Going Random, Part 2 of 2

Description: After catching up with the week's security and privacy news, we conclude our two-part series discussing the need for, and applications of, random and pseudorandom numbers. We discuss the ways in which a computer, which cannot produce random numbers, can be programmed to do an extremely good job.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-301.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-301-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 301, recorded May 18th, 2011: Going Random, Part 2.

It's time for Security Now! the show that covers your security online; your privacy, too. And here he is, our security guru, Steve Gibson from GRC.com. He's the man behind SpinRite and many other wonderful inventions. Can we call software an "invention"?

Steve Gibson: Oh, unfortunately, too many people do.

Leo: Yeah, that's a hot topic lately, issue of software. You never did patent any of your stuff, did you?

Steve: I never did for myself. I did some subcontracting for a while where the people I was doing the work for said, oh, we've got to get patents on this. And in fact I did apply for one patent on a CryptoLink technology which would allow it to stealth its open ports so that you could be running a CryptoLink server, but no bad guys could see it. Essentially it's a single-packet authentication technology. I did get the response back from the patent office just recently; and they said, ah, this looks like bunk.

Leo: Really. They refused it.

Steve: Oh, yeah. Apparently that's the dance you do. And it of course enriches the attorneys because so far I've already spent $10,000 on this thing. So it's like, okay. So I have on my list of things to do is to figure out what it is they're complaining about. And

then this is just the way it works now is you say, no, that doesn't apply because of this, and you apparently didn't read paragraph 3B because that's what's different about this, which makes this unique. I mean, we did a full search, and nobody else has done this before. So it looks like it's a new thing. I'm only doing it, though, because there's some provision in the tax code that allows income derived from the sale of all substantial rights and title to a patent to be treated as long-term capital gains.

Leo: Oh, that's good.

Steve: And that's actually the way I sold my light pen to Atari was I had patents on various aspects of the hardware. And so I was able to get the income that I received treated as capital gains. It was much more beneficial than regular income.

Leo: Much lower tax rate, yeah.

Steve: But my CPA has recently told me that I'm too close to GRC for me to license it to the corporation because I own all of GRC.

Leo: You can't really license it to yourself.

Steve: So I can't. So frankly, I'm probably just going to abandon the effort. The only reason I was doing, I mean, I absolutely don't care about, and I'm actually sort of against, patentability. And I also thought, well, it would give somebody something to own someday if they wanted to buy it from me. But that's probably never going to happen, either. So it's like, I don't think I want to keep sending money to the attorneys in wheelbarrow-size quantities.

Leo: No kidding. I know that feeling. So today we're going to complete our Going Random conversation from two weeks ago.

Steve: Exactly. We started, sort of laid the foundation for why it is that crypto systems need randomness. I'm going to review that briefly after we catch up with the week's news, but then talk about how we get randomness from machines that are designed not to be random, which of course is our computers. Every time you multiply or add, you want the same result. So how can you ever get a machine to give you something random? And the answer is you can't. But that's not good enough. So we still need it, even if we can't have it.

Leo: But we're going to get it.

Steve: Yes, we are.

Leo: Steve will explain how. Now it is time to get the latest news of the security

world from Mr. Gibson.

Steve: So there's something interesting going on with Mozilla that it's not really an update, but it's a planned update. Essentially, Mozilla is getting a little annoyed with the fact that 12 million people are still back on Firefox 3.5.

Leo: How many?

Steve: 12 million. They're the people who do not listen to us, although I can't really say much because I'm on 3.6 and not willing to move until I'm sure that 4 settles down and all the add-ons that I cannot live without now are compatible with v4, which...

Leo: Is this analogous to Microsoft moving people off older IE versions? Is 3.5 insecure in that way?

Steve: No, I think it's a matter of the problem of, I would say, one of the other reasons Microsoft moves people off of old versions, is they just get tired or stretched too thin of supporting them because, I mean, Mozilla is an open source project. And it's being done for the good of humanity. But for people to stay back on 3.5 as they're, for example, fixing things that come from the common code base that all versions of Firefox share, they've got to take the time and trouble to sort of remember what 3.5 was and then make the changes that are relevant to 3.5 when they're also fixing 3.6 and 4.0. So here's the news: 5, version 5 of Firefox will be released on June 21st. So here we are, this is May...

Leo: What? They just did 4.

Steve: I know. So here we are on May 18th. And so literally four weeks from now we're going to have 5 already. So what they're going to do is, on June 21st - and their goal is to get everybody off of 3.5 by the end of June. So on June 21st, with the release of v5 of Mozilla Firefox, on that day they will release the next version of 3.6. Right now I'm, for example, using the latest, which is 3.6.17. So on June 21st they will release 3.6.18 - or one eight, I feel more comfortable saying it that way because that's right. And that will be offered to everybody on 3.6.17, like the one I have now, but also those using 3.5.19.

So the point is, and this is the first time they will have ever done this, they're going to be automatically updating people to a major revision change, not just a minor revision change. So the good news is 3.5 and 3.6 are very close to each other. But essentially, they're not going to be asking people or requiring 3.5 users to do anything because they've come to the conclusion these people aren't. I mean, they're just sitting back on 3.5 and staying current with that, but they're not making the move to 3.6, very much like I'm staying with 3.6 and haven't made yet the move to 4.0. But with this on June 21st, they will silently upgrade 3.5.x users to the latest 3.6, essentially retiring the whole support and forward-moving updates for 3.5. So, I mean, if that weren't enough to move people off of it, if they were security conscious, and Firefox users tend to be, then the fact that they're just making it automatic would do that.

Now, what's interesting is they've got a bug, it's Bug 650030, which prevents this from working. So on their bug list, the bug's title is "major update automation should support background updates, not just advertised ones." And so what they call a "major update" is, for example, a 3.5 to 3.6 change, as opposed to updating the next decimal dot versions. And "background updates" are those where the next restart of Firefox automatically brings the user to the next version. So Firefox downloads it silently in the background and has it prepped and ready so when you launch it, it just is running the newer one. And then what they call an "advertised update" is one where the user is prompted and needs to accept that before downloading and applying the update.

So they need to fix this problem that they have. And when they do, they will update Firefox silently, and we're going to be saying goodbye to 3.5 on that day and hello to 5. Which as you say, I mean, I was surprised, too. It's like, wait a minute. We had 3 for a long time. Although I did see some things - you may have run across it to, Leo - where there was some commentary about the Mozilla Project being too slow, that they were just moving too slowly and weren't keeping up with the pace that we're seeing, for example, from Chrome, which is really moving itself along, and even IE. So I think they're deliberately working to improve their rate of adding features.

Leo: It's kind of silly because, especially on Chrome, version numbering doesn't really mean anything. I mean, what's the difference between Chrome 10 and 11? You tell me. I can't tell.

