Spell-Jacking

Description: This week we look at last week's Patch Tuesday and at the changing cyber insurance landscape. We visit and revisit a collection of major network breaches at Uber, Rockstar Games and LastPass. We look at another significant problem facing 280,000 WordPress users and at a recommended mitigation for the future. We examine the cost to processing performance of the most recent Retbleed security mitigations, and look at Google's very welcome use-after-free vulnerability technology. And after sharing a few pieces of feedback from our listeners, we examine a somewhat surprising consequence of enabling Chrome's enhanced spell check and provide some mitigations.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-889.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-889-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here, as always, the week jam-packed with security news. We'll talk about the breaches at Uber and Grand Theft Auto and whether they're related. We'll also look at Google's very welcome use-after-free vulnerability technology. And as long as we're talking Google, a very important setting you'll want to turn off in Chrome. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 889, recorded Tuesday, September 20th, 2022: Spell-Jacking.

It's time for Security Now!, the show where we cover the latest security news, the latest breaches, ransomware. And this guy right here's in charge, Mr. Steve Gibson. Hello, Steve.

Steve Gibson: Yo, Leo.

Leo: Good to see you.

Steve: Good to be with you again. We were discussing fezzes before we began recording. The years, I mean, I have collected them since I guess you've stuck them on my head when I've come by TWiT in previous years.

Leo: Right, right.

Steve: And I, you know...

Leo: A fez is actually a nice hat because - but it is brimless. So I'm not sure exactly its intent.

Steve: Yeah. I'm a Kangol hat wearer.

Leo: Yeah, because at least a little brim over your eyes to cover your eye...

Steve: Yeah, that's right.

Leo: Yeah.

Steve: So for Episode 889, we're going to talk about something which I don't mean to like over-alarm anyone. This is not a big problem. But it's an interesting information leakage which people should be aware of, if nothing else. And it was named by the people who stumbled upon it "Spell-Jacking." But we've got a lot to talk about. We've got last week's Patch Tuesday. Also the changing landscape of cyber insurance as a consequence of all of the attacks that the world is now being subjected to. And of course we've sort of seen the writing on the wall there. We're also going to revisit sort of a collection of recent major network breaches. Of course, Uber made the headlines everywhere. But also Rockstar Games got hit by the same cretin. And then we have the final update from LastPass about what happened there, which I know that a lot of our listeners had been holding their breath about.

We're going to look at another significant problem facing 280,000 WordPress sites and at probably a useful recommendation for future mitigations of similar things. We're going to examine the cost of processing performance for the most recent Retbleed security mitigations. Someone did, actually an engineer at VMware, they have a department called Performance Engineering at VMware. And he lived up to his department's title. Then we're going to look at Google's very welcome use-after-free new vulnerability mitigation technology, you know, all of these problems that we're finding in browsers, virtually, are use-after-free problems. And to their credit, they've tackled this thing, like this seemingly intractable problem. And then after sharing a few pieces of listener feedback, we're going to take a look at a surprising consequence of enabling Chrome's enhanced spellchecking and talk about some mitigations there.

Leo: Nice.

Steve: So I think we have a great podcast for our listeners. And a sad but sort of like, okay, well, of course, Picture of the Week.

Leo: Aww. Aww.

Steve: Yeah.

Leo: Yeah, sad, but a sign of the times, shall we say?

Steve: It is a sign, indeed. The signage of the times.

Leo: The signage of the times.

Steve: So our Picture of the Week is poignant. We've often shown pictures of the big automated electronic signage which is showing something, well, basically demonstrating that behind it is a unhappy Windows system. And that turned out to be the case last week. This was a large sign, like a roadway-style sign, just acknowledging Her Majesty the Queen, showing 1926-2022. And unfortunately the looks like Windows 10 system that was driving the signage was running low on disk space, so popped up on the sign, again, sort of apropos of what the sign was acknowledging, it says "Low disk space. You're running out of space on this PC. Manage storage," and blah blah blah. Anyway, I just thought it was an interesting coincidence. Life does that to us sometimes.

This is - I decided we have a new name, Leo, for the third Tuesday of every month.

Leo: Oh?

Steve: And that's "Patch News Day."

Leo: Patch Aftermath Day.

Steve: That's right. So last Tuesday Microsoft updated their range of software to resolve a total of 63 flaws, including, well, either one or two publicly disclosed zero-day vulnerabilities, depending upon whether you use Microsoft's more liberal definition of zero-day, which does not depend upon having a vulnerability in active use. But either way, one of the vulnerabilities was being actively exploited. In fact, it was being so widely exploited that researchers with DBAPPSecurity, Mandiant, CrowdStrike, and Zscaler all encountered it in the wild and reported their findings of it to Microsoft.

So its CVE designation is 2022-37969, and it is a "Windows Common Log File System Driver Elevation of Privilege." And given that it's a file system driver which runs in the kernel, any of those many attackers who were apparently using this and having some fun with it were obtaining full system root-level privileges on the machines they were attacking. So the good news is that was happening to unwitting Windows users, and presumably in targeted attacks. So not widespread, but it did come to the attention of four different security firms.

Other than that, there were 30 remote code execution vulnerabilities; 18 elevation of privilege vulnerabilities. So the most problems solved were the two worst classes of problems you can have, which are remote code execution and elevation of privilege. Then there were 16 fixes for the Edge browser, you know, Chromium vulnerabilities; seven information of disclosure vulnerabilities; seven denial of service vulnerabilities meaning you could crash something; and then one of these oh-so-generic security feature bypass vulnerabilities. Okay. And there were some admins reporting problems with group policy management and settings after installing last Tuesday's problems. So Microsoft didn't get away completely unscathed. But there were no reports of anything widespread affecting

typical Windows users. So, you know, a quieter Patch Tuesday than we've been seeing recently.

Lloyd's of London Ltd., of course the famous insurer, has told its global network of insurer groups that new or renewed cyber insurance coverage policies must exclude nation-state attacks as of March 31st, 2023. So about six months from now. Lloyd's cited systemic risk to the insurance market as a whole as a reason for the change, also adding that policies must also exclude losses from war unless there is a separate exclusion for this type of, I guess an exclusion for the exclusion.

And it's not surprising; right? This sort of dialing back on cyber insurance coverage is what we've been expecting. And it's here. Insurance firms are seeking ways to get control of the spiraling cost that they're seeing driven by recent increases in cybercrime, especially ransomware. Nation-state attacks are often most targeted and more about espionage than just casual theft or causing damage. But the consequences do sometimes spill over to do considerable damage to other organizations.

And the NotPetya incident of 2017 appears to be the primary factor driving this decision. We talked about this a couple times. There was a protracted legal battle between Merck and its fleet of insurers over, get this, $1.4 billion that Merck was claiming in damages caused by that attack. And remember, Leo, we talked about it must be that they were just like trying to replace all the PCs that they had. I mean, $1.4 billion.

Leo: There's also loss of business revenue, too, though, remember. I mean, if you're down for five weeks, that could be a billion dollars in a company.

Steve: That could hurt you, yeah. So anyway, as we've noted before, cyber insurance coverage had previously been relying on an "acts of war" exclusion to address incidents such as these, but last year's ruling - oh, I forgot to mention that Merck won their battle. They got their $1.4 billion from their insurers. So basically in response to this, the insurers are saying, okay, let's rethink contracts moving forward here. So there's now an explicit nation-state exclusion. The invasion of Ukraine has stoked fears among insurers that similar cyber exchanges may slip their containment and cause, like, ancillary damage. There's also been at least one smaller incident of this nature, the AcidRain malware that was aimed at Ukraine's Viasat service at the start of the war. We talked about this at the time back in February, or I guess it was March. But that also ended up hitting and affecting, basically it sort of lost containment. It also affected a large wind turbine system in Germany. And it was insured, and so they had to pay up.

So insurers are looking to pull back on risk as companies are increasing their demand for cyber insurance coverage, I mean, like, companies are saying, oh, we never really thought about cyber insurance coverage before, but that seems like a good thing. And Lloyd's has been planning a change of this sort for some time. They've been drafting an assortment of contractual clauses throughout last year, working to clarify when cyber attacks can be considered acts of war and catastrophically damaging enough to be excepted from coverage.

So anyway, I just wanted to mention that this is happening, that costs are going up. It's now estimated, in fact, that the amount of money that insurers are going to be asking for for the coverage that companies wish will be expensive enough that half of companies will say, you know, that's just too much for us. We're going to go without. So insurance is always tough, right, because you're paying for it whether you need it or not until you do need it, and then you sure wish you had it. So anyway, we are seeing a general tightening in the insurance market, basically a raising of operating costs for corporations as a consequence of so many that have been insured and forced major payouts.

Okay. So then, as you mentioned, Leo, we also had a number of major breeches occurring recently. Uber got a lot of attention, mostly I think because the attacker was so brazen. The attacker was just plastering the fact of this attack everywhere. So it wasn't like Uber had any choice, any opportunity to keep it quiet. They suffered, as a consequence, a significant embarrassing network intrusion. Last - I got a kick out of their first posting. Last Thursday, let's see, it was in the evening, 6:25 p.m., they just tweeted, this came from Uber_Comms, saying: "We are currently responding to a cybersecurity incident. We are in touch with law enforcement," which they've actually been, like, talking that up a lot as if maybe that's going to scare people, I guess. And they said they would be posting additional updates as they become available. So that was Thursday.

The day after that, last Friday, in the interest of keeping the lines of communication open, although they still didn't have lots of information after what is arguably a very short time, they said: "While our investigation and response efforts are ongoing, here is a further update on yesterday's incident." They said four things: "We have no evidence that the incident involved access to sensitive user data, like for example trip history. Second, all of our services including Uber, Uber Eats, Uber Freight, and the Uber Driver app are operational. Third, as we shared yesterday, we have notified law enforcement." As I said, they keep talking about that. It's like, okay. "Fourth, internal software tools that we took down as a precaution yesterday are coming back online this morning." So, good that they're keeping lines open.

And then, finally, three days after that, which brings us to yesterday the 19th, we get a significantly more comprehensive update. They said: "While our investigation is still ongoing, we are providing an update on our response to last week's security incident." They said: "An Uber external contractor had their account compromised by an attacker. It's likely that the attacker purchased the contractor's Uber corporate password on the dark web after the contractor's personal device had been infected with malware, exposing those credentials." And then listen, get this: "The attacker then repeatedly tried to log in using the contractor's Uber account. Each time, the contractor received a two-factor login approval request, which initially blocked access. Eventually, however, the contractor accepted one, and the attacker successfully logged in."

So as I understand this, and there's some confusion in the wording of this and in the reporting, but it sounds like because they also later said that they've tightened their multifactor authentication parameters, this appears to have been a brute-force multifactor authentication bypass. And we've seen that happen in the past since the typical multifactor authentication uses six digits, you know, and six digits is a clear compromise between convenience and security. We talked about this years ago when this first surfaced. There is literally for a single challenge a one in a million chance of correctly guessing a given multifactor authentication challenge. But if nothing stops someone from making as many guesses as they wish, as often as they wish, 100,000 guesses would yield a 10% chance of guessing, you know, of getting one correct.

Anyway, so that appears to be what happened is that they acquired username and password credentials, but they were stopped from an easy authentication by multifactor authentication. But because Uber had not configured strong lockout policies, which I'll talk about in a second, that guy was still able to get in. So they said: "From there, the attacker accessed several other employee accounts which ultimately gave the attacker elevated permissions to a number of tools, including G-Suite and Slack. The attacker then posted a message on a company-wide Slack channel" - which they're posting - "which many of you saw, and reconfigured Uber's OpenDNS to display a graphic image to employees on some internal sites."

Leo: Oh, boy, that's mean. You can guess what that image was.

Steve: Uh-huh.

Leo: Oh, lord.

Steve: A graphic image. So they said: "Our existing security monitoring processes allowed our teams to quickly identify the issue and move to respond. Our top priorities were to make sure the attacker no longer had access to our systems; to ensure user data was secure and that Uber services were not affected; and then to investigate the scope and impact of the incident. Here are some of the key actions we took and continue to take." And they list six.

"We identified any employee accounts that were compromised or potentially compromised and either blocked their access to Uber systems or required a password reset. Second, we disabled many affected or potentially affected internal tools. Third, we rotated keys, effectively resetting access, to many of our internal services." Meaning, you know, updated their passwords. "Fourth, we locked down our codebase, preventing any new code changes. Fifth, when restoring access to internal tools, we required employees to re-authenticate. We are also further strengthening our multifactor authentication policies. And finally, we added additional monitoring of our internal environment to keep an even closer eye on any further suspicious activity." You know, and this could sound like an ad for the idea of sticking canaries within a network in order to catch somebody doing something as quickly as possible.

Anyway, they said: "The attacker accessed several internal systems, and our investigation has focused on determining whether there was any material impact. While the investigation is still ongoing, we do have some details of our current findings that we can share.

"First and foremost, we've not seen that the attacker accessed the production, i.e., public-facing systems that power our apps; any user accounts; or the databases we use to store sensitive user information like credit card numbers, user bank account info, or trip history. We also encrypt credit card information and personal health data, offering a further layer of protection. We reviewed our codebase and have not found that the attacker made any changes. We also have not found that the attacker accessed any customer or user data stored by our cloud provider, AWS S3.

"It does appear that the attacker downloaded some internal Slack messages, as well as accessed or downloaded information from an internal tool our finance team uses to manage some invoices. We're currently analyzing those downloads. The attacker was able to access our dashboard at HackerOne, where security researchers report bugs and vulnerabilities." And of course we know HackerOne well. They said: "However, any bug reports the attacker was able to access have been remediated." So no access to anything sensitive there. "Throughout," they wrote, "we were able to keep all our public-facing Uber, Uber Eats, Uber Freight services operational and running smoothly. Because we took down some internal tools, customer support operations were minimally impacted and are now back to normal."

And finally: "We believe this attacker, or attackers, are affiliated with a hacking group called Lapsus$, which has been increasingly active over the last year or so. This group typically uses similar techniques to target technology companies, and in 2022 alone" - that is, this year - "has breached Microsoft, Cisco, Samsung, Nvidia, and Okta, among others. There are also reports over the weekend that this same actor breached

videogame maker Rockstar Games. We are in close coordination with the FBI and U.S. Department of Justice on this matter and will continue to support their efforts. We're working with several leading digital forensics firms as part of the investigation. We'll also take this opportunity to continue to strengthen our policies, practices, and technology to further protect Uber against future attacks."

So I find nothing to fault Uber here. Well, except they acknowledged their multifactor authentication, if that was in fact basically brute forced in order to allow a bad guy in, could have been strengthened, and it now is. Their response was immediate, their communication has been swift and balanced, and I would imagine that their forensics team is likely glad to be able to get some sleep after a handful of probably very sleepless nights. That they are users of HackerOne speaks well of them, and they appear to have a well-running security component to their IT systems since their continuous auditing of systems were able to provide them with a lot of relevant information when those audits were queried.

They also identified a weakness, as I mentioned, in their multifactor authentication configuration, and they're tightening up. So I guess a takeaway for everyone would be to think about the fact that just as username and password authentication should lock out for some period of time after a reasonable number of failed attempts, the same remains true even after multifactor authentication has been added. That is, there's really no good reason for an authentic user to fail five times in a short period of time to properly authenticate themselves. Clearly, if they've got multifactor authentication, they've got something which is generating six digits for them to enter. And, you know, you could understand a typo or a timeout once or twice. But not many times in a short period of time. So the lesson may be here that just adding multifactor authentication doesn't mean that you can then decide you no longer need other lines of defense like short, like some time-limited auto resetting lockout of an account.

And Rockstar Games. Of course they are famously the publishers of Grand Theft Auto. And what happened to them was a massive leak of videos for the not-yet-released Grand Theft Auto 6, which I guess is like a highly anticipated superhot topic. Uber said: "There are also reports over the weekend that this same actor breached videogame maker Rockstar Games." And it's certainly believable that this is the same guy or gang, although I wasn't able to ascertain why they believe that.

Leo: Oh, because when he released the videos, he used the handle, something like "Uberhacker."

Steve: Okay.

Leo: "I hacked Uber" or something like that. Which, you know, that's not probative. But, you know.

Steve: Yeah, that's it. Well, but it is - what I was going to say was that he does seem to be liking making a splash.

Leo: As Lapsus$ is. I mean, that's what Lapsus$'s motivation mostly seems to be is getting attention.

Steve: So this guy put out more than 90, nine zero, videos showing game play from the upcoming GTA 6 on Sunday. Reports indicated that he's believed to be a teenager, but I was also unable to learn what backs that up. I just saw that in passing. And in this instance he was using the handle "Teapot." And he said he plans to leak more game play and even some of the game's source code. Although I wonder about that because he doesn't seem like the kind of person who holds anything back. And so maybe he doesn't have source code. Maybe he does. We'll see. Anyway, Rockstar Games confirmed that the videos were authentic to Bloomberg's main games reporter, but has not commented on the news of the hack or if the hacker did indeed steal the game's source code.

So there's breach number two. Breach number three is coming back to update from our previous coverage. And that is LastPass. Last Thursday LastPass published their final official port-mortem, exactly three weeks following their initial breach disclosure, which of course left some aspects of the attack unknown because it was unknown at that time. So here's the final word from their CEO.

He said: "On August 25th, 2022, we notified you about a security incident that was limited to the LastPass Development environment in which some of our source code and technical information was taken." He said: "I wanted to update you on the conclusion of our investigation to provide some transparency and peace of mind to our consumer and business communities.

"We have completed the investigation and forensics process in partnership with Mandiant. Our investigation revealed that the threat actor's activity was limited to a four-day period in August 2022. During this timeframe, the LastPass security team detected the threat actor's activity and then contained the incident. There's no evidence of any threat actor activity beyond the established timeline. We can also confirm that there's no evidence that this incident involved any access to customer data or encrypted password vaults. Our investigation determined that the threat actor gained access to the Development environment using a developer's compromised endpoint. While the method used for the initial endpoint compromise is inconclusive, the threat actor utilized their persistent access to impersonate the developer once the developer had successfully authenticated using multifactor authentication."

Okay, now, that's interesting. It appears that we have another incidence of multifactor authentication bypass, essentially. We don't know enough about the way LastPass has set things up. But it must just be that the bad guys obtained an authenticated session token from a successfully logged-in endpoint. At least that's sort of what it feels like. It does remind us, though, that simply adding multifactor authentication isn't any sort of universal cure.

So they said, or he continues: "Although the threat actor was able to access the Development environment, our system design and controls prevented the threat actor from accessing any customer data or encrypted password vaults." And of course that's what we were hoping to hear, and it looks like we are. "First," he said, "the LastPass Development environment is physically separated from, and has no direct connectivity to, our Production environment." So that was one big question we had when we talked about this three weeks ago is was the Production environment ever in danger from a breach of the Development environment? And he's saying no. They are deliberately physically separated. So that's just great. That's exactly what we want. Great news.

He says: "Secondly, the Development environment does not contain any customer data or encrypted vaults. Third, LastPass does not have any access to the master passwords of our customer vaults. Without the master password, it's not possible for anyone other than the owner of a vault to decrypt vault data as part of our Zero Knowledge security model," he says. So that's what we would hope. And that's what we were talking about before. He's making the point that I made three weeks ago, which is that mistakes can

happen to anyone. But as long as the security architecture of the system is designed for "Trust No One" operation, all of their users will be completely protected.

And providing another example of proper design, the CEO explained. He said: "In order to validate code integrity, we conducted an analysis of our source code and production builds and confirm that we see no evidence of attempts of code-poisoning or malicious code injection." And he said: "Developers do not have the ability to push source code from the Development environment into Production." Again, really good isolation there. He said: "This capability is limited to a separate Build Release team and can only happen after the completion of rigorous code review, testing, and validation processes." So from what he's saying, it really does sound like they have built this system properly, in a significant chain.

He said: "As part of our risk management program, we have also partnered with a leading cyber security firm to further enhance our existing source code safety practices which includes secure software development life cycle processes, threat modeling, vulnerability management, and bug bounty programs. Further, we have adopted enhanced security controls, including additional endpoint security controls and monitoring. We've also deployed additional threat intelligence capabilities, as well as enhanced detection and prevention technologies in both our Development and Production environments."

And again, this is what I was saying last time we talked about it was all of that is what we would like to have happen. That is, their existing architecture was good. They realized it could be made better. And that's what they have done. So I think that, you know, those who have chosen to remain with LastPass should have every reason to feel that LastPass is a competent caretaker of their users', of their own pre-Internet encrypted data. As I was thinking about that, I realized that our long-term listeners will recall that early acronym we developed. You know, remember PIE.

Leo: The first one was PEE. I'm glad you changed it for PIE. Do you remember that?

Steve: Oh, yeah. That was Pre-Encryption Environment or something? You're right. Good memory, Leo.

Leo: Yeah, Pre-Entry Encryption or something. And you realized that PIE might be better, yeah.

Steve: PIE was better than PEE, indeed.

Leo: Pre-Egress Encryption maybe.

Steve: That's, whoa, thank you whoever was listening.

Leo: That was me. Pre-Egress Encryption.

Steve: Oh, no kidding. You remembered.

Leo: You know, I know that I don't remember anything that happened recently. But boy, I've got a good memory for that old stuff.

Steve: Welcome to the club. So we have a CVSS of 9.8 for WordPress. Last time we talked about a big vulnerability hitting WordPress I came away suggesting that anyone who's using WordPress in anything other than its barest out-of-the-box essential configuration that is, anyone who has added any of the gazillion tantalizing WordPress add-ons ought to give serious thought to running a third-party application firewall on their site. There are three or four of those, but one stands out, and that's WordFence. It's the one that we keep referring to since they appear to be most on top of this particular chunk of the industry. They're the ones who are discovering problems more than any of the others. I remember I went, like, looking for other WordPress add-on companies, and I found that there were others, but they weren't nearly as active as these guys. Anyway, as we know, WordPress matters; right? It's just shy of 40% of the Internet's websites.

So to that end, WordFence last week put out a report which serves as a perfect case in point. The report is titled "PSA," as in Public Service Announcement, "Zero-Day Vulnerability in WPGateway Actively Exploited in the Wild." And they said: "On September 8th, 2022, the Wordfence Threat Intelligence team became aware of an actively exploited zero-day vulnerability being used to add a malicious administrator user to sites running the WPGateway plugin." They said: "We released a firewall rule to Wordfence Premium, Wordfence Care, and Wordfence Response customers to block the exploit on the same day, September 8th." I don't know what those three things are, but they have some sort of a product lineup.

They said: "Sites still running the free version of Wordfence will receive the same protection 30 days later." Well, you might as well not have it if you don't get it for a month. Anyway, the Wordfence firewall, they said, has successfully blocked over 4.6 million attacks targeting this vulnerability against more than 280,000 sites in the past 30 days.

Okay. So this WPGateway plugin is a premium plugin tied to the WPGateway cloud service, which offers its users a way to setup and manage WordPress sites from a single dashboard. Part of the plugin's functionality exposes a vulnerability that allows unauthenticated attackers to insert a malicious administrator into the site.

So they said: "We obtained a current copy of the plugin on September 9th and determined that it is vulnerable." Okay. So their model is they've got these application firewalls deployed. They see something happening. And in WordPress, since it's all PHP based, they're going to be seeing some odd-looking query that's being made. So that brings it to their attention. They look at what plugin the PHP query is targeting, and in this case the next day they got a copy of the plugin and analyzed it to see what was going on. So they said: "We determined that it is vulnerable, at which time we contacted the plugin vendor with our initial disclosure." They said: "We've reserved vulnerability identifier CVE-2022-3180 for this issue."

So they said: "This is an actively exploited zero-day vulnerability, and attackers are already aware of the mechanism required to exploit it." Clearly, because that's how it came to their attention. They said: "We're releasing this public service announcement to all of our users. We are intentionally withholding certain details to prevent further exploitation. As a reminder, an attacker with administrator privileges has effectively achieved a complete site takeover."

They said: "If you are working to determine whether a site has been compromised using this vulnerability, the most common indicator of compromise is a malicious administrator with the username of 'rangex,'" R-A-N-G-E-X. They said: "If you see this user added to

your dashboard, it means your site has been compromised." So they also mentioned that, if you're unable to get an update to this WPGateway, remove it from your system, or take some action to prevent it from being accessed because that's the way these bad guys are compromising so many WordPress sites.

So personally, I'm no longer running any WordPress sites. I was for a while to maintain a blog, which I infrequently posted to. But if I were doing so today, I would - first of all, I tend not to be adding lots of bells and whistles to my site. I'm happy with more of the bare minimum functionality. But if you had lots of sites that you're using this cloud-based dashboard in order to manage them, you should know that that's a problem. I'd take a look at these WordFence people. They seem to be good folks. Although again, as I said, there are a bunch of similar offerings because this sort of add-on is useful. And in fact, as we'll see, it's exactly a web application firewall which led the people who created it to today's podcast topic.

Okay. This I loved. Two months ago, on July 19th, Security Now! Episode 880 was titled "Retbleed." As we discussed at the time, it's another of the ongoing and ever-evolving speculative attacks against the Intel and AMD microarchitectures. Until security researchers began looking closely and developed the Spectre and Meltdown attacks, Intel and AMD were both happily inventing and incorporating all sorts of clever tweaks into the execution path of their processors for the sake of improving their performance. In essence, all of these performance-tuning tweaks involve having the processor's microarchitecture learning something about the code it's executing.

Unfortunately, well, the good news is that allows it to correctly anticipate what's likely to happen again. But it allows the microarchitecture to be probed by software that understands that there is basically a trail of breadcrumbs following behind the code. When this all works, it allows the code to execute much more smoothly. But the big question we always ask and ponder, but so far we haven't received much of a clear answer to because Intel doesn't want to tell us, is what's the performance impact of turning off these features? What happens if we disable this microarchitectural optimization in the interest of enhanced security?

Friday before last, on September 9th, a VMware engineer in VMware's Performance Engineering department posted the results of his analysis of exactly this into the Linux Kernel Archive list under the subject "Performance Regression in Linux Kernel 5.19." Which is the, you know, just recently updated and released kernel. And it's the one that incorporates treatment for Retbleed. He wrote: "As part of VMware's performance regression testing for Linux Kernel upstream releases, we have evaluated the performance of Linux kernel 5.19 against the 5.18 release, and we have noticed performance regressions in Linux VMs on ESXi as shown below."

Get this: Computing performance has fallen 70, seven zero, percent. Network throughput, down 30%. Storage bandwidth down 13%. He said: "After performing the bisect between kernel 5.18 and 5.19, we identified the root cause to be the enablement of IBRS mitigation for spectre_v2 vulnerability by commit," and then he's got the hex number of the commit that added that to the kernel. And it's titled "x86/bugs: Report Intel Retbleed vulnerability." He said: "To confirm this, we have disabled the above security mitigation through kernel boot parameter" - and it's spectre_v2=off - "in 5.19 and re-ran our tests and confirmed that the performance was on par with 5.18 release."

Okay. So the IBRS to which the engineer refers stands for "Indirect Branch Restricted Speculation," which Intel describes as an indirect branch control mechanism that restricts speculation for indirect branches. Doing that is necessary, as our Retbleed podcast two months ago explained in detail, to prevent a surprising rate of data exfiltration from otherwise completely secure operating systems. Consequently, since that would be bad,

the Linux Kernel starting with 5.19 does so by default unless it's prevented from doing so with a kernel boot override parameter.

And to their dismay, what the VMware Performance Engineering folks discovered was that, compared to the immediate preceding Linux kernel 5.18, 5.19, as I said, sees a 70% reduction in the performance of compute intensive tasks, a 30% reduction in network throughout, and 13% reduction in mass storage performance. I have a link in the show notes to VMware's Linux Kernel Archive posting for any Linux aficionados among us who will want all the details on this and on the specific benchmarks which were run, and how it was done. The guy who posted this from VMware provided complete details.

My feeling about all of this has not changed from its first appearance. And Leo, it's the opinion you and I had from the beginning. End users have never had much, if anything, to worry about from any of these subtle microarchitectural attacks. It's the big datacenter cloud server guys running many different virtual machines across a heterogeneous client population that does have cause to worry. So if I were a Linux hotshot, I'd be disabling all of these Spectre-ish mitigations and free up my processors to run with as much "intuition" about the code they are running as possible. The only danger you would face would be a cross-process information bleed. And in order to have a cross-process information bleed, you've got to have something running in your machine which is already running in your machine which is able to perform this sort of operation.

Now, having said that, I do recall that we talked about Retbleed being operable from code in a browser. And of course that sort of blurs this boundary. But typically it has to run for a long time to get any information, and it's got to find out where the information is, blah blah blah. So again, my sense is end users really don't have much of a concern. If you've got no problem at all under the latest Linux kernel from a performance standpoint, then obviously leave the stuff enabled. But recognize, depending upon which distro you're using and which Linux kernel it's got, and what mitigations it has by default, there is a huge difference in performance when you turn these Spectre mitigations on versus off. And we finally have some what look like very good numbers to support that. So again, I guess I would say it's an individual preference, but you need to understand that there is some serious performance hanging in the balance.

And I also wanted to tell everyone about some very encouraging news from Google's Chromium team. Okay. So the trouble with my doing that is that it really gets down into some technical weeds, and I did not want to devote an entire podcast to tackling a single complex topic that most of our listeners won't care that much about. Mostly you're going to want the headline. So this involves pointer reference counting and what they call poisoning quarantined pointers before the pointers' release. It is some seriously cool, but also complex, pure computer science. But again, all we really need to know is that the single most troublesome aspect of the Chromium code base, that is, those ubiquitous use-after-free errors and their exploitation, are finally going to be resolved. So I do want to share some of what Google explained.

So here's how they begin their explanation. They said: "Memory safety bugs are the most numerous category of Chrome security issues; and we're continuing to investigate many solutions, both in C++ and in new programming languages. The most common type of memory safety bug is the use-after-free. We recently posted about an exciting series of technologies designed to prevent these. Those technologies, collectively *Scan" - star as in a wild card scan - "are very powerful, but likely require hardware support for sufficient performance." In other words, we can't have that today.

They said: "Today we're going to talk about a different approach to solving the same type of bugs. It's hard, if not impossible," they wrote, "to avoid use-after-frees in a non-

trivial codebase. It's rarely a mistake by a single programmer. Instead, one programmer makes reasonable assumptions about how a piece of code will work, then a later change invalidates those assumptions. Suddenly, the data isn't valid as long as the original programmer expected, and an exploitable bug results."

They said: "These bugs have real consequences. For example, according to Google's Threat Analysis Group" - their TAG team - "a use-after-free in the Chrome HTML engine was exploited earlier this year by North Korea. And as shown in the percentage bar chart below, half of all known exploitable bugs in Chrome are use-after-frees." And I have a chart in the show notes which I grabbed from their blog posting where these blue bars in this percentage bar chart are shown pretty much, especially later on, that's every quarter from the second quarter of 2015 through the first quarter of 2021. And certainly from around 2019 on, those blue bars have represented about half of the total bugs that Chrome has seen. And of course we're talking about them all the time on the podcast.

Okay. So then they introduce the concept of what they call their MiraclePtr, you know, Ptr. They said: "MiraclePtr is [modestly named] a technology to prevent exploitation of use-after-free bugs. Unlike aforementioned *Scan technologies that offer a non-invasive approach to this problem, but would require hardware that doesn't yet exist, MiraclePtr relies on rewriting the codebase to use a new smart pointer type, which is raw_ptr. There are multiple ways to implement MiraclePtr. We came up with around 10 algorithms and compared the pros and cons of each. After analyzing their performance overhead, memory overhead, security protection guarantees, developer ergonomics, et cetera, we concluded that the BackupRefPtr was the most promising solution." So that was the one from around 10 that they end up choosing as the implementation for their so-called MiraclePtr.

They said: "The BackupRefPtr algorithm is based on reference counting. It uses support of Chrome's own heap allocator, known as PartitionAlloc, which carves out a little extra space for a hidden reference count for each allocation. Raw_ptr increments or decrements the reference count when it's constructed, destroyed, or modified. When the application calls free or delete, and the reference count is greater than zero, PartitionAlloc quarantines that memory region instead of immediately releasing it. The memory region is then only made available for reuse once the reference count reaches zero. Quarantined memory is poisoned to further reduce the likelihood that use-after-free accesses will result in exploitable conditions."

In other words, typically what we've seen is the pointer which has been released is pointing to something useful like still into the operating stack, where real damage could be done. So they are deliberately writing something illegal into that pointer so that if someone had access to it, it wouldn't be pointing to anything useful. And they said: "In the hope that future accesses lead to an easy-to-debug crash, turning these security issues into less dangerous ones."

And, they said: "We successfully rewrote more than 15,000 raw pointers in the Chrome codebase into raw_ptr, then enabled BackupRefPtr for the browser process on Windows and Android, both 64 bit and 32 bit, in Chrome 102 Stable." And note that we're all running 105 now. So it's been in there for a while. They said: "We anticipate that MiraclePtr meaningfully reduces the browser process attack surface of Chrome by protecting around 50% of use-after-free issues against exploitation. We are now working on enabling BackupRefPtr in the network, utility, and GPU processes, and for other platforms. In the end state, our goal is to enable BackupRefPtr on all platforms because that ensures that a given pointer is protected for all users of Chrome."

So after that they continue, that's about like - that's just the first piece of this blog posting because they get into a deep discussion of implementation overhead, which comes down to the overhead of a four-byte reference count for each pointer. But the

bottom line is hats off to the Chromium folks for not simply chasing their tails endlessly under the fantasy and fallacy that they're eventually going to find all of the use-after-free bugs. We all know that as new code is being written, and as old code is being modified, as many new bugs are being introduced as old bugs are being found and eliminated.

That's, you know, just take a look at that chart. You can see that there's actually no progress being made along those lines. If anything, there's more blue today than there was back in 2015 when they began doing this. So yay to them actually tackling this and developing some new technology. The only people who will be made unhappy by this news are those who've been making their living off of discovering new ways to break into Chrome. Life promises to be much more difficult for them, which is wonderful.

I have a couple little bits of Closing the Loop from our listeners. Oh. Peter G. Chase, and actually a handful of our sharp-eyed listeners, looked at that Picture of the Week. Well, I'll say what Peter said first. He said: "The Picture of the Week looks a lot like they're actually bypassing the meter. That looks like a meter box." And that hadn't occurred to me. But it did to a number of our listeners. And of course that's even more interesting; right? So the idea would be that one of those big round-faced electric meters has essentially the same sort of feet sticking out of it, prongs, that two bar fuses would. And so what's actually happened is that the people who are using that power - thanks for having it on the screen, Leo, right. The people who are using that power have bypassed the meter, which would normally be plugged into the front of that, thus monitoring the amount of electricity being used by those people. So if so, it looks like they're getting free electricity. Anyway, thank you, all of our listeners, for pointing that out.

Dan Taylor said: "Hi, Steve. Years ago you mentioned PEGASYS TMPGEnc." And I recall that. He says: "As the video editing software you like and use. It's been at least 10 years since my first purchase. I have three different packages from them. The latest is Video Mastering Works 7," he says, "and I love it! I just thought I'd say thanks for steering me in their direction so long ago." And it's funny, Leo, I mean, you and I were messing with media back then. I remember that TMPGEnc, for me, I didn't have any video editing software, but that was my go-to MPEG-2 encoder back when MPEG-2 encoders were like littered with all kinds of bells and whistles in the UI that allowed you to, like, tune it in order to get the best compression and image appearance.

Leo: I probably wasn't using Windows at the time.

Steve: Yeah, maybe not, although...

Leo: I wouldn't know.

Steve: And I didn't know. But they're still there.

Leo: Nice.

Steve: As Dan says. And they've got a bunch of nice-looking software.

Leo: Nice.

Steve: Ed, oh, boy, Grigoleit.

Leo: Grigoleit. I don't know.

Steve: He says: "I'm a long-time Security Now! listener, but I do not recall hearing you mention the following benefit in the past, and it may be useful to some of your listeners. I am a CISSP, and I must maintain my certification by completing CPE credits. I've been submitting my Security Now! attendance as a qualifying Webinar (1.5 CPEs) for several years, and these submissions have always been approved. Best. EdG." So Ed, thank you for sharing that, for any of our listeners for whom that may not have occurred and who may also be needing CPE credits. Cool that the podcast is a source for those.

Leo: Yeah, that's great.

Steve: Bob O'Brien said: "In an alternate universe where everyone used SQRL, would that stop/thwart phishing attacks because they wouldn't have one's master key or encrypt the login handshake?" So, okay, Bob. SQRL, I realized that when I talked about this last week I didn't explain why or how SQRL solves the problem. It solves it in a couple ways. First of all, the key being generated by SQRL is tied to the domain name. So if you're at a phishing site with a bogus domain name, SQRL will generate a key. But it won't be the right key. It'll be a SQRL key for the phishing site, rather than the SQRL key for the site you think you're logging onto. So you automatically get that protection.

But that client-provided session, CPS feature that I mentioned, that's actually an additional layer of protection. The reason that all of these technologies are prone to man-in-the-middle attacks is that the man in the middle is literally monitoring all of the traffic in both directions, unencrypting it, sniffing it and then reencrypting it. The way SQRL's client-provided session system works is that the SQRL client itself, which is separate from the browser, initiates a connection to the server in order to negotiate credentials. So it automatically bypasses a man in the middle, going around it, talking to the proper server, even if it had the proper domain, which in a phishing scenario it would not. So you sort of get multiple layers of protection from that. Anyway, thanks for the question and for pointing out that I hadn't really explained it.

And Leo, you're going to love this one. Get a kick out of this. A tweet from our favorite Dutch ex-regulator, Bert Hubert.

Leo: No. He heard us?

Steve: Yup. He said...

Leo: I'm sorry, Bert.

Steve: Bert said: "International attention for my resignation as regulator of the Dutch intelligence and security services in the Security Now! podcast at minute 58:40."

Leo: Wow.

Steve: "Thank you, Steve Gibson @SGgrc and Leo Laporte."

Leo: I didn't mean to make fun of your name, I'm sorry.

Steve: No, I think he really appreciated that...

Leo: He needed the attention. That's the whole point of resigning; right?

Steve: Exactly. And the reason he blogged about it was to say this is my beef about this legislation and the problem it has.

Leo: Good.

Steve: And so we helped him shine a big spotlight on it. So Bert, you're welcome. And one last bit.

Leo: Yes.

Steve: Bill Sempf, he tweeted: "Another winner, Steve. Second time now that you have recommended a book series that my wife loves."

Leo: Is that Silver Ships?

Steve: He's clearly talking about the Silver Ships. And there is, in the third book, there's something I need Lorrie to read, my wife. And I think she'll enjoy getting to that point. But there's an alien which this author creates. Oh, they are so wonderful, these aliens. So enough said.

Leo: Can't wait. I'm reading William Gibson's "The Peripheral" because they're making a TV show out of it, which is coming next month. And I thought, I've had this book for ages. I should read it. Because I love William Gibson. It's not a good audiobook. I can't follow it at all. I guess his stuff is so visual that you kind of have to see it in your head. And it's just not working for me. So two days from now, Silver Ships.

Steve: Good. Good, good, good. You won't regret it. Is "The Peripheral" going to be a movie or a cable...

Leo: A series. A series on, I want to say HBO or Apple TV. I can't remember which.

Steve: Oh, good, good, good. So we don't have to have commercials.

Leo: Yeah. Did you read that? I know you're a William Gibson fan. He wrote "Neuromancer," which is the cyberpunk novel.

Steve: Of course. Of course.

Leo: And that one I really loved. But I think it might be better for me to watch the show.

Steve: Yeah, I think I'll do that, too, because my reading has been hijacked.

Leo: Yeah, no kidding.

Steve: Oh, it's so good.

Leo: Yeah. You're going to be on Silver Ships for the rest of your life. "The Peripheral" is Amazon Prime Video. And I'm just looking at it right now. Looks pretty good. Looks like it'll be a good show, anyway. Great, high sci-fi. Lot of VR.

Steve: We need some. We need some.

Leo: Yeah, very futuristic. So it should be interesting. Okay, Steve. I am ready to talk about Spell-Jacking.

Steve: Okay. So again, I don't want to overstate this, but this is something I know a percentage of our listeners will really care about. So it comes from a company named Otto, O-T-T-O. Actually it's O-T-T-O hyphen J-S dotcom, as in Otto JavaScript. It's not a company we're talked about before. And they look like an interesting group. They describe themselves with a tag line which is kind of interesting: "Built on innovating cybersecurity and protecting modern freedom." Okay.

Leo: What?

Steve: Well, that's a mix. I know. And they explain. They said: "Otto was created by a team of innovators with 30 years of combined expertise in cybersecurity, third-party JavaScript, and MarTech/AdTech. Before launching Otto we worked on and created solutions for some of the world's largest banks, media companies, and even the U.S. government."

Okay. So what they appear to have is some powerful add-on technology for helping to manage what exactly a sophisticated modern website is doing, which when you think about it may not always be obvious or necessarily under control of a site's designers. Consider the challenges when all sorts of third-party libraries with their own complex dependency chains are being pulled into a website visitor's browser on the fly. And then to that mess add advertising, which may also be bringing along its own scripting.

Anyway, the bullet points for them says: "Stop Client-Side Attacks. Plug Otto into your application security suite and protect your supply chain. Traditional WAFs (Web Application Firewalls), API security, DDoS, and bot protections are all essential components of your AppSec suite, but they don't protect the client-side gap in your third-party supply chain."

And so they said: "Visibility. Otto provides continuous monitoring and analysis of first, third, and Nth-party script behavior and vulnerabilities," which is like - it's what I was talking about before where who knows what scripts the scripts are loading the scripts for.

Also "Protection: Advanced Malware Guard and Script Shield defend your website from trojans, phishing, malicious code injections, Magecart, and client-side attacks with real-time integration." And "Control: Take control over client-side application security with precision Script Policy and Dynamic CSP automation."

Anyway, so given that these guys are deep into watching and analyzing exactly what a user's browser is doing, it's not surprising that they found, that they would have been the ones to catch some unexpected and unwanted behavior from Chrome and Chromium-based browsers, specifically Chrome and Microsoft's Edge. So what they found: Google's Chrome and Microsoft's Edge Editor, which is of course Chromium-based, the Enhanced Spellcheck features are phoning home to expose their users' in-the-clear passwords, usernames, email addresses, dates of birth, Social Security numbers, and so forth. Basically anything entered into form fields that is not recognized locally by their spellcheckers is sent back, as entered, in the clear to Google or Microsoft to request spelling suggestions from their remote recommendation engines.

Now, that's obvious; right? I mean, that's what Enhanced Spellcheck is. Except that there by default are no limits on what is being sent back. And so it includes anything not in the local dictionary, which certainly better be your passwords. So again, it's not the end of the world. But in an environment where we want and expect our browsers to locally hash our passwords so that they never leave our browsers in the clear, sending every one of our pre-hashed passwords to Google or Microsoft without our knowledge or permission seems like something that someone should have thought about and should be preventing. And if the passwords, as I said, that we're using are present in our spellchecker's local dictionary, so that they're not being sent to Google or Microsoft, well, then we have bigger problems because you don't want to be using passwords that are in your own local language.

So I have a screenshot in the show notes of the Chrome browser's POSTed query and Google's reply when a user was logging into Alibaba with the test password "sharepassword*123." And you can see that being sent in a little JSON blob on the left. And Google replied with the spellcheck suggestion "share password." Which is what you would expect it would recommend. But in the process the user's password, with no encryption of any sort, went to Google, and they got it. And the same thing happens if you're using Microsoft Edge. So again, not the end of the world, but not a good look. Also you can argue that it's never useful to run someone's password or email or Social Security number through spellcheck. All it's going to do is it's going to, like if it did correct your password, it would mess it up.

Leo: It would be wrong, yeah.

Steve: So, yeah.

Leo: That's terrible.

Steve: So Otto-js's co-founder and CTO Josh Summitt, he discovered the spellcheck leak while he was testing the company's scripts behavior. And he explained, he said: "If 'show password' is enabled, the feature sends your password to their third-party servers. While researching for data leaks in different browsers, we found a combination of features that, once enabled, will unnecessarily expose sensitive data to third parties like Google and Microsoft. What's concerning is how easy these features are to enable and that most users will enable these features without realizing what's happening in the background." And as we'll see in a minute, I was guilty of having done so at some point in the past. I checked, it was like, oops. Ouch.

So this unsuspected and inadvertent leakage could obviously lead to serious trouble for consumers and major industries when it comes to privacy, data protection, and client-side security. Not to mention that it's a clear violation of HIPAA and similar privacy regulations which rigorously restrict third-party access to sensitive private information. In principle, when these features are active, any terminology, such as medical conditions, which are not known to the local spellchecker, will be shared with Google or Microsoft. And if your browser is also logged into either company's websites, since the POST query is being made to their servers, your logged-in session ID cookie will accompany the aberrant spellcheck query. So they also know who you are. Not that there's anything wrong with Google or Microsoft receiving this. Not that there's anything, any reason to believe they're logging it or collecting it or doing anything. But they're receiving it from you.

So Otto-js noted that five websites and services of concern were - and they didn't do an extensive test, but they found out that they noticed that Office 365 does this. Alibaba's Cloud Service does this. Google Cloud Secret Manager site does this. AWS Secrets Manager did this. They added an update that AWS had already mitigated the issue. And LastPass site was doing this when they were notified. They noted that both AWS and LastPass had immediately and already fully mitigated the issue. The Otto guys said that LastPass was the first to respond to outreach and first to fully mitigate the risk. They quoted LastPass's Christopher Hoff, who's their Chief Secure Technology Officer, saying: "It is disconcerting that customers can inadvertently expose confidential data by enabling innocuous browser features and not understand that anything they type including passwords could result in that data being sent to third parties."

And in the show notes I grabbed a snippet of the LastPass login page now, after they made the fix. And you can see highlighted there in the body tag, it opens the body tag. It says spellcheck="false" in double quotes.

Leo: Yay.

Steve: That's all that's necessary to shut this behavior down on that page. But nobody is doing it. The Otto-js researchers created a demonstration video to illustrate how spell-jacking could easily expose a company's cloud infrastructure - servers, databases, corporate email accounts, and password managers. In the video, an employee had enabled enhanced spellcheck features when he was using that to create a document. But that feature remains enabled for all sites. And the user then visits after that, goes to his enterprise database credentials, and shows them being spell-jacked and being sent to Google when he clicks on the Show Password button in order to verify that he entered his password correctly.

So the video uses a common scenario in the workplace to illustrate how easy it is to enable the browser-enhanced spellcheck features, and how an employee could inadvertently expose their company without ever realizing it. Most CISOs would be

extremely alarmed to learn that their company's administrative credentials were unwittingly shared in cleartext with a third party, even one they generally trust such as Google or Microsoft.

Otto-js tested more than 50 websites and sorted 30 of those into a control group spanning six different categories of websites which people use frequently and which have access to highly sensitive Personally Identifiable Information (PII) data. Okay. So remember that by default, all data entered into forms that's not recognized by the browser's local spellcheck dictionary will be sent to retrieve remote suggestions. But passwords will not be sent until and unless the user clicks the "show password" button. So that's some relief there.

In Otto's testing, five websites per category were selected based on top ranking in each industry. They were testing to create some benchmark for how much exposure might be occurring. So the six categories they selected were online banking, cloud office tools, healthcare, government, social media, and eCommerce. Of the reference group of 30 websites tested, 96.7% did send Personally Identifiable Information back to Google and Microsoft. Only one did not. 73% sent passwords when "show password" was clicked. But those not sending passwords only didn't because they lacked the "show password" feature on their site. So otherwise they would have.

And interestingly, the only control group website that had mitigated the issue, there was one out of those total of, what was it, 30 sites, yeah, one out of 30, thus 96.7. The one that had was Google. So Google was aware of this and didn't want that Personally Identifiable Information and possibly passwords even being sent back to them. Though Google did mitigate the issue for email and some services, they have not mitigated it for some of their other services like Google's Cloud Secret Manager. Also, Auth0, a popular single-sign-on service, was not in the control group, but was the only website other than Google which they found that had correctly mitigated the issue. So props to Auth0.

Anyway, so the point is the knowledge of this is out there, but it has not yet been receiving wide attention, which is one of the reasons I wanted to put it on everyone's radar today. As I noted with the example of LastPass's mitigation, companies can mitigate the risk of sharing their customers' Personally Identifiable Information by adding spellcheck="false" to the containing page, or to all input fields, although this might create problems for users who want spellcheck, I suppose. Alternately, that override could just be added to the form fields that might contain sensitive data like username, password, and so forth. Or please describe your medical condition.

Fortunately, the enhanced spellcheck feature is not enabled by default. But once it's been enabled, it remains so. I was curious about my own settings. So I went to Settings in Chrome and entered "enhanced spell," that's as much as I needed to put into the search bar. And what do you know? It turns out I have enhanced spellcheck on.

Leo: Of course you do.

Steve: I'm sure I turned it on.

Leo: It's just, if you added all your passwords to the dictionary, then it would always be okay; right?

Steve: That's right. Then it will never, yes, it will never need to go ask Google or Microsoft...

Leo: There's another solution.

Steve: If they have a suggestion for an improvement.

Leo: Because it's always spelled right.

Steve: You mistyped your password. It looks like gibberish. So it was enabled in my instance of Chrome. I have no idea when I may have turned it on. But it's been on ever since. Google makes it clear what's going on. They say right there under the option: "Text you type in the browser is sent to Google." But you'd sort of think they wouldn't send your password field data. But they do. Anyway...

Leo: Well, that's up to the people, as you showed, who have password fields to make sure spellcheck="false."

Steve: Yes. Yes.

Leo: I think, I would hope everybody would now go, oh, yeah, I guess we need to do that; right?

Steve: Let's hope that happens.

Leo: I'm looking at Firefox, and it says check your spelling as you type. And I have it checked. So it probably is default.

Steve: Yeah.

Leo: In all likelihood.

Steve: So while simply turning off enhanced spellcheck will resolve all concerns, Otto-js does offer a free Chrome extension that will alert users when they are visiting a website that has the risk of data leaks caused by enhanced spellcheck. Now, the problem is all websites do, like, virtually. So it may be a little annoying. Maybe it only pops up when you're on a password page, and so it's just going to remind you of that.

Leo: I wonder where Firefox sends my password.

Steve: Yeah, that's a good question.

Leo: Another question; right? I mean...

Steve: Yeah.

Leo: This might be something you want to turn off there, as well. The operating system does spellchecking, most operating systems. Right? So you probably don't need the browser to do it.

Steve: True. Although it makes sense that Chrome would be bringing along their own just because they're Chrome.

Leo: Yeah, yeah, yeah.

Steve: And, you know, and they want it cross-platform and the same for everything and so forth.

Leo: Right.

Steve: So anyway, for what it's worth, I've got the links at the very end of the show notes for today. There's something called Otto-js ShopSecure, which they call "free browser protection for shoppers." I don't know why it's not for everybody. And then also Otto-js Developer tools, which they said is "free runtime script testing tools," which might be of use or interest to our more techie users. So I've not looked at either of those. But this just popped up on my radar that personally identifiable information was by default going to Google and Microsoft. May not be something you care about. Maybe not be something you're exposed to if you never turned on enhanced spellcheck. But I'm sure there's a bunch of listeners who are saying, ooh, crap, I didn't know.

Leo: As far as I can tell, Firefox brings its dictionary with it. So this is a very Google-y thing to do. Oh, no, no, no. We're not going to have an on-disk dictionary to do spellcheck. We're going to send it back to the server and let them do it. I think Firefox just is not a problem because they use their on-disk dictionary. So they wouldn't be sending it back to the home office.

Steve: So long as they don't reach out and check, if something's not in the dictionary, make a query to see.

Leo: I think Google sees that as a feature. That's like, you know, when you do a Google search, and you mistype it, this is actually a way I know some people check their spelling. They type it in Google search field to get the right spelling.

Steve: I do it all the time, as a matter of fact.

Leo: Yeah. Yeah. So that's kind of a feature of Google's. Turns out not to be such a good one.

Steve: Not when it's in your browser, and not when you don't know it's there.

Leo: Yeah, yeah.

Steve: Now all of our listeners know.

Leo: Good job, Steve, once again. This is why you listen to this show; right? Valuable, valuable stuff. You can get this show in a couple of ways. You can always watch us do it live, if you're in a hurry. I've got to know what happens today. Just we do this show every Tuesday about right after MacBreak Weekly, 1:30 to 2:00 p.m. Pacific. That's 4:30 Eastern time. It's 20:30 UTC. The livestream is at live.twit.tv so you can just go there, there's live audio or video, and listen along. And if you're doing that, chat with us: irc.twit.tv. Actually I'd appreciate if you did because the people in there, they're not talking about Security Now!. I think we have a lot of people who aren't - you're sometimes a little over their head. So come on in there and raise the IQ a little bit. We'd appreciate it.

Steve: Sometimes when my wife and I are out walking, some neighbors will say, so you do a podcast? Should I listen to that?

Leo: No.

Steve: No. No.

Leo: It's for a very special person. You, it's for you. The Discord, also a very good place to chat if you're already a Club TWiT member. After the fact, on-demand versions of the show are always available. Merely go to the website GRC.com. That's Steve's site. You can pick up a copy there. He has some unique formats. He has the 64Kb audio, same as we do at TWiT.tv. But he also has 16Kb audio for the bandwidth-impaired. He also has transcripts written by Elaine Farris, so that's really nice if you like to read along while you listen, or you want to search. And every show has a transcript, so you can search those transcripts and find whatever you're looking for. That's a very nice feature at GRC.com.

While you're there, support Steve. Pick up a copy of his bread and butter, SpinRite, the world's best mass storage maintenance and recovery utility. Currently 6.0; 6.1's coming. And you'll be getting it for free if you buy it today. You can also leave feedback there at GRC.com/feedback. And there's lots of other free stuff. It's well worth checking out, including ShieldsUP!, which of course is the first thing everybody should do when they get a new router is test it on ShieldsUP!.

We have copies of the video as well as the audio at our website, TWiT.tv/sn. There's a dedicated YouTube channel to Security Now!. That's a good way to share clips from people. Just go to YouTube.com/securitynow, and you can just make a little clip and share it that way. That's a great way to do that. Of course if you have a podcast player, you can subscribe. We've been around for 18 years. If it doesn't have Security Now! in its directory, I don't know what they're doing, what they're playing at. Just subscribe. That way you'll get it the minute it's available of a Tuesday evening.

If you are watching or listening after the fact, and you still want to interact, we also have TWiT Forums at TWiT.community. Those are open to all. And a Mastodon

instance, which is like Twitter only better. It's federated. And that's at TWiT.social. Again, open to all. So please join both of them. Love to have you in both places. Steve, I think we've done everything we possibly can do to save the world.