobody can ever access except the person who has the matching key. And so that's the way to do it.

People, I saw a lot of people all excited about Ladar Levison's email service coming back online. And it's like, okay. None of this ever moves me. In fact, Google even has a hosted S/MIME service that they're putting together to make it, again, to increase the security of email in transit by using TLS-style authenticated connections. The problem is, as soon as it moves out of that protected environment, and if it's stored encrypted, and Google's announcement of this hosted S/MIME service says yes, you know, we store it under our own encryption, well, that means that a judge can say, okay, and we'd like you to decrypt it now, please.

So TNO applies to messages in the same way that it applies, as we've been discussing now for years, to user data stored in the cloud. At the moment, it's more of a manual process. But I would argue, if you need to send something to someone, and you actually

do care about it being secure, then Trust No One. Encrypt it yourself, and then it doesn't matter how you send the blob. Nobody will ever be able to see what's inside there.

Leo: It could be pretty much automated. I mean, if you have GPG, Gnu Privacy Guard, it pretty much automates the process for you. So it's about as easy as can be. Once you've got somebody's public key, you can even say, "Every communication I have with them should be encrypted with this key."

Steve: Right, right. So if both endpoints are going to be communicating, and of course that's...

Leo: That's the only challenge.

Steve: Yes. And famously, that was the whole Edward Snowden and Greenwald...

Leo: And Greenwald couldn't figure it out, yeah.

Steve: Right. Right. So from the Blast from the Past Department, and it actually is a bit of a blast, we have the return of SQL Slammer, of all things. Check Point's blog noted that, starting on November 28th of last year and running through December 4th, so just until a month before last, Check Point observed a huge spike in the long-dead SQL Slammer networking traffic.

Okay, so what was SQL Slammer? This predates, as I was saying at the top of the podcast, it actually predates the podcast. Fourteen years ago, back in 2003, many web servers using SQL database were misconfigured to expose the standard SQL query port, which is 1434, to the public Internet. So a web server does want to expose port 80 and 443 for doing web services. It probably has processes running on the server which want to connect locally to the SQL database using the localhost 1434 port, but never intended for that SQL database to have the port 1434 made public.

Well, it was. Microsoft SQL Server 2000 and the Microsoft Developer Edition Tools 2000 both exposed that port and had a buffer overrun vulnerability. And the reason it was so bad was it would work over UDP, which allows IP spoofing and single-packet transactions. You don't have to do the whole TCP connection setup and everything. You're able just to spray UDP packets with abandon. And so what happened was the receipt of a single UDP packet containing this SQL Slammer worm at port 1434 would go into the server, blow the buffer, cause the execution of the packet, and, statically, it would essentially run a subprocess in the server that then itself began spewing that packet back out at random IP addresses all over the Internet.

So at the time there was a huge number of exposed SQL database servers. And they all got found by - and the reason we called this a worm is exactly that behavior. A worm is by definition a self-propagating thing on the Internet. And so someone somewhere sent one of these UDP packets, they launched this worm at one SQL server, infected it, and it started spewing these UDP packets at random that eventually found other SQL servers that were exposed, which in turn began spewing at random.

And before long, I mean, like almost immediately, the Internet was filled with all this SQL

Slammer traffic. And all of the exposed SQL servers were found and infected and participating. And so, and if any servers were, like, offline for the weekend, and then came back up online on Monday, wham. They were getting slammed, literally, from all the other SQL servers when they happened to choose that server's public IP by chance. So it was quite a mess.

Now, here we are, 13 years, 14 years later, and Check Point noticed a return of SQL Slammer and, you know, scratched their head. They don't know where it came from or why. What must have happened is that, because old code never really dies, someone forgot about this being a problem and for some reason, somehow, established a bunch of new Microsoft SQL Server 2000s on the Internet. And somewhere there was probably an SQL Server in a closet that never stopped scanning. Remember, we refer to this often as Internet Background Radiation, IBR. These worms will never die. So somewhere there was one or two that didn't die. They just kept sending out their SQL Slammer UDP packet at random.

Something else caused a population of Microsoft old - what now, SQL Server 2000? - so 17-year-old SQL Server to be brought up on the Internet, and that closeted SQL Server ended up hitting them, and the worm began to live again. And as the population of infected machines grew, the rate of probes being sent out by all of those grew. And so it accelerated the rate of finding all the rest. And before long there was a whole bunch of SQL Slammer traffic on the 'Net. So this is the way these things actually happen.

