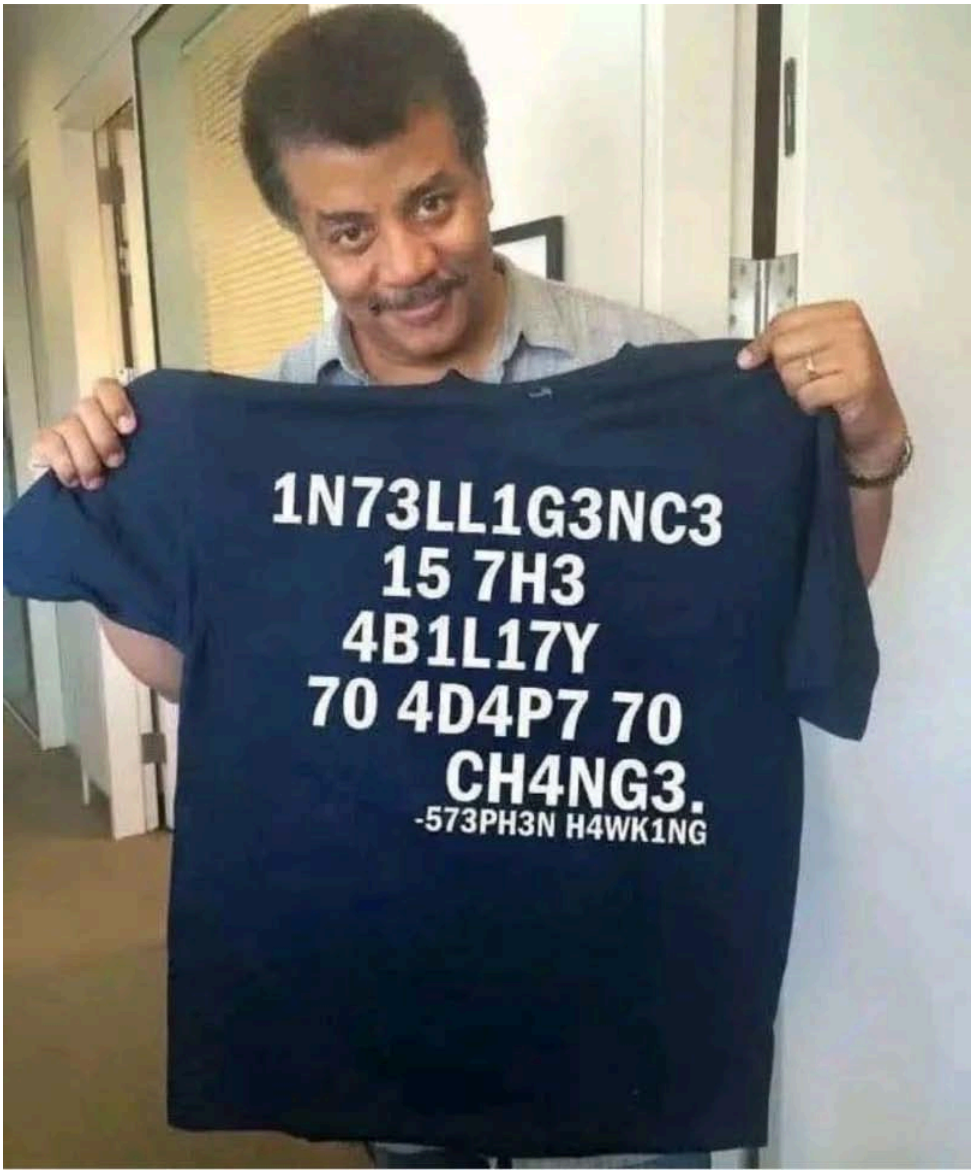
Security Now! #1021 - 04-15-25

Device Bound Session Credentials

This week on Security Now!

Android to get "Lockdown Mode". • What's in the new editions of Chrome and Firefox? Why did Apple silently reenable automatic updates? • My new iPhone 16, Chinese tariffs and electronics. • Dynamic "hotpatching" coming to Win11 Enterprise & Edu. • Why is it so difficult for Oracle to fess up? • Another multi-year breach inside US Treasury. • An Apple -vs- the UK update. • "Thundermail" (Can't someone come up with a better name?) • The (in)Security of Programmable Logic Controllers. • When LLM's write code and hallucinate non-existent packages. • Wordpress core security and PHP gets an important audit. • Device-Bound Session Credentials update session cookie technology.

Here’s one to think about...



Neil deGrasse Tyson shares a quote credited to Stephen Hawking

Security News

An Android "Lockdown" Mode?

Nothing has been announced yet, and it's certainly not official. But it would make sense for Android to follow in Apple's footsteps with a higher-security "lockdown" mode for Android. And with Google's annual I/O developer conference happening next month, it might be announced then, which might make it available in the August-September time frame as part of Android 16.

So it's believed that Google has been quietly working on a new more-secure mode for Android that was likely inspired by Apple's iPhone Lockdown Mode. According to a placeholder documentation page, which currently 404's, and based on analysis of Android beta images, the new feature would be named the Android Advanced Protection Mode (AAPM).

As with Lockdown Mode, AAPM would not be intended for regular Android users. It would be of use for probable target individuals who are more likely to face threats from oppressive regimes, advanced spyware, and network surveillance attacks. The feature is believed to:

- Disable older and less secure 2G cellular connections.
- Block users from sideloading apps from unknown sources.
- Enable "Memory Tagging Extension" to block the exploitation of memory-related exploits.
- And force a reboot of any devices after more than three days of disuse.

The forced reboot feature was spotted by Android Authority to flush any RAM resident malware that may have taken up residence in a device during its owner's absence.

Although Google has offered no official confirmation of any such new Android Advanced Protection Mode, a large amount of code to support it is present in Android 16 betas, which suggests that it may be made official soon. For example, Android Authority found the message that informs users they may not sideload apps. There's also support for a new API to allow apps to detect when the mode has been enabled on a device so that they may apply any of their own security-enhancing behavior. For example, a web browser might disable its internal Just-In-Time compilation when the mode is detected as active since we know that JIT has been a vulnerable point of attack in the past. Or, as another example, Instant Messaging apps might disable their automatic display of multimedia content since, again, we've seen security vulnerabilities discovered and leveraged in the interpreters that are used to display media.

So there are many signs that something resembling Apple's Lockdown Mode will be coming soon to Android. Like Lockdown Mode, it probably reduces a device's convenient functionality by too much to be used by most people. It would make the smartphone much less fun to use but also much more secure to use. So it's our well-understood tradeoff of convenience versus security.

Chrome: 135 / Firefox: 137

While I was perusing recent news I saw that Chrome had recently moved to release 135 and Firefox was now at 137. Among the changes in Chrome was the title of today's podcast, *"Device Bound Session Credentials"* which we'll be getting to during this week's deep technical dive. Nothing else really stood out about Chrome's 135 (though Device Bound Session Credentials is plenty).

The biggest news for Firefox 137 is reported to be Tab Groups, though the ability to use Firefox's URL field as an ad hoc calculator for quick math excites me more and I'm sure it's going to get much more use. Somehow, I've broken the habit of having a seemingly near infinite number of

tabs serving as place holders for things I plan to get back to eventually. But I know for a fact that many of our listeners are operating with many many hundreds of open tabs in their browsers. So for these people I would imagine that having the ability to group tabs might come in very handy.

The Firefox 137 blog page explains: *Tab groups begin rolling out today! Stay productive and organized with less effort by grouping related tabs together. One simple way to create a group is to drag a tab onto another, pause until you see a highlight, then drop to create the group. Groups can be named, color-coded, and are always saved. You can close a group and reopen it later.* So I thought "Great, let's try it!" ... but no matter what I tried, when I attempted to drag one tab on top of another, at some point, presumably once a center-line was crossed, the underneath fixed tab would suddenly scoot over to fill the gap that was left of the tab I was dragging. No matter what I did, I was unable to, in any way, merge two tabs into a single group.

