IoT Infancy Part 2

Description: After Steve rants a bit about the reality of OS versions and security, he and Leo cover the past week's security events, including a new zero-day vulnerability affecting all previous versions of Windows; a truly horrifying and clever chip-level exploit; yesterday's Android Security Update; a sad side-effect of Microsoft's GWX pressure; Mark Zuckerberg's old LinkedIn password; Facebook's plans for optionally encrypting Facebook Messenger; five things that challenge self-driving cars; and some miscellany. Then we conclude our look at the horrifying problems with our infantile Internet of Things.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-563.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-563-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We're going to of course cover the security news. And as usual, there's a lot of it, including a new Windows vulnerability. Hey, only $95,000 if you've got the scratch. We'll also get back into this IoT security issue. He's going to take a look at three different makes of baby monitors and show you just how insecure they can be. That's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 563, recorded Tuesday, June 7th, 2016: IoT Infancy Part 2.

It's time for Security Now!, the show where we talk about the latest insecurity and privacy and protecting yourself online with this guy right here, Mr. Steven Gibson of GRC.com.

Steve Gibson: Yo, Leo.

Leo: Yo, Steve. Good to see you.

Steve: So it's California Primary Day.

Leo: I voted.

Steve: Yup. I did, too. I'm a permanent by-mail voter.

Leo: I am, too. But you know what I do is I take the mail ballot, mark it up at home because that's easier.

Steve: Yup, and then go down to the polling place.

Leo: But I like to get my sticker.

Steve: Yup. Fun to have the "I Voted" sticker. I like that, too.

Leo: So in fact I told them at the polling booth, first I said, "Thank you for your service, I appreciate you doing this, and I only came here for the sticker." And they said, "All right, put your ballots in the box there, and here's your sticker."

Steve: So we're going to continue with last week's coverage, thanks to some really, really great reporting that Rapid7, the security firm Rapid7 did when they analyzed the security of 10 different baby monitors. First part, last week, we talked sort of generically about sort of the fundamental problem of a non-security focused product which wants to create a turnkey appliance that is, in this case, selling to a market that they're not security people, I guarantee you. I mean, we may have some listeners who have baby monitors, but not the other way around. It's not like they're all concerned about or even conscious of the issues of security.

So here are these products selling on the Internet which offer remote viewing of the activity in front of the camera. And unfortunately, as we will cover in detail at the end of this podcast, they're offering much more access than any owner could possibly believe. And this is the way the world is at this point. I'm hoping, and I'm going to summarize at the end, one of the consequences of us not fixing this quickly.

So anyway, I think a really interesting podcast. We've got a bunch of stuff to talk about. I'm going to begin with, before the security news, what I'm going to try to keep from being a rant. I called it the "reality check timeout." So I just have to say something because it's been - the pressure's been building for a couple months. And it's like, okay. But I'm going to keep it - our listeners will not have to turn the volume down on their earphones.

There is a new zero-day Windows exploit on the market, and literally "market," selling for $95,000. A horrifying and very clever chip-level exploit, that is, silicon-level, which has been called by the press "demonically clever." And we're going to talk about the details of how that works. Yesterday was the June update for Android, with a bunch of fixes. A little bit of news on two fronts about Facebook, one that affected Mark Zuckerberg directly, and one involving Facebook's Messenger program.

I ran across an interesting little aside in The New York Times I want to talk about, about the five things that give self-driving cars the biggest problems. Many of our listeners who were really listening and paying attention when I was talking about SQRL's authentication management, all brought up the same issue. So I want to cover that. We have a little bit of miscellaneous stuff. And then we're going to get into the horrifying details of why you

want to be very careful when you buy a baby monitor that offers Internet connectivity.

Leo: Especially if you're monitoring more than your baby.

Steve: Yes, exactly. You don't want this aimed into your bedroom.

Leo: Yeah. You know what I'm saying here.

Steve: And I was worried about maybe the crib next to the bed, so they'd get sort of a package deal. Oh. Anyway.

Leo: On we go. Steve Gibson, Leo Laporte, and the security news.

Steve: So, okay. My reality check timeout.

Leo: Uh-oh.

Steve: During which I promise not to raise my voice.

Leo: Okay.

Steve: What I keep seeing is Windows 10 people saying that Windows 10 is more secure.

Leo: Yeah.

Steve: Which is utter nonsense. And the distinction I need to draw for, like, again, the reality check, is that added security features is completely different from security. And in fact you can argue strongly that added security features decrease the security of the system until they are proven. I will never forget Steve Ballmer prancing around onstage before the release of Windows XP, claiming that it was the most secure operating system ever. And we said at the time, you can't say that. That's not something that anyone can declare. XP had more features. In fact, it was the first Windows operating system with a firewall. Unfortunately, they forgot to turn it on. So it wasn't until Service Pack 2 that it was turned on.

So the point is, I don't argue that Windows 10 has more security features. But that's completely separate from it being more secure. And in fact every lesson that we are taught by feedback from reality is that new code has bugs. And the more new code you have, the more bugs you have. And it takes time to find them and remove them. And Leo, you were talking, I think it was on Saturday, you were mentioning Debian's stable track and how there's been some controversy over some guy that wants to change the XScreenSaver on the Debian...

Leo: Yeah, JWZ.

Steve: Yeah. And they're saying no, don't mess with it. And I was just sort of smiling because it's like, yes, that is the correct philosophy if you want stability. If you want a stable system, if it's not broken, don't fix it.

Leo: That's actually really a good point. That's the greybeards' attitude in Debian is, if you use Debian stable, you're going to be using old software.

Steve: Right.

Leo: Because it's proven. And we'll do patches, but we're not going to be upgrading to the latest stuff all the time.

Steve: I only moved GRC's servers off of Windows 2000 a couple years ago because Windows 2000 didn't understand any of the new SSL/TLS protocols. So I had to. But that thing sat there and ran literally years at a time, just like a bulletproof Unix box that just sits in a closet, and you forget about it because it never has a problem. I mean, that's what you want, if you want stability.

Now, I get it. In fact, Paul's term "Windows enthusiast" is the perfect one because, again, I don't have anything against Windows 10. As everybody knows, I use Windows. I love Windows. But I also understand that new is not necessarily better. And that's the other thing is that this is not, Windows 10 is not a new operating system. That's another myth that Microsoft pushes and has always pushed because they want to get people to upgrade. Until Windows 10, it was revenue-generating for them. Now they're going to arrange to generate their revenue somehow else.

But how do we know it's not new? It's because, when a defect is found, it affects all previous versions of Windows. Think about it. We always see that. Every problem that is found affects every previous version. Well, if it was a new operating system, there would be a disconnection between the problems. There is no disconnection between the vulnerabilities because it's all the same single piece of code. The other way you know is, if you drill down a few levels in the UI, you find dialogues that haven't changed in 20 years because the code's fine, they haven't screwed it up, and they've left it alone.