Leo: Steve Gibson, we are talking security on Security Now!, as always.

Steve: We are indeed. So we talked a couple weeks ago about this big hack of Cellebrite, which is the well-used by law enforcement mobile hacking service.

Leo: I saw a number of articles that said this was the one the FBI had used to get the San Bernardino iPhone, but I don't think we know that. I don't think we ever found that out.

Steve: We don't, right, I don't think we know for sure. But we do know from the records that have been leaked that it's heavily used by law enforcement.

Leo: Oh, they have a lot. They make that device that can suck the data out of any cell phone in seconds.

Steve: So nearly a terabyte, 900GB of phone-hacking tools and databases and information of all kinds, we reported some weeks ago when the industry was covering this, were allegedly stolen from them. We're beginning to see some of this data surfaced. The hackers' ReadMe in the public release of some of this noted that much of the iOS-related code is very similar to that used in the jailbreaking scene populated by a community of iPhone hackers that typically breaks into iOS devices and releases its code publicly for free.

Jonathan Zdziarski, I always trip over his name, Jonathan Zdziarski, and we've talked about him often, who's a well-known forensic scientist, agreed that some of the iOS files were nearly identical to tools created and used by the jailbreaking community, including

patched versions of Apple's firmware designed to break security mechanisms on older iPhones. A number of the configuration files also referenced

"limerain," the name of a piece of jailbreaking software created by infamous iPhone hacker Geohot. He said he wouldn't call the released files "exploits," however, meaning Jonathan said. Why can't I say that?

Zdziarski also said that other parts of the code were similar to a jailbreaking project called QuickPWN, but that the code had seemingly been adapted for forensic purposes. For example, some of the code in the dump was designed to brute force PIN numbers, which may be unusual for a normal jailbreaking piece of software.

Zdziarski noted: "If, and it's a big if, they used this in UFED or other products" - that's the Cellebrite flagship product - "it would indicate they ripped off software verbatim from the jailbreak community and used forensically unsound and experimental software in their supposedly scientific and forensically validated products." So this is really not a surprise. Essentially, this Cellebrite Group are just vacuuming up all of the hacking and cracking work being done by the open hacking jailbreaking community, pulling it together, sticking their own label on it, and then reselling it as a package to law enforcement, apparently not really making any changes to it. So it's like, oh, well, you know. Not a big surprise, but interesting to know that basically they're just recycling what's available publicly with a little more work.

There was also, Techdirt reported, some worrisome movement in the Encrypted Web Extensions standardization process. We've talked about this a couple times. This emerging, it's called EME, the Encrypted Media Extensions. I was going to say, wait, not encrypted web extensions, EME, Encrypted Media Extensions. So there's an ongoing fight and dispute, essentially, as a W3C, the World Wide Web Consortium, over encrypted media extensions, which is the HTML5 digital rights management scheme that several companies want incorporated, embedded in the web standards. And that fight took two worrying turns recently, reports Techdirt.

First, Google slipped an important change into the latest Chrome update that removed the ability to disable its implementation of EME, the encrypted media extensions, which further neutered the already weak argument of supporters that DRM is optional. So it's becoming non-optional. And in the latest version of Chrome, no longer optional.

But the other development is even more interesting and concerning. Dozens of W3C members and hundreds of security professionals have asked the W3C to amend its policies so that its members cannot use the encrypted media extensions to silence security researchers and whistleblowers who want to warn web users that they're in danger from security vulnerabilities discovered in browsers. So far, the W3C has stonewalled on this point. This weekend the W3C executive announced that it would not make such an agreement part of the EME work, and endorsed the idea that the W3C should participate in creating new legal rights for companies to decide which true facts about browser defects can be disclosed, and under what circumstances.

So that's a problem. One of the major objections to EME has been the fact that, due to the anti-circumvention copyright laws of several countries, including ours, the U.S., it would quickly become a tool for companies to censor or punish security researchers who find vulnerabilities in their software. The director of the standards body called for a new consensus solution to this problem; but, not surprisingly, "The team was unable to find such a resolution."

So there's headbutting going on. The new approach will be a forced compromise of sorts

in which, instead of attempting to carve out clear and broad protections for security research, they will work to establish narrower protections only for those who follow a set of best practices for reporting vulnerabilities. In the words of one supporter of the plan, it won't make the world perfect - now, this is a supporter; right? It won't make the world perfect, but we believe it's an achievable and worthwhile goal.

