Bluetooth

Description: After first catching up with a bunch of fun and interesting security and privacy news, Leo and I plow into a meaty and detailed description of the technology of Bluetooth device interconnection and its cryptographic security. A follow-on episode will cover the past hacking attacks against Bluetooth.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-280.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-280-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 280, recorded December 22, 2010: Bluetooth Security.

It's time for Security Now!, the show that covers your security, your privacy, everything you need to know to protect yourself online. And here he is, ladies and gentlemen, the star of our show, via telephone today...

Steve Gibson: Yes, not quite as online as usual.

Leo: ...because Skype is down, which probably is story number one for Security Now!. I've never heard of such a thing.

Steve: It would be nice to know what the cause was. There have been so many people lately, major sites have been down because they've been involved in one way or another with WikiLeaks, and the so-called Anonymous group have vented themselves on one organization after another that has indicated any resistance to staying faithful to that cause.

Leo: The Skype folks are blogging and tweeting and saying, yes, we know our server is down.

Steve: Oh, I'll bet they know.

Leo: Yeah. And can you imagine. I just cannot remember that ever happening before. Now, I think you and I have done one show via phone before, and maybe it was when I was in Canada or something.

Steve: In the Dark Ages, Leo.

Leo: In the dim dark ages.

Steve: Yes.

Leo: But if people will just bear with us, we'll just have phone-quality audio. If you're watching the video, we do have a video, actually a pretty good video of Steve, as good as Skype, and that is through Google and their gTalk plug-in for Gmail. So at least we've got video. And we had some problems. We were going to use it for audio, but we have some problems with the audio. And I imagine on This Week in Google we'll be doing it, as well. So, by the way, the Skype blog is now down. So that I find kind of interesting. That you wouldn't expect to be tied to the same servers that they use for Skype.

Steve: Unless it really is an attack on their network.

Leo: I'm wondering now if it might be a DDoS on their network.

Steve: Yeah. If it was on their network, it could bring down their whole infrastructure, which would be all of the various pieces that they've got.

Leo: Wow. So what is our topic of discussion this day, Steve?

Steve: Well, for a long time people have asked about Bluetooth security, what is the security protocol, technology. It's something that, well, all of us are using, probably have had the occasion to use from time to time, if nothing else. And of course some people, as laws are enacted that require that we not hold cell phones while we're driving, whether we use cell phones wirelessly with the little headsets - and of course in California there is that law now, so you see people walking around with the little Bluetooth radio stuck in their ear, blinking.

So it's finally time, here on our last episode of 2010. I thought it'd be great to cover the inner workings of the Bluetooth function and security protocol. And probably week after next we will - or week after that one because we'll have a Q&A week after next, since we're going to skip the episode through the holidays - we'll talk about the attacks that have been launched against Bluetooth, having first established the context for how all the crypto stuff works.

Leo: Well, I can't wait.

Steve: And we've got a ton of news. I want to, before I go any further, remind - or I want to, not remind, remind myself not to forget to thank everybody who posts things to me in Twitter. It's like I have this huge network of people out there surveilling what's happening in the world. And many people say, I'm sure you've already seen this several times. It's like, yes, well, in fact I have, but I'm glad not to miss it. So, I mean, this has really been useful for me, as it turns out, here toward the end of, what's it been, maybe six months that I've been experimenting with Twitter? People are out there seeing stories and saying, hey, can you explain what this means? And it's perfect because it gives me a lead to follow up. And then I figure out what it means, and I bring it to all of our listeners.

Leo: All right. I'm thrilled to get to talk about that. Let's get to the security news.

Steve: Tons.

Leo: Tons of security news. So, Steve, let's get to the news of the day, shall we?

Steve: Well, following up on - well, okay. First updates. No security updates of any sort.

Leo: Woohoo!

Steve: However, we do, yeah, we do have a new problem with IE versions 6, 7, and 8. There's a new, a so-called "use after free" problem, which is where memory that's been allocated and then released back to the system, or freed, as it's called, still has a pointer which code can maliciously use. And then when that memory gets reused, it sort of reinvokes it. So it's sort of zombie code. And initially - this has been known for a few weeks and was not believed to be exploitable. But some exploit proof-of-concept code was recently posted to the 'Net. This is remote code execution vulnerability. It's another one of the CSS, the cascading style sheet parsing problems. We've seen a number of those in IE recently. So here we are, just having done our world-breaking or record-breaking update of 40 different patches in a large number of security issues last week, and we've got another problem. So with any luck Microsoft will be fixing this as their first act of 2011.

Leo: It's amazing. By the way, for those of you just tuning in, Steve is on the phone right now with us. We do have a video of him thanks to Google Talk, but Skype is down. And by the way, Steve, Twitter is now down.

Steve: Really.

Leo: Well, I think it's related. I think people...

Steve: [Indiscernible] just an overload.

Leo: Everybody's tweeting "Skype's down," and they broke Twitter. You know, it just shows you, we're so reliant on the Internet these days. And we've talked about this. It's in some ways a very fragile system.

Steve: Well, yes, I mean, and not even "in some ways." One of the comments that I made last week was that people have been surprised, for example, that the Anonymous group, using that LOIC, the low orbit ion cannon, that was their name for their distributed denial of service attack tool, and all it was doing was making standard TCP connections and transferring data to, like, major sites like Visa and MasterCard and so forth. But it turns out that normal web traffic is the way our systems are scaled, so that they're able to handle your typical normal amount of user access, which really is very spotty and not continuous. You click a link, you ask for the page, you get the page, you ask for a few images and things, and then typically you're there sort of figuring out what you just got for a while before you do it again. So what's interesting is that it doesn't really take that much to overwhelm many of these high-profile sites. And as you say, we're really dependent upon it.

Which actually is one of the stories I wanted to mention is, as I'm sure you have seen, Leo, the FCC yesterday, that's on Tuesday, voted to enact the so-called "Open Internet" order, which was the long-awaited Net Neutrality bill. However, it really disappointed a lot of people. Net Neutrality is this notion where an ISP, someone who we pay to give us connectivity, would be formally prohibited from any sort of traffic shaping or content discrimination. The concern is, for example, that if Comcast and, for example, NBC merged, which I think that there's discussion of that happening right now...

Leo: That's right.

Steve: ...that Comcast could preferentially carry NBC's content over competing content.

Leo: Well, it's already happened because Comcast has gone to Level 3, who provides the backbone for Netflix, and said you are going to pay more for Netflix.

Steve: Yes. And there's a peering, a big peering dispute with Level 3 and Comcast. Yeah, essentially the idea would be - so if Comcast was kept as a strict, just a conduit - which to me really seems like the right solution, is not give these carriers any motivation for preferring any Internet content over any other, just make them a pipe. But it looks like that's not going to happen because we've got the tendency to merge companies and reduce choice, unfortunately. So anyway, the point is that this is a very weak Net Neutrality bill...