Then I noticed that the phrase "Tab groups" was highlighted in the blog posting as a link. Clicking that I discovered the likely cause of my trouble. That more detailed page said: *"Starting in Firefox version 137, you can use tab groups to manage open tabs in Firefox by grouping them together and labelling them."* – Sure. Right. Except it's not working. Then it said: *"This feature is experimental and is being introduced to the Firefox user base through a progressive rollout. It may not yet be available to all users."* Oh. Okay. The Mozilla folks seem pretty excited about this and they also noted that Firefox's new Tab Grouping system also works for Vertical Tabs. I long ago satisfied my need for vertical tabs using a pair of add-ons: "Tree Style Tab" and "Tab Session Manager" which do everything I need. But once support for native Tab Groups does finally arrive in my Firefox I may take a look at switching to Firefox's native vertical tabs and using Tab groups. Mozilla's "Tab-Group" knowledge base page has a complete explainer for all of the various features of tab groups: Creating a tab group, Adding a tab to a group, Removing a tab from a group, Moving a tab from one group to another, and Managing tab groups. It's possible to name them and give them meaningful distinctive colors. So all of this seems like a useful new set of features for those who manage their days with browser tabs. For anyone who wants more information and details, (https://support.mozilla.org/en-US/kb/tab-groups) the link to the knowledge base page is in today's show notes.

But earlier I said that the feature that appealed to me the most was the ability to use Firefox's URL field as a quick ad hoc calculator... and even though that feature is also part of a progressive rollout, that one is alive and working for me already. It couldn't be any easier to use. Mozilla writes: *"You can now use the Firefox address bar as a calculator. Simply type an arithmetic expression and view the result in the address bar drop-down. Clicking on this result will copy it to your clipboard."* That's all there is to it and it's very slick. I'm often reaching for the calculator that's located next to me at my workspace. I'm sure that Mozilla's calculator will be limited to simple arithmetic expressions. But I'm not always at my workspace, either.

The integrated calculator appears to be part of a larger address bar refresh and update, even though it's listed on its own. But the address bar has a bunch of new behaviors. Mozilla explains:

- Unified Search Button: A new, easy-to-access button in the address bar helps you switch between search engines and search modes with ease. This feature brings the simplicity of mobile Firefox to your desktop experience.

- Search Term Persistence: Now when you refine a search in the address bar, the original term sticks around, making it easier to adjust your queries and find exactly what you're looking for.

- Contextual Search Mode: Firefox detects if you are on a page that has search capability and offers that option for you to directly search with the page engine from the address bar. Use this option at least 2 times and Firefox will suggest adding the search engine to your Firefox.

- Intuitive Search Keywords: You can access various address bar search modes with convenient and descriptive keywords (e.g. @bookmarks,@tabs,@history, @actions).

That "Contextual Search Mode" where Firefox is supposedly detecting pages which offer their own searches is surprising and seems both aggressive and error-prone. It'll be interesting to see how that works out.

Beyond all this, Firefox 137 now identifies all links within the PDFs it's integral PDF viewer displays and turns them into hyperlinks. It's also possible to add your signature to PDFs without leaving Firefox and signatures can be saved for reuse later, and Firefox now provides native support for the HEVC media format under Linux.

It occurs to me that all of this further supports my ongoing contention that our web browsers have become incredibly complex and only continue to become more so! And we're going to see yet another example of that at the end of today's podcast.

Apple silently turned on automatic updates
A posting over in OSXdaily had the headline of a Public Service Announcement, reading: *"PSA: Automatic Update Enables Itself with MacOS Sequoia 15.4 & iOS 18.4"* The posted piece said:

> *This is important and relevant to most Mac, iPhone, and iPad users: Installing the latest updates for MacOS Sequoia 15.4 for Mac, iOS 18.4 for iPhone, and iPadOS 18.4 for iPad, will forcibly enable automatic software update for system updates on your device.*

Given the fact that Updates can again be turned off, their use of the phrase "forcibly enable" seems unwarranted. That implies that it would no longer be possible to again disable automatic updates. Which is indeed possible. The piece continues:

> *Some people may already have these auto-update features enabled on their devices and not mind this change (nor would they notice a difference), whereas there are many other people who intentionally disable automatic update and do not wish to have the auto-update feature forced upon their devices.*

<Sigh> Okay.

> *With Automatic Updates enabled, this means your Mac, iPhone, or iPad, will automatically download and install system software updates onto your device(s) as they become available, without your approval or prompting.*
>
> *Automatic Updates can be problematic for many reasons. For one, not everyone has the bandwidth available to automatically download huge software updates. Additionally, not everyone wants to install the latest software updates when they become available, many users prefer to wait a little while to see if there are any critical bugs or issues discovered before putting the latest system software on their device (and this is a reasonable caution, though it's not common, Apple has dumped out some bad software updates in the past that had to be*

pulled due to various issues). And of course, many Mac, iPhone, and iPad users, just simply prefer to manually update and manage their devices on their own, without the computer or device doing it for them.

Then the piece goes a bit off the deep end by writing:

But your personal computing behaviors and your opinion is irrelevant, as Big Cupertino knows what is best for you, your iPhone, your Mac, and your iPad.

Right. And as we know, for the vast majority of their users they probably do know what's best.

Apple has decided that you will have automatic updates enabled on your devices, and your installation of iOS 18.4, MacOS Sequoia 15.4, or iPadOS 18.4 was apparently used as an agreement to that setting change. If you don't like that, you can change it back and disable automatic system software updates.

The rant continues at some length, but we don't learn anything more beyond the fact that this author really really dislikes the idea that Apple might feel that having automatic updates enabled for the masses is sufficiently important that it should be done. I can certainly agree that it would have been polite for Apple to ask before re-enabling disabled automatic updates, since if Apple were to find them disabled on a device it would have had to be deliberate. But perhaps there are instances where it could have been malicious.

In any event, since I know there are many listeners of this podcast who strongly prefer taking and having manual and deliberate control over the updating of anything, I wanted to make sure that everyone knew that the move to these latest macOS, iPhoneOS and iPadOS releases will have re-enabled any previously disabled automatic updates going forward. Note that Apple may have been worried that Sequoia 15.4 and the iOS 18.4 might have some problems, and so wanted to have the option of quickly fixing anything that might arise. My point is, we really don't know what thinking may have precipitated this at Apple... but there likely was some.

Steve's new iPhone 16 and Chinese tariffs

I'll also take a moment to note that I'm now the proud owner of a shiny new iPhone 16 Pro.

As I've mentioned before, I had been happily using an older iPhone 12 Pro without any problems. But I became concerned last week over the threat of Chinese import tariffs significantly inflating the price of iPhones. The threat appeared to be real with Apple in a panic, flying iPhones in from India. But after poking around Apple's site I decided that my older iPhone 12, which was still working just fine, would almost certainly last me through whatever tariff turbulence we were going to be experiencing for the next few years. I later mentioned this to my wife, Lorrie, whose response was *"My god, buy yourself a new phone! Yours is old and small!"* So last Thursday, I returned to the Apple store and did that.

As we know, I'm not someone who always needs to have the latest and greatest. My stash of Palm Pilots in the refrigerator is testament to that. I'm also a testament to the "if it's not broke don't fix it" school of thought. So I usually use electronics until they're worn right down to the nub. But I have to say that the 16 **is** a lot more responsive than the 12 was, and since I no longer wear a watch, every time I saw Lorrie's phone displaying the time of day on its dim OLED screen I thought that was a terrific feature. We purchased hers for her birthday and she lives on

that thing way more than I do on mine. She'll be sitting right next to a booted up desktop computer with a full size screen and a keyboard that actually invites typing rather than actively fighting against data entry, and she'll be squinting at websites on her phone.

In any event, last Friday, the day after I purchased the 16, the news broke that imports from China of smartphones and electronics were being exempted from the 154% import tariffs that had formed part of my purchase motivation. But then over this past weekend the US Commerce Secretary, Howard Lutnick explained during an interview on ABC's "This Week" Sunday morning show, that in another month or so, a new set of tariffs specifically targeting all semiconductor imports would be taking effect and that smartphones would be caught up in that.

A few months ago I purchased a new set of servers for GRC that I haven't gotten around to deploying yet, but they're here. When the second one of an earlier set of five died a few months ago I decided that I needed to be ready in case I lost another. So now I'm somewhat more glad that I already have those in hand in case their cost might soon be increasing. They were not inexpensive and it appears that a few months from now they might become more expensive.