So they're trying to come up to a compromise. But the argument is this is not a real compromise because, by working to determine where the line should be drawn, it creates an important presumption, or an implicit presumption, that there is a line to be drawn. Having a policy is a tacit endorsement of the use of DRM for censoring security researchers, to whatever degree, because the argument is not about to what degree such use is acceptable, but whether such use is appropriate at all.

So another battle here. And maybe we're going to end up coming out, we the industry, coming out on the right side of this. But, boy, there is a lot of pushback from the content holders that want DRM to get pushed out to the browser. And if we have that, the question then is when vulnerabilities are found, what happens? My take is that, if responsible disclosure is criminalized, that is, I believe researchers are still going to find, going to be digging into this and finding problems.

The question then is, what do they do with the knowledge that they gain because having done that is a criminal activity, which doesn't mean it won't happen, but it will mean that it gets forced underground. So I think disclosures will still be made. But unfortunately, they won't be able to be made responsibly. They'll be made anonymously and publicly, creating reputational damage for the companies who could have fixed those problems under responsible disclosure prior to public disclosure, and also endangering end users, who'll be using products with known public and unpatched problems. So it's a mess. And unfortunately, it really sounds like there is a serious dispute that's not getting resolved at the moment in the W3C.

I'm going to skip a few things that are sort of minor. And I did have, I found an interesting question I wanted to cover from a tweet from Passeride. He said: "Thoughts" - he was asking for my thoughts - "on salting password hashes with characters in the username randomly and deterministically selected based on unhashed password?" So this was interesting. We've talked about salting passwords and the need for a per-account salt. We were talking about it just last week, that if all passwords were salted, well, first of all, if all passwords were unsalted, that was a big problem. But if it was a global salt that was used, that was mixed with the password, then the problem would be, once that global salt was known, then you essentially had the equivalent of an unsalted password. The only safe thing to do was to obtain a random per-user salt, which you store along with the user's account, and use that when the user puts their password in the first time, and then also when they put it in again.

So Passeride is saying, okay, what if the salt is not random? What if it's derived from something the user puts in, like their name and the unsalted password? And I'm sure that close listeners to the podcast will realize that that doesn't give us any protection. Now, using it, incorporating characters from the unhashed password complicates things. But it doesn't make it completely random. And so I do think it's clearly superior if the salt is unrelated in any way to what the user puts in. It's a completely random nonce which is unique per user and isn't tied, deterministically tied, to information that the user provides.

And then Matt Clare also asks, he says: "Why do botnets prefer IRC?" And he suggests: "Tor hidden too much work? XML too fancy? Hiding in plain sight safer?" And he notes that he's been a listener since Episode 1. And I remember looking at this back in my

early days of looking at DDoS attacks. And I realize that IRC is used for a couple of reasons. It is a very simple protocol. It's so old that it's been implemented in virtually every language. There's Perl script and Python and, I mean, you can find IRC chat clients everywhere. And IRC stands for Internet Relay Chat.

So that's a convenience for botnets. What that allows is that allows botnets to all convene in an IRC chatroom on a specific server or servers, yet to be controlled by the botmaster somewhere entirely else. And the Internet Relay Chat, because it relays chat over the Internet, will automatically propagate the commands that somebody issues anywhere in the world in IRC to the server that is listening for incoming commands in that chatroom. And so botnets prefer it because it is, as Matt suggested, simple to use, but it also has some strong benefits in terms of the botmaster hiding themselves. They never have to connect to the server where the bots are aggregating. They're able to use IRC itself to propagate their commands to the botnet. Which makes it handy.

And some final things. I did want to note that Season 2 of "The Expanse" series has returned. A huge percentage of our listeners really liked the first season. It's based on a series of books that I read a couple years ago when I knew that Syfy channel was going to be producing it. And the series does stand out as remarkably high quality compared to most of the stuff that's there. I wanted to recommend a series of four books called the "Intrepid Saga," from an author M.D. Cooper. He has a site called Aeon14.com, spelled with an "A." So A-E-O-N-1-4 dot com. That's all you need to know to find it. Aeon14.com will take you there.

Four books in the first set. There is a fifth one that I'm now anxiously waiting for that's - I don't have to wait very long. It's due out in April. But the fourth book ends that first set, so you don't have to worry about being left at a cliffhanger. On the other hand, number five will probably be ready by the time you are. I really enjoyed them. And especially the fourth one. It's good character development. It is sort of military sci-fi, set in the far future, in the year 4000 something. So the solar system has been fully populated. We've terraformed some worlds. A lot of the planets have elevators going up to rings that are circling them. Really nice, rich, well fleshed out future vision.