Leo: Very disappointing, yeah.

Steve: ...which has really upset the people who have been following this closely. It doesn't do anything for mobile broadband.

Leo: That's the real problem.

Steve: Yeah.

Leo: I mean, it just basically says, okay, you do anything you want.

Steve: Yeah, exactly. And you've got to worry, too, when the carriers are happy with this bill.

Leo: Oh, yeah.

Steve: Because it's like, oh, okay, wait a minute, maybe we ought to read this again and see what it was we just gave them. Also, if you read the language, it says, like, nothing. For example, it says, "Bars wireline-based broadband providers from 'unreasonable discrimination' against web traffic." Well, what does that mean, "unreasonable"?

Leo: Right.

Steve: Unreasonable to the CEO of that company?

Leo: Yeah, who's to say, exactly.

Steve: Or to you and me? So anyway, it just seems like it's - actually it's worse than having nothing because now it seems that there is something, and so now they're not going to work on it. And even so, apparently there's already talk of repeal of this.

Leo: Yeah, it's funny, it made no one happy. Which I guess in some ways means maybe it was a good compromise. I don't know. I was very disappointed. And, frankly, I don't know if the government should be involved in this. But we are seeing evidence that somebody needs to do something.

Steve: Well, analogies have been drawn to the idea of turning the internet into things like radio and TV and cable. And it's like, oh, no.

Leo: I don't want to go there.

Steve: Oh, yeah. So that would not be good.

Leo: It's really difficult to know what to do.

Steve: Well, when you bring big bucks and commerce into the picture, and companies get really big, they start saying, hey. They get aggressive. And it's not clear that the customer wins in these cases.

Just when we were recording the podcast last week, that news had hit alleging that the crypto system, the crypto framework in OpenBSD may have been compromised, as long ago as a decade ago, by a government contractor called NETSEC. Additional news has come out in the meantime. And as it turns out, I saw the name of an old friend of mine, who's now ex-FBI. He was with the FBI. His name is E.J. Hilbert, who was involved sort of in this side of the tech-y stuff. That's why I knew him was because he was involved in the technical side and also was local. It was over coffee when I was getting ready to do my eCommerce system that he gave me some tips about things I wanted to make sure I did in terms of configuration because he had seen lots of credit card fraud perpetrated against other organizations that had not configured things correctly. And I've already passed all these on to our listeners over the last several years. So there's nothing new there. But he had some knowledge of it.

And what it appears to be is that this company, NETSEC, produced for a while - and they're gone now, but they produced 10 years ago a crypto accelerator. So there was hardware that accelerated crypto. And of course we needed that much more once upon a time, 10 years ago, when our processors were much less capable of just doing crypto in software with sufficient speed. So hardware acceleration of cryptographic operations was not uncommon and was much more common back then.

And what is believed to be the case is that a fork of the OpenBSD UNIX was used experimentally by this company to experiment with modifying their own drivers for this particular card to introduce some leakage, purely as a proof of concept, does this kind of thing even work. Never at any point was there any notion of this code being put back into the public, being merged back into the regular OpenBSD source base. So this is, as we hoped, and as we suspected last week, a complete red herring. It is the case that, if you parse the facts that were presented, technically they're true. But any conclusions drawn that implies that there's something, like, out in the wild that has been compromised is, like, many steps away from being true. So that's good news.

Leo: Yeah. And frankly, besides being a relief, kind of not surprising. I just couldn't fathom that this could possibly be the case.

Steve: It is the case, though, that many people have been looking at the code, and some bugs have been found. And some messy functions have been cleaned up. So, as is always the case when you look at something more carefully, oftentimes you're able to say, oh, look, we're smarter now than we were 10 years ago. This is a better way to do that. So it's sort of been good. This is, like, dusty old code that no one's really looked at for a long time. And so in dusting it off they've said, oh, we can fix this up a little bit and make it better.

Leo: Good.

Steve: So that happened, yeah. I got a kick out of one little news blurb, and that was the way that some of the WikiLeaks fight-back guys, members of the so-called Anonymous group, had been tracked down was from document metadata which they had left into some documents that they had released. And I thought this was a good time to

mention to our listeners, who are security and privacy conscious, sort of a reminder about document metadata because, for example, Microsoft Word is famous for doing this. And even PDF files now contain a lot of stuff. The idea being that the data that you see in the file itself, the actual presentation onscreen is now just a portion of what the actual binary file contains, and that there's a lot of stuff that you don't see when you look at page one, page two, page three printed out and so forth, which is actually present. And it sort of accumulates as this document moves along.

Microsoft has some pages that talk about this awareness. And it's from there that I got a list, their list of things that they know they include in Word. So, for example - well, Word, Excel, and PowerPoint, their main flagship products - your name, your initials, your company or organization name, the name of your computer, the names of the network server or hard drive where the document was saved, the names of previous document authors, document revisions and versions, comments and more.

So it's just sort of a little heads-up to our listeners that, if you were concerned about the history, the past, the ownership, the authorship and so forth, it really makes sense, for example, to export the document into a text file, which has no metadata, or maybe into an RTF file, which is only presentation data and not data that isn't displayed. Or print it to - you could print an existing document, for example, into a PDF. Then you've got it in a format where it's only containing the stuff that you printed. Although, again, it could also know anything about the system that it was being transferred on. So it's a little sticky these days to be completely anonymous and to maintain privacy, even with documents that don't appear to be releasing any information about you.

Leo: It's amazing, isn't it, yeah.

Steve: Speaking of all the feedback that I've received from people through Twitter, someone tweeted me, a Tim Raymond, just a heads-up that a couple days ago Secunia, who produces that very nice personal software scanner called PSI, thus the acronym for personal software scanner, just took it to v2.0, with an updated user interface and now the ability, if you choose, it will take responsibility for maintaining and updating the obsolete or down version software installed on your machine.

So to remind people what this is, it's free, completely free, Secunia. You can just put that into Google. It'll take you to Secunia. And in the upper right-hand corner of their page is a link either for downloads or specifically for PSI, can't remember which. But it's easy to get there. And this is a scanner which we've all used, anyone who was interested in this in the past, and, I mean, to very nice benefit. I like it. It will just rummage through your hard drive and look at the versions of virtually everything you've got installed. It's very comprehensive. It's also very small. I think it's only a couple meg in size. And it alerts you of things you've got on your computer that are no longer current.

And we're very used to things like Acrobat and Windows updating themselves, certainly Chrome browser from Google, updating themselves on a more or less constant basis. But there's a lot of apps that don't have that built in, or that only do it when you run them, and you may be running a very old version the first time you start something up again. So this will look at it even when it's not running and say, hey, by the way, there's a whole new version of this. So it's nice, and I can endorse it easily. So it's now at v2.0 with new features.

Leo: Cool.

Steve: Google has enhanced their warnings. We've talked before about how, when you do a Google search, and it brings up a page of links, you may see, and I've seen a couple times, a warning saying this site may be compromised, the idea being that Google's own bots are looking at sites, when they go into them in order to index them, and flagging when they see, for example, clear malware. They'll warn you that this is probably not a link you want to click on. And I think when you do click on it, it takes you to an intercept page that Google puts up, I mean, to really protect you from doing this by mistake. And then you have to deliberately say, okay, yes, I really want to go to this place that you think is bad, in order to get past it.

Well, they've enhanced that, making it more sensitive, adding another phrase where it says "This site may harm your computer," instead of "This site may be compromised." Wait, I got it backwards. It always used to say "This site may harm your computer." Now what they've added is "This site may be compromised." Meaning that they've detected some things that they sense indicates that the site may not be under the full control of the site's owner. So various types of not necessarily malicious stuff, but just something that seems fishy to them, they'll now warn people of. So I think it makes so much sense for search engines to be able to take this kind of responsibility because for most of us, that's the way we view the 'Net is through the lens of the search system, which finds things for us that we want to explore further.

In the ongoing battle with Google and their inadvertent collection of wireless data, I did note that Connecticut's Attorney General Richard Blumenthal had given Google a couple weeks, several weeks ago, to turn over the data that they had mistakenly and inadvertently collected from Connecticut's residents. And as of 5:00 p.m. Friday, which was the end of last week, that is, December 17th, Google had not done so, and is apparently refusing to do so. Which I applaud them for. From my bird's-eye view, I really can't see any use for Google turning this stuff over. And it just seems to me just more stirring the pot without any particular value.

Leo: Hmm. Good.

Steve: So, yeah. Again, I think Google can just say, look, we got this random stuff. Just let us delete it. In fact, they have deleted all of the UK data finally. So that was good.

We've talked about IP space depletion and how the clock is running out for, like, late summer of 2011 for there literally to be no more IP addresses. Everybody who's got one can keep theirs, pretty much. But the rate at which they really have been allocated is accelerating as of course Internet becomes a bigger deal. And we're running out of our 4.3 billion IPs. In a little weird event, there's an ISP in Okinawa who was given a chunk of the 49-dot block. So it used to be that there were no IPs beginning with 49. That was one of the major class A networks that just had - it's, like, five that we've talked about before that just wasn't ever used, which is why Hamachi had been using the 5-dot network.

Well, the 49-dot network is beginning to be handed around. And I got an interesting tweet from Brandon Carlson, who is in Okinawa, saying that a provider that he uses, GL Broadband, has a page up to explain to their users that, as a consequence of the fact that the routing tables on the Internet are taking longer than they should to be updated,

that's why their users are unable to access Xbox Live, ESPN Player, LiveStrong, Meebo, NFL Gamepass, the Nova Southeastern University, and the Washington Heights Church, just to name a few.

So what's happened is that routing tables - we've been talking about the problem with inadvertent updates to routing tables, which was for some length of time diverting traffic through China. Routing tables are something that ISPs are being increasingly careful about letting change. And so even though the allocation of this new IP, 60 million IP space, of which this GL Broadband has received a chunk, even though it's official and legitimate and real, there are some routers that have not gotten the message. And those are the routers of the specific sites, or routers related to the specific sites that some GL Broadband users cannot access. So their traffic is probably able to go to those IPs of those sites, but Internet connections require a round trip in order to establish a connection, in order for any data to come back, as well.

And so packets are trying to get back to 49 dot something dot something dot something, and they're being thrown away. It's a so-called "black hole," which means just sent into oblivion because those routers, routers somewhere along the path back, still don't know where that particular 49-dot IP range is. So they're inaccessible, essentially. Again, it's largely functional, but there are little patches sort of grayed-out around the Internet. So I thought this was an interesting little anecdote for the kind of problems we're seeing as major new blocks of IP space are brought online for the first time, as will be happening for the next nine months until there is no more.

Leo: At least that long, yeah.

Steve: Yeah. Also our friends in Florida at - I'm blanking on their name. Sun. Alex.

Leo: Oh, yeah, yeah, yeah. I know who you're talking about. Sun thing.

Steve: Sun something. Ah. Shoot. Well...

Leo: The chatroom will tell me in a second. It just takes a little while here.

Steve: Yeah, they will. Anyway, they were blogging about a new problem that has been picked up by a number of other people. The next annoyance...

Leo: Sunbelt.

Steve: Sunbelt Software, yes.

Leo: And the winner is a Monkey Mind.

Steve: And I was also - they just got acquired by somebody, a three-letter acronym, GLT or something [GFI]. Anyway. So Sunbelt Software, a great following and source of

security information, blogged recently to people who follow their blog to watch out for fraudulent defraggers.

Leo: Oh, geez.

Steve: That's the latest thing to happen. There's so many useful free software out there, it's not surprising that the bad guys are going to be mixing their own malware in with the good stuff.

Leo: Of course.

Steve: So there's HDDRepair, HDDRescue, HDDPlus, UltraDefragger, ScanDisk, DefragExpress, and WinHDD have all been identified as bogus. They claim to be a free defragger to make your computer run faster, the way it used to. And who doesn't want that? What these things do, though, they're scareware. You run them; they actually do no defragging at all, but they apparently do something. And then they come back with a note that, oh...

Leo: Oh.

Steve: ...you've got serious problems, baby. We're going to need another $20, or an initial $20, or more in some cases, to fix this problem. So again, this is going to catch a certain number of people who unwittingly download this and don't know any better. So I thought I would take the opportunity to mention my top three defraggers for Windows.

Leo: Oh, good. Do you really need them, though? I mean, you're the expert.

Steve: Not so much.

Leo: I think NTFS and OS X on the Mac side, the HFS Plus file system, do a much better job of keeping themselves from getting fragmented.

Steve: Well, they do. But my feeling is, for example, when you initially set up a system, and you install a million secure Windows patches, that process of installing all those patches and replacing all the files, it really does mess things up. So I do like to do a post-installation defrag because, from a file recovery standpoint, if anything really bad happened...

Leo: Oh, that's a good point.

Steve: ...to the master dictionaries on your drive, having the files' pieces be contiguous makes recovery possible. I mean, it really makes a difference from a data recovery standpoint. So but the downside is, we know for example that, as people move more and

more toward solid-state drives, there you really do not need to defrag them because there's no benefit, there's nothing, no head moving at all, nothing mechanically happening, and much lower probability of there being, like, file system level problems that would be caused by bad sectors because these are solid-state drives.

But the point is that, not only are they solid state, but they don't like to be written to. We know that solid state drives have a write limit. Whereas physical drives aren't degraded by writing, solid-state drives are. So defragging, which is a write-intensive thing, is something you'd rather - you'd probably not want to do on a solid-state drive as much as on a physical drive. So...

Leo: So what do you recommend?

Steve: Vopt.

Leo: Oh, yeah. That's been around for ages. That's, like, Golden Bow; right?

Steve: Golden Bow's Vopt.

Leo: That's, like, as old as SpinRite.

Steve: Actually, yes. In fact, the author, Dennis somebody and I, were friends back in the very early days of SpinRite.

Leo: Yeah. I think I remember it from your recommendation, probably in InfoWorld.

Steve: Probably. And it's also been our friend Jerry Pournelle's Chaos Manor choice.

Leo: Jerry Pournelle loves it, yeah. That's right.

Steve: I mean, he just can't stop talking about it. Today it's a little pricey at $40. There is an extremely good free one called - they used to be called JK Defrag. It's been recently renamed My Defrag, and it's just at MyDefrag.com, which I can recommend without reservation. It's open source. The guy makes the source available. Very nice UI. I like defraggers where the resolution of the screen - where you can, like, see all the little fragments, when it's high resolution instead of low resolution, because I just, like, I'm fascinated. I can just sit there and watch my hard drive be defragged for hours, which is typically how long it takes. I mean, it makes SpinRite seem reasonable in terms of data recovery.

And then a very reasonable-priced product that I really like also is called PerfectDisk by a company called RaxCo. It's $29.99, or $29.95. And the one thing it is able to do that neither of the others do, being commercial, and being very mature, is it will do a boot-time defrag of your system files, the things that are in use that normally no other in-Windows defragger is able to touch.

Leo: That's huge.

Steve: Yes. So the directories and the MFT, the Master File Table, which is like the major table of contents for the NTFS file system, there are these major pieces of the drive that cannot be defragged. So all you're able to do is defrag sort of the things it points to, that is, the pieces of the files, but not the directories and the major indexes. Because this thing has the ability to do a boot-time defrag, you're able to say, yes, do all of that next time I reboot. Then when you reboot, it takes you to a sort of a pre-boot screen and rummages around for a while, doing that work that can only be done outside of Windows, while Windows has not yet mounted that volume itself, which is very handy. So it's unique. Actually there are a few other programs that'll do that, but this is my favorite of those that'll do that, and it's just $29.95. And it also does regular, in-Windows defragging, does a very nice job. But they don't have a high-resolution display. You see only kind of like big blocks moving around. I like to see all the little bits if I'm doing a defrag.

Leo: Well, that's good. Thank you. Oh, there's more?

Steve: No, no, no.

Leo: That's it.

Steve: That's it. Except that one other piece of news that I learned about from, again, my Twitter friends...

Leo: Hang onto it.

Steve: Okay.

Leo: Because I think, with any luck, Skype is back.

Steve: Oh. Oh, my goodness. That's so much better.

Leo: All right. So one more thing.

Steve: Yes. So it's funny because I thought you were going to say, okay, hold on a second, we're going to hear from our sponsors before we...

Leo: No, actually we want to hear from Skype. So by the way, suddenly it's like "The Wizard of Oz," where we went from black-and-white to color. Skype has now started working, and so Steve is back on Skype. And this just really underscores the quality

of a Skype connection, if you're feeding it with good mics and good cameras and all of that.

Steve: Yeah, baby.

Leo: Yeah, baby.

Steve: Okay. So thanks again to the great group of people, who are keeping an eye out for interesting things, on Twitter. Did you hear about the SSL keys being released through this group called the Little Black Box?

Leo: Unh-unh.

Steve: Okay. Get this. If you think about it, any device, any embedded device like our Linksys, Cisco, D-Link, whatever routers, and even something that provides SSL VPN, they need to have a private key. And the private key, that is, in order to do an SSL connection, just as we've talked about how servers have their private SSL key, and then their public one is in the certificate which they offer in order to establish a connection. Well, embedded devices have to have private SSL keys also. And this has long been known as a potential vulnerability, which no one has so far taken really good advantage of. But the SSL key is embedded in the firmware, and the firmware is downloadable, which means the keys can be found. Which means that SSL in the case of our embedded devices is not secure. It can't be. Because the thing, the sole thing you have to do is keep your private key private, which is arguably impossible to do in an embedded device.

Now, it's not such a problem, for example, for our routers, because hopefully everyone has remote administration shut down on their router, so that there's no WAN-side access to their administrative interface. And even if they did, we would hope they've by now changed the username and password and made them very strong, if for some reason they did need to leave open WAN-side access. So the only time you would probably be establishing an SSL connection to your router, if you even bothered to, is over your own LAN, when you were using your browser to connect to it over standard SSL port 80 to the router.

But there now exists this database of more than 2,000 private SSL keys. What they've done that makes it special is that they have associated the private keys with the public keys and made it searchable. If you go to code.google.com/p/littleblackbox, that's where this project lives, code.google.com/p/littleblackbox. All you have to do is you can give this code that's been written the path to any of these embedded devices' public certificate file, and it will find the public certificate, use that to look up the private certificate, the private key. Or you can give it the SHA-1 hash of the public certificate, which is a way that public certificates are fingerprinted and communicated. Or you can give it a pcap, for example, a Wireshark or WinPcap file of the capture of data, and it will look for the public certificate in the exchange. Or you can give it access to the live network interface, and it will listen for the public certificate exchange.

Basically, what these guys have done is they've made it extremely easy to, I mean, it could hardly be any easier to find the private key, to look up the private key, given the public key, during any SSL exchange with up to 2,000 embedded devices and routers and

SSL VPNs. Many, for example, DD-WRT has a VPN version. It's been possible in the past. There have been databases like this. But in order to do it you needed to know, for example, the specific device, the model, the vendor, the firmware version, sub-version and so forth. And then you would need to, like, do a lot of this dirty work yourself.

Well, this has made it far easier. We haven't really seen any exploit of this yet. And again, I don't want to worry people needlessly, that is, this isn't the end of the world as we know it because, again, all this would allow you to do would be to, in the case of an embedded device, to potentially eavesdrop on or perform an effective man-in-the-middle attack. I do think we're going to be talking about this in the future because I can imagine ways that this information could be leveraged further, for example, in the case of VPNs, which are depending upon a key which is static.

Now, from the standpoint of fixing this, this needs to be fixed in the long term. As you can imagine, Leo, this is not a good thing. What it means is that part of the installation of an embedded device which doesn't exist today would be generating its own unique key pair. That is, there is no reason why every Linksys router of a certain firmware version needs to be using the same essentially asymmetric key pair, a private and a public key. The technology exists for those to be generated on the fly. But no one bothers with it yet.

I can foresee a point in the future where there will be an option, probably in the UI, of routers where you can press a button, and it will no doubt take a while, especially on the underpowered processing chips that typical embedded devices have. But it's something I imagine we're going to be seeing before long because it's really not a good idea for a database to exist containing, I mean, essentially the private credentials of all the embedded devices that we're using. And that exists now.

Leo: No kidding.

Steve: And it's easy to use.

Leo: It's almost like rainbow tables. Or no?

