

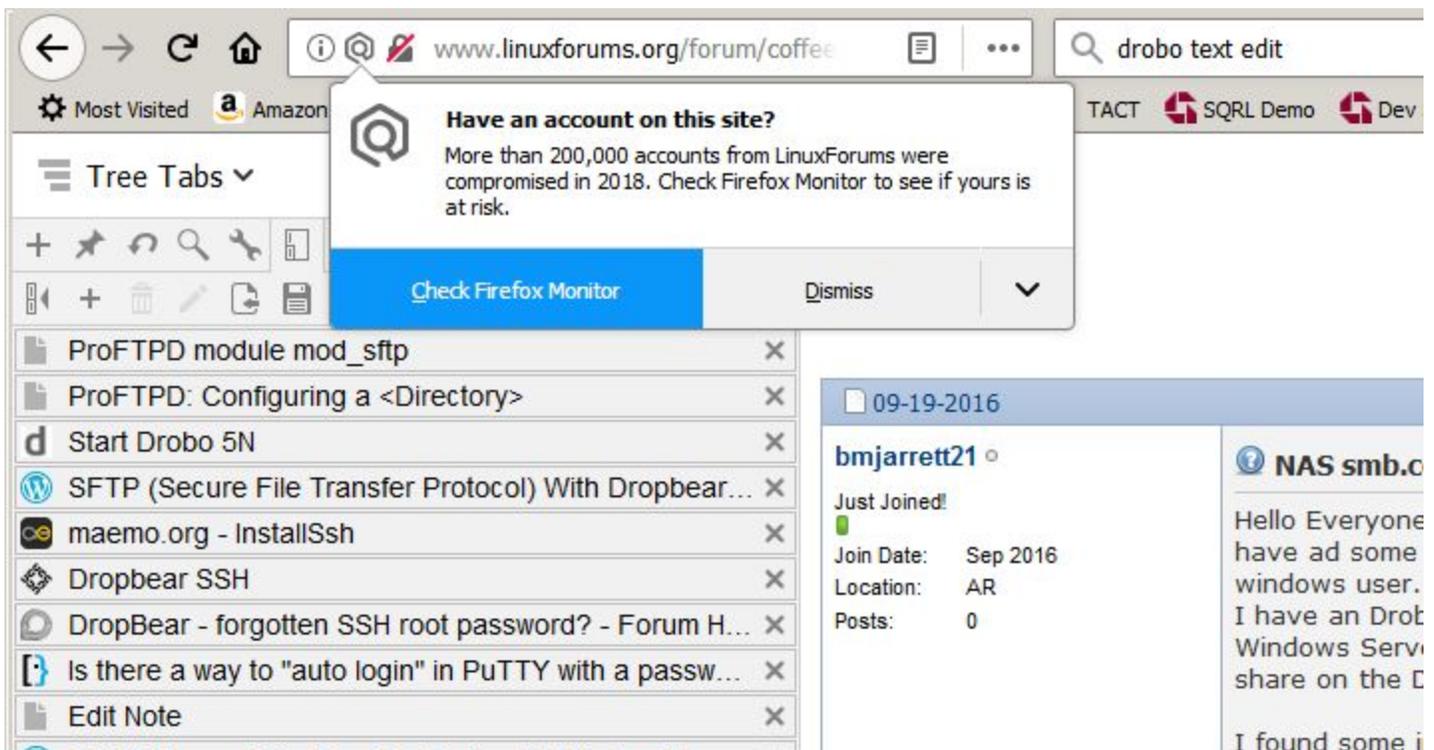
# Security Now! #700 - 02-05-19

## 700 and counting!

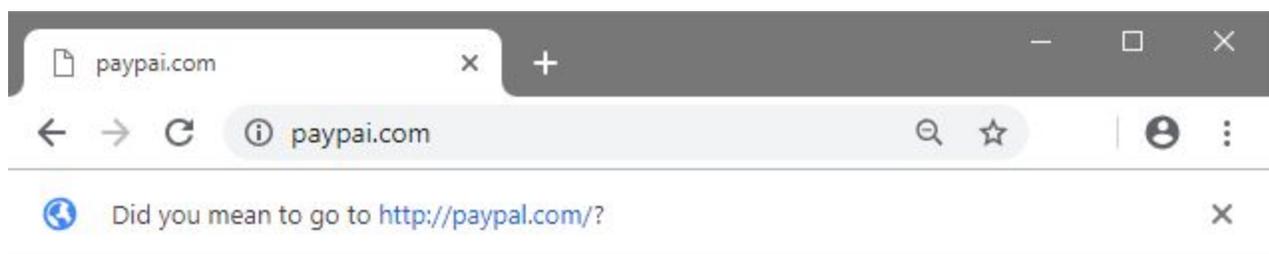
### This week on Security Now!

This week we discuss Chrome getting spell-check for URLs, a bunch of Linux news with reasons to be sure you're patched up and some performance enhancements, updates, additions and deletions from Chrome and FireFox, more Facebook nonsense, a bold move planned by the Japanese government, Ubiquity routers again in trouble, a hopeful and welcome new initiative for the Chrome browser, a piece of errata, a quick SQL update and some follow-up thoughts about VPN connectivity.

### FireFox Warned Me!



### And So Did Chrome!



## Security News

### Chrome to get URL spell-checking

Less success for "typo-squatters"

Okay, so first some terminology:

Typo-Squatting is formally known as "IDN (or International Domain Name) Homograph Attack". [https://en.wikipedia.org/wiki/IDN\\_homograph\\_attack](https://en.wikipedia.org/wiki/IDN_homograph_attack) ... but typo-squatting is a lot catchier!

Similarly, although I think that "URL spell checking" is rather clear, Google calls their forthcoming technology "Navigation suggestions for lookalike URLs." It's now under active experimentation in the Canary 70 version of Chrome and, if all goes as planned, will be appearing in the mainstream release before long.

If you're on the Canary circuit, you can enable the feature here:  
`chrome://flags/#enable-lookalike-url-navigation-suggestions`

The flag itself appears to be present in the just-released stable channel Chrome 72... But it doesn't appear to be wired up to this new functionality yet.

---