Steve: Right. It's just changed.

Leo: It's just changed. So is there a version race? I mean, that's crazy. A version number race? That would be crazy.

Steve: I think it's not so much numbering as the Mozilla people themselves feel like their whole process had become somewhat bogged down. And so they're working to streamline the process so that they're better able to just get stuff vetted and tested and beta tested and out to users so that, I mean, I'm looking at Chrome thinking, hmm, you know, looks pretty good. Of course you thought that a while ago...

Leo: Oh, I love Chrome, yeah.

Steve: ...and made the jump.

Leo: Yes, I'm a big Chrome fan, yeah.

Steve: So we've got some bad news in the Mac space. Ed Bott, whom you and I have known forever, who's been in the PC business forever, has a column or a blog over on ZDNet. And he managed to get a hold of an AppleCare employee who provides the AppleCare support, and sort of officially but anonymously interviewed this person about what the AppleCare people are experiencing relative to this Mac Defender problem that we've talked about. We introduced Mac Defender on this podcast a week or two ago, noting that in the last couple weeks it was becoming a growing problem for Mac users,

that Mac users were not accustomed to fighting the virus wars that Windows users have been fighting for years.

And what was really interesting - so Ed Bott in his blog posting posted a sanitized dialogue that he had with this AppleCare employee. And one of the things that really caught my eye I thought was interesting, which was that, because Mac users tend to believe that their Macs are impervious to these kinds of problems, they consequently tend to believe the Mac Defender pop-up - which sometimes also calls itself Apple Security, but it's basically the same code with just a different name. They tend to believe those to a much greater degree than Windows users do because Windows users are so beaten up at this point by all of this nonsense that our guard is up to a much greater degree. So the fact that Macs haven't historically had a big problem with viruses is giving these newly created malware for Macs sort of a boost, sort of a head start, more than would otherwise be the case, which I thought was really interesting.

What came out in this dialogue with this AppleCare Apple employee was that it is true that the Macs are more secure at this point, or at least to this first-generation malware, by needing to have the admin password in order to install themselves permanently in the machine. Yet social engineering is still being effective. And what we are seeing in general, I mean, if we looked at the thousand-mile view of the history of the last 10 years, what we're really seeing is, as security is becoming more of a focus for people, as people are in fact doing a better job with security, and our operating systems are getting tightened up, and there are just today far less obvious openings for malware to get into machines, it is social engineering, even over on the Windows side, which is still catching people out and getting them to install these things.

So, for example, the Mac Defender problem, this employee said that in the course of two weeks he went from maybe getting a call or two a day to more than half of the calls he now receives being Mac Defender problems. And that Mac Defender does trick people into having them put in the admin password that allows it to install into their system. And the people he had spoken to had not fallen for this fake purchase, enter your credit card information so you can purchase the software so that we're going to fix your system. But he had spoken to other AppleCare reps whose users, when they called Apple for help, had input their credit card information. And the scam is - I thought this was interesting, too - is that the software, even when they give it a good credit card information, says, oh, we were unable to process your card. Give us another one. And in some cases users went through, like, their five credit cards, putting them each in in succession, which of course...

Leo: Oh, that's clever. Now I understand why they kept, I mean, they would want that.

Steve: Yep, yep. So...

Leo: I have to, by the way, validate that it is on the rampage because I'm getting more calls on the radio show. But also lots of calls from Windows users bit by the same thing. The same idea, you know.

Steve: Oh, yeah, exactly. The one thing that was a little disturbing was that the AppleCare employees had been instructed by Apple, despite the fact that it is easy to remove and easy to instruct people to remove, AppleCare has been told not to remove it

for customers who call, on the theory that customers should not be given the expectation that AppleCare will be responsible for these things. Apple is saying that's the role of antivirus. That is not something that falls within our purview.

And this employee whom Ed interviewed said, well, I've helped some people out when they've explained their situation or they really needed help. I mean, he's had, I guess, I don't know what this thing shows people, but in the interview they talk about mothers screaming on the phone to AppleCare over the images that her children are being subjected to as a consequence of this being on their Macs. So whatever it does, apparently it's producing offensive imagery of some sort when this happens.

And so Ed says, well, aren't calls, AppleCare calls being monitored by Apple? And so if they heard you doing this, you could get yourself in trouble. And the guy said, yeah, I know, but even though Apple is saying we shouldn't help people, we sometimes have no choice but to do that.

Leo: Interesting.

Steve: Yeah, interesting. And in related news there is now the first do-it-yourself malware kit for Mac OS X. There have been famous Zeus and - Zeus is probably the most well-known malware kit for Windows. There's another one, Spybot, I think, or it's Spy something. Anyway, there is now one just that is being - percolating around the underground forums. It's called the "Weyland-Yutani Bot." And what's kind of fun for sci-fi people, Weyland-Yutani was the name of the fictitious corporation that ran terraforming in the second "Alien" movie, in "Aliens." And their slogan was "Building Better Worlds" or something.

So anyway, that's the name of this is the Weyland-Yutani Bot is what this do-it-yourself malware kit creates. Brian Krebs was able to get into a Russian-language forum and have a dialogue with the author, who provided him with a video. And there is a video on YouTube, also, of this thing operating, which shows the various UI components of the builder and the admin panel and the fact that it supports encryption. It's still low profile. And at the moment it supports what's called "web injection" and local form grabbing in Firefox and Chrome, not yet Safari.

Web injection is the process whereby, when a legitimate web page is displayed on your computer, like a banking web page, these bots will now inject additional content into the page. For example, if you're only being asked normally by the bank for your username and password, the bot will add your credit card CSC code and your billing address and things. It'll add additional fields. So it's legitimately coming from the bank, yet you're providing much more information in response than you normally would. And of course this is the malware gathering additional information so that they're able to collect data so that they can steal more information from you.

So this is, I mean, we're beginning to see what was inevitable, Leo. And that is that the Mac is catching up, as it has caught up in popularity, or as it is catching up in popularity, it's now become a target for the bad guys. And it's going to be a problem for Mac users who at some point are no longer going to really be able to say it's a much more secure platform than Windows. But I need to say, I tweeted this earlier today when I ran across this, one of the coolest bits of wisdom and advice I've seen. Brian Krebs on a different blog posting of his quoted three things. He said, If you've installed a program, update it regularly. Okay, well, that's something our listeners hear you and me say all the time.

Leo: Know very well.

Steve: They know very well. Number two, if you no longer need a program, remove it. And that's also one of my main focuses. We've talked about getting rid of Java if you don't need it because it's a problem. Certainly disabling JavaScript if you don't need it and so forth. And just in general getting rid of stuff you're not using because it's all baggage, and it's just more opportunities. What he adds to that wisdom that we've commonly shared, which I think is just fantastic, is the best advice for protecting users from social engineering attacks. And that is, if you didn't go looking for a program, an add-on, or download, don't install it.