Steve: It's very much like that, actually, yeah. A rainbow table, of course, is where hashes have been precomputed. Here the idea is that even though it's been known that embedded devices had private keys, it wasn't easy to get them. You had to, literally, you had to have high motivation for getting the private key. Now it's trivial because the public key, I mean, the whole point of an asymmetric key is that the public key is in the certificate that is exchanged during the beginning of an SSL connection. And all of the crypto security comes from the fact that from the public key you cannot in any reasonable amount of time determine the private key. Except the private key is in the firmware. It's embedded in it. And so, if there's a database where you could now look up the private key from the public key, just because someone went and did that work once of extracting...

Leo: That's a lot of work, though.

Steve: ...the private key - yeah, but we've got a lot of little people out there running

around.

Leo: Yeah. So obviously there was some board somewhere where they said, hey, run this program and give us what you get.

Steve: And more are being added to the database on an ongoing basis. In fact, part of this Little Black Box project has a means for people to submit new public and private key pairs...

Leo: Oh, I see.

Steve: ...that have been found. And more than 2,000 already. If you look, everything you've ever seen is already listed in this database. So that's not good.

Leo: Hmm. Interesting.

Steve: Yeah. Following up on last week's mention of, in the Q&A, of ShopShield, which was one of these credit card replacement deals, I told people that I had had no chance then to do any vetting of it to see what I thought about it and to check into it further. Well, I feel a little bit mixed. It's not inexpensive. It's $10 a month or $100 per year to use this, which is a little daunting. They do have a pay-as-you-go variant where it's $2 per generation of a temporary credit card. And if I were going to use it, that's probably what I would do. For example, if I was ordering something from someone that just seemed really sketchy, where I really, I mean, like, they didn't offer any referral to PayPal or Google Checkout; if I had no choice but to give a sketchy site my credit card information, I mean, whereas otherwise I would probably just not buy from them, but if it's something I really wanted and they were the only people that had it, and I had to give them my credit card number, this is somewhere where it's worth $2 to me to run it through this ShopShield as a third party.

Am I willing to pay $10 a month? Well, to me, I'm not sure how much I would use it. So that's a tradeoff that I'd have to think about. It seems like a lot of money. I wish it was less. And they, in general, they're clearly trying to monetize the service that they're offering. They offer the ability, for example, to create a one-time email connection where you give somebody an email link, and then that email is bounced to you, using them as an anonymizing service. And that's a dollar. It's like, oh, come on. I mean, that seems really outrageously expensive for how simple it is to do. So I wanted to follow up on that. I'm less excited than I was hoping I was going to be after looking at all of the gory details.

And you'll remember also in errata that I mentioned that one of our questioners from last week, Nick in Thief River Falls, mentioned that Chrome does have side tabs. And you went looking for them, presumably using Chrome for Mac last week, and you didn't see that option. It's not there in Chrome for Mac.

Leo: Oh.

Steve: Only Chrome for Windows.

Leo: Oh, rats.

Steve: I don't know why. But I went looking for it, too.

Leo: Yeah. I didn't see it, and now I know why. All right.

Steve: Yes. It's not there, unfortunately.

Leo: But it's a nice feature, and I'm sure they'll add it eventually.

Steve: That would be great. It is very nice to have tabs running down the left-hand side. I like it a lot. And then I just wanted to mention a listener of ours, John Newcomb, who had a nice experience with SpinRite. He said, "Dear Steve, just wanted" - actually a nice thing to say about us, too. Funny in this context of this horrible first half of the show. He said, "Dear Steve, just wanted to tell you how much I appreciate you for all the great info I get from your podcast with Leo Laporte. You have such a wonderful radio personality. And with Leo, you guys are a great team.

"I recently bought a copy of SpinRite that I have added to my bag of tricks. I'm a computer tech and work for a company who serves the dental industry. SpinRite has already saved me a lot of time. I was in an office the other day working on a machine that would not fully boot Windows. I ran SpinRite, and it recovered several bad sectors, which allowed me to then image the drive and install a new one. Thank you for making my job easier. Your software works so well; and, using it and observing all the little details you took care of, I think it's clear how much care you put into it. Sincerely, John Newcomb, Oakland, California." So thanks very much for the update, John.

Leo: Very cool. We're going to take a break, come back with Bluetooth security.

Steve: Yes, how Bluetooth operates.

Leo: I love that. Steve, I'm ready to talk Bluetooth. I use Bluetooth all the time. I've got my Bluetooth headset on. My Ford SYNC is using Bluetooth. I use it on stereo headphones. And so I'm very concerned. How secure is it?

Steve: Well, and for me, I use it in tethering mode with my Blackberry when I'm away from WiFi.

Leo: Right.

Steve: It's a great solution. I have the MiFi from Verizon, but that's just one more thing

you've got to pay $69 a month for. And if you have a phone which is tethering-capable - I guess the iPhone is still not? Is that the case?

Leo: No, it is now. It's always been in Europe. And I believe you now can do it. AT&T allowed it on the iPhone 4.

Steve: Okay. So there you would be - does it tether with Bluetooth or with WiFi?

Leo: Now you're asking me - I think it's - pretty sure it's Bluetooth. But I don't know. No, maybe it's just USB. I don't know. That's a good question.

Steve: Anyway, so in my case...

Leo: I never use it. But on the other hand, I am using Bluetooth tethering all the time on my Android phones. And so that's another security, I didn't even consider that, another security issue. Yeah.

Steve: Yup, I was just going to say that, Bluetooth on Android. So the technology bears the horrible age of about a decade. When you look at it closely, first of all, this came from Ericsson, Ericsson before they became Sony Ericsson, and Sony bought the Ericsson phone line. Ten years ago, actually I think it was in '99, so almost 11 or 12 years ago, Ericsson wanted a new technology that would allow them to link their phones to peripherals like wireless headsets, for example. And the good news is that they really understood and appreciated the need for security. The name Bluetooth actually comes from some Danish king who was Harald Bluetooth, who consolidated Denmark, and I guess part of Norway, back on the early 900s.

Leo: United Scandinavia, and so this is a uniting technology.

Steve: Exactly. That was the goal. And in fact the logo...

Leo: That's what Ericsson says, anyway. I don't know if it's true.

Steve: Yeah. The logo, apparently in whatever wacky language they had, HB, Harald Bluetooth, in some language is something that looks like an asterisk, and then something that looks like a "B." And so that's actually, that's where that logo came from, where it's that sort of that pointy, looped "B" with funky little lines coming out the back side. Actually, if you look at it, you can sort of see an asterisk embedded with a "B." And so that was the source of the logo. So they introduced this in '94, after a couple years of work. And then about four years later Ericsson was joined by Nokia, IBM, Toshiba, and Intel to form the so-called Bluetooth SIG. They were the original members of the SIG. And now it's literally thousands of companies.

The spec started off a little rough because it is, as I said, it shows its age. It is just a disaster of a protocol. If you sat somebody down with the spec - which by the way is

1,200 pages, I mean, it's like the U.S. tax code - if you sat them down with it, there is absolutely no chance at all that they could produce something that worked with anything else. Their stuff would work with their own stuff, but there's no chance they would have interoperability. And not surprisingly, that was the big problem with Bluetooth in the beginning was that people would implement their Bluetooth technology, and they wouldn't be able to connect with anything else. They could work with their own stuff, but not anybody else's equipment.

So over the years there has been a code base, essentially, that has been refined and is available to people who want to implement Bluetooth, that basically is more useful than the spec. The specification is just - it's horrendous. People who have used Bluetooth are aware that it's not something that you have to configure. So the goal, Ericsson's goal, this was going to be a consumer network, something where these things could somehow find each other in a process called "discovery," and just interoperate. However, they also wanted it to be secure.

So what they did was they created this notion of pairing, where you pair two Bluetooth-enabled things through some sort of an easy-to-use process. Devices may or may not have screens. They may or may not have keyboards. It could be, for example, just speakers, where it's somehow going to just work. Yet you still want there to be this pairing process. So what Ericsson worked out is this notion of a device being discoverable, which if you've got a user interface that is on your device, you can typically turn that on and off.

One of the most important things to do in security with Bluetooth is have your device not be discoverable by default. That is, you only want it to be discoverable during the period of time that two devices need to literally discover each other because having it be discoverable begins to open it up to some security problems. Mostly they're implementation problems of the particular vendor that has put things together. And the history of Bluetooth security has been compared to other histories of security relatively okay. We'll talk in a couple weeks about where this has not been okay. But the number one problem, the chief source of trouble has been that people have left their devices in a discoverable mode rather than not.

Leo: We've talked about that many times before.

Steve: Yes, yes. And in fact...

Leo: I think we did a TV segment on it.

Steve: We did. And in fact turning on a Bluetooth device - we were in Vancouver at that point.

Leo: Right, right.

Steve: And there was like, everybody had their phones were discoverable.

Leo: Now, by the way, I should point out, that was a couple of years ago. And I think it's changed completely.

Steve: Yes.

Leo: Most phones default to not discoverable.

Steve: Yes. And the really nice thing is if the UI won't allow them to stay that way. There's no reason that it ought to be like you set it to discoverable and leave it. It ought to be, I'm going to do some pairing, so press this button, and I've got five minutes, and then it's going to turn off. And so what that pairing mode would be, would be Bluetooth discoverability would be on. Now, what this means is that the device responds essentially to an inquiry message in the Bluetooth protocol. So one of the concerns has been that, even without being paired, your devices are part of anyone's Bluetooth network that has a Bluetooth device on. So there's sort of an inherent low level of promiscuity happening between all Bluetooth devices, all the time.

So the other thing I would recommend and do is turn off Bluetooth completely, if it's not inconvenient to do so. For example, I've got Bluetooth on my Blackberry. It's off all the time. Not only is discoverability off, but the Bluetooth radio is off. And that's the case with my laptop, which is the thing that I would normally connect my Blackberry to, if I wanted to get on the 'Net when I was not near WiFi. So, and that also has the advantage of saving some power because the Bluetooth radio, although the Bluetooth technology is very lean in terms of power consumption, it's not zero power consumption. So if you want your handset, your handheld to run longer, and if you're not actively using Bluetooth, and if you're not going to, like, use it soon, turn off the radio completely because there's no way to be more secure than not to have it on at all.

So Bluetooth operates under what's called the "unlicensed ISM band," which is 2.4 GHz. Nominally, it gives us about 10 meters of range. So think of that as, like, around 30 feet is the distance between two devices which are running what's called "class 1," which is one milliwatt of power. There is a class 2 and a class 3. Class 2 devices have 10 times the power and substantially more range. And then class 3 devices can go at 100 milliwatts and extend up to 10 meters. So it is the case that, if you have a class 3 device pumping out much more power and with a much more sensitive receiver, that you can't count on that 10-meter range for security because it can be as many as 300 feet, 100 meters. So you can get some distance out of Bluetooth. So one of the things people generally fall back on for Bluetooth security is that it is a short-range technology. It's something that's nice about it, but it's not something that you can count on. And so it's meant for a so-called PAN, Personal Area Network. Or also in the Bluetooth spec they refer to it as a "piconet," very small little networks.

Leo: Yeah, that makes sense.

Steve: So every Bluetooth device has something that's exactly analogous to a MAC address on Ethernet adapters. We've talked about MAC addresses, how they are - a MAC address is 48 bits, where 24 bits is assigned to a manufacturer, and another 24 bits is a serial number of that manufacturer, so that the concatenation of those two 24-bit pieces, in the case of an Ethernet MAC address, will be unique.

We have exactly the same thing, a 48-bit unique device ID for every single Bluetooth device. So no two of them are the same. So that gives us a six-byte address which will be unique. When the two devices are paired during this initial handshaking pairing agreement, they exchange their 48-bit identities, and they also agree upon during this pairing a 128-bit key. So that's the thing which is agreed upon during pairing and stored in each of their databases because anyone who's used Bluetooth will be aware of the idea that you pair once, then the devices are known to each other and do not need to go through this pairing process again. The spec also allows for a 248-character, user-assignable name.

And, for example, I think I called my Blackberry a "Crackberry" when I was setting it up, and I noticed that when I was pairing with my laptop, it said, oh, I've just found Steve's Crackberry. Is that the device that you want to pair with? It's like, yep, that's the one. So that's a user-assignable string just for your own human recognition purposes when you are trying to identify the device to help you find it.

So this pairing database contains the 128-bit key, which is arrived at randomly; and the 48-bit device ID of the associated device; and this 248, up to 248 character, although normally user interfaces that you work with will limit you to something much shorter. But it's long enough for you to convey your meaning to identify what that device is. There's that triplet which is established for each Bluetooth device. There is a sort of a range of pairing authentication from none to extreme, based on a so-called PIN. And I'm sure Leo, for example, if you've paired something before, some devices just have like a default of 0000.

Leo: Right. That would be none.

Steve: Which is to say, exactly.

Leo: No security, yeah.

Steve: No security. The idea of the PIN is that you want something, the only time where there is known protocol vulnerability - meaning a vulnerability that the spec understands, not a vulnerability because of a mistake made in the protocol, where crypto guys are going to attack this thing, but an understandable man in the middle, we know we have a problem at this particular moment that's during pairing because you have two devices that don't know each other, and you need some means of authentication.

We've often talked about how authentication is the only way that you can theoretically solve a man-in-the-middle problem is if each end is able to somehow authenticate. If you don't have that, then there's no way of knowing that there wasn't a third party that inserted themselves in the middle where you're mistakenly authenticating with that third party, that is, each endpoint is authenticating with that third party, which has now successfully managed to negotiate itself a role in the middle. So what you absolutely have to have is something out of band, "out of band" meaning some channel of information not in the band, not in the same stream as the rest of the protocol.

And that's what the PIN provides. It provides something that you, the user, can provide to each end that no man in the middle could know. And so the most robust thing would be a long, unique passphrase which you would type into each endpoint. In that case, that

would be merged into their pairing protocol, and no man in the middle would know what that was because they're seeing the result of it after encryption. And only when the other side sees that the other side obviously knows the same passphrase, is pairing achieved.

The problem is, with a little headset or speakers or something, you don't have that user interface. So there is this notion of, well, we're going to do the best job we can with pairing, given the limitations. But the good news is you probably care much less about a man-in-the-middle attack on your headset, your Bluetooth headset, than I do, for example, when I'm pairing my phone to my laptop, and I want to absolutely make sure that no bad guy is interposing themselves in between my connection, my laptop's connection to the Internet. So I want to stress, though, that the vulnerability that exists, to the degree that it exists, is only during that moment of pairing. So, for example, if you went out into the desert, or you went into a large parking lot where you knew, you could visually see that there was no one close to you, if you were really paranoid, to perform your pairing, then once that's done, both sides have this 128-bit key which is plenty of strength, as we'll see, in their pairing database. And then that process never needs to be done again.

So there's this short window of opportunity at the moment of pairing, where the vulnerability is only a function of whether you have the ability to provide information that anyone listening could not have. So, for example, sometimes, if you are pairing your keyboard to a computer, the keyboard doesn't have a display, but your computer does. So the computer can say, oh, good. I see that you're trying to pair this keyboard to me. Type the following into the keyboard. So your computer is able to give you a nice long passphrase. Normally what we're seeing is six characters because there's been a sort of an agreed-upon spec that says one in a million is enough. Six decimal characters gives us 000000 to 999999. One in a million chances of guessing that.

Leo: Especially in such a short period of time.

Steve: Exactly.

Leo: You can't brute-force it because it's over in seconds.

Steve: Exactly. And there is, in the spec, is an exponential delay in failed attempts, so that it doubles for every failed attempt. And so it quickly becomes impossible to brute-force this. And as you said, you've got devices that are, like, saying, I'm sorry, we either succeeded or failed in the pairing.

Leo: Yeah. Most of these devices don't give you an infinite amount of time to try.

Steve: Exactly. And so if you have a computer with a screen, you're able to type in, you type into the keyboard, which has no display, the sequence that the computer prompted you. So in that way you have established out-of-band communications. The weakest pairing is with something that has absolutely, it's got no keyboard, it's got no user interface. And there you may be paired with another device that says, well, what's the PIN for this device? And it's just, by agreement, it's 0000. Well, so anybody could know that. Anybody, technically, if they were listening right at that moment, was very good with radio and had the ability to intercept this communication, they could do so. But

that's the only known vulnerability in the protocol.

And it's a tradeoff that I think is entirely reasonable, Leo, as you said. In the case of a keyboard, you can type in a long passphrase. Between two more sophisticated devices you can do something even longer. So it ended up being, from a consumer standpoint, an efficient and useful protocol, and I think a nice set of tradeoffs. So once the devices have established their pairing relationship, they use an interference avoidance technology known as "frequency hopping."

In any group of Bluetooth devices, or the so-called piconets, you can have up to eight devices in a single piconet. So, that is, you could have eight devices that are sort of essentially connected together. One is always a master, and then you can have up to seven slaves. A device can be a master of one piconet and a slave of another, or it can be a slave in multiple piconets. But no device can be two masters at the same time. This notion of a master is established because that MAC address, which is the Bluetooth address, that 48 bits, that's known, by virtue of this pairing, to every device that the master is paired with. And that 48-bit MAC address provides the pseudorandom sequence for frequency hopping.

So the way any kind of interference is avoided is that, within this 2.4 GHz band, there's a large number of individual channels. And the master and all of the slaves that are in the same piconet, and it's typically just two, you'd normally just have two devices that are paired together, but the spec allows for eight. The master establishes the clock and, using its own address, and the slaves using their knowledge of the master's address and using the clock which is publicly transmitted, that allows them to do frequency hopping where they're constantly changing the frequency they operate at.

That's done not as much for avoiding bad guys, but for avoiding interference because you may have - turns out that this unlicensed frequency range, this 2.4 GHz, is very popular. Wireless phones, like older wireless handset phones used in homes, all kinds of devices and instrumentation use this because it's a block of frequencies in this band that were set aside so that you needed no license to use it, given that you keep power at or below 100 milliwatts.

So the problem is that you may have, if we stayed on a single frequency, there might be something interfering on that one frequency. So the idea is that we do this frequency hopping so that we're avoiding any interference. And the later specifications have added something called "adaptive frequency hopping" where all of the devices will recognize when there's interference and avoid, while they're hopping, avoid channels that may be congested in order to improve the reliability of all this. The data rate is nominally, the original Bluetooth spec nominally one megabit per second. But there's enough overhead that we lose about 25 percent of that. So you actually get about 721 kbits per second of actual throughput.

But then later on so-called EDR, Enhanced Data Rate technology, was added. Where we used to send a single frequency-modulated signal, they added something called "phase modulation" to the actual - down at the RF level, at the radio level. So they were able to, instead of sending a single symbol, they were able to send two or three. And so the later technology is able to run at three megabits per second, so we get an effective data of 2.1 megabits per second through the whole technology.

Now, the final piece of concern, and it is concern, is the actual crypto. Being 10 years ago, we did not have AES. RC4 existed, but it was still patent-encumbered. So Ericsson did the unfortunate thing of making up their own. And everyone who...

Leo: As we know, that's a bad idea.

Steve: Everyone listening is now going, oh, no.

Leo: Why, why, why?

Steve: It's a bad idea. The good news is they used a - okay, I don't know how to describe this. I'll just describe it. I don't know how to characterize it.

Leo: Okay.

Steve: It's bad. It hasn't been penetrated, but it can only be because no one has really tried that hard because there isn't anything really that valuable typically transiting a Bluetooth connection. After all, if you're talking in your car, well, the only one who can really intercept your communication is in the car with you because you're in a steel-enclosed container, and you've got a weak radio signal at a milliwatt, which has a distance of about 10 meters. And besides, they can hear what you're saying, so there's no need to intercept it because it's carrying...

Leo: That's a good point.

Steve: It's carrying the audio that anybody with ears can hear. So typically it's low-value stuff. But the technology uses a pseudorandom sequence generator. Now, that's not bad because that's what RC4, after all, is. So you can have...

Leo: That's all we have, really, right, is pseudorandom.

Steve: Yes, exactly. Well, no.

Leo: I mean, computers...

Steve: Block ciphers like AES, where you're taking and mapping 128 bits into a different 128 bits, that provides a keyed transform of a block of 128 bits into another. But the stream ciphers, like RC4, can be very good. I was just saying last week, and I mentioned at the top of this show, that Skype is using RC4 right now. We're talking over RC4-based encryption. And nothing can crack it because it's done right. I'm not so sure nothing can crack this one. What they did was - now, again, 10 years ago, so I forgive them, they needed something that would be low power, that is, low compute power, that could be easily implemented in hardware, even hardware-assisted software, or maybe even largely in hardware, like in a chip, instead of having a processor that's being driven by firmware.

So they used shift registers, which is never a good place to start. There's something

called LFSRs, Linear Feedback Shift Registers. The idea is you use - a shift register is, think of it as a string of bits that shift in one direction so that there's an input, think of it on the left, and every time this is clocked, all the bits in the shift register move one to the right. So bits that go in come out the other end the number of clock pulses later that this thing is long. So if you had an eight-bit shift register, the bits would come in, and eight bits later they would come out.

Well, there is a technique for generating pseudorandom streams where you take multiple taps, it's called "taps" on the shift register, where you've got this string of bits, and then you look at maybe three of them, three specifically located bits, and you XOR all those together, along with the output. And the result of that is what you feed back into the input. So what this means is that this will, as the bits are shifted along, the bits, which are ones or zeroes, in the particular points where the shift register is tapped, they determine the next bit being shifted in.

Now, in fact, this uses four of these linear feedback shift registers with lengths of 25, 31, 33, and 39 bits, respectively. If you add those together, 25 plus 31 plus 33 plus 39, you get 128. Which is not a coincidence, obviously. The idea is that the key that is being used to key this is dumped into the shift registers. So you have a number of problems. For example, if a shift register had zero in it, say it was all zeroes, well, if you tap it three times, and then take the output and XOR those together, you're going to get zero. So if you feed a zero in, the shift register's never going to have anything other than zero, so it's a degenerate case. So you've got problems with, like, looping in the shift register where you might end up with a certain combination of bits that don't generate much randomness in there.

So to deal with that problem they take the four outputs from the four shift registers and feed them into another random-looking blob of logic. I mean, I've looked at all this, and I remember thinking, oh, this really doesn't look good. Again, this is the kind of thing where the engineers 10 years ago said, look how confusing this is. Nobody will ever figure this out. I mean, that's where bad crypto comes from is from that kind of idea. It's like, oh, this is so one-way engineered that nobody will ever be able to reverse-engineer this.

Well, again, I'm sure that it's because real crypto people have never really tried very hard. For example, this is frighteningly like the GSM - GSM cellular? And maybe it's not surprising because Ericsson was involved in that, too. GSM uses four linear feedback shift registers, and it's been cracked because the feeling was there's more value in cracking cellular, digital cellular, than in digital Bluetooth. So this is different. It's been a concern. My sense is, again, it's good enough for what Bluetooth is used for, which is generally low value.

Now, again, because one of the things that makes me so uncomfortable from a crypto standpoint is they keep making it more complex, rather than it being elegant. This is the least elegant cipher system I've ever seen because, once they have that mechanism, then the key, the crypto key is derived from the secret key which was negotiated from the key that they originally shared. I mean, there were good things they did. For example, this 128-bit key that they agree upon during pairing is never itself used for any crypto operations, which is what you want. You don't ever want to use that key itself. You only want to use derived keys.

So they derive keys based on random numbers that they generate, and based on a shared random number which is publicly transmitted, and based on the 48-bit master's device address and 26 bits of the master device's real-time clock, which is incremented for each time slot. So that means that every one of these time slots, as we hop around in

frequencies, and as packets are occupying time slots, are being keyed differently. So that's good. That means that, horrible as this cipher kludge is, we're never using it for long. That is to say, every single packet has a different key. But that mess of data which is munged down to make the key is then loaded into these four linear feedback shift registers and this other finite state machine which they hang on the end of it, just like to determine what goes into the inputs because otherwise they're not stable enough. It's then run forward 200 clocks. And the last 128 bits to come out of it is used as the key.

So this tells you that the engineers were never really sure that this was secure, so they just kept adding stuff to it, like this wacky, run it 200 times to kind of get it warmed up. And then we have no idea what the last 128 bits are going to be that come out, so we'll use those.

Leo: It's like chitty-chitty-bang-bang.

Steve: It is. It is just atrocious. And now you can imagine why nobody working on their own who had the spec for this could ever make it work with anybody else's equipment because they'd get an inverter stuck in there wrong somewhere, or it'd be 201 clocks, or it'd be the even phase versus the odd phase, or, I mean, anything could go wrong with this to break it. And you would never know. I mean, it's just this bizarre Rube Goldberg contraption that generates, finally it generates, a stream of ones and zeroes finally emerge, confused and dizzy, out of the other end of this thing. And that you XOR with your plaintext in order to create ciphertext.

Leo: I love it.

Steve: Somehow we're able to hear you on your wireless headset, Leo.

Leo: Amazing.

Steve: It's a miracle. So that's how Bluetooth works. That's the technology, the crypto, the pairing. The takeaway is, never leave things discoverable; and, moreover, never leave your Bluetooth radio on if you don't want it to be. And in a couple of weeks we're going to come back and talk about bluejacking, bluesnarfing, bluebugging, and all the things that people have cleverly come up with to subvert this system that we've just described. And that will wrap up our coverage of Bluetooth security.

Leo: Next week we'll answer your questions. Actually next week we will not answer your questions. Next week we will be...

Steve: We will entertain you.

Leo: We will entertain you. We'll have a best-of because, for the first time in memory, first time ever in six years, Steve is allowing me to force him to take a week off for the holidays. And we will have a best-of which will be, in its entirety, the

classic Portable Dog Killer episode. And but we'll be back in two weeks, which will be Episode 282, so we'll be back on the even-numbered questioning system.

Steve: Yes.

Leo: And if you have a question, GRC.com/feedback is the place to go. Also, while you're there, get a copy of SpinRite, for crying out loud. Make that your Christmas gift to yourself.

Steve: Oh, and it'd be a Christmas gift for me, too.

Leo: And for Steve. Make that yabba-dabba-do go off in his office. GRC.com. SpinRite is the world's greatest hard-drive maintenance utility. You've got to have it. We just had a hard drive die, and I wanted to get it SpinRited immediately because it just - it works so well. You can also find a lot of free stuff there, including all of his free security tools. ShieldsUP! is the best known, but there are many others there, and his DNS Benchmark. Steve, you're the greatest. And of course 16KB and transcriptions of each and every show, including this one. I will snarf you next year, Steve.

Steve: Cool. And you're going to Pogoplug me?

Leo: I'll Pogoplug you, as well.

Steve: And we'll say Merry Christmas to our listeners, and we won't see everyone, but we'll be talking to everyone in 2011.

Leo: Happy New Year. We'll see you next year on Security Now!. Bye bye, Steve.