### Linux Systemd vulnerabilities: definitely time to catch up your patch up!

We talked previously without much specificity about the recognized trouble with the systemd daemon which a number of the more popular Linux desktop-oriented distros have adopted.

Now, Capsule8, a US cybersecurity company has published a working proof of concept which weaponizes two of the three vulnerabilities which were reported by Qualys late last year.

<https://lwn.net/Articles/776404/>

On Wednesday, 9 Jan 2019, Qualys posted the following detailed report:

"System Down: A systemd-journald exploit"

We discovered three vulnerabilities in systemd-journald (<https://en.wikipedia.org/wiki/Systemd>):

- CVE-2018-16864 & CVE-2018-16865, two memory corruptions (attacker-controlled `alloca()`s);
- CVE-2018-16866, an information leak (an out-of-bounds read).

CVE-2018-16864 was introduced in April 2013 (systemd v203) and became exploitable in February 2016 (systemd v230). We developed a proof of concept for CVE-2018-16864 that gains EIP control on i386.

CVE-2018-16865 was introduced in December 2011 (systemd v38) and became exploitable in April 2013 (systemd v201). CVE-2018-16866 was introduced in June 2015 (systemd v221) and was inadvertently fixed in August 2018.

We developed an exploit for CVE-2018-16865 and CVE-2018-16866 that obtains a local root shell in 10 minutes on i386 and 70 minutes on amd64, on average. We will publish our exploit in the near future.

To the best of our knowledge, all systemd-based Linux distributions are vulnerable, but SUSE Linux Enterprise 15, openSUSE Leap 15.0, and Fedora 28 and 29 are not exploitable because their user space is compiled with GCC's `-fstack-clash-protection`.

"systemd" is used by Arch Linux, CentOS, CoreOS, Debian, Fedora, Mageia, Mint, Red Hat Enterprise, Linux, Solus and Ubuntu.

