
# Password Complexity Calculations

**Description:** This week, Leo and I discuss how, while still on probation, Symantec issues additional invalid certificates; how Tavis Ormandy found a very troubling problem in Cisco's web conferencing extension for Chrome; yesterday's more-important-than-usual update to iOS; and renewed concerns about LastPass metadata leakage. The SEC looks askance at what's left of Yahoo. We talk about a troubling browser form autofill information leakage. Tor further hides its hidden services, and China orbits a source of entangled photons. Then Heartbleed three years later, a new take on compelling fingerprints, approaching the biggest Pwn2Own ever, some miscellany, and, finally, some tricks for computing password digit and bit complexity equivalence.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-596.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-596-lq.mp3

---

SHOW TEASE: It's time once again for Security Now!, your favorite show. Every week, Steve Gibson joins us and talks about the latest security news. He's found, well, somebody found and he's going to talk about, a really horrible flaw that exists in Chrome and some other browsers. You won't believe this one, and it's been around for a while. He'll also talk about passwords once again, the easy way to determine if your password is really good. Well, easy if you love logarithms. It's all coming up next on Security Now!.

**Leo Laporte:** This is Security Now! with Steve Gibson, Episode 596, recorded Tuesday, January 24th, 2017: Password Complexity.

It's time for Security Now!, the show where we cover your security and privacy online, all over the globe. And it's going to be more and more important. This guy here, he's doing it, Steve Gibson of the Gibson Research Corporation, creator of SpinRite, discoverer of spyware, author of the very first antispyware program, and our guru. We sit at his feet and learn each and every Tuesday.

**Steve Gibson:** And sweeper of cobwebs.

**Leo:** Well, that we've got to do, too, a little bit, get those cobwebs out of the brain. We had yesterday, I did Triangulation with my old friend from ZDTV, Alex Wellen. And in 1999 he and his partner Luke Reiter pitched ZDTV with a show called

"Cybercrime." And it was, I mean, here we are, 17 years later. It was the earliest days of malware and hacking. In fact, we showed a little clip of it from YouTube that had - what was the - was it Melissa? It was one of the early viruses.

**Steve:** It was so quaint back then.

**Leo:** It was quaint. It was half an hour a week.

**Steve:** It had a macro in your email. Aw, ain't that adorable.

**Leo:** Exactly. And little did we know, I mean, how advanced this would all get and how much it would change. Yeah.

**Steve:** Yeah. So we have - there was just a ton of news. And a number of our listeners have become very engaged, a large number actually, in this whole question of password complexity, password formation. And, for example, I saw so many times over the last week, well, okay, Steve, you said you don't like to use dictionary words. But if you had enough of them, wouldn't that be strong? And I thought, okay. I've sort of glibly danced past some math which I use all the time for answering those questions. And I've never explained it.

And I thought, what I should do is give our listeners, who are obviously interested and capable, the tricks that I use for answering those questions so they can perform their own tests. They can look at the number of words in the dictionary and determine how many bits of entropy that number of words, if randomly chosen, would have. And then, by multiplying by the number of those words, they can determine the total bit strength of a password generator and so forth. So the topic for this week is Password Complexity Calculations. And it sounds kind of freaky and scary, but it's actually amazingly simple, thanks to the incredible beauty of logarithms, which of course make slide rules popular. Now, I realize children may not know what a slide rule is.

**Leo:** I don't think slide rules are popular anymore, but okay. It's how they work.

**Steve:** Yes. The idea that you can add lengths to multiply numbers is like, what? Because of the special shape of this curve. So it turns out there's, like, a couple simple things that we're going to explain that allow us to determine some really cool things about passwords. But first, there was so much news this week. While still on probation, Symantec has stepped in it again with being caught issuing invalid certificates and not acknowledging them, so it was caught by a third party. Tavis Ormandy, our friend at Google, has found a very troubling problem in the 20-plus million widely distributed Cisco WebEx extension to Chrome, which allows any site you visit to install any malware that it wants on your computer.

**Leo:** Oh.

**Steve:** Not good. We have yesterday's more-important-than-usual update to iOS. There was also a flurry of concern about LastPass metadata leakage in the last week due to a posting that someone put up about their discovery. And so we're going to explain about that. News of the SEC looking askance at what's left of Yahoo after the Verizon acquisition. A very troubling browser form autofill information leakage that actually works, and a way our listeners can frighten themselves and then think about whether they want to leave it the way it is or what they want to do. An improvement coming later this year in Tor's Hidden Services feature to make them less discoverable. China, orbiting a - I can't even believe we're there, but maybe we are - a satellite which distributes entangled photons, as in quantum entanglement, to distant locations on the globe to allow instantaneous undetectable communications.

We're also going to review Heartbleed, which is now coming up on three years old. An interesting new take on the compulsion to produce fingerprints. We're approaching the biggest Pwn2Own competition ever. And we now know how that's going to get laid out. And after all that, a little bit of miscellany, and then we'll talk about computing password complexity. So I don't think anyone's going to find this podcast comes up short.

**Leo:** Lots of good stuff. Anyway, on we go.

**Steve:** So our Photo of the Week is just - it's fun, a snapshot of things to come. It shows, for people who can't see it, a guy with one of our mobile coffee carriers in his hand, apparently in the process of leaving the house. The front door is open. He's turning back to look at this black cylinder which is speaking to him near the front door. And we know that that's probably the Amazon Echo. And it says, "Your phone told your Fitbit that told your Nest that told your Sonos that told me that you owe your wife an apology. Two dozen red roses are only $29.99 for a limited time." And he sort of looks at it with a little worried look on his face, like, okay, you know, is that the case? So, ah, yes.

**Leo:** Love that.

**Steve:** It's all connected, and who knows what's going to happen? And actually this ties into a couple notes we have about that Amazon technology later on in our miscellany.

**Leo:** Oh, good, yeah.

**Steve:** So, okay. As we covered at the time, back in 2015, Symantec was caught having issued some certificates that they shouldn't have, which is of course a big no-no because essentially, as we've often discussed, the PKI, this Public Key Infrastructure for browser certificate trust, the whole thing relies on essentially something impossible, we could argue, and thus its weakness. That is, the perfect performance and behavior of a huge, and I would say ever-growing, except some are dying because they misbehave too badly that they stop being trusted, but hundreds of individual entities all trusted in parallel to never make a mistake in issuing a certificate which asserts the identity of the entity to whom they issue the certificate.

So unfortunately, for the system to work functionally, all of the browsers everywhere must trust all of the certificates issued by all of the certificate authorities everywhere, in this many-to-many mapping where a single mistake can have significant consequences.

Well, basically it violates the integrity of the system for the entity that is named in that certificate. So everyone tries to do the best job they can. But the nature of it requires monitoring and policing.

So one of the things that's on my list of in-depth coverage is a system that Google has designed and has brought up called the Certificate Transparency Log, which is essentially a publicly accessible database of all issued extended validation certificates. At this point they are asking all certificate authorities to post to this certificate transparency log any extended validation certificates they issue.

And when you think about it, this is a nice additional layer. It doesn't, like, lock down anything. But if we didn't have a central visible clearinghouse, then there's no visibility at all into what certificates certificate authorities issue because they have a one-to-one transaction with the person asking for a certificate, and that person then puts it on their web server. And only when people visit do browsers then see the certificate that they were issued. There's sort of no oversight globally available. So this is a really worthwhile first step.

So looking at this Certificate Transparency Log, a security researcher named Andrew Ayer, who is also the founder of a certificate authority reseller known as SSLMate, he was just browsing through the Google Certificate Transparency Log that's maintained by the Google Certificate Transparency Project. And he noticed some oddities. And then when he looked closer, he probably did some data analysis through the log, he found 108 invalidly issued certificates by three Symantec-owned certificate authorities. Nine of those 108 were issued without the permission or knowledge of the affected domain owners. So that's a big no-no. That is, certificate authorities aren't supposed to just make certificates for random domains that they feel like. I mean, they have a role in this larger scheme. And it's crucial that they not step outside of that.