Leo: Yeah, but the problem is people feel like they did, right, because they got a warning that said you've got spyware which looks like, it's designed to look like it came from the operating system. And then the offer to download a fix so that - that's why they enter the password, both on Windows and Mac. They go, well, yeah, I want to install this. I asked for it.

Steve: The other thing, the other main vector, for example, is like Koobface on Facebook will provide you a link. And the dialogue that comes up says, oh, you need to install the following video player in order to play this video. And so the user goes, oh, well, that makes sense, and so they do it. But my point is that the reason I really like what Brian says is, if you didn't go looking for it - and I understand, Leo, this is what confuses people. So that's why this is so important, if you think about it. If you didn't go looking for a program, an add-on or a download, don't install it. That is to say - and unfortunately it also means, I mean, it applies to legitimate. You would not be responding to legitimate offers also. But the idea would be, if something is being offered to you, treat it with skepticism.

Leo: Right.

Steve: Because that's how these social engineering attacks are getting you is...

Leo: If you see a message that makes your heart pound, don't trust it. I mean, part of their success is they scare you. So if that message scares you, that should be a sign that maybe it's not all it looks to be.

Steve: Well, but for me this decision point is, was it offered to me, or did I go get it? Did I go looking for it? Because I think that's really, I just - I tweeted it because I thought that's exactly right. You'd be skeptical to the point of just not doing it if something is offered to you. As opposed to, I mean, and this is what you and I do, we sort of have that intuitively, it's like, oh, well, I'm going to go find this somewhere else. Or I'm going to go do my own Google search to find the...

Leo: That's exactly what I tell the radio audience, is if - see, I don't know if it's enough to say don't install it if you didn't go out looking for it because there are

many of them running security software that would say, would you like me to remove this virus? They see these messages.

Steve: Although someone responding to my tweet this morning did say that he tells people, make sure you know what your antivirus program is, and don't respond to different antivirus messages.

Leo: Well, and there is no such thing as the Apple Security Center. That's a Windows thing. And I had a call on the radio show last week who said, I was scanning - my friend invited me, was having problems with her computer. She invited me over. I scanned it with MSFT, and it said, oh, I can't disinfect, you need the Windows Advanced Toolkit - which doesn't exist - and I installed it. And I said, did it cost you any money? He said, yeah, it was $79. And I said there's no such thing. There is no Microsoft Advanced Toolkit. And so it's know your operating system, be skeptical, and I think that Google has a very good tip, which is, if something like that pops up, just Google its name. Because if it's malware, you will see hundreds of entries immediately that say, "Don't install that."

Steve: Right. But also, Leo, look at the fundamental effectiveness of this new approach, the social engineering approach.

Leo: Well, it's always been the case. Social engineering, in fact, Steve, Kevin Mitnick, our favorite hacker, wrote a whole book on the subject. That's the best technique. Social engineering, it works.

Steve: Yeah. Too often. So I just caught a little blurb that I wanted to inform our listeners of, that just sort of made me shake my head. And that is that the American Civil Liberties Union, the ACLU, is kind of keeping an eye on what law enforcement is doing. And I'm glad they're doing that, keeping people honest. They filed a Freedom of Information Act, an FOIA request asking for information about the FBI's warrantless wiretapping, the FISA provision, and, for example, which ISPs were providing that information, how was this being used, what was being done, and who was the FBI working with.

In court documents which were filed in reply by the FBI, the FBI formally stated: "Specifically, these businesses would be substantially harmed if their customers knew that they were furnishing information to the FBI. The stigma of working with the FBI would cause customers to cancel the companies' services and file civil actions to prevent further disclosure of subscriber information."

Leo: [Derogatory sounds]

Steve: So I just think there's something we're doing wrong here if that's the situation. That is, if the FBI is afraid to tell us what it's doing, then doesn't that seem wrong somehow? I don't know. I mean, I guess it's that it's warrantless, that without warrant they're able to now use FISA to compel ISPs to spy on their users just because they want to, because they ask. And then when privacy rights organizations say, okay, just tell us

what you're doing, you're a taxpayer-funded organization, what are you doing? They say, well...

Leo: We can't.

Steve: ...we can't.

Leo: It would be bad for business.

Steve: Yeah.

Leo: If you knew your ISP was spying on you, you probably wouldn't use them. So we'd better not tell you.

Steve: Yeah, no kidding, yeah.

Leo: Oh, come on.

Steve: I know.

Leo: Holy moly. Holy moly.

Steve: So also, just Thursday, new legislation was put forth, I think it was in the Senate with 11 co-sponsors, called PIPA, the Protect Intellectual Property Act.

Leo: I don't like the sound of that one.

Steve: I know. Well, now, we know about the old unpronounceable acronym.

Leo: COICA.

Steve: COICA, right, C-O-I-C-A. And that was Combating Online Infringement and Counterfeits Act. And we know what it tended to stumble around doing because we covered the fact that they blocked out a top-level domain because some subdomain was in fact offering counterfeit purses or something. But they didn't realize that this was a multidomain hosting provider, and as a consequence tens of thousands of other websites were taken down by mistake. So, whoops. The so-called "domain seizure" didn't work so well. So...

Leo: This is going to turn me into a libertarian, they keep this stuff up, I'll tell you.

Steve: I'm there, Leo. I know. So instead of domain seizure, the new PIPA, P-I-P-A, it would allow the Justice Department - I mean, if it passes, right now it's just introduced legislation - to allow the Justice Department to obtain court orders compelling ISPs' DNS servers to stop returning results for particular websites. Meaning that the sites would still be available outside the U.S. So rather than essentially removing domains from the root name servers, from the root servers on the Internet, they will instead send to ISPs - I don't even know what the mechanism would be because, I mean, there's lots of ISPs and a gazillion DNS servers. But this plan is to place filters in the results of DNS servers so that lookups will fail.

So it's like, okay, I mean, that's not going to work either. These attempted sort of lame technological approaches to solving problems that are legitimately created by the nature of the Internet and digital media are bound to fail. There's all kinds of ways around this. If you're trying to get this kind of content, you can just use foreign DNS servers. They work just fine. And there's way more of those than there are local ISP DNS servers.

So it's just - it's nuts. I have to say, though, I was surprised by something I saw just the other day. Another of my recent tweets actually is that Netflix is now the No. 1 source of traffic on the Internet. Which was - I think it's very cool. But what surprised me...

Leo: It's giving Comcast ammunition, that's the unfortunate thing.

Steve: Yeah, it is. But what surprised me is that BitTorrent is No. 2.

Leo: You obviously don't know many BitTorrent users.

Steve: I don't. I'm not a torrent user. But that's like, whoa, No. 2 is BitTorrent. So it's like, okay, yeah.

Leo: Is email still in that list anymore? I don't think it is. Used to be that was, like, No. 1 by far.

Steve: No. In fact, some amazing things got bumped off the list, they don't even show up on the chart anymore, that were interesting.

We talked last week about the breach in Chrome's sandbox. Remember that the French security group VUPEN came up with a zero-day exploit against Chrome.

Leo: And I was skeptical. I said I would like to see the code, which they weren't showing anyone.

Steve: Well, what happened was there was a little bit of a Twitter war, it turned out. Three days later Google's Chris Evans, who's their information security engineer and a

tech lead, he tweeted, "It's a legit pwn, but if it requires Flash, it's not a Chrome pwn."

Leo: Except that Flash comes with Chrome.

Steve: Yes, it does. But it does turn out that this was a Flash exploit. So this was Flash in the sandbox that allowed Flash to get out of the sandbox.

Leo: But, I mean, Google's the one who said, look, we're sandboxing Flash. That's why we bundle Flash. We want to make it safer. So they're the ones who were trying to protect us.

Steve: I agree. And in fact...

Leo: So I don't think that that gets them off the hook by any means.

Steve: Well, and the VUPEN guys tweeted in reply, they said, "Flash bugs are equivalent to Chrome sandbox escapes from an attacker's perspective. You're thinking like developers."

Leo: Right. "It's not our code."

Steve: Exactly.

Leo: "It's our product, not our code."

Steve: It's code that we install by default in the browser so that it's able to run.

Leo: That's a good point. That is thinking like a developer.

Steve: Okay, so I completely forgot last week, Leo, to talk about LastPass.

Leo: Well, didn't we talk about it two weeks ago? I thought we did.

Steve: No, what happened was this all happened just after we began, after we finished recording.

Leo: Oh, that's right.

Steve: So I did a special appearance with Tom on This Week in Tech. And I thought,

well, why is everyone asking me why I forgot, why I didn't mention LastPass last week? And so looking at my own tweet stream, I realized, oh, I said I was going to.

Leo: You have my problem, which is I do so many shows, of course we talked about the LastPass issue ad nauseam, but perhaps not on this show.

Steve: Exactly. So I have to do so briefly. Essentially what happened, for those who didn't see my special appearance on This Week in...

Leo: No, TNT. Tech News Today.

Steve: TNT, Tech News Today, right, which I did that afternoon with Tom because you and I had already recorded it, but this was important. The guys at LastPass saw some traffic, some just sort of like traffic volume that they couldn't explain. They didn't know what it was because it's all encrypted. So they're just seeing blobs going back and forth of pseudorandom stuff, but it looked uncharacteristic to them. And so just, I mean, that's all it took.

Leo: I loved it that they blew the whistle on themselves, being proactive.

Steve: Yes. I mean, as far as we know, there wasn't even a breach.

Leo: We don't even know, right.

Steve: Exactly.

Leo: They just wanted to be extra cautious. I love that.

Steve: It was very impressive. So what they did was they sent out the word to users to change their master password because the worst that could have happened, and this is what I talked about with Tom, the worst that could have happened is that an attacker could have gotten the hash that they use for authenticating users and the users' encrypted blob. Some people don't understand the way LastPass works and thought that a breach of LastPass's database security immediately meant that bad guys were able, consequently, to decrypt the blob. But that's the whole reason I endorse LastPass is LastPass themselves cannot decrypt that blob. So, for example, you don't have to worry about them responding to an FBI subpoena for your information. They can't. They don't have the ability to decrypt the information. And that means that an attacker who gets what they have can't do it either. So they don't have it to give.

Now, the only weakness would be a brute-force attack against the hash, and we've talked about that, what that means, also. So if you had a good master password for LastPass, then the algorithm is your email address is lowercased because email addresses are not case sensitive. So LastPass wanted to eliminate the possibility that you would be entering your email address with different case when you're trying to

authenticate to LastPass, to log on to LastPass, than you normally use. Since email doesn't care, they don't care either. So you lowercase the email address. Then you merge that with your password. And that's hashed to create the private key which you use to encrypt your database. That never leaves your browser client. Then your password is hashed again with that key. And that's what is sent to LastPass to authenticate you.

So again, they're getting a hash of your password and your secret key, which itself is generated from a hash of that password and your email address after it's lowercased. And hashes, hash functions are beautifully designed to be one way. They used SHA-256, the strongest state-of-the-art hash we have, generating a 256-bit result, so certainly strong enough. I guess SHA-512 obviously is twice longer, but it's not clear that there's any point in using that. And so what an attacker would have to do would be to repeat that process with every possible password.

Leo: Do they go one extra by doing multiple salts? Is that the one we talked about last week?

Steve: That's actually what they're going to do. When they looked at what the vulnerability, what the theoretical vulnerability was, it would be somebody using a very weak password, like a dictionary word or something, the kind of password that no one should now be using, which because it was weak, you could mount an offline attack where the user's email address, which again the attacker might have because LastPass does have that, that would be hashed with this test brute-force password. That would get hashed. Then it would be hashed again to produce the authentication token. And if the attacker did have that, they would look for a match. That would then - they still wouldn't be able to decrypt the user's data. But then they would be able, then they would have the password as a consequence of going forward through the two hashing functions, brute forcing it. Then they could authenticate to LastPass and obtain the blob which then they would be able to decrypt it.

So if bad guys got the database, and if users had weak master passwords, then there's a possibility of doing, over some length of time, a brute-force attack. And so erring way on the side of caution, the LastPass guys said, okay, just change your master password. We're sorry to put you through the inconvenience. You don't have to change any of your other site passwords, just the one master password. That way nothing that might have been taken, if anything was, and we don't know that anything was, nothing that might have been taken would still be vulnerable even to a brute-force attack.

Leo: And they also offer, and I use, two-factor authentication. So if you use that, you're safe, too.

Steve: I was going to say, anybody with a YubiKey, for example, absolutely was safe because, again, these guys have looked for every opportunity to help make us secure. They take the fixed portion of the front of the YubiKey token and mix that in also, which no attacker could ever have. So there's just no way to compromise if you were using the multifactor authentication. So bottom line is, they responded immediately. Their site was overwhelmed by people who needed to change their master password. They're now thinking, well, maybe we did overreact, frankly. But better safe than sorry.

Now, the one thing they're going to do is, and we talked about this last week, this notion of key stretching. And that's what you were referring to, Leo. The problem with brute-

forcing a hash is hash functions have been - there is hardware, for example, to really perform very fast SHA-256's. For example, that's how Bitcoin mint guys are using GPUs in order to try to sign and generate the hashes for the Bitcoin transactions. So code exists for GPUs in people's machines to do SHA-256 very fast.

So the solution is to require a great many of them per authentication. And the number I saw was 100,000, which is to say, to authenticate, the SHA-256 function would be run 100,000 times, meaning its output put back into its input, maybe mixed with something else, and then that generated and done again. And so what that does is, no matter how fast a brute-force attack on that content could be mounted, this would slow it down by 100,000 times because every single guess would have to be done 100,000 times. So I'm going to keep track of that, and I'll let our users know whether that gets implemented and when. But, I mean, basically they've just done - they had done everything they could before. This further raises the bar, and it's not clear whether actually anyone even got data from them.

Leo: The bar needed to be raised.

Steve: Yeah.

Leo: I think that's really a good model for companies like Sony and others. I mean, be proactive. Jump on it.

Steve: Yeah. In miscellaneous news, I wanted to mention that I've been receiving really nice feedback from people who took my recommendation of Mark Russinovich's "Zero Day" book to heart, many of them saying they're only a few chapters in and hooked already. So people are liking it a lot.

There was a really interesting TED Talk about the danger of filter bubbles, as it was called. I tweeted the URL for that and recommended that my followers check it out. You can also find, if people are not following me or aren't using Twitter, just - of course you can always look at my Twitter stream by going Twitter.com/SGgrc, and you can see everything that I've tweeted recently, that I've been referring to as it happens in this podcast. Or just go to YouTube and just search for "filter bubble."

It's a really interesting nine-minute TED Talk which demonstrates sort of the social issue of the fact that increasingly the results that we're getting from searches are being customized for us, and what that means. I just - I recommend it to our listeners. It's a little scary. This guy doing some research [Eli Pariser] had two different friends both Google 911. And he shows the radically different search results that Google returned because of what Google knows about these people.

Leo: Now, only if you're logged in. I mean, if you log out of your Google account, nothing.

Steve: Is that the case?

Leo: Yes.

Steve: Okay. Because, I mean, they...

Leo: So you have to opt into that, basically.

Steve: Google has cookies and IP addresses and...

Leo: No, no. You can opt out. You can log in and opt out of history and tell them not to keep history. If you don't log in, they have no history. And so you will not get customized results.

Steve: He also shows a result with Facebook, with Facebook noting over time the kinds of things he clicks on.

Leo: Well, that's true.

Steve: And that Facebook, in trying to do a better job of giving him what he wants, he ends up not seeing information that he wishes he were seeing. Anyway, YouTube, 'filter bubble," search for that. It's a really interesting nine minutes. Just apparently it's happening more and more and worth just being aware of.

Leo: It's intentional. And it's designed to make your search results better. But if you don't like it, you can, at least with Google - obviously Facebook you can't turn it off because you're always logged in.

Steve: Well, and in general, search in general, we know that, for example, the reason we're being given custom advertising is that advertisers are able to charge more when their message is going to an audience that's more likely to care about what they're offering. So in general we're seeing the web attempting to conform to who it thinks we are. And that's why tracking is become so controversial is a lot of people started to have an "ick" factor relative to the notion of profiles being assembled about them. And of course those are being used to customize their experience of the web. And he just brings up the point that, well, maybe that's not such a good thing in all cases.

Leo: Yeah, be aware of it, I guess.

Steve: Yeah. My favorite tweet of the week was sent to me from @luridsorcerer, that's his Twitter handle, it's Andrew Lingenfelter. He said, "The online software license option I'd like to see would have a third option. You have 'agree' and 'disagree'? The third option would read, 'Didn't read the license, but agree anyway.'" Which I just got a kick out.

Leo: Well, that's what I do.

Steve: That's what we all do.

Leo: That's what we all do.

Steve: No one reads the license. You can't.

Leo: So I think it would be nice to have that checkbox, yeah.

Steve: Yeah. That'd be nice for a third option. And someone else this morning, Simon Bodger in Canada, he said, "Steve, love the PEE acronym." P-E-E. He says, "As a parent" - and remember that PEE stands for Pre-Egression Encryption, that is, encrypting stuff before it leaves your system in order to be safe with your data in the cloud, Pre-Egression Encryption. So he says, "As a parent, I'm always saying, 'Before we leave, did you pee?' Sounds like good advice for my data, too."

Leo: I love the acronym.

Steve: I had some interesting tweets about that.

Leo: I bet you did.

Steve: That was a fun one. And a listener of ours, Jeffrey Wurzbach, he said, "I downloaded a copy of SpinRite from someone." Or borrowed. "I borrowed a copy of SpinRite from someone. After SpinRite..."

Leo: What?

Steve: Well, you know, that happens. I recognize that. But he says, "After SpinRite unbroke my machine, I decided to buy it. My computer was taking a very, very, very" - got three verys there - "long time to boot Windows. Both my main boot disk and my secondary disks were randomly powering up and down. I suspected something was broken in the hardware of the disk. But not having the money for a new computer, I had to do something. So I used SpinRite on the main disk. It was able to get the computer to boot about 50 percent faster and get all of my critical data onto my network storage drive. SpinRite saved me many hours of frustration and four-letter words. If only I had SpinRite four years ago when my parents' computer died." So the good news is he tried it, liked it, bought it, and has it now.

Leo: Yay. Yay.

Steve: For the next disaster. So thank you for sharing that, Jeffrey.

Leo: Let's talk randomization.

Steve: So, yeah. One of the things, and I mentioned this briefly last week, that sort of annoys me sometimes is the people who make the blanket statement that security through obscurity is no security at all. Because the fact is, all security depends upon obscurity. It just depends upon having and knowing what the attack model is and understanding what needs to be obscure. So let's draw an example. Say that you had a non-keyed cipher algorithm, that is, an encryption algorithm that did not use a key. So it was just this algorithm that, when you ran it, you put in something and out came a blob that was encrypted. And you'd think, wow, that's fantastic, that's great.

The problem is, you absolutely have to keep the algorithm secret because the only protection that you have that's created by this algorithm is the algorithm itself. So in order for this to be effective, that has to be kept absolutely secret. If it gets out, then anybody can look at what's been encrypted and decrypt it, or impersonate you by encrypting something themselves. The point is that that's a perfect example of bad crypto and bad security because you're inherently depending upon the algorithm being kept secret.

Now, contrast that to the way we know things are correctly being done these days. We have public algorithms where, I mean, they're public from day one. There are public competitions, as we saw with the AES competition to end up choosing Rijndael to be the standard cipher that so many people are now using. So, I mean, it was way public. It was taken apart by academicians. It was analyzed. It was really torn apart.

The reason we can do that, though, is there's only one reason. And that is, it's a keyed cipher. Which is to say the strength of the result is the combination of what is absolutely well known, which is to say the cipher itself. And it's because it's well known that we trust it because academicians, cryptographers, have been able to pound on it. They've been able to, like, reduce its strength by having it go fewer rounds and, like, watch it work, and see how quickly, as you increase the number of rounds the cipher goes, it becomes impossible to reverse engineer it.

So it's two things. The total secrecy that we get is composed of something we know, which is how this algorithm works, but something which is still obscure, that is, it's still kept secret. And that's the key which is used to run the cipher. So it is not the case that we get security if everything is known. It's that we carefully design what parts of our system are known, but we then still have secrets which we keep. And the reason then that we need, in cryptography, we need random numbers is that we don't want an attacker through any means at their disposal to be able to guess what the secret is. We're going to have a secret. In SSL communications, HTTPS that we've talked about, there is a secret. There is a handshake which goes on between the endpoints where they arrive at a secret by using random number generators. And we depend upon the randomness for security.

For example, this hasn't always been done correctly. The very first version of Netscape's SSL, Netscape back in the original early browser days, they were the designer of SSL v1.0. They used the time of day, the process ID, and something else. There were three components. I don't remember what they were. But the point was they weren't - and they used those to seed the random number generator which their SSL connections then used for establishing secure handshakes. And it wasn't long after - but they didn't publish

it. They kept it secret. And of course these are the kinds of secrets you can't keep.

A year later, some researchers figured out what Netscape was doing, looked at the fact that they were using time of day, process ID, and one other factor, but still predictable. And it turns out that something like process ID, it's like, well, we don't always know we're going to have the same process ID; but if you start a computer up, and you do a couple things with it, it's typically going to be this. Or it's going to be some range which allows an attacker from the whole possible spectrum of process IDs to zero in on, well, it's probably going to be one of these. The point is, they didn't start with enough randomness. And it turns out that version 1 of their SSL protocol could be attacked by virtue of its lack of good random number generation.

So we've seen that over and over. And in fact it's a lesson which some people don't learn very well. Microsoft in '07 had the same problem with the pseudorandom number generator that was in Windows 2000, and it's the same one that's in XP. They had some researchers who - and again, Microsoft didn't publish this. They maintained - they kept it strictly secret and tried to obscure it. Some researchers ended up taking it apart, figuring out how it works, and were then able to demonstrate that, because of what Microsoft had done, the decisions they had made were not cryptographically sound. And if they were ever able to obtain the state of the random number generator, then they were able to know everything about its future. So if something ever got into your computer, they could take a snapshot of it. And there was no more mystery, even though Microsoft had gone to some effort to try to enforce that. It just - it wasn't done well.

So there's a strong incentive for saying whatever it is that is done, in order to have it cryptographically sound, you really need to have it looked at by people. You need it to be public, never secret. And you need to just say, this is what I have done. What does the world think about it? Have I forgotten anything? Because we know mistakes can happen.

So one of the problems that computers have, as we've discussed, is that they're 100 percent deterministic machines. If you start the computer in a known state, and it goes through a series of processes, it's always some time later going to arrive at another state, that is at a given state. It's adding and subtracting, multiplying, dividing. It's jumping, it's doing the things it's doing. But it's always following instructions, which unless they change, what it does and where it arrives is not going to change. So there's this notion of state which can be thought of as a quantity.

For example, I was talking about, last week or two weeks ago when I introduced this, the idea of taking a 32-bit integer and using a simple algorithm calling a linear congruential pseudorandom number generator, where you multiply that value by a constant, and then you add a different constant, and it produces a new result. And if you choose the multiplier and the addend correctly, that 32 bits will jump all over the place until it comes back to the original one. And it'll occupy the full, every possible combination of 32 bits before it returns back to the beginning.

The problem is that it is a very weak algorithm. It's simple for, like, a cryptographer would just not take two seconds to figure out what this thing was doing and be able to predict the entire future of numbers being generated. The other problem it has, though, is that its state is only 32 bits. That is, at no time can it have more complexity than that. That is, it's got one of 32-bit values, and it's next going to move to a different one of 32 bits. And there's no way for it to be more unknown than that.

So what we try for in contemporary pseudorandom number generators is much more entropy, much more state, so that the system is not easily knowable. And if someone did get, if a bad guy somehow got a snapshot of it at a given time, there would still - the

future is not completely predictable. That's key for real security. We can often assume that the bad guys cannot get into the system. For example, if someone's eavesdropping on an SSL communication, well, we want a protocol and a random number generating system which is strong against someone looking at all of our packet traffic. And no matter how many connections we establish, if there was a man in the middle or someone sniffing the initiation of connections over SSL, we would want them, no matter how much they looked at the output, never to be able to guess what the past or the future had been.

Sometimes we don't have strength. For example, a malicious process could get into the computer where it has access to the machine and could get a snapshot of the pseudorandom number generating system. If we're trying to prevent against that, then we need some ongoing source of entropy always sort of being added into a pool. And cryptographers talk in terms of entropy pools, the idea being that you feed in anything that you can get which is not predictable by an attacker, certainly not one outside the machine, and often not one in the machine. That would require them to constantly be monitoring your random number generator. And frankly, we're really not designing our systems to be strong against an attacker in the machine, but rather if we generate a large random key that we use to encrypt our drive, or we use to encrypt communications. The point is that we want anyone coming along afterwards to have no basis for knowing what that key was, how we arrived at that key.

So we're still faced with a problem of where do we get this kind of true random information. Historically, all kinds of things have been done. What we want is we want the notion of uncoupled independent events where things happen that are not related to events that have already happened in the past or would happen in the future. A popular source, for example, is some sort of electrical noise. There's noise generated in electrical processes that are the result of quantum fluctuations, low levels of heat, low levels of photon radiation entering the system. There's, for example, been a project where a CCD imaging array, a Charge-Coupled Device imaging array is put in a completely black box. And it turns out that even though there's absolutely no light getting to this imaging array, the signal it outputs still has some non-zero content. There's noise. It's like sort of imaging hiss. And it's extremely random. There is absolutely no way for an attacker to know what that hiss is going to be from one instant to the next.

Another possibility is just using the least significant bit of an analog-to-digital converter. For example, the sound input on a sound card, even with nothing plugged in, you typically don't get an absolute flat line. You get a few values in the least significant bits, which is just noise in the amplifier that feeds into the analog-to-digital converter. So, and that again, it won't be following a pattern. It's just, it's noise. It's not exactly predictable. And that's what you want. Or a radio, just that hiss you hear when you tune a radio in between carrier frequencies is just the electronics in the radio sort of just listening to what's going on in the environment. And that white noise that you hear is also not exactly predictable. Or a Geiger counter, which is measuring individual photons which are coming in and ionizing the gas in the Geiger tube, is producing a not absolutely predictable series of events. And famously, SGI, who years ago wanted a good source of real random content, actually patented, they have Patent 5,732,138, which is aiming cameras at lava lamps and using the...

Leo: That's a chaotic system, I think.

Steve: It absolutely is. You cannot, I mean, you can stare at it, no matter how long you stare at the lava lamp you really don't know what it's going to do. You cannot predict

what it's going to do. And so SGI set up a lab, a darkened room where they had cameras looking at lava lamps, and they were digitizing the images of the lava, which is hot wax, moving through that fluid. And as you said, Leo, it's a chaotic system. You cannot predict what it's going to do. More recently there are some very good random number generators using a partially silvered mirror. For us, when we look at a partially silvered mirror, we can sort of see through it and sort of not, that is, we'll sort of both see what's on the other side of it and what the mirror is reflecting.

But at the quantum level, what that actually means is that some light photons are reflected off of it, and some go through. You need that behavior in order to partially see through a mirror. What that means is that you can put a photo detector, a photon detector, on one side of it, and have it count the events of things that go through. And it turns out that's quantum uncertainty down at the quantum mechanics level, and it is extremely random. It is not producing a predictable result. It may not be 50/50, but it turns out you really don't need 50/50-ness in order to generate a source of something that is, while it may not be purely random, it is absolutely unknowable.

So then there's other things that have been done. For example, people have looked at the arrival time of Internet packets, network traffic packets. And in fact anything that you do, if you have sufficiently high resolution in a counter, in a timer - and that's the key. What's really neat about our current trans-gigahertz machines, we've got 2.4GB, 3.0GB and so forth, there's a counter in all of our Intel chips, and non-Intel chips have them, too, that is running at the speed of that system clock. It is running at 3 billion, that is, gigahertz, 3 billion counts per second.

Now, if you took a snapshot of that timer when a network packet arrived, and then another network packet, and another network packet, well, if you didn't have much resolution, if that counter weren't running very fast, then you wouldn't be really getting much randomness, much entropy from that. But with it running at 3 billion counts per second, even if you tried to send packets to your computer at an exactly perfectly timed rate, variations in the packet assembly, in the interrupt system, in the computer, I mean, just 3 billion counts is so fast relative to sort of the real world events that are happening, that the least significant bits, the one bit, the two bit, the four bit, they're absolutely unpredictable.

And that means you can use other things that are happening. You can use the timing of reads and writes on disk drives because, as we've spoken about in the past, disk drives are not spinning at an absolutely constant rate. The sector timing is used to servo the drive to keep it running at about the right speed, as is the distance from the heads to the platters. But that's varying. So that means that, if you look closely enough, you're going to get a huge amount of uncertainty. I mean, again, look closely enough. If you use this 3GB counter that we have, or 2.4 or whatever, there's no way to predict what the least significant bits are when you take a snapshot of it. Or when you move the mouse, or press keys on the keyboard. They're just, if you're spinning that counter fast enough, the lesser significant digits are much more predictable than the least significant digits, which in the case of the 1 bit, it's changing 3 billion times per second. And you're going to say, okay, now. What is it now? I mean, that's so fast, that gives you really good unpredictability.

And then, finally, just because you could never have too much randomness, we know, for example, that the Trusted Platform Module, the TPM, which is in most laptops today - I wish we were seeing it more in desktops. I know that it is available in some desktops, but not yet universally. The TPM has a true random number generator in hardware in it. There it's using thermal noise down at, again, at the quantum level in order to generate an offer on behalf of the system that is hosting this TPM, true physical random numbers.

Not pseudorandom. Anytime we have pseudorandom we're saying that there's an algorithm which is generating these. And by the nature of an algorithm, we know what each number is going to be in succession. With a true random number generator, it's not algorithmically based. It's just every number that it produces, it's producing freshly.

Now, the problem is that the rate at which we're collecting entropy may be lower than the rate at which we want to consume it. Let me say that again. It may be that the rate at which we're producing it or able to collect new randomness, new entropy, is lower than the rate that we want to consume it. So, for example, the true random number generator in the Trusted Platform Module will have some bandwidth. It'll have some rate at which it's able to give us numbers based on the physical processes that it is tracking. And you can imagine lava lamps, yeah, they're chaotic, but they're not moving very fast.

Leo: We need more bits.

Steve: We need more bits. So what has been done is very clever. And this is represented by the state-of-the-art pseudorandom number generators which, for example, Bruce Schneier has designed. There's one called Yarrow that Schneier and John Kelsey designed, and a newer one called Fortuna which Bruce designed with Niels Ferguson. The way they work is they have an entropy pool, that is, they have a - think of it as a buffer, just a big buffer. And whenever something happens that they can say, okay, well, we're not sure how much entropy is there, but it's probably got some.

For example, imagine that every time something happened you took a snapshot of that 3GHz counter and added it to the entropy pool. So packets come in, hard drive sector read or write happens, mouse gets moved, keyboard gets hit, even something like the graphics display adapters often have a 60-cycle or a refresh interrupt. And that's not coupled to the counter running in the machine. It's generated by a separate crystal on the display card, separate from the crystal on the motherboard. So there you could, at 60 times a second, you could sample on the display interrupt the exact count of this 3 or 2.4, whatever it is, gigahertz timer running very fast in the computer and just dump it into the entropy pool. You don't really care how much entropy it has. You just know there's a lot of uncertainty in the least significant bits of that. Nobody can guess. And in fact, if you did, if you plotted them out, and people have, you would see absolutely no predictable pattern in those least significant bits. They're just uncoupled, and they're related to physical processes down at the quantum level.

So what state-of-the-art random number generators do is they collect this entropy pool, adding it to this pool, until they decide that they have collected enough of it to reseed a pseudorandom number generator. And, for example, the most recent one, Fortuna, which Bruce and Niels did, they hash the content of the pool to generate a new key for our friend the AES Rijndael cipher, which they have being driven by a counter. And after they have hashed the pool, they empty it. And then they begin again putting new entropy into it. Meanwhile they have an absolutely unknowable key which is the key to the AES cipher which they're driving from a counter.

Now, what they know is that they cannot pull from this forever because the counter will wrap. And we can't ever allow the counter to wrap around with the same key or it will be reproducing the identical sequence of pseudorandom numbers that it originally produced. But that's not a problem. We've got 128 bits that we're encrypting with Rijndael which will last us a long time, relative to the rate that we're able to collect new random things happening from the environment, which collect up to, we only really need - we need at least 128 bits, or, for example, 256 if you wanted to use a 256-wide key for Rijndael.

So as soon as we collect, like, maybe say 1,024, just so we know that we've got more, then we hash that down into 256, and we rekey the cipher so that it is - it's constantly being rekeyed. But it's able to give us a tremendous bandwidth of pseudorandom data which is based on the truly random data which we're accumulating in this pool at a much lower rate. So our physical processes won't give us megabytes per second of true randomness, but they will give us enough that we can use that to constantly reseed a pseudorandom number generator which can run at whatever speed we want. And that's what we're doing now in state-of-the-art crypto systems, is we're using some sort of physical processes to accrue pools of true entropy, which is absolutely unpredictable, which we then hash into a key to drive a very simple and very efficient pseudorandom number generator which gives us, while not absolutely random numbers, it meets our need of both having a high bandwidth and being absolutely unpredictable.

And that's all we need. We can't know - no attacker, looking at the past or future, can know what the number is we're going to have now because of the strength of this cipher, even when just driven by a counter with an unknown key. That much we know we've proven. And by using an entropy pool, there's no way for us or anyone else to know what it is that we're going to be giving to that counter and cipher as its key from one moment to the next.

And so we've solved the problem very cleverly. Even though we've got technically a deterministic system, there are enough nondeterministic bits, literally, that are available because of the resolution at which we're able to measure these various physical processes that they're unpredictable, unknowable, and they give us a source of absolutely cryptographically strong pseudorandom numbers, at whatever speed we want them.

Leo: Now we understand that weird JavaScript page that you've put together.

Steve: Well, actually, yeah. What I did on that JavaScript page, I mentioned it a week before last, it's GRC.com/r&d/js.htm. What I'm doing there is I am tracking mouse and keyboard movements. The problem is I don't have access to, in JavaScript, to a really high-precision counter. I've only got a millisecond resolution. Which is better than nothing, but it's not really good. So the way I solved the problem of a client, a JavaScript client, a browser having really good entropy, is I have GRC give the page a starter 256 bits. And then everything the user does at their end is unknowable by GRC. So I seed the random number generator with something that is really good, that is, a pseudorandom number coming from GRC. The problem is, from a security standpoint, I know - I, GRC, know what I gave the user.

Leo: That's right.

Steve: So we want to immediately diverge what GRC provided from what the user's going to use. So, for example, the JavaScript takes a snapshot of the time of day when it starts to load. It takes another snapshot of the time of day when it finishes loading. It asks the JavaScript pseudorandom number generator, which is not very good, but it's got an unknown state to GRC, it asks it for 270 bits, which the JavaScript pseudorandom number generator provides. And so the point is, everything that I can capture that is going on on the user side is poured into a hash that hashes what GRC provided, immediately diverging it in a way GRC can never know because it's happening all on the client side.

So we get the best of both worlds. We're not only dependent, that is, the user's browser, whatever process it's doing, isn't only dependent upon randomness coming from his computer because we know that's not very good. But I'm able to provide, I GRC am able to provide a starter which is very good, but we don't want me to know what the user's going to use. So we immediately hash everything we can on top of it to diverge it from anything GRC knows. I mean, and if you just hash one bit, it's instantly going to be a different value. And we're just - I have no idea what the user's time of day clock is, how long a page is going to take to download, certainly not what the state of their JavaScript pseudorandom number generator is. So you put all that together, and it's extremely strong.

Leo: And it changes as I move my mouse, so it's always different.

Steve: Yes, exactly. Every event is rehashing that event timing. And any details, like the X and Y position of the mouse, any information I get is being poured into the hash and rehashing the key so it's constantly evolving and ends up being very secure.

Leo: Yeah. Cool, too.

Steve: Yeah, it's neat.

Leo: Well, thank you for filling us in on random. I think that's it, that's everything we need to know about random.

Steve: I think it is. We've never talked about it before, but it is crucial that we have, for crypto processes, that we have a source of really good random numbers. And I hope Microsoft is listening because this is better than anything that they were doing back with Windows 2000 and XP. And I do think that they have evolved their technology since then because they finally understand it's important.

Leo: Yeah. And I think it's also, from a pure computer science point of view, a fascinating conversation. It's certainly practical, but it's also interesting.

Steve: Well, and the NIST has published a set of tests that can be used to measure the quality of randomness. There's, like, 16 of them: mono bit frequency, block frequency, cumulative sums, runs, longest run of ones, binary matrix rank, DFT spectral, non-overlapping template matchings, overlapping template matchings, Maurer's universal statistical test, approximate entropy, random excursions, random excursions variant, serial, Lempel-Ziv compression, and linear complexity. And it's funny because some very good random number generators fail various of these tests in different ways.

Leo: Really, that's interesting.

Steve: Yes. Some pseudorandom number generators fail them in different ways. But I

got a kick out of that Lempel-Ziv compression. That's LZ compression. That's the original technology that we all use in, like, PKZIP and everything. That is the zip compression. And of course, as we've talked about it, one of the measures of randomness is compressibility. Right?

Leo: Yeah. Yeah.

Steve: Because anything with a pattern can be compressed because the compressor finds the pattern and uses that in order to get compression. But pure random noise is incompressible. There's nothing that a compressor can latch onto in order to make the result smaller. So that's kind of cool.

Leo: Yeah.

Steve: And the DFT spectral, that would be discrete Fourier transform, which would be to say do a Fourier transform of the noise to see if there are any frequencies which are occurring in it because it ought to be absolutely flat spectrum. There should be no peaks of any particular frequency. Sort of like you'd see in a gas chromatograph, where you have, like, peaks, you're identifying something. So these are tests which are really ruthless and very rigorous that the NIST has solidified on as part of their formal publication of, like, is this random enough?

Leo: Fascinating stuff.

Steve: Cool stuff.

Leo: But hardly random. In fact, you could find this show right there on the Internet at GRC.com. Steve makes 16KB versions available, as well, for those of you who have bandwidth caps or don't want to spend a lot of time downloading. Transcripts, that's the smallest version, if you like to read along, and all the show notes, including that R&D page, is at GRC.com. That's also where you'll find SpinRite, the world's finest hard drive maintenance and recovery utility, a must-have. If you've got hard drives, you need them.

Steve: It works.

Leo: Next week a Q&A episode. Go to GRC.com/feedback to leave your questions. And you can follow Steve on Twitter, @SGgrc. GRC.com's the thing to know. Just memorize that, Gibson Research Corporation. You can also watch us do this show every Wednesday live, 11:00 a.m. Pacific, 2:00 p.m. Eastern, at live.twit.tv, or subscribe on iTunes or any other podcast client. Or just go to TWiT.tv/sn for Security Now!. We've got all the episodes there and all the subscription buttons you could want. And the chance to buy a brick for the brick TWiT house. Thank you so much, Steve. We will see you next week.

Steve: Talk to you then, my friend.

Leo: On Security Now!.