Listener Feedback #108

Description: Steve and Leo discuss the week's major security events and discuss questions and comments from listeners of previous episodes. They tie up loose ends, explore a wide range of topics that are too small to fill their own episode, clarify any confusion from previous installments, and present real world 'application notes' for any of the security technologies and issues we have previously discussed.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-282.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-282-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 282, recorded January 5, 2011: Your questions, Steve's answers #108.

It's time for Security Now!, the show that covers your security and your privacy and all the stuff you need to know online. Here he is, ladies and gentlemen, the man of the hour, Mr. Steve Gibson of GRC.com, the Gibson Research Corporation - antivirus guru, spam fighter, privacy expert. And also we talk a lot about all kinds of technologies on the Internet. Good day. Happy New Year, Steve.

Steve Gibson: Hey, Leo. Great to be with you, our first episode of 2011.

Leo: Wow.

Steve: Absolutely. We have a Q&A, since we did a replay in the week between Christmas and New Years of one of our favorite episodes, everyone, all of our listeners' favorite episodes, The Portable Dog Killer. So we'll pick up where we left off. And from now on our Q&As will be on, well, until we do another little bump, our Q&As will be on even-numbered episodes.

Leo: Okay. We're not tied to it. We do whatever. We've got some great questions for you. We've got security updates. And this will be the one place today that you probably won't hear any CES news, I would imagine.

Steve: We're just barely in front of CES. But of course that's going to dominate the news

cycle then now for the next five days, probably.

Leo: Yeah, oh, yeah. Certainly us. We start our coverage Wednesday night at 7:00 p.m. Pacific, 10:00 p.m. Eastern with the Digital Experience. Thursday we'll do ShowStoppers. We also have other stuff going on. I think 1:00 p.m. Thursday, Sarah and I will welcome you officially to CES, 'cause the show opens tomorrow at 9:00 a.m. And then all day Friday, Saturday, and Sunday. So we're going to have a lot of coverage on the TWiT network: live.twit.tv. And many of those shows, I think all of it, will appear on the TWiT Specials feed, TWiT.tv/specials.

Steve: Yay.

Leo: Yay. But usually CES doesn't have much privacy or security news. It's not really a great place for that.

Steve: Right, it's consumer stuff. But we have certainly, I mean, you and I both have a passion for that stuff. I'm sure our listeners do, as well.

Leo: Shinies.

Steve: Yes, toys.

Leo: Toys.

Steve: Boxes being delivered to the front door.

Leo: Yes, we do love that, don't we. So let me ask you if there's any security news. What's going on in the world around us?

Steve: Well, I half planned not to do a Q&A, only to do a major mega news update.

Leo: Really.

Steve: Well, because I figured that, having missed, essentially missed a news cycle by doing a repeat last week, we would have two weeks' worth of news accumulated rather than just our normal one.

Leo: True, true.

Steve: And nothing happened.

Leo: It's kind of slow. Even the hackers take the holidays off.

Steve: Even the bad guys. Bad guys have mothers and families. Exactly. So, yes, we've got some news. Over on the security side, two IE problems, actually an IE and a Windows problem. There's a big kerfuffle going on right now with Microsoft accusing a Google employee of having not given them sufficient notice of the bug that we did talk about two weeks ago. It had just surfaced, and we knew that it was a CSS problem of some sort. We didn't really know what it was. Now we know that it involves the at-import rule processing of cascading style sheets in IE, which makes everyone's eyes cross, but that's where the problem is. And it exists in all releases of Internet Explorer, manifesting itself on XP, Vista, and Win7.

It's a remote code exploitable problem. Unfortunately, this one DLL that it targets is called the mscoreie.dll. It was not compiled with the so-called /DYNAMICBASE switch. That's the switch that the compiler can accept which allows it to float around, allows that DLL to float around and load in different places in memory. In other words, that's the famous Address Space Layout Randomization, ASLR, which is now actively working to support many of the kinds of attacks that we've seen in the past. Because this DLL was not compiled with that switch, maybe it's incompatible with floating around, who knows, I mean, there may have been a reason for it rather than just oversight. Or they may have just, among all the code that goes into IE, they may have skipped one.

So what happened is the bad guys figured out how to use the fact that this DLL always loads at a known location. That allows them to use so-called ROP, or Return Oriented Programming gadgets, where they jump to the end of subroutines in that DLL, and then the subroutine finishes and returns to the caller, that is, to them. So by cleverly using sort of the tails of existing code, that also allows them to avoid the DEP protection, the Data Execution Protection, because they're not executing data, they're actually executing code. They're executing Microsoft's code, but certainly in a way that Microsoft never intended.

So what happened is, and I'm going to talk next week about the whole concept of fuzzing. It's not something we've talked about before. But this revolves around a fuzzing technology, which is to say throwing lots of bizarre stuff at code and looking for any misbehavior. The guy who did this is someone we've talked about before, Michal Zalewski, who is a security guy at Google. Back in July he notified Microsoft of this problem and gave them his code which made the problem happen. He heard nothing from them. And back then he told them that he planned to release this code in January of 2011.

Leo: That's fairly typical from a security researcher, to say here's the code, you've got some time to fix it, but then we have to tell the world.

Steve: Six months.

Leo: Yeah.

Steve: Six months, exactly. And so what's happened is now Microsoft is coming back and saying, well, we didn't get - this was not responsible disclosure. We weren't told in time.

And they're saying we were unable to duplicate the problems. Well, they never told him that. They just didn't respond at all when he sent them this whole care package.

Leo: But they were busy.

Steve: With all their other problems.

Leo: Yes.

Steve: I know. So now they're saying that we need more time. And he said, sorry, I told you last summer. I gave you the whole ability to reproduce it. They're saying, well, it didn't reproduce. Other people, and now Microsoft, have been able to reproduce it with that July code, even though what he's now released publicly is a much more updated code. The significant thing about this, well, so we have one more IE problem. This tool found more than a hundred problems spread across every browser there is. So we'll be talking about that tool next week.

Leo: Sorry about that. I had my mic on. You're probably wondering why I was...

Steve: [Indiscernible].

Leo: ...apologizing for driving you crazy. Go ahead, I apologize.

Steve: So, anyway, so the problem we talked about a couple of weeks ago, Microsoft has now acknowledged. The good news is, next Tuesday will be the 11th, since this last Tuesday that's just behind us would have been the 5th. So - is that right? Anyway, whatever it is, it's the second Tuesday.

Leo: Yesterday was the 4th, so it'll be the 11th.

Steve: Perfect. So Tuesday the 11th, that'll be the second Tuesday of January. Maybe, if Microsoft is on their game, they'll be able to fix this. And there's another new zero-day problem that they have found across all versions of Windows in the graphics side of their OS. Not much is known. They've got a security advisory posted. But hopefully we'll be talking about these both as being in the past tense next week.

Leo: Yay.

Steve: In big news that many people sent and faxed and emailed and tweeted, I mean, I was getting hit from all sides, after Christmas, on Tuesday the 28th at the Chaos Computer Club, the CCC Congress in Berlin, which we've covered in years past...

Leo: Is that like a hacker con?

Steve: Yes, it is, "Chaos" being the clue to that. Two guys with Security Research Labs, which is SRLabs.de, showed why and how the GSM Association was wrong. About a year before, and we covered this at the time, the assertion was made that the GSM cellular network was too hackable. And we talked about the crypto that they use, in fact I referred to them just two weeks ago because GSM, being very old technology, uses a sort of pre-crypto approach. They also use those linear feedback shift registers, which Bluetooth uses in a relatively safe way. GSM, unfortunately, is not so lucky with their use of the same technology.

And in fact actually it's because of the Bluetooth technology arising originally from Ericsson, that is, a GSM-oriented cellular phone designer, before Sony bought them, and they of course became Sony Ericsson. That's why Bluetooth uses the same sort of technology that GSM uses, sort of from a common heritage. The problem is that there is a huge amount of known, what's called "known plaintext" in crypto, in the cellular protocol, meaning that a lot of what's sent out over the wire, or sent out over the air, we know what it is that has been encrypted. And the technology generates, uses these linear feedback shift registers to generate a pseudorandom bit stream which is then exclusive ORed with the plaintext to create the ciphertext.

