Security Now! #904 - 01-03-23

Leaving LastPass

This week on Security Now!

This week, since a single topic dominated the security industry and by far the majority of my Twitter feed and DM's, after a brief update on my SpinRite progress, we're going to spend the entire podcast looking at a single topic: LastPass.

There are no words...

Security News

If there was any security news I never had time to track it down since I was digging deep into today's topic which exploded my Twitter feed. So, next week I'll be looking back over everything that has transpired in the security world – other than LastPass – since our last podcast of last year, and we'll catch up.

SpinRite

On the last podcast of 2022 I mentioned that the 8th alpha release of SpinRite had been the prior weekend. There's been no release since. Mostly, those releases were intended to resolve some mysterious behavior that people were reporting on seriously damaged drives when they were testing SpinRite against their boxes of old drives which many of them had been holding onto in anticipation of this day. Because I was focused upon resolving those mysteries I was not also fixing the growing number of non-mysterious things that people were noting. Those were things like during intense data recovery SpinRite was not updating its on-screen clocks, or if a system hung or crashed the permanent log of all the work done up to that point would be lost, because SpinRite's log was being kept in RAM until SpinRite's graceful exit. And thanks to all of the feedback during that first flurry of testing, I saw that SpinRite could be much more aware of drive trouble now that it wasn't being isolated from the drive by the system's BIOS.

Essentially, the feedback from that extensive initial testing showed me some ways in which I could improve upon my first take. I want 6.1 to be as good as it can possibly be, not only because we're going to be living with it for a while during the work on SpinRite 7, but because every way in which I can make SpinRite 6.1 better today will be inherited by SpinRite's future.

So, I've reworked some significant portions of SpinRite. Logs are now being incrementally written to non-volatile media so that everything that has happened so far will be saved in the event of a system hang or crash. Only if the user wishes to write SpinRite's log back to the drive SpinRite is running on will the log be spooled to RAM and written at the end (and they're notified that's what's going to happen.) I've completely rewritten SpinRite's clock management and completion estimation system, and SpinRite is now providing much more feedback about the state of critically ill drives, since that information is available.

I have a list of ToDo items, and our GitLab contains a lot of feedback. So I plan to continue to work on the 9th alpha release until I've done as much as I can with all of the information I have. Even though we may still need a few more iterations to get to beta, that will give us time to be verifying all of the changes I've just implemented.

Leaving LastPass

We last talked about LastPass following the November 30th disclosure of the second breach, the one following their first breach four months earlier, in August. I noted then that the first announcement of the first breach was followed-up exactly 21 days later with the results of their forensic analysis. So I noted that we might expect to receive an update on the second breach three weeks later. Three weeks after Wednesday November 30th was Wednesday December 21st and the follow-up arrived one day later on December 22nd.

The "Likes" and click-bait driven tech press and some security researchers postings suggested that LastPass had deliberately timed their "bad news" for release shortly before Christmas in an attempt to have it swept under the rug. Given the timeline we've seen LastPass adhere to previously, I think that's clearly nonsense, and it's the kind of crap that causes any careful reader to wonder whether what follows will be objective reporting of facts or a subjective smear.

Two days after Christmas, Jeremi Gosney, a widely recognized expert in password cracking, wrote a lengthy and inflammatory take down of LastPass over on his new Mastodon account, having recently moved there from Twitter. While Jeremi made some excellent points that I'll be touching on here in a few minutes, he suffered from the "piling on" syndrome when he wrote:

> *LastPass's claim of "zero knowledge" is a bald-faced lie. [...] nearly everything in your LastPass vault is unencrypted. I think most people envision their vault as a sort of encrypted database where the entire file is protected, but no -- with LastPass, your vault is a plaintext file and only a few select fields are encrypted.*

What does one say to that? There's nothing about that that accurately conveys the truth or even a sense of the truth. As evidence of that, another well-known researcher, Wladimir Palant, the creator of Adblock Plus, who blogs from his site titled "Almost Secure", has been blogging a great deal about LastPass recently. The day before Christmas, Wladimir posted an article titled "What data does LastPass encrypt?" and he provided a snippet of code that could be dropped into any web browser's developer console to retrieve the logged-on user's data blob directly from LastPass' cloud server:

```
fetch("https://lastpass.com/getaccts.php", {method: "POST"})
  .then(response => response.text())
  .then(text => console.log(text.replace(/>/g, ">\n")));
```

Any current LastPass user can obtain their encrypted vault data to examine it for themselves. Open any browser. I used Chrome for this and I also tested on Edge. Login to LastPass so that you're looking at your vault. Press F12 to open the developer tools. Select the "Console" tab to move to that view. You'll have a cursor. So paste the short 3-line JavaScript query shown above into the console and press Enter. If everything worked, your page will fill with a large XML format dump of name=value pairs. But the vault is far larger than your page. Look carefully at the bottom of the page where Chrome or Edge will be saying "Show More" and also offering a "Copy" button. Click "Copy" to move all of that query response data onto your machine's clipboard.