Of the remaining 99 certificates, they were issued without proper validation of the information contained in the certificate, the company information. Now, the problem is that, as we know, these certificates would have enabled their holders to spoof the identities of sites that were protected with HTTPS or TLS communications. And the whole idea, we talk about this often, privacy through encryption is nice. But unless you have authentication, unless you absolutely know as robustly as possible the identity of the other end of the connection, then without that, you have actually no guarantee whatsoever. You may be scrambling your bits, but you might be scrambling it to somebody other than who you expect to.

So authentication is, you know, that's really what the certificate provides, thus all of this drama and why it's so important. So it is the case that many of the certificates issued contained the string "test" in various places. And they were revoked shortly after being issued, in as little as an hour. But that's nevertheless a significant violation of the CAB rules, the Certificate Authority Browser Consortium rules. And as we know, because I went off on Google and Chrome about this years ago, Chrome in particular, this whole CRL set, Certificate Revocation List is utterly broken. It doesn't work at all. And to the degree that it kind of works, it turns out that it's only extended validation certificates that they're concerning themselves with, despite the fact that that's a minuscule fraction of the total certificates in the industry. So the fact that they were revoked really provides no security, I mean, no practical security.

And as we also discussed back then, when a browser tries to verify through the online technology whether a certificate is still valid, unless it affirmatively is told that it is invalid, it fails open. That is, because the whole validation system is still not as robust as it should be, it's just sort of been limping along coming forward over time, the browsers

don't want to inconvenience their users with false blocks of access. So unless it's told it's invalid, it assumes it's not. So the point is revocation isn't any excuse. And no one argues that this wasn't a mistake.

And what's interesting is these were not EV certificates because, as a consequence of Symantec's misbehavior in 2015, Google required that Symantec post all issued certificates into the Certificate Transparency Log, not just extended validation certificates. So this has just happened. It's not clear what's going to come of this. The three CAs that Symantec now owns are the Symantec Trust Network, GeoTrust, and Thawte, spelled T-H-A-W-T-E. So those are all Symantec properties, and they were up to some mischief, which as a consequence of this Certificate Transparency Log we now know. If it weren't for that, and if it weren't for the fact that Symantec were on probation and had to post the brief creation and in some cases miscreation, and certainly creation in violation of the CAB guidelines, if they didn't have to post that, this would have never been seen. It would have just been under the radar.

So here is props to Google for coming up with this concept of making high-value certificates visible in a way that helps us deal with the fundamental problems that we have of the system being fragile by nature. And then also in requiring Symantec's behavior to change in response to some earlier missteps, which did catch, as a consequence, later missteps. And thanks to Andrew - for whatever reason, he was poking around in the log - for finding this and bringing it to the industry's attention. So, and what'll happen is two years ago a bunch of people were fired immediately after this was discovered. I'm sure that Symantec will be, I mean, they're wanting to be responsible. Now that they know this happened, and now that they know we know this happened, I imagine some additional heads will roll. And hopefully the industry at large, the CA industry, is observing this and thinking, oh, you know, that's a doghouse we don't want to be in.

And more props to Google, in this case Tavis Ormandy, who is a bug hunter whom we often are referring to. He found something very distressing buried inside of Cisco's WebEx, which is their web-based teleconferencing system, the WebEx extension for Chrome. And reading Tavis's log of what he did, you just sort of think, I mean, he did some things that were a reach at best, and then they shocked him when they worked. It's like, oh, my lord, this worked?

So what happened was he found what we call a "magic cookie" inside of the Cisco WebEx browser extension. And it's those GUIDs, G-U-I-D, Globally Unique Identifier, which are sort of groups of hex, eight characters in the first group, and then four sets of four, and then eight in the last group. So there's one of those following "cwcsf-nativemsg-iframe" and then hyphen and this GUID. So he thought, okay, that's interesting. So it turns out that magic cookie was an enabling token for Cisco's access to a very powerful DLL which they had installed as part of this browser extension in the user system. This was essentially - I'm not going to use the term "backdoor." We'll call this an escape hatch. This allowed the extension to reach down into API functions in this DLL, which is down in the system, and get it to do things that Cisco's WebEx browser extension for doing teleconferencing needed.

The problem is what Tavis found was a lack of control. That is, this really wasn't that hidden. It was part of the manifest in the browser, so it was discoverable. And anyone else could do what Tavis did. Now, what Tavis arranged was simply, you know, the standard test of can I run calc.exe? And sure enough, without any permissions from a non-Cisco domain, Tavis was able to cause the Cisco WebEx browser extension to run code of his choice, in this case calc.exe. And it looks like it's very possible to cause this to download whatever they want, that is, whatever the user of this API wants, and cause it

to run.

So essentially this would have allowed a no-user-interaction privileged remote code execution on any of the 20-plus million machines where this WebEx extension was present in the Chrome browser, and the Chrome browser was being used. So any malicious website or an advertisement, because this also works in iFrames, could silently install malware in the computer of anyone visiting with Chrome who had the WebEx extension also installed and enabled at the time, as it typically is.

So to Cisco's credit, Tavis posted this secretly under Google's 90-day, if this isn't fixed within 90 days, whether it is or not, it goes public, at the very end of last week. And Tavis was very impressed. They responded within that day with a patch. So there's an update to this Cisco WebEx extension, 1.0.3. That's what you want to make sure you have. So you can use the Chrome UI to browse the extensions you have installed, check the version numbers, verify you've got 1.0.3. So Cisco responded immediately. However, Tavis, while he was impressed with the response speed, he was less impressed with the nature of the response. They asked if it would be okay if they simply restricted the domain from which the magic URL could be used to *.webex.com.

Now, until now it didn't matter where it was being run from, so you had no same-origin protection at all in this, which is what caused this serious concern. So locking the use of that to any subdomain or machine name of WebEx.com does control it to a greater degree. Tavis accepted the update, but hopes - and I got a kick out of this - that the Cisco WebEx.com site doesn't, as he put it, have lots of cross-site scripting problems since, as we know, now the only protection is the same-origin lockdown. But as we've been discussing a lot recently, cross-site scripting allows an attacker to potentially inject their own script that runs as if it was coming from the vulnerable website, which would in this case bypass that somewhat weak protection afforded by this URL same-origin to anything.webex.com.

So, yeah. Basically problem kind of solved. There is a better solution for anyone who is concerned by this because, again, now we're trusting that there isn't a mistake anywhere, and I mean anywhere, on the WebEx.com site that anyone could find that would then allow this to be bypassed. And now the whole world knows about this problem. This was all responsibly disclosed. It was fixed before it went public. But still, now we know that there's a potential exploit if a way could be found to invoke this through WebEx.com.

So there is a better solution. In Chrome, you can use user profiles. So for users that, for example, need to have the Cisco WebEx extension available, they could uninstall it from their primary default profile, create another user profile, switch to that, install it there, and use that user profile when they're explicitly wanting access to the WebEx extension, which would essentially make it unavailable to their normal surfing around the 'Net and daily operations. So anyway, I have a link in the show notes to a page that discusses this in greater detail. There was another researcher who took much greater umbrage at this glitch and detailed exactly how to use the Chrome UI in order to solve the problem, what to do, and also how to use multiple profiles in order to sequester this WebEx extension for those who have to have it installed.

By the way, if it's installed, and you put it in two years ago because you thought you were going to need it and you don't, now, very much like Java, this would be a good time to remove it. Because if you're not actually using it, as a general rule, anything that isn't in constant use does increase the attack surface. And here's a perfect example of a big chunk of attack surface that was just discovered. And in any event, verify that you do have v1.0.3. And if you don't, then by all means update because the earlier versions

don't provide even this weak level of protection.

And normally we don't talk about iOS updates. I run around whenever they are coming out, update all of my various devices. Yesterday's update was a little more important than usual because it contained a raft of WebKit vulnerability fixes. So kernel problems, there were two of those, buffer overflows that would allow arbitrary code execution with kernel privileges. But that would typically, for example, have to be from an app that got through Apple's scrutiny. And certainly when they find a problem like this they go back and check for any apps they might have missed because they didn't know there was something to look for. So that normally doesn't really raise to the level of concern.

