Security Now! #889 - 09-20-22
Spell-Jacking

This week on Security Now!

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 user-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.

Enough Said…

Security News

This is Patch News-Day:

Last Tuesday, Microsoft updated their range of software to resolve a total of 63 flaws including either 1 or 2 publicly disclosed 0-day vulnerabilities depending upon whether you use Microsoft's more liberal definition which does not depend upon having a vulnerability in active use. But either way, one of the vulnerabilities was being 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 finding to Microsoft.

Its CVE designation is 2022-37969, a "Windows Common Log File System Driver Elevation of Privilege." Given that it's a file system driver which runs in the kernel, any of those many attackers who were found to be having fun with it were using it to gain full SYSTEM root privileges on the machines they were attacking.

The categorical breakdown of the patches is:

- 30 Remote Code Execution Vulnerabilities
- 18 Elevation of Privilege Vulnerabilities
- 16 Edge - Chromium Vulnerabilities
- 7 Information Disclosure Vulnerabilities
- 7 Denial of Service Vulnerabilities
- 1 Security Feature Bypass Vulnerabilities

There were reports from some Admins of group policy problems after installing last Tuesday's updates, but no reports of anything widespread affecting Windows end-users.

Lloyd's of London backing away from Cyber-Insurance

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

This sort of dialing-back on cyber-insurance coverage comes as no surprise. But it's here. Insurrance firms are seeking ways to pare back cyber-insurance coverage as costs mount, driven by recent increases in crime (particularly ransomware). Nation-state attacks are most often targeted and more about espionage than theft or causing damage, but the consequences sometimes spill over to do considerable damage to other organizations.

The NotPetya incident of 2017 appears to be the primary factor driving this decision. A protracted legal battle between Merck and its fleet of insurers over $1.4 billion in damage caused by that attack finally ended in a ruling in Merck's favor last year. Recall how skeptical we were that the cost of an attack could be $1.4 billion? And we joked that Merck must be using this as an excuse to replace ALL of their PCs across the entire organization. But, really... $1.4 billion?

As we've noted before, cyber insurance coverage had previously been relying on an "acts of war" exemption to address incidents such as these, but last year's ruling establishes a legal precedent which now undermines that position. The invasion of Ukraine has stoked fears that similar cyber exchanges will slip containment and cause similarly vast damage, particularly to critical infrastructure. There has already 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, but also ended up in a large wind turbine system in Germany.

Insurers are looking to pull back on risk as companies are increasing their demand for cyber insurance coverage, and this swing in market dynamics is causing a major reorganization of the industry. Lloyd's has been planning a change of this nature for some time, drafting an assortment of new contractual clauses in late 2021 that were aimed at clarifying when cyber attacks can be considered acts of war and catastrophically damaging enough to be excepted from coverage.

The exemption terms of the new Lloyd's agreement name several specific countries: China, France, Japan, Russia, the UK and the US. It also names specific essential services that can be exempted from cyber insurance coverage if nation-state attacks cripple them: financial institutions, financial market infrastructure, health services and utilities among them. Managing agents have some leeway to include their own clauses, but must explain their approach and seek permission from Lloyd's first.

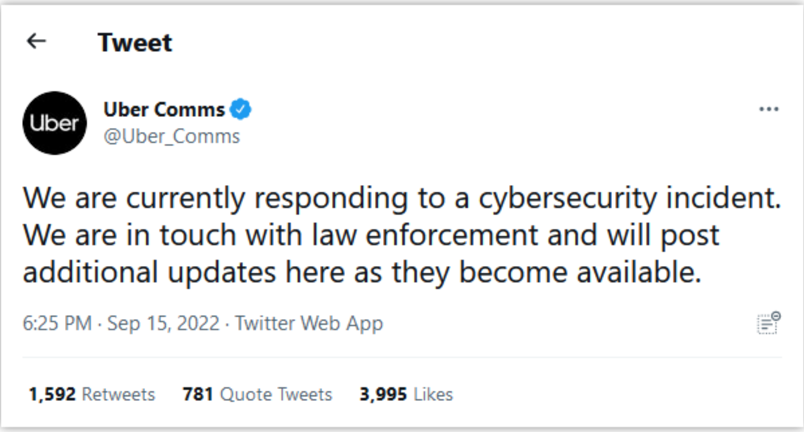
All of the coming changes to cyber insurance coverage hinge on how "nation-state attacks" are defined. This is likely the most controversial part of the arrangement — and that's where the courtroom battles will be fought. Lloyd's says that they will defer to government attribution, but that's not the only qualifier. In the absence of such attribution, the company reserves the right to make an "inference which is objectively reasonable" on its own. Uh huh. It intends to make these decisions if a government takes "an unreasonable amount of time, does not, or is unable to attribute the cyber-operation to another state or those acting on its behalf."

The "acting on its behalf" language creates particular confusion regarding certain types of non nation-state attacks. As we know, Russia's hacking groups, which the state tolerates and largely ignores unless they make too much noise, and to which the State does not provide any material support, are often behind ransomware attacks. The language seems to be included specifically to tie these sorts of independent criminal groups to nation-state attacks, making it a "get out of paying" loophole for the insurers.

With the rising cost of sufficiently comprehensive cyber-insurance, projections estimate that the number of organizations unable to afford adequate cyber insurance coverage will double in just a year. With average ransom demands now reaching into the multiple millions of dollars, and clean-up costs often in the tens of millions, many companies entirely count on insurance to cover payments when ransomware attacks hit them.

The general tightening of the cyber insurance coverage market due to increased costs began over a year ago, with AXA France becoming the first of the major insurers to cut ransomware payment reimbursements from their offerings. By late 2021 there was an established trend of insurers making changes to prices, limits and coverage; with coverage limits generally cut in half by the end of the year.

Uber Oops!
Uber suffered a significantly embarrassing network intrusion by a hacker who brazenly posted many screenshots showing the sweeping extent of his access and intrusion into Uber's inner sanctum networked systems. Last Thursday evening they Tweeted:



Uber Comms
@Uber_Comms

We are currently responding to a cybersecurity incident. We are in touch with law enforcement and will post additional updates here as they become available.

6:25 PM · Sep 15, 2022 · Twitter Web App

1,592 Retweets **781** Quote Tweets **3,995** Likes

The next day, on Friday, in the interest of keeping the lines of communication open, they posted:

While our investigation and response efforts are ongoing, here is a further update on yesterday's incident:

- *We have no evidence that the incident involved access to sensitive user data (like trip history).*
- *All of our services including Uber, Uber Eats, Uber Freight, and the Uber Driver app are operational.*
- *As we shared yesterday, we have notified law enforcement.*
- *Internal software tools that we took down as a precaution yesterday are coming back online this morning.*

And three days later, yesterday the 19th, we have a significantly more comprehensive update:

While our investigation is still ongoing, we are providing an update on our response to last week's security incident.

An Uber external contractor had their account compromised by an attacker. It is 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. 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.

In other words, this appears to have been a brute-force MFA bypass. Since the typical MFA uses 6 digits — in a clear compromise between convenience and security — there is literally a one in a million chance of correctly guessing any given MFA challenge. But if nothing stops someone from making as many guesses as they wish, 100,000 guesses would yield a 10% of guessing one correctly.

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 to a company-wide Slack channel, which many of you saw, and reconfigured Uber's OpenDNS to display a graphic image to employees on some internal sites.

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:

- *We identified any employee accounts that were compromised or potentially compromised and either blocked their access to Uber systems or required a password reset.*
- *We disabled many affected or potentially affected internal tools.*
- *We rotated keys (effectively resetting access) to many of our internal services.*
- *We locked down our codebase, preventing any new code changes.*
- *When restoring access to internal tools, we required employees to re-authenticate.* ***We are also further strengthening our multi-factor authentication (MFA) policies.***
- *We added additional monitoring of our internal environment to keep an even closer eye on any further suspicious activity.*

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 providers (e.g. 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 are currently analyzing those downloads.

The attacker was able to access our dashboard at HackerOne, where security researchers report bugs and vulnerabilities. However, any bug reports the attacker was able to access have been remediated.

Throughout, we were able to keep all of our public-facing Uber, Uber Eats, and 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.

We believe that 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 has breached Microsoft, Cisco, Samsung, Nvidia and Okta, among others. There are also reports over the weekend that this same actor breached video game maker Rockstar Games. We are in close coordination with the FBI and US 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 will also take this opportunity to continue to strengthen our policies, practices, and technology to further protect Uber against future attacks.*

I find little to fault Uber, here. 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-sleepless nights. That they are users of HacckerOne speaks well of them, and they appear to have a well-running security component to their IT systems such that their continuous auditing systems were able to provide them with a lot of relevant information when queried.

They also identified a weakness in their multi-factor authentication configuration and noted that they have responded to that by tightening it up. And a takeaway for everyone would be that just as username and password authentication should lockout for some period of time after a reasonable number of guesses, the same remains true even after multi-factor authentication has been added.

Rockstar Games: Grand Theft Auto 6 Massive Leak

That latest Uber update noted that *"There are also reports over the weekend that this same actor breached video game maker Rockstar Games."* It's certainly believable that this is the same guy or gang because they love making a large splash and the Rockstar Games breach was another such. More than 90 videos showing gameplay from the upcoming GTA6 were leaked online Sunday by an individual who claimed to be the same person who hacked Uber last week. He's been busy. Some reporting indicates that this person is believed to be a teenager but I was unable to learn what backs that up. Going by the handle "Teapot", the hacker said he plans to leak more gameplay and even some of the game's source code. Rockstar Games confirmed 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.

LastPass Breach Update:

And while we're on the subject of security breaches, last Thursday LastPass published their final official port-mortem exactly three weeks following their initial breach disclosure which left some aspects of the attack unknown. Here's the final word from their CEO:

> *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. I wanted to update you on the conclusion of our investigation to provide 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 is no evidence of any threat actor activity beyond the established timeline. We can also confirm that there is 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 multi-factor authentication.*

So it appears that we have another incidence of multi-factor authentication bypass. We don't know enough about the way LastPass has things set up. But it might just be that the bad guys obtained an authenticated session token from a successfully logged-in endpoint. It does remind us, though, that simply adding multi-factor authentication isn't any sort of universal cure.

> *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.*
>
> ***Firstly,*** *the LastPass Development environment is physically separated from, and has no direct connectivity to, our Production environment.*
>
> ***Secondly,*** *the Development environment does not contain any customer data or encrypted vaults.*
>
> ***Thirdly,*** *LastPass does not have any access to the master passwords of our customers' vaults – without the master password, it is not possible for anyone other than the owner of a vault to decrypt vault data as part of our Zero Knowledge security model.*

So 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:

> *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.* ***Developers do not have the ability to push source code from the Development environment into Production.*** *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.*
>
> *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 deployed enhanced security controls including additional endpoint security controls and monitoring. We have also deployed additional threat intelligence capabilities as well as enhanced detection and prevention technologies in both our Development and Production environments.*
>
> *We recognize that security incidents of any sort are unsettling but want to assure you that your personal data and passwords are safe in our care.*

So, I think that those who have chosen to remain with LastPass should have every reason to feel that LastPass is a competent caretaker of their users' Pre-Internet Encrypted data.

Our long-term listeners will recall that early acronym we developed on this podcast: PIE, which stood for Pre-Internet Encryption. Meaning that before sensitive data leaves the user's control it is encrypted under a key that ONLY the user controls.

A CVSS 9.8 for WordPress

Last time we talked about a big vulnerability hitting WordPress I came away suggesting that anyone who is 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 3rd-party application firewall on their site. There are three or four of those, but "WordFence" is the one that we keep referring to since they appear to be most on top of this industry which powers 40% of the Internet's websites.

And, to that end, WordFence is out with their report which serves as a perfect case in point. The report is titled: "PSA: Zero-Day Vulnerability in WPGateway Actively Exploited in the Wild"

> *On September 8, 2022, the Wordfence Threat Intelligence team became aware of an actively exploited 0-day vulnerability being used to add a malicious administrator user to sites running the WPGateway plugin. We released a firewall rule to Wordfence Premium, Wordfence Care, and Wordfence Response customers to block the exploit on the same day, September 8, 2022.*
>
> *Sites still running the free version of Wordfence will receive the same protection 30 days later, on October 8, 2022.* ***The Wordfence firewall has successfully blocked over 4.6 million attacks targeting this vulnerability against more than 280,000 sites in the past 30 days.***
>
> *The 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 functionality exposes a vulnerability that allows unauthenticated attackers to insert a malicious administrator.*
>
> *We obtained a current copy of the plugin on September 9, 2022, and determined that it is vulnerable, at which time we contacted the plugin vendor with our initial disclosure. We have reserved vulnerability identifier CVE-2022-3180 for this issue. (CVSS 9.8)*
>
> *As this is an actively exploited zero-day vulnerability, and attackers are already aware of the mechanism required to exploit it, we are releasing this public service announcement (PSA) 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.*
>
> *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**".*
>
> *If you see this user added to your dashboard, it means that your site has been compromised.*

I'm no longer running any WordPress sites. I was for a while for a blog. But if I were I would be seriously looking into adding one of those WordFence dynamic firewall services to my site.

What cost, Security?

Two months ago, on July 19th, Security Now episode #880 for 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 micro-architectures. 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 micro-architecture learning something about the code it's executing. That allows it to correctly anticipate what's most likely to happen again, which allows the micro-architecture to run ahead of the executing code to smooth the way.

But the big question we always ponder, which we seldom receive a clear answer to, is what the performance impact would be if this or that micro-architectural optimization were to be disabled in the interest of enhanced security.

Friday before last, on September 9th, a VMware engineer in their "Performance Engineering" department posted the results of his analysis of this into the Linux Kernel Archive list under the subject "Performance Regression in Linux Kernel 5.19." 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.*
>
> - *Compute (up to -70%)*
> - *Networking (up to -30%)*
> - *Storage (up to -13%)*
>
> *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 6ad0ad2bf8a6 ("x86/bugs: Report Intel retbleed vulnerability").*
>
> *To confirm this, we have disabled the above security mitigation through kernel boot parameter (spectre_v2=off) in 5.19 and re-ran our tests & confirmed that the performance was on-par with 5.18 release.*

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 of 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 is 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 immediately preceding Linux kernel 5.18, 5.19 sees a 70% reduction in the performance of compute intensive tasks, 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 the Linux aficionados among us who will want all of the details on this and on the specific benchmarks run, since VMware shared all of that in their posting:

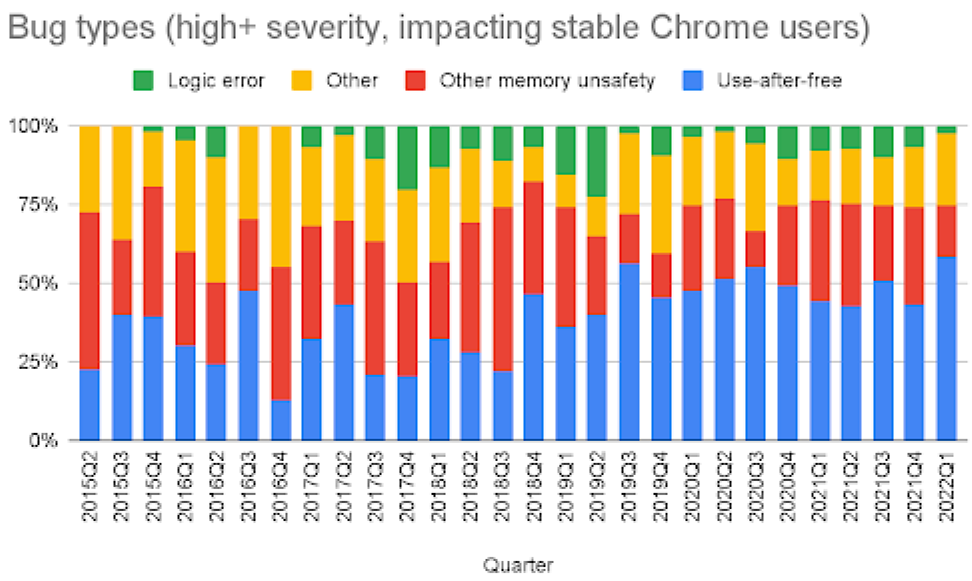
https://lkml.iu.edu/hypermail/linux/kernel/2209.1/02248.html

My feeling about all of this hasn't changed from its first appearance: End users have never had much, if anything, to worry about from any of these subtle micro-architectural attacks. It's the big datacenter cloud servers 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.

Use-after-freedom: Google's "MiraclePtr"

I wanted to tell everyone about some very encouraging news from Google's Chromium team. The trouble is that I didn't want to devote an entire podcast to tackling a single complex topic that most of our listeners won't care that much about. It involves pointer reference counting and poisoning quarantined pointers before their release. It's some seriously cool but complex pure computer science. But again, all we really need to know is that the single most troublesome aspect of the Chromium code base—those ubiquitous user-after-free errors and their exploitation — are finally going to be resolved. Here's how Google begins their explanation:

> *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, pronounced "star scan") are very powerful but likely require hardware support for sufficient performance. Today we're going to talk about a different approach to solving the same type of bugs.*
>
> *It's hard, if not impossible, 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 bit 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.*
>
> *These bugs have real consequences. For example, according to Google Threat Analysis Group, a use-after-free in the ChromeHTML engine was exploited 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:*



They then introduce the concept of what they call their "MiraclePtr":

> *MiraclePtr is 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, raw_ptr<T>. There are multiple ways to implement MiraclePtr. We came up with ~10 algorithms and compared the pros and cons. After analyzing their performance overhead, memory overhead, security protection guarantees, developer ergonomics, etc., we concluded that BackupRefPtr was the most promising solution.*
>
> *The BackupRefPtr algorithm is based on reference counting. It uses support of Chrome's own heap allocator, PartitionAlloc, which carves out a little extra space for a hidden reference count for each allocation. raw_ptr<T> increments or decrements the reference count when it's constructed, destroyed or modified. When the application calls free/delete and the reference count is greater than 0, 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 0. Quarantined memory is poisoned to further reduce the likelihood that use-after-free accesses will result in exploitable conditions, and in hope that future accesses lead to an easy-to-debug crash, turning these security issues into less-dangerous ones.*
>
> ***We successfully rewrote more than 15,000 raw pointers** in the Chrome codebase into raw_ptr<T>, then enabled BackupRefPtr for the browser process on Windows and Android (both 64 bit and 32 bit) in Chrome 102 Stable. We anticipate that MiraclePtr meaningfully reduces the browser process attack surface of Chrome by protecting ~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.*

They continue with a discussion of implementation overhead, which mostly comes down to the overhead of 4-byte reference counters for each pointer.

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

The only people who will be made unhappy by this news are these cretins who make their living off of discovering new ways to break Chrome. Their life promises to be more challenging now.

Closing The Loop

Peter G. Chase / @PchaseG

> *The picture of the week looks a lot like they're actually bypassing the meter; that looks like a meter box.*

Dan Taylor / @14aHermosa

> *Hi Steve, Years ago, you mentioned PEGASYS TMPGEnc as the video editing software you like and use. It's been at least 10 years since my first purchase. I have 3 different packages from them. The latest is: Video Mastering Works 7. And I love it! I just thought I'd say Thanks for steering me in their direction, so long ago.:-)*

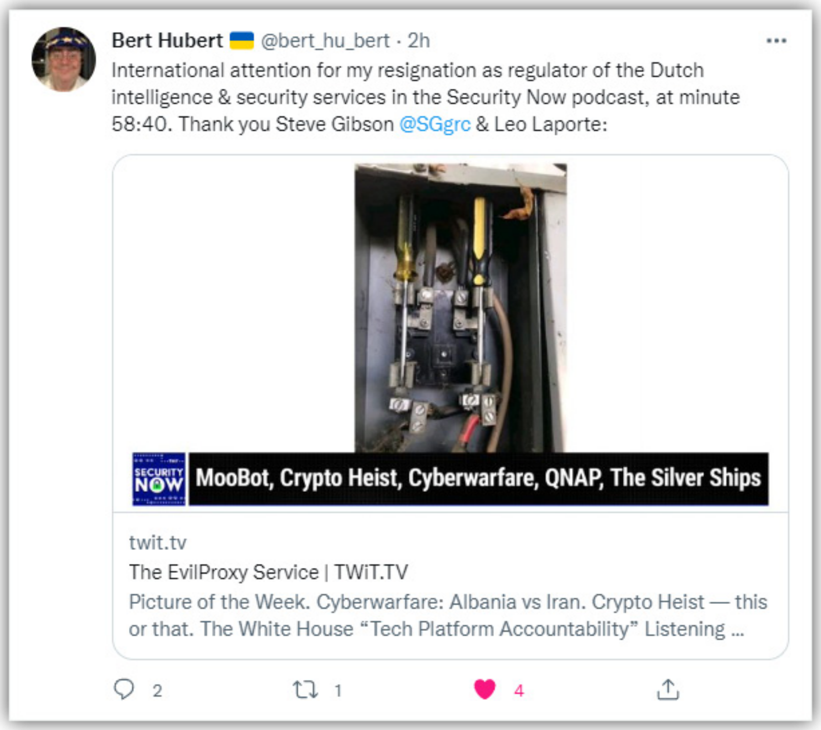
Ed Grigoleit / @EGigabyte

> *I am a long-time Security Now listener but 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 must maintain my certification by completing CPE credits. I have 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*

bobobrien / @bobobrien

> *In an alternate universe where everyone used SQRL, would that stop/thwart phishing attacks because they wouldn’t have one’s master key to encrypt the login handshake?*

A tweet from our favorite Dutch ex-regulator:



Bert Hubert 🇺🇦 @bert_hu_bert · 2h

International attention for my resignation as regulator of the Dutch intelligence & security services in the Security Now podcast, at minute 58:40. Thank you Steve Gibson @SGgrc & Leo Laporte:

twit.tv

The EvilProxy Service | TWiT.TV

Picture of the Week. Cyberwarfare: Albania vs Iran. Crypto Heist — this or that. The White House "Tech Platform Accountability" Listening ...

Sci-Fi Discovery

Bill Sempf / @sempf

Another winner, Steve. Second time now that you have recommended a book series that my wife loves.

Spell-Jacking

Otto is not a company we've talked about before. And they look like an interesting group. They describe themselves with the tag line: *"Built on innovating cyber security & protecting modern freedom"* — which seems like an interesting mix. Then they explain: *"otto was created by a team of innovators with 30 years of combined expertise in cyber security, 3rd-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 US government."*

What they appear to have is some powerful add-on technology for helping to manage what exactly a sophisticated modern website is doing — which may not always be obvious or necessarily under the control of a site's designers. Consider the challenges when all sorts of 3rd-party libraries, with complex dependency chains, are being pulled into a website visitor's browser on the fly. And then to that mess ad advertising which also bring along its own scripting. Their bullet points say:

> ***Stop Client-Side Attacks:*** *Plug otto into your application security suite and protect your supply chain. Traditional WAFs, API security, DDoS, & bot protection are all essential components of your AppSec suite, but they don't protect the client-side gap in your third-party supply chain.*
>
> ***Visibility:*** *otto provides continuous monitoring & analysis of 1st, 3rd, & Nth-party script behavior & vulnerabilities.*
>
> ***Protection:*** *Advanced Malware Guard & Script Shield defend your website from trojans, phishing, malicious code injections, magecart, and client-side attacks with real-time mitigation.*
>
> ***Control:*** *Take control over client-side application security with precision Script Policy & Dynamic CSP automation.*

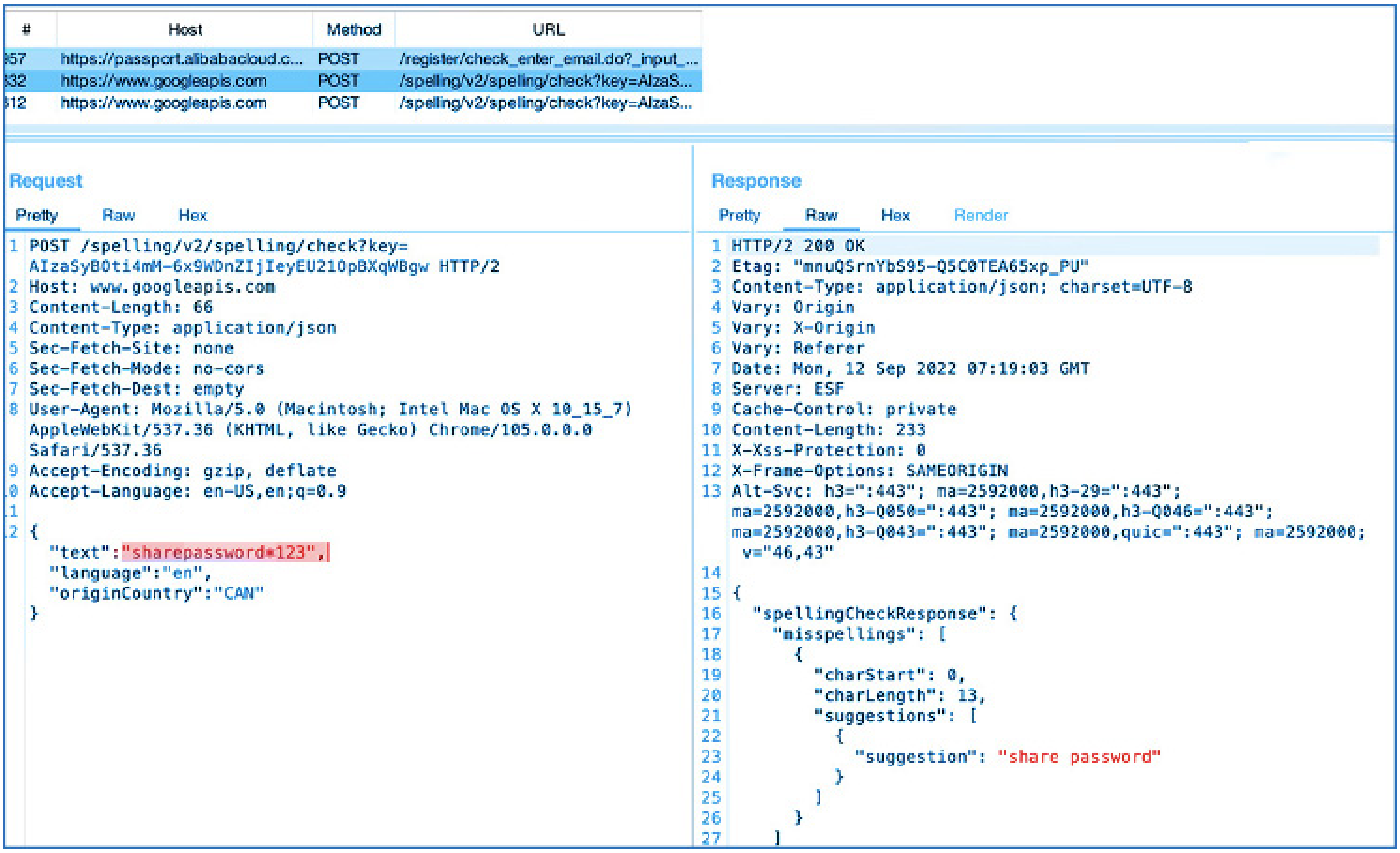
Okay. So, given that these guys are deep into watching and analyzing exactly what a user's browser is doing, it's not surprising that they would have been the ones to catch some unexpected and unwanted behavior from our Chrome/Chromium based browsers.

In summary, here's what they found:

Google's Chrome & Microsoft's Edge Editor Chromium-based 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 spell checkers is sent back, as entered in-the-clear, to Google or Microsoft to request spelling suggestions from their remote recommendation engines.

This is not the end of the world. But in an environment where we want and expect our browsers to **locally** hash 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 be preventing. And if the passwords you’re using **are** present in your spell checker’s local dictionary, so that they’re not being sent to Google or Microsoft, you have bigger problems.

I have a screen shot of Chrome’s POSTed query and Google’s reply when a user was logging into Alibaba with the test password “sharepassword*123” and Google replied with the suggestion “share password”:



#	Host	Method	URL
57	https://passport.alibabacloud.c...	POST	/register/check_enter_email.do?_input_...
32	https://www.googleapis.com	POST	/spelling/v2/spelling/check?key=AlzaS...
12	https://www.googleapis.com	POST	/spelling/v2/spelling/check?key=AlzaS...

Request

```
POST /spelling/v2/spelling/check?key=
AIzaSyBOti4mM-6x9WDnZIjIeyEU21OpBXqWBgw HTTP/2
Host: www.googleapis.com
Content-Length: 66
Content-Type: application/json
Sec-Fetch-Site: none
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: empty
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0
Safari/537.36
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{
  "text":"sharepassword*123",
  "language":"en",
  "originCountry":"CAN"
}
```

Response

```
HTTP/2 200 OK
Etag: "mnuQSrnYbS95-Q5C0TEA65xp_PU"
Content-Type: application/json; charset=UTF-8
Vary: Origin
Vary: X-Origin
Vary: Referer
Date: Mon, 12 Sep 2022 07:19:03 GMT
Server: ESF
Cache-Control: private
Content-Length: 233
X-Xss-Protection: 0
X-Frame-Options: SAMEORIGIN
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443";
ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443";
ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000;
 v="46,43"

{
  "spellingCheckResponse": {
    "misspellings": [
      {
        "charStart": 0,
        "charLength": 13,
        "suggestions": [
          {
            "suggestion": "share password"
          }
        ]
      }
    ]
```

Again, not the end of the world, but not a good look. Also, never useful to run someone’s password, eMail, or social security number through spell check.

otto-js co-founder & CTO Josh Summitt discovered the spellcheck leak while he was testing the company's script behaviors detection. He explained:

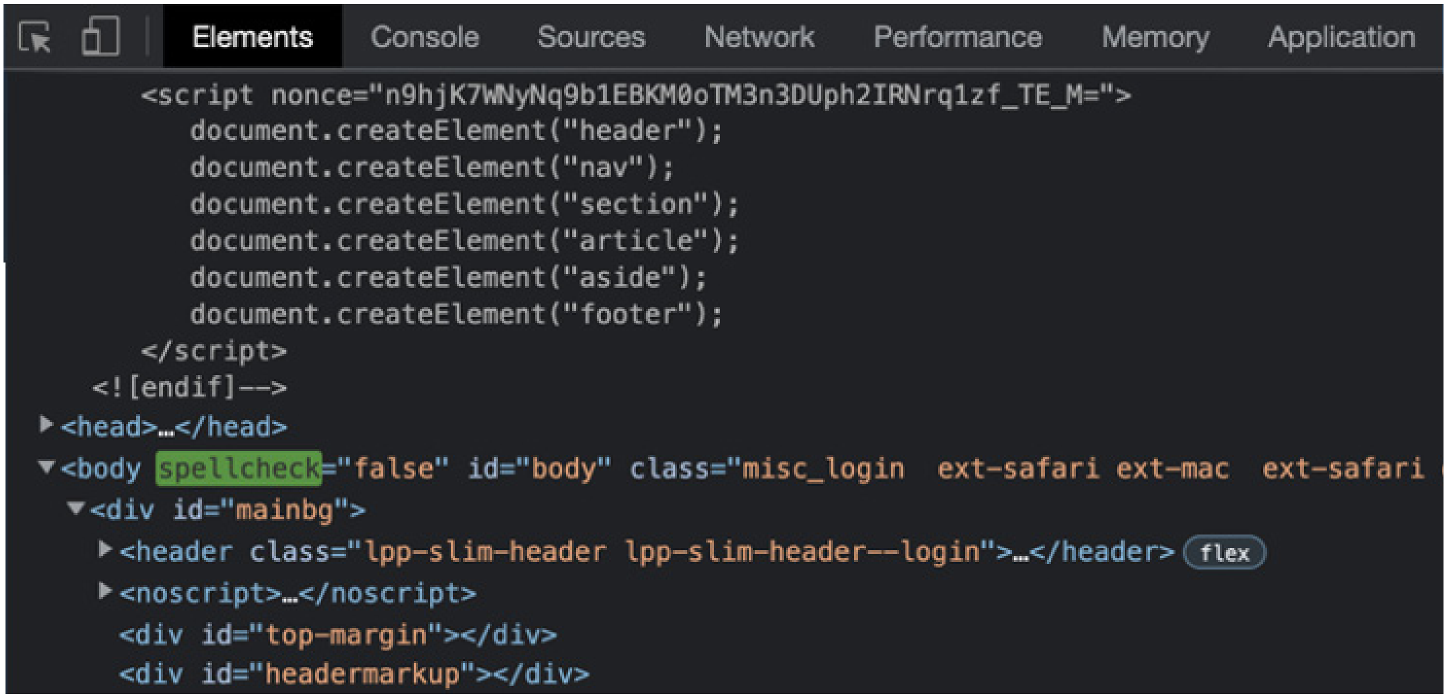
“If 'show password' is enabled, the feature sends your password to their 3rd-party servers. While researching for data leaks in different browsers, we found a combination of features that, once enabled, will unnecessarily expose sensitive data to 3rd parties like Google and Microsoft. What's concerning is how easy these features are to enable and that most users will enable these features without realizing what’s happening in the background.”

This unsuspected and inadvertent leakage could 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 HIPPA and similar privacy regulations which rigorously restrict 3rd-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 spell checker, 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 current logged-in session ID cookie will accompany the aberrant spell check looking query.

Otto-js noted that 5 websites and services of concern were

Office 365
Alibaba - Cloud Service
Google Cloud - Secret Manager
AWS - Secrets Manager (UPDATE: has already fully mitigated the issue)
LastPass (UPDATE: has already fully mitigated the issue)

But 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' Christofer Hoff, 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."*



```
      <script nonce="n9hjK7WNyNq9b1EBKM0oTM3n3DUph2IRNrq1zf_TE_M=">
        document.createElement("header");
        document.createElement("nav");
        document.createElement("section");
        document.createElement("article");
        document.createElement("aside");
        document.createElement("footer");
      </script>
    <![endif]-->
 ▶<head>…</head>
 ▼<body spellcheck="false" id="body" class="misc_login  ext-safari ext-mac  ext-safari
   ▼<div id="mainbg">
     ▶<header class="lpp-slim-header lpp-slim-header--login">…</header> flex
     ▶<noscript>…</noscript>
      <div id="top-margin"></div>
      <div id="headermarkup"></div>
```

In the show notes I've included a screenshot of LastPass's updated sign-in page definition showing where the term **spellcheck="false"** was simply added to the page's <body> tag, thus fully disabling spellcheck on that page's login form. It's as easy as that.

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 as he creates a document. But that feature is now enabled and remains enabled for all sites the user visits until he returns to settings and manually disables enhanced spellcheck.

https://youtu.be/Onb0Usgs04I

In the video, a company's enterprise database credentials are being spell-jacked when the employee switches over to the company's cloud services account and clicks "show password," which then shows that the password is being sent to Google.

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 the 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 6 categories of websites which people use frequently and which have access to highly sensitive PII data. Remember that by default, ALL data entered into forms that's not recognized by the browser's local spell check dictionary will be sent to retrieve remote suggestions. But passwords will not be sent until and unless the user clicks the "show password" button.

In Otto's testing, 5 websites per category were selected based on top ranking in each industry and tested to create a benchmark of how much exposure might be happening. The 6 categories were:

- Online Banking
- Cloud Office Tools
- Healthcare
- Government
- Social Media
- eCommerce

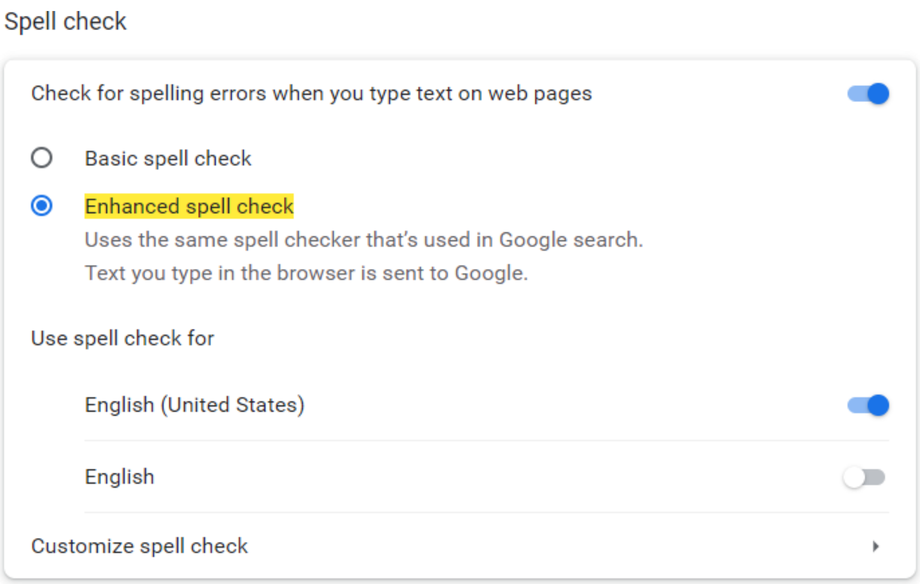
Of the reference group of 30 websites tested, 96.7 percent did send PII data back to Google & Microsoft. Only one did not.

73 percent sent passwords when "show password" was clicked. But those not sending passwords only didn't because they lacked the "show password" feature.

And, interestingly, the only control group website tested that **had** mitigated the issue was Google. So Google was aware of this and didn't want that PII 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 services like Google Cloud' Secret Manager.' Auth0, a popular single-sign-on service, was not in the control group but was the only website other than Google that **had** correctly mitigated the issue. So 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' PII by adding "spellcheck=false" to the containing page, or to all input fields, though this might create problems for users. Alternatively, that override could just be added to form fields which might contain sensitive data.

Fortunately the enhanced spellcheck features are not the default setting in Chrome, but once they've been enabled they will remain so until disabled. I was curious about my own settings. So I went to Settings and entered "enhanced spell" into the search bar. And what do you know:



Spell check

Check for spelling errors when you type text on web pages ☑

○ Basic spell check

● Enhanced spell check
Uses the same spell checker that's used in Google search.
Text you type in the browser is sent to Google.

Use spell check for

English (United States) ☑

English □

Customize spell check ▸

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: *"Text you type in the browser is sent to Google."*

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. So, for users who rely on enhanced spellcheck — I'm pretty sure that I do — this extension would allow us to leave enhanced spellcheck enabled while also reminding us where we're on a page where phoning home to the Google mothership might not make us comfortable. Otto-js actually has two add-on extensions and that say that either add-on provides alerts and quick tips to disable. I have the links here at the end of the show notes for anyone who's interested in pursuing this approach:

otto-js ShopSecure - free browser protection for shoppers
https://chrome.google.com/webstore/detail/otto-shopsecure/agdacffjkhhfiebgpebahknkihmefjok

otto-js Developer Tools - free runtime script testing tools
https://chrome.google.com/webstore/detail/otto-javascript-security/lcmaikahgebmdmnckjbaikfllpmgabei

So, as I said at the outset, not the end of the world, but something to keep in mind.