I certainly have no crystal ball, and any rational actor, looking at the past month of tariff actions would be foolish to place any large bet. I'm quite certain that no one really knows what the future holds. But I clearly heard the US Commerce Secretary state that the administration's intention is to use higher import tariffs on all products containing semiconductors to force a shift in semiconductor manufacturing from offshore to the US. So, independent of the practicality, feasibility and sanity of any of that, we may indeed see the cost of devices containing semiconductors rising. What I **would** be willing to bet on is that prices are certainly not going to be dropping anytime soon.

I wanted to take a moment to talk about this since, I'm now more glad than I was that I had purchased those new servers a few months back. I would likely be doing that now for strategic savings if I hadn't already. I certainly don't know any more about what's going to happen than anyone else. And this could all change tomorrow. But if any of our listeners were waiting on the purchase of any big-ticket items containing semiconductors, it might be worth considering that prices may indeed be higher six months from now than they are today. I would certainly not place any bets on them being lower.

And as for my iPhone 16 Pro, if Apple ever does get around to deploying some AI, I'll be glad to have a device that allows me to experiment with it. And in the meantime, it's nice to have a dim clock on the lock screen, and to be able to edit text messages I've already sent.

Windows 11 Enterprise gets Hotpatching!

We were just talking about Apple silently enabling updates. Microsoft also recently made some news for Windows 11 Enterprise and Education users who will be getting updates on steroids in the form of the much anticipated no-reboot-required "Hotpatching", where Microsoft will only require a once-per-quarter full cold reboot with all other interim updates able to be applied directly to Windows' running "in memory" code. Microsoft's announcement blog posting about this is titled: *"Hotpatch for Windows client now available"* where David Callaghan, writing for the Windows IT Pro Blog explains:

> *Hotpatch updates for Windows 11 Enterprise, version 24H2 for x64 (AMD/Intel) CPU devices are now available. With hotpatch updates, you can quickly take measures to help protect your organization from cyberattacks, while minimizing user disruptions.*

Hotpatching represents a significant advancement in our journey to help you, and everyone who uses Windows, stay secure and productive. So, let's talk about the benefits, how it works, and how you and your organization can take advantage of this advancement as part of your Windows servicing journey.

Hotpatching offers numerous enhancements when it comes to keeping Windows client devices up to date:

- *Immediate protection: Hotpatch updates take effect immediately upon installation, providing rapid protection against vulnerabilities.*
- *Consistent security: Devices receive the same level of security patching as the monthly standard security updates released on the second Tuesday of every month.*
- *Minimized disruptions: Users can continue their work without interruptions while hotpatch updates are installed. Hotpatch updates don't require the PC to restart for the remainder of the quarter. (Note: OS features, firmware, and/or application updates may still cause a restart in the quarter.)*

You'll first create a hotpatch-enabled quality update policy in Windows Autopatch through the Microsoft Intune console. All eligible Windows 11 Enterprise, version 24H2 devices managed by this policy will be offered hotpatch updates in a quarterly cycle. The hotpatch updates follow the same ring deployment schedule as standard updates. Devices receiving the hotpatch update will see a different Knowledge Base number tracking the hotpatch release and a different OS version than devices receiving the standard update that requires a restart.

Hotpatch updates operate on a quarterly cycle:

- *Cumulative baseline month: In January, April, July, and October, devices install the monthly fixed security update and restart. This update includes the latest security fixes, cumulative new features, and enhancements since the last cumulative baseline.*
- *Subsequent two months: Devices receive hotpatch updates, which only include security updates and do not require a restart. These devices will catch up on features and enhancements with the next cumulative baseline month (quarterly).*

This cycle reduces the number of required restarts for Windows updates from twelve to just four per year, thanks to eight planned hotpatch updates annually. To enable hotpatching for Windows client devices, you'll need:

- *A Microsoft subscription that includes Windows 11 Enterprise E3, E5, or F3, Windows 11 Education A3 or A5, or a Windows 365 Enterprise subscription*
- *Devices running Windows 11 Enterprise, version 24H2 (Build 26100.2033 or later) and with the current baseline update installed*
- *An x64 CPU including AMD64 and Intel (Arm64 devices are still in public preview)*
- *Microsoft Intune to manage deployment of hotpatch updates with a hotpatch-enabled Windows quality update policy.*

We've known for some time that patching Windows on-the-fly without rebooting is both possible and practical, since this has been an aftermarket feature that the gang over at 0patch have been offering for some time. So in instances where Microsoft has strategically decided to abandon Windows security, the availability of those "0-patches" may be a godsend. But, bringing this to

Windows enterprise and education client machines means that millions more systems will receive the benefits of on-the-fly hot patching. Microsoft is not yet suggesting that this boot-avoidance technology might be available for their latest server platforms, but avoiding unnecessary server reboots would appear to be a nice feature for the future.

I don't have any problem with a brief one-a-month reboot of any of my workstation machines. Microsoft has invested heavily in minimizing the time required to install updates. They no longer require the huge amounts of time they once did.

Oracle Update

TL;DR: "They're still lying and denying."

Security researcher Kevin Beaumont, publishing on Medium from his "doublepulsar.com" site posted under the headline *"Oracle attempt to hide serious cybersecurity incident from customers in Oracle SaaS service"*. Kevin wrote:

> *Being a provider of cloud SaaS (Software-as-a-service) solutions requires certain cybersecurity responsibilities — including being transparent and open. The moment where this is tested at Oracle has arrived, as they have a serious cybersecurity incident playing out in a service they manage for customers.*
>
> *Back on March 21st, Bleeping Computer ran a story around a threat actor named rose87168 claiming to have breached some Oracle services inside *.oraclecloud.com*

Our listeners may recall that the fact digging Lawrence Abrams did for BleepingComputer was so thorough as to cross the line from evidence to proof of Oracle's apparently deliberate obfuscation and misdirection about the incident. Kevin continued:

> *Oracle told Bleeping Computer, and customers, "There has been no breach of Oracle Cloud. The published credentials are not for the Oracle Cloud. No Oracle Cloud customers experienced a breach or lost any data"*
>
> *The threat actor then posted an archive.org URL and provided it to Bleeping Computer, strongly suggesting they had write access to login.us2.oraclecloud.com, a service using Oracle Access Manager. This server is entirely managed by Oracle.*
>
> *Oracle have since requested Archive.org take down the proof and the wayback machine no longer shows the page.*
>
> *The threat actor then provided a several hour long recording of an internal Oracle meeting, complete with Oracle employees talking for two hours. The two hour video includes things like accessing internal Oracle password vaults, and customer facing systems.*
>
> *Both Hudson Rock and Bleeping Computer were then able to confirm with Oracle customers that their data — including staff email addresses — was in data released by the threat actor.*
>
> *The threat actor, rose87168, is still active online and releasing data — and threatening to release more. They have also released data to cybersecurity threat intelligence providers. In data released to a journalist for validation, it has now become 100% clear to me that there has been cybersecurity incident at Oracle, involving systems which processed customer data. For example, the threat actor has publicly provided complete Oracle configuration files — current,*

too. As one example, they have provided Oracle webserver configuration files.

All the systems impacted are directly managed by Oracle. Some of the data provided to journalists is current, too. This is a serious cybersecurity incident which impacts customers, in a platform managed by Oracle.

Oracle are attempting to wordsmith statements around Oracle Cloud and use very specific words to avoid responsibility. This is not okay. Oracle need to clearly, openly and publicly communicate what happened, how it impacts customers, and what they're doing about it. This is a matter of trust and responsibility. Step up, Oracle — or customers should start stepping off.

Kevin then provides three updates. First:

Update 1 — Oracle rebadged old Oracle Cloud services to be Oracle Classic. Oracle Classic has the security incident. Oracle are denying it on "Oracle Cloud" by using this scope — but it's still Oracle cloud services that Oracle manage. That's part of the wordplay.

Second Update:

Update 2 — Although Oracle used the archive.org exclusion process to remove evidence of writing to one of the Oraclecloud.com webservers, they forgot to remove a 2nd URL that clearly shows the threat actor, rose87168 having posted their email address on an Oracle Cloud page.