*The whole story:* CVE-2018-16865 is a vulnerability triggered by code in systemd's logging software (journald) that allocates temporary memory to contain a log entry without first checking that the request is of a sensible size. This allows for ready code execution, but it is thwarted by ASLR.

Fortunately, or not, the second bug CVE-2018-16866 allows specially-formatted text sent to the system log to cause systemd to write out a message containing data from parts of memory that the user is not supposed to see. This is a classic information disclosure vulnerability which then provides the attacker with the information required to exploit the previous vulnerability to run code to accomplish a malignant goal.

This beautifully demonstrates the way two vulnerabilities, neither by themselves very potent, can each provide what the other is lacking to enable a significant breach of system security. It's not clear how this might be triggered remotely, since the attacker needs to see the output from the first to gain a foothold with the second... But code running on a machine could use this to breach the system privilege controls.

So... If you're using Linux, make sure you have updated your systems recently. And if you've been putting it off waiting for a good reason... That reason has arrived. :-/

---

And Speaking of Chrome 72...

We get a bunch of bug fixes and updates, but also some significant deprecations and removals:

### **Don't allow popups during page unload**

Pages may no longer use `window.open()` to open a new page during unload -- thank God! The Chrome popup blocker already prohibited this, but now it is prohibited whether or not the popup blocker is enabled. Whew!

### **Remove HTTP-Based Public Key Pinning**

HTTP-Based Public Key Pinning (HPKP) was intended to allow websites to send an HTTP header that pins one or more of the public keys present in the site's certificate chain. But the crucial importance of never mis-issuing a HPKP header, nor failing to anticipate an upcoming certificate change, led to low adoption since the risks weren't worth the reward. So support for the little-used feature is being dropped from Chrome.

## Remove rendering FTP resources.

As we know, FTP is a non-secured legacy protocol. When even the Linux kernel is moving away from it, it's time to move on. One step toward FTP's gradual deprecation and removal is to deprecate rendering resources from FTP servers and instead download them. Chrome will still generate directory listings, but any non-directory listing will be downloaded rather than rendered in the browser. I can't imagine who might be SHOWING an image over FTP... but, hey, if it's possible, someone's doing it!

*And as for deprecations...*

## Deprecate TLS 1.0 and TLS 1.1

We've been following the journey from SSL to TLS with TLS v1.0 now nearly 20 years old. Since TLS 1.0 and 1.1 have a number of non-critical but still annoying security weaknesses. Specifically...

- TLS 1.0 and 1.1 use MD5 and SHA-1, both weak hashes, in the transcript hash for the Finished message.
- TLS 1.0 and 1.1 use MD5 and SHA-1 in the server signature. (Note: this is not the signature in the certificate.)
- TLS 1.0 and 1.1 only support RC4 and CBC ciphers. RC4 is broken and has since been removed. TLS's CBC mode construction is flawed and was vulnerable to attacks.
- And TLS 1.0's CBC ciphers construct their initialization vectors incorrectly.
- TLS 1.0 is no longer PCI-DSS compliant.

So... Since TLS v1.2 resolves those problems, and it's solid and well supported, it's time to put the nearly twenty-year-old TLS 1.0 and 1.1 out to pasture.

Supporting TLS 1.2 is a prerequisite to avoiding the above problems. The TLS working group has deprecated TLS 1.0 and 1.1... and with Chrome 72 they are now deprecated there, too. Their full removal is scheduled for Chrome 81 in early 2020.

Google/Chrome has this to say about their deprecation policy:

### Deprecation policy

To keep the platform healthy, we sometimes remove APIs from the Web Platform which have run their course. There can be many reasons why we would remove an API, such as:

- They are superseded by newer APIs.
- They are updated to reflect changes to specifications to bring alignment and consistency with other browsers.
- They are early experiments that never came to fruition in other browsers and thus can increase the burden of support for web developers.

Some of these changes will have an effect on a very small number of sites. To mitigate issues

ahead of time, we try to give developers advanced notice so they can make the required changes to keep their sites running.