Open a text editor (I used Notepad++ on Windows) then paste the clipboard into the editor and save the file so you don't lose it. Passwords begin with p= followed by a double-quoted string. (p="....."). AES-CBC encoding has two parts, the first being a pseudo-random initialization vector which CBC requires and the second being the encrypted string. The older format passwords will only have a single ASCII-encoded string which is the ECB format encryption of the data since ECB does not use any initialization vector – which is the reason its use has been deprecated. I wrote a quick RegEx to scan my encrypted vault for any non-CBC passwords and none were found (**p="[^!]**). That's not conclusive. But I'm pretty sure that my vault contains very old and original passwords which predate LastPass' switch to CBC. So, anecdotally, I'm not seeing evidence of old ECB encryption that wasn't autonomously upgraded.

For those who are interested, Wladimir's blog posting titled: *"What data does lastpass encrypt"* provides additional details and I've linked to it here in today's show notes:

https://palant.info/2022/12/24/what-data-does-lastpass-encrypt/

But on the question of whether any sensitive unencrypted information is contained therein, every listener can now obtain and examine their encrypted vault blob directly from LastPass' servers for themselves. I know with absolute certainty that we have a bunch of competent hacker/coders here. I'm not going to let this distract me since I'm going to stay focused upon SpinRite – which those same hacker/coders very much want me to finish. But I would love to have some in this community create transportable code that will de-obfuscate the simple non-encrypted HEX and BASE64 encodings of URLs and other information so that we're able to see more of what's in the LastPass blob.

So grab your blob now and next week I'm sure I'll be able to share some solutions for turning it all into plaintext. And if the wider Internet has already done so I'm sure I'll be receiving pointers to that. In the meantime, for what it's worth, someone who replied to Wladimir's post obtained and examined their encrypted blob and was not concerned. They wrote:

> *Thank you for this article. As a longtime LastPass user, the latest news was unsettling. I dumped my data following your instructions and took a very careful look. I found that username/id, passwords, account names, account numbers, and especially the notes (extra) were all encrypted. I did searches on specific sensitive data, and literally inspected every page, and found no sensitive data that was in plaintext. On the other hand, all of the URLs were in hex, which to my mind is not encrypted; this for me is a lower priority, as my browsing history is collected by everybody these days, as evidenced by instant ads for something I just started searching for. I was considering changing password managers, but do not plan to at this point. It has prompted me, however, to change my master password (which has 2FA protection also), and to review my most sensitive data and ensure it is as secure as possible; strong passwords and 2FA, etc. Thanks again for this article; it was most useful.*

Okay. At this point there are about five different directions I want to take this, because we have so much more ground to cover – more than you might guess. By the end of this podcast I believe that you'll have sufficient information to make an informed decision on your own.

And it won't be the result of a bunch of inflammatory and exaggerated half truths.

Let examine what LastPast posted on December 22nd which has stirred up this hornet's nest:

> *Based on our investigation to date, we have learned that an unknown threat actor accessed a cloud-based storage environment leveraging information obtained from the incident we previously disclosed in August of 2022. While no customer data was accessed during the [initial] August 2022 incident, some source code and technical information were stolen from our development environment and used to target another employee, obtaining credentials and keys which were used to access and decrypt some storage volumes within the cloud-based storage service.*
>
> *LastPass production services currently operate from on-premises data centers with cloud-based storage used for various purposes such as storing backups and regional data residency requirements. The cloud storage service accessed by the threat actor is physically separate from our production environment.*
>
> *To date, we have determined that once the cloud storage access key and dual storage container decryption keys were obtained, 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.*
>
> ***The threat actor was also able to copy a backup of customer vault data from the encrypted storage container which is stored in a proprietary binary format that contains both unencrypted data, such as website URLs, as well as fully-encrypted sensitive fields such as website usernames and passwords, secure notes, and form-filled data.***

That last sentence was the biggie. LastPass admitted that their off-site cloud backup of their customers' mostly-encrypted vault data was exfiltrated and is now in the hands of malicious actors. This is the eventuality that all of this TNO client-side encryption was designed to address. The idea is that as the user has chosen a strong and unguessable master password, the only thing the bad guys got was some less critical metadata, including some web browsing and IP usage history.

One thing needs to be made very clear: While changing and strengthening your LastPass master password **now** would make any next similar loss by LastPass less dangerous, only changing the passwords of the websites stored in your LastPass vault will have any effect. The vault blobs that have been stolen were encrypted under your LastPass master password at the time. THAT is the one that the bad guys will be trying to crack. Therefore, changing your LastPass master password now will have no effect upon that.

So let's talk about cracking these mostly-encrypted vault blobs...

Way back before some clever hacker realized that they could program a GPU to run a cryptography hash function very very quickly, passwords were protected by hashing them, like, once. That was plenty since the crypto geniuses who designed these hash functions did so specifically to provide the guarantee that passing a password through a hash would produce a unique value that could not be reversed to obtain the password that was originally fed into the hash. So, no problem, right? Well, not quite right.

The way to defeat this mechanism is to get a super-fast hashing engine, like a GPU, then run every possible password through the hash and see whether what comes out matches what came out when the user originally hashed their password.

Somewhere along the way hackers realized that doing that everytime they wanted to crack a password was highly redundant. And this was around the time that hard drive mass storage prices were falling. So the concept of a "Rainbow Table" was born. The idea was to run all possible passwords through a common hash function just one last time, but this time to store the result along with the password that was input. Then, when presented with a hashed password, the password that had generated that hash could simply be looked up if that password had ever been added to the Rainbow Table.

The trouble was, hash functions were designed to be both secure and efficient where efficient means fast. But "fast" is the reverse of what we want for passwords, since we want to prevent this sort of brute-force password guessing to obtain a hash.

So Password Based Key Derivation Functions – known as PBKDF – were created. There are a handful of popular PBKDF's that have various features and tradeoffs. But most rely upon iterating some core function. If performing a single password hash is too fast, then hash the hash, and then hash that, and hash it again, and so on.

Where I'm headed with this is that back in 2008 when Joe Siegrist founded LastPass, a single hash provided sufficient security. GPUs hadn't yet been deployed for hashing and computers were much slower than they are today. But through the years all of that changed. So LastPass began "iterating" their PBKDF.

The switch from 1 iteration to 500 occurred in June of 2012, four years after LastPass' initial launch. That only held for about eight months until February of 2013 when LastPass' default iteration count was jumped by a factor of 10 to 5,000. And finally, five years after that, nearly five years ago in February of 2018 the last jump was made to 100,100 iterations. If this sounds somewhat familiar it's because we talked about this change on the podcast at the time.

Now let's switch from password hashing iterations to the strength of the password itself.

LastPass originally enforced a minimum master password length of 8 characters. But that, too, needed upgrading as our systems became faster, and as asking users to invent stronger passwords became more socially acceptable. Remember how far we've come from the days when a user's single password, that they reused across all of their websites, was written on a yellow Post-it note stuck to their large CRT screen. Back then, passwords were sheer annoyance with no perceived value. How things have changed.

Five years ago, in 2018, LastPass decided that they needed to update their minimum 8-character password length and to recommend using numbers and both uppercase and lowercase letters. But Wladimir Palant noted something distressing about LastPass' master password strength policy. He wrote:

When LastPass introduced their new password complexity requirements in 2018 they failed to enforce them for existing accounts.

In other words, LastPass chose to leave their existing customers' possibly too-short 8-character passwords alone. Maybe they didn't want to ruffle feathers or perhaps they were worried about the support requirements. And presumably anyone who has changed their LastPass password in the last five years will have been required to strengthen it when setting its replacement.

While it's true that LastPass has no idea about the length of their customers' passwords, since all they receive is a well-hashed blob, it would have been trivial for LastPass to add some logic to the LastPass clients to scold their users when the client notices that the master password being submitted no longer meets contemporary complexity requirements. Most users would have taken heed of such advice from their password managers... and, again, these days we've all grown accustomed to such requirements.

So, assuming that Wladimir is correct, and I have read another anecdotal account from someone who, today, still has an 8-character LastPass master password protecting an unused account, this means that an unknown number of LastPass user accounts, whose data was recently stolen from the LastPass cloud backup, may have only been protected by 8-character password.

And – unfortunately – it gets worse. There are also reports within the LastPass-watching security community that LastPass' PBKDF2 iteration count which was also jumped from is too-low setting of 5,000 up to 100,100 in 2018, was not pushed out to all LastPass users. If this is true, it's truly horrifying because taken together it means that there might be LastPass vaults which were allowed to exist for the past five years, since these improvements were made, which are protected by short 8-character passwords hashed 5000 rather than 100,100 times.

All LastPass users should check their master password iteration counts. Open your LastPass vault, select "Account Settings" on the lower left. On the default "General" tab at the bottom of the page click "Advanced Settings" and under "Security" look for Password Iterations. It should be at least 100,100. If you're planning to remain with LastPass, you should increase that setting to 350,000 which is only a bit higher than what is currently considered best practice.

Since I want to share some of my feelings about some comments recently made by the well known password cracker, Jeremi Gosney, and since everyone listening to this knows quite well who I am, I wanted to introduce Jeremi a bit. The relevant paragraph of his Linked-In bio says:

I am also a core developer of Hashcat, a popular OpenCL-based open source password recovery tool, and I am widely regarded as one of the world's top password crackers. I was named one of the "Top 100 Security Experts" in 2013. I was also one of the winners of Cloudflare's Heartbleed challenge, and was one of the first to publish a working private key recovery exploit for Heartbleed. My work and research has been featured in hundreds of news articles, and has even been incorporated into university classes and certification courses. I additionally served as a judge on the Experts Panel for the Password Hashing Competition.

After our second break we'll look more closely at some of the points that Jeremi made.

Okay. So, this guy is clearly a serious developer techie who is well able to understand everything that he sees within a password manager. His takedown of LastPass is so scathing that I couldn't help wondering whether he had swung from loving it to hating it. Here's how his posting began:

> *Let me start by saying I used to support LastPass. I recommended it for years and defended it publicly in the media. If you search Google for "jeremi gosney" + "lastpass" you'll find hundreds of articles where I've defended and/or pimped LastPass (including in Consumer Reports magazine). I defended it even in the face of vulnerabilities and breaches, because it had superior UX and still seemed like the best option for the masses despite its glaring flaws. And it still has a somewhat special place in my heart, being the password manager that actually turned me on to password managers. It set the bar for what I required from a password manager, and for a while it was unrivaled.*
>
> *But things change, and in recent years I found myself unable to defend LastPass. I can't recall if there was a particular straw that broke the camel's back, but I do know that I stopped recommending it in 2017 and fully migrated away from it in 2019. Below is an unordered list of the reasons why I lost all faith in LastPass:*

I'm not going to drag everyone through all of Jeremi's ranting because I don't think it's rational nor fair to LastPass. Recall that even though Jeremi is clearly a highly skilled and qualified technologist, it was he who wrote in this rant that:

"LastPass's claim of "zero knowledge" is a bald-faced lie. [...] nearly everything in your LastPass vault is unencrypted."

We already know that's not true. We know that everything sensitive and important **is** encrypted. How could it not be? Perhaps Jeremi is choosing to take issue with the "zero" in "zero knowledge" even though he knows quite well that that's not what "zero" means in this context. But amid his ranting he did make a number of points and they bear consideration:

The biggest concern is another of those legacy issues which LastPass seems to have been reticent to address. Recall that a few months back last October, in our podcast #893, we talked about Microsoft's decision to leave their Office 356 message encryption using the Electronic CodeBook ECB cipher mode. And to illustrate the danger of that we showed the classic Linux Penguin where using Cipher Block Chaining (CBC) mode encrypted the penguin's image into a pure high entropy rectangle of noise, whereas the use of ECB left a very visible penguin in the image.

Well, it appears that LastPass was also originally using the clearly less secure ECB mode to encrypt passwords. Then, somewhere along the way they realized that this was no longer the right solution, so they began encrypting any newly saved passwords with the more secure CBC mode, but for some reason they never proactively re-encrypted the original less secure ECB passwords under CBC.

As a consequence, a user's vault will contain a mixture of old passwords encrypted under ECB mode with newer ones under CBC. I can find no possible reason for this being left as it has been. I also have no verification that this is true. But I have seen others noting that the use of ECB by LastPass meant that users who were reusing the same password across multiple sites would

therefore have identical ECB encryptions. Which is true. And while that's a bit disturbing, I wondered whether at one time that might have been a deliberate feature to allow LastPass to spot password reuse within their users' vaults and to perhaps warn them. As we know, this is readily done on the client-side since the client has access to the entire LastPass vault.