And a third and final update:

Update 3 — Multiple Oracle cloud customers have reached out to me to say Oracle have now confirmed a breach of their services. However, Oracle are only doing so verbally, they will not put anything in writing, so they're setting up meetings with large customers who query. This is similar behaviour to the breach of medical PII (personally identifiable information) in the ongoing breach at Oracle Health, where they will only provide details verbally and not in writing.

Over on Mastodon, Kevin posted:

And now, a class action lawsuit has been filed against Oracle over a data breach at Oracle Health, which Oracle has not acknowledged in public.
https://storage.courtlistener.com/recap/gov.uscourts.txwd.1172831612/gov.uscourts.txwd.1172831612.1.0.pdf

This Oracle thing keeps getting more and more wild, I've never seen a response so bad from a large org. They're throwing their own security staff under the bus by having them face customers, rather than the corporation actually take responsibility.

Oracle's handling of all this could be taught – and should be taught – as a short course in how NOT to ever handle a data breach. This whole business of only having verbal conversations and refusing to put anything into writing feels like attorneys being asked how to run a company. I'm

not sure that's a formula for success. Through my years as a small businessman I've had occasion to receive the advice of attorneys. I always thank them, and pay them, and carefully consider the value of their advice. But what they would advise, often seems to follow reactions to worst-case scenarios, whereas I've found that being more open and trusting and optimistic has worked better for me.

One of our listeners, whose first name is Keith, wrote from Canada:

> *Hi Steve, Thank you for covering the Oracle cloud breach in the latest episode highlighting the significance of the breach and the SEC violations. Given the "OCI classic" breach as they're dubbing it now, and the separate Oracle Health breach, I'm thoroughly confused on how they haven't had to disclose to the SEC. As a Canadian Oracle Health customer it's very frustrating to me that they seem to be above SEC regulations and still refuse to disclose breaches to us so we can be proactive in protecting our organizations. I'm a huge fan of you, Leo and the show. Thanks for everything you guys do!*

I wouldn't know what to tell Keith. Regulations only have teeth if they are backed by the certainty of enforcement. And to say that things are somewhat confused in the U.S. at this particular moment could safely be considered an understatement. Both our DOJ and SEC are currently preoccupied with trying to figure out which end is up and what their priorities should be. So it may be that Oracle lucks out on this one and that it slips by on the government side. But as I noted, US citizens have already filed lawsuits that may force depositions and place additional facts on the record.

US Treasury Office of the Comptroller of the Currency (OCC) longstanding breach

The United States Treasury has something known as the Office of the Comptroller of the Currency (OCC). A couple of months ago, in January of this year, CISA discovered that the emails for nearly 100 of the OCC's staff had been intercepted since the breach originally occurred back in June of 2023 encompassing more than 150,000 pieces of email. That's right, since June of 2023, nearly two years ago, none of the nearly 100 staffers at the US Treasury's Office of the Comptroller of the Currency have enjoyed any actual email privacy. It's all just been an illusion. And Treasury does appear to be either a high priority target or to have less than adequate security since this OCC breach is the third Treasury office to recently disclose a breach. Before this, we had the Office of Foreign Assets Control (OFAC) and the Committee on Foreign Investment in the US (CFIUS). For both of those two previous intrusions the US government credited the Chinese hacking group Silk Typhoon.

This news connected with something I heard over the weekend. An Asian analyst was interviewed by Freed Zakaria during his Sunday morning show on CNN. She made the comment about how at some point, as tensions between the US and China escalated, China might decide to weaponize all of the data they'd been collecting through their pervasive cyber intrusions into the US. That gave me a bit of a chill because, unfortunately, it really made sense. We've seen a great deal of evidence of Chinese, apparently state sponsored, actors rummaging around inside US government and industry networks. But nothing overt and obvious has come of it. It might be that an "attack" – as such – would take the form of using all of the information that's been gleaned against US interests. In other words, "weaponizing all that data." We don't know that this recent and long running US Treasury Office of the Comptroller of the Currency email breach was the same people who were previously found to have breached those two other US Treasury offices; there's been no attribution so far. But at this point it would almost be surprising if it wasn't again the Silk Typhoon group.

Apple -vs- UK

There's some news on the Apple -vs- the UK and what Apple will do about the UK's demands to be able to obtain the stored iCloud data for anyone in the world they request. Apple Insider's headline was: *"UK iCloud backdoor mandate hearing must be made public — eventually"*. They wrote:

> *After a legal challenge by Apple, the hearing about blowing open Apple's iCloud encryption in the UK for the sake of national security will not be kept secret, but it's not clear when the details will be made public. After the hearing about a mandated back door happened behind closed doors, Apple very nearly immediately filed an appeal, with the backing of most of the world's governments, privacy advocates, and journalism organizations. That appeal has been heard, and at some point, the results of the hearing will be made clear.*
>
> *The Investigatory Powers Tribunal **rejected** claims from the UK government that national security would be hurt by revealing the results of the hearing, or exposing who attended the hearing. In short, the appeal found that there was no reason to restrict what it calls open justice, so the results of the hearing must be made clear — in due time. It's not clear when that will happen, as case management orders will be made only after Apple and the UK government have time to consider the ruling, and propose drafts.*

So, bureaucracy. Whatever is going to happen will apparently grind away slowly. But the fact that the UK government now knows that it will not also be able to conduct everything in secret may, hopefully, dampen their zeal somewhat and reign them in. What's interesting about this is that there's no middle ground here. There's no gray area. UK users either will or will not have the ability to enable Apple's Advanced Data Protection for their stored iCloud data. It seems unlikely in the extreme that the UK's demand to be able to obtain the data belonging to anyone they choose anywhere has any chance of happening. But they might well force Apple to disable ADP for citizens of the UK. We'll see.

ThunderMail & Thunderbird Pro

I missed this news when it happened 10 days ago, but I felt the need to come back to put it on everyone's radar because what Mozilla is doing with a suite of new cloud service offerings which they're calling Thundermail and Thunderbird Pro will, I'm sure, be of interest to many of our listeners for much the same reason we choose to use Mozilla's Firefox. Mozilla wrote:

> *Today we're pleased to announce what many in our open source contributor community already know. The Thunderbird team is working on an email service called "Thundermail" as well as file sharing, calendar scheduling and other helpful cloud-based services that as a bundle we have been calling "Thunderbird Pro."*
>
> *First, a point of clarification: Thunderbird, the email app, is and always will be free. We will never place features that can be delivered through the Thunderbird app behind a paywall. If something can be done directly on your device, it should be. However, there are things that cannot be done on your computer or phone that many people have come to expect from their email suites. This is what we are setting out to solve with our cloud-based services.*
>
> *All of these new services are (or soon will be) open source software under true open source licenses. That's how Thunderbird does things and we believe it is our super power. It is also a major reason we exist: to create open source communication and productivity software that respects our users. Because you can see how it works, you can know that it is doing the right*

thing.

The Why for offering these services is simple. Thunderbird loses users each day to rich eco-systems that are both products and services, such as Gmail and Office365. These eco- systems have both hard vendor lock-ins (through interoperability issues with 3rd-party clients) and soft lock-ins (through convenience and integration between their clients and services). It is our goal to eventually have a similar offering so that a 100% open source, freedom-respecting alternative ecosystem is available for those who want it. We don't even care if you use our services with Thunderbird apps, go use them with any mail client. No lock-in, no restrictions – all open standards. That is freedom. So what Are The Services?

Thunderbird Appointment

Appointment is a scheduling tool that allows you to send a link to someone, allowing them to pick a time on your calendar to meet. The repository for Appointment has been public for a while and has seen pretty remarkable development so far. It is currently in a closed Beta and we are letting more users in each day. Appointment has been developed to make meeting with others easier. We weren't happy with the existing tools as they were either proprietary or too bloated, so we started building Appointment.

Thunderbird Send

Send is an end-to-end encrypted file sharing service that allows you to upload large files to the service and share links to download those files with others. Many Thunderbird users have expressed interest in the ability to share large files in a privacy-respecting way – and it was a problem we were eager to solve. Thunderbird Send is the rebirth of Firefox Send – well, kind of. We've rebuilt much of the project to allow for a more direct method of sharing files (from user-to-user without the need to share a link). We opened up the repo to the public earlier this week. So we encourage everyone interested to go and check it out. Thunderbird Send is currently in Alpha testing, and will move to a closed Beta very soon.

