# SECURITY NOW!

Transcript of Episode #905

## 1

**Description:** This week, in a necessary follow-up to last week's "Leaving LastPass" episode, we'll share the news of the creation of a terrific PowerShell script, complete with a friendly user interface, which quickly deobfuscates any LastPass user's XML format vault data. What it reveals is what we expected, but seeing is believing. Then we're going to examine the conclusions drawn and consequences of the massive amount of avid (and in some cases rabid) listener feedback received since last week, and some of the truly startling things that listeners of this podcast discovered when they went looking.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-905.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-905-lq.mp3

SHOW TEASE: It's time for Security Now!, the shortest name in the history of Security Now!. Steve Gibson will explain what "1" means and give more very important advice. In fact, honestly the story, the LastPass story gets worse and worse. If you were a LastPass customer, or still are, you've got to listen to this episode. Security Now! is next.

**Leo Laporte:** This is Security Now! with Steve Gibson, Episode 905, recorded Tuesday, January 10th, 2023: 1.

It's time for Security Now!. Yes, it is. You've been waiting all week, I know you have. Security Now! maven, legend, man about town, Steve Gibson is here to talk about the latest security news. Holy cow, Steve. Last week's Leaving LastPass episode was a biggie.

**Steve Gibson:** Oh. Yeah. It absolutely broke all of the records that we've had for, I mean, just judging, you know, my immediate feedback is to look at the number of likes and retweets on my weekly announcement of the podcast. And last week's was about eight times more than we've ever had before. And actually as a consequence of that and the nature of the feedback that I received through this next week, we're going to continue because what happened was interesting. We have to have a follow-up to last week's Leaving LastPass episode.

And I want to share the news of the creation of a terrific PowerShell script, complete with a friendly user interface, which quickly deobfuscates any LastPass user's XML format vault data. What it reveals is what we expected, but seeing is believing. And it's a little startling to see what's there with no decryption of the vault needed. Then we're going to examine, as I said, essentially the conclusions drawn and the consequences of the massive amount of avid, and in some cases rabid, listener feedback received since last week. There were some truly startling things that listeners of this podcast discovered when they went looking. So we've got a great Picture of the Week. And this episode's title is unique for us. It's just a single digit, "1."

**Leo:** I thought I made a mistake. I thought there was an error, typing the title. It's just "1." Number one.

**Steve:** Security Now! Episode 905 for January 10th, 2023, titled "1."

**Leo:** Okay. Okay.

**Steve:** And everyone will find out why in a little bit.

**Leo:** There was a Broadway show called "Nine." And now there's an episode of Security Now! called "1." No one knows why, but you will find out in a moment, shall we say. And I'm very curious to hear about what's going on with LastPass, too.

**Steve:** You don't know how curious you need to be.

**Leo:** Oh, geez.

**Steve:** There's something really - something to...

**Leo:** You asked me if I could try this XML formatter on my vault data. And I deleted my vault more than a year ago, and I hope to god so did LastPass, but that's a question for later, too. I think the calendar, the clock on the wall, the atomic clock says it's time for our Picture of the Week, Steve.

**Steve:** So we've actually shown this one before, but I ran across it again, and I thought, this is just so good. Sometimes things are just so clever.

**Leo:** And this is not a Photoshop. I own both these books.

**Steve:** Yup. I do, too, yup. So on the left we have O'Reilly's official bible, "JavaScript: The Definitive Guide." And it's got a big rhino on the cover. And it looks like maybe it's about three inches tall. I mean, this is like...

**Leo:** You could use it as a doorstop.

**Steve:** Yes. You could use it to keep your car from rolling down a hill.

**Leo:** Yes. Yes, you could.

**Steve:** Okay. And on the right we have, also from O'Reilly, written by Douglas Crockford, who's a renowned JavaScript person, this one's got a butterfly on the cover. And it says "JavaScript: The Good Parts."

**Leo:** There's not many is the takeaway on that.

**Steve:** It's about maybe a quarter inch thick. So just, you know, just so much smaller than the definitive guide. Anyway, I just, that's just, as a visual joke, that's just...

**Leo:** It's a great one. And it's true. I mean, those are the books.

**Steve:** Yeah. Okay. So LastPass aftermath. And for those of you who are no longer using LastPass, who moved away from it a year ago, blah blah blah, you know, you may think that some of this doesn't interest you, but we've got, as I said, LastPass aftermath. And there actually will be some math later in this podcast, but there'll be no test on it. So you don't have to worry about taking notes. But there will be something for everyone here.

Okay. At the top of the news for our listeners this week is that my call for the creation of a LastPass vault deobfuscator, you know, it's not a decryptor because we can't decrypt the vault easily. But as we know, a lot of the non-encrypted information was obfuscated. It was just converted into hex, which you should do when you're moving stuff across the Internet, which many of the areas of the Internet still don't support - or maybe that's not true anymore, but everyone still does it - 8-bit bytes, so like 7-bit characters is the reason that weird things have to be converted. And also, if you're using XML, where for example you've got angle-bracketed starting and ending formatting stuff, well, if the content of what you're sending had an angle bracket in it, that would foul up the XML parser. So I asked our listeners, I said, I'm sure there's a bunch of coders out here. Let's see if somebody by this time next week, which is today, can come up with something.

**Leo:** Wow. They responded, too.

**Steve:** Sure enough. I got a whole bunch of people did a bunch of different things, and several listeners produced JavaScript solutions. Paul Holder, who I know you know from...

**Leo:** Love Paul. He's a Java guy, not JavaScript. He's a Java guy.

**Steve:** Yes, exactly. And in fact he wrote a portable solution in Java. Several others wrote solutions in C, C++, and C#. And I was sure that would be the case.

**Leo:** We've got some real coders out there. That's great.

**Steve:** Yeah.

**Leo:** That's great.

**Steve:** Okay. But the solution which surprised me and really captured my attention was both the smallest of all, by far, and the most powerful. It was implemented as a Windows PowerShell script.

**Leo:** Oh, interesting, yeah.

**Steve:** And having seen now what PowerShell's scripting language can do with its full access to the .NET language, it's clear to me that I'm going to need to make some time, I don't know, someday, to take a much closer look at it. It's sort of amazing what has quietly been happening over there. I've got a picture of this little PowerShell script which I guess you would call an "app," even though it's a script, you know, I mean, Perl is technically a scripting language, but you make apps with it. So we have a solution that I'll explain in detail in a moment. And it is this week's Security Now! podcast 905 shortcut. So everyone can get the PowerShell script by going to grc.sc/905. And when you hit ENTER, that will present you with a ZIP file. Well, anyway, I'll explain all that in a second.

So the way we got here is almost as interesting as what we got. And I confess that it would have never occurred to me. It began with a Twitter DM from a listener named Rob Woodruff. Rob tweeted to me. He said: "All right, Steve. You asked, and I delivered. I wrote a PowerShell script to parse the XML file that is your LastPass vault, identify any values encrypted with ECB rather than CBC, and decode the URLs from hex to ASCII. I chose PowerShell so that it will run on any modern Windows computer. It's not fancy, but it appears to work."

Now, he's not actually talking about this at the moment. "Not fancy, but it appears to work. You'll need to specify the InFile, OutFile, and Format parameters on the command line" - none of that's still true - "or it will prompt you for them. InFile is the path and filename of the XML file, your LastPass vault. OutFile is the path and filename of the output file." He says: "Format is the format of the output file, either CSV (Comma Separated Values) or HTML." Anyway, and then he sent me a link, "You can download it here," and he sent me a Dropbox link.

Okay. So that was Rob's first of what ended up being many messages between the two of us. And that's, as I said, far from where we ended up. About four hours later, this was on Friday evening, he followed up. He said: "This version of the script has a GUI." And then he sent me a Dropbox link. You know, so a Graphical User Interface. Upon seeing this, as I said, I thought, okay, I'm going to have to pay more attention to PowerShell. So I replied: "Holy crap, Rob. You're a PowerShell wizard. I'll check this out tomorrow. Thank you so much."