And at this point Sol is getting a little crowded, and so we're sending off world ships, first to terraform, and then colony ships on multi long missions with the colonists in stasis to go and spread humanity around. And so this is set in that sort of a backdrop. And it's just - it's very enjoyable, a very strong female protagonist, Tanis Richards. And I think people will like it. And, boy, the fourth book, it just really comes together. It's almost like someone else wrote it. I don't know. Maybe he came back from vacation or something before he wrote it. But it just grabbed me. So I can't say enough good things about it.

The Healthy Sleep Formula that we've talked about briefly and that has become extremely popular and helping a lot of people, is in the process of being made into a packaged product so it's simpler for people to take. HealthySleepNow.com, for anybody who's interested. There's an Indiegogo project that's being put together. I'm not involved in it in any way, so I'm not making any representations about it, although the person who's doing it is an advocate of it and is soliciting people's feedback and input. So HealthySleepNow.com.

And what else? Oh, and I did get a tweet from someone named Morgan Speck, who copied me on a tweet to Hover. He said: "You did everything for me automatically," to Hover. "I see now why @SGgrc was raving about how great you all are. Thanks!" And that gave me an opportunity to note that I just moved another domain. I have a domain BeyondRecall.com, which I'll be using in the future as a spinoff, a high-speed secure wiping facility for hard drives and SSDs. I moved it from Network Solutions. I'm

migrating all my stuff from Network Solutions over to Hover. And I did that just last week. And again, Hover, I'm so happy with the job that they're doing.

I did get actually two tweets on the first of February, so the day after the last podcast, last Wednesday. It sort of held me in suspense. The first one is from a guy whose Twitter handle is @arcanecode. First one came in at 6:09 a.m., and he said: "Not much longer for #SpinRite to run on this laptop. Crossing my fingers @SGgrc." So he wrote to me. And he enclosed, he attached a Twitter pic that showed the box counting down the amount of time remaining. And then about 3.5 hours later, at 9:29 a.m., he again tweeted: "YAY!" in all caps. "#SpinRite fixed my hard drive. My laptop now works without the hard drive constantly churning. Thanks @SGgrc." So that was nice to see, happening sort of in real-time.

And a nice note to remind people that it's not only for data recovery and when you're worried that all might be lost. But when the laptop isn't working the way you think it used to, or the desktop seems to pause - that's actually a really scary sign. If your desktop seems to sort of freeze or pause for a while, there's no visual indication that your hard drive is having a problem at that time. But that's often what is going on. We know that because people run SpinRite, and then that all stops. So it was the drive behind the scenes, or the OS, trying, hitting a bad spot and working with the drive to deal with it. Running SpinRite makes the fix permanent.

And lastly, two-armed bandits. I got a kick out of this story because I knew our Security Now! listeners would appreciate it. Wired.com provided the coverage for this. And Leo, you had already, I guess you said that you and Lisa had seen this.

Leo: Yeah. Lisa is a slot-machine player, so she had a serious interest in this overall.

Steve: So I'm just going to summarize this because we're just about out of time. Essentially what happened is Putin outlawed gambling because he thought that that would reduce the amount of underworld activity in Russia. And that forced the resale of a large number of electronic slot machines out onto the market. That allowed the examination and reverse-engineering of this large population of slot machines. And it was discovered that they were not using high-quality, hardware-based, like quantum-level noise in a reverse-bias diode that we were talking about just last week. They're using pseudorandom number generators. And they're not very good.

So as we talked about the problem with pseudorandom number generation as why, for example, I don't use them in SQRL, because if you are ever able to obtain a snapshot of the state of a pseudorandom number generator, then you can predict all of the future because it's just a software algorithm. And so I did a podcast, we called it "Harvesting Entropy," a few years back, where I explained in detail the way I'm avoiding that trap in SQRL. SQRL doesn't have a high need for randomness. Generally it's just deterministic working from what the website generates. But when you create an identity, you want it to be an absolutely random 256-bits. So I arranged to do that with something that generates very, very high-quality entropy and not anything where taking a snapshot of it would allow you to predict the future. It specifically protects against that.