As our listeners know, when you XOR the ciphertext again with the bitstream, you get back the plaintext. That's how you decrypt it. The other little quirk about XORing, though, is if you were to XOR instead, you XOR the encrypted data with the known plaintext, what you get back is the bitstream, that is, the original cipher bitstream. So what's been generated in the meantime are two new technologies. There is now, available for downloading on BitTorrent, a 2TB rainbow table for GSM. So to remind our listeners, a rainbow table is essentially a table of some crypto operation, typically hash tables, which have been performed once.

Leo: Kind of pre-calculated.

Steve: Exactly, pre-calculated. So it's a one-way function which has been performed once, and then the results stored so that you have sort of the results of this operation stored, so you don't have to...

Leo: In the old days before calculators, maybe not before slide rules, but you'd get these sine and cosine tables.

Steve: Yeah, exactly.

Leo: Just pre-calculated math.

Steve: Exactly. So what these allow you to do is they allow you to essentially perform the reverse function, even though the point of the crypto is for there not to be one. That is, like for example a hash is meant to be a forward function such that you put all this stuff in, and you get out a result. But it's a one-way function. You can only go forward.

You can't go backwards. But think about it. If you were to record in a table all of the results of going forward, then you look up in the table the result, it's going to be matched up with what you put in to get that result out, which allows you to go backwards. So it defeats that one-way function.

Leo: Whoops.

Steve: Yeah. So that was one part. The second thing they did, one of the arguments that the GSM Association made where they were defending themselves, saying oh, no, no, no, don't worry, GSM technology, which is global, is still secure, don't have to worry. They were talking about how, oh, it takes huge amounts of money and expertise and wideband technology and fancy antennas and field-programmable gate array, custom boards, the point being that there was a bar way too high. Okay. These two guys demonstrated the entire attack on voice and text using four modified $15 cell phones.

Leo: Oh. I was going to say Commodore 64s. It's worse. It was on the phone.

Steve: For a total cost of $60.

Leo: Ai, ai, ai, caramba.

Steve: Four $15 phones. They reprogrammed the phones to work differently, so they weren't still cell phones like in the old days. But the code to do that is open source. They added a faster USB connection so that they were able to move data at the speed they needed to. But basically they've completely demonstrated a low bar, that is to say, less than a hundred dollars and a soldering iron, and go to BitTorrent, and you can get the - it's funny, too, because Bruce Schneier in his blog also picked up on this, as did Engadget and a bunch of other people. Schneier mentioned that 2TB, we don't even blink when we say that anymore. Ten years ago...

Leo: That used to be a lot.

Steve: ...that would have been impossible. It would have been, I mean, not impossible, but hugely...

Leo: You're not going to download 2TB.

Steve: No. Now people use them as doorstops. So it's just not a problem.

Leo: Oh, lord. All right. Moving on, WikiLeaks.

Steve: Yeah, well, there was just a couple things. The Washington Post had an article. One of their staff reporters, Joby Warrick, had some little tidbits that I hadn't seen

before. Apparently the CIA had been asked to put their data onto the SIPRNET. Remember we did talk about the SIPRNET a couple weeks ago as being this secure, essentially a secure version of the Internet. It uses Internet technology, Internet protocol, Internet routers. It's just not the 'Net that we're all on and connected to. It's a separate one.

And the CIA's rationale for denying that they just dump all of their assets onto that 'Net is just one of security. They said, there are two many people, there are 2.5 million people who have access to that. Not just government, but also government contractors have access. And this is the so-called Net-Centric Diplomacy was the name that this system was given, which was set up when it was decided that there was just too little communication, too little interagency communication in, what's the term they use, smoke-stacking, I think, or...

Leo: Oh, siloing or...

Steve: Siloing is one. I've also heard them, like, the idea being that they're just like tubes that are surrounding the entities that are not intercommunicating. The NSA has taken this all much more seriously. Being the NSA, they're saying that they're...

Leo: Stovepiping.

Steve: Stovepiping, that's...

Leo: The chatroom said stovepiping, [indiscernible].

Steve: That's the term I was looking for, yes. The NSA, being the NSA, is proactively taking the stance that they no longer can trust anything in their own network and systems, and they're assuming that they've been compromised, and they're going to act accordingly. So it sort of reminded me of our standard wisdom with PCs, is once malware gets into your computer, you really can never know that you got rid of it all. I mean, once it's happened, you really need to just, hard as it is, pull your data off and then set up the system from scratch and then restore your data. Because otherwise there's just no way to know that you got rid of it all.

But the way apparently this all sort of got out of control is there was a keyword flagging system in this Net-Centric Diplomacy technology such that, if the key tag SIPDIS, which is probably an acronym for something, SIPDIS, that would flag any document or communication like a cable as a appropriate for archiving in this Net-Centric database. But what ended up happening, because embassy staff were never really given a clear protocol or guidelines, except they were told, okay, we've got to stop keeping all this information to ourselves, we're supposed to share it with everybody, so they started flagging everything with this SIPDIS since there was no clear policy.

And as a consequence, after the WikiLeaks problem happened, people who knew what this Net-Centric database should contain looked into it more carefully and closely and just saw that it was full of improper and inappropriate information, stuff that wasn't about maintaining the security of the country. But as we've seen, all kinds of other stuff got loose. And quoting one line from this Washington Post article, Joby wrote: "Partly

because of its design, but also because of confusion among its users, the database became an inadvertent repository for a vast array of State Department cables, including records of the U.S. government's most sensitive discussions with foreign leaders and diplomats. Unfortunately for the department, the system lacked features to detect the unauthorized downloading by Pentagon employees and others of massive amounts of data, according to State Department officials and information-security experts."

So not only was the system a repository for much more than it was designed to receive, and anything that was transmitted over this SIPRNET that was tagged with this SIPDIS flag would automatically get archived in this Net-Centric Diplomacy system, but additionally there was no controls. There was no monitoring. There was no auditing or logs of any access to this database. So it has all the earmarks of something that was sort of thrown together way too quickly, in a hurry to solve the problem, but without the kinds of mature security and privacy controls on it that all of us would know such a system would need. And we know what happened as a result.

Leo: It actually isn't easy to find the Bruce Sterling article. He wrote it for Webstock, which is a New Zealand conference. If you go to Webstock.org.nz and search for "Blast Shack," it's a fairly long piece, a couple of thousand words, and I think quite interesting, especially if you're interested in hackers and what that has to do with WikiLeaks.

Steve: And sort of their psychology. I did read that piece, by the way.

Leo: Did you read it? Yeah. I thought it was quite good. He wrote it right before Christmas. I like Bruce a lot. We've got to get him on the show.

Steve: Well, and...

Leo: Stuxnet.

Steve: Yeah, Stuxnet. There's continuing to be information coming out. I picked up a few more tidbits that I just thought I would share, which was a report from ISIS, Institute for Science and International Study, indicated that apparently as many as a thousand out of 10,000, that is to say, 10 percent of the centrifuges which were at one point commissioned and running in the nuclear facility in Iran, which it is believed Stuxnet was targeting, have since been decommissioned, it's believed because they were made to malfunction physically, essentially physically damaged by Stuxnet. They're called IR-1 centrifuges, and they are supposed to run at precisely 1,064 Hz, that is, believe it or not.

Leo: Is that like 1,064 RPM or RPS?

Steve: Well, it's not clear what the relationship between the AC frequency going into the motor, it'll be a synchronous centrifuge motor which is locked to the frequency that it's given.

Leo: I see.

Steve: And so in order to exactly control its speed. But what we do know is that Iran was apparently running theirs a little slow, about 1,0007 Hz, specifically to reduce stress. So, as I understand it, already at the 1,064 Hz, which is the spec for these IR-1 centrifuges, they're at about the limit of what they can physically handle. It's interesting how much, I mean, just a flywheel itself generates tremendous stresses inside of itself. There was a project that Ben Rosen of Rosen Research fame, and then later Rosen Motors, Ben created a flywheel-based energy store for an electric car drive train. And it turns out that advanced energy stores like that, they use bristles rather than solid disks because solid disks self-destruct when they spin that fast due to essentially, I mean, they're pulling themselves apart. So you end up with a hub of bristles. And that's what you spin in a vacuum because that's an architecture which is stable under those kinds of forces.