To which Rob replied: "My pleasure, Steve. I really enjoyed the process. First time doing a GUI in PowerShell. Not sure I can claim the title of Wizard, though. I had ChatGPT do most of the heavy lifting." And he said: "Speaking of which, if you haven't played around with it..."

**Leo:** Wow. Maybe ChatGPT is a PowerShell wizard.

**Steve:** He said: "Speaking of which, if you haven't played around with it," meaning ChatGPT, "you must. It will blow your mind." Okay, now, I mean, mind blown. I said, okay, being unsure whether Rob might be pulling my leg, I wrote back. I said: "Rob, are you not kidding? Did ChatGPT really have a hand in that?" I mean, this was like a functioning app with graphical user interface. And he said: "Steve, I'm not kidding.

ChatGPT is supposedly fluent in every written language, including programming languages. I told it in English what I wanted to do, and it spat out PowerShell code."

**Leo:** Well, more than that, it's apparently familiar with the LastPass XML format.

**Steve:** He said: "It's not perfect, of course, and I spent a lot of time debugging the hex to ASCII conversion. Ultimately, I ended up using a code snippet for the conversion that I found using Google because ChatGPT couldn't seem to figure it out. Similarly, when I asked ChatGPT to add a GUI, it got most of it right on the first try, but it had two of the buttons being overlapped by text fields." Well, boohoo, you know, fine, thank you.

**Leo:** Wait a minute. ChatGPT does GUIs, too? Wow.

**Steve:** Yes, yes.

**Leo:** PowerShell, no less.

**Steve:** Yes, yes. This thing is 12K, Leo. That thing that you saw is a 12K script. Okay. So he said: "So I had to adjust the positioning of the elements manually." He said: "Overall, though, it saved me a lot of time. I probably wouldn't have even tried to tackle this project without ChatGPT."

Okay. So we've clearly entered a very different world. What Rob explained is consistent with everything we've heard about ChatGPT. It's not yet perfect, but it's very good, and it can typically get you 95% of the way there. And then, yeah, you need to go in and fix the things it got wrong. But wow. So anyway, I thought that that, you know, that was just very cool.

I then worked with Rob through Friday evening and through the weekend to polish and perfect this little 12K gem to ready it for today's podcast. Remember that, being a script, it's inherently open source and therefore readily verifiable. And remember from last week that the key to obtaining the LastPass vault is using the developer features of any browser that's currently logged into LastPass. Though this part is still a bit inconvenient, there's no way around that without a huge amount of work to recreate everything that a browser does. And actually, as I was putting this together, putting the show notes together last night I thought, huh, maybe we'll just ask ChatGPT to write us a browser.

**Leo:** Or SpinRite. Maybe - no.

**Steve:** Actually, somebody did. I received a DM from someone who said: "How would you write SpinRite?" And it gave a complete description of how to do that.

**Leo:** Wow. Not assembly code, just the description.

**Steve:** No, it didn't do it. It didn't write any code. But it explained...

**Leo:** It understood what SpinRite does, which is interesting, yeah.

**Steve:** Yes, yes. I'm sure it went out there and, like, found out what SpinRite was and said, okay, here's how you'd write it.

**Leo:** We are in an interesting time, I have to say.

**Steve:** Oh, it is, it is really, yeah.

**Leo:** We're right on the cusp of something. Not sure how good it's going to be.

**Steve:** Okay. So, I know. Well, it's going to change the world; right?

**Leo:** Yeah.

**Steve:** I mean, kids are going to grow up in a different world than we did. And of course old fogies are all like, when I was a kid you had to actually use a pencil and sharpen it. But, you know.

**Leo:** Yeah, I say that all the time. I know how that is, yeah.

**Steve:** So it's necessary to still involve the developer console of the browser. Okay. So just to - I want to give people the instructions they need, especially those not familiar with running a PowerShell script. So first grab the PowerShell script that Rob wrote from my server. I'm hosting it, although he also has it on GitHub. And there's an update button on his GUI that'll take you over to his page on GitHub, in case it goes through some revisions. But as I said, grc.sc/905. That will return a tiny ZIP file containing one file titled "Analyze-LastPassVault.ps1." That's the PowerShell script.

And as we've learned on this podcast, encapsulating the script in a ZIP prevents Windows from tagging the script inside the ZIP with the dreaded Mark of the Web. So you don't need to see the warning about the dangers of using something that's been downloaded from the Internet. As a very useful security measure, Windows will no longer run unsigned PowerShell scripts. And before this all began, I didn't even know you could sign a PowerShell script. But turns out you can, and I did. So since I didn't want anyone to be put off by that, I mean, you know, it gives you all kinds of scary warnings and things, and you can say, yeah, run it anyway. And since signing also does a useful verification that nothing has been altered, I signed Rob's final script with GRC's EV code signing cert. So Windows will see that it's been signed, and it will run it all without complaint.

So after downloading it, launch a PowerShell prompt from the Windows menu, or you can just type "PowerShell" into the search box, and it'll get you there. This does not need to be run with elevated admin privileges, so anyone should be able to do this. Start the script by entering - and this is odd and reminiscent of Linux. You need to say .\Analyze-LastPassVault.ps1 in order to get it to go.

**Leo:** That's just saying execute it from this directory.

**Steve:** Exactly, from the current directory.

**Leo:** It won't do it otherwise. It's not in the path.

**Steve:** Don't go looking around for it.

**Leo:** Yeah, it's not in the path.

**Steve:** And so press ENTER. The Analyze LastPass Vault app will be displayed on your desktop. You'll find complete instructions there in the app itself, right there in the UI, for proceeding; so you can likely race off on your own without the rest of this. But I can provide a little bit of additional background for clarification. With Rob's app running, switch to the browser and log into LastPass so that you're looking at your vault, like, you know, many people did last week when we first talked about the way to grab your vault. Press the F12 key or CTRL+SHIFT+I, which toggles the developer mode. And so that will suddenly subdivide your screen vertically into the browser on the left and the developer stuff over on the right. Then select the Console tab in the developer window. And things may be scrolling like crazy. Wait for them to settle down. That'll eventually stop. Once you have a cursor flashing there at the bottom, you're ready.

To make this as easy as possible, we built in the three lines of JavaScript code that needs to be dropped into the browser at this point. So you just press Copy Query button in Rob's app, which places those three lines onto the system's clipboard. So then bring the browser to the foreground so that it has the system focus, and hit CTRL+V to paste those three lines into the browser. And you'll see them appear there after the cursor. Then press ENTER to execute the query, and the page will fill with XML expressions. That's your encrypted LastPass vault. As I described last week, down at the very bottom of the screen will be the options to display more of the text that was cut off because it's way longer than will fit on the screen, also to copy the entire query results to the system's clipboard. That's what you want. So click on Copy.

Now, switch back to Rob's app and click the Paste button. That will paste the captured clipboard directly into Rob's app without needing to go through the intermediate step of saving it to a file. And note that if last week you had already copied your XML-format vault out of the browser and may have terminated your account with LastPass, you can also provide Rob's app with that filename to load and then process.

Okay. So with the vault made available to the app, either by pasting it or opening a file, specify an output filename to receive the deobfuscated data and choose the output format, either CSV or HTML. The CSV format will ideally require something in your system for viewing it, like Excel. I like the HTML format since your browser, and we all have a browser, will happily display that. So set the filename, ending in either CSV or HTML as needed, and finish by clicking "Analyze." The file will be written, and then the resulting file will be launched for display by whatever handler that your system has registered for handling that type of file. And as I said, if you exported HTML, the results will be displayed in a nicely formatted scrollable window with one line per login record. Actually, same thing for CSV.

So Rob's app displays an "OK" on the far left if the encrypted account information was encrypted using the better CBC cipher mode. And we'll review that again a little bit in a second. If the record's encrypted information was encrypted with the suboptimal information-leaking ECB cipher mode, you'll see the message "WARNING: Encrypted with

ECB." Next on the line, and likely most interesting to everyone, will be the deobfuscated URL that's associated with the website's encrypted logon record. Taken as a whole, these are all the sites for which LastPass's vault contained your logon information.

And it's notable that the vault does not appear to contain the user's unencrypted email address to associate who they are with the vault, nor is there any indication of the number there of PBKDF2 iterations that were being used to obtain the vault's decryption key from the user's email address and password. That information is necessary to run the Password-Based Key Derivation Function to decrypt the key which is used to decrypt the vault. Otherwise, it requires brute force. So it must be that it's provided to the user's client through another query, other than the one we just used in order to get the vault's bulk content.

But LastPass's infamous December 22nd breach update, in that they said: "The threat actor copied information from backup that contained basic customer account information and related metadata including company names, end-user names, billing addresses, email addresses, telephone numbers, and the IP addresses from which customers were accessing the LastPass service."

So to that we can now add everything that Rob's LastPass vault analyzer makes crystal clear. Given the amount of effort that will generally be required - and we have a bracing update on that coming up next - to brute force the decryption of the encrypted information in even one LastPass vault, coupled with the fact that tens of millions of LastPass user vaults were obtained en masse, the direct threat of decryption for a single individual, unless directly targeted, would seem to be very small.

**Leo:** But because they have the metadata, it's easier for them to target somebody; don't you think?

**Steve:** Correct. Well, yes.

**Leo:** Like if you had the login to the Strategic Air Command's Missile Control Center in your LastPass vault...

**Steve:** You'd be highly motivated.

**Leo:** People would want to get into it; right?

**Steve:** Yeah.

**Leo:** They're going to have to pick and choose, obviously. They're not going to decrypt them all. Why would they?

**Steve:** Correct.

**Leo:** Yeah.

**Steve:** But assuming that it's possible for the bad guys to associate company names, end-user names, physical addresses, email addresses...

**Leo:** There you go.

**Steve:** ...telephone numbers and IP addresses with specific vaults, and that must be possible because LastPass has to do it, and this is the information that they lost.

**Leo:** Yeah, it's the metadata that they didn't encrypt.

**Steve:** Yes. Even without any decryption, what you get is a comprehensive dump of exactly who logs on exactly where.

**Leo:** Right.

**Steve:** And if you scroll horizontally all the way to the right of Rob's HTML output, there's also a Last Touched field.

**Leo:** Oh, geez.

**Steve:** Containing a timecode that shows when the last logon at that domain occurred.

**Leo:** Oh, my god.

**Steve:** So even without the use of any brute force decryption of the vault's encrypted contents, this represents at best a significant privacy compromise for every LastPass user. Every user.

**Leo:** And LastPass could have - there was no technical reason they didn't encrypt that data. They could have. Yes? Other password managers do.

**Steve:** Yes. Well, so I did a test. I was wondering why they didn't encrypt the URLs. So I thought, well, maybe if I'm not logged in, so that my vault is encrypted, right, if I'm not logged into LastPass, so I have no vault present, maybe LastPass doesn't add the little LastPass highlights to the username and password field if it knows it doesn't have the ability to fill that in.

**Leo:** Who cares?

**Steve:** So I thought, okay. And it turns out...

**Leo:** By the way, Bitwarden does not do that. A lot of password managers don't. I think that's a silly little JavaScript feature. Go ahead. Turns out what?

**Steve:** So it turns out that's not the case.

**Leo:** Oh.

**Steve:** Which is to say - no, no. Which is to say LastPass always puts that...

**Leo:** It's always there.

**Steve:** ...in all of the username and password fields. So I was trying to give them the benefit of the doubt.

**Leo:** Yeah, no, you tried.

**Steve:** Saying, well, maybe they're being smart about where the user, where it knows the user has the ability to log in. No. So Leo, I can find no rationale...

**Leo:** And we're not saying that they should encrypt it with AES-256 in your vault. But they should probably hash it so that, I mean, it shouldn't be plaintext in your metadata in the vault. A lot of this stuff.

**Steve:** I see no - I don't get...

**Leo:** There's no benefit to doing it.

**Steve:** I don't yet see a reason for them doing it. And I tried one idea, but that wasn't it.

**Leo:** Yeah. And you could see why - so there's already a class-action lawsuit. And the guy who started it claims, and we haven't verified this, but claims that he lost $53,000 in cryptocurrency. You could see why this is important. If you're going to attack somebody, find somebody maybe who has a shared crypto wallet on Coinbase. That kind of information is being leaked. It gives you a way to prioritize which vaults you go after.

**Steve:** Oh, wait, honey. There is way more [crosstalk].

**Leo:** Oh, no. I mean, it's perfectly possible, we don't know, that there is somebody right now running his old bitcoin mining apparatus, because he doesn't have any use for that anymore, against all of the vaults, and just the first ones that come out are the ones he's going to use, right, because they have bad passwords or whatever. Or their PBKDF2 is 500.

**Steve:** One last note before we look at what's been learned from a week of feedback from our listeners, which as I said is a little bit bracing.

**Leo:** Oh, boy.

**Steve:** If executing Rob's PowerShell script produces errors rather than the presentation of a nice graphical interface, as it initially did for me on my Windows 7 machine, I wanted to note that it's possible to update any Windows PowerShell support, even Windows 7, to the latest version 5.1. Microsoft wants you to have it. So I have a link for that at the bottom of page 4 of the show notes that will help anyone update their PowerShell script, just in case you get an error.

Okay. So next. What more do we know this week that we didn't know last week after our listeners had the chance to understand and peruse their LastPass settings? We don't yet have any good sense for whether, or the degree to which, encrypted content was allowed to remain encrypted under the less desirable ECB cipher mode. I expect feedback from the use of Rob's analyzer to fill in a lot of that information for us next week. So I'd appreciate knowing when and how many ECB warnings, if any, are seen by our listeners. I didn't have any ECB encryption in my vault, even though I'm sure that I had login credentials dating from my first use of LastPass. Rob reported that his vault contained one ECB-encrypted entry.

**Leo:** This would be a really old password; right? I mean...

**Steve:** Yeah, well, I had really old ones, too. So it's just...

**Leo:** We don't know what the...

**Steve:** We don't, yeah.

**Leo:** ...timeframe was of [crosstalk].

**Steve:** We're having to reverse engineer some of this.

**Leo:** Again, this is something we need to know, and LastPass has not been forthcoming with it, yeah.

**Steve:** Yup, yup. Okay. But the ECB versus CBC issue is not that much of a big deal. I don't want to overstate that. Since AES-ECB, Electronic Code Book, being based upon the AES cipher, has a cipher block size of 128 bits, sets of 16 eight-bit characters, thus forming a block of 128 bits, are encrypted from that pattern of 128 bits into a different pattern of 128 bits. That's encryption. So this means that every instance of the same password will encrypt into the same pattern of 128 bits. As I noted last week, the presence of password reuse would therefore be obvious just by inspecting a user's encrypted vault without the need for any decryption.

By comparison, AES-CBC, Cipher Block Chaining, uses both the encryption key, which would not change from one password to the next, and also a random initialization vector, you know, think of it like salting a hash. It is different for every password. So this would completely obscure the presence of any identical passwords. Otherwise I really can't see any reason for preferring one over the other, but that's a good reason.

**Leo:** That's a big one, yeah, because now the guy, again, we're talking about triaging the vaults.

**Steve:** Yup.

**Leo:** You could look at a vault, say look at that 400 reused passwords, without decrypting. And that's a big deal.

**Steve:** So we're going to likely learn by this time next week, we'll have feedback from our listeners who have used Rob's tool. We'll know how much ECB is still around.

**Leo:** Good, good.

**Steve:** Okay, now, Leo, are you centered over your ball?

**Leo:** Oh, no. Now what? Oh, no. Okay, yes.

**Steve:** By far the most worrisome fact that was revealed when our listeners checked the settings of their LastPass vaults was the degree to which many - and I do mean many - of their password iteration settings were found to be below the 100,100 iterations mark. And in a revelation that I'm still trying to get my head around, I heard from many listeners whose PBKDF2 iteration count was set to 1.

**Leo:** What?

**Steve:** Yes, 1. And thus the title of today's podcast.

**Leo:** 1.

**Steve:** 1. Many people have an iteration of 1, which is to say, why bother?

**Leo:** What does that mean? Does it mean you could, I mean, you still have to - a hash cannot be reversed, but you can use rainbow tables easily.

**Steve:** No, because it is salted.

**Leo:** Okay.

**Steve:** I've got all the math here. We're going to understand what this means.

**Leo:** Oh, good. Okay, good.

**Steve:** So I also received many reports of iterations still being set to 500, and many set to 5,000.

**Leo:** 5,000 was the default for many years.

**Steve:** 5,000, it was up until five years ago it was 5,000. That's when it changed to the 100,100.

**Leo:** And as we mentioned on Ask the Tech Guys on Sunday, the OWASP recommendation is 300,000.

**Steve:** Yes. And probably not adequate, and we'll be talking about that in a second.

**Leo:** Maybe even should be higher.

**Steve:** Yeah. I'm thinking 1234567 would be a good number. That's 1,234,567.

**Leo:** And of course the reason you don't use larger ones is because it's slower on especially a mobile device.

**Steve:** Yeah. Turns out unless you're being powered by a hamster, you're probably okay.

**Leo:** You can do millions. Okay, that's good to know.

**Steve:** Okay. So at first you might think that anyone whose iteration setting was 1 should be about 100,000 times more concerned than someone using the new default of 100,100. But it's actually quite a bit worse than that because that 100,000 to 1 simple math, it assumes that vaults were being selected at random for attack, which would probably not be true.

**Leo:** Right.

**Steve:** We need to assume that the attackers obtained every user's account metadata, including their vault's iteration counts.

**Leo:** Oh, that's in the metadata. It is. Obviously. That's how people are seeing this. Yeah, yeah.

**Steve:** Yes. It has to be because LastPass has to have that.

**Leo:** They need to know, okay.

**Steve:** Those counts need to be recorded somewhere because no one's vault could be decrypted without knowledge of the count. And the count is not particularly sensitive information.

**Leo:** Right.

**Steve:** Unless it's 1. And LastPass would have backed...

**Leo:** Oh, my god. Oh, my god.

**Steve:** Oh, Leo, it's so bad. LastPass would have backed it up since the loss of that, the iteration count data, would have been even worse than the loss of the vault backups themselves because it would have made them worthless. So assuming that the attackers obtained the iteration counts for every LastPass user, as they probably did from LastPass's backup, if opportunistic brute force decryption of user accounts was their intent, it would be a reasonable strategy for the attackers to start with those LastPass users whose counts were 1. Why would they not?

**Leo:** Actually, this would be a good use for a PowerShell script to triage the millions of vaults you have, looking for low iteration counts and perhaps maybe some custodial bitcoin wallets or, you know, things [crosstalk].

**Steve:** Leo, just ask ChatGPT.

**Leo:** He'll do it. Or she'll do it. Or it'll do it, yeah. Which one should I crack first, Chat?

**Steve:** Yeah. Unfortunately, there's a well-known expression to describe the situation in which all of those LastPass users who at the time of this breach had their LastPass password iteration counts set to 1. And that expression is "low-hanging fruit."

**Leo:** Do we have a theory how somebody could have 1? It was never the default.

**Steve:** Yes, it was.

**Leo:** What?

**Steve:** Yes.

**Leo:** In the earliest days of LastPass, 1 was the default.

**Steve:** For the first four years, from 2008 to 2012.

**Leo:** Oh, I didn't realize.

**Steve:** Because a salted hash was considered strong enough.

**Leo:** That's enough. And of course one iteration.

**Steve:** That's what it was when I looked at it, and I said, this is fine.

**Leo:** Yeah.

**Steve:** You know, you need to salt. You can't just hash. You need to salt your hash.

**Leo:** Must salt it, yeah, we've said that many times.

**Steve:** And so there was no problem. Okay. So last week we did not look deeply into the actual performance of today's GPU-enhanced password cracking. This week, we need to get a sense of scale. Current estimates of GPU hardware-enhanced password cracking places the time required to crack a 100,100 iteration PBKDF2 protected password, where that password has high entropy of 50 bits, at 200 years. So one GPU, 100,100 iterations, high entropy password, 200 years. But since GPU use scales linearly, dividing that cracking task among 200 GPUs...

**Leo:** Yeah, one year.

**Steve:** Which is now quite mature cracking technology, could crack the same password having 50 bits of entropy in one year. Okay. So one GPU, 200 years. 200 GPUs, one year. But I also note that studies have shown - Wikipedia says it too, they agree - that most practical passwords have an entropy of around 40 bits.

**Leo:** A lower entropy because you have to remember it. They're not truly random.

**Steve:** Yes. It turns out that it's difficult to actually get 50. So, okay. We'll get back to what lowers true entropy in a second. Okay. But having 40 bits of entropy is approximately - 40 versus 50, 40 is approximately 1,000 times weaker than 50 bits.

**Leo:** Oh, that's a problem.

**Steve:** Because bit strength scales exponentially.

**Leo:** Oh, boy.

**Steve:** In other words, random bits are worth a lot because each additional truly random bit on average doubles the time required to crack. So the difference between 40 bits and 50 bits is 10 bits. 2^10 is 1024, thus a thousand times weaker. Okay. So if we assume that our attackers will have the use of 200 GPUs which, in this era of GPU-laden cryptocurrency mining rigs seems entirely reasonable, cracking a typical password having 40 bits of actual entropy would require 71.338 days, if that password was protected by 100,100 iterations of PBKDF2.

Okay. Just to restate that because this is an important benchmark, a typical strength password, 100,100 iterations of key derivation, that's attacked by a 200-GPU password-cracking rig would fall against that attack in an average of 71.338 days. So if you're thinking that all of those 100,100 iterations might still not be providing you with sufficient protection, you're probably not entirely wrong. Using one million iterations would be 10 times stronger, bringing us to 713 days, just shy of two years. That seems much safer.

Okay. But the very bad news is that for all those whose LastPass iteration count was, for whatever reason, discovered to still be set to 1 - and there were many such people who reported that this past week.

**Leo:** And those are our listeners.

**Steve:** Yes.

**Leo:** Those are not unsophisticated people. Those are our listeners.

**Steve:** Yes, yes. Those same 200 GPUs could crack that same 40-bit entropy password in an average of 61.56 seconds.

**Leo:** A minute.

**Steve:** Or just over one minute per single-iteration password crack.

**Leo:** Wow.

**Steve:** Given that, it appears to be the height of negligence, if not bordering on criminality, that for some reason, for whatever reason, many listeners of this podcast, and I'm sure a great many more non-listeners, have no effective protection.

**Leo:** Sad. This is the most loyal LastPass users, the ones who started using it in the very earliest days. Which, by the way, is probably you and me.

**Steve:** Yeah, yeah. Fortunately, you and I moved to 100,000.

**Leo:** We did.

**Steve:** Back five years ago when we talked about this.

**Leo:** Because we listen, yup, yup.

**Steve:** They have no effective protection from the cracking of their LastPass vaults and the resulting disclosure of every single one of their website logon credentials, their credit cards and, as you mentioned before, any other confidential documents and personal papers that were stored for them by LastPass.

Okay. Now, not one of these many people told me that LastPass had reached out to them to explain that due to their effectively non-existent password encryption they are at heightened risk following the data breach, and that they should therefore immediately rotate all their logon credentials being managed by LastPass and assume that any information stored in their LastPass vault had been compromised. As far as I know, that has not happened. And it certainly should have. LastPass knows everyone's iteration counts. And now, so do the criminals who stole them.

LastPass somehow failed to update those iteration counts for a decade after the default was raised from 1 to 500 in June of 2012. And they have not immediately and proactively assumed responsibility for that by informing their users, whose iteration counts were dangerously low, in many cases set to 1, that unfortunately they should now assume that the encrypted content of their LastPass vaults is now in the hands of criminals. The industry at large has been grumbling about LastPass not being forthcoming about the details of the breach. But we now have all the information we need to assess LastPass's culpability.

A couple of weeks ago, when I was first updating myself on LastPass's client settings over the holidays, I changed my iteration count from 100,100 where, thank goodness it was still set after we talked about this five years ago, to 350,000, as is now recommended by OWASP. That change of my iteration count took, oh, perhaps five seconds. I had to provide my LastPass master password again, so that the LastPass client could rehash that password under the new iteration count. And that was it. This could and should have been automated since its first increase from 1 to 500 back in 2008. No user should have been allowed to set a dangerously low iteration count; and every LastPass client, which must know its user's iteration count in order to function, should have taken proactive responsibility for continually bumping it up every five years or so to whatever is currently considered safe.

So, the most startling and deeply disturbing news I received throughout last week was not only that many of our listeners' iteration counts were still 5000, and many even 500, but that many discovered that theirs was still set to 1. I am a 67-year-old lifelong entrepreneur and businessman. And you would have a difficult time finding anyone who is more deeply opposed to frivolously turning attorneys loose on each other. It's one thing to fight over an ongoing contract dispute in order to reach a resolution. That makes sense to me when the parties cannot negotiate an accord in the absence of objective judgment. But attacking an entity after the fact, for something that was done which I'm sure they now regret, still leaves a bad taste in my mouth. On the other hand, if a lawsuit were to be brought against LastPass, not because they made a mistake that upset their customers, but over actual provable damages arising from reliance upon LastPass' assertion of the safety of vault data, then I would not consider that to be unwarranted ambulance chasing.

There's no way to paint the presence of an iteration count of 1, used for the derivation of a LastPass vault's decryption key, as anything other than a critically debilitating product defect. And for that, if actual damage results, LastPass could be, and I think should be, held wholly responsible.

**Leo:** So I should point out that all the major password managers, including our sponsor, Bitwarden, 1Password use PBKDF2. So I just went into my Bitwarden and set my PBKDF2 - it's in the Security Keys section of your settings - I set it to two million.

**Steve:** Good.

**Leo:** And of course what I'm going to do is see if anything got really, really slow. I bet it didn't. Frankly, the processor in my iPhone is better than the processor on this Lenovo. But assuming that everything's usably slow, but not too slow, I'm going to keep it at a high number. And so that's a warning that others should also do this. Right? It's not a LastPass-only problem. The Bitwarden default is 100,000. But set it higher; right?

**Steve:** Well, after we take a break, we're going to talk about the true strength of our passwords.

**Leo:** Okay. Especially since I probably - my master password, in all likelihood, might not be 40 bits of entropy. It might be a little bit less, as probably it is for a lot of people because you have to memorize it; right?

**Steve:** Yup.

**Leo:** So I use, you know, a passphrase and some numbers and stuff. But that's certainly not fully random, that's for sure.

**Steve:** That's our next topic.

**Leo:** Oh, good. You are so good. You anticipate everything. Get the show notes, GRC.com, because there's a lot of information in there. You should always get them anyway, GRC.com, when you get the show. I'm wondering if we should push it out in the RSS feed, too. We can do that, push a PDF of the show notes along with the show. Would you have any objection to that?

**Steve:** No, no.

**Leo:** I'm thinking maybe we should start doing that because they're so valuable. I don't know. Listeners, let us know. I mean, a lot of people would get it and go, what did I get, I don't want it. But I think it would be valuable.

**Steve:** So I want to amplify something I touched on both last week and this week. I mentioned that an increase in iteration count provided a linear increase in strength, whereas the increase in strength provided by adding bits is exponential. I want to be certain that everyone fully appreciates the implications of that.

A few weeks ago, when I increased my LastPass client's iteration count from 100,100 to 350,000, that gave me an increase of 3.497, so about three and a half times. But not four times. But if I had increased my password's entropy by just 2 bits, that would have been a full factor of four increase in cracking resistance. True entropy is quite difficult to calculate because very few of us are using a chunk of text from GRC's Perfect Passwords page for our master password since those are impossible to remember.

So we have the situation that a single character's true entropy is difficult to calculate. If any character in a password is related to any other character in that password in any meaningful way other than having been chosen purely at random, in other words, if there's any reason for a character to be what it is rather than something else, then that character's...

**Leo:** So if I use a passphrase, for instance.

**Steve:** Yes.

**Leo:** Because then those characters are logically related by English grammar to one another.

**Steve:** Yes. Yes. Then that character's contribution to the true entropy of the whole is reduced. Its contribution of entropy would be significantly less than it would otherwise be.

**Leo:** Because it's not random.

**Steve:** Right, it's not.

**Leo:** I before E except after C or whatever it is, yeah.

**Steve:** Yup. So this is why the first thing that password-guessing crackers do is use dictionary words in various ways and base their attacks upon the frequency of characters occurring in the natural languages of the password's user.

**Leo:** ETAOIN SHRDLU.

**Steve:** Yup. Those attacks model the lack of entropy that many users employ when they're choosing their passwords.

**Leo:** Yeah. I've got to go check my password. I'll be back.

**Steve:** So what's the idealized potential entropy of a single character? In a byte-oriented system, a single character typically occupies 8 bits. So we might be inclined to say "8 bits." But ASCII only uses the lower seven of those 8 bits. So assuming a non-UNICODE standard ASCII character set, there are a total of 95 printable standard characters available if you use upper and lowercase alphabetic, the 10 numeric digits, and all the other special characters. That gets you to 95.

So here's the point I want to drive home: Increasing my iteration count from 100,100 to 350,000 yielded that just shy of three and a half times increase in password-busting protection, 3.497. But just adding one single randomly chosen additional character to the end of a password increases the resulting password's anti-cracking strength by 95 times. 95.

**Leo:** This is the Password Haystacks stuff you told us about years ago.

**Steve:** Yes, exactly. And so this is why, when it comes to passwords, size does matter. You get far more attack protection by using even slightly longer passwords, where strength increases exponentially with length, than you do by increasing iteration counts, where strength only increases linearly.

Okay. So there were a bunch of interesting bits of feedback and questions from our listeners. I'm going to continue talking about some of these things using them as the prompting. Via a DM I received the note. Someone posted: "I have a corporate LastPass account and a personal pro account. The personal account was updated to 100K iterations, but the corporate account was still at just 5K. My personal account is still exposed, though, because I took advantage of the ability to share passwords between my personal and corporate accounts to reduce the number of logins. I assume that if they crack the corporate, they would have the personal anyway. Good news, my password has more than 25 random characters derived from your Perfect Passwords. The bad news is that it is so long and random that I used the same password for my corporate and personal LastPass accounts.

**Leo:** Oh, that's not good.

**Steve:** Well, 25 truly random characters, chosen from the Perfect Passwords page, as this user did, will have been selected from an alphabet of 95 possible characters. So that's 95 x 95 x 95 and so on for a total of 25 times. That's 95 raised to the 25th power. I used the Password Haystacks page to quickly do the math and show me that the resulting password has 2.8 x 10^49.

**Leo:** Holy cow.

**Steve:** Yeah, baby.

**Leo:** But how do you memorize it?

**Steve:** You don't. You have that somewhere else.

**Leo:** You write that somewhere, okay.

**Steve:** Yes. You keep that in your Apple Notepad and copy and paste it.

**Leo:** Oh, lord.

**Steve:** And importantly, all of those characters are all equally likely to appear combinations. True entropy. If we take the Log base 2 of that number to determine the equivalent binary bit strength, we get 164.2. So in other words, it contains a little over 164 binary bits of true entropy. Another way to look at that is that each character, when truly chosen randomly from a set of 95 possible characters, contributes 6.57 bits of entropy. 6.57. In other words, this person has absolutely nothing to worry about. His password has slightly more than 164 bits of true entropy. It will never in many lifetimes be cracked by today's or even any projected technology of tomorrow. Remember, quantum computers won't help with this sort of symmetric crypto problem. They are of no use.

Okay. But there's something else worth noting. Recall from last week that Mr. Grumpy Pants - what was his name? Oh, yeah, Jeremi Gosney. He noted that LastPass's vault encryption key was derived from only 128 bits of entropy.

Okay. So now consider this crazy 25-character totally random password which has a bit more than 164 bits of entropy. If the attackers knew that - and there's no way they could. But if they did, it would be far quicker to just forget about the user's insane password and attempt to directly brute force the vault's encryption key itself since it has "only" has 128 bits of entropy. I have "only" in air quotes because my point is that 128 bits already has so many possible combinations (3.4 x 10^38) that there's never any reason to go above that. Haystacks tells us that 20 characters chosen from that 95-character alphabet offers 3.62 x 10^39. Okay. Once again, 128 bits is 3.4 x 10^38. Twenty random characters is 3.6 x 10^39. So 10 times stronger than what 128 bits can do.

Okay. So last word on this, in summary, do not use only 20 characters unless they are truly chosen from among all possible characters randomly. But if they are, there is no need or benefit gained from using any more. Twenty purely random characters from an alphabet of 95 is 10 times more than what you get from 128 bits. And 128 bits is considered by the entire industry all we need for now.

Dave wrote: "On Security Now! Episode 904, Steve asked for feedback on the current value of the LastPass Password Iterations field. Mine was set to 1. I have no idea how/why it is 1 because I never changed it." Well, there's why. "Needless to say, I have downloaded and installed Bitwarden, and I am changing the password on every site in my vault as rapidly as I can." So, yes, Dave has the right idea. He was typical of many of our listeners. And there's an example from among many of what our listeners discovered to their horror last week; and, sadly, it might be because he never changed it that it remained set to 1.

As you said, Leo, the most loyal early adopters of LastPass, they're the ones who are, in a phrase, effed. As we know, he should not have had to change it. That should never have been his responsibility. But we're on the outside here, looking in. We have no idea of the real story behind this iteration fiasco. But there is no way to forgive this from LastPass. None. This is more than a mistake. This had to be someone's boneheaded decision.

With their acknowledgement of the importance of increasing the iteration count over time, evidenced by its default being jumped from 1 to 500 to 5000 to 100,100, someone must have made the decision not to bother bringing older existing iteration counts into compliance with current best practices. Someone must have decided that it would, I don't know, result in too much customer confusion and support calls, so let's just leave it wherever it is.

And the galling thing is it could have been done 100% transparently. I am no smarter than their crypto people. So they know this, too. When the user provides their email address and password to log into their client, at that moment the client has everything it needs to perform the upgrade transparently. Start iterating on PBKDF2. Pause at the current iteration count and take a snapshot of the current key at that point. Then keep going to the new larger iteration count and take a snapshot of that new key. Now decrypt the vault with the current key, which was sampled midstream, then reencrypt the vault with the larger final iteration count key. And, finally, update the stored iteration count. Done. Totally transparent. No user confusion. And a company as big as LastPass, now focused on the enterprise and everything, for reasons I can't possibly explain, never did that. I mean, not only is not everybody at 100,100, there are people at 5000 and 500. There are people at 1. And change your passwords.

Okay. David Lemire. He said: "Hi, Steve. Thanks for the excellent coverage of the LastPass breach and its consequences in SN-904. I can confirm both your smooth experience transferring from LastPass to Bitwarden, and Leo's note about Bitwarden having a lower size limit on secure notes than LastPass's. I had to delete one or two very large notes before I could successfully import my vault." He said: "Thankfully, they were obsolete."

He said: "I have one technical security question. Given the threat of rainbow tables, wouldn't it make sense for each individual account to have its own iteration value within a suitably secure range, rather than a common default value," he says, "which I realize can be changed. Combining an unpredictable iteration count with salting the hashing process should raise the work factor for the creation of rainbow tables, as well as the comparison process, by a considerable factor."

Okay. Now, I didn't mean to confuse things last week with my mention of the possibility of attacking known salt-free hashing schemes with precomputation attacks. My intention was to paint a history to remind us of where we've been and how we got to where we are today. Everyone has always been protected from precomputation attacks by the inclusion of their email address as the salt for the PBKDF2 function. Joe Siegrist was doing this from day one, with an iteration count of 1. Unfortunately, back in 2008, Joe was, as I said, also iterating only once through PBKDF2. And as we now know, for some unlucky souls, that for whatever reason was never changed.

Someone is also likely to ask if a user deliberately set their iteration count to 1, what would happen if they didn't understand what that was about? You know, like what if that happened? My answer to that would be that it should absolutely never have been allowed. LastPass would certainly not allow any user to leave their password blank. A low iteration count is effectively no different. LastPass was lifting the count over time, and that should have always been the minimum that any LastPass user client would accept as its count.

I received a question via email: "Hello. About the LastPass breach, Episode 904, the risk on passwords and metadata was explained very well. I wished the risk for files stored in LastPass could be explained, too, for example, copies of personal ID, passports, driver's licenses." He said: "I can go through all my passwords and change them, but changing my real life documents will be much more difficult. I guess many other LastPass users will have this problem, too. Thanks." Signed Boris. So Boris makes a great point. We didn't stop to consider much that we were only talking about login credentials primarily, you know, but the many greater privacy dangers that might arise from having the contents of the LastPass vault's secure notes storage compromised. Depending upon what was in there, the damage from disclosure could be significant.

**Leo:** Oh, yeah. My social's in there, my driver's license, my passport, everything.

**Steve:** Yup. Full identity theft information.

**Leo:** Oh, good lord, yes.

**Steve:** So I also received a bunch of these, basically saying: "Replying to @SGgrc, I exported all my stuff to Bitwarden earlier tonight. The process couldn't have gone more smoothly. Up and running on both my laptop and phone."

**Leo:** Yay.

**Steve:** So I was glad for that.

**Leo:** Bitwarden, I'll say it again, is a sponsor, but that has nothing to do with anything.

**Steve:** I explained my rationale for choosing it last week.

**Leo:** We like it because it's open source. There's all sorts of benefits.

**Steve:** Yup.

**Leo:** Incidentally, so I have changed my PBKDF2 to two million, as I said, which I think is as large as it can go. I guess there's no practical limit. But I notice now, because I also changed my password, so I got logged out everywhere, that it only added a few seconds to loading the vault.

**Steve:** Yup.

**Leo:** So a minor, minor amount of time. That's two million. That's big. I also, one of the nice things about LastPass, I changed my password because I want more entropy. And I used Password Haystacks to pad it out and all that stuff. But they also give you the option, risky though it might be, and explain the risk, of rotating your vault key. That's that 128-bit key for your vault. And you can do that, as well. And I thought, you know, I've had this for a couple years. Maybe I should rotate that vault key, too. So they really give you the options you need, I think, so make sure you're safe, even if that vault got exfiltrated.

**Steve:** Yup. And I will have a request for Bitwarden...

**Leo:** Oh, good.

**Steve:** ...by the end of the show.

**Leo:** I know some people.

**Steve:** Okay. So via direct message: "Steve, I listened to your podcast twice, but what I don't understand is I thought you said it would not matter if somebody had our blob of data because the keys only reside on our devices. So even if they had our master password, how would they crack into the blob without having the keys? Thanks for all you do."

Okay. So if there was some confusion there, let me clear that up. The key that's required to decrypt the LastPass vault key is derived only and completely from three pieces of information: the user's email address, the user's password, and the iteration count. No other information is required. The only one of these three things that LastPass and the attackers do not know is the user's password. They have their email address and iteration count. So with an iteration count that's too low, it's quite feasible for a modern attacker to simply guess and test at ultra-high speed all possible passwords until they find the right one.

Also via direct message: "Hi, Steve. Do you think that having a non-standard number of iterations, let's say 168,429 makes that particular password not worth the effort to try to decipher since 95% of all passwords will have either 5000 or 100,100 iterations?" And this question came up often.

Since each user's iteration count is known, making it non-standard will have no effect. If the attackers have adopted that "low-hanging fruit first" strategy, which is what seems by far the most likely way to reap the rewards of their score, they would sort the entire LastPass vault backup database by iteration count, and prioritize attacks against all of those unlucky souls whose iteration count matches the title of this podcast. From there, sorted by iteration count, they would proceed upward. And let's not forget that a significant amount of privacy-related information is immediately available since all of the URLs for the sites where we have stored our logins is in the clear.