Unfortunately, it looks like from the coverage of how widespread this problem is, none of the slot machines that are in use are using high-quality true random numbers. They are using some indeterministic. And so here's how this works. Of course I called this "Two-Armed Bandits" because the bandits themselves have two arms, but they are exploiting one-armed bandits, the common designation for slot machines. By watching slot

machines operate, and in the later attacks using Skype to stream the video back to the exploiters, who happen to be in - I think they were in St. Petersburg, I think they're in Russia - the current state of the pseudorandom number generator could be determined just by watching the outside of the slot machine.

And first of all, I think that's unconscionable that this design would be so poor. But that's the story. So by essentially playing the slot machine for some length of time, while capturing what it does, it is possible from looking at the series of outputs to reverse-engineer fully the software state of the pseudorandom number generator and then predict the future. So the way this was done was that - the way it was first caught was the bad guys would hold their smartphone in their hand, and one quarter of a second before they were supposed to press the button to stop the spinning of the wheels, their phone would vibrate.

So the phone was set to vibrate a quarter second before pushing a button would cause a payout. And even though it was, you know, your timing might be a little bit off, so you wouldn't get a payout, but that was enough of an advantage, having the phone vibrate. And the person, as quickly as they could, immediately pushed the button on the slot machine, biased it enough that these guys were generating 10s and 100s of thousand dollars a week in payout across the U.S. And later, because they were caught in the video surveillance of the casinos, they were caught holding the phone, they started wearing it in a shirt pocket and with an abrasive grill so that they would be able to feel it easily vibrate in order to make it less obvious what they were doing.

But it turns out that, because the randomness of the slot machines wasn't sufficiently random, and you could, by looking at the outside of it, you could determine the current state of its pseudorandom number generator inside, you could then predict the future. And that would give you a bias to substantially overcome the house odds and end up getting a lot of money out of the one-armed bandits.

Leo: Unless you're playing nickel slots, in which case...

Steve: Yeah, it's not going to be much money.

Leo: Not going to be much worth it. Well, it is a great story. It just shows you the flaws in random number generators.

Steve: And these things generate so much money, Leo, they could afford a reverse-bias diode in there.

Leo: Yeah, put a diode in.

Steve: As I said, they must just be repurposed pong machines or something from the 1970s. I don't understand. I mean, they are so expensive.

Leo: It's old Russian technology. It's probably Soviet technology. Here is, I thought, for you, your benefit, I thought you would enjoy this from Stack Overflow. They do

an analysis from time to time on popular programming languages based on the number of questions they get, and the differential between languages used during the weekday and during the weekend.

Steve: Interesting.

Leo: People unfortunately during the weekday ask a lot about SharePoint and PowerShell and VBA. But on the weekends it's Haskell and assembly language, baby.

Steve: Nice.

Leo: Number one and number two.

Steve: Yeah, you know, assembly's making a bit of a resurgence, thanks to IoT, because these things have to be super cheap. And money matters, and cost matters. And there just isn't much room in those little things.

Leo: You know, Forth is also getting a little bit of a resurgence because these little chips, these little Arduino and Stamps and things, Forth is perfect for that.

Steve: Absolutely.

Leo: Hey, great fun. Thank you, Steve, as always. You can catch Steve's act on his website, GRC.com. That's where you should go to get SpinRite, the world's best hard drive maintenance and recovery utility and Steve's bread and butter. Make that yabba-dabba-doo happen, baby. He also - some places you make it rain, other places you make it yabba-dabba-doo. You can also find the podcast there. He has audio, 64Kb audio and, uniquely, transcripts, so you could read along, side by side. That's often the best way to consume the show.

GRC.com. While you're there, there's a feedback form, GRC.com/feedback. There's all sorts of great information. I can just go on and on. But really it's a fun site just to browse around, explore. GRC.com. We have audio and video of the show on our web page, TWiT.tv/sn for Security Now!. And of course you can watch us live every Tuesday, 1:30 p.m. Pacific, 4:30 Eastern, 21:30 UTC at live.twit.tv, TWiT.tv/live, or now on YouTube live at YouTube.com/twit, however you like to watch it. You can also get it on demand. And you know that. Just go to our website or find yourself a place to subscribe, a podcatcher, and you'll get every episode. Two episodes away from No. 600.

Steve: Woohoo!

Leo: Celebrating - what did you say? - 12 years. That's about right.

Steve: Yeah.

Leo: Wow. Steve, thanks so much, and we'll see you next time.

Steve: Okay, my friend. See you next week.

Leo: Bye-bye.

Steve: Bye.