So what has been found is, relative to Stuxnet, is that it was instructing the centrifuges, which are already running near their limit at 1,064 Hz, to run at 1,410 Hz, which would either immediately damage them physically or would cause them to, like, crumble within about 15 minutes. And so further analysis of the code has shown that it would spin the centrifuges up to this higher speed and hold them there for a while, and then bring them back down, and then go quiescent for 27 days. So there was, like, 27 days would elapse between these attacks. And it's believed that's just to have the thing hiding most of the time, so that it wasn't obvious what was going on.

Leo: That's clever, actually. That's actually very clever.

Steve: It's very clever. And then the thing that really caught my eye was that it deliberately disguised this activity by sending commands to shut off the warning and safety controls that would otherwise have alerted the plant operators.

Leo: Clever.

Steve: So to me, Leo, I mean, as an engineer myself, if someone said, Steve, we need you to do this, I would need one. I mean, you know...

Leo: You'd have to have this centrifuge.

Steve: Yeah.

Leo: There's no way you could do it without - you'd need to know a lot about it.

Steve: Exactly. For this kind of design, especially when I heard that it was sending commands to shut off the warning and safety controls that would normally alert plant operators, you can't do that from spec sheets. I mean, in order to do it reliably, you would need one of these centrifuges in a system, working, in order to proof and debug and perfect your code, which to my mind really does up the ante of the notion that this

was probably state-sponsored by some state, meaning government-level sponsorship of this. To this point I was like, well...

Leo: Is that because they're so expensive? Is that why?

Steve: Well, yeah, you just don't order a nuclear refining centrifuge.

Leo: You can't just buy it from a catalog?

Steve: I got some yellowcake at a yard sale. I'd like to...

Leo: So is that the only thing these centrifuges are used for is concentrating - okay.

Steve: Yeah, it's the only thing they can - that's what they do.

Leo: I think we know it is a government [indiscernible]...

Steve: They're not available from eBay. So, yeah.

Leo: Wow. Did they also have to have the Siemens equipment? I would imagine they did, as well.

Steve: Yes. You would need a working, essentially, a working installation of...

Leo: So we can assume it's a nuclear power that did it; right?

Steve: It really has to be.

Leo: Or somebody who really has high aspirations. I mean, look, it's obviously Israel.

Steve: It's got to be. I don't mean it's got to be Israel.

Leo: Government.

Steve: It's got to be a nuclear power. It has to have been somebody who essentially had a facility like this, so they were able to create the same centrifuges, the same controllers, the same technology. Because you just can't write and debug code blind. You'd have to test it. You'd have to see that it was doing the right thing in order to know that you were

going to generate a payload for this worm that was going to be effective at the other end. So...

Leo: So in detective work you always say opportunity, motive...

Steve: Means, motive, and opportunity.

Leo: ...and means. So the means require a nuclear power. The motive is pretty clear, somebody who would want to slow down Iran's nuclear capability. I mean, it's not just some hacker in a closet somewhere.

Steve: No.

Leo: And opportunity, well, that could be anybody because the Internet gives everybody the opportunity.

Steve: Well, and we've got worms. I mean, a lot of other people got infected by Stuxnet who weren't running any IR-1 centrifuges.

Leo: Well, that's interesting. So they released it to the wild. It wasn't a spear phishing attack.

Steve: Correct. It just jumped around and jumped onto people's thumb drives. It was believed to have been brought in on a thumb drive. But many other organizations found this curious code and said, well, we have a worm with an unknown payload. And it's taken quite many months to peel the onion layers off this payload. And the more we see, the more we learn, the more I'm coming to the opinion that it really took a substantial effort to put this thing together.

Leo: Here's a question. Could it have been accidentally self-inflicted? Could Iran have been making such a thing?

Steve: No, because - oh. You mean like to get somebody else?

Leo: Yeah.

Steve: Yeah, well, certainly. I mean, yes. They would have had, obviously, the whole setup there if they were trying to target somebody else's equipment. There were two countries, I think it was Iran and maybe Pakistan, I can't remember now, there were two countries that were known to have exactly this type of equipment. So apparently there are many different versions of stuff that you can use.

Leo: That's interesting. I guess you could, if you were a detective trying to figure this out, as presumably somebody is, you'd say, well, who has this? I mean, not everybody uses the same stuff.

Steve: Right. And you might want to ask also, who ordered one?

Leo: That's interesting.

Steve: Anyway, on the ongoing saga of IP space depletion, which is our other news of 2011, we'll be following this, the IANA will be handing out, probably in the next few weeks, so it's expected by the end of January, the last of the /8 netblocks to each of the major registries. So there's APNIC, which is Asian Pacific NIC. There's AfriNIC. There's a couple, there are like a handful, four or five major registries that each handle their region of the globe. And they receive the netblocks from the IANA that essentially makes those then active. And then they turn around and start satisfying the needs of the people who they supply. So there's a multilevel tiered hierarchy of allocation of this space. We're still, however, on target for late summer, early fall doomsday.

Leo: Oh, good. I'll put it on the calendar.

Steve: When no more IPv4 addresses are available.

Leo: That's soon.

Steve: It's really soon.

Leo: We're not ready.

Steve: People are beginning to scurry around. There is a deployment guideline. NIST has released their final version of the IPv6 deployment guidelines, which is an 118-page document. We're going to do a podcast just on it soon because it's got a whole bunch of really interesting material in there that I know our listeners will be interested in because sooner or later we're all going to be affected by IPv6. At this point, very few of us are. But that's not going to be the case for long.

In a matter of errata, I wanted to respond to many of our listeners who have said, hey, Steve, whatever happened to the How the Internet Works series? So I wanted to let everyone know that I have not forgotten about it. There are just a few things, like this deployment guideline, and I want to talk about attacking Bluetooth next week, following up on our discussion of how Bluetooth works two weeks ago. So there are a few more things that I've sort of just got backlogged. And then we will absolutely plow into our very careful, basically soup to nuts, from the bits all the way up to the top-level operation of the Internet - packet flow and routing and all that stuff.

Leo: Look forward to that.

Steve: Which I think people will enjoy. And I had just a very nice short note from someone named Rick Shepherd that I wanted to share. He says - he's in Reno, Nevada. And he said, "I could easily give you several specific SpinRite stories and go on about how our site license has been used to save many hard drives. I could do that, but so many already have. Instead, I will state only this: Of all the products I have ever bought, sold, researched, or discussed, only SpinRite holds one special title. Nobody I have ever dealt with, anywhere and in any capacity, has ever said anything bad about SpinRite. No other product has that kind of record." So I thought that was neat.

Leo: I think that's true. And I have to say, I hear, because we advertise a number of different products, I hear whenever anybody's unhappy about any of those products. And there is no product, however good, that doesn't get an occasional ding. Look at Amazon, there's no, I mean, look at the Kindle, everything. Somebody doesn't like something. I've never seen - that's a very good point. Nobody's - I've never, well, now, we don't want to tempt fate here.

Steve: Yeah. Thank you, Rick. Next topic.

Leo: Next topic. We have great questions, 10 of them, from you the listeners. You went to GRC.com/feedback, and you asked those questions, and I'm glad you did. And I'd like to ask you a question. Have you taken the TWiT survey? Now, our sponsors know exactly how many people listen to a show. But what they don't know, and what they would love to know, is kind of who you are, not individually, but as an aggregate. Are you young? Are you old? Where do you live? That kind of thing.

Steve: Because we're not tracking you, so we don't know anything about you, you have to tell us.