Thunderbird Assist

Assist is an experiment, developed in partnership with Flower AI, a flexible open-source framework for scalable, privacy-preserving federated learning, that will enable users to take advantage of AI features. The hope is that processing can be done on devices that can support the models, and for devices that are not powerful enough to run the language models locally, we are making use of Flower Confidential Remote Compute in order to ensure private remote processing (very similar to Apple's Private Cloud Compute). Given some users' sensitivity to this, these types of features will always be optional and something that users will have to opt into. As a reminder, Thunderbird will never train AI with your data. The repo for Assist is not public yet, but it will be soon.

Thundermail

Thundermail is an email service in search of a better name. Okay that's not actually what it says. I just think that "Thundermail" sounds dumb. You just can't put "thunder" in front of everything and have it work. Anyway, it also supports calendars and contacts as well as mail.

They wrote: We want to provide email accounts to those who love Thunderbird, and we believe that we are capable of providing a better service than the other providers out there.

*Email that aligns with our values of privacy, freedom and respect of our users. No ads, no selling or training AI on your data – just your email and it is **your** email. With Thundermail, it is our goal to create a next generation email experience that is completely, 100% open source and built by all of us, our contributors and users. Unlike the other services, there will not be a single repository where this work is done. But we will try and share relevant places to contribute in future posts like this. The email domain for Thundermail will be Thundermail.com or tb.pro. Additionally, you will be able to bring your own domain on day 1 of the service.*

Now that starts being interesting. Having Mozilla behind a 100% open source privacy respecting email service where we're also able to bring our own domain. Presumably by pointing our own domain's MX records at Mozilla's offering.

Heading to thundermail.com you will see a sign up page for the beta waitlist. Please join it!

(I did, immediately.)

Final Thoughts / Don't services cost money to run?

You may be thinking: "this all sounds expensive, how will Thunderbird be able to pay for it?" And that's a great question! Services such as Send are actually quite expensive (storage is costly). So here is the plan: at the beginning, there will be paid subscription plans at a few different tiers. Once we have a sufficiently strong base of paying users to sustainably support our services, we plan to introduce a limited free tier to the public. You see this with other providers: limitations are standard as free email and file sharing are prone to abuse.

It's also important to highlight again that Thunderbird Pro will be a completely separate offering from the Thunderbird you already use.

Or in my case, "once used", since I so happily switched from Thunderbird to eM Client.

While Thunderbird and the additional new services may work together and complement each other for those who opt in, they will never replace, compromise, or interfere with the core features or free availability of Thunderbird. Nothing about your current Thunderbird experience will change unless you choose to opt in and sign up with Thunderbird Pro. None of these features will be automatically integrated into Thunderbird desktop or mobile or activated without your knowledge.

This has been a long time coming. It is my conviction [this post was written by Ryan Sipes Managing Director of Product for Thunderbird] *that all of this should have been a part of the Thunderbird universe a decade ago. But it's better late than never. Just like our Android client has expanded what Thunderbird is (as will our iOS client), so too will these services.*

Thunderbird is unique in the world. Our focus on open source, open standards, privacy and respect for our users is something that should be expressed in multiple forms. The absence of Thunderbird web services means that our users must make compromises that are often uncomfortable ones. This is how we correct that.

I hope that all of you will check out this work and share your thoughts and test these things out. What's exciting is that you can run Send or Appointment today, on your own server.

Everything that we do will be out in the open and you can come and help us build it! Together we can create amazing experiences that enhance how we manage our email, calendars, contacts and beyond. Thank you for being on this journey with us.

https://blog.thunderbird.net/2025/04/thundermail-and-thunderbird-pro-services/

We all want Mozilla to stay alive. If not for Thunder-Whatever then for the sake of Firefox. So if their addition of cloud-based services appeals to people as a reasonable alternative to Office 365 and Gmail, and that creates a revenue stream to support all of Mozilla, then I'm all for it. So you can head over to https://thundermail.com to sign up for news.

Meta extends "teen accounts" protections

Over in the age restrictions world is the news that Meta has extended teen account protections. The existing "teen accounts" security protections which exist on Instagram will also be extended to Facebook and Facebook Messenger accounts. The feature prevents children under the age of 16 from modifying a series of privacy settings on their accounts without a parent's approval. This includes settings related to who can contact the account and what content they see on the site. Meta is also expanding these restrictions so that, for example, teens won't be able to live-stream on their sites without a parent's approval.

Patch Tuesday

With our podcast two weeks ago falling on April Fools Day, that made last week's podcast fall on the earliest possible Patch Tuesday day, April 8th. Looking back at the news of last week, Microsoft patched 126 vulnerabilities, one of which was an actively exploited 0-day, as Elevation of Privilege in the Windows Common Log File System driver – which tends to be a vulnerability magnet for some reason. Microsoft's security team indicated that the now-patched 0-day was being exploited by the RansomEXX ransomware group. And that makes sense since once you somehow arrange to get code running on a well locked down Windows machine, that code will likely be running under the account of a user with deliberately restricted privileges. So even though "you're in", it's still generally necessary to arrange to obtain administrative privileges if, as a ransomware intrusion, your goal is to do a lot of damage.

Google also patched a pair of 0-days last week in Android. One of the fixes is a patch for a Cellebrite exploit used by Serbian authorities to unlock the phones of journalists and anti-government protesters. The exploit and the hacks were first detailed in an Amnesty International report in February. There are no details on the second 0-day other than it leverages an undisclosed flaw in the Android kernel USB-audio driver. But being in the Android kernel suggests that it was likely a powerful root-level exploit. This also makes it the third month in a row that Google has fixed 0-days in the Android OS. As we know, it's very difficult to get every detail right.

Security of Programmable Logic Controllers

If I wasn't so excited about talking about *"Device Bound Session Credentials"* today, as we will be shortly, I would be spending our time digging into a 25-page, recently published piece of security research which examined the status of the security of PLCs — the critical Programmable Logic Controllers that generally contain just enough computational ability to figure out when to turn off the toilet paper rolling machine, to then cut the paper and start another role after first painting a little bit of glue onto the cardboard tube so that the new end of the paper sticks.

In a very real sense, PLCs are what actually run the world. We've talked about them extensively in the past on this podcast specifically because they are silent workers that essentially make all of today's infrastructure go. In a very real sense, they are today's infrastructure. And as a consequence, their security is crucial. In the Abstract of their paper, the team of researchers wrote:

> *Billions of people rely on essential utility and manufacturing infrastructures such as water treatment plants, energy management, and food production. Our dependence on reliable infrastructures makes them valuable targets for cyberattacks. One of the prime targets for adversaries attacking physical infrastructures are Programmable Logic Controllers (PLCs) because they connect the cyber and physical worlds. In this study, we conduct the first comprehensive systematization of knowledge that explores the security of PLCs: We present an in-depth analysis of PLC attacks and defenses and discover trends in the security of PLCs from the last 17 years of research. We introduce a novel threat taxonomy for PLCs and Industrial Control Systems (ICS). Finally, we identify and point out research gaps that, if left ignored, could lead to new catastrophic attacks against critical infrastructures.*

I won't dig into this further because we have more to get to this week. But here's a brief summary written by a security reporter who did dig into it. He wrote:

> *A team of academics has conducted a review of 133 papers, 119 attack methods, and 70 defense methods that target PLCs to assess the actual impact of a possible cyberattack targeting these devices. The research found that even if most PLCs have built-in access control features, most of these have been shown to be ineffective. Where encryption has been used, the algorithms are often ineffective. Disabling unused protocols and monitoring is the best way to prevent and detect attacks.*

If anyone is interested in more detail, I have a link (https://arxiv.org/pdf/2403.00280) to their 25-page research analysis in the show notes.

LLM Package Hallucinations

Okay. I've got one that's pretty much guaranteed to make you just shake your head. Six researchers, four from the University of Texas at San Antonio, one from Virginia Tech and the last from the University of Oklahoma, have just published a paper titled: *"We Have a Package for You! A Comprehensive Analysis of Package Hallucinations by Code Generating LLMs"*. In their usage, just to be clear, by "package" they mean a reference to some open-souce code library that it would be handy to have and to add to a project. Here's what this team if six wrote for their paper's Abstract: (I have a link to the entire paper in the show notes.)