There was also a problem in libarchive such that unpacking a malicious crafted archive could lead to arbitrary code execution. And once again, that's something that you want to fix. But once fixed, it's probably not a big problem. The WebKit vulnerabilities, of course, don't require or don't submit themselves to Apple's normal application submission scrutiny and even after-the-fact verification. This is anybody with a version of iOS earlier than yesterday's release, which was 10.2.1, which is what everyone wants because we got 10.2 not long ago, but now we're at 10.2.1.

There were a whole bunch of potential problems found in WebKit, from a number of security people, sort of all over the place, but among them about half of them were from Google's Project Zero. So again, props to Google. They're succeeding in making us more safe. So, for example, there was - and remember that Apple never is hugely forthcoming about this. Their approach is to say, "Oh, this just improves security and fixes some things and adds some features, so please install it," without giving us lots of juicy detail.

But, for example, there was a maliciously crafted web content, able to exfiltrate data cross-origin, meaning that it was a breach of the same-origin policy for data exfiltration. And we know how important it is that we not allow non-source origins to have access to anything outside of their own domain. There was an arbitrary code execution problem as a consequence of a memory initialization issue, which has been addressed. Another arbitrary code execution problem, another cross-origin problem, the ability for a malicious website to open pop-ups - which I guess that's not such a huge problem, but an annoyance if nothing else. Because remember the days when pop-ups, as you and I, Leo, were talking at the beginning of the show about the quaint days 17 years ago of security vulnerabilities, or even 10 years ago. And pop-ups used to be a problem until browsers said, okay, we just can't let anyone who runs a script to pop up junk on the screen.

So just wanted to say there were a bunch of things that potentially have significant impact. So this is one that you want to act on sooner rather than later. And as I've commented before, I proactively tell my various iOS devices to go get themselves updated. It seems to take, I don't know if I'm at the end of a list somewhere, but I don't know why I would be because I've got so many of these things everywhere. But I don't get actually a proactive alert of an update for about a week. So I didn't want to let a week go by if other listeners are in the same category.

Okay, there was a huge amount of attention to a blog posting at HackerNoon.com. And the title was "LastPass Does Not Encrypt Everything in Your Vault." And it was like, oh, my god. Well, now, the good news is it caused a huge furor because it matters. LastPass matters. It is the leading password manager in the industry now. So if there's a problem, then a lot of people care. I care. It turns out, first of all, this is one of those things that happens from time to time that is somebody looking around rediscovers something that is already well known. So if this sounds familiar, it's because we talked about it previously. It's a feature. It's not a bug.

And in fact, as I said, it's a well-known and previously well-examined feature. Some metadata required for the use of bookmarklets and favicons must be accessible and are exchanged with the LastPass servers outside of the user's encrypted storage. And it is specifically because LastPass cannot see into the users' encrypted storage that, if we want the features that - and I don't want to use the term "compromise" because our privacy is not being compromised except to the fact that metadata does represent some leakage between us and LastPass. Not globally, because it is all over TLS encrypted communications. So it's not something that anybody who's sniffing traffic at Starbucks can see.

But what this person discovered was some obfuscated URLs. They were hex encoded just to allow them to be transmitted over an ASCII channel. And they're required for, among other things, the history tracking feature which the user can disable if they don't want that tradeoff. But for example, in the previous coverage there are other things that we know are not encrypted, and we've always known are not. The email address that we use to log in is not encrypted. They have to have that in order to send us notifications and email. The IP address of logins to LastPass is not encrypted because, if we want features like the country restriction, which is very handy, I mean, like, I don't travel out of the U.S. almost ever, so I've got that turned on. I don't want LastPass to even consider non-U.S. domestic logins to my LastPass account.

So that's strong. I would argue that's a tradeoff that's easy to make, just to have my IP address not encrypted. And the IP address of previous website logins. That's what LastPass uses if we have the tracking history enabled, which allows us to monitor for our own purposes, our own monitoring, the IPs from which logins have happened to our account. And under the LastPass Vault > Account Settings > Advanced Settings > Privacy, there is Track History, which you can turn off, if you would rather shut that whole process down. And then that information will not be provided to LastPass. But on balance, it's the kind of thing that, if you're suddenly worried something strange might be going on, to have that history available, I think, again, that's a worthwhile tradeoff.

Okay. So this is just a little quickie. We were talking about - extensively, actually, in recent months - about the problems that Yahoo has had. I mean, yes, massive in scale; but also very questionable in the handling of, or I was going to say the handling of the disclosure. But it would be more appropriate to say the lack of disclosure. Years go by, and no one is told publicly of a breach that they apparently know about, which is really not what you consider responsible behavior.

So the news this week is that the SEC is launching an investigation into Yahoo's improper handling of past cyber attacks. And I noted in some of the - I was digging around to understand if there was anything more here. There was some comment that most of Yahoo's assets were sold, of course, recently to Verizon. So it's not clear what's left. Verizon probably said, "We'll take all the good stuff. You keep the problems with the SEC. Those are yours."

**Leo:** Yeah, I wonder who's responsible for breaches once a company's sold. Certainly not the new owners.

**Steve:** No. It's not, no, exactly. So they just got the goodies and left sort of a damaged shell.

**Leo:** A mess, yeah.

**Steve:** A damaged shell. But what I did find interesting were a couple of little nuggets. We, the nation, the U.S., still don't have strong uniform legislation in place. And following that very widely publicized, and we certainly discussed it on the podcast, the big Target breach, there was some legislation drawn up called the Personal Data and Notification Act, which would have required companies to notify the public of data breaches within 30 days of discovery. It was introduced in 2015, so about two years ago. But it never made it through the House of Representatives subcommittee review. So it never emerged as legislation that could be voted on. And apparently there was frantic industry lobbying that caused the representatives on that subcommittee to say, "We have better things to do."

But it had some teeth. Which is, again, why the lobbying was so heavy. It would have carried - get this - a shockingly serious penalty of up to $1,000 per day per person affected. And it would have superseded the 47 state laws, including the District of Columbia, which have varying standards of when notification must be made and what types of information breach qualifies. So if I read this right, as long as a company notified within a month, within 30 days, they would be okay. But, boy, if they took longer than that, at $1,000 per day per affected person...

**Leo:** Oh, wow.

**Steve:** Yes.

**Leo:** That's, let's see, a thousand, times half a billion, times, oh, what did it take them, three years?

**Steve:** Uh-huh.

**Leo:** Hmm. Maybe Verizon will cancel that acquisition. That's quite a fine. Holy cow.

**Steve:** So, never happened, but it does - I think it indicates, if nothing else, that there was some intent. And I'm watching the actions of this new administration. No coherent pattern yet. No coherent philosophy seems to be emerging that I've seen yet. But I'm interested to see, like, on things like this, where will the new administration come down? So it'll be interesting to see. Certainly, at the moment, having federalized this, and allowing all, you know, individual states to have their own rules is a problem for services like Yahoo, which are, well, I was going to say they're certainly countrywide, but they're also global.

So at some point we're going to need some better handling. And I think, clearly, if there were this sort of statute in place, then we would not have had to wait two years to find out. Yahoo!, as a consequence of the lack of any strong enforceable legislation, just chose not to tell us. Well, if you were being charged $1,000 per day per person whose information was escaped from your database, and you knew it, then you'd be motivated to just say, by the way, we have suffered a breach. We're still investigating it. We'll let

you know more. You know, that'd be better than nothing for a couple years.

Okay. Here's one that's not fluffy. A Finnish web developer and hacker discovered, and I have verified, that several web browsers, which at least include Chrome, Safari, and Opera, but interestingly not Firefox - actually, in this case because Firefox is lagging a little bit in this feature, which now they're glad for - as well as some plugins and utilities such as LastPass, can be tricked into giving away, as in divulging, a user's personal information through their profile-based autofill, auto form fill systems. And the attack is surprisingly simple. Leo, if you go to the link in the show notes here, this GitHub.io link, it'll bring up a tiny little form. You do not want to do this on camera. That is, don't submit the form on camera. But view the source because this is trivially simple to do.

So the attack is surprisingly simple. When a user begins to fill in information in a web-based form containing just simple text boxes, such as their name and email address, the browser's autofill system, when enabled, which is intended to help you to avoid tedious repetition of standard information, such as your address and et cetera, but also including credit card information and lots of other stuff, will populate other profile-based information into any of the other standard identifiable text boxes, based on the text box label.