And that brings me to another of Jeremi's points. He writes:

> *LastPass has terrible secrets management. Your vault encryption key [is] always resident in memory and [is] never wiped, and not only that, but the entire vault is decrypted once and stored entirely in memory.*

As we know, I've been saying recently that it would be nice if the LastPass vault were being incrementally decrypted so that only the one password needed for login was decrypted from the opaque blob, after which its plaintext would be overwritten. But according to Jeremi, that doesn't appear to be the way LastPass manages the user's vault.

And as for the encryption key always being resident in memory, that's a pure requirement of any password manager that isn't constantly pestering you to reauthenticate to it. None of us want to be constantly doing that. But if you did, LastPass offers the ability to auto-logout after 'x' minutes of inactivity at which point it would presumably wipe all decrypted content from RAM and require the user to login again before its next use.

So that's not "terrible secrets management" as Jeremi characterizes it. It's a necessary tradeoff made for convenience and every other password manager will need to be similarly terrible in order to get its job done without pestering its user to death.

And we all need to appreciate that none of the password managers are pretending to protect their users from client-side machine attacks. There is simply no protection for that – ever from anyone. That isn't available. We're getting the promise that remote websites cannot access our vaults and that our password manager providers – and anyone who might attack them – also cannot access our vaults. And on that last point it appears that LastPass has made a series of design policy decisions through the years, for reasons only they know, that may have left their users less secure than they could have been in the event of the attacks they have just suffered.

Jeremi also notes that while LastPass' vault key uses AES256, its 256-bit key is derived from only 128 bits of entropy. If true, that's also unfortunate. Although 128 bits of pure entropy is plenty, why not take the opportunity to generate and use all 256 bits for the AES key?

Again, there may be an engineering reason, but overall it feels as though the original security design of LastPass, which in 2008 was ample – with even Jeremi jumping up and down defending it – has not aged well and that LastPass' caretakers have not been as excited about keeping LastPass on the cutting edge as crypto enthusiasts, like Jeremi or I, would have been.

And that's why, after having surveyed all of the available commentary, and thinking about everything I've recently learned, I've decided to pull up stakes and leave LastPass.

Jeremi noted that this most recent breach is LastPass' 7th in the past 10 years. I didn't verify that number, but we all know they've had their share. And while everyone knows that I'm the first person to forgive a mistake under the theory that they are often unpreventable, this recent breach of their cloud backup provider which their November disclosure said they shared with their affiliate, GoTo, makes one wonder whether some corners may have been cut in the interest of profit. Only insiders know.

But there really isn't any excuse for the engineering decisions they've made which have made the consequences of their now having lost their customer vaults potentially much more serious. They could and should have pushed their legacy users to move to a longer and stronger master password. They didn't. They could and should have had their LastPass clients upgrade all older ECB password encryption to CBC. They didn't. They could and should have absolutely upgraded every user from 5,000 iterations of PDKDF2 to at least 100,100 iterations. They didn't. They could and should have kept me as a loyal and faithful LastPass user & evangelist. They didn't.

So, what's my next Password Manager? After our final break I'll explain the thinking and reasoning behind my choice.

During my post-incident survey of security professionals, three cloud-based password managers kept being mentioned over and over. They were: DashLane, 1Password, and Bitwarden. We all know that Bitwarden is an active sponsor of the TWiT network and a frequent advertiser on this podcast. So I was glad to see other knowledgeable researchers praising it. Here's what Jeremi Gosney wrote once he had calmed down a bit from his being jilted by LastPass. He wrote:

> *So, why do I recommend Bitwarden and 1Password? It's quite simple:*
>
> *- I personally know the people who architect 1Password and I can attest that not only are they extremely competent and very talented, but they also actively engage with the password cracking community and have a deep, *deep* desire to do everything in the most correct manner possible. Do they still get some things wrong? Sure. But they strive for continuous improvement and sincerely care about security. Also, their secret key feature ensures that if anyone does obtain a copy of your vault, they simply cannot access it with the master password alone, making it uncrackable.*
>
> *- Bitwarden is 100% open source. I have not done a thorough code review, but I have taken a fairly long glance at the code and I am mostly pleased with what I've seen. I'm less thrilled about it being written in a garbage collected language and there are some tradeoffs that are made there, but overall Bitwarden is a solid product. I prefer Bitwarden's UX and I've considered crowdfunding a formal audit of Bitwarden, much in the way the Open Crypto Audit Project raised the funds to properly audit TrueCrypt. The community would greatly benefit from this.*

I know from my Twitter feed that many of my Twitter followers, or at least those who are Tweeting to @SGgrc, are using 1Password. I've not looked at it closely. But from what Jeremi says, that would appear to be a solid choice. Another password cracking enthusiast by the name of Steve Thomas ranks DashLane first, Bitwarden second and 1Password third, but only because DashLane is using his favorite pet password key derivation function known as Argon2. Argon2 is a memory-hard function designed in 2015 which is highly resistant to GPU attacks. But its implementations need to be careful about side-channel leaks since the original design accesses memory in a password-dependent sequence. As a consequence, improvements have been made since then. But a password manager's choice of its key derivation algorithm is incidental at most, and the strength of any good function can simply be turned up as needed over time. So it appears that all of these three are in the running.

Next, I went to checkout the personal plans these three offered. Blessedly, I don't need a family plan or a business plan or anything other than a "Just please keep all of my passwords safe, secure and synchronized among all of my various devices", plan. Although the value of a password manager is now well proven, so that asking for some money should not be a problem, I like the idea of being able to turn people on to it so that they can take it out for a spin without needing to pay in advance for a year's commitment. In other words, a useful free tier – such as LastPass once had but then abandoned – is part of my criteria for the perfect password manager.

Personal Plan Details for

DashLane	$0 - 1 device only	$33/yr for unlimited devices
1Password	– no free tier –	$36/yr for use
BitWarden	$0 - unlimited	$10/yr to support

Unfortunately, DashLane's free plan only allows for the use of a single device. So you can't use it on both your desktop and even one mobile platform? That's crazy. Why allow any devices if you can't use at least two? So you need to pay, and their minimum plan is $33 per year, in advance, which does provide for an unlimited number of devices.

1Password doesn't even try to offer a free tier. If you want to use 1Password you pay $36, in advance, for a year to use it.

Bitwarden is the only one of these top three to offer an actually useful free plan which allows for the use of any number of devices. And as Leo often notes when he's talking about Bitwarden, since their free plan really does everything you'll probably need, their $10 per year paid plan, at less than 1/3rd the price of the others, is mostly just there just to support them.

And, as Jeremi noted, the icing on the cake is that Bitwarden is also 100% open source. I saw a Tweet pass by some time ago that our old friend Alex Neihaus was moving his family to Bitwarden and self-hosting a Bitwarden server in his own cloud. So those are the sorts of things that the use of a truly open password manager can provide. And while not of interest to everyone, I imagine that it's the sort of thing that would appeal to this podcast's audience.

On December 7th Bitwarden posted a blog titled:
https://bitwarden.com/blog/new-deployment-option-for-self-hosting-bitwarden/

And in my digging around over the holidays I stumbled upon a Bitwarden page which linked to their past annual 3rd-party outside network security and penetration testing audits. I could not find that same page later, but I found another that provides the same info:

https://bitwarden.com/help/is-bitwarden-audited/

I also noted that they're signed up with HackerOne to offer and manage bug bounties.

The reason I originally chose LastPass and was comfortable endorsing it, was that its author opened up its internals to me so that I could understand exactly how it worked. And it was solely on that basis and for that reason that I chose LastPass. I don't regret the decision I made back then. LastPass has been a flawless companion for many years. But as we've observed earlier on this podcast, the world has been changing ever since, and LastPass no longer fits the way it once did. It and its organization is beginning to act and feel a bit too old and creaky... which is not what anyone wants in their password manager.

An example of a different more aware and contemporary approach is Bitwarden's description of their management of their user's master password. They write:

> *SHA-256 is used to derive the encryption key from your master password. Bitwarden salts and hashes your master password with your email address locally, before transmission to our servers. Once a Bitwarden server receives the hashed password, it is salted again with a cryptographically secure random value, hashed again, and stored in our database.*
>
> *The default iteration count used with PBKDF2 is 100,001 iterations on the client (client-side iteration count is configurable from your account settings), and then an additional 100,000 iterations when stored on our servers (for a total of 200,001 iterations by default).*

That approach certainly beats a legacy password manager that reportedly never bothered to push its longtime users to use a longer master password or to upgrade their decade-old crypto.

So, I went to, yes... https://bitwarden.com/twit/ and signed up for their $10 per year plan. I have no problem supporting them, and I like it that I'll be able to recommend them to others without newbies needing to pay anything up front.

Then I went to a menu item I had never used in my LastPass vault: **"Export"** and I exported a 77 Kbyte CVS file. I opened my shiny new Bitwarden web interface and under "Tools" at the top was the menu item "Import Data". From a drop down menu there, I selected the import source as being a "LastPass CSV", provided the filename and watched a perfect error-free transfer of my entire legacy LastPass data into Bitwarden. My password database, auto-fill credit cards and all of my secure notes made the move without incident or complaint.

I'm still just dipping my toes in, but it has happened. I am making the move. I don't have sufficient experience yet to provide useful feedback, but I'm sure I'll have my reactions to share by next week after I've been using it for a while.

The last thing we need to talk about is the remediation of any danger which may arise from any prior use of LastPass.

When I was deciding upon the title for today's podcast I was tempted to title it "Don't Panic" with a nod to Douglas Adams, because I doubt that anyone should panic. As I noted earlier, you should definitely check your LastPass Password Iterations to assure that it had been bumped up to 100,100. And **please** shoot me a note if you discover that it's not 100,100. I would love to have some corroboration of that most disturbing claim if some LastPass users were never upgraded.

But assuming that your iteration count is 100,100, and that you're using a master password with good entropy, other than some incidental personal information disclosure of the sort that other commercial entities you work with on the Internet also have, your actual risk of having your own vault decrypted is very low.

If you were using a very high entropy password with 50 bits of entropy, a single GPU attempting to crack the LastPass default of 100,100 iterations of an SHA256-based PBKDF2 would require 200 years and an estimated cost of $1.5 million. Now, 50 bits is a lot of entropy. Studies have shown that the average password only contains somewhere around 40 bits of entropy. Since that's 10 fewer bits, the strength is 2^10 weaker. 2^10 is 1024. So a 40-bit password would be about 1000 times less strong. So, that brings us down to a couple of months and $1,500 to crack. And this of course reminds us why the strength – and length – of our password is so crucial for avoiding brute force attacks. Remember "Password Haystacks?" It taught us the lesson of how easy it was to create long and highly brute-force resistant passwords.

The second factor mitigating our risk (if you'll pardon my pun) is the presence of second-factor authentication. Everyone who is listening to this podcast almost certainly has a time-based authenticator and has added the requirement for its use to their most important online accounts. For iOS I still prefer "OTP Auth" which uses iCloud synchronization. As I scroll through all of the accounts I have registered in OTP Auth, it's comforting to see that I have that additional layer of protection beyond what was contained in my LastPass vault.

But this leaves at theoretical risk any crucial high-value credentials that might have been saved in that vault which are **not** protected by a second factor. Since LastPass was not encrypting our eMail addresses and website URLs, there was definite leakage of who we are and where we go, without any need to decrypt anything. This opens LastPass' users to the potential for potent phishing attacks by leveraging what might be learned from an examination of the unencrypted data contained in their vault. So everyone needs to be on heightened alert for convincing-looking online scams sent to your eMail addresses and referring to websites you use, which I assume we're all going to see once our LastPass vaults are de-obfuscated.

The final takeaway is that if you are concerned that your LastPass master password was not high-quality, high-entropy and long at the time of its theft from LastPass, the risk of brute forcing might be higher for you. So it might be worthwhile for you to take the time to scan through your vault, after importing it into your next password manager, and manually changing the login passwords of any of your important accounts which are not also protected by some form of strong second factor authentication.

The last page of today's show notes contains a link appendix for anyone who's interested in reading the original source material that I found and shared above:

Jeremi Gosney's rant:
https://infosec.exchange/@epixoip/109585049354200263

Wladimir Palant's four recent LastPass blog posts:
https://palant.info/2022/12/23/lastpass-has-been-breached-what-now/
https://palant.info/2022/12/24/what-data-does-lastpass-encrypt/
https://palant.info/2022/12/26/whats-in-a-pr-statement-lastpass-breach-explained/
https://palant.info/2022/12/28/lastpass-breach-the-significance-of-these-password-iterations/

Steve Thomas: https://tobtu.com/
On Twitter: https://twitter.com/sc00bzT
And Mastodon: https://infosec.exchange/@sc00bz