> *The reliance of popular programming languages such as Python and JavaScript on centralized package repositories and open-source software, combined with the emergence of code-generating Large Language Models (LLMs), has created a new type of threat to the software supply chain: package hallucinations. These hallucinations, which arise from fact-conflicting errors when generating code using LLMs, enable a novel form of package confusion attack that poses a critical threat to the integrity of the software supply chain. This paper conducts a rigorous and comprehensive evaluation of package hallucinations across different programming languages, settings, and parameters, exploring how a diverse set of models and configurations affect the likelihood of generating erroneous package recommendations and identifying the root causes of this phenomenon.*

Using 16 popular LLMs for code generation and two unique prompt datasets, we generate 576,000 code samples in two programming languages that we analyze for package hallucinations. Our findings reveal that the average percentage of hallucinated packages is at least 5.2% for commercial LLM models and 21.7% for open-source LLM's, including a staggering 205,474 unique examples of hallucinated package names, further underscoring the severity and pervasiveness of this threat. To overcome this problem, we implement several hallucination mitigation strategies and show that they are able to significantly reduce the number of package hallucinations while maintaining code quality. Our experiments and findings highlight package hallucinations as a persistent and systemic phenomenon while using state-of-the-art LLMs for code generation, and a significant challenge which deserves the research community's urgent attention.

Okay. So that's part one. LLM's are still just making stuff up – including the names of add-on packages that it would be nice to have. And just as "typosquatting" has developed over time into a serious threat, researchers are warning that something which is being called AI "slopsquatting" is on the horizon. Here's what the Risky Business security newsletter wrote:

Security firms, open-source experts, and academics are warning about a new supply chain vector they're calling slopsquatting. The technique's name is a combination of terms like AI slop and typosquatting. It revolves around the increasing use of AI coding tools to generate blocks of source code that may sometimes make their way into production systems.

A recent academic paper [the one whose Abstract I just shared] analyzed 16 AI coding models and found that these tools generate shoddy code that often includes and loads packages and libraries that don't exist. DevSecOps company Socket Security says that such behavior opens the door to slopsquatting—where threat actors study the LLMs and then register package names hallucinated or likely to be hallucinated in the future.

The attack looks farcical and impractical, but so did typosquatting when it was first described years ago. Yet, years later, it is one of the most pervasive and common sources of supply chain issues in the software development industry. It may sound ridiculous that developers would not spot a typo in the names of packages they install, but reality has shown they don't.

Does it actually sound that far off that developers would not spot non-existent packages in huge blocks of code they're using when cutting corners? The use of AI coding tools is increasing, and the chances that developers may use code blocks generated through these tools is also growing exponentially, along with the chances of a successful slopsquatting attacks.

So that's Risky Business wrote. This raised my curiosity so I looked further. The Socket Security folks further summarized some of the paper's findings. They wrote:

The researchers tested 16 leading code-generation models, both commercial (like GPT-4 and GPT-3.5) and open source (like CodeLlama, DeepSeek, WizardCoder, and Mistral), generating a total of 576,000 Python and JavaScript code samples. Their key findings were:

- *19.7% of all recommended packages didn't exist.*
- *Open source models hallucinated far more frequently—21.7% on average—compared to commercial models at 5.2%.*

- *The worst offenders (CodeLlama 7B and CodeLlama 34B) hallucinated in over a third of outputs.*
- *GPT-4 Turbo had the best performance with a hallucination rate of just 3.59%.*
- *Across all models, the researchers observed over 205,000 unique hallucinated package names.*

These findings point to a systemic and repeatable pattern—not just isolated errors.

And here's the key: These hallucinations are not just one-off events. If they were, they could not be weaponized. They are persistent. The Socket Security guys explained:

In follow-up experiments, the researchers reran 500 prompts that had previously triggered hallucinations, ten times each. They found an interesting split when analyzing how often hallucinated packages reappeared in repeated generations.

When re-running the same hallucination-triggering prompt ten times, 43% of hallucinated packages were repeated every time, while 39% never reappeared at all. This stark contrast suggests a bimodal pattern in model behavior: hallucinations are either highly stable or entirely unpredictable.

Overall, 58% of hallucinated packages were repeated more than once across ten runs, indicating that a majority of hallucinations are not just random noise, but repeatable artifacts of how the models respond to certain prompts. That repeatability increases their value to attackers, making it easier to identify viable slopsquatting targets by observing just a small number of model outputs.

This consistency makes slopsquatting more viable than one might expect. Attackers don't need to scrape massive prompt logs or brute force potential names. They can simply observe LLM behavior, identify commonly hallucinated names, and register them.

So just a cautionary tale about the potential for the weaponization of LLM outputs. We know that bad guys would like nothing more than to get their code included into high-profile product offerings. If future coders become too comfortable with directly using LLM-created code without scrutinizing it – just copying, pasting and testing what the LLM produces – it's no longer far-fetched to imagine that the LLM's mistaken output itself might have been weaponized for the purpose of causing the download and inclusion of a malicious library.

If we were to take this a step further, imagine arranging to seduce LLM's to train on tasty valid libraries, which they would then tend to invoke into their solutions, only to have any retrieval by any NON-LLM return a malicious version of that package.

There's no such thing as a free lunch!

Wordpress Core Security

We wind up talking about Wordpress because such a large portion of the Internet's websites are running Wordpress CMS – Content Management System – code. The core Wordpress offering has become extremely solid over time. But its huge plug-in ecosystem is another matter. That plug-in ecosystem is Wordpress' primary attraction but also its primary weakness as a secure platform.

WordFence is an independent Wordpress-focused security firm. During the previous year, security researchers discovered and disclosed more than 8,000 WordPress site vulnerabilities. But fully one quarter of those have remained unpatched. Many of the affected plug-ins are obscure, but many are popular but unmaintained.

But as I noted, the Wordpress core has grown increasingly solid with only 5 of those 8,000 known issues disclosed last year impacting the WordPress core.

So the takeaway here is, as I've said every time we have previously considered the important Wordpress landscape: Be very very careful about what you add to the base Wordpress core offering. Only add those features you really need and will use, and check to see the history of any add-on's maintenance to verify that someone is still around to maintain that code.

PHP's language interpreter gets a security audit

Wordpress, like a great many other web-facing systems such as GRC's web forums, our email system and our like shortener, are all written in PHP. Also in the news was that PHP's language interpreter recently received a security audit. QuarksLab received a commission to really examine the core component of PHP. Last Thursday they posted their results, writing:

The Open Source Technology Improvement Fund, Inc, thanks to funding provided by Sovereign Tech Fund, engaged with Quarkslab to perform a security audit of PHP-SRC, the interpreter of the PHP language.

The audit aimed to assist PHP's core developers and the community in strengthening the project's security ahead of the upcoming PHP 8.4 release. The codebase was analyzed within a defined scope, which was established and agreed upon by both PHP's core developers and the OSTIF (Open Source Technology Improvement Fund) teams. Based on this scope and the allocated time frame for the audit, an attack model was developed and approved by the PHP team.

The assessment was conducted within a set timeframe, with the primary focus on identifying vulnerabilities and security issues in the code according to the defined attack model.

The following scope of work was defined by PHP Foundation and the OSTIF.

Key tasks included:

- *basic tooling evaluation;*
- *improve SAST tooling to enhance the existing GitHub CI without extra cost and with low maintenance;*
- *build fuzzers compatible with oss-fuzz for potential critical functions that are not currently covered;*
- *cryptographic and manual code review.*

High priority tasks were:

- *php-fpm master node and php-fpm worker glue code;*
- *FPM pool separation;*
- *MySQL Native Driver;*
- *RFC 1867 HTTP header parser and MIME handling;*
- *PDO: emulated prepares;*
- *JSON parsing with a focus on json_decode;*
- *OpenSSL external functions and its stream layer ext/openssl;*

- *libsodium integration ext/sodium;*
- *functionalities related to passwords ext/standard/password.c;*
- *functionalities related to hashing ext/hash;*
- *functionalities related to CSPRNG ext/random/csprng.c.*

How did they proceed?

To assess the security of PHP-SRC, Quarkslab's team first needed to familiarize themselves with the structure of the project and understand the key tasks outlined in the audit's scope. To achieve this, Quarkslab experts gathered and reviewed the available documentation and project resources. With a clear understanding of the features to be evaluated, Quarkslab developed an attack model that incorporated all the requested key tasks. This model was then presented to PHP's core developers, and once approved, the assessment began.

The evaluation employed a combination of dynamic and static analysis. The static analysis focused on scrutinizing the source code to identify vulnerabilities related to the implementation and logic of the specified assessment targets. Dynamic analysis was used to complement the static review by speeding up the process through fuzzing and validating or refuting the hypotheses generated during the static analysis.

And what did they find?

During the time frame of the security audit, Quarkslab has discovered several security issues and vulnerabilities, among which were:

- *2 security issues considered as high severity;*
- *6 security issues considered as medium severity;*
- *9 security issues considered as low severity;*
- *10 issues considered informative.*

Most vulnerabilities have been shared via security advisories on the PHP-SRC GitHub repository. Other bugs and issues are provided only in this report. Four CVEs were issued, one for each of the two high severity vulnerabilities and two others for two of the 9 low severity vulnerabilities.

They produced a detailed 106-page full audit report and I have a link to it in the show notes.
https://ostif.org/wp-content/uploads/2025/04/24-07-1730-REP-V1.4_temp.pdf

However they also wrote: *"This audit report contains two security issues currently redacted while PHP maintainers are actively working on the fixes. Details will be provided after fixes are applied by PHP maintainers. Fixes are complex and in progress."*

In other words, two of the 17 security-related problems they discovered were too severe to publicly report until they've been fixed. Although it's speculation at this point, this suggests that many earlier releases of PHP are also very likely to be in identical trouble and that, depending upon what bad guys could do with it if they knew, we may be facing a critically important security update across all still supported release versions of PHP.

Device Bound Session Credentials

While I was scanning through recent events, I noted that Chrome had recently moved to 135 and Firefox had moved to 137. So I scanned through Chrome's mind-numbing list of things that had been fixed, added or changed.

There were several truly new features added by the W3C, which Firefox and Safari are also echoing. The most interesting of them was something called "Device Bound Session Credentials" which is the soon-to-be-available feature that named today's podcast. Once I understood what this was about, that it was right, and given that this new technology is intended to be an extremely secure replacement for session cookies, I knew we needed to update the record because session cookies would not be long for this world, and that would be a big deal and change everything.

As we've had the occasion to discuss many times in the past, the entire model of the web is for a user client, typically an interactive web browser, to request some resource from the Internet using a URL which contains the unique address of the requested object. In respect to the browser's connection to it and supplying the address of the requested object, a web server returns whatever it is that the browser requested, then they may and often do, disconnect.

When you think about it, it's sort of incredible to consider how far we have stretched that simple basic query and reply model. Look what we've created with it!

This original model, the thing that Sir Timothy John Berners-Lee first conceived of as the World Wide Web, never had any notion of a "session". That is, there was no way for anyone to "log on" to anything, since doing so would require that this "logged on" state would be saved somewhere. And Tim's original idea was entirely stateless. The "web" was just a mass of pages containing links to other pages. And that was it.

That changed in June of 1994 when MCI asked Netscape to come up with some way for the user's browser to retain transaction data so that MCI wouldn't need to retain it at their end. So a Netscape engineer named Lou Montulli came up with the idea of a web browser cookie that a web server would give to a visiting web browser and every time thereafter, if the web browser contained a "cookie" that matched the domain the browser was querying, the browser would voluntarily return that "cookie" token in all of its queries to the server.

Even back then this was somewhat controversial, since it suddenly meant that not every query from a browser was independently and entirely anonymous. But by the same token, the web server would usually have the browsing user's IP address. Still, people were aware of this in the mid 1990's.

Through the years, the cookie specification was formalized and new features were added. Many years ago we talked about the Firesheep hack where HTTPS was only briefly used during login to a website, after which the connections would drop back to less compute intensive plaintext HTTP. The trouble was that this exposed the user's "session cookie" which was the only way remote servers had to recognize a user's repeated activities. So if a bad guy were to sniff a cookie they could instantly impersonate that logged on user.

This obvious flaw was fixed by switching to always keeping all traffic encrypted using HTTPS. But if a browser ever even once made the mistake of issuing an HTTP query to the remote server, whatever cookies it might be carrying for that server domain would be sent in the clear. So the

formal cookie specification was again tweaked so that the server could set a "secure" flag with a cookie. This would instruct the browser to never send the cookie out over an unencrypted HTTPS query. So today, all responsible cookie setting now uses the "secure" flag to prevent any cookie leakage.

But if you stand back for a moment and consider how much work we're asking these poor old original cookies to do, and how much more technology we have readily available to us today than we did 31 years ago back in 1994 – especially our lovely crypto technology – the need to replace these trusty and crusty old cookies – which are just dumb pseudo-random bits of gibberish – with something far more powerful, resilient and resistant to abuse becomes quite hard to resist. And today it's something we could do so easily.

That session cookie replacement is now on the horizon, it's everything it could be, and it's called *"Device Bound Session Credentials"* — or **DBSC** for short.

So what are Device Bound Session Cookies? The World Wide Web Consortium's (W3C's) public Github page, part of which I'm going to share, is quite dense and matter-of-fact. But don't worry if some of this is initially confusing and flies over your head. This is enough of a change from the way things have always been done for the past 31 years that it will likely take another podcast or two for all of what this means to sink in. But we'll all get there together. Here's what the W3C considers to be their "explainer". They write:

Device Bound Session Credentials (DBSC) aims to reduce account hijacking caused by cookie theft. It does so by introducing a protocol and browser infrastructure to maintain and prove possession of a cryptographic key.

The main challenge with cookies as an authentication mechanism is that they only lend themselves to bearer-token schemes. On desktop operating systems, application isolation is lacking and local malware can generally access anything that the browser itself can, and the browser must be able to access cookies. On the other hand, authentication with a private key allows for the use of system-level protection against key exfiltration.

DBSC offers an API for websites to control the lifetime of such keys, behind the abstraction of a session, and a protocol for periodically and automatically proving possession of those keys to the website's servers. There is a separate key for each session, and it should not be possible to detect if two different session keys are from one device. One of the key goals is to enable drop-in integration with common types of current auth infrastructure. By device-binding the private key and with appropriate intervals of the proofs, the browser can limit malware's ability to offload its abuse off of the user's device, significantly increasing the chance that either the browser or server can detect and mitigate cookie theft.

DBSC is bound to a device with cryptographic keys that cannot be exported from the user's device under normal circumstances, this is called device binding in the rest of this document.

DBSC provides an API that servers can use to create a session bound to a device, and this session can periodically be refreshed with an optional cryptographic proof the session is still bound to the original device.

At sign-in, the API informs the browser that a session starts, which triggers the key creation. It then instructs the browser that any time a request is made while that session is active, the browser should ensure the presence of certain cookies. If these cookies are not present, DBSC will hold network requests while querying the configured endpoint for updated cookies.

DBSC's goal is to reduce session theft by offering an alternative to long-lived cookie bearer tokens, that allows session authentication that is bound to the user's device. This makes the internet safer for users in that it is less likely their identity is abused, since malware is forced to act locally and thus becomes easier to detect and mitigate. At the same time the goal is to disrupt the cookie theft ecosystem and force it to adapt to new protections.

DBSC's primary threat model is that of an attacker who can read and tamper with the user agent, such as with a malware-compromised browser, in which the malware can read and modify browser memory and secrets stored on disk. In many operating systems, malware may be able to obtain privileged (root, kernel, etc.) access. DBSC aims to address this threat by establishing a cryptographic protocol in which secrets can be stored in dedicated systems (such as secure enclaves), though DBSC does not specify how implementors should store, backup, or sync keys as long as such storage is robust against the described threat.

As a secondary consideration, DBSC also mitigates against certain types of network and server compromise, such as network Attackers-in-the-Middle (where an attacker can read or modify network traffic) or HTTP server log leaks (where a server mistakenly logs full HTTP request/response headers to logs which can be read by unprivileged insiders).

In all of these scenarios, DBSC aims to enforce the specific constraint that temporary read/write access to a user agent or network traffic does not enable long-lived access to any established DBSC sessions. For example, if an attacker has malware running within a victim browser process, they should be unable to continue to authenticate as the victim browser once that malware is removed. (Note, however, that the definition of "long-lived" depends upon the configured refresh period; within that period, attackers may continue to have short-lived access to any established sessions.)

And as for "Non-goals"

DBSC will not prevent temporary access to any browser sessions while the attacker has ongoing access to a compromised user-agent. An attacker with ongoing access to a compromised user agent (or decrypting middlebox, etc) will be able to continuously access fresh DBSC-controlled bearer tokens, and an attacker with malware running on a compromised device will, on many modern operating systems, be able to treat even secure elements as a signing oracle, in order to provide proof-of-possession of the DBSC secret keys.

So what makes Device Bound Session Credentials different?

DBSC is not the first proposal towards these goals, with a notable one being Token Binding. This proposal offers two important features that we believe makes it easier to deploy than previous proposals. DBSC provides application-level binding and browser initiated refreshes that can make sure devices are still bound to the original device.

For websites, device binding is most useful for securing authenticated sessions of users. DBSC allows websites to closely couple the setup of bound sessions with user sign-in mechanisms, makes session and key lifetimes explicit and controllable, and allows servers to design infrastructure that places verification of session credentials close to where user credentials (cookies) are processed in their infrastructure.

Other proposals have explored lower-level APIs for websites to create and use protected private keys, e.g. via WebCrypto or APIs similar to WebAuthn. While this works in theory, it

puts a very large burden on the website to integrate with. In particular, since the cost of using protected keys is high, websites must design some infrastructure for collecting signatures only as often as needed.

This means either high-touch integrations where the keys are only used to protect sensitive operations (like making a purchase), or a general ability to divert arbitrary requests to some endpoint that collects and verifies a signature and then retries the original request. The former doesn't protect the whole session and violates the principle of secure by default, while the latter can be prohibitively expensive for large websites built from multiple components by multiple teams, and may require non-trivial rewrites of web and RPC frameworks.

DBSC instead allows a website to consolidate the session binding to a few points: At sign-in, it informs the browser that a session starts, which triggers the key creation. It then instructs the browser that any time a request is made while that session is active, the browser should ensure the presence of certain cookies. The browser does this by calling a dedicated refresh endpoint (specified by the website) whenever such cookies are needed, presenting that endpoint with a proof of possession of the private key. That endpoint in turn, using existing standard Set-Cookie headers, provides the browser with short-term cookies needed to make other requests.

Whew! Okay. We finally got some sense for what's going on here. Many previous efforts to replace cookies have been proposed in the past but none have taken hold for various reasons. DBSC presents a carefully crafted compromise.

Rather than constantly and continually using expensive public key crypto to prove its identity, DBSC sets up a secondary "cookie supplier" for a website. The website tells the browser which cookies it needs to be providing. And if the browser doesn't have those, then, and only then, it separately connects to the "cookie supplier" where it uses rigorous state of the art crypto to authenticate its DEVICE – not its browser, its hardware device – to the website's cookie supplier. Having done so, the cookie supplier returns regular old fashioned cookies which the browser will then use when subsequently transacting with the main website's pages.

The explainer continues:

This provides two important benefits:

- *First, session binding logic is consolidated in the sign-in mechanism, and the new dedicated refresh endpoint. All other parts of the website continue to see cookies as their only authentication credentials, the only difference is that those cookies are short-lived. This allows deployment on complex existing setups, often with no changes to non-auth related endpoints.*

- *Second, if a browser is about to make a request where it has been instructed to include such a cookie, but doesn't have one, it defers making that request until the refresh is done. While this may add latency to such cases, it also means non-auth endpoints do not need to tolerate unauthenticated requests or respond with any kind of retry logic or redirects. This again allows deployment with minimal changes to existing endpoints.*

Note that the latency introduced by deferring of requests can be mitigated by the browser in other ways, which will be discussed later.

TPM considerations

DBSC depends on user devices having a way of signing challenges while protecting private keys from exfiltration by malware. This usually means the browser needs to have access to a Trusted Platform Module (TPM) on the device, which is not always available. TPMs also have a reputation for having high latency and not being dependable. Having a TPM is a requirement for installing Windows 11, and can be available on previous versions. All our studies are for public key cryptography using ECDSA_P256 algorithm.

Chrome has done studies to understand TPM availability to understand the feasibility of secure sessions. Current data shows about 60%, and currently growing, of Windows users would be offered protections. Studies have also been done on the current populations of TPMs, both for latency and for predictability. Currently the latency for signing operations averages 200ms with only 5% of signing operations exceeding 600ms, and the error rate is very low, currently around 0.001%.

Based on this research, TPMs are widely available, with a latency and consistency that is acceptable for the proposed usage.

What about privacy considerations

An important high-level goal of this protocol is to introduce no additional surface for user tracking: implementing this API (for a browser) or enabling it (for a website) should not entail any significant user privacy tradeoffs.

There are a few obvious considerations to ensure we achieve that goal:

- *Lifetime of a session/key material: This should provide no additional client data storage (i.e., a pseudo-cookie). As such, we require that browsers MUST clear sessions and keys when clearing other site data (like cookies).*

- *Cross-site/cross-origin data leakage: It should be impossible for a site to use this API to circumvent the same origin policy, 3P cookie policies, etc.*

- *Implementing this API should not meaningfully increase the entropy of heuristic device fingerprinting signals. (For example, it should not leak any stable TPM-based device identifier.)*

- *This API—which allows background "pings" to the refresh endpoint when the user is not directly active—must not enable long-term tracking of a user when they have navigated away from the connected site.*

- *Each session has a separate new key created, and it should not be possible to detect that different sessions are from the same device.*

- *Registration and refresh will only be performed over a secure connection (or with localhost for testing).*

To achieve these goals, we add the following constraints to DBSC requests:

- *Registration and refresh are made in the context of the request that triggered them. For registration, this is the request serving the Sec-Session-Registration header. For refresh,*

this is the request deferred due to missing cookies.

- *Cookie refresh only occurs if the cookie is accessible. DBSC will not attempt to refresh a third-party cookie if third-party cookies are blocked.*

- *Proactive refreshes must only occur if any tab has a page from the site loaded.*

Enterprise support

While DBSC addresses a general problem of session hijacking, and can be applicable to any browser consumer, it is possible to expand this protocol to better support enterprise use cases. By adding specifics to key generation, we can provide a more secure environment for enterprise users. This is the goal of DBSC(E), which is an extension to DBSC. The high-level design of DBSC(E) is described in the DBSC(E) Overview. DBSC(E) removes the vulnerability DBSC has, where a malware, if already present in the device during the key generation, can potentially take over the session. DBSC(E) proposes to mitigate this vulnerability by introducing device key chaining.

I'm fully aware that this was quite a lot to digest just now. And we're at the end of a lengthy podcast with no time to dig further into exactly how this works.

But at least the essence of this new system is probably clear:

Cookies still exist but they are short lived rather than persisting as they often do these days, essentially forever.

As cookies near their shorter end of life, the browser will be able to "ping" a website endpoint at any time that's separately responsible for helping it to refresh its expiring identity authentication cookies.

To do this, the authenticator will send a cryptographic challenge that the browser must sign and return, and the browser can only do so using an unexportable private key that's buried in the hardware of the device that's running the browser. The only thing that can be done with that key is signing cryptographic challenges to prove that the device has the key.

Once the browser returns the challenge properly signed, the cookie provider will refresh the cookies for the domain and the browser will then continue to be able to use the original website without trouble.

The cleverness of this solution is that it minimizes the changes that are required for the rest of the website by concentrating the new authentication scheme in one location. And by using shorter lifetime old-school cookies it achieves compatibility with existing systems while also using the cookies as a form of short-term identity cache so that the system's far far slower crypto hardware is not overwhelmed and is only needed to occasionally refresh the cookies.

Chrome, Firefox and Safari are all at work adding "Device Bound Session Credentials" to their web browser offerings. I'm sure we'll be talking about this more in the future.