So what we have basically is NT with about nine layers of different sugarcoating UI wrapped around it. They keep changing the way it looks. Oh, let's go 32-bit colors. Oh, let's have shadows. Now let's do 3D. Oh, we don't like that anymore. Now let's make it flat. They're changing the surface, but it is all a single old operating system. And I'm happy that it's old because new code just has problems. So the point is we know it's the same old operating system because every problem we talk about affects all the versions, all the way back, until they stop supporting them. So it's monolithic. It looks different, but it's not different.

So anyway, I just - I wanted to make a couple of those little points, and that is that security features does not mean more secure. It actually means less secure, at least initially. And it's not a new operating system. It's the same old operating system because all the problems are shared with Windows XP and 7 and as far back as it goes. And we'll

be talking about a few of those today because, again, same operating system, different coating on the top.

And speak of the devil, first item here of news is that it turns out that all currently supported versions of Windows suffer from a newly discovered zero-day exploit.

Leo: Oh, that's ironic.

Steve: Oh.

Leo: Because they didn't change it in Windows 10.

Steve: Right. It's the same operating system. It just has a different UI on top. So a Russian forum was found offering - some guy, I love the name of this guy, BuggiCorp, B-U-G-G-I Corp, is offering a pervasive Windows zero-day exploit for which he wants $95,000. Now, the good news is - well, good is relative.

Leo: That's a lot of money.

Steve: But at least it's a local privilege escalation. It's not a remote code execution, which would be chilling.

Leo: It's worth $95,000? Wow.

Steve: Well, yeah. And in fact he goes to some lengths to demonstrate the veracity of his claim. So, translated from the Russian, he wrote: "Exploit for local privilege escalation (LPE) for a zero-day vulnerability in [a module of Windows] win32k.sys. The vulnerability exists in the incorrect handling of window objects, which have certain properties. The vulnerability exists in all OS versions, starting from Windows 2000."

Leo: Wow.

Steve: So like I said, same piece of code, win32k.sys. We have it today, and we had it back 16 years ago. "The exploit is implemented for all OS architectures, x86 and x64, starting from XP, including Windows Server versions, and up to current..."

Leo: To be clear, that's the 32-bit compatibility layer for Windows 10; right? I mean, it's not - because Windows 10 is 64-bit. I would guess that's the 32-bit compatibility there.

Steve: Correct. Right, right, right.

Leo: Yeah. So that would be in there, of course.

Steve: Yeah, exactly. And it's win32k.sys, so it's the portion that they've had all along to run the 32-bit code under Windows. So they said: "...up to the current variants of Windows 10. The vulnerability is of 'writewhatwhere' type," he writes, "and as such allows one to write a certain value to any address in memory, which is sufficient for a full exploit."

Leo: Oh, yeah.

Steve: Yeah. "The exploit successfully escapes from application containment" - meaning it breaks out of any sandboxes - and bypasses "all existing protection mechanisms such as ASLR, DEP, SMEP, et cetera. The exploit relies solely on the KERNEL32 and USER32 libraries." So again, 32-bit. "The source code project of the exploit and a demo example are written in C and assembly with Microsoft Visual C 2005," because that's what they have in Russia. "The output is a lib file which can later be linked to any other code" - just to make it convenient for the malware authors - "and additional output from the source code project is a demo EXE file which launches" - oh, so you have the demo, a demo EXE - "which launches CMD EXE and escalates the privileges to the system account." And he goes on.

So anyway, the point is that this was found by the Trustwave group, and they said that: "Although such an exploit cannot provide the initial infection vector like a remote code execution would, it is still a very much needed puzzle piece in the overall infection process. For instance, an LPE [local privilege escalation] exploit paired with a client-side remote code execution could allow an attacker to escape an application that implements sandbox protection, like Google Chrome, Adobe Reader, et cetera."

So what was rare was to find this zero-day exploit offered for sale. Typically that's not done. There's, like, other exploits and malware pieces and things, but not something this significant. So they wrote: "Finding a zero-day listed in between fairly common offerings is definitely an anomaly. It goes to show that zero-days are coming out of the shadows and are fast becoming a commodity for the masses," they wrote, "a worrying trend indeed."

So that got a lot of press coverage, too, because people are like, oh, no, it was zero-day. But it's not, again, this is not being found in the wild yet, which is normally where we encounter and talk about zero-days. This instead is like pre-wild. This is "Who wants a zero-day," and then get one.

Okay. So this is just chilling. Andy Greenberg writing for Wired covered it, and I'll just read the first two paragraphs of what he said, then talk about this in more detail. He said: "Security flaws in software can be tough to find. Purposefully planted ones - hidden backdoors created by spies or saboteurs - are often even stealthier. Now imagine a backdoor planted, not in an application, or deep in an operating system, but even deeper, in the hardware of the processor that runs a computer. And now imagine that silicon backdoor is invisible, not only to the computer's software, but even to the chip's designer, who has no idea that it was added by the chip's manufacturer, likely in some," he writes, "far-flung Chinese factory. And that it's a single component hidden among hundreds of millions or billions" - on the chip, he meant - "and that each one of those components is less than a thousandth of the width of a human hair.

"In fact, researchers at the University of Michigan haven't just imagined that computer security nightmare? they've built and proved it works. In a study that won the Best Paper Award at last week's IEEE Symposium on Privacy and Security, they detailed the creation of an insidious, microscopic hardware backdoor proof-of-concept. And they showed that, by running a series of seemingly innocuous commands on their minutely sabotaged processor, a hacker could reliably trigger a feature of the chip that gives them full access to the operating system. Most disturbingly, they write, that microscopic hardware backdoor would not be caught by practically any modern method of hardware security analysis and could be planted by a single employee of a chip factory."

And I agree with everything that they wrote, having dug into this. They named it A2 because it's an analog attack. And what it uses, although they didn't call it this, in electronic terminology it's known as a "charge pump." So what's tricky about this, what's so clever - and, I mean, it's like this is the stuff of nightmares for anyone who's concerned about fabricating secure processors - is that we think of a computer processor as a massive array of interconnected gates, which is deterministic. That's the key. If it is a massive array of interconnected gates, then it starts from power up in a known state, and it executes instructions, each instruction of which moves it from one known state to the next known state, based on what that instruction does. That's a programmed instruction computer.

What these guys did was they added - it was described in the documentation as a capacitor - probably a capacitor and a couple diodes. But, I mean, that's just vanishingly small in terms of the real estate of this chip. I'm sure everyone's sort of familiar, having seen photomicrographs of what a processor looks like these days. You just kind of - your eyes glaze over dealing with this incredible complexity of what modern chips have become.