Chrome currently has a [process for deprecations and removals of API's](#), essentially:

- Announce on the [blink-dev](#) mailing list.
- Set warnings and give time scales in the Chrome DevTools Console when usage is detected on the page.
- Wait, monitor, and then remove the feature as usage drops.

You can find a list of all deprecated features on [chromestatus.com](#) using the [deprecated filter](#) and removed features by applying the [removed filter](#). We will also try to summarize some of the changes, reasoning, and migration paths in these posts.

---

## Facebook must really like being in the doghouse

<https://techcrunch.com/2019/01/29/facebook-project-atlas/>

I'm not of the camp that believes this is sinister. I just think it's ungoverned and irresponsible. I suspect it's what happens under conditions of explosive growth and lack of adult supervision when you tell a hoard of recently degreed 20-something coders to "just do stuff... and we'll keep what works".

In today's installment of "What has Facebook wrought now?" we have TechCrunch's report, headlined with: "Facebook pays teens to install VPN that spies on them."

Josh Constine writes: Desperate for data on its competitors, Facebook has been secretly paying people to install a "Facebook Research" VPN that lets the company suck in all of a user's phone and web activity, similar to Facebook's Onavo Protect app that Apple banned in June and that was removed in August. Facebook sidesteps the App Store and rewards teenagers and adults to download the Research app and give it root access to network traffic in what may be a violation of Apple policy so the social network can decrypt and analyze their phone activity, a TechCrunch investigation confirms.

Facebook admitted to TechCrunch it was running the Research program to gather data on usage habits.

Since 2016, Facebook has been paying users ages 13 to 35 up to \$20 per month plus referral fees to sell their privacy by installing the iOS or Android "Facebook Research" app. Facebook even asked users to screenshot their Amazon order history page. The program is administered through beta testing services Applause, BetaBound and uTest to cloak Facebook's involvement, and is referred to in some documentation as "Project Atlas" — a fitting name for Facebook's effort to map new trends and rivals around the globe.

Seven hours after this story was published, Facebook told TechCrunch it would shut down the iOS version of its Research app in the wake of our report. But on Wednesday morning, an Apple spokesperson confirmed that Facebook violated its policies, and it had blocked Facebook's

Research app on Tuesday before the social network seemingly pulled it voluntarily (without mentioning it was forced to do so).

An Apple spokesperson provided this statement. "We designed our Enterprise Developer Program solely for the internal distribution of apps within an organization. Facebook has been using their membership to distribute a data-collecting app to consumers, which is a clear breach of their agreement with Apple. Any developer using their enterprise certificates to distribute apps to consumers will have their certificates revoked, which is what we did in this case to protect our users and their data."

In other words... Facebook was deliberately bypassing the Apple iTunes store to offer an unsanctioned app to their users. And... Facebook's Research app requires users to 'Trust' it with extensive access to their data -- asking their users to install a custom root certificate into their devices.

Facebook's Research program will continue to run on Android.

TechCrunch asked Guardian Mobile Firewall's security expert Will Strafach to dig into the Facebook Research app, and he told us that "If Facebook makes full use of the level of access they are given by asking users to install the Certificate, they will have the ability to continuously collect the following types of data: private messages in social media apps, chats from in instant messaging apps – including photos/videos sent to others, emails, web searches, web browsing activity, and even ongoing location information by tapping into the feeds of any location tracking apps you may have installed." It's unclear exactly what data Facebook is concerned with, but it gets nearly limitless access to a user's device once they install the app.

The TechCrunch report quotes Will Strafach, saying: "The fairly technical sounding 'install our Root Certificate' step is appalling. This hands Facebook continuous access to the most sensitive data about you, and most users are going to be unable to reasonably consent to this regardless of any agreement they sign, because there is no good way to articulate just how much power is handed to Facebook when you do this."

TechCrunch's report goes on at length for anyone who's curious. I have a link in the show notes. And, again, this just seems dumb to me rather than diabolical.

