

Security Now! #1066 - 02-2-26

Password Leakage

This week on Security Now!

- CA's warn us to urgently prepare for the inevitable.
- Three U.S. states attempt to ban 3D printed firearms.
- Denied ransom, ShinyHunters leaks 967,000 personal details.
- "Billions" of U.S. social security numbers leaked.
- Is Apple planning to add cameras to three new gadgets?
- No more security fixes for Firefox on Windows 7 & 8.
- Russia blocks the official Linux kernel site they need.
- Will the U.S. "freedom.gov" site post EU blocked content?
- LLM's will offer secure passwords. DO NOT USE THEM!
- As predicted, the "ClickFix" attack strategy takes over.
- A listener believes his computer is compromised.
- How could three popular password managers get things wrong?

"But Officer..."



Security News

Adding insult to injury

The subject line of the email I received from DigiCert Wednesday made me shake my head. Its subject was: "*Urgent: Revalidate domains expiring Feb 24 due to new 199-day validity requirement*" Here's what they wrote with great urgency:

Dear Valued Customer, We're reaching out with an urgent request to check your certificate domain validations before February 24, 2026.

As we communicated in previous emails, February 24 is the date when domain validation reuse periods will shorten to 199 days (down from 397 days) in accordance with the CA/Browser Forum's Ballot SC081v3. Our records indicate that you or your subaccounts have existing domains that will expire on February 24 because of this change. If your systems require immediate certificate issuance, your issuance could be delayed if you don't check and revalidate these domains before February 24.

What do you need to do? DigiCert CertCentral® now displays which of your current domains will expire on February 24, 2026, due to the change to 199-day domain validation reuse periods.

Steps for revalidating domains that expire February 24, 2026: In your CertCentral account, go to Certificates > Domains. Find the new "Validation expires SC-081" column next to the Validation expires column. Select domains that expire February 24, 2026, to open their Domain Details page. Scroll to the bottom of the Domain Details page and initiate revalidation by selecting your Domain control validation (DCV) method and clicking Submit.

We're here to help. We understand industry-driven compliance changes pose significant challenges and we're standing by to assist you. Please don't hesitate to contact your DigiCert account manager with any questions or concerns about the change to 199-day domain validation reuse periods:

Thank you for trusting us with your digital security. / DigiCert Team

Since DigiCert, a voting member of the CA/Browser forum, voted to bring about these changes, it seems odd for them to be sympathizing with their customers over the inconvenience that these changes create. To be straightforward, they would state that these changes have been made in the interest of improved security. We might disagree with that, as we know I do, but at least then they would be genuine.

What puzzled in their note, which I read closely, was their statement that "*Our records indicate that you or your subaccounts have existing domains that will expire on February 24 because of this change. If your systems require immediate certificate issuance, your issuance could be delayed if you don't check and revalidate these domains before February 24.*" As we know, it's never the case that anything is ever done that causes existing certificates to suddenly become invalid. It is always the case that the "not valid after" date continues to be honored. When you think about that, it's clear that a certificate that contains a built-in "not valid after" date means "valid until" and there's no way to "after the fact" have that no longer be true. This is why I went to all of the trouble last week to establish a new code signing relationship with IdenTrust while they were still allowed – by the CA/Browser forum – to issue three-year code signing certificates.

I am now the proud holder of a code signing certificate that will be valid until February of 2029, and nothing that happens between now and then, no matter what new insanity the CA/Browser forum may enact, will change that.

The answer to the mystery of what DigiCert means here is the phenomenon I spoke about last week, which is the new need to decouple certificate "qualification" from certificate "issuance". Since a certificate, once issued, will always live out the duration of its valid life unless it's revoked, back when certificates were issued with ten- or five-year lives, qualification for the certificate was determined at the time of certificate issuance, and that would be that.

But with the changes that are bringing about the shortening of certificate lifetimes, automation is effectively required. As we know, the industry intends to keep marching certificate lifetimes downward until they reach a maximum of 47 days. We're about to drop to 200. Then it will be 100, and finally 47. So the "issuance" of OV – organization validation – certificates needed to be decoupled from the qualification to receive OV certificates. Before the middle of next month, the CA/Browser forum would allow organizations to go up to 825 days, which is around 27 months, before they needed to be re-validated. But those 825 days now drops to 398 days, which is what DigiCert's letter was about:

They're saying that one or more of the validations they last performed of Gibson Research Corporation's identity occurred more than 398 days ago and until today – literally today – that was just fine with everyone, since those validations were more recent than 825 days ago. But it's now February 24th, so those older validations, which were just fine yesterday, are no longer recent enough for the CA/Browser forum which says that if we're going to issue any new certificates we're going to need to check you out again from scratch. So don't you go thinking that you'll just be able to push a button and issue yourself a new certificate. Oh, no. Your button has been disabled. You no longer qualify until we've had the chance to look you up and down again and make certain you're still you. And, what's more, from now on this will be an annual event. Oh yes. Those wild times where you could go 825 days between re-examinations? Those wild west days are over and they're never coming back.

This explains why, once my previously paid certification with DigiCert is over in two years I'll be happily dropping organization validation OV TLS certs in favor of the much cleaner and simpler DV domain validation certificates that Let's Encrypt's automation has been gleefully issuing to anyone and everyone who wishes to bring a server online.

"You want me to print what?!?"

We have another example of lawmakers apparently thinking *"We don't know how you techies are going to do it, but that's not our problem. We're going to make it a law so that it becomes your problem."* (And thanks to our listener Tom Minnick for bringing this to my attention.) Tom sent a link to the reporting on the well known and popular *"Adafruit"* website. Adafruit, for those who may not be aware, is a highly regarded hobbyist/maker/hardware electronics website & retailer. They posted the news of this new numbskull legislation under their headline: *"California's New Bill Requires DOJ-Approved 3D Printers That Report on Themselves."* Here's what they wrote:

*California's new bill requires DOJ-approved 3D printers that report on themselves targeting general-purpose machines. Assembly Member Bauer-Kahan introduced AB-2047, the **"California Firearm Printing Prevention Act,"** on February 17th. The bill would ban the sale or transfer of any 3D printer in California unless it appears on a state-maintained roster of pre-approved makes and models... certified by the U.S. Department of Justice as being equipped with "firearm blocking technology." Manufacturers would need to submit attestations*

for every make and model. The DOJ would publish a list. If your printer isn't on the list by March 1, 2029, it cannot be sold. In addition, knowingly disabling or circumventing the blocking software is a misdemeanor.

It gets worse. Much worse. Is everyone sitting down? Adafruit continues:

We've been tracking this pattern. Washington State's HB 2321 requires printers to include "blocking features" that can't be defeated by users with "significant technical skill" (good luck with that on open-source firmware). New York's budget bill S.9005 buries similar requirements in Part C, sweeping in CNC mills and anything capable of "subtractive manufacturing." California's version adds a certification bureaucracy on top: state-approved algorithms, state-approved software control processes, state-approved printer models, quarterly list updates, and civil penalties up to \$25,000 per violation.

As Michael Weinberg wrote after the New York and Washington proposals dropped... accurately identifying gun parts from geometry alone is incredibly difficult. Desktop printers lack the processing power to run this kind of analysis, and the open-source firmware that runs most machines makes any blocking requirement trivially easy to bypass.

I'll interrupt to note that, once again, when printers that can print weaponry are outlawed, only outlaws who wish to print weaponry will own outlaw printers. Nothing will be accomplished to curtail the fact that a 3D printer can be used to print a dangerous machine. Adafruit continues:

The Firearms Policy Coalition flagged AB-2047 on X, and the reactions tell you everything. Jon Lareau called it "stupidity on steroids," pointing out that a simple spring-shaped part has no way of revealing its intended use. The Foundry put it plainly: "Regulating general-purpose machines is another. AB-2047 would require 3D printers to run state-approved surveillance software and criminalize modifying your own hardware."

As we've said before on this blog, when we covered Washington and New York, it doesn't matter if you're pro- or anti-gun. The state should prosecute people who make illegal things, not add useless surveillance software to every tool in every classroom, library, and garage in the state. And as you can see, these bills spread—that's how a small group can push legislation into the entire country. First, Washington proposed theirs, then New York, now California. Once those three states pass a law, that's 20~25% of the country by GDP/population and thus every manufacturer is forced to comply with a bad decision in order to stay in business. If you're a maker, educator, or manufacturer anywhere in the US, even outside these states, this is a problem. It's a problem now.

<https://blog.adafruit.com/2026/02/19/californias-new-bill-requires-doj-approved-3d-printers-tha-t-report-on-themselves/>

Adafruit's article mentioned Michael Weinberg. Michael is the Executive Director of NYU's Engelberg Center for Innovation Law and Policy. He's a board member of the Open Source Hardware Association, and – as he describes himself – a maker of poorly made things. He's also an astute thinker, and since I think this topic is extremely interesting and that our listeners are also likely to find it so, I wanted to also share what Michael wrote in the wake of the New York and Washington state bills. The title of Michael's blog posting was: *"3D Printers Cannot Effectively Screen for Gun Parts"*. He wrote:

This post is a handy reference for the technical reasons why requiring 3D printers to screen for gun parts is not an effective way to reduce guns or gun violence. I am publishing it on the occasion of both New York and Washington State introducing bills to require this type of screening. In addition to a topic I have been researching for over a decade, the question of how to know if a 3D printer is printing a gun part is something I've spent a lot of time working on while overseeing trust and safety at a large 3D printing service provider.

This post is not about debating the larger legitimacy of gun control. In order to focus on the technical reasons why requiring 3D printers to identify and refuse to print gun parts does not work, it assumes that gun control is a reasonable and legitimate action of governments. Broadly speaking, it is responding to requirements that all 3D printers check prints to make sure that they are not gun parts. If the part is a gun part, the printer would refuse to print it.

The short version is that accurately identifying gun parts is incredibly difficult, and the hackable nature of desktop 3D printers makes it trivial to circumvent any requirements to try.

Here's the slightly longer version: Matching Files is Fragile

The first reason that requiring 3D printers to identify gun parts is ineffective is because analyzing 3D files is complicated. Any attempt to identify gun parts will miss many parts that are actually for guns, and may flag a number of parts that have nothing to do with guns.

Expensive engineering design software is good at evaluating specific properties of a 3D file (like where mechanical stress will occur over a lifetime of use). However, even that software cannot tell you what a part actually does (is that spring for a door, a shock absorber, or a catapult?). This challenge is exacerbated by the fact that guns are just mechanical objects. That means that there are many ways to design any individual part, and many individual parts of guns will resemble mechanical parts with totally benign uses. Put another way, devoid of other context, a switch for a gun safety looks a lot like a switch for a door.

Broadly speaking, there are two ways to think about doing file matching: Algorithmic Analysis This approach imagines a piece of software that can analyze a file and determine with some level of certainty if it is a gun part or just a hinge. Assuming that this software exists, which it does not at the time of this writing, it is reasonable to expect that such an analysis would be reasonably computationally expensive.

3D printers do not have the on-board processing power to do this kind of analysis. Requiring that they include chips capable of this kind of analysis would fundamentally change the economics of 3D printer design, akin to requiring that all bikes include jet engines.

I'll interrupt Michael for a moment to comment that he's not exaggerating how totally inadequate any 3D printer is for performing any sort of complex analysis. 3D printers are extremely simple and inexpensive. They have nearly no brainpower themselves. They're extremely simple robots that read instructions from a USB stick or SD card. There are some that fix a liquid resin using an image and others that move a plastic extruder around in 3-space, saying "*move the plastic extrusion head from where it is now to coordinates X, Y & Z.*" The resin-fix images, or the instructions with their coordinates, were created outside the printer by a real computer that's running some sort of engineering, drawing, design software. Once the design is ready, it's converted into fabrication instructions that are typically written to a storage device and transferred to the standalone printer which simply follows the instructions step-by-step. These printers are inexpensive because they could hardly be any more rudimentary. Michael continues:

Of course, the 3D printer could upload the file to a cloud somewhere and let the processing happen there. However, internet connectivity is not a default feature on desktop 3D printers. You could require that all 3D printers maintain a constant connection to the internet in order to operate but, again, that would fundamentally change how many people use their printer. There are also many legitimate use environments where constant internet connectivity is neither possible nor desirable. And, of course, this raises the question of who is responsible for maintaining that directory and keeping it secure.

What about Blacklisting? If it's not possible to analyze the true purpose of each file, it might be possible to at least match them against a known database of gun parts, right? This approach also has some serious shortcomings. First, there is a question of keeping that database up to date on the printer. That would require constant, or at least regular, internet connectivity for the printer. That raises the same issues as discussed in the last section. Second, also as discussed above, analyzing and matching 3D files is computationally expensive. The most logical way to do that with the processing power of a 3D printer would be to use a hash table of known gun parts, comparing a hash of the file to be printed against the table.

The primary problem with both geometry matching and hash matching is that it is incredibly fragile. The smallest change that had no impact on the functioning of the part would change its geometry and hash, effectively hiding it from the blacklist. That would make it trivial for anyone to circumvent. Identifying which changes are functional and which are merely aesthetic is not easy. That is especially true if people are making those changes with the specific goal of tricking the printer into printing a gun part.

3D Printers Print Themselves: The second reason this proposal is ineffective is because 3D printers are made in an incredibly distributed way. There are dozens of ways to make your own 3D printer using open source, user-modifiable parts. Even non-open source printers are highly hackable. As a result, there is no way to mandate that a technology that starts in a 3D printer remains in a 3D printer. The software that runs most printers is open source, meaning a single update would circumvent any screening measures.

This places 3D printers at the opposite end of the spectrum from 2D printers. Anti-counterfeit systems prevent 2D printers from printing currency. To the extent that these rules are effective (and I'm no expert, but they are often cited in these discussions as successful models), it is because the 2D printing industry is fairly concentrated and proprietary. 2D printer companies are actively hostile to users who want to modify their products, significantly raising the barrier to hacking around any countermeasures.

Desktop 3D printers are the opposite. They all trace their heritage back to open source printers, and users expect to be able to modify, extend, and hack their printers. That means that workarounds for a screening mandate would be easy to develop, distribute, and implement. Many open source software packages might even include the circumvention by default, meaning users would implement it without even actively intending to do so.

3D Printers are General Purpose Machines: This post is focused on the technical challenges with requiring 3D printers to screen every file it prints for gun parts. Nonetheless, it would be incomplete without a brief mention of how potentially invasive this sort of requirement is. 3D printers are general purpose machines that can be used for good or ill. Just as we do not require the phone company to monitor every phone call in order to prevent customers from using phones to commit bank fraud, we should be wary of requiring our 3D printers to monitor every print in order to prevent one possible type of print.

That type of invasion might be reasonable if it was effective. However, for the reasons described, it is unlikely to prevent even a modestly motivated person from using their printer to create gun parts. If an intervention is both highly invasive and unlikely to be effective, it is probably not an ideal policy.

Okay. So Michael did a great job of detailing the specific 3D printing issues which would surround any attempt to manage or control what a 3D printer can and cannot print. And while I have no interest in ever owning or printing a firearm, I'm a proud Californian and I'm annoyed by the fact that the state I love is enacting such moronic legislation.

But the broader concern is the large and growing degree to which modern technology appears to be outpacing the legislators' ability to understand what they can and cannot have. They cannot have a practical law to force 3D printers not to print gun parts. They just can't; no matter how much they may want it. They also cannot have a law that absolutely preserves everyone's privacy while at the same time preventing child predators from abusing that privacy to commit their crimes. We all wish it were possible to have both, but we know it's not.

I read some of the proposed new California Bill AB-2047. It's really quite awful. I have the link to the Bill's full text in the show notes for anyone who may be curious:

https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=202520260AB2047

A bad law that hits the books can usually be challenged by those who have a vested interest in the preservation of the status quo. But in the case of 3D printing, which is mostly a hobby interest, it's unclear who might have the deep pockets required to stand up against it and fight it out in court. If that doesn't happen it might be that the sale or transfer of 3D printers would be outlawed in those states which enact these dumb laws. Here's two lines from California's proposed legislation:

The bill, beginning on March 1, 2029, would prohibit the sale or transfer of 3-dimensional printers that are not equipped with firearm blocking technology and that are not listed on the department's list of manufacturers with a certificate of compliance verification, except as specified. The bill would authorize a civil action to be brought against a person who sells, offers to sell, or transfers a printer without the firearm blocking technology.

We're approaching March 1st of 2026. So the purchase ban, if saner heads do not prevail and it does come into effect, is still three years away. This means that, no matter what, it will be possible for Californians to purchase a 3D printer for the next three years. If you live in an affected state, currently New York, Washington or California, and you've been thinking that you might like to explore 3D printing in your garage, keep an eye on the date. There may be a deadline for doing that.

967,200 Users' Private Data Exposed

Friday before last, under their headline "Fintech lending giant Figure confirms data breach" TechCrunch reported:

Figure Technology, a blockchain-based lending company, confirmed it experienced a data breach. On Friday, Figure spokesperson Alethea Jadick told TechCrunch in a statement that the breach originated when an employee was tricked with a social engineering attack that allowed the hackers to steal "a limited number of files." The statement said the company is

communicating "with partners and those impacted," and offering free credit monitoring "to all individuals who receive a notice." Figure's spokesperson did not respond to a series of specific questions about the breach.

The hacking group ShinyHunters took responsibility for the hack on its official dark web leak website, saying that after the company refused to pay a ransom they published 2.5 gigabytes of allegedly stolen data. TechCrunch saw a portion of the data, which included customers' full names, home addresses, dates of birth, and phone numbers.

A member of ShinyHunters told TechCrunch that Figure was among the victims of a hacking campaign that targeted customers who rely on the single sign-on provider Okta. Other victims of the campaign include Harvard University and the University of Pennsylvania (UPenn).

Okay. So first of all, I loved the quote from their spokesperson "a limited number of files." Right. Who cares how many files escaped? It's size that matters and 2.5 gigabytes of customer personal data can do plenty of damage even if it's all contained within a single file.

Then, last Wednesday, Troy Hunt's Have I Been Pwned site scooped up the deliberately posted leaked breach data and examined what had been exposed. 967,200 – so nearly one million – of Figure Technology's customers all had their names, physical addresses, dates of birth, email addresses and phone numbers released after Figure refused to pay up. We know that not paying is the right thing for them to do. But it makes you wonder what the ShinyHunters group are thinking. Presumably as a result of asking for too much, they got nothing for their efforts. As I've noted recently, they have zero interest in any of this data. Its **only** value to them is the value that Figure may place on keeping it private. And once Figure said "no deal" they had to release it otherwise their threat to do so would be meaningless. That means they're unable to resell the data since it's now freely available on the Internet. And we've seen reports that, in general, more victims are deciding to take it on the chin and declining to pay. Partly, this may be due to the fact that being attacked and extorted is no longer a shocking announcement to make. It's certainly no badge of honor. But neither is it impossible to recover from. Just say "oops!", apologize to your affected customers, offer them a free year of credit monitoring and get on with business as usual. Leo and I both discovered that through no fault of ours, all of our data, including our social security numbers, was already out there swimming in that big Internet ocean.

But ShinyHunter's failure to obtain anything of value suggests that perhaps the value of stolen data property is falling and that if they don't wish to come up empty they may need to drop the size of their "ask".

Speaking of leaked social security numbers

Last Wednesday, UpGuard posted a curious headline: "Social Insecurity: Billions of Social Security Numbers and Passwords." Now, at first "Billions" seems really bad, right? But given that Social Security Numbers are specific to the United States whose current population is around 342 million, and that a grand total of around 450 million social security numbers have (ever) been issued since 1936, the claim of "2.7 billion" social security numbers seems somewhat sketchy. But their posting explains what's going on, and it seems legit. They wrote:

The week of January 12, 2026, the UpGuard Research team detected an exposed Elastic database with around 3 billion email addresses and passwords, and 2.7 billion records with Social Security numbers. That amount of data suggests it was created by recombining prior

SSN breaches like the OPM breach in 2015 or the National Public Data breach in 2024.

On the other hand, if even a fraction of the records were real—if only 10%, or 270 million records, or even 1% were real—the exposure would be a dire bellwether for the state of privacy in America. With the help of some unfortunate friends, we were able to confirm that at least some of it was real. And with the help of K-pop and some American presidents, we were able to approximate when the passwords were collected.

Discovery and Attribution

While most exposed databases require investigation to determine if they contain sensitive data, this one was obvious. The database had one index named "ssn" and another named "ssn2," each containing millions of records with nine-digit numbers in a field labeled "ssn". The database also had several indices that were collections of emails and associated passwords,

On January 16, we submitted the IP address and explanation of the issue to the FBI's IC3. We also submitted an abuse report to Hetzner, the hosting company. They replied saying they would forward the issue to the customer. After we clarified that their customer was in gross violation of privacy laws, all public access to the database was removed on January 21. Hetzner replied once more:

Dear Sir or Madam,

Thank you for your report. This is our customers statement:

Hello, we contacted our client and explained what ssn database hosting not acceptable.

Client now deleted this file from server.

So, problem solver for now.

UpGuard's report continues. They poke around inside their database locating some people they know closely enough to confirm their social security numbers. The data is authentic.

Unfortunately, since one of them whose data is present in the data happened to also have had her identity stolen in the past, they draw the entirely unwarranted conclusion that this means that this breach was the source of the identity theft. As we know, there's really not much personally identifiable information that is NOT by now loose in the Internet and available online.

I wanted to share this story to drive home the point that there has been so much prior leakage of our personal data that we really have very little to no control over it any longer. That's all just an illusion. It's certainly the case that the use of services like one of TWiT's sponsors "[Delete.Me](#)" makes a lot of sense for anyone who wishes to be as proactive as possible. But my feeling is that beyond that, doing everything that's within our power to minimize the **impact** of any personal data loss that has almost certainly occurred makes the most sense.

These guys did recognize that the database appeared to contain a great deal of redundancy and also a fair amount of incorrect noise. So it wasn't of the highest quality, which is what we'd assume when we learn that it was "billions" of records containing social security numbers.

At this point, protecting ourselves is the best we can do. I assumed that the GRC shortcut I would have previously created would be [GRC.sc/credit](#) and, sure enough, that bounced me directly to the Investopedia page which links to the three main credit bureaus. Any time you are not actively needing to have your credit queried, the best advice would be to keep it frozen.

Three new Apple AI gadgets with cameras?

Apple watcher and insider, Mark Gurman, has reported that Apple is believed to be working on a smart pendant, smart glasses, and new AI-based AirPods. And that all products will be equipped with a camera that will feed data into an AI system. At this time it's unclear what the AI will be doing for its user. And this does seem like an odd thing for Apple to be doing since people almost universally object to being surreptitiously recorded. Of course, Apple also famously invested about a decade working on the very poorly kept secret Project Titan – The Apple Auto. That never turned into a product. So it might be that these various camera-equipped gadgets will have a similar fate.

Firefox 115 (ESR) to be the last release for Windows 7, 8 & 8.1

Anyone who has continued to use Firefox on a Windows 7 or 8 machine will not receive security updates after this month. This month is it. Firefox support officially ended three years ago in January of 2023 but security fixes have continued. Those now end.

Oopsie! (Russia blocks their own Kernel source)

Roskomnadzor apparently got a bit trigger happy recently. As part of its recent accelerated Internet crackdown, which we've been following, this time it appears that Russia's Internet watchdog accidentally blocked the official website of the Linux kernel. The block was quickly lifted after upset Russian IT engineers reminded Roskomnadzor that all of the country's native OS distros run on Linux.

Does the U.S. really plan to host UK banned content in the UK?

Get a load of the first line of this reporting from Reuters. It reads:

WASHINGTON, Feb 18 (Reuters) - The U.S. State Department is developing an online portal that will enable people in Europe and elsewhere to see content banned by their governments including alleged hate speech and terrorist propaganda, a move Washington views as a way to counter censorship, three sources familiar with the plan said.

Okay, what?! Triple-sourced reporting says that the U.S. is planning to do what exactly? And looking at the nascent website: <https://freedom.gov/> Something does appear to be afoot. Last Wednesday, Reuters reported:

The site will be hosted at "freedom.gov," the sources said. One source said officials had discussed including a virtual private network function to make a user's traffic appear to originate in the U.S. and added that user activity on the site will not be tracked.

Headed by Undersecretary for Public Diplomacy Sarah Rogers, the project was expected to be unveiled at last week's Munich Security Conference but was delayed, the sources said. Reuters could not determine why the launch did not happen, but some State Department officials, including lawyers, have raised concerns about the plan, [imagine that] two of the sources said, without detailing the concerns.

The project could further strain ties between the Trump administration and traditional U.S. allies in Europe, already heightened by disputes over trade, Russia's war in Ukraine and President Donald Trump's push to assert control over Greenland.

The portal could also put Washington in the unfamiliar position of appearing to encourage citizens to flout local laws. In a statement to Reuters, a State Department spokesperson said the U.S. government does not have a censorship-circumvention program specific to Europe but added: "Digital freedom is a priority for the State Department, however, and that includes the proliferation of privacy and censorship-circumvention technologies like VPNs."

The spokesperson denied any announcement had been delayed and said it was inaccurate that State Department lawyers had raised concerns. The Trump administration has made free speech, particularly what it sees as the stifling of conservative voices online, a focus of its foreign policy including in Europe and in Brazil. Europe's approach to free speech differs from the U.S., where the Constitution protects virtually all expression. The European Union's limits grew from efforts to fight any resurgence of extremist propaganda that fueled Nazism including its vilification of Jews, foreigners and minorities. U.S. officials have denounced EU policies who they say are suppressing right-wing politicians, including in Romania, Germany and France, and have claimed rules like the EU's Digital Services Act and Britain's Online Safety Act limit free speech.

The EU delegation in Washington, which acts like an embassy for the 27-country bloc, did not immediately respond to a request for comment about the U.S. plan. In rules that fall most heavily on social media sites and large platforms like Meta's, Facebook and X, the EU restricts the availability — and in some cases requires rapid removal — of content classified as illegal hate speech, terrorist propaganda or harmful disinformation under a group of rules, laws and decisions since 2008.

Rogers of the State Department has emerged as an outspoken advocate of the Trump administration position on EU content policies. She has visited more than half a dozen European countries since taking office in October and met with representatives of right-wing groups that the administration says are being oppressed. The department did not make Rogers available for an interview.

In a National Security Strategy published in December, the Trump administration warned that Europe faced "civilisational erasure" because of its migration policies. It said the U.S. would prioritize "cultivating resistance to Europe's current trajectory within European nations." EU regulators regularly require U.S.-based sites to remove content and can impose bans as a measure of last resort. X, which is owned by Trump ally Elon Musk, was hit with a 120 million-euro fine in December for noncompliance.

Of course, last week we noted that of the \$2.4 billion euros in fines \$2.2 remained unpaid. So I doubt that Elon's strategies are much affected. Reuters continues:

Germany, for example, in 2024 issued 482 removal orders for material it deemed supported or incited terrorism and forced providers to take down 16,771 pieces of content. Similarly, Meta's oversight board in 2024 ordered the removal of a Polish political party's posts that used a racial slur and depicted immigrants as rapists, a content category EU law treats as illegal hate speech.

Calling the U.S. plan "a direct shot" at European rules and laws, former State Department official Kenneth Propp, who worked on European digital regulations and is now at the Atlantic Council's Europe Center, said [freedom.gov](https://www.freedom.gov) "would be perceived in Europe as a U.S. effort to frustrate national law provisions."

Also involved in the U.S. portal effort is Edward Coristine, a former member of Musk's job-slashing Department of Government Efficiency, two sources said. Coristine works with the National Design Studio, created by Trump to beautify government websites. Reuters was unable to reach Coristine for comment. It was not clear what advantages the U.S. government portal would offer users that are not available from commercial VPNs. The web address freedom.gov was registered on January 12, according to the federal registry get.gov. On Wednesday, the site had no content but showed the National Design Studio's logo, the words "fly, eagle, fly" and a log-in form. Before Trump's second term, the U.S. government helped fund commercial VPNs and other tools as part of efforts to promote democracy globally and help users access free information in China, Iran, Russia, Belarus, Cuba, Myanmar and other countries.

So it's going to be interesting to see what happens next. The site does not currently show me what's described. But I'm reaching the site from Southern California. So it will be interesting to learn what our European and British listeners see when they go to <https://freedom.gov/>. One thing I can assert with clarity is that under this country's current administration, there's never a dull moment ... even when we might wish for one!

DO NOT use an LLM to generate your passwords!

I hope I don't need to tell anyone listening not to ever, ever, use an LLM to directly generate a password. In other words, never ask an LLM for a password. Never say: *"Could you please generate a highly secure long password with 20 characters of all kinds including a mixture of upper and lowercase alphabetic, numbers and special characters?"* No. Don't do that. What you get will look wonderfully strong, but the LLM is quite likely to give the same password to others.

We've spent so much time through the years on this podcast examining just how very difficult it is to generate and obtain high-quality passwords, that the idea of asking a parrot for a password is almost painfully bad. Having apparently run out of useful things to explore, the site "Irregular" gave their detailed in-depth exploration of LLM password generation the headline: *"Vibe Password Generation: Predictable by Design"* Well, at least they got that right.

This is so nuts that I'm not going to spend much time on it. Their posting is long and thorough. But they were kind enough to lead with an Executive summary of their findings. They wrote:

- LLM-generated passwords (generated directly by the LLM, rather than by an agent using a tool) appear strong, but are fundamentally insecure, because LLMs are designed to predict tokens – the opposite of securely and uniformly sampling random characters.
- Despite this, LLM-generated passwords appear in the real world – used by real users, and invisibly chosen by coding agents as part of code development tasks, instead of relying on traditional secure password generation methods.
- We've tested state-of-the-art models and agents, and analyzed the strength of the passwords they generate. Our results include predictable patterns in password characters, repeated passwords, and passwords that are much weaker than they seem, as described in detail in this publication.
- We recommend that users avoid using passwords generated by LLMs, that developers direct coding agents to use secure password generation methods when needed, and that AI labs train their models and direct their coding agents to prefer secure password generation out of the box.

“ClickFix” attacks take over

I’ve been saying recently that the technique of asking a user to authenticate themselves by pressing the Windows + R key to open the Windows Run dialog, then press Ctrl+V to paste followed by the Enter key – terrifies me because it is so powerful and potent and because I could see so many people falling for it because most people have very little idea how their computers operate. They just follow instructions.

This highly potent form of attack has been dubbed “ClickFix”. Recall that I recently shared exactly such a pop-up that one of our listeners had encountered. And that sent me off on a rant about how irresponsible I felt Microsoft was being about not tracking the source of anything pasted into the system’s global clipboard. A web browser is a very clear security boundary with all manner of creepy crawly things clamoring to escape. So it should be utterly impossible for automation in the browser to place anything on the system’s global clipboard that can then be pasted outside of the browser’s security perimeter, specifically into the Windows Run dialog.

Seeing how obviously dangerous and effective this form of attack promised to be, I wasn’t surprised to read the report that Huntress Labs published last Tuesday.

Huntress set the stage for their lengthy report by writing:

Columbia, Maryland – February 17, 2026 — Cybercrime has become the world’s third-largest economy, with costs projected to reach \$12.2 trillion annually by 2031. Today, Huntress exposes the tactics, techniques, and procedures (TTPs) fueling this multi-trillion-dollar illicit market in its 2026 Cyber Threat Report. The in-depth analysis sheds light on the playbook used by organized, profit-driven cybercriminals, uncovering how they weaponize legitimate tools, exploit everyday behaviors, and leverage a vast underground network to exploit people, businesses, and employees across the globe.

To produce this report, Huntress analyzed proprietary telemetry from over four million endpoints and nine million identities across the 230,000+ organizations it protects worldwide. This robust dataset served as the foundation for uncovering critical insights into the evolving ransomware ecosystem, shifting adversary tradecraft, and actionable strategies to help organizations prepare for the year ahead.

Then, under the topic of “Key Findings Include” the item that caught my eye was:

Over half of all malware loader activity came from ClickFix: *In 2025, attackers didn’t need to break in when they could just trick users into giving them access. No technique did this more effectively than **ClickFix**, which fueled 53% of all malware loader activity. By masquerading as routine tasks, like solving a CAPTCHA, ClickFix and its variants tricked users into becoming unwitting accomplices, facilitating the silent installation of infostealers, ransomware, and remote access tools.*

I’ve specifically and explicitly reached out to many of my friends to warn them of this attack. It’s just too diabolical and too likely to succeed. One of my friends who works for a large non-profit charitable organization receives regular employee-level security training. When I told her about this she commented that they had never been warned about this type of attack. There’s likely a delay between the growth of an attack and its inclusion in training. But that leaves a very dangerous gap and, as Huntress found from their analysis, 53% of all successful breaches are attributable to the success of these ClickFix attacks. **PLEASE warn your friends to be careful!**

Listener Feedback

Doug Smith

I've enjoyed you touching on AI coding topics over the last few episodes of the podcast. Although the capabilities of AISLE that you covered recently sound fantastic, they aren't available to everyone yet. However, some aspects are available in other forms.

*For example, Claude Code has a built-in **/Security-Review** command that does a really great job. Although it's good at checking the latest changes before a git commit and push, I've taken to using it for things like checking WordPress plugins before installing them on my sites. In one case, this turned up multiple severe security issues in a plugin for connecting to a specific service I required. I was able to present the results and a working test exploit to the vendor and work with them toward fixes.*

*I also saw that Anthropic has a new **Claude Code Security** feature in limited testing right now that looks like it will continue to move security reviews significantly forward.*

*You recently suggested that the way to work with AI coding might be with test-driven development. That, and more, is exactly what the **Superpowers** add-on for Claude Code does that Leo mentioned a few episodes ago. It forces good planning, test-driven development, and code reviews by multiple AI agents, each with particular specialties. Here is the description of the workflow from the GitHub readme:*

brainstorming - Activates before writing code. Refines rough ideas through questions, explores alternatives, presents design in sections for validation. Saves design document.

using-git-worktrees - Activates after design approval. Creates isolated workspace on new branch, runs project setup, verifies clean test baseline.

writing-plans - Activates with approved design. Breaks work into bite-sized tasks (2-5 minutes each). Every task has exact file paths, complete code, verification steps.

subagent-driven-development or executing-plans - Activates with plan. Dispatches fresh subagent per task with two-stage review (spec compliance, then code quality), or executes in batches with human checkpoints.

test-driven-development - Activates during implementation. Enforces RED-GREEN-REFACTOR: write failing test, watch it fail, write minimal code, watch it pass, commit. Deletes code written before tests.

requesting-code-review - Activates between tasks. Reviews against plan, reports issues by severity. Critical issues block progress.

finishing-a-development-branch - Activates when tasks complete. Verifies tests, presents options (merge/PR/keep/discard), cleans up worktree.

The agent checks for relevant skills before any task. Mandatory workflows, not suggestions.

Thanks for your work that benefits me and so many others every week! /Doug

I wanted to share this because it so nicely chronicles the evolution that we're seeing in our understanding of how to employ AI. We've learned that as an AI's context window nears full the

hallucinations begin. So we work to prevent that. We've learned that rather than using a single AI and context window, we get far better results from using multiple AI agents, each with their own smaller context windows and thus each bringing their own perspective. And I have no doubt that a year from now we'll have learned much much more.

Eric

Hello Steve, I wanted to share an issue I recently ran into that makes me believe some malicious application has gotten onto my PC. I'd value your advice on whether I should completely reinstall Windows and all my applications. You're welcome to share this if you think others could benefit! Thanks for everything you do! Best Wishes, Eric Richardson.

DESCRIPTION OF THE ISSUE

Last week, while examining logs on NextDNS, I decided to download them for a better review of activity. Upon examining the logs, I saw many DNS queries for 26-character-long domains:

*xdu1xjw0lnfppq4xdtoz1brlh.com
204cyuhtyp0ooyxey3so4gdzha.com
1sumk15wu3h4xkycmztiml2jpf.com
en4cftqucyy1ppnokmrmxlynte.com
...*

These DNS queries only came from my PC. My wife's and daughter's laptops weren't affected.

I looked up some of the domains on ICANN's DNS lookup, but found no entries. Google confirmed my suspicion that these lookups were likely malicious: DGA (Domain Generation Algorithm) activity. I checked the entries in my NextDNS logs and noticed these queries were NOT blocked! I confirmed that Domain Generation Algorithms (DGAs) Protection was enabled, so I don't know why the query would not have been blocked.

Okay. So far I'm reading along and looking at Eric's evidence and I'm in complete agreement with everything he's seeing. I'm thinking that, yeah... that really does look pretty bad and quite suspicious. Then I get to his next sentence:

Tracing entries in my NextDNS log, I see that queries to isc.org seem to precede queries to these 26-character domains. I also see several queries to rebindtest.com, which do appear to be blocked by NextDNS.

His mention of "isc.org" stopped me in my tracks because I suddenly knew exactly what was going on with Eric's machine and his NextDNS logs. And this was further confirmed by his mention of "rebindtest.com". Since Eric was understandably concerned and wondering whether he would need to wipe his machine and reinstall Windows I immediately wrote back, saying:

Oh Eric! That's the DNS Benchmark! Those are the queries generated by running the benchmark! Your machine is NOT infected. Instead you have great DNS! <g> The tip-off for me was your mention that queries to isc.org appear to precede them, and the clincher (though we already have sufficient evidence), was queries to rebindtest.com — which is my domain that I maintain for the Benchmark's use.

Eric replied:

Oh thank goodness! Thank you for replying so quickly!

The first thing the DNS Benchmark does, in the process that I can “characterizing” any DNS resolver, is to check whether it’s online at all by asking it for the IP of the “[isc.org](https://www.isc.org)” domain. ISC is the Internet Systems Consortium. The ISC has been around since 1994 and the birth of the Internet. I chose to have the DNS Benchmark check for a resolver’s online status by querying for the IP of “[isc.org](https://www.isc.org)” since even Roskomnadzor wouldn’t have any problem with [ISC.ORG](https://www.isc.org) nor feel any need to block it.

And those wacky 26-character-long dot com domains are randomly generated. Though not one of them will ever exist, that’s the point. They are, therefore, prevented from ever being in any DNS cache since none of them will have ever been seen before. More importantly, queries for the IP addresses of each of them will be guaranteed to generate an “NX DOMAIN” (non-existent domain) error status. The Benchmark absolutely knows that’s the result it’s going to obtain, but the resolver it’s asking – the one being tested – has no way of knowing that. So it must forward each and every one of those queries to the Internet’s upstream DOT COM servers. When the Dot Com nameserver receives the resolver’s query it’s going to think “what the heck are you talking about” and send back the expected non-existent domain reply... but it’s the length of time that’s required for us to receive that reply from the server being benchmarked that we care about. This tells us how well connected the resolver we’re testing is to the Internet’s dot com nameservers.

The ICANN registry shows that I registered [rebindtest.com](https://www.rebindtest.com) nearly 16 years ago in August Of 2010. It’s my own domain which returns IP addresses for the various private networks such as 10-dot and 192.168, etc. A DNS resolver should really never return a private network address for a publicly queried domain. We’ve talked about this in the past. Bad guys can use that to probe around inside a user’s local LAN. Their browser will believe that it’s connecting to the server at the domain “[tricky-bad-guy.com](https://www.tricky-bad-guy.com)”. But if DNS for “[tricky-bad-guy.com](https://www.tricky-bad-guy.com)” resolves to 192.168.0.1 then the browser may actually be connecting to the LAN’s internal gateway router. So this is just one of the many things the DNS Benchmark is able to show its users about the DNS resolvers they are currently using and others they might be considering switching to.

In any event, if anyone else might think to look at their DNS providers logs and see the sorts of admittedly suspicious DNS lookups that Eric spotted, if you DO NOT own and have run GRC’s DNS Benchmark, then you would certainly have cause for concern. But assuming that you’re an owner of my latest utility, you have no cause for concern.

Stephen Clarke-Willson

I was reading this ACM article and hit a paragraph that made me instantly think of you and “defaults”. The paragraph:

Before version 4.0.0 (published in 2017), Redis (red-diss), the extremely popular key-value store, offered no access controls in its default configuration. Frequently, new users of Redis would unintentionally expose their instance publicly, and this insecurity would result in data spills or become a vector for host exploitation. As of version 4.0.0, Redis enters a “protected mode” when run with its default configuration and without password protection. This limits

access to loopback interfaces. As the Redis company itself has since touted, the introduction of protected mode has caused the number of publicly accessible Redis instances tracked on Shodan.io, a popular internet host aggregator, to decline substantially. In 2017, it had identified roughly 17,000 exposed Redis instances; in 2020, that number had declined to 8,000 in an audit by security company TrendMicro. I like how simple the solution was - limit access to the loopback interface. Very nice.

Full article: <https://queue.acm.org/detail.cfm?id=3773095> - Stephen from Sammamish, WA

They've certainly improved the situation by dropping the clearly exposed instances to 47% of what they were. So at this point either those exposed REDIS key-value stores have been sitting there for the past 9 years since before version 4, or they were configured with some authentication and can therefore again be misconfigured. As we all know, authentication should never be depended upon to block malicious access and I believe that a misplaced reliance upon authentication and a lack of adoption of backup measures, such as never binding to a public-facing interface unless truly necessary, remains an easily remedied source of security failures.

It's also a shame that one of my favorite tricks has never been adopted: One of the most iron-clad rules of Internet routing is that any packet which is received by an Internet router will have its incoming TTL — Time To Live — value decremented. And if in doing so that value is decremented to zero, that packet will never be forwarded toward its destination. A router might simply drop the packet like a hot (or perhaps dead) potato, or it might elect to send an "ICMP Time Exceeded" message back to the packet's originator. If this rule were not absolutely obeyed by every router the Internet could conceivably "fill up" with zombie packets that live forever, refuse to die, and circulate.

So, as a security tool, if there was some need to expose a server to the Internet in, for example, some sort of cloud-hosted configuration, as a listener of our shared recently, and IF it were possible to set the TTL for that server's outbound packets to, say, 2, then any other nearby clients of that public server, for example, within the same cloud infrastructure, could connect to and use it without trouble, while at the same time no one in faraway China or North Korea could possibly get to it. Since a TCP connection requires round-trip verifications from each end, any of the packets sent from the server would die after two Internet router hops. No one probing that server would ever even be able to detect that its services were available.

Unfortunately, packet TTL has never been adopted as a security measure. It's considered to be part of deeper Internet infrastructure, thus not something to be messed with and not subject to application-level manipulation. As a result the interfaces for setting a connection's TTL are not commonly available to applications — even if they have any interest in employing them.

Password Leakage

Way back in the early days of this podcast we talked about the technology to securely backup and securely store our data in the cloud. Of course, back then, what we had were remote storage providers and clouds were white puffy things that slowly drifted across the sky. No one was calling anything and everything that was “remote” a cloud. But that was then and today everything is “in the cloud.”

At the time, I crystallized the concepts surrounding the only sort of encryption that made sense using the abbreviation TNO, which was short for Trust No One. This was repurposed from a prominent poster on the wall of the X-Files agent Fox Mulder. Mulder was famously paranoid. So a poster reminding him to Trust No One made sense. It also made sense for anyone who might be considering sending the personal and private contents of their PC off to a remote server.

The underlying concept behind TNO encryption was simplicity itself, which was part of its appeal. The idea was that any and all data that was going to be sent offsite would first be encrypted using a secret key which would never be shared so that all the remote storage provider would be receiving and storing would be a massive blob of pseudo-random data. As we know, regardless of what is fed into properly designed encryption, what emerges is indistinguishable from pseudo-random noise.

Later, we used another abbreviation, PIE to stand for Pre-Internet Encryption. The concept there was also simple. You would always encrypt anything you cared about before it ever left the domain of your machine to be sent out over the Internet.

Along the way we also examined the more technical details of how this should be done. We looked at the need for the user’s password to be strong, and at the use of PBKDF – Password Based Key Derivation Functions – to significantly impede the use of brute-force password cracking technologies and techniques.

What I want to point out is that all of this is extremely straightforward, simple to do and utterly bulletproof. It works and it works perfectly. Nothing we talked about back then was difficult to implement back then or now. So what’s the problem? How can today’s contemporary password managers, that all rightly require the most state-of-the-art security available, still be having trouble – today – with something as simple as those concepts of TNO and PIE?

That question has two answers: “Practicality” and “featuritis.” In the case of today’s password managers, it’s the need to go from a dead-simple, rudimentary and utterly secure system concept, which was what we had with our TNO/PIE, and evolve it into a workable and practical solution. Suddenly it’s not so simple.

For example, in the Pre-Internet Encryption Trust No One backup solution we discussed in the early days, what would happen if our user forgot his password? Trust No One cuts both ways. If you have truly trusted no one else with anything, then the well-known abbreviation that comes into play is “SOL”. It’s reminiscent of Leo’s Bitcoin wallet containing a now-valuable Bitcoin that’s protected by a long-forgotten password. The good news is it is super-secure and no one is going to open that wallet without its password, which is also the bad news, since that “no one” includes Leo.

So, what our original super-secure system is missing is any form of password recovery. Yes, this super simple system is completely secure... but it is also completely unforgiving. We know that any practical password manager for the masses **must** necessarily provide some means for

dealing with the inevitable “I forgot the password for my passwords.” But what’s also inevitable is that the moment we start adding such “get out of jail” features we invariably start chipping away at the pristine security we originally enjoyed. It’s exceedingly difficult to have it both ways.

There’s also the pressure to maintain feature parity among the competing managers by offering some form of “friends and family sharing.” And if all that wasn’t challenging enough, the password managers have also been confronted with rapidly evolving cryptographic cracking technology. This often requires backward compatibility with earlier releases. We saw LastPass stumble badly over this with the need to increase their client-side PBKDF iteration count while being reluctant to force their original users to keep up with the times. Every additional feature increases the complexity of the system; and we know that complexity is the enemy of security.

Today’s password managers are not only bristling with features, but they’re also under continual pressure to match each other’s features since many users will make their choice of password manager from a feature comparison grid while considering little else. All of this made password managers a terrific subject for the group of Swiss security researchers who decided to dig into the operation of three password managers to learn whether and to what degree the addition of all these bells and whistles may have come at the cost of their users’ security.

So here’s what the team wrote in the overview Abstract of their 28-page research findings paper:

Zero Knowledge Encryption is a term widely used by vendors of cloud-based password managers. Although it has no strict technical meaning, the term conveys the idea that the server, who stores encrypted password vaults on behalf of users, is unable to learn anything about the contents of those vaults. The security claims made by vendors imply that this should hold even if the server is fully malicious. This threat model is justified in practice by the high sensitivity of vault data, which makes password manager servers an attractive target for breaches (as evidenced by a history of attacks).

We examine the extent to which security against a fully malicious server holds true for three leading vendors who make the Zero Knowledge Encryption claim: Bitwarden, LastPass and Dashlane. Collectively, they have more than 60 million users and 23% market share. We present 12 distinct attacks against Bitwarden, 7 against LastPass and 6 against Dashlane. The attacks range in severity, from integrity violations of targeted user vaults to the complete compromise of all the vaults associated with an organisation. The majority of the attacks allow recovery of passwords. We have disclosed our findings to the vendors and remediation is underway.

Our attacks showcase the importance of considering the malicious server threat model for cloud-based password managers. Despite vendors’ attempts to achieve security in this setting, we uncover several common design anti-patterns and cryptographic misconceptions that resulted in vulnerabilities. We discuss possible mitigations and also reflect more broadly on what can be learned from our analysis by developers of end-to-end encrypted systems.

The “malicious server model” is the one we want. It’s the model that was explicit in our original foray into TNO. The “No One” who we were not trusting was the entity who was holding our encrypted back up data. Although all of the responsibility for not losing the decryption key was ours, in return for that responsibility we obtained the warranted guarantee of our invulnerability.

The beginning of their introduction sets the stage and also shares some additional statistics about the market share of the native built-in browser-based solutions. They write:

Despite the rise of alternative authentication methods, users today still have to deal with passwords, often numbering in the hundreds. Password managers help to tame the problem by providing a tool to securely store passwords, reducing the challenge of remembering many passwords to remembering just the one "master password" for the password manager. Cloud-based password managers outsource the storage to a remote server under the control of a service provider. At an abstract level, a user's passwords are collected in a single object which is then encrypted by the user's client under a cryptographic key derived from the user's master password, creating an encrypted vault. The client then uploads the encrypted vault to the server. When a user wishes to access a password for a particular service, their client authenticates to the service, retrieves the encrypted vault, and decrypts it locally with a user-provided copy of the master password. Importantly in solutions of this type, the service provider does not see the vault plaintext and therefore does not immediately learn the user's passwords or other sensitive data. This is akin to the situation with end-to-end encrypted (E2EE) cloud storage, and while the terms E2EE or client-side encryption are sometimes used by vendors in this space, the most commonly used term is Zero Knowledge Encryption. The term Zero Knowledge of course has a specific technical meaning in the context of interactive protocols, but here the term is being used with a different meaning, as we shall see. The cloud-based approach has multiple advantages: users can access their encrypted vaults from multiple devices; vaults can store other sensitive information beyond passwords (e.g. credit card data, personal documents); and the service can be extended to allow sharing of sensitive data within a family, group or organisation. The "access from anywhere" feature creates work for vendors, who have to support access from web browsers as well as stand-alone applications running on different OSes. Many vendors have offerings which allow the cloud storage element to be self-hosted by an organisation instead of by the vendor.

Three prominent providers in this space are Bitwarden, Dashlane and LastPass. At the time of writing, Bitwarden claims to have 10 million users, Dashlane 19 million users and 24,000 business customers, and LastPass 33 million users and 100,000 business customers. A 2024 report based on a survey of 1000 US consumers gives further insight into the popularity and market share of password managers. The built-in password managers of Google and Apple now represent 55% of the market, up from a combined share of only 15% in 2021. Bitwarden and LastPass are the next two largest, according to the study, with 11% and 10% market share, respectively. Dashlane now has only 2% market share, down from 7% in 2021 when it was amongst the market leaders. There is a long tail of smaller players in the market.

I thought it was interesting to see that the password managers built into Safari and Chrome are enjoying a 55% share of the market. That makes sense to me. While I require strong cross-platform support from my chosen password manager, my wife does not. She lives in Chrome on both her PC and her iPhone. I don't think she ever uses Safari and she despises Edge. So her needs are fully met without the use of any additional password manager. But I use many more features of my 3rd-party password manager and I cannot imagine operating without it.

Okay. So what did their detailed research reveal? They wrote:

We give a detailed analysis of Bitwarden, Dashlane and LastPass, presenting a cornucopia of practical attacks. In the artefacts that accompany our paper, we give Proof of Concept (PoC) implementations of all of these attacks, demonstrating their feasibility. The attacks allow us to downgrade security guarantees, violate security expectations, and even fully compromise users' accounts. We provide a table listing the various attacks and their impacts. Worryingly, the majority of the attacks allow recovery of passwords – the very thing that the password managers are meant to protect. We group the attacks into four categories: attacks exploiting

the key escrow features used for account recovery and SSO login, attacks based on lack of integrity of the vault as a whole, attacks enabled by the sharing features, and, finally, attacks exploiting backwards compatibility features. These attacks reveal common design anti-patterns and cryptographic misconceptions. Lack of authentication of public keys is widespread. When combined with key escrow and sharing features, this results in the adversary being able to fully compromise vaults. Another recurring failure mode is (wrongly) assuming origin-authentication of public key ciphertexts, leading to key substitution attacks against Bitwarden. LastPass stands out for lacking any form of ciphertext integrity, using AES-CBC as its main encryption mode.

Okay. By that they mean that LastPass is not authenticating its decrypted results. AES in CBC (Cipher Block Chaining) mode provides state of the art encryption, but after decrypting there's no means for authenticating the result. Our longtime listeners will recall the early days when we talked about the importance of authenticating and, assuming that decryption and authentication would be separate steps, the question was in which should they be performed?

Today, there are very good solutions for this. For SQRL's design I chose AES in GCM mode which is a lovely protocol that simultaneously provides encryption and authentication at the same time. But today, LastPass may be stuck in the past.

The researchers finish their introduction writing:

Thanks to legacy code and backwards compatibility exploits, we can downgrade Bitwarden and Dashlane to similarly hazardous states. We also show that integrity is only achieved for single fields in individual items, instead of at the vault level. This enables cut and paste attacks within items and across the vault. Such attacks can often be chained to compromise the confidentiality of the vault as well. These attacks work even when proper authenticated encryption is used. They are possible because of insufficient key separation in vaults with complex structures and/or a lack of cryptographic binding between data and metadata.

Okay. What all of that means is that no matter how much you may want to, and no matter how well intentioned you may be, it's just not possible to check your own work. It is truly necessary to have highly motivated, highly skilled and highly creative security researchers – who want to find problems and who have no ego stake in not finding any problems – scrutinizing these products that have become as complex and feature-laden as today's password managers.

I don't see any point in spending more time digging more deeply into the problems they found, since they've already been corrected. The problems were always predominantly theoretical since they depended upon some form of deep compromise of the provider's server-side infrastructure. But even those issues have now been addressed. To my mind, this is a classic case of "the safest security solution is the one that's been heavily challenged and audited by the industry's top security researchers. So I feel more confident than ever with my choice of Bitwarden as my password manager. All three of these password managers are better today as a result. And I should mention that the reason those three were chosen among many others was due to the availability of at least some client side source code being available.