Someone named Zapper tweeted to me: "@SGgrc Steve. Being a longtime and very trusting listener I have been a LastPass user on your recommendation. I will now migrate to Bitwarden. May I ask you to clarify the security risk if my LastPass master password was 20-plus characters?"

Okay. Zapper's question is also quite common. So I want to reiterate that longer is always better, even much better, and more random is also better because it increases true entropy. But as for 20-plus characters, if your password is truly 20 random characters, that's 131 bits of true entropy which is absolutely secure. So no need to go larger.

Via DM: "LastPass," he says, "I changed my 30-character master password, but I still feel uneasy. I started changing all passwords, but have not migrated off of LastPass yet. Any thoughts on ensuring LastPass removes all vault info upon cancellation of user account?"

So that's a really good question. I think LastPass needs to affirmatively answer this question, if they don't have it already somewhere in their FAQ. This user is not talking about the consequences of the theft immediately, but rather the safety of remaining for a while, I guess, with LastPass. A 30-character master password will be very, very, very secure. Even if it's the lowercase alphabet, in order, abcdefghijklmnop and so forth, with the digits 1234 scattered among the 26 letters somewhere to pad it out to 30. The resulting hash from that conveys nothing of the password's length. So no one attacking would have any idea how long the password is. This was the key message underlying Password Haystacks. No attacker would be trying any 30-character passwords, having no idea how long the password is, until they had exhausted all shorter passwords, and that will never happen.

So, you know, the reason we don't recommend somebody using that 30-character password in a lot of different places is that we don't know that everyone is hashing it securely and storing it securely. And we've just had a big example of the largest password manager on the planet not storing things securely. So, but using that one time in a situation where you know how the password is being managed, it's being deeply hashed with a high iteration count, that would be a perfectly acceptable password. But don't use that one.

Skynet said: "Moved my vault to Bitwarden and set the PBKDF2 iteration to one million. On my iPhone 11 Pro Max it performs fine. One million iterations. Go big or go home, Steve."

**Leo:** I went double that.

**Steve:** Yup.

**Leo:** And it's fine. It added a few seconds. That's all it adds. And it only adds it the first...

**Steve:** And you don't have to do it that often.

**Leo:** It only adds it the first time you download the password vault.

**Steve:** Correct.

**Leo:** Yeah. So it's not - it's a minor - boy, and it feels a lot better. And I got rid of the entropy. Instead of using a passphrase, which had English words in it, so the order of those letters was not completely random and following spelling rules, I used an acronym. I used, not an acronym, an initialism. So I used the first letters of a long phrase and added some extra padding with other stuff.

**Steve:** Yup. Good.

**Leo:** So I feel like that's - now, there's still less randomness because there's, you know, there's some grammar to that sentence. But I don't...

**Steve:** But again, length matters.

**Leo:** It's very long now. It's 59 characters. I think it's long enough.

**Steve:** Nobody will know what your length is. Nobody will know.

**Leo:** Oh, I just said it. Never mind. Forget I said that. By the way, that's one thing I do when I create passwords for - we should mention passwords for sites don't have the same issues in most cases. The difference here is somebody was able to download the vault and at their leisure brute force it. You don't have that leisure with a site unless the site gets breached, and their password database gets captured, and it's not properly encrypted and all that. You don't - the same rules don't necessarily apply to individual passwords you're generating. Although, again, you know, it's worth doing a long one if you're using a password manager.

**Steve:** Well, and this is a perfect example of all your chickens in one basket; right?

**Leo:** Yeah.

**Steve:** I mean, it's the master password.

**Leo:** That's the biggie.

**Steve:** And LastPass said the only password you need to remember.

**Leo:** Well...

**Steve:** Well, uh-huh.

**Leo:** But what I was going to say is I vary up the length of passwords I use on sites. So I don't always use 19 characters. That's a setting in Bitwarden, but I mix it up. So they don't even know that, which is helpful; right? Yeah.

**Steve:** Yeah. So for PBKDF2, I don't see any reason for using an iteration count lower than a million. And as I said, I'd probably use 1,234,567. And it doesn't matter that it's not secret because that doesn't help anybody. It's all being salted anyway. So each individual crack has to happen by itself. And, you know, if you find out that it takes too long on some platform, you can always turn it back down. But I'd start at 1234567. But really, in this era of GPU-driven password cracking, where GPUs hash at lightspeed, PBKDF2 is showing its age. Cranking up iterations is just running ahead of a moving train. It would make much more sense to just get off the tracks.

**Leo:** Well, I asked about that last week. Argon2 is another option.

**Steve:** Yes. Since SQRL's entire security model is based upon the security of a single password-based key, I gave this a great deal of thought years ago. Everyone, including Bitwarden, ought to abandon the use of any non-memory-hard password key derivation which GPUs excel at. SQRL uses "scrypt," also known as Scrypt, S-C-R-Y-P-T, which absolutely requires a block of dedicated memory which cannot be shared among cores. Scrypt's many parameters are tunable, and I don't now recall exactly how much memory I required, but I think it was 16 megabytes. I chose that because every smartphone can spare that, and it's only needed briefly to process a user's password entry. But significantly, GPUs are unable to follow since they're unable to run Scrypt of that size at all.

**Leo:** Oh, nice.

**Steve:** So when that much memory is required, GPUs are out of the game. So switching away from PBKDF2, whose time has passed, ought to be on every password manager's roadmap for the future.

**Leo:** Good to know. How about Argon?

**Steve:** And there's Argon, yes. I mean, that's the same thing. These are all memory hard. And so just as a little quick example, the way this works is the algorithm allocates 16MB of memory. Then it uses a very secure hash, a keyed HMAC, based on the password that you've giving it, to fill this memory with pseudorandom data where each value is a pointer within that memory. So it fills 16MB with pointers from zero to 16MB. Then it follows the pointer trail, jumping throughout all of that memory.

Well, it turns out there is no way that has ever been found to short-circuit that process. That is, you don't technically need to use 16MB. But if you didn't, you would need to compute what some random pointer somewhere out there in 16MB space would be. And then, when you go to it, you would need to then compute what the other pointer somewhere else would be. In other words, if speed is your goal, the fastest way to solve this puzzle is just give it 16MB, let it fill it with pointers, and follow these pointers all around hell and gone, you know, within this 16MB space, and the path you take ends up resulting in the key that this thing synthesizes. And it cannot be short-circuited, and no GPU can do this.

So you just switch to something that GPUs cannot do. And then you're not constantly having to, as I said, run ahead of the train, staying on the tracks. You just say, nope, let's use a different protocol. And again, I've been using, you know, SQRL uses a 16MB Scrypt algorithm, and it runs everywhere perfectly, on iPhones and Androids. Nobody has ever had a problem with it. And it cannot be accelerated by a GPU.

Okay. A couple last bits. Robert van den Breemen, he said: "@SGgrc Thanks for the honest podcast on LastPass. Your vetting years back made me use it for many years. Now moving to Bitwarden. Now that I'm changing my 1000-plus passwords, I see how broken the system of password login really is. Why is there no change password API?" And, you know, that's a really good question. A uniform, standardized, cross-site password change API would make rotating all of one's passwords an automatable operation.

**Leo:** Right.

**Steve:** I think that the problem is mostly per-site resistance. One of the complaints most people have is that every site is different, has different password requirements, additional bells and whistles like security questions or not, and different password recovery approaches. And this arises from the fact that every site wants to be different. There's no uniformity, so each site gets to invent the user experience flow that they prefer for their particular whim. It didn't have to be this way, but it's the way it is. Okay. And last one.

**Leo:** And I should point out, the reason the Internet is the way it is, is because nobody sat down and designed the whole thing from scratch. If they had, they would have put a password rotation system in.

**Steve:** Yes.

**Leo:** But it wasn't designed that way.

**Steve:** Right. And you would never tell your mother that she must type https://.

**Leo:** Even, by the way, when somebody asked the creator, the founder of the web about that, he said, "I never thought humans would ever have to type that. It was supposed to be machine-readable."