---

## **The Japanese government**

In a few weeks the Japanese government is going to try its hand at the practice that's come to be known as "credential stuffing" against its own the country's massive installed base of IoT devices, including anything it can find from the enterprise network level down to end-user routers whose owner never changed their default passwords. NHK, Japan's national public broadcasting organization reported that the government had approved this first-of-its-kind venture several weeks ago.

So, in mid-February, staffers from the National Institute of Information and Communications Technology (NICT) will take many previously successful username and passwords and use them to break into as many as 200 million randomly selected IoT devices located throughout Japan -- routers, webcams, DVRs, electric toothbrushes... you-name-it.

Then, the owners of the breached devices will be told to bolster their cybersecurity.

The intent behind this interesting white-hat move is to reduce the viable attack surface that's available to attackers before the approaching Tokyo Olympics and Paralympics in 2020. Sophos noted that some systems did go down around the time of the opening ceremony for the Winter Olympics in Pyeongchang, South Korea, last year.

Although the immediate goal is to tighten up Japanese citizens' Internet-facing security before the Olympics, the result will almost certainly be significantly improved security overall. The NICT has reported that IoT devices were at the heart of a large number – 54% – of the cyber attacks it detected in 2017.

I've several times mentioned that way back in the early days of the Code Red and Nimda worms, I participated with some discussions with the US Federal government. I was on a conference call with the head of the DoJ and we security types were BEGGING to be allowed to write a sanitizing worm that would find the vulnerable devices the fix them.

We were told in no uncertain terms that doing so would be illegal and would open us to the full wrath of the Federal government.

Today, though... Don't things seem a bit less black and white? We have leaks from the CIA and the NSA which suggest that no one here is a Boy Scout. And tying the hands of the white hats while the black hats run free and cripple our cyber infrastructure is seeming less and less correct in practice.

So... If this Japanese experiment bears fruit -- and you can bet that all governments are watching closely -- it might go a long way toward opening some doors and changing some minds about the feasibility of being a bit proactive.

And it COULD be done incrementally. Individual ISPs could be chartered with the ability to inspect the publicly available ports of their customers and to deal with them one-on-one to resolve any detected vulnerabilities.

There would be a market for a big hardware security maker to produce a carrier-grade network vulnerability scanner specifically for use by ISPs.

---

### **Firefox 65:**

I have Firefox v65 and I'm not have any problems because I'm not using Avast, AVG, BitDefender, ESET or Kaspersky.

As we have discussed before, in order to hold onto their diminishing relevance by remaining able to peer into, and scan, the content of HTTPS connections, all of those AntiVirus addons are performing user-sanctioned man-in-the-middle interception of all HTTPS traffic.

As we know, for the signatures of a website's certificates to be trusted, the website's private key must be pre-installed into the local system's root certificate store. In order to transparently intercept HTTPS traffic, a user's A/V addon generates a unique public and private key pair during

installation and places the public key into the user's local certificate store. Then it pretends to be any remote server the user connects to, generating a certificate on-the-fly, signed by its built-in private key so that the web browser will trust it.

This all went awry when Firefox 65, as part of its enhanced security warnings began alerting people who were using these A/V products that something was wrong with their connections and that the sites they were visiting should not be trusted.

Much Chaos ensued. Mozilla quickly halted the roll out of FF65. Avast and AVG were quick to issue patches to disable HTTPS scanning of Firefox.

However, amid all of this, I stumbled upon the coolest workaround hack that I never knew existed: It's possible to instruct Firefox to use the Windows certificate root store.

Firefox has always carried its own security library and private root store. It has never used Windows. But a Firefox config setting will instruct FF to rely upon Windows' certificate store:

The config setting is: "security.enterprise\_roots.enabled"

Enter about:config into the Firefox URL field.

Whittle down the bazillion config options by searching for "enterprise" (like the starship).

Double-click the item to toggle it TRUE.

Now Firefox will see the A/V root certs, too, and won't complain.