Leo: Yes. Oh, that's a good point. There's a nice spin on it. We don't want individual information. You will take the survey, but you'll take it anonymously, and we'll aggregate the information. And that lets us say to our sponsors, for instance, well, we've got 53 percent males, 25-54, whatever it is they're looking for. It is at TWiT.tv. I made a bit.ly link because, if you're running ad blockers, and I suspect more than any other show we do that Security Now! people run all sorts of ad blockers and Flash blockers and JavaScript blockers. So if you're running NoScript or an ad blocker, you can go to bit.ly/listenersurvey. It's at bit.ly/listenersurvey, or just go to TWiT.tv, if you're not running a blocker, or if you're not blocking our stuff.

Right at the top there's a banner. It's a five-page survey, take you five minutes, 10 minutes to do, depending on how long you think about it. Should be pretty quick. At the end we'll ask you if you want to be part of our panel, the Podtrac panel. Our ad agency does all this. We don't do it. The ad agency does it. And the Podtrac panel helps us with research about advertising and so forth.

Advertisers, I'll give you as an example, often will say, okay, we want to do a survey

before you run the campaign and after, to see if there was recognition of the product, if people heard it, if they understood it. And those panels help us an awful lot, as well. It's one of the reasons people keep coming back to TWiT, frankly, to advertise, because we work. So you can help us. TWiT.tv, just right at the top there, or go to bit.ly/listenersurvey. And we'd appreciate your support. Are you ready for a question, my friend?

Steve: Let's do it.

Leo: All right, let's do it. Let's do this thing. Number one from - I love his name - Ranga Reddy in Asbury Park, New Jersey, home of Bruce Springsteen, wonders about SSDs and full drive encryption. And I wonder, now that he mentions it, I want to know about this, too. Been a big fan of the Security Now! podcast since year one. Security and privacy issues sometimes annoy and anger me; but, since I listen to your podcast while working out, it fuels my workout. That's good. Put the frustration and anger right into that Stairmaster. You recently stated that SSDs don't need defragging. They also don't need SpinRite.

Steve: Correct.

Leo: Because the excessive writes of defragging would ruin the drive. And fragmentation is not an issue because there is effectively zero seek time on a random-access drive like an SSD. How about full-drive encryption? Is that going to cause the same thrashing of bits? TrueCrypt, BitLocker, FileVault and so on? Does full-drive encryption affect SSDs?

Steve: Well, sort of. My advice with SSDs and defragging is maybe to just defrag it once. Once a system has all been established, and all the endless security updates are installed, and your apps have been installed on a machine, I would say I like the idea of just sort of organizing it one time, just because, if you've ever looked at the fragmentation of a drive after a full system setup, it's just a catastrophe. I mean, it's a disaster. And if something ever did happen to the directory structure of the file system, then having the files defragged, that is, in contiguous runs of sectors on the SSD could help data recovery software to guess where the file extents are, where the file begins and ends and so forth. So I like the idea of doing it once. It's absolutely something you do not need to do all the time.

Now, relative to whole-drive encryption, when you encrypt the drive, that would run across the entire drive contents once, converting every sector of 4,096 bits, every 512-byte sector of the SSD from plaintext into ciphertext. So that's something that happens one time when you apply the whole drive encryption, and it happens again if you ever remove the whole drive encryption. But in use there is no difference. So it's sort of a little bit like defragging, that is, there's a cost to it to apply the encryption, but not to use it. So there's no additional wear and tear that's occurring on the SSD because it's running through TrueCrypt or BitLocker or FileVault or any of those. One pass across.

And again, I'm overly concerned about SSD reliability, just because I'm overly concerned about data reliability in general. SSDs are solid state. They are robust. They are far more reliable than hard drives, which is why SpinRite exists and how I've made my living for

the last couple decades.

Leo: You know about it. You know all about it. You're the guy to ask, yeah.

Steve: Exactly, yeah. But at the same time, defragging an SSD all the time makes no sense because there is a wear...

Leo: There's no benefit.

Steve: There is a wear factor on solid-state drive technology, which is why they go to all this drive-leveling approach, so that even if you appear to be rewriting the same spot over and over and over, you're actually writing in different physical areas of the SSD so that you don't burn out one particular area. You really can burn them out. I know, for example, Mark Thompson has done so with CompactFlash drives.

And I will say again, you absolutely want to turn off your swapping. You do not want to have a swapping file, the virtual memory on that drive. Typically these days I think the need for virtual memory is diminishing because it's so easy to run two or three gigs of regular solid-state primary RAM on your machine, I often don't have a swap file on any of my machines that have physical drives. It's just becoming less necessary, I think. But you really, it doesn't make sense to have a swap file on an SSD because, even though it would offload your RAM, then you really are exercising that SSD all the time while the system is copying RAM in and out of the drive. So that you do want to turn off. But encryption, I think it's a good idea, and there's no downside to it.

Leo: That's actually really good to know. We talk a lot about SSDs on our This Week in Computer Hardware show. If you're interested in SSDs in general, Allyn Malventano is a wiz on SSDs. And he talks a lot about the various controllers. The SandForce controller apparently is the one he likes the best. There's all sorts of very interesting stuff. And we will be talking a lot about it at CES - another plug - with Allyn. We're doing This Week in Computer Hardware I think on Friday at around noon Pacific.

Steve: Cool.

Leo: Kevin Ottum in Des Moines, Iowa offers some comments on Peter F. Hamilton books, which you and I both love.

Steve: Yup.

Leo: Merry Christmas. Avid listener. SpinRite owner. I heard when you were talking about the science fiction you've both been reading. I really appreciate your suggestions. I've read several. I started with "Fallen Dragon" - Steve and I agree that's our favorite, certainly the most accessible - then "Pandora's Star" and "Judas Unchained" - which is a lot of work, those are long books - and enjoyed them very

much.

A few weeks ago I completed listening to the audio versions of the Void trilogy - which I haven't listened to. He says they were fantastic. Give them another look. I will. In fact, I'll put them on my list. I, too, was a bit put off by the mysticism and religious cult aspects in the synopsis. But in actually reading the books I was pleasantly surprised. I will remain spoiler-free, he says, but I'll simply recite one of Arthur C. Clarke's laws: "Any sufficiently advanced technology is indistinguishable from magic." Being a fan of hard science, I don't think you will be disappointed. Reminds me of Heinlein. Remember how he had cars that would grab energy from the air, and it felt like magic? I think they even referred to it as magic.

Steve: Right.

Leo: But just because it was ill understood. Also, this takes place in the same universe as "Pandora's Star." And since technology allows humans to live practically forever, many characters from that series return. Well, that's good because I loved the characters. Peter writes the best characters. One of the thing I like about him, besides the fact that he's good, hard science, is he's extremely good at characterizations, descriptions. He's a vivid writer.

Steve: So I wanted to let our listeners know, and Kevin, that I'm already up to speed with him and this. I'm still really enjoying the book I'm reading now, the fourth in the series of the Helfort Wars, I think is the name of the series.

Leo: Is that also Peter Hamilton?

Steve: No. But I was wondering about the Void trilogy, and I did a little more digging, and I discovered the same characters from "Pandora's Star" were there, who just, as you said, Leo, I really like. I mean, we have Paula is back, among other people who are still knocking around. And what Kevin said, that any sufficiently advanced technology is indistinguishable from magic, he's implying that, yes, while it looks like mysticism and who knows what, it's actually technology. So I've got the first of the trilogy, which I'm excited about because I'm looking for something really good and really long to keep me occupied while I'm exercising. And no one can do that better than Peter Hamilton.

Leo: Yeah, that's one of the things I really like about long books is you exercise more. And by the way, they are all on Audible: "The Dreaming Void," "The Evolutionary Void," "The Temporal Void." Those are the three?

Steve: Yes.

Leo: Highly rated on Audible, wow. And they're each over 20 hours. If you read all three of them, or listened to all three of them on Audible.com, you'd have well over 70 hours' worth of enjoyment.

Steve: And the good news is, they're all done. It's very frustrating to read...

Leo: Oh, that was hard with - yeah.

Steve: Yes. He does these multivolume monstrosities. And it's like, oh, now I've got to wait for the next one to come out.

Leo: We finished "Pandora's Star" before "Judas Unchained" came out.

Steve: Yes.