**Leo:** Ah. So this has hidden fields.

**Steve:** Yes. What the view source shows is that in the CSS - scroll down a bit - they set the offset to, I think...

**Leo:** Fortunately, I don't have any browser-based information in there. You see that all of that stuff is missing in mine.

**Steve:** Correct.

**Leo:** So I'm wondering if it has to do with you allowing the browser to save that information. Because I use LastPass for it.

**Steve:** Correct. Exactly. And so for me, Firefox doesn't have this autofill facility. It's on the drawing board for them. And hopefully now they'll be aware of this and will take some action. So normally all of the form fields come up on the screen. But with CSS you're able to set an absolute offset. So all that page does is set the horizontal position, I think that's what it was, to -550 pixels, pushing them off the screen to the left so you don't see them. But the browser doesn't check to see whether it's visible or not. So you fill in the first one, which is your name, which is normally, you know, it'll give you a dropdown list, and you go, oh, yeah, Leo Laporte. Then, without you seeing it, all these other fields are populated.

And it did it for me, that is, under Chrome. I don't use Chrome for much. But it had my full street address - street, city, state, country, domain, zip code, everything. And if I were a Chrome user, and I had said, yeah, hold onto this other information for me, like credit card number, the CVV, the expiration date, all of that can be captured. And it is a trivial attack.

Now, I should also note, the reason not only did I not want you to, if there was anything sensitive for users to see it because it just comes right up, but there is a site which this GitHub.io page uses to display this. It's a benign site. It's httpbin, B-I-N, dot org. Which is a well-known, very cool, longstanding site which provides an array of tools which echo back browser query parameters. So, for example, if you were doing HTML, and you wanted to see what the POST data was that your browser sent, you could post it to httpbin.org/post. And all that does is it takes the information your browser sent and then sends it back to you in a nice JSON-formatted display, which is what you put up on the screen a second ago. But the point is that it is https://httpbin.org. But it is going to a third-party site.

So I did verify it works. There's no reason not to trust these people. But they're getting any information that your browser happens to fill the forms in with. It would be nice if they had a non-obfuscated version, that is, without the -550 pixel shift, so that you could see for yourself what information was being filled in. That ought to be convincing enough. It's a little more dramatic as it is this way. So maybe one of our listeners will do that because you could see the source. Anyway, it would be trivial to do. So if somebody does that and sends me a tweet or a link, I'll share it.

**Leo:** But you could probably do it just in real-time, using the developer tools. I bet you I can. Let me just see.

**Steve:** Yeah. I wonder if you could see the form populated without sending it somewhere.

**Leo:** Well, I think I can change - can I change these numbers? Yeah, I can change these numbers.

**Steve:** Oh, I see, you mean edit the page text, yes, yes, yes.

**Leo:** On the fly. Yeah, edit on the fly. Not a difficult thing to do, it's just...

**Steve:** In order to bring them back.

**Leo:** ...painstaking. Because there's quite a few of them. I'll do that. You keep on talking.

**Steve:** Okay. So, good, perfect. So again, it worked for me in Chrome. I didn't see anything scary because I've never given Chrome my credit card number.

**Leo:** Well, I know that some of these fields have the designation "LastPass." I wonder if it calls LastPass to pull that information, as well. Or maybe that's just a choice on there.

**Steve:** It doesn't. But LastPass is monitoring the page itself.

**Leo:** Ah.

**Steve:** So when it sees those form fields, it says, oh, I'm going to be helpful.

**Leo:** So any page that has a form could do this to you.

**Steve:** Yes. I mean, this is really big.

**Leo:** It's horrible.

**Steve:** Yes.

**Leo:** Wow. All right. I've modified those. Let's see. Oh, yeah, here you go. Look at that.

**Steve:** Yup.

**Leo:** Look at that. So your phone, your organization, your address. All I did was take these fields and move them onscreen.

**Steve:** Right.

**Leo:** They were off to the left.

**Steve:** Literally right.

**Leo:** Yeah, well, that moved them right. Your address, your postal code, your city. Here's the credit card, or the country, the credit card field. Wow. Wow, it's all there, baby.

**Steve:** Yeah. And, I mean, so here's an example of another feature that was put there for convenience, but has been, well, should we say repurposed? I mean, basically the developers didn't consider that they should provide security for invisible or in any way unseen fields because - so in practice, you would go to a site that simply wants your name. You know, create - you're going to add a post to a blog.

**Leo:** You see this, but you give them this.

**Steve:** Tell us your name. Exactly.

**Leo:** Wow.

**Steve:** Just trivially.

**Leo:** That's horrible.

**Steve:** It really is. It's shockingly simple.

**Leo:** I guess you would go into the settings and turn off form fill.

**Steve:** Correct. So users can protect themselves from this kind of phishing attack by disabling the autofill system within their browser or their extension settings. So that's the way to do this.

**Leo:** That's why it didn't work. I have that disabled already.

**Steve:** Ah. Yes.

**Leo:** I must have been smart in earlier life.

**Steve:** Well, and when I was using Chrome, and I typed in, like, "S"...

**Leo:** I'm sure it's on by default.

**Steve:** Yes. But the next item in the dropdown list was autofill settings. So Chrome was, like, presenting them to me, so it should be easy for a user to decide, I don't think I need autofill until we get this -

**Leo:** Well, and then you can also go deeper into it and see what information Chrome has. I have autofill turned off. But it's been saving stuff. And actually this is all pretty old stuff. So, you know what, Chrome the first time says, do you want me to remember forms so I can fill these in? And I'm sure what I said is no. But I didn't say it. So I'm going to delete all of these addresses so it doesn't by accident - or maybe just put in innocuous stuff; right? Don't have any credit card info on there, which is good.

**Steve:** Yeah, that was, I mean, I was like, oh, wow.

**Leo:** Oh, wait a minute, I do have credit card info. Holy cow. Holy moly. Okay, we

should probably take all that stuff out; right? Holy moly. Oy oy oy. Okay. Golly. So turn off autofill, and maybe as a second level of defense remove all that form information, if it's stored there.

**Steve:** Yeah, yeah. So later this year we have an upgrade coming to Tor's so-called "dark web" technology. That's what the popular press calls it. We know of it as Tor Hidden Services. This change, which is forthcoming, although it's going to require an upheaval, as major changes do, to the existing system, and probably will be able to be incrementally adopted, will require an upgrade to Services' servers.

So just to step back a little bit. As we know, the way that - the Internet was never designed for secrecy or privacy. It was designed for robustness. It was designed to work against all odds. And that it achieves brilliantly and with amazing scalability over time. The fact that we're still using the Internet that we showed in that diagram that had five machines on it, and now it's got - now we no longer have enough IPv4 at 4.3 billion endpoints, IP address endpoints, and it still works. Same system that started off in the beginning because they just did that part right.

What they know they didn't do right, but they weren't trying to because it was a miracle that it worked at all, and as we've often said, is privacy and secrecy. It wasn't intended for that. And so it's always been having a problem with that since. We know that we can make our data private by encrypting it and being sure about who we're encrypting it to. But the fact of the data moving across the Internet is still in plain sight. And we can almost consider this sort of a metadata, that is, okay, there went a packet from point A to point B. Everyone can see that happen. Even though they can't see what's in the packet, the fact of the packet, the metadata of its existence, is known.

So Tor was a system designed to robustly thwart, I won't say "prevent" because, as we know, you can't prevent what's impossible to prevent. But it really upped the ante. The idea with Tor was you would have a large collection of Tor nodes. And Tor, T-O-R, started off as being an abbreviation for The Onion Router, you know, T-O-R, The Onion Router, the idea being that there would be a subset of the Internet that was participating in this Tor blob. And traffic that would go into all kinds of entry points in this big blob and be untrackable and undecryptable as it hopped around through the blob. And then it would come out of some other random place. And so even though it ended up emerging in public, if a whole bunch of traffic went into this black box, and a whole lot of other traffic came out of different points and different times, it would be very difficult to determine which went in, which specific packet went in, and when and where it came out, thus creating a big mystery in the middle of this otherwise public routing system.

Now, of course, there have been lots of attacks of that because, if you bring sufficient resources to bear, as I said, since it's not actually possible to do that perfectly, then there are ways to defeat it. Another service, though, was added. So note that what Tor originally did was create a black box in the middle of the Internet so that things went in, and they came out. But the point was they were coming out and then going to their destination public service. So the fact that you were connecting to that service was - it did a good job of obscuring that connection and your interaction. Good, but not perfect. But the service itself was still public.

So later Tor added this concept of hidden services so that the services themselves, like the websites, could exist in the black box, could be part of this Tor mystery, so that then your traffic goes in, it still bounces around to scramble anybody who's trying to track it, but then it ends up at a service which is not public, that is, it is part of the black box. So

that further increased the security guarantees that Tor was providing.

The problem was that, again, because the Internet wasn't designed for this, and arguably the original first-generation hidden services, Tor Hidden Services system didn't do everything it possibly could. It turns out that at last year's Black Hat a paper was published that demonstrated there were, of about 300 directory servers, which is sort of the equivalent of DNS for Tor's Hidden Services. These are hidden directory servers. Of the 3,000 that exist, a little over 100 of them were found to be suspicious, that is, they appeared to be using technology to unmask the existence of hidden services, which the system was explicitly designed to protect. So there have been famous hidden services, and they've got funky-looking domains. They all end in dot onion.

The WikiLeaks anonymous upload system is at the well-known Tor Hidden Services dot onion domain of "wl," as in WikiLeaks, "upld" as in upload, and then "3ptjvsgwqw," which is just gibberish, in order to make the entire domain name longer, 16 characters. Silk Road, that we talked about back in its day, was "silkroadvb5piz3r.onion." So that's what onion addresses look like. And Tor users were able to use those within the Tor network to get to a server where their traffic never emerged back out onto the public 'Net in the same fashion.

So it turns out the idea, the original concept was only if you knew that crazy, wacky, unobvious, difficult-to-discover domain name, or kind of URL, 16 characters, maybe with something on the beginning, but then with gibberish attached to it, dot onion, only if you knew that could you find the services because they were supposed to be hidden. Well, it turns out that, even without knowing a hidden services address, it has been possible for hackers, law enforcement, security firms, snoops, and others to discover the existence of these services independently.

So the developers involved with Tor don't like that. They have said the only people who should know about your hidden service are the people you tell about it. And while that's a pretty simple concept, they admit, it's currently not true. So the next generation of hidden services will use a new method to protect the secrecy of those addresses. Rather than declaring their dot onion address to a hidden service directory within the Tor system, next-generation hidden services will instead derive a cryptographic key from the known dot onion address. That is, they know their dot onion address, but that's the secret they will keep.

They then use that to produce a cryptographic key, and that key will be placed in the next-generation hidden services directories, rather than the dot onion address itself. What that means is that any Tor user who in advance knows the name of the hidden service they want, can perform the same derivation to check the key and then route themselves to the correct hidden service. So essentially it's creating another level of cryptographic indirection which prevents the kind of discovery that researchers now know has been going on. Since the hidden service directories cannot derive the dot onion address from the keys which they hold, only those who know the hidden services keys can discover the hidden services address.

So the original Tor project cofounder, Nick Mathewson, was quoted saying: "The Tor network is not going to give you any way to learn about an onion address you don't already know." And then, finally, this next generation of hidden services will also be switching from using 1,024-bit RSA encryption keys to shorter, but tougher-to-crack, and of course my favorite, the ED25519 elliptic curve, the behavior of which is what enabled the whole SQRL concept in the beginning. And of course we're seeing this thing really taking over the industry in the last few years because it offers so many benefits.

So these changes, of course, mean that the hidden service URLs will change, too. They're expanded from 16 characters, as they are currently, to 50, five zero. But Nick argues that that change doesn't affect the dark web addresses' usability since 16-character gibberish is already too long to memorize. So they will be 50 characters - substantially longer, but substantially more secure - and use a technology that prevents, essentially, any form of discovery of the existence of these hidden services.

And the plan is that - oh, and in fact I think it was Nick himself who maintains an open family wiki and an open family calendar as a Tor hidden service. And all of the members of his family know the secret address and are able to easily and freely publicly share it. There's no password on that server because only members of his family are able to get to it. So an interesting idea. It'll be fun to see this happen. And I'm sure we'll be talking about this later in the year as this thing gets deployed.

Okay. Now, there isn't much to say in detail, but just I was caught off guard by the fact that such a thing exists. China has actually put into orbit, it exists, it's called the Mozi, I think it is, M-O-Z-I. It is an operating proof-of-concept quantum satellite, which is now operational. So quantum entanglement is interesting, and I'm not going to get into the details because I'm not qualified. I'm not a quantum physicist. And there's more that I could say. But the concept is it is possible to produce a pair of photons. And it actually applies to things other than photons, as well. But photons are easy because they're particles of light. So you can produce a pair of photons which are entangled, as the term is, at the quantum level.

And, for example, if a single photon had a property that was equally likely to be true or not, and another photon had a property, the same property, equally probable to be true or not, then taken together there are four possible combinations for that property of two photons; right? They could both not have it. They could both have it. One could have it; the other wouldn't. Or the other could have it, and the other wouldn't. Four possibilities. Well, normally, if they're not entangled, we would expect those properties - and the properties individually had a 50/50 probability - then the combination of them would have 25% probability, all equal one quarter probability. When you entangle them, what happens is you disturb that equal distribution of probabilities such that they are interconnected, and now there's 0% probability of two of the four shared states, and 50/50 of the other.

Okay. So what this means is - and there was an experiment done a few years ago that was mindboggling, where two entangled photons were sent in opposite directions and polarized. And at some time later, that is, downstream, the polarization of one of them was rotated, and the polarization of the other entangled photon spontaneously changed. And here's where it gets really spooky. Not at light speed. Instantaneously. So there was some spooky connection between them that existed, that was created and then demonstrated.

So what China has in orbit now, and I guess experiments will start, is they have put into orbit a photon entanglement device which, if this is credible, is supposed to be beaming down to the Earth, to specific points, pre-entangled photons such that the recipients at distant locations will be receiving photons that in principle allow FTL, that is, Faster Than Light, transmission of information by, just as was done with this tweak the polarization of one photon and the other one responds instantly. Although lots of problems to be solved. Bandwidth is probably low. You're not going to have a video conference over this thing anytime soon. I mean, we're still way in the weeds. But just the idea that this, like there's something up there now that's going to be sending entangled photons down to multiple locations, that's mindboggling. So I had to share that.

Okay. It's been nearly three years. It was April of 2014 that the big news of Heartbleed hit. And we talked about it a lot. To remind people, this was a bug found in the highly popular OpenSSL library, which was found through code inspection, and there was, I mean, so it was clearly a bug. The question was, was it exploitable? And for quite a while people thought, eh, well, okay, it's bad. We're going to fix it. But we don't really think you could actually do anything with it.

So a number of organizations, and CloudFlare, I think it was CloudFlare was one of the most notable, put up a test, a contest, said here is a Heartbleed-vulnerable server on a domain we just created. Can you get its private key? And of course we know that's the keys to the kingdom. The private key is what you never want to allow anyone to exfiltrate from your server because that then allows them to defeat the guarantees that the certificate that the private key was made from, or that the certificate was made from the private key, from being honored. So as we know, yes, it turns out there was a low statistical probability. This essentially allowed an exfiltration of large blocks of the server's private memory.

So the question was, okay, that's not good. But is there anything in that private memory that can be accessed by this bug that we care about? And sure enough, keys, cookies, passwords, all kinds of debris, but some that was valuable, and debris you didn't want, you never intended to have escape was exfiltratable. So many people, I mean, this came to everyone's attention. The EFF and Bruce Schneier all considered Heartbleed to be a catastrophic flaw. We were all shocked that it had been around for so long. And there was an immediate move to fix this.

The curve, however, of rate of fixing has not been kind. About a month and a half later, toward the end of May of 2014, we went down from around half a million of the Internet's secure servers at the time of the first revelation, about six weeks later we were down to 1.5% of the 800,000 most popular secure, TLS-enabled, secured websites still vulnerable to Heartbleed. Shodan published a report, I think it was yesterday, that showed that here we are today, nearly three years later. If this report is correct, and I'm a little skeptical because the numbers are still so horrifying, this shows Shodan identified at least 199,500 websites still vulnerable. Most are in the U.S.A. I've got a link in the show notes.

In fact, I can say it, if anyone's curious: www.shodan.io/report/ and then all - no, I was to say "all caps," but it's not - DCPO7BkV. That's their report. Really interesting graphics. It's very easy to scan and look at. But I think it was 43,000 of this just shy of 200,000 servers are in the U.S. I think the second most prevalent was in South Korea. The largest holder of vulnerable servers, 6,376, was SK Broadband, South Korea Broadband. But what troubles me, and the reason I'm a little skeptical, is that this shows Amazon.com in the number two position with their AmazonAWS.com.

**Leo:** So lots of people host on AWS. So it's not - well, maybe not because it's AWS.com. I don't know.

**Steve:** Yeah, see, so it's Amazon servers.

**Leo:** Yeah, but we're on Amazon servers. I mean, a lot of people are on Amazon servers.

**Steve:** Okay. So what I'm saying is what they may be doing is working, for example, from version number. They have to be using some sort of a priori information from the server because the nature of Shodan, I'm sorry, the nature of Heartbleed...

**Leo:** Well, they reverse DNS, so - right?

**Steve:** No. The nature of Heartbleed doesn't allow you to easily determine the actual presence of the vulnerability. So it's probably based on the version that the site is publishing of OpenSSL. And we also know that Amazon AWS, because we talked about this before, they've already spawned their own SSL or TLS stack because OpenSSL had so many features they weren't using. So my strong suspicion is that this is not an accurate report. It got a lot of press coverage about 200,000 websites still vulnerable. I doubt any of Amazon's vulnerable, despite the fact that this says 5,163 are. It almost certainly is the case that Shodan is just doing a scan of SSL or TLS version numbering, like what is the library behind it. And Amazon probably has their own independent versioning that happens to collide with what OpenSSL was. At least that's my guess. Again, I could be wrong.

If this is true, this is horrifying. But it does sort of, I mean, certainly we know from all our experience that there will still be problems, that something like Heartbleed never completely goes away. The servers in the closet aren't going to be updated. On the other hand, the server in the closet probably has low value. And it's still, even in the case of a high-value target, it was very difficult, but not impossible, we found, to exfiltrate data using the Heartbleed vulnerability, which is now three years old.

So, okay. Here's a couple fun things. A Minnesota court on the Fifth Amendment and compelling fingerprints to unlock a phone. We've been having fun talking about biometric or not. You can't be told to divulge a password that you have memorized because that's clearly testimony. But a fingerprint, it's felt, is not testimonial. And so as we know, courts have recently been using the argument that forcing a person to use their finger to unlock a phone is not afforded Fifth Amendment protection against self-incrimination because using a finger does not qualify as being testimonial.

So a recent case, the State v. Diamond, who was the defendant, was decided just yesterday by this court in Minnesota. Actually it went to appeal. In the original case, the defendant Diamond initially refused to comply. He said, no, I'm not going to let you make me use my finger because I don't want to. The judge...

**Leo:** No, you don't have to.

**Steve:** Exactly. The judge explained that using his finger was nontestimonial, so he could not plead the Fifth, and he would be held in contempt of court, probably means put in a cell until he changed his mind, if he refused. So he complied. But then we presume he got a fancier attorney, who came up with another angle, and they appealed that decision. On appeal, Diamond argued that the government had violated his Fifth Amendment rights because the government made Diamond select which finger to use.

**Leo:** Oh, wow. That's not going to - that's interesting.

**Steve:** Specifically, Diamond argued...

**Leo:** So he's testifying by choosing the right finger.

**Steve:** Correct. He had some information that they didn't have.

**Leo:** That's really good. I like it.

**Steve:** Isn't that great? I love that. He said, quote, he "was required to identify for the police which of his fingerprints would open the phone," and that, quote, "this requirement compelled a testimonial communication." So everybody, just keep that in mind. I thought that was very clever. It's like, okay. Yes, I do have fingerprints. But I don't want to tell you which one. And so we're back to something you know.

**Leo:** Interesting.

**Steve:** Very cool. Okay. In March we have the CanSecWest in Vancouver 2017, which is the 17th annual security conference. And part of that is the Pwn2Own competition which is now going to have its 10th year. And, boy, a lot has changed in 10 years. That first year, and we've been covering it because it's always fun and interesting, if nothing else to see how quickly and sadly our browsers that we wish were more secure or, well, we wish they were perfectly secure, how quickly they're just cut through. The first year the prizes were a laptop and $10,000. Whoo, 10 years ago; right? Last year more than $450,000 in cash and prizes were awarded over multiple categories. So a lot of evolution in nine years.

And very much as we were talking about software and security seven years ago when, "Oh, look, an email macro, how quaint," 10 years ago, for example, a single bug was used, or a vulnerability was used to exploit QuickTime. Last year a significant chain of interconnected vulnerabilities was required to complete a compromise and fully win a category. So everything's been getting more secure, but we've got a lot more people looking at it. And with increasing prize awards and prestige comes a lot more attention.

So this year we now know the so-called Zero Day Initiative, who are the people who put on the Pwn2Own and organize this, this year the Zero Day Initiative will be offering more than $1 million, spread across five categories: Virtual Machine Escape, obviously from guest to host; Web Browser and Plugins; Local Escalation of Privilege; Enterprise Applications; and Server Side. And the first two are the most interesting. So in VM Escapes, Virtual Machine Escapes, in their laying out the rules they explained that VM Escapes were first added last year with VMware. And remember that we've talked about some fun ones, like I think it was the long-unused floppy disk driver, the virtual floppy driver was sitting there in VMware, and turns out it wasn't secure. And so it was possible to leverage something you absolutely didn't even need. Again, minimize attack surface wherever possible. But it was there, and that was a way to get out of, to break out of the VM containment and get down into the containing hypervisor because that's where the actual driver was residing. So if you could arrange to execute code in it, you broke out of a virtual machine containment.

This year, in addition to VMware, it's being expanded to include Microsoft's Hyper-V. An

attempt in this category must be launched from within the guest OS from a non-admin account and execute arbitrary code on the host, that is, the underlying operating system. Both the guest and the host operating system will be running the 64-bit versions of Windows 10. A successful exploit in either of those VM products, VMware or Microsoft's Hyper-V, will net $100,000 for the contestant, plus what they described as a "lucky 13 Master of Pwn points." They have another thing in addition to dollars. You get these Pwn points.

The second category was Web Browser and Plugins Exploits. And a demonstrated exploit in Microsoft Edge gets you $80,000 and 10 Master of Pwn points; an exploit of Google Chrome, the same $80,000 and 10 Master of Pwn points; Apple Safari, $50,000 and 8 Master of Pwn points; Adobe Flash in Microsoft Edge, the same, $50,000, 8 Master of Pwn points; and Mozilla's Firefox, $30,000 and 5 Master of Pwn Points. And this is interesting because, as we know, the award is regarded as proportional to the difficulty. So here's Microsoft Edge and Google Chrome at parity as the two highest awards at $80,000 and, sadly, Mozilla Firefox at the other end at $30,000 because it's regarded as today not as difficult to find a problem with as Edge and Chrome. And Safari is stuck there in the middle at $50,000, along with Flash in Edge, also at $50,000.

And then in addition they say: "Moreover, contestants may earn an additional $30,000 if their entry achieves system-level code execution on Windows-based targets, or will receive an additional $20,000 if their entry achieves root-level code execution on macOS-based targets. And the Windows-based targets will be running in a VMware Workstation virtual machine. If the contestant escapes the browser or plugin AND the containing VMware Workstation virtual machine to achieve code execution on the host operating system, the contestant will receive an additional $100,000." So this is going to be fun to see - some serious money and lots of motivation.

**Leo:** Can you use the amount offered as a rough gauge of how easy to hack these platforms are?

**Steve:** Yes.

**Leo:** So the more you get, like if you hack Apache, wow, you're good because that's the biggest prize. Or Hypervisor or...