Note, also, that FF plans to do something more with the next major release, FF66, where some sort of MITM warning will be presented to warn of non-authentic certificates.

---

### **Ubiquity routers are again in trouble:**

Jim Troutman, the co-founder of internet exchange point NNENIX (Northern New England Neutral Internet Exchange) tweeted last Tuesday early morning:

Heads up! Ubiquiti networks devices are being remotely exploited, via port 10001 discovery service. Results in loss of device management, also being used as a weak UDP DDoS amplification attack: 56 bytes in, 206 bytes out.

Attackers are sending small packets of 56 bytes to port 10,001 on Ubiquiti devices, which are reflecting and relaying the packets to a target's IP address amplified to a size of 206 bytes (amplification factor of 3.67).

So we have a classic UDP traffic reflection attack: Since this port 10001 discovery service is intended to be lightweight it uses UDP protocol. And, as we know, since UDP protocol is lightweight specifically because it does zero connection setup handshaking, the source IP of any incoming UDP packets can simply be spoofed. As a result, any well-meaning reply packet will be returned to the spoofed source, thus potentially flooding an unwitting Internet device that never asked for any of this in the first place.

Then the guys at Rapid7 Labs got involved. Some quick sleuthing by the security community showed that this trouble had been brewing since last summer and Ubiquiti recently acknowledged that they are aware of the issue and are working on a fix.

It turns out that the service is used for a variety of things, including device discovery to facilitate easily locating of Ubiquiti devices in a managed environment. The protocol is quite simple, requiring a simple 4-byte message that elicits a large response including the name, model, firmware version, IPs, MACs, and sometimes the ESSID if it is a wireless device.

Interestingly, the Rapid7 guys were able to throw 4 bytes at the router to elicit a reply, so they were stating an amplification factor is 30-35x, but they might not have been factoring in the per-packet UDP overhead which definitely does matter for any bandwidth flooding attack. However, their scan of the Internet revealed 498,624 unique IPv4s with port 10001/UDP open. Of those, 487,021 unique IPv4s were confirmed to be answering to this discovery protocol, and 486,388 unique physical devices based on MAC address tuples were found in the responses. So... There are just shy of half a million confirmed devices currently wide open and ready to reflect and amplify traffic for a bandwidth flooding attack. As we've seen before with Ubiquiti devices, more than half are located in Brazil.

One mitigating factor is that any incoming flood traffic would be coming from port 10001, which is not under the attacker's control. So blocking that UDP port at the network perimeter would provide some potential for mitigation. Though it's still a potentially large incoming attack to block.

Rapid7 decoded the responses from the devices and produced a breakdown by type and number. <https://blog.rapid7.com/2019/02/01/ubiquiti-discovery-service-exposures/>

The top six were the:

- NanoStation @ 172,563
- AirGrid @ 131,575
- LiteBeam @ 43,673
- PowerBeam @ 40,092
- NanoBeam @ 21,360
- NanoBridge @ 20,440

However, when they examined the device names being returned in many of their queries they noticed something troubling:

- HACKED-ROUTER-HELP-SOS-HAD-DUPE-PASSWORD 9,146
- HACKED-ROUTER-HELP-SOS-WAS-MFWORM-INFECTED 3,891
- HACKED-ROUTER-HELP-SOS-DEFAULT-PASSWORD 1,628
- HACKED-ROUTER-HELP-SOS-VULN-EDB-39701 1,168
- HACKED-ROUTER-HELP-SOS-HAD-DEFAULT-PASSWORD 1,096

This led Rapid7 to conclude that the attackers had also identified additional vulnerabilities within these routers and had exploited over 17,000 of them... As evidenced by the defaced hostnames.

And, finally, the Rapid7 folks examined the firmware versions being reported and saw that along with being exposed to attackers, most of the devices in each product family are running outdated firmware. So we have the perfect storm of an exposed service, unpatched vulnerabilities, and default credentials (since some of the hacker-changed device names noted the presence of default credentials).