Leo: And we were just like [nervous fretting sounds]. Let's go to question #3. Dusan Maletic in Babylon, New York suggests that "do not call" is fundamentally different than "do not track." 278 you talked briefly about "do not track" for Internet browsing and its similarities to the "do not call" list and systems for telephones. One difference hasn't been mentioned: "Do not call" applies to the system where every single individual is precisely defined and numbered, by your phone number, obviously.

Steve: Exactly.

Leo: Hence "do not call" can be implemented without any challenges. But the "do not track" idea on the Internet unfortunately has built-in problems. To be tracked, you've got to be identified. But not to be tracked also involves identification first, then a demand not to be tracked under that identity. You cannot track who you do not know how to track.

So unfortunately, the very act of identifying yourself not to be tracked accomplishes exactly what the tracker wants most: establishing your unique identity. Well, that's an interesting point. Couple that with the inability to easily determine what the other party is doing with your information. With "do not call," you and the government can positively track who called whom and when. But what the backend server does with info harvested by scripts from your browser is unknown to you and the government. The offender can claim whatever they want. So the whole system is fundamentally flawed. The best "do not track" option is an educated user. Dusan Maletic. I guess he's right. What do you think?

Steve: Well, the problem is, first of all, once upon a time, when tracking avoidance was as easy as disabling third-party cookies - which is still 99 percent of the way tracking is done and still a useful thing to do, I believe.

Leo: Oh, yeah.

Steve: Back then, educating the user was really all you had to do. Turn off third-party cookies, you're not going to get tracked. Then of course we got Flash with Flash Objects, and now we've got scripting and HTML5 that actually builds in the ability to create static

tracking capability into the browser. So we're really losing ground here in this battle, which is why I really think that it's going to be some legislation, at least in the U.S., that gives users control of this.

The good news is that - we've seen opt-out technologies, which always annoy me. The idea is, well, if you don't want us to track you with cookies, then go click this button and get a cookie to opt out of being tracked with cookies. Which is really what Dusan is talking about. Now, if that was a unique cookie that you got, that obviously is really a huge problem because then you're being tracked with your unique "don't track me" cookie. Assuming that everyone who says "don't track me" gets the same "don't track me" cookie, then you would be opaque as a group.

But the good news is all it would take is browser manufacturers, or a spec, a do-not-track spec for browsers, to create a new header, a query header. We know that there are query headers like host, and expires if, and the main URL, and of course cookies, and a number of things which the browser is able to send out with a query. All we need is a universal definition of another header, which would be "do not track." And so if the user says they don't want to be tracked, they configure their browser to only issue queries with that "do not track" header.

Now, it's true that it's completely up to us trusting the other side not to be tracking us. That is to say, does that mean we won't accept cookies from them? Or we will accept cookies from them, but they're not going to track us with those cookies? I mean, and this is why I think ultimately legislation is going to have to be put in place to criminalize, I mean with some serious penalties, the tracking of people who have explicitly made their intentions clear that they do not wish to be tracked. And so we need the browser to work on our behalf, as our agent, to make that assertion. For every single query we make, it would be flagged with "do not track this." And then we have legislation at the other end that enforces policies of those who otherwise would wish to track us in order to give our request for non-tracking some clout.

And then we've got the problem of opting in. Because the argument for tracking is, which I've always found a little specious, frankly, is oh, well, we need this to generate revenue. So our advertisers are only going to pay us if we allow the people who visit our site to be tracked. So it's like, well, okay. I mean, it's one thing to see ads. I've never been convinced that this whole customized ad concept works. Have you ever felt, Leo, that you're getting ads meant for you when you go to random sites?

Leo: Well, when I go to my Gmail, it'll give me ads that are based on the email I'm reading. Whether that means...

Steve: Which is very different, of course.

Leo: Yeah, yeah. But that's all they can do. I mean, okay. So I'm looking at an email I got about CES and somebody in school. And the ads were day trip to Kyoto, Antarctica expeditions - I am going to Antarctica. It doesn't mention that in the email, so obviously it's keeping track of that. See polar bears for less. European cycling tours. Disney's performing arts. Transatlantic tips. It must know that I'm going on a cruise. So it is targeted. Those are Google ads in Gmail.

Steve: Right. And so I don't have a problem with that because they're using the context

of what they know about you, not going out to a third party. And so the argument has always - the pro-tracking argument has been that, because they will learn about who people are over time, the ads that are served by third parties will be more relevant to you. And I have never experienced that. You know what I mean?

Leo: Well, they'd like to. Look, let's face it. That's what they want to do. It's not that they're trying to appease us or somehow blow smoke because advertisers don't want to waste energy on advertising to people who aren't interested. But you're right, I just went to Mashable, which has a lot of ads, and UC Davis is offering me a degree. I guess that wouldn't be appropriate. When it comes to support, oh, here's a web server. Yes, you're right. And I think that that's often the case. But that's just because their system ain't working.

Steve: Well, exactly. I think all of this is - I've always found it questionable that tracking actually works at all. Yet it's clearly a privacy violation. So the whole concept seems ill-advised and lopsided.

Leo: Right. You're distinguishing that from what Facebook and Google do because Facebook knows about you.

Steve: Yes.

Leo: And Google knows about you.

Steve: And Google search, when you go to Google, and you put in a bunch of keywords and search...

Leo: Right, it's tied to the search.

Steve: I mean, that's a brilliant, I mean, that's why Google is the size they are. It's a brilliant use of immediate, oh, look, here's some paid ads down the column, and some things on top. And I know what they are. I know that they're sponsored insertions out of my search results. I mean, I have no problem with that. Very different from, exactly as you said, going to some random site that has ads and expecting them to be as relevant to you as opposed to somebody else. I just - I've never seen that actually functioning, even in the days when I wasn't fighting tracking.

Leo: Right. Your point is tracking cookies do nothing.

Steve: I don't think they do anything but upset everybody.

Leo: Right. Good point. Good point. Moving to #4, Pete Burtis in New Hampshire: One more thing for your list of "must haves" before you'll implant a chip. We talked

about putting RFIDs subcutaneously for identification purposes. Quick and simple: I want any implanted chip I have to be able to authenticate the remote device that's trying to authenticate me before it does anything else. Let me load the public keys of devices I trust onto my implanted chip; or better yet, let me load my company's root certificate onto my chip. And then I'll automatically trust every RFID door lock at my company. If done correctly, this makes tracking you from a distance by your chip impossible because it'll ignore queries from any devices it doesn't trust, and also addresses things like replay attacks. Sure, your device memory might be an issue. But if it's implanted anyway, why not just make it a little bigger? Oh, dear. He says: A 16-gig microSD card would implant nicely between my thumb and forefinger. Thanks for the great show. Pete.

Steve: Well, first of all, that is an absolutely great idea. The problem is that it's not just memory, it's processing power. And public key crypto is very expensive in terms of processing power. We now take it for granted because we've got chips in our machines that are running at 4 GHz. But don't forget they've got big honking heat sinks on them with fans rapidly taking all the heat that they generate off as quickly as possible. So the only reason this technology, the public key crypto is feasible, is that we can afford to dump a huge amount of electricity into the chip, which is going to turn into heat while it's doing this processing work.

What's inherent with an implanted chip is that you certainly don't want a battery in there that is going to leak chemistry into your body, and also expire after some length of time. So all of these chips are, from a power standpoint, they are passive in that they work as a transponder. That is, they are excited by the magnetic field which briefly powers them enough to just barely receive something and send something back. So at least for now, we just don't have the state of the art to bury a chunk of a computer in us. I mean, I guess you could find someplace it would fit. But then, still, the problem is power. You really, you don't want something highly invasive, contacts on your skin or a socket.

Leo: It's like a pacemaker or something.

Steve: Exactly. So powering it is the problem. I will say that, having thought about this a lot since we did the tracking podcast - and listeners I credit with reminding me about Bluetooth. For me, I think Bluetooth solves a lot of my problems because I do have a Bluetooth-enabled Blackberry, and I could easily set up a little receiver in the garage so that, instead of a key on the garage door, I just have a button. And the button is only enabled when my cell phone is nearby, and that's in my pocket. So it sort of gives me - and the same thing for the front door. So it gives me sort of the best of what I was trying to achieve, and I don't have to wait for any scar tissue to heal.

Leo: There's another technology called Near Field Communications that Google is really pushing heavily, NFC. In fact, my latest phone, the Nexus S, the latest Google phone, has NFC built into it. There's very few places you can use it now. But it's not - it wouldn't be very good for a garage door because you'd have to get out of the car. It doesn't have much range. It has a meter or less.

Steve: Right, and that's by design. It's very low power.

Leo: Right. I kinda like that.

Steve: Yes.

Leo: And because it's tied to the phone, you could have certificates. You could have all sorts of information going on. You've got a very powerful processor behind you.

Steve: Plenty, plenty.

Leo: You've got lots of memory.

Steve: That's plenty of technology.

Leo: I suspect that's going to win over implantation of chips. Just, I don't know. Call me crazy. Let's see. Question #5, Scott in Winters, California. He wants to know about DDoS and spoofing the source IP. Very few people know more about DDoSing than Steve Gibson, having fought it. Steve, in last week's listener feedback you mentioned the DDoS - what is that?

Steve: Distributed.

Leo: Distributed Denial of Service attacks and software being used by the WikiLeaks-related attacks - Anonymous was going after people like PayPal and Amazon, who had banned WikiLeaks or canceled their accounts - and how the IP addresses of the attackers are not hidden with the method that Anonymous was using. One of the more popular tools for DDoS is a network utility called Hping at Hping.org. It lets you send a flood of packets, a DoS attack, and spoof the originating IP address if desired. Oh, we're getting to raw sockets, I have a feeling. You can literally say you are Microsoft.com or 192.168.1.1 or whatever you want with a single command-line option: "-a --spoof hostname:". Use this option in order to set a fake IP source address.

This option ensures - it's okay for us to tell everybody this. I guess everybody knows it. Anybody who wants to know it knows it. You could Google "DDoS." This option ensures that the target will not gain your real address. However, replies will be sent to the spoofed address. So if you're looking for your ACK, forget it. Most of the time people don't care about the ACK. In order to see how it's possible to perform spoofed idle scanning, see the HPING3-HOWTO. Yeow! he says. I like that.

Steve: Well, I got a big kick out of the fact that it has its own domain. So this is not - Hping.org, there you go.

Leo: Yeah. Anybody wants to do any DoS attacks, here's what you...

Steve: There you go.

Leo: So does it have to use raw sockets to do the spoofing?

Steve: Yes. It would need to. So it would be unable to do that on Windows platforms.

Leo: Thanks to Mr. Steve Gibson, by the way.

Steve: Yup, thanks to me.

Leo: The reason I want to give you credit for that is you got so much heat, so many websites, oh, Steve [mumbling]. I guess Windows fanboys were mad that you were attacking Microsoft, saying you've got to take this raw socket stuff out. It's in UNIX; it's in more powerful operating systems. But by putting it in a consumer operating system, they were just turning Windows into an attack machine.

Steve: Well, and it was a mistake. I really, I sincerely believe that it was just something Microsoft hadn't considered.

Leo: It was on their checklist of capabilities of Linux, well, we've got to have that.

Steve: Well, yes. And their 95, 98, ME, sort of that were originally their more end-user consumer line, never had full raw sockets. You were not able to just generate any packet that you wanted to and stick it on the Internet. The stack, the so-called TCP/IP stack, it provided your local machine's IP address automatically as that packet was leaving the machine.

What happened was that Microsoft ended the life of what was sort of originally the 16, then the 32-bit machines, and they took NT, which they had evolved into Windows 2000, and then they evolved that into XP. And so my concern was that they were taking a more of a commercialized operating system, which did have full raw sockets, that technology existed in Windows 2000 and in NT, and without thinking, they gave it the XP candy coating, making it look more friendly because Windows 2000 was more of a server platform originally. And so they were going to be putting it out, it was going to be preinstalled in all these machines and laptops, and it just didn't need this capability to allow software on the machine to just arbitrarily set the source IP of the packets that it was sending.

There is no, in the original Internet design, there is no defensible real reason for ever lying about the source of a packet because the whole point is communication. And you can only have that if the remote end, the recipient of your packet, knows how to communicate back to you. So it made sense that, I mean, it's the reason the earlier Microsoft operating systems worked so well is that they just did this for you. They didn't have any problem doing anything on the Internet. They were full Internet citizens without this capability.

And so my argument to Microsoft was don't let this go out into the mass market because

it's just going to cause trouble. And it really did. In fact, it was the MSBlast worm that blasted Microsoft with their own raw sockets, at their own IP address. And it was after getting burned by that they finally understood, oh, that's what Gibson meant.

Leo: Right. And there are powerful operating systems that you can do the things you need to do if you need raw sockets. But let's not put it in a machine that's ill protected, in the hands of a consumer that doesn't understand security.

Steve: Well, and look how many machines are infected with bots now. All of those bots would be capable of dramatically worse attacks. Now, the fact is, many sites were taken down by the Anonymous group using their tool which did not do spoofing. And so that was one of the arguments against my argument was, wait a minute, you don't need spoofing to do denial of service attacks. That's true. But if you don't use spoofing, then you can be backtracked. And that is what happened for users who unwittingly were wanting to support the attack on people who were acting against WikiLeaks. They were using the, what was it, it was that low earth orbit thing...

Leo: Yeah, LOIC, yeah.

Steve: ...LOIC, in order to attack people who the group Anonymous wanted to go after. And the consequence of that was that they were creating standard TCP connections and just moving payloads of data. So it was just a data saturation attack. But to create a TCP connection you have to have a roundtrip. And if you're going to have a roundtrip of data, then you are by definition having to disclose your IP in order for that roundtrip to get completed. So it's, yes, there are tools like Hping out there. They run under Linux OSes are typically where they're being used. And they work well. And the good news is that it's becoming common enough that this is no longer fringe available. That's why I didn't mind talking about it here.

Leo: Question #6, Mr. Gibson. And this one comes to us from Jim Stevens in Massachusetts: Thanks for the incredible podcast. I've been a big fan since the beginning. In a not-so-distant episode, Steve described his frustration regarding the losing battle we all face attempting to make our machines secure. I personally am sick of spending half my life making sure friends' and families' computers receive the latest updates for all software, educating them about Sandboxie and NoScript, and trying to explain why antimalware software in general is a "reactive" approach to problems, not a proactive approach. Life is too short for this. The analogy I use for malware is cancer. Once you have it, it's almost impossible to get rid of. It's much better not to get it in the first place. I wouldn't disagree with that.

My question: You recently spoke about completely re-architecting our machines to prevent security problems in the first place. I think we talked about the Turing or the von Neumann model where data and code are combined versus other models where they're not.

Steve: Right, the Harvard architecture.

Leo: Harvard architecture. It seems a major cause of our problems are caused by buffer overruns in stack space. It's true. In your stack episode, you described how the stack is used for both the return address for functions and for data, for local variables. If too much data - and by the way, often for code, as well. If too much data is written to a local variable, it can overwrite the return address of the function and, if carefully designed, could cause undesired code to be executed off the stack.

Why don't we have two stacks? One stack contains only return addresses for functions and would allow recursion and all the other functionality we use today. The second stack could contain data for local variables. Overflowing a buffer, while bad, would just cause data corruption, and not unintended code execution. Of course, we'd have to synchronize the two stacks so that the correct local variables are popped when a function returns, but this problem seems minor compared to the frustration we have to deal with now. What do you think? Jim Stevens, reinventing computer technology on the fly. What do you think?

Steve: Well, I would say that's a useful idea.

Leo: It's a restating of that same concept of not mixing data and code.

Steve: Yes. And it's also the case that, if we were really good about enforcing the structures we have now, for example, we have data execution protection. And in the Intel architecture is the so-called "non-execute bit," where any page of memory can be flagged as this should not be executable. So even though we don't have the physical architecture of a Harvard machine - the Harvard architecture was named after the university, Harvard University Mark I computer, which used paper tape and relays. So it was, like, a really early machine. But there was - so there was no concept that instructions and data were the same.

All of the machines we have today, even like when we were talking about my favorite old PDP-8 back in the early '70s, that's a single block of memory which is homogeneous. It contains instructions and data. The instructions can refer to themselves, to their neighbors, to data. And there's nothing, there's no division between instructions and the data. In a Harvard architecture, the hardware itself enforces this differentiation so that instructions, even if they're, like, referring to their own location, they're not referring to themselves, they're inherently referring to data in another physical space, which may happen to be the same offset, the same address as the instruction. But the instruction can't refer to itself.

In fact, I remember some time ago we were talking about a voting machine whose design I liked and admired because, even in this day and age, they deliberately created a Harvard architecture in order to increase the security of the voting machine so that no instructions could be executed out of the data space. There was no way for data to, like, bring instructions along and modify the instructions that were in a physically separate place.

So, first of all, if we just did what Jim suggested, that would be a solution. If we kept return addresses in a separate space, not on the same stack as data - and in fact there are microcomputers, smaller machines today that have their own return stack separate from a data stack. Not really for security, this was more done just for cost saving and because they are sort of toy machines, not big industrial PC mainframe sort of machines

like what we're all sitting in front of. So Jim's idea would work.

But as you said, Leo, there are many ideas like this. And essentially we have all the tools we need now, even with the Intel architecture, which is a von Neumann architecture, where everything is mixed together, because we are able, if we only did it right, and if we only did it consistently and thoroughly, we are able to cause data never to be executed. And remember, we talked earlier, too, about the return-oriented programming where bad guys are able to avoid data execution protection by executing the little tail ends of existing subroutines. So there's another example where even this doesn't solve the whole problem because it's really a big problem.

Leo: Question #7 is Daniel in Provo, Utah: You mentioned in Episode 278 that hard disk-encrypting ransomware is making a comeback, this time with public key encryption. It seems that for something like an entire hard drive, known plaintext attacks may be helpful in determining the decryption key even if cipher block chaining is used. So the potential quantity of known plaintext on a whole hard drive makes it very likely that enough information can be determined to make a strong attack and get back the original data, especially given good reverse engineering of the code that did the encryption in the first place. In other words, the bad guys who aren't using strong encryption aren't going to be very effective if somebody really needs to get back in.

So the best remedy to avoid getting snared by these guys and to use whatever governmental remedies may exist to get their operation shut down and provide the necessary disincentive against similar operations in the future - oh, that's the best remedy. Sorry. The best remedy is to avoid getting snared and shut their operations down. But the technological approach may still be helpful for some.

Thank you so much for the podcast and your other great products. I found Security Now! a few years ago and made a point of going back and listening from the beginning. It's been instructive and entertaining. What is he trying to say? I'm not sure I understand.

Steve: Well, he's suggesting that one of the known ways that cryptography can be attacked is if you know what the non-encrypted data is. And we were just talking about that earlier in this podcast relative to GSM. With the GSM system, the technology they use is a simple XORing of their cryptographic stream of pseudorandom data with the plaintext to get the ciphertext. And it is because so much of the GSM protocol is known that, if you XORed that with what was in the air, you'd get back the cipher stream. That's the Achilles heel of GSM when you've got those rainbow tables that allow you essentially to reverse a one-way function, go the other way on the one-way function.

What's different here is, assuming that the guys that did the ransomware did a good job, this doesn't really apply. First of all, think TrueCrypt. TrueCrypt is whole drive encryption. It is, for example, what the ransomware people might use. And it's open source. So here you've got an industrial strength, I mean, TrueCrypt is as good as we know how to make hard drive encryption. And it's open source. So bad guys, I mean, as far as I know they did. I have no knowledge of that one way or the other. But if you have TrueCrypt out there, and they've solved all the problems for you, why not just take it and use that cryptography?

So the way TrueCrypt operates, even under a strong cipher with, like, AES, 128 or 256-bit AES, is it's taking blocks of 128 bits at a time and encrypting them. And it is probably

using, I don't remember now whether TrueCrypt uses cipher block chaining, but that's the technique where you use the results from the prior encryption, mixing it in with the next one, which creates a dependency from the start all the way down through the end of the sector, so that it's not just like you're encrypting each of the 128-bit blocks standalone, so that they would not be interdependent, but instead you're, immediately after the first block, you're mixing the results of that into the second. And it is, fortunately, a reversible process, allowing you to perform decryption.

So the result of that is that we're sufficiently removed from the environment of GSM, that even if you had, even if you knew a huge amount of what was on the hard drive, you're still not going to be able to reverse the enciphering that ransomware does, anymore than you could reverse what TrueCrypt has done. And we know how strong TrueCrypt is. Governments have pounded on it, trying to decrypt the contents of bad guys' hard drives. And unfortunately, in that case, they've been unable to. It's just it really is the case that even a known plaintext attack against a hard drive is ineffective if the encryption is done correctly. And with TrueCrypt we have an open source model of how to do it correctly.

Leo: Question #8, Charles in Houston, Texas, he's wondering about something called FHSS: I just purchased a Lorex LIVE Snap wireless video baby monitor because I found it on sale in a local store. It looks really convenient. I haven't opened it yet because I'm not very sure about the security of the wireless video feeds from the cameras. That was a problem some years ago. Somebody noticed that there was no security on those videos. You could drive by and see anybody's baby monitor. And the company's not responding when I ask for more information.

The fellow I spoke with on the phone sounded like he was reading the information for the first time himself. He was in a call center, didn't seem to be a company employee. And they haven't responded to email. Well, I'm sure they're a Chinese company. I mean, c'mon. How much did you pay for that baby monitor? The only information I find about the product from their website is that it uses something called "FHSS," and they advertise that, "The long-range digital wireless signal has a range of up to 450 feet with clear line of sight and is secure and interference-free. It won't interfere with your cordless phone, microwave, or router, and no nosey neighbors will be able to eavesdrop on you and your family."

Their lack of response of specific details violates my TNO policy. I need more information. My look into FHSS doesn't give me much of a warm fuzzy. On the other hand, it isn't just a simple broadcast on one frequency that anyone could tune in to. The convenience factor for this system is why I haven't returned it and moved on. What are your thoughts? Here's hoping you can help me before it's too late to return the system. Charles.

Steve: Well, the acronym he's using, FHSS, is well known. It stands for Frequency Hopping Spread Spectrum.

Leo: Oh, well, that would be effective.

Steve: Yes. So long as the bad guy doesn't have the same receiver and/or the same key. I tracked it down, looked at the website, looked at the documentation. The fact that they're talking about using frequency hopping spread spectrum, that's something you do,

you get security from it, but you probably do it more for interference prevention because the idea, just as we were talking with Bluetooth, Bluetooth is also a frequency hopping spread spectrum technology.

In the Bluetooth case, we'll remember from a couple weeks ago that the hopping, the frequency hopping sequence, which has to be known in order to eavesdrop, is based on essentially the equivalent of the Bluetooth device's MAC address and the secret key which was established between the endpoints during the Bluetooth pairing. At that point, then, they're able to use the clock from the master of the Bluetooth in order to synchronize themselves and agree about which frequency they're going to be on from one packet of Bluetooth to the next.

The question is, how has this technology in this I'm sure inexpensive - as you've commented, Leo, and we don't know who designed it - baby monitor, how has it been designed to create a unique hopping sequence between any two pairs of monitor and receiver, if it has been? It seems to me unlikely that it has been. It's probably got a clock. The receiver probably looks at the clock and essentially has a hard-wired frequency hopping sequence. Again, I have no knowledge of this. I don't know for sure. But I wouldn't be at all surprised if somebody else with the same monitor could receive your signal from yours.

Maybe not. Maybe there's some pairing procedure, or they come from the factory pre-keyed in pairs so that you can't buy another pair, and they would be the same. That would be great, if that were the case. In which case, if there's any evidence of that, that there's a pairing between them of some sort, same code, for example, then I would say it's very secure. Absent that, if another one of the same make and model monitors could receive the same signal, then it's obviously the case that all of them are running the same frequency-hopping, spread-spectrum sequence, in which case you've got good immunity from noise, but no effective security. Probably they're set up as pairs, I would hope, in the factory, which would make them a nice device.

Leo: Yeah. I mean, c'mon, what are they going to see? Presuming you're not wandering around naked in your baby's bedroom.

Steve: Right.

Leo: And you'd have to work pretty hard to get it. Aaron in Oregon says "about 154 nonillion years." That's a number I don't hear a lot, nonillion. That's how long the website howsecureismypassword.net says a desktop PC would take to create your Perfect Passwords, your GRC-generated passwords, which are, to refresh people, 64 bytes, right, 64...

Steve: Characters.

Leo: ...characters long, including alphabetic and punctuation and numeric; right?

Steve: Right. So it's saying how long would it take to crack a GRC-generated password, is what that howsecureismy password.net...

Leo: And I presume he's talking about something like RSA or some sort of prime number encryption. But the question is, how do you calculate it?

Steve: Right.

Leo: That's his question.

Steve: Right. And so the good news is, first of all, it's a fun little site. I would commend our listeners to go check it out. It's howsecureismypassword.net. And don't take it very seriously, but it's cute. You have to have scripting enabled, or it won't do anything. In fact, it tells you in huge lettering on your screen that you have to turn scripting on, if you don't. Even so, the lettering is pretty large. And it's very cute. It runs JavaScript. And you type a password in. And if you use any of 500 most common passwords, then it'll say, well, that's one of the most common passwords, so that's not going to take long at all. Otherwise, as you enter your password, it's continuously estimating how long it would take to crack. What's nice is that the source code for the whole thing is provided, so I was able to look at it and answer the question, how does one calculate something like this?

Now, the fact is there's no set rule for how you calculate this. This site is doing a nice job, but not a fantastic job. What the code does is it looks at the string you have entered so far. And if it finds anything that is lowercase, then it assumes you have an alphabet of 26. If it finds anything that is uppercase, it assumes you have an alphabet of another 26. If it finds any digits, then it assumes that you have an alphabet of another 10 because you've got digits zero through nine. If it finds any of a set of special characters, 13, like exclamation point, at sign, pound sign, dollar sign, percent sign, and so forth, then it assumes you've got special characters in your password from an alphabet of 13.

So that tells it in its estimate, in its estimation, how large the alphabet is from which the number of characters you have entered are taken, and it does some simple math to figure out how many combinations then. Essentially it raises the alphabet size to the power of the number of characters you've typed, which would give us the total number of possible combinations of that password. Then it assumes that your computer, your PC-scale machine, can test 10 million of those a second, which is pretty fast, depending upon what it's actually trying to do. I mean, that's a lot of testing. If you were cracking into a website, you certainly can't do 10 million attempts per second. So again, that's why I don't take any of this very seriously.

Also this doesn't check to see whether you have alternating upper/lowercase, how many special characters you have, how many digits. If you have one digit, it doesn't change it, really, or it doesn't give you a different calculation than if you had two digits. So it's a sort of a simple-minded approach. Mostly what I was interested in was what the heck is "nonillion." And...

Leo: It's like, well, billion, trillion, quadrillion, quintillion - how many zeroes is in a nonillion?

Steve: So the code answered that question for me because he has a thousand years, then we've got a million, then a billion, a trillion, quadrillion, quintillion, sextillion,

septillion, octillion, and then nonillion.

Leo: And then there's decillion, and I don't know what you do when you get to 11.

Steve: Yeah. So...

Leo: It's Latin.

Steve: But for what it's worth, nonillion is probably about as far as you need to go. First of all, we're all long dead many, many nonillion years before this thing has cracked your password. And if 154 nonillion years is the result of this calculation for one of GRC's passwords, which is 64 characters long and looks like absolute gibberish with all of those different character sets engaged, then I don't know how you get anything more complicated than that.

Leo: Sufficient.

Steve: Yes.

Leo: Sufficient is the word.

Steve: It is, indeed.

Leo: Our last question, Steve. Jared in Australia doesn't understand why software flaws are so hard to find. Gosh darn it, every day, every month, a new flaw. Regarding the possible trouble with BSD's security from as far back as 10 years ago - by the way, I think we're all in agreement that that was BS.

Steve: Yes. Not even D. Just BS.

Leo: No D. I don't understand how it's even reasonable for something that might be wrong to exist within BSD's code for that long. Why wouldn't people know? It's open source. Why aren't these problems just seen and fixed, if they exist? What am I not getting? Well, don't confuse open source with closed source because that's a big difference.

Steve: That's a big difference. But of course it is the case that - and I've mentioned this before. And this also leads us into next week's topic, which is this fuzzing technology, which is very interesting and very powerful. You may remember, Leo, that back when we were talking to the guys at eEye, that they had a whole roomful of machines that they were just - they were, like, throwing parameters at different network protocols and finding bugs when the machines crashed. So they were keeping a log of what was going on. And when a machine would crash, they would say, ooh, that's not good. What did we

just ask it that made it crash? And can we leverage that into an exploit? That was one of their approaches.

So that's sort of the opposite side of the spectrum from sitting down with the source code and staring at it. And the reason that's effective where staring at the source code isn't is that source code is communication. Source code tells you what it's trying to do. And so when you're reading it, you're sort of agreeing with it. You get sucked into its own intent. The intent of the programmer is communicated through the source code. In fact, the better the coder is, typically, the better their intent is. We've all seen examples like Forth, which is very difficult to read. Now I'm going to get hate mail for having said that.

Leo: If it's properly written, it reads just like English. In reverse. Always.

Steve: And what was that, there was a wacky, oh, PL/1 was another language.

Leo: PL/1, yeah. APL you had to have a special keyboard.

Steve: And APL. And in fact there were contests. I'm sorry, I meant APL, not PL/1. PL/1 was actually very Pascal-like and easy to read. It was APL. There were contests to see who could, like, do the most work in a single line of APL. So incredibly dense, very powerful operators that didn't communicate clearly. But most code, and you see this in open source code, the programmer is trying to communicate, is actively trying to communicate what the code's intention is for the purpose of, at some point in the future, communicating it to other people. And so fuzzing, which is our topic for next week, is sort of the opposite side of that. It knows nothing about the programmer's intention. It's just trying to see what it can find.

And so, to answer Jared's question, as a non-programmer, I can really understand why it would be confusing that something just - you could have these bugs in code that no one can see. And I'm, as a developer, having done a lot of bug-hunting myself, I know that I get seduced by what I'm reading. I'm reading the intention of the programmer rather than the detail at the level that the computer executes it. And there's a surprising difference between those two things.

Leo: And so this leads us to next week.

Steve: It does indeed. Fuzzing. Fuzzy-wuzzy.

Leo: Fuzzy-wuzzy.

Steve: Fuzzy-wuzzy had a...

Leo: Fuzzy-wuzzy [indiscernible].

Steve: ...had a computer.

Leo: Steve, well, it's been great, a great way to start 2011. I love it. If people have questions for next time, which will be Episode 284, if all holds true, you go to GRC.com/feedback, and you can ask those questions there. It's not exactly an email, it's a form.

Steve: Yup.

Leo: Makes it easier for Steve to parse. But you can find many other things there, so it's worth going, including SpinRite, the world's finest hard drive and maintenance, recovery and maintenance utility. Everyone should have SpinRite, if you've got a hard drive. Also let's not forget all those freebies Steve gives you, that great software, including the DNS benchmarking utility, I love that; Perfect Paper Passwords, his passwords utility. GRC, as in Gibson Research Corporation, dotcom.

Steve also has the full show notes there, the 16KB version for the bandwidth-impaired, which he himself does, edits himself with his own little hands; the transcriptions, which he pays for himself because Steve is devoted to you. You'll find it all at GRC.com. Steve, thank you so much.

Steve: Thanks, Leo. And have a great week in Las Vegas at CES.

Leo: We're off tonight.

Steve: I'm sure I and our listeners will be following you closely. Looking forward to it.

Leo: Can't wait. Thank you, Steve. We'll see you next time on Security Now!.