So what they did, they took an open source processor design. I don't remember, it's like OS 1200, I think, stands for open source 1200, or open core maybe it was, OC 1200. But there is, there exists now open core designs where it's all laid out, and you can have it fabbed, or run it through programmable arrays and get yourself a nice, high-performance, state-of-the-art RISC processor. So they took the silicon fab for that, and they added just a couple microscopic components such that, every time one of the signal lines in the chip is raised, it couples a little bit of the energy of the edge of that raising through this capacitor into an adjacent line. And the problem is, that isn't enough to change the state of the coupled-to line unless you do it with a sufficient frequency for a sufficient time.

So it's a charge pump. It pumps charge from one line to another in a way that no static analysis of the function of the chip could ever find it because it uses basically an analog-coupled pattern of access in order to change the state of an adjacent line. And where they chose in the design essentially removed the memory management restrictions from the program that was currently running so that any program that was running with normal OS hardware-enforced containment, which is what all of our programs have now, when the kernel decides to run the thread in a process, it sets up access to memory so that the process has a very filtered view into the actual physical memory that is available on the computer. So that the process running sort of sees - it's called a map. It's through the memory mapper. It sort of sees an abstraction of the actual physical memory.

What this then allows is a process running on this chip that knows about this tiny defect that was added to the design to release itself from all OS constraints so that the mapper is shut down, and it has access to all the memory on the system. It's basically a behavior-based switch that was added that would not be found by any diagnostics running on the processor. And what's horrifying is that there's really no defense against

this. The only thing you could do would be to apply the highest level of security throughout the entire design process, including all the way to and through fabrication.

And in their paper they get a little defensive about why they did this. And it's like, well, okay, yeah. We've shown the world this could be done. But mostly it's to make sure people understand this is not just theoretical. And what they proposed was that some other kind of trusted component that would also be integrated on the chip, which doesn't exist today, would be an overseer which would independently watch to make sure that no process running on the system had obtained that kind of access. And I'm like, well, okay. That doesn't really do it. Here we're back to the problem of trying to anticipate all the things that something could do that wants to be nefarious. And you can't. That's the classic virus/antivirus, goodware/malware problem, is that they designed their little hack to change the memory mapper.

So they're saying, okay, so now we need a memory mapper overseer to guard against that. But there are all kinds of other ways you could subvert a chip maybe a little less horribly than just remove the memory mapping facility because that does just completely drop all of the defenses of the system. But there are other things you could do that are equally troublesome. So there's just nothing but bad news, essentially.

Now, there has been a concern that designs of chips could have deliberate hardware backdoors installed in them. And that's a concern. This ups the ante sort of infinitely because now it's not just the design, I mean, you could test the design after fabrication and never find that this behavior-based flaw had been inserted into the chip. So, wow. I mean, I guess I'm glad that they brought this to light, although unfortunately I don't see a practical mitigation for it. I mean, it really sort of says to a degree that maybe helps us appreciate the fact that, much as we want to have security, security is not an absolute. It is a relative. And I don't see how that changes in the future. Wow.

Yesterday Google released the June 6th or the June Android Security Bulletin, which had a bunch of goodies in it. They offered over-the-air updates to all currently supported Nexus devices. And it flashed by me, and I think I want to say, like, Nexus maybe 6, 7, and 8? I think I remember seeing, like, three major version numbers, maybe 5, 6, 7? I just don't know. But anyway, it's the ones that are currently receiving updates. They all got them.

Leo: Nexus 5. Let's see. I have a 6P, so I think probably the Nexus 5 and 6P and the 5X.

Steve: Okay, good. They did release these to their partners a month ago, on May 2nd, so all of the carriers got them who want to push these out to their customers. Google wrote that "The most severe issue is a critical security vulnerability that could enable remote code execution on an affected device through multiple methods such as email, web browsing, and [wait for it] MMS." And we know why. The Mediaserver module is back in the hot seat.

There is a critical Mediaserver flaw which exposes Android devices to remote code execution such that an attacker could send vulnerable devices a malicious media file that corrupts memory during the processing, the interpretation of that file. And because Mediaserver runs at the kernel level with system kernel-level privileges, the resulting code would essentially be in the Mediaserver, so it can do pretty much anything it wants to. So an important update to follow up on.

And that was one of a dozen other issues in Mediaserver. There were a bunch of, I think it was 12 other elevation-of-privilege issues which could be exploited by a local app. So not a remote "send you a media file and you're taken over," but rather inadvertently run a local app on the device, and it could then - the app could use it to gain system-level privileges, which would let it break out of the app sandbox, app containment. Also there is a problem with the WebM encoder, and that is a critical remote code execution flaw.

So once again, you know, we talked a few weeks ago about sort of the general problem, like why are we having problems with these media display playback things? And it's because they are interpreters by nature. Essentially, the files are little programs. The nature of the metadata in these files makes them little programs which the code interprets. And it is very difficult to make that bulletproof. You just have to check every single parameter perfectly and not make any mistakes with overflows or signed versus unsigned or is this a byte or a word or a double word and so forth. It's just difficult to make that perfect. And as we know, it's got to be perfect, or somebody can find a mistake. So a bunch of things.

And not to be forgotten - and actually for some reason Qualcomm was in the hot seat, although a lot of these problems were not Qualcomm's. But Qualcomm's drivers that exist in these devices, there were five problems in the WiFi driver, two each in the video and sound driver, one in the GPU driver, and one in the camera driver. So there were just a bunch of things to get fixed. And so follow up on that if you're an Android user, and you didn't automatically get these things over the air.

I skipped a story last week, just because I thought, really? The senior editor of PC World, Brad, and I don't know how to pronounce his last name, C-H-A-C-O-S.

Leo: Chacos?

Steve: Chacos? Yeah. We'll say - and sorry, Brad, if that's not how you pronounce it.

Leo: Because I'm making it up, too.

Steve: He did a story which was titled "Fearing forced Windows 10 upgrades, users are disabling critical updates instead." And I thought, really? But then two days ago I got email that routed itself either through Sue or Greg and found its way to me, from someone who was a little desperate. I think her name was Victoria. She had a Dell Vostro, and she was concerned that it was so old that it wouldn't run Windows 10. So she had turned off Windows Update and had 117 pending updates and was writing for, like, okay, how do I use Never10 to, like, make sure that if I, like, shouldn't I have these turned back on? And I said yeah, probably a good idea to keep your Windows updated.

And so I said, "Run Never10, then reboot your machine to make sure that it takes, then run it again to make sure that it still says that Windows updating is disabled, then turn on Windows Update and suck in those 117 pending updates." I said, "Then I'm sure it'll ask you to reboot. So then reboot and run Never10 just once more to make sure that, after all those updates, everything's still happy, and Windows 10 updating is still disabled, and you should be good to go."

So I did want to mention that I mentioned last week the crazy rate of uptake of Never10. I think I said we were, like, at 33 or 32,000 per day. We peaked over the weekend at

35,000 downloads per day. And I said that at that rate last week, we would cross a million. We did that. I said we would cross million by this podcast today. We did that at 10:00 a.m. this morning.

Leo: Yeah.

Steve: More than a million downloads of Never10.

Leo: That's awesome.

Steve: So, and it's, let's see, where are we, 1,005,295 at the moment. And it is slowing down a little bit. We're now down to about 33,000 downloads per day. But it did, it's interesting, as Microsoft sort of turned up the heat, Never10 took off. So as I said last week, I was glad that it had a chance to be in place so that people were aware of it, so that it could provide a solution for people who wanted the opportunity, the option to choose not to upgrade to Windows 10.

The only reason this is interesting, I think, is because Mark - oh. I forgot to mute that little laptop.

Leo: [Crosstalk] leave it going.

Steve: So Mark Zuckerberg was in the news because apparently he had an easy-to-hack Twitter password from 2012 because Mark has, you know, I guess he's now over on Facebook, not surprisingly. He hadn't apparently tweeted, his last tweet in Twitter was in January of 2012. But he apparently shared his Twitter password with his LinkedIn account. And a group calling themselves OurMine Team claim to have used the data from that massive LinkedIn breach, which we talked about last week, the additional 117 million accounts, to find Zuckerberg's password and see if it worked anywhere else. The password was "dadada."

Leo: No. How do we know that?

Steve: Well, because it's an unsalted...

Leo: They said it?

Steve: Yeah. Everyone knows it now.

Leo: Dadada.

Steve: Yeah, dadada. Dadada.

Leo: That's a terrible password.

Steve: Yeah, it wasn't even - and only four years ago it wasn't very good.

Leo: Well, but he doesn't care because these are throwaway accounts.

Steve: Well, yes. The problem was he still had his LinkedIn account, I mean, sorry, he still had his Twitter account, and they took it over.

Leo: He shouldn't have used the same, I mean, I presume he didn't tweet a lot. Maybe he did, I don't know.

Steve: Well, no. He hasn't used that account for four years because now he's Mr. Facebook.

Leo: Right. He always has been, right.

Steve: And so he's not going to use Twitter. He's going to use...

Leo: It was to collect information about Twitter, not to use it, not to promote it. I remember I followed him on Google+, which didn't get hacked. I guess he didn't use dadadada.

Steve: Correct. Google+ was not hacked.

Leo: He never posted there.

Steve: His Pinterest account apparently was also the same password. So he was using it in several places.

Leo: Dadadada.

Steve: Yeah. There was some information that his Instagram account also, but that turned out not to be the case. So just Twitter and Pinterest.

Leo: What a terrible password.

Steve: Also in Facebook news, they have stated that, as...

[Begin clip]

Leo: Dadadada. I'm sorry. I think this is where it comes from. It was a hit called "Dadadada." Either that, or it's a very easy thing to type on the keyboard.

Steve: It's also easy to type.

Leo: Yeah, dadadada.

[End clip]

Leo: Okay, enough of that. I couldn't resist.

Steve: So Facebook Messenger, as we know, used by about 900 million people, will be getting optional, opt-in, end-to-end encryption. Of course this follows Google's recent announcement of the same in Allo, their forthcoming messaging app. And as with Allo, Facebook will have this be opt-in because its use would blind Facebook to Messenger's content and prevent the use of their planned machine learning features, which is the big new idea these days and what everyone wants to roll out. And I also heard the word - I've heard you, like, not really being happy with bots, Leo. And apparently this is, like, would be a problem for their bots.

And of course the Guardian in their coverage noted that last month Google faced some blowback from privacy activists over their opt-in decision. But as I said at the time, I think this is entirely appropriate. And it'll be interesting to see, once we get some numbers, what percentage of users choose which. Because I think we're going to see - remember the studies that have been shown, how people will sell their passwords for, like, a piece of candy or something, I mean, it's just ridiculous. It's like, here's a piece of candy. What's your password? Oh, thanks, and then give up their password.

Leo: Yeah, why not?

Steve: Similarly, I think, if someone turns on encryption thinking, oh, good, and then sees that they broke a lot of other nonsense that they like, they'll go, oh, bad, and turn it off again because they would rather have these messengers suggest their replies to their conversations or whatever. So it's like, okay. So I think it's fine. I like that people have an option. And as we said, if anyone really wants security - because, remember, these apparently are per-conversation decisions. You don't just switch all of Messenger to encrypted. Presumably you say I want to have a secure connection with this other person, but leave all the other ones unencrypted. So as we said, if you really want security, just get a secure messaging app. They exist, and that's a no-brainer. And then you know that your conversations there will be secure.

And this is not security related, but I just thought it was interesting from a technology standpoint. And that's The New York Times had a story about the five things that give self-driving cars headaches. And the author who did a lot of research and rode around said: "Fully automated cars don't drink and drive. They don't fall asleep at the wheel.

They don't text. They don't talk on the phone or put on makeup while driving. With their sensors and processors, they navigate roads without any of these human failings that can result in accidents.

"But there is something self-driving cars do not yet deal with very well - the unexpected. The human brain is still better than any computer at making decisions in the face of sudden, unforeseen events on the road - a child running into the street, a swerving cyclist, or a fallen tree limb. Here," he writes, "are five situations that, for now at least, confound self-driving cars and the engineers working on them."

And I have to say, as an oldster, I'm sort of bemused by this. I'm just totally gobsmacked by the speed with which this has happened because I'm just amazed that we're moving this quickly into self-driving cars. I just - I'm, again, I guess I'm showing my age. It's like, how big is your hard drive? Really? So I'm impressed that they do as well as they do.

And I guess the thing that brings me over a little more to the side of self-driving cars is now watching how distracted drivers must be by their smartphones. I see cars weaving on the street, you know, crossing, just wandering out of their lane, across the line. And of course the dead giveaway is when the lights turn green, and the cars just sit there at the light. No one's moving because they've taken that opportunity to check in with Facebook.

Leo: My daughter got a ticket for that exact thing. It wasn't that she drove into anything, she just didn't go when the light was green because she was typing, and she got a ticket.

Steve: Who?

Leo: My daughter, Abby.

Steve: No kidding. Well, whoops.

Leo: Not recently.

Steve: Well, yeah.

Leo: A year ago. But I was glad. I was thrilled she got that ticket.

Steve: Well, and so I see it all the time. And so I'm just thinking, okay, maybe this is not such a bad idea. But anyway, so five things. First is unpredictable humans, i.e., bad drivers. And it's like a self-driving car cannot control the behavior of other drivers. And so it's just - that's an issue. They do the best job they can, but it's not like - that's something aside from their ability to look at the road and know where they are and use their sensors and so forth.

Another problem is bad weather, a.k.a., where did the lines on the road go? And, I mean,

think about it. We, with all of our human senses, rain, snow, fog, really, if you're in a heavy rain, everybody is going a lot slower because it's just much more difficult to drive safely under those conditions. So how do you feel for some poor algorithm trying to make sense of those conditions.

Leo: Poor algorithm.

Steve: With all of its different sensors. Yeah, I can just, I mean, I just imagine. Also, detours and rerouted roads. And the article brought up something I didn't realize, but in retrospect it's so obvious. This author said that "Google's bubble-shaped self-driving cars rely heavily on highly detailed three-dimensional maps, far more detailed than those in Google Maps, that communicate the location of intersections, stop signs, onramps, and buildings with the cars' computer systems. Self-driving cars combine these pre-created maps with readings from their sensors to find their way around."

And it's like, it's so obvious after you think about it. It's like, yes. What you'd want to do is you would want to create detailed useful 3D models of what doesn't change, that is, where the roads are, where the signage is, all of those things that are fixed, so that the car is not encountering them for the first time, just with its own onboard sensors, but rather the sensors are positioning it in real-time within a pre-existing, highly detailed 3D model. That's what you want to do.

The problem is what happens when that model breaks, when things change from that model, because the better that model is, the more the car will tend to rely on the knowledge that is essentially embedded in that model. And to the degree that that knowledge is suddenly incorrect because of something in the real world changing, that causes a problem also. So there has to be sort of a resilience to it where it uses the model, but there's a constant check on how much it feels this model represents what its sensors are actually showing it. And it's able then to pull back from the model when reality starts to diverge. So just really interesting, from a design standpoint, decisions to make.

And then the second to last is road discolorations. Is it a pothole or a shadow? And of course we know that self-driving cars use radar, lasers, and high-definition cameras to scan the roads for obstacles, and the images that those sensors generate are processed by high-power processors to identify pedestrians, cyclists, and other vehicles. But it turns out that potholes are tough. They lie below the road's surface, not above. And a dark patch in the road could look like a pothole or be an oil spot or a puddle, or even a filled-in pothole. So it turns out that's a challenge for the designers of these.

And then the last one sort of took me a little bit aback because I hadn't thought about that. And that is ethics, believe it or not, having to make tough decisions. And this writer wrote: "In the midst of busy traffic, a ball bounces into the road, pursued by two running children. If a self-driving car's only options are to hit the children or veer right and strike a phone pole, potentially injuring or killing the car's occupants, what does it do? Should its computer give priority to the pedestrians or the passengers?" And it's like, yikes. I mean, it's the Kobayashi Maru, is what this is.

Leo: It's a no-win scenario.

Steve: It's a no-win scenario. And, wow. I hadn't thought about it. But, yeah, humans,

we're driving and have to make a similar sort of decision. But imagine having to embed ethics into silicon.

Leo: You're right.

Steve: I hadn't thought about it before. Interesting. Okay. So something cool. Many people - and I'm impressed, actually, by how many people picked up on this and wrote. And in my show notes I have two tweets from people who tweeted the question. But just at - I don't know what time zone that is. But I just received a piece of email from someone who said it really well. His subject was "Rekeying DOS," as he called it, or DoS as in denial of service.

He said: "I read SN-562, where you talked about rekeying SQRL identities if their master key is compromised. If my key is compromised, what prevents the attacker from rekeying my identity on sites, effectively locking me out of those sites? I might not know I've been compromised, and the attacker can start with popular social, bank, and email sites to guess if I have an account there, and then lock me out. Without password recovery or something like that, I've had my identity stolen permanently. Just want to hear your thoughts on this."

So everybody will be glad to know we have that completely covered in a really cool way. There's another part of SQRL which came much later. It was the result of heated deliberations, to put it politically correctly, in the SQRL newsgroup about this problem. And so I went off one morning to Starbucks with my engineering pad and pencil and big eraser, and thought, okay, I've got to come up with a solution for this. And most of the time was spent staring off into space. But I always, you know, I drew some circles with some arrows pointing at them, just sort of had an anchor for my thoughts. And I ended up with what I think is a true invention because I've never seen it before or since.

It is a new way, a sort of repurposing Diffie-Hellman key agreement, the idea being that two people would create a private and a public key, and they would exchange their public keys. And then by having their private key and the other person's public key, they're able to each synthesize the same result. And that's done twice in what we have called now the Identity Lock Protocol. And it's like, even when I had figured it out, I still didn't understand it. I mean, it's one of those things where the equations prove it works, but I don't quite know why. But it does work. I've written it. I've implemented it. We've tested it. Other people have duplicated that work. It's all documented on the site and on the Identity Lock Protocol page.

But here's what it does. We call the act of creating a SQRL login on a site "association." That is, you associate your SQRL identity with an account. So like you'd already have an account on Amazon or on Google or wherever, and that site allows you to log in with SQRL. So you go there, and you say, hey, I want to add my SQRL identity here so that I can use that for logging into the site. So we call that "associating your identity."

When that is done, that is, the act of associating, in addition to the SQRL public key, which we know is how the site knows you, and what the site uses for verifying your signed challenge when it subsequently challenges you to prove you are who you say you are, two additional pieces of information are given to the site to hold, which is how this identity lock works. What happens in practice is that you can create an association, the SQRL identity association, with your client. But you cannot change it without something additional that the client never contains.

So the invention part of this was the ability for the client to provide something unique that involves a large random number. Again, remember that linkability was something SQRL had to have so that we're giving a different - every site gets a different identity. I needed this identity lock to similarly be different for every site so there's no linkability between sites. So when you associate your identity to a site, you give it a couple other blobs to hold onto. And if you want to change your identity, if you rekey your identity, you must, in order to do that, provide a higher level of authentication.

And that's what we have with the so-called "rescue code." The rescue code is never meant to be memorized. It is never stored in a SQRL client. It's a deliberately long, completely random number. The user can't choose it, specifically because users choose "dadada" if you give them a choice. So this is 20 completely random digits that is created just once when you create your identity. You print it out, you fold it up, and you put it somewhere secure. This is the way we solve the password recovery problem. That is, if somehow you forget your password, that big, random, you may never need it in your entire life, but it's there, that's password recovery.

But it also does a couple other things. It's the thing that you must provide when you rekey your identity. So if a bad guy completely compromised your SQRL client and got everything it knew, that bad guy doesn't get the rescue code because it is never - it's not put on the clipboard. It's never stored. It's never in the identity or in the client. It just briefly exists and is embedded in the identity. Actually, the identity is encrypted under a variant of that, and you may never need it. But you can use it, if all else is lost, to recover your identity, to recover your password, but you must provide it as part of rekeying.

And there's one more little piece, which the person who wrote this sort of alluded to. What if a bad guy went somewhere and tried to rekey your identity? Well, first of all, now we know they can't. Only you can rekey your identity by using the rescue code. But the other cool thing in this design is we allow anyone without the rescue code to disable logon with SQRL at any site, without the rescue code.

So in this scenario, say some foreign government you believe got a hold of your identity, or a bad guy, whatever. So just as this guy was proposing in his email, someone could go to popular sites. You could, if you believed this was a problem, immediately go to a site with your SQRL identity and disable authentication without the rescue code. And so you would go, if you were in trouble and did not have your rescue code with you, so you were unable to rekey your identity right now, what you can do is disable authentication. And only a client carrying the rescue code is able to subsequently reenable authentication. So we made that asymmetric. You, using your SQRL client, can disable authentication with no additional authentication, but you cannot reenable it without the additional rescue code.

So anyway, as I said last week, what has evolved is a really beautiful, complete solution. And all of the problems with the font printing and stuff that I talked about last week is resolved. It's running very nicely now, printing under Wine. And I'm making a few changes to the UI, and then it's just nailing down a bunch of stuff from sort of my longstanding list of things I want to fix, and we're done. So we're getting close.

Leo: Yay. Steve is fully hydrated, caffeinated, addlepated, and ready to go.

Steve: So I've been meaning to say for a couple weeks, just to acknowledge what a win the little recommendation of The Sequence puzzle toy game...

Leo: Oh, we love that game, man. That is such a good game.

Steve: And in fact it is, absolutely hands-down, based on feedback, the all-time winner of all of the puzzle stuff that I have recommended in the past. And what I wanted to note was that it is - I'm in Level 40-something, and there are people, like, way past me. But I've never used a hint, and I don't want to use a hint. I just...

Leo: Wow.

Steve: Sometimes it takes me three days.

Leo: It's hard. I'm impressed.

Steve: It is hard. There are some which are just - and though for me it's that aha moment. It's like, oh. The one thing I wish it had is single-step. I was tweeting this to somebody.

Leo: Right. Debugging. You want debugging.

Steve: Yes. And they tweeted back, oh, you mean like a debugger? It's like, yeah. Because you start it, and it just goes. And it's like, wait, wait, wait. Oh, wait, wait, wait. Wait, I couldn't see what happened. So, and I've been meaning to contact the author. But anyway, I just wanted to acknowledge what this is, the reason we love it, it is programming.

Leo: It's programming, absolutely.

Steve: It is an elegant, visual programming environment that has every facet of programming because you have a goal, which is to move this little puck from location A to location B. And you have a set of tools, and they're like instructions. They each do one thing. And you need to interconnect them, that is, have them do their one thing in sequence, and that's why it's called The Sequence, in order to achieve this result. That's programming. And it just - it's so well implemented and so charming. I'm delighted that it's available cross-platform. So I just hadn't had a chance to come back around and mention it again after first talking about it.

And people have talked to me about the Human Resource Machine, which is another - it's more of an explicit list of actions programming toy. I don't know if the Human Resource Machine is over on Android. I know that it's on iOS. I downloaded it, but it's in the queue because I'm going to take my time, and I'm going to work my way through all of The Sequence, no matter how long it takes. I'm not in a hurry. And I don't do it - I don't spend all day. I'll pick it up. And what's interesting is that I can, like on the third day I'll pick it up, and I'll solve it, even though I spent 15 minutes on two previous days staring at it, and it didn't come to me. Then it's like, okay, it's time. And I solved it.

Leo: Wow.

Steve: And one other bit of miscellany is people will remember this really beautiful replica that an engineer named Oscar created of the PDP-8/I console. What's behind me blinking are PDP-8/E emulators based on an Intersil chip, the 6100, that actually was a PDP-8 on a chip. What Oscar created essentially was a switches-and-lights peripheral for the Pi. Why can't I think of the first name, the Something Pi. What's the name of that thing? The little circuit board Pi. I can't think of it.

Anyway, the 2015 version we talked about. I just wanted to bring people up-to-date that there's a 2016 version. Raspberry, that's the first word, the Raspberry Pi. So this is - and it's got much more lights. The PDP-8/I shows you all of the various registers, whereas the PDP - and it was an earlier version of the PDP-8. They went for reducing the cost, and that's why you can see there's a knob there that allows you to choose which one of the many registers the lights will show. Well, those are broken out on the PDP-8/I. So it's cool because there's just lots more lights to blink.

Anyway, so what surprises me is that he's offering this kit for $145. But in 2016, he's exactly duplicated the rocker switches. He had some close ones on the original 2015 design. Now they're pre-painted, beautiful rocker switches that are exactly the same shape as the original PDP-8/I. And this kit's $145 as a peripheral to the Raspberry Pi, and all the software is there and free and available. So I just wanted to bring it up again because I think it's, for someone who wants to play - and this of course ties into programming because the PDP-8 had a 3-bit opcode. It had eight instructions. One of them was microprogrammed, that is, there were lots of bits that did different things in the arithmetic instruction. But otherwise there was, like, jump and add and store and not many.

And so it's entirely practical to write little PDP-8 programs on a pencil and paper, very much like The Sequence. Anyway, I created a bit.ly link long ago. It's still functional, and it's bit.ly/pdp8kit - numeral 8 - pdp8kit. And so, if anyone missed that the first time or is interested in looking at what Oscar did for 2016, I wanted to make mention of it.

And I'm self-conscious now about talking about the Healthy Sleep Formula. I don't want to bore people who don't care. I should mention it is the second, next to Never10, most popular page on GRC, about 2,600 visits per day. So all I'm going to say is that there's been some changes. Everybody who cares please go check out the Healthy Sleep Formula page. I have updated it with a bunch of big news, major news.

Leo: We should emphasize that Steve makes no money off of this. He's not selling vitamins. It kind of sounds like he is, but he's not. He's just doing this for you.

Steve: No, and in fact people have wanted me to have links in there that would generate some revenue for me, and I've refused because I don't want there to be any appearance of impropriety, or like I'm pushing this. No, this is me sharing the result of my own exploration. So anyway.

Leo: Good job. And by the way, Steve's not a physician. You should consult your physician before taking supplements of any kind.

Steve: Right.

Leo: Your mileage may vary.

Steve: Right. Josh in Bartow, Florida. I got a really nice note from him which I really appreciated. He said: "Long-time listener finally uses SpinRite to recover data." So he was referring to himself in the third person in the subject line. He said: "Hi, Steve. I'm a longtime listener to the podcast. I appreciate the work you do each week to keep us safe and informed. Two years ago, I purchased a copy of SpinRite. Like many other listeners, I bought a copy just to support you and everything you do for us without asking for anything in return. I've used SpinRite in maintenance mode a couple of times, but never tried to recover data. Well, until yesterday, that is.

"A friend of mine contacted me for help because his computer was acting 'weird.' He had rebooted the machine, and suddenly it was asking him to 'Insert Boot Media.'" That's never good. "The BIOS didn't recognize his hard drive, and Windows would not load. He did have a partial backup on an external hard drive? however, it had been a couple of years since the last time he backed everything up. He has two small children, and he was worried that he had lost a lot of precious and irreplaceable photos, along with lots of other files.

"I loaned him my copy of SpinRite, recommending that he run it on Level 2. This evening, he sent me another message. SpinRite fixed the problem. Windows booted normally, and all of his files were intact. My friend is now hard at work backing up all of his files to his external hard drive. I told him to get Carbonite (using the SECURITYNOW offer code, of course). I also told him to buy a copy of SpinRite for himself so he can maintain the drive. He's planning to do both. Thanks again for making such a great podcast and such a great piece of software. Josh."

Leo: That's nice.

Steve: And Josh, thanks for sharing your experience.

Leo: All right. So let's get to the baby monitors.

Steve: Okay, so - oh, boy.

Leo: Which actually is probably fundamentally similar technology; isn't it? I mean, it's just a - it's a camera and a speaker, a microphone.

Steve: Well, okay. So the problem is - and believe me, the Ring guys are light years ahead of these products because the problem here is that you're pushing your cart through Target, and there on the J-hook is the iBaby M6. And it's like, oh, look. It's only $39.95 on sale. Let's get one. And so it's like, no one's ever heard of iBaby. Someone is surprised that Apple lets them sell "i" anything else. So it's like, oh, great. And so they, for a Sunday project, Dad mounts this on the bookcase and plugs it in and then goes to iBaby.com and creates an account, whatever. And it's like, oh, look, we're able to look at

our baby from anywhere in the world.

So the point is this is what's happening, and there is no concept of security. And of course iBaby has, right there on the label, "secure, state-of-the-art, military-grade encryption." So it's like, oh, well, fine, then. It's encrypted. Must be great. Right? No.

Anyway, so Rapid7, in choosing baby monitors as their target, sort of to set the groundwork for this, they said: "The research presented focuses on the security of retail video baby monitors for a number of reasons. Baby monitors fulfill an intensely personal use case for IoT. They are usually placed near infants and toddlers, are intended to bring peace of mind to new parents, and are marketed as safety devices. By being Internet-accessible, they also help connect distant family members with their newest nieces, nephews, and grandchildren, as well as allow parents to check in on their kids while away from home. They are also largely commodity devices, built from general purpose components, using chipsets, firmware, and software found in many other IoT devices.

"Video baby monitors make ideal candidates for security exploration. Not only are they positioned as safety and security devices, and therefore should be held to a reasonably high standard for security; but the techniques used in discovering these findings are easily transferable to plenty of other areas of interest. Other products of direct interest to commercial and industrial consumers and security researchers - like commercial security systems, home automation systems, onpremise climate control systems and so forth - share many of the insecure design and deployment issues found in baby monitors."

So Rapid7 took 10. And in their notes, one of the things that was a little chilling was that they endeavored to be responsible with security disclosure. But in some cases there was no one to contact. They couldn't find, like, registered domains for some of these companies. There was one that was using some other proxy that didn't belong to the company, and they couldn't find the company. So despite the fact that it was, like, amazingly insecure, there was no one to tell. There was, like, no contact information, even on the packaging. It was just generic, and horrifically insecure. So, I mean, just shudder-worthy.

So here's an example, and it happens to be iBaby Labs, Inc., with their iBaby M6. So the website, iBabyCloud.com, has a vulnerability by which any authenticated user to the iBabyCloud.com service - get this - is able to view the camera information of any other iBaby user, including video recordings, due to what Rapid7 called a "direct object reference vulnerability." The object ID parameter is eight hex characters, corresponding to the serial number of the device. This small object ID space enables, obviously, a trivial enumeration attack, where an attacker can simply brute-force the object IDs of all cameras.

Once an attacker is able to view an account's details, then links that are available provide a filename that shows the available, as they call them, "alert" videos that the camera has recorded. And using a specific AWS CloudFront endpoint, which they found via sniffing the iOS app functionality, this URL can harvest the filename which allows the camera's contents to be viewed. So effectively, anyone is able to view videos that were created from that camera, stored on the iBaby cloud service, until those videos are deleted, without any further authentication. So essentially you sign yourself up to iBaby cloud service. And you then start simply - there's a serial number which is eight hex digits. And so you simply change, run through those, and you're then able to access the entire install base of iBaby video streams. I mean, like, that's all there is to it. There's no authentication on a per-account basis required.

There's another product that iBaby makes called the M3S. Now, it runs an open Telnet

server with fixed login credentials of admin for the user and admin for the password. Now, that's not a huge problem because it's behind - we assume it's going to be behind the residence's NAT router. So that would mean that its IP was not public, as long as the IP and port were not publicly mapped through the router. Of course, Universal Plug & Play does allow anything inside the network to map the device through to the public Internet.

But we'll come back to this because it turns out that the majority of these devices all had locally exposed Telnet and/or web servers that were statically enabled. And that's my problem with this, is that it would be one thing if there was a button you pressed, and it brought the server up which you needed for configuration, for example, for 15 minutes, and then shut it down. And if that wasn't long enough, press the button again, and you back up, and you keep trying. But they don't. They saved on a button. They just haven't - the server just comes up, and it's running, and it's there. Then it's admin/admin, or it's user/user.

And while the good news is we're behind a router, the problem is, if this trend continues, if IoT devices individually assume that they're secure because they're hiding behind a NAT router, what happens is it creates a massive target-rich environment for anything that gets into the local network. Remember that any software running on any computer, also in the network, has scanning access to everything on the local network. And to the degree that IoT devices just say, eh, we're going to leave this server open because, after all, it must be behind a NAT, well, yeah.

But if there's 20 different things that are all relying on that privacy, then it creates a huge opportunity tension for bad guys to arrange to get onto the LAN, or apps that have a little extra functionality that wasn't documented when you downloaded it to take a look around on the LAN and see what's available. And of course they're able, unless the user has disabled Universal Plug & Play, they're able to find things that are locally available and map them out to the public Internet using Universal Plug & Play, which typically doesn't show up in the router's UI, and that that allows them to gain public access to those assets behind the NAT router. So, I mean, this is not good.

Philips, the electronics company, Philips Electronics N.V., they've got something called the In.Sight B120/37. Like these others, it runs an open Telnet and web services server on its camera, with static, well known to anyone who owns one, passwords. So that's one problem. But of more concern is their web service that goes along with this, that in Rapid7's taking a look at it, turns out to contain multiple cross-site scripting vulnerabilities, which allows any valid account on that site, on that web service, to obtain access to the video streams of any other account. And then what's even of greater concern is that all of the camera streams are proxied through a public cloud provider called, and I can't pronounce this, Y-O-I-C-S. Yoics. It should be Yikes.

Leo: I think it is Yoics.

Steve: Anyway, with a public hostname and port. In their analysis, the ports appear to range from 32,000, which of course is a common programmer's default, to 39,000, so about a range of 7,000 ports from 32,000 up to 39,000. And those are tied to hostnames having a very simple pattern of proxy 1 through 14, essentially. Apparently 2 didn't show up, but 1, and then 3 through 14 dot yoics, Y-O-I-C-S, dot net (proxy[1,3-14].yoics.net). So they note, given this small and readily enumerated parameter space, that is, what, 14 or actually 13 possible hostnames from ports ranging from 32,000 to 39,0000, anyone could enumerate those. And so basically you just try to connect.

Once found - oh, and all you have to do is you test for HTTP 200 response to a connection, and you know you found one. That allows you access to the camera through the proxy. And then you have administrative privileges available without any authentication of any kind to the web scripts which are running on the little web server in the camera. And you can access a streaming URL to view the live video and audio stream accessible from the camera remotely. So again, anyone who knows anything about how this works basically has the entire install base of users of these cameras available to them.

Then there's something known as the Summer Infant Baby Zoom - when I was writing down it was "Zoom," I thought, you bet, baby, zoom right in - the Infant Baby Zoom WiFi Monitor and Internet Viewing System. First of all, the Rapid7 guys found an authentication bypass which allows for the addition - oh, boy, this one was a good one - the addition of an arbitrary account to the camera without authentication. There's a web service, MySnapCam, which supports the camera's functionality. You issue, that is, a bad guy issues an HTTP GET request to that server at swifiserv.mysnapcam.com/register, because you're saying I want to register. Then ?fn= and you provide your first name; &ln= and then your last name, "ln" for last name; &email= and then your email address.

So this is a standard URL parameter tail with name value pairs separated by ampersands that we've seen for years, that we've been using for years, that you would typically have, a simple GET style submission to a browser would generate. Oh, and then the userType=3, for whatever reason; and the userGroup ID, which you just make up. You just can enumerate through those. No authentication required. That creates an account for you in that user group. And then it sends you email to the email address you gave it, providing you with additional credentials, which you then use in order to log in and access the cameras belonging to that group. I mean, it just - it's mindboggling.

So you submit this request to the server, giving it your email address. It registers you to that account, sends additional information to the email address you provided, requiring nothing else. So no, I mean, it's like, it's not obviously insecure. But anyone taking any time, as these guys did, to, like, examine the system just sees that it offers absolutely no security.

There was something, another camera, from Lens Laboratories. I love the name. It was called the Lens Peek-a-View. It's like, yup. Multiple open web servers on their products, both web and Telnet, defaulting to user/user or guest/guest credentials. And this was one where Rapid7 was unable to find anybody to complain to, to tell them that your camera has wide-open servers. And again, several other devices out of this set of 10 all ship with statically present, always open Telnet or web servers, which the concern is, over time, if this becomes the trend, if this is the way IoT systems are, it creates a huge opportunity for anything that's able to get onto the LAN and scan the network.

Again, the absolute first order of business has got to be separate your network. Take the time to create a WiFi branch, an arm that is separate from the WiFi that everything else uses, and put your IoT stuff there. I mean, if for no other reason than, in theory, nothing can infect those devices, insecure as they are, well, except many of them reach out to the network. And what I just read, some of the things I just read does allow a remote person to get onto that device with admin rights. So I didn't run through the details; but these cameras, several of these cameras are providing active access to the LAN they are on to anyone who knows about them in the world. Which means, if that is your LAN, they're on your LAN.

So again, we've got to separate the IoT network from the network that your other machines on the network use. And really it's both ways protection. You don't want the

IoT devices to get over into your normal computer LAN. And similarly, if something mischievous got onto your computer, you'd like it not to be a able to find 20 open Telnet and web servers sitting there waiting for connections. So it makes sense from both angles to separate these onto two separate networks.

Hopefully, as I said, I called this "IoT Infancy" because I think the industry as a whole is in its infancy. But this just can't be the way we allow these systems to operate. At this point, people are just buying them innocently, from Amazon or Target or Walmart or wherever, and plugging them in, and they work. But, boy, they're just incredibly insecure.

Leo: And they're baby monitors. That's the sad side of the whole thing; right?

Steve: I know. Yeah, exactly. So, I mean, in the worst case, as has been noted, the Ring Doorbell is aiming out of your house into your yard.

Leo: At least it's your street, yeah.

Steve: I don't care if anybody sees that. But, boy, I don't want it pointing into my home, into my baby's crib.

Leo: Well, I'm sure if there were an onslaught of smart doorbells from China, there were commoditized doorbells - although when Philips makes a mistake like that, that's kind of surprising.

Steve: Yeah.

Leo: I mean, that's a big company. Geez.

Steve: Yeah. I just...

Leo: Although they may just be slapping their name on some Chinese commoditized product.

Steve: Yes, yes, yes. And this is the lesson that we have seen over and over and over. The goal is make it work. Does it work? Ship it. And it costs nothing to print "encrypted, high-security, military-grade," you know. Ink is free. And so the fact that it says that on the label means absolutely nothing.

Leo: Yeah, yeah. Ladies and gentlemen, you see why you listen every week to this show? It's an education in how to do it wrong, among other things. "You're doing it wrong" episodes continue. Steve Gibson is at GRC.com. That's where you'll find SpinRite, his bread and butter, the only thing he charges for, world's best hard drive

recovery and maintenance utility.

But there's lots of free stuff there, including, of course, this show, SQRL, the sleep stuff, everything. It's all at GRC.com. That's where you'll go to leave questions, if you have questions for next week's episode, GRC.com/feedback. The podcast has not only audio there, but also written transcripts, so you can read along while you listen. That's something unique that Steve does, and I think it's great.

You'll also find Perfect Paper Passwords, I mean, you know what, it's the kind of site you go, and you might peruse it for a while. Set aside some time. You can also find the show at our site, TWiT.tv/sn for Security Now!. And we have audio and video there. You can also subscribe in your favorite podcatcher. Doesn't matter which one you use. Just get it every week because you don't want to miss it.

And my blog, as I mentioned a couple of times, I'll mention it one more time, has scripts on it. Three different listeners wrote scripts for downloading every episode, with little modification: a Bash script, a Python script, and a PowerShell script. And that's at LeoLaporte.com, if you want to get that. Steve, fun. Thank you. Great stuff.

Steve: Yay.

Leo: Questions next week.

Steve: My pleasure. Be back with you for Q&A next week, Leo.

Leo: And don't forget you can tweet Steve @SGgrc. And he accepts DMs from anyone, so send your lengthy DMs to Steve. Have a great week, Steve.

Steve: Thanks. Bye.