Ubiquity does have a command to turn this off, though it's enabled by default:

<https://community.ubnt.com/t5/EdgeRouter/UDP-broadcasts-on-port-10001/td-p/461223>

```
ubnt@ubnt:~$ configure
ubnt@ubnt# set service ubnt-discover disable
ubnt@ubnt# commit
ubnt@ubnt#
```

I still think that the little Ubiquity EdgeRouter X's are truly amazing little routers... but it does look as though they are better used by someone who knows what they are doing and who is also keeping those little boxes of technology up to date with the latest firmware.

---

### Google Working on Chrome "Never-Slow Mode" for Faster Browsing

Bleeping Computer put me onto an interesting experiment that one of the Chrome developers is working on. He calls it "Never-Slow Mode" and the idea is that if, while a web page is loading, something is taking too long, or something is too large, it will be abandoned and ignored in favor of getting the page loaded.

<https://chromium-review.googlesource.com/c/chromium/src/+1265506>

"Never-Slow Mode"

PROTOTYPE -- DO NOT COMMIT

Adds `--enable-features=NeverSlowMode` to enforce per-interaction budgets designed to keep the main thread clean (design doc currently internal).

Currently blocks large scripts, sets budgets for certain resource types (script, font, css, images), turns off document.write(), clobbers sync XHR, enables client-hints pervasively, and buffers resources without `Content-Length` set. Budgets are re-set on interaction (click/tap/scroll). Long script tasks (> 200ms) pause all page execution until next interaction.

Caps do not apply to workers and size caps are lifted for resources loaded from Service Worker Cache Storage.

Current caps; all values are wire (transfer/compressed) size:

Per-image max size:	1MiB
Total image budget:	2MiB
Per-style sheet max size:	100KiB
Total style sheet budget:	200KiB

Per-script max size:	50KiB
Total script budget:	500KiB
Per-font max size:	100KiB
Total font budget:	100KiB
Total connection limit:	10
Long-task limit:	200ms

TODO: <iframe> depth is not yet limited and font-loading is not yet greedy. Feature-policy header to trigger on per-page basis is not implemented. No UI is implemented to inform users that a page is slow.

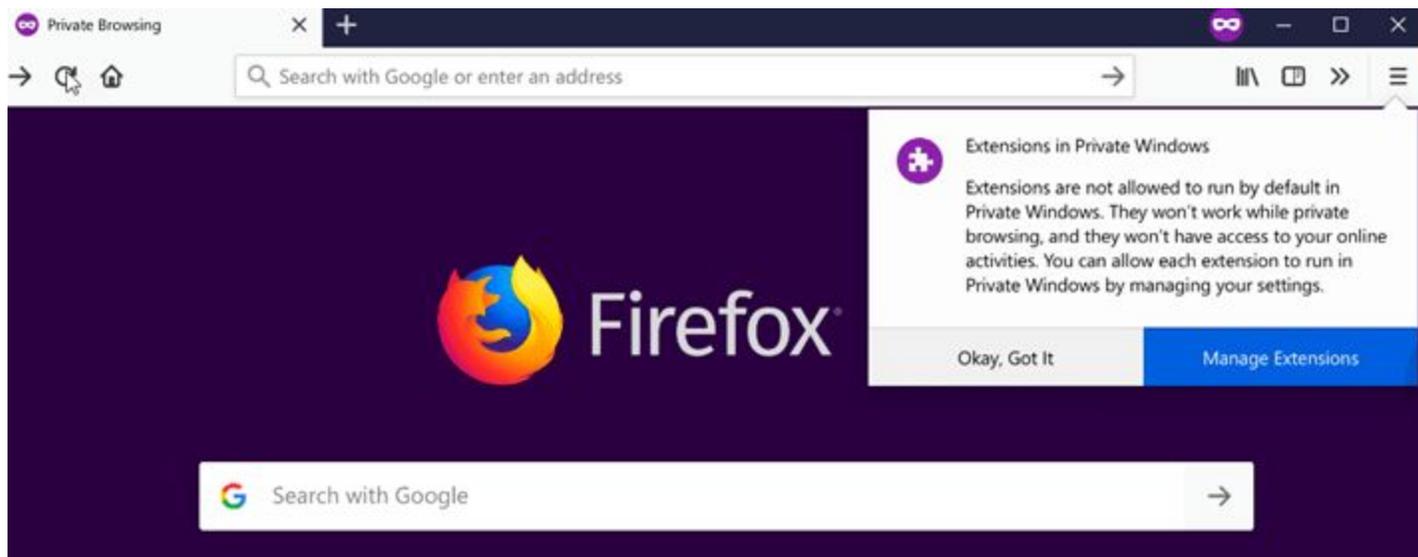
While this seems fraught with peril... I really do like the underlying incentive behind this. Today there is essentially zero push-back against sloppy coding, lazy image sizing, and massive code libraries being downloaded just so that a single function can be accessed. People don't care because they're busy and no one is making them care. Let the page's consumer pay in bandwidth and time. So the idea that the #1 web browser in the world might start actively pushing back against needlessly slow page loads, is right up my alley.