**Steve:** Well, if you hack Edge and Chrome at $80,000 each, those are regarded as the two hardest to hack. And unfortunately, Firefox is regarded as the least difficult to hack. That only gets you $30,000.

**Leo:** Right.

**Steve:** Yeah, so that's interesting. And I think we're going to have a fun competition. I think it's in March.

A couple little bits of miscellany. Oh, I got a request or a tweet from a Pete Stringer, who asked if I was able to put details of the router that we've been talking about so much in GRC's Linkfarm, and I did. It's there now. And that is, of course, the Ubiquiti EdgeRouter X. So remember the Linkfarm, GRC.com/linkfarm. That's become a repository of these

sorts of high-value things. Somebody also, I saw a tweet go by, I don't think I responded to him, what was the site that contained that broad spectrum of security information. And that was PrivacyTools.io. But, for example, it, too, is there on the Linkfarm. So if there's something that's like one of our core things that this podcast gets excited about, check GRC.com/linkfarm, and you may be able to find a direct connection to it.

And speaking of our cartoon at the top of the show, I did want to note the news, for those who hadn't already heard, that the Amazon - I guess I have to say "Echo"; right? Is that still okay? I won't say the "A" word.

**Leo:** Yeah. Now you can say "Computer."

**Steve:** Yeah, well, yeah, but now that's a problem. I tend to use the term "machine," so we're not in danger. But it turns out that, the point is, they have added to - so now there is the "E," you can wake up the Amazon Echo device with that word, the "E" word; the "A" word, by Amazon's own name, Amazon; and Computer. So for those of us who appreciate Scotty talking to the mouse of the Macintosh on "The Voyage Home," "Computer" - and people are looking at him like, okay, is this guy nuts - we can now wake up our Amazon Echo by addressing it by "Computer." Apparently that's - and of course that applies to Dots, as well. The Dot is v48.12. And I don't know if the Dot and the Echo are synchronized because I just saw some postings about that. So if anyone's interested, that works.

And then, in a related note, someone tweeted me, and I think he wasn't aware of previous dialogue about this. His Twitter handle is Brad Online, or his name Brad Online. He said: "@SGgrc, I've noticed Amazon ads for the Echo do not trigger the device."

**Leo:** Yeah. They used to, but of late they don't.

**Steve:** Yes. It turns out - and so he said: "I hypothesize there is a tone that plays to kill the response." Well, as it happens, I ran across some dialogue that appeared to be authentic. I haven't verified it myself. But there is, there appears to be, a deliberate notch at 5kHz in the spectrum of those commercials. And so it used to be a big problem. It caused lots of ruckus. I looked back in doing some googling, there was, like, tons of stories about how the early Echoes were being fired off by the Amazon commercials all the time. Amazon responded. And I think what they did is they, in a firmware upgrade, I mean, the firmware is having to do a spectrum analysis anyway.

The way you do this kind of speech recognition is you chop the spectrum up into pieces and analyze the varying power contained in different little slices of band within the spectrum. And so it would have been trivial for them to remove a small notch, which would not affect the audio we hear at all, but it would stand out glaringly in a spectral analysis. And there are people online who have done this and found a missing chunk of audio spectrum in the commercials for this device that allow it to speak the words and not have the device respond. So just a cool little bit of trivia.

And believe it or not, I found yet another interesting application for SpinRite as a consequence of somebody who listened to last week's podcast. He didn't give me his name, and he sent it as "SM," but looks like his first name is Sergey because that was in his email. And he's in Lake Stevens, Washington. He was wondering about cabling errors, which is what we discussed last week, or recently. He said: "Hey Steve, I just finished" -

oh, yeah, it wasn't, it was recent, it was 593. "I just finished Episode 593, and I think you're wrong on cabling errors. I'm a listener since Episode 1 and a SpinRite user. I played with SpinRite v6 a lot. I even" - get this. "I even bought a large batch of 'completely dead' drives from eBay, and I was able to use SpinRite to restore most of them to full error-free operation." So that had never occurred to me that eBay would sell a big box of dead drives, and you could use SpinRite's magic to bring some subset of them back to life. They probably didn't cost very much, if they were dead. But now they're alive. Then you just stick them in your Drobo, and off you go.

Anyway, he says: "But while SpinRiting a few of the" - oh, he got 36 drives - "a few of the 36 drives I was getting cabling errors. So I would swap cable and try again. I even bought two different brands of SATA cables so I could use them for swapping. And despite everything I could, a couple of those drives were still showing cabling errors. So it looks like it's not always a cable's fault for cabling errors. What do you think? I think it must be something inside the drive. Please explain if you can. Can't wait till the next episode. Thanks."

Okay. So, correct. I called it "cabling errors" because "cyclic redundancy check" wouldn't have much actual meaning for the audience that I've aimed SpinRite at, which is "I want you to fix my drive, please." And I'd say, "Yes, just press this button," and off it goes. So almost always, but almost always, the cause for a CRC, a communications error, is the cable. But it doesn't have to be. And if you've got a really dead drive that lightning struck once, and there's a big welt in the cover, and black soot and a little crater, and you still think that's a good place to store your data, and you're getting cabling errors, well, yes, it was probably the lightning strike that didn't do the drive any good.

The point is something is causing there to be a fault in the data transmission from between the endpoints. It's going to almost always be the cable or the connection. So you just pull the cable off and reseat it, plug it back in again, problem solved, 99.999% of the time. But it doesn't have to be that. It could certainly be that something just killed some aspect of the drive, that maybe a bit is stuck. So when that bits needs to be other than in its stuck state somewhere, it gets an error. So I did come up with something that was much more meaningful than saying that there was a cyclic redundancy check problem, which is mostly accurate. But there are certainly other possible causes for receiving that.

And, finally, a couple tricks for password complexity. This was a heavily tweeted and emailed in the mailbag question. I just found two representative tweets. James Brooks tweeted: "If I use a pseudorandom word generator to string together five-plus words for a passphrase, is it still weak?" And he's referring to Episode 594. Ben Maughan said: "Love SN, listened for years. Character-based passwords just use a different dictionary from word-based passwords. Words can be as good." And then in a follow-up tweet he said: "Actually, Oxford dictionary estimate about 100,000 nouns, so a random five-noun password is better than 12 random characters, I think." So I thought, okay.

**Leo:** Wow, boy, that just doesn't sound right.

**Steve:** I know. And so...

**Leo:** I think some of this is coming from the website, what is the name of it, for these passwords?

**Steve:** Diceware?

**Leo:** Diceware.

**Steve:** Yup.

**Leo:** So I'm really glad you're going to address this because it doesn't - but I want to hear what you say.

**Steve:** Okay. So here's what we want to do. We want to understand, first of all, we agree that the individual items of the dictionary - whether they're characters and the dictionary is the alphabet, or they're words and the dictionary is an actual dictionary - that they are chosen with equal probability and no association, that is, remember, if you start making a phrase, if you start making a sentence, now they're no longer independent. And as I did say a lot last week, it is crucial that there not be linkage between them.

So anything, I mean, even one of many people's favorite idea is to use the first character of a well-known phrase. Well, the problem is well-known may be well-known not just to you. And so it's certainly possible that a block of things to test are the first letters that people tend to choose, of phrases that people tend to choose. And we know that there's some weird astral connection between people because too many people chose "monkey." Like, why "monkey"? So there's never been an adequate explanation for the fact that "monkey" was number five on the password lists for so long. Just it's bizarre.

So, okay. So the choice has to be, of the individual items, has to be random. So the question is, how many, or what's the means or the approach for determining how many possibilities there are? Well, we know that, if you take the number of words, like 100,000, and there's only one of them, then there's a one in 100,000 chance of guessing it. If there's two of those 100,000 nouns, now it could be any 100,000 for the first one; and, because they're not interlinked, there's no reduction in entropy for the second. So it's another 100,000. So you multiply those two 100,000s.

And if you have a third noun, now, again - and again, no linkage between them so the third one is every bit as likely as the earlier ones, now that we have all of those two 100,000s multiplied another 100,000 times. So again, we multiply. In other words, it is the total number of combinations is the dictionary size raised to the power of the number of objects - the number of nouns, the number of characters, and whatever. So, for example, in the case of an alphabet, where our dictionary is the set of available typeable characters, we know that there are 95 of those - lowercase alpha, uppercase alpha, the digits zero through nine, and then all, I think there's 33 special characters that can also be typed. That sums to 95. So there you've got 95 times 95 times 95, that is to say, 95 to the power of however many characters in your string.

But there are a couple simple math tricks that can be used because, for example, we'd like to know what is the - because we think in terms of 128-bit encryption key, or 96-bit, or 64-bit, or 256-bit. The term "bit" is a binary digit. That's what "bit" stands for. And so how many binary digits or bits is a password made up of objects from a dictionary of a certain length equal to? Well, it turns out there is a cool trick that involves logarithms. As I mentioned at the top of the show, a slide rule has a logarithmic scale. And the way the slide rule works is you are adding two distances to get the sum of the distances. But

you're adding them based on the scale, which is logarithmic.

And it turns out that an interesting property of logs, I mean, THE interesting property of logs is that the logarithm of a quantity A, added to the logarithm of a quantity B, is equal to the logarithm of their product. And that's the secret of a slide rule. That's how you can linearly add two things, two quantities that you find on the scale. And in doing that you are summing the logs of each, and the result is the product of them. Which is how addition gets turned into multiplication. And so you can multiply and divide. Division is just the reverse. It's subtracting logs. So there's that.

But the other cool trick is that you can change the base of a logarithm by dividing by the log of the base you want to change to. I'll give an example. So say that we have our alphabet of 95 typeable characters. How many bits is one of those characters equal to? Using any calculator, and it doesn't matter whether you use the natural log, which is often LN on a calculator, or the log base 10, which is typically LOG. You have to be consistent, always use the same one. But if you take the log of the size of the alphabet, 95, divided by the log of 2, what you get, just instantly, is the equivalent number of binary bits required to enumerate the items in that alphabet, that is, those 95 items.

So, for example, in this case the natural log of 95 is 4.554. The natural log of 2 is 0.6931. When you divide 4.554 by 0.6931, you get 6.57. That's the exact number of binary bits represented by an alphabet of 95 characters. So if you have 12 of those, you simply take 6.57 times 12. And that gives you the equivalent bit strength, in the terms we're often talking about, of 12 characters, where each character can be any one of 95. And you could always take 95 to the power of how many characters. You could do that. Or notice what I did was I took the log of the size of the alphabet over log of 2 to get the equivalent in bits.

Well, you could take the log of the size of the alphabet divided by the log of 10, and what you get is the equivalent number of decimal digits. So you don't even have to do a raise it to the power. You simply divide by log of 10, rather than log of 2, and it converts it, essentially, to an alphabet of decimal size or, in the case of log of 2, an alphabet of bit size to get the equivalent. So what this simple little trick allows you to do is to easily, yourself, just with any calculator, and of course if you didn't have a physical calculator or one in your phone, of course I love 42 PCalc on iOS. And do you know if it's available on Android, Leo?

**Leo:** No, it's not. It's iOS only, yeah.

**Steve:** Oh, shoot. Yeah, it's...

**Leo:** But there are good calculators on Android, as well.

**Steve:** Yeah, yeah, exactly. And you can certainly find an online calculator. If anyone is curious, anyway, that's how this works. So you would take 100,000 and take the log of that and then divide that by the log of 2. That gives you how many bits per noun. Then you decide how many nouns you need. But your intuition is right, Leo. People think that 100,000 is a lot. But the problem is, as anyone who plays with this math will now discover, to get an equivalent strength from words, compared to high-entropy large alphabet character set, requires just too many words because there isn't that much entropy per word.

But again, just dividing by the log of 2, dividing the log of the size of the alphabet by log of 2, that'll tell you - and I've been talking about combinations. But that's entropy. It's the number of equivalent bits of entropy, given that you have a well-chosen, equally probable choice of objects from your set. That gives you the measurable entropy of a single one of those. And so if you have 10 of them, you just multiply that by 10 because entropy scales linearly that way. And so people can play with this themselves and gauge whether they think words are as good as they would like them to be. My passwords are all relatively short, but high character set in terms of values per location, no intercharacter coupling, no tricks, just obtained randomly. And of course the problem is people say, oh, well, yeah, but I could use lots of words. Except then you hit a website that says, oh, sorry, you're limited to 20 characters. Well, now you can't use 15 words and 200 characters.

**Leo:** So Wolfram Alpha has a password strength calculator. So you can just go there and do it.

**Steve:** Does it actually do the math? Does it show you what the...

**Leo:** I presume it does. Let me see. What is it, horsestaplemonkey. I don't know. Let me just - I'll show you, yeah, I mean, all of Wolfram Alpha's stuff does that kind of stuff. Let me see if I can...

**Steve:** I don't think that's really what we want. Or, I mean, that could be useful and valuable also. But what we really want is just a simple logarithm, just to be able to determine, to know what it is that we're getting.

**Leo:** Yeah. I don't know how they calculate it. I'm sure they'll say. And I presume that they're smart enough to do the right thing. I don't know. I don't know what the - "very weak." I would guess that they're using something similar to that, your system. But anyway, there's another way to do it.

**Steve:** Yes. There are many.

**Leo:** I like logs. I like logs.

**Steve:** Oh, they're just fabulous.

**Leo:** So I shouldn't believe the FAQ at Diceware or - because they imply that this stuff is very, very secure; right? They're just underestimating the computational power of an enemy.

**Steve:** Well, and I think the problem is practicality. We do know that we are often hit with maximum password size limitations. So if you're limited in the size of the password you can use, there isn't room to have as many, I mean, words are low entropy. No

question about it. You could say "gopher," and everybody knows from that little bit of sounds what those characters are to make up the word "gopher." So the problem is individually they're low entropy. So the only way to make up for that is to use a lot of them. Well, but if the website you're trying to use it at restricts you in total length, you don't have a choice. You cannot use low entropy if you don't have much size. Which is why I just go for maximum entropy, smallest size. And this little trick allows you to determine how much entropy is in different things that you want to try. And so that's just what I wanted to share.

**Leo:** Good. And that wraps this sucker up; right?

**Steve:** Yes.

**Leo:** Nice and tidy with a bow. We do this show every Tuesday, just so you know. And if you want to watch us do it, you can because we stream it live on YouTube, YouTube.com/twit; and on our website, TWiT.tv/live; and, you know, all those places. So you can watch it live. The time to tune in would be roughly 1:30 Pacific, 4:30 Eastern, 21:30 UTC, thereabouts, on Tuesday afternoons. Of course you can always get it on demand and have it at your leisure, whenever you wish, from the website TWiT.tv or from Steve's website, where he puts it all up on there, along with transcripts, too, at GRC.com. You can also go there to get SpinRite, the world's best hard drive maintenance and recovery utility, Steve's bread and butter, and lots of stuff like passwords. Steve's password generator is awesome and truly random.

**Steve:** Yes.

**Leo:** That's the key. We want to remind you we're doing a survey of our audience. We'd like to hear from you so we can give you more of what you want, less of what you don't want. And, yes, our advertisers are curious. Are you men? Women? Aliens? Who's listening? It's very short this year. We decided to make it as easy as possible, just a few basic questions at TWiT.tv/survey.

If the question doesn't apply to you, or you don't find an answer in the options that you like, just skip it. That's fine. TWiT.tv - people say, "Well, I can't answer this." One guy said, "You asked me what I do for a living, but I'm retired. That's not an option." Okay, just skip it, then. That's fine. You don't have to answer every one of them. TWiT.tv/survey.

I think we ask - I guess "retired" doesn't show up. I don't know. I didn't design it, so I'm not going to speculate on why we ask the questions we do. But I think that it was questions that we need to know, primarily for advertising, I would guess. And certainly I use it for programming. I think that's of use. But mostly, if you just listen to the show, then I'll know you like it. That's really all it requires. Thank you, Steve.

**Steve:** Okay, my friend.

**Leo:** Hope you feel better, and we'll see you next week.

**Steve:** Perfect. Will do. Talk to you then, buddy.