**Steve:** It was link following.

**Leo:** Yeah. It wasn't supposed to be anything a human would ever have to see.

**Steve:** You were on a page, and it had nice English links or whatever language.

**Leo:** Right. So it's all under the hood.

**Steve:** Yup, exactly.

**Leo:** Yeah, yeah. But so Tim Berners-Lee did not design it. You know, but that's what happens in the real world. Stuff gets done, and it works.

**Steve:** Yup.

**Leo:** And it changes, and all of a sudden you've got Twitter. It's called evolution, or devolution, whatever it might be.

**Steve:** Okay. Last one. Andy Olson tweeting from @AvgAndy. He said: "@SGgrc I listened to the recent LastPass episode. Switching to Bitwarden. Just wanted to note that password changes are necessary, but I found I can also change usernames on a lot of important sites. If user login is email, change that, too."

And yes, I agree with that, especially since today's standard for password recovery, you know, handling the "I forgot my password" event, is to send the password reset link to the user's account email. So, while it's not imperative, if it's easy for you to also change your account's email while you're at it, from what it was when your LastPass vault was copied and stolen, then there's no reason not to.

Okay. So once again I've used up our time this week covering this news, which is huge for this podcast's listeners since such a large number of us chose and have been using LastPass. But even for those who had not chosen to stay with LastPass or long ago chose a different password manager, all the information about GPU cracking strength and PBKDF2 iteration counts is universal, as is the need to urge whatever password manager that it's time to move away and beyond PBKDF2. There are many cryptographically verified alternatives, and they should be used. You know, racing ahead of GPUs no longer makes any sense.

As I said, next week we'll be able to get some sense for the amount of the ECB cipher mode that our listeners discovered in their LastPass vaults with the aid of Rob's and ChatGPT's very nice PowerShell script. But unlike this week's bombshell that many iteration counts were 1, the presence of any lingering ECB won't present a five-alarm fire.

**Leo:** I really wish LastPass were giving us more information. I understand they don't want to because it's a black mark on their escutcheon. But I think their escutcheon has been scuttled already. So maybe you should just start telling us what we need to do. It's my guess that when they did up the iterations, they didn't retroactively fix everybody's vault. Maybe they couldn't? Maybe they didn't know how to? Maybe they didn't want to.

**Steve:** Yeah, I've covered all that. They could.

**Leo:** They could have.

**Steve:** They didn't. They could have, they should have, we don't know why they didn't.

**Leo:** Yeah. At least send out an email saying - and maybe they did. I don't remember. But, yeah.

**Steve:** No. An email is insufficient. Leaving anyone's iteration set to 1...

**Leo:** As you said, it's like [crosstalk].

**Steve:** ...the title of this podcast, is unconscionable.

**Leo:** Yeah.

**Steve:** I mean, you could argue it is not true that, like today, when this theft occurred, that offered any protection. And they were assuring everybody that, if you use a good long and strong password, you are safe. That is not true.

**Leo:** That's a huge thing, that because of something they did, your good long strong password was insufficient.

**Steve:** Yes.

**Leo:** That's not good. As always, great stuff, Steve. I appreciate it. One thing I'll add to one of your commentators who had a shared vault.

**Steve:** Ah, yeah, yeah, yeah.

**Leo:** He had his personal vault and his business vault in LastPass. We do that, as well, here. It's my understanding those are kept separately. So it's just a separate vault. It's not like there's one vault. Even, now, he did reuse the same password on both.

**Steve:** But it was 25 totally random characters.

**Leo:** It was a good password, yeah,

**Steve:** He has nothing to worry about.

**Leo:** And he has a good number of PBKDF2 iterations. We are actually - we use LastPass Enterprise. And we're talking about what we need to do. It's a massive - and that's the other thing I would say, we got a number of emails and comments from people in our forums that, well, you don't understand, I've got a thousand passwords in there. I do understand. I completely understand. That's the problem.

**Steve:** And Leo, even without any decryption, think of the profiling that can be done of a person.

**Leo:** Well, that was terrible. They left that unencrypted, all that metadata. Yeah, that's terrible, terrible. No, I don't - this is the other thing that I don't know, but I think there's more. The other shoe will drop down the road because how can LastPass survive this? And when the stakeholders at LastPass say, well, you've got no customers, at some point they're going to turn off the servers, and there's going to be another collapse because I don't see how they stay in business. What happens when a password manager goes out of business?

**Steve:** That's a really good question. Not something we've had to face before.

**Leo:** No. I mean, that's a really - so that's why I think it does behoove everybody, as you've done and I've done, to move off, even though, yes, it's a big pain in the - it's easy to move, actually. You said that, and I'll say it. It's easy to move.

**Steve:** Oh, my god. I had resistance to it, and it was like, wait a minute, I'm done?

**Leo:** Yeah, it's trivial. And most password managers work roughly the same as LastPass with the autofill and the authenticator. Oh, another thing somebody was asking in the chatroom, hardware key, like you know I've mentioned I use YubiKey, recommended? Good idea? I would just point out that at least Bitwarden, I think LastPass too, they all have fallbacks, usually to an authenticator.

**Steve:** The only thing, from a crypto security standpoint, the only thing those offer is the convenience to their user of when you log into your vault.

**Leo:** Of a YubiKey.

**Steve:** It provides no additional security.

**Leo:** If you could, right, because the vault doesn't use this for the brute forcer. They don't - it's not the second factor.

**Steve:** They don't care.

**Leo:** They don't care.

**Steve:** They don't care, yeah.

**Leo:** And if you turned off secondary fallbacks, then I guess this would be stronger than, say, an authenticator, certainly a lot stronger than an SMS message.

**Steve:** Yes. And in fact that was one of the features that SQRL had was that after you got comfortable with it, in the UI you could check a box saying Do Not Allow Any Other Authentication Mechanism.

**Leo:** The fallback is the weakest link. Whatever the weakest link is, that's the weakest link. That's all you've got. All right, Steve. I am so glad you did a second episode on this. I know there are a lot of people saying, yeah, but what about? But there hasn't - it's good news. The hackers rested. They were busy changing their LastPass passwords. So, you know, we had a little breathing space. Next week the news.

**Steve:** Yes. We will, we'll do a big news catch-up. I'm sure, as I said, we'll briefly talk about how many people found ECB in their vaults.

**Leo:** I'll be curious, yeah.

**Steve:** It'll be, yeah, it'll just be a curiosity point. It's not the end of the world. And but then we'll catch up on all the news that's been happening.

**Leo:** I wish I could look at my old vault. And this is the other thing LastPass hasn't said. When did that backup get made? I deleted my LastPass vault a couple of years ago, but maybe the backup was from 2.5 years ago.

**Steve:** Yeah.

**Leo:** In which case my vault's in there, too. And so, and they haven't told us anything. So it's very disappointing.

**Steve:** No, in fact I did, I went to the blog to see whether, I mean, just recently, a day or two ago, to see whether there was any response, any further information, anything else. No.

**Leo:** No. They're just hoping this goes away. At this point they've got nothing to do. They're hoping it goes away. And it's not. I'm sorry to say it's not going to go away. One other point I will make on Bitwarden, because it's open source, if you have advice for Bitwarden or a change you'd like to see, for instance changing to a memory-hard hashing, that's where you go to GitHub, and you do a pull request. Not a pull request, a PR. Not a pull request. What is it you do? Anyway, you issue, you give them an issue saying, please. That's how you do complaints to Bitwarden.

**Steve:** It's time. It's time.

**Leo:** It's open source. So you go to the open source repository, and you enter an issue saying there's an issue, you should use a memory-hard hashing algorithm. And they have to respond. They will respond to that. And you know what will happen because it's open source? Somebody will write it.

**Steve:** Yeah.

**Leo:** Say here it is.

**Steve:** Well, it's already been written. It just needs to be hooked in.

**Leo:** Hooked in.

**Steve:** It'll take an afternoon.

**Leo:** Well, then they should do it, yeah. Thank you, Steve. Bless you, Steve, for the job you do. It's so important, and we really are grateful to you. Steve's website, GRC.com.