*Give me the option and I'll turn it on!*

---

## Firefox is finally catching up with Chrome, Opera, and Microsoft Edge

Web browser extensions often track users. They employ tracking scripts like Google Analytics or redirects through servers that are used to track a user's browsing and search behavior. Since this is not behavior that users typically want while they are Incognito, Chrome, Opera and Edge have stopped running browser extensions by default when the user switches into their "please don't remember anything I do while I'm in here" mode.



The good news is that Firefox will similarly begin blocking extensions. Users will be notified that extensions are not running and will have the ability to selectively reenable those (perhaps such as Lastpass or SQRL) that they trust and wish to be able to continue using. This new feature has landed in the Firefox nightly builds and will be moving to the production builds once any bugs have been worked out.

## Linux kernel gets another option to disable Spectre mitigations

People want more control over the Spectre mitigations for the sake of performance.

<https://www.zdnet.com/article/linux-kernel-gets-another-option-to-disable-spectre-mitigations/>

ZDNet writes: Believe it or not, the mitigations for the Spectre-class of CPU vulnerabilities are now some of the biggest enemies of system administrators.

So, Leo, this update article follows some of what we've been saying about these threats, and it confirms the feeling I've always had, which was that Intel was (undrestandably) significantly minimizing the performance hit that disabling these powerful performance enhancements would cause.

ZDNet says (and I'll be paraphrasing a bit for brevity)... Despite being security-focused patches, these mitigations are known to introduce huge performance hits to Linux systems.

A recent benchmark showed that just one of the many Spectre mitigations --namely the one named Single Thread Indirect Branch Predictors (STIBP)-- introduced a 30 percent performance dip for PHP servers, causing system administrators to reconsider applying some of these patches.

Despite being more than an old, the Meltdown or Spectre vulnerabilities have remained a theoretical threat, and no malware strain or threat actor has ever used any in a real-world attack. Consequently, during the past year, system and network administrators have called on the Linux project for options to disable these protections.

Many argued that the threat is theoretical and could easily be mitigated with proper perimeter defenses, in some scenarios. Even Linus Torvalds has called for a slowdown in the deployment of some performance-hitting Spectre mitigations.

The Linux kernel team has reacted favorably towards these requests and has been slowly adding controls to disable some of the more problematic mitigations.

For example, since Linux Kernel 4.15, administrators can disable the kernel's built-in mitigations for the Spectre v2 vulnerability (CVE-2017-5715) with the "nospectre\_v2" kernel command line parameter.

Since Linux Kernel 4.17, administrators have been empowered to disable all mitigations for Spectre v4 (CVE-2018-3639) with the "nospec\_store\_bypass\_disable" command line parameter.

Similarly, a way to disable mitigations for Spectre v1 (CVE-2017-5753) has been added in the Linux Kernel 4.19, with the addition of the "nospectre\_v1" parameter.

These three parameters were added despite the fact that the kernel already had existing "spectre\_v2" and "spec\_store\_bypass\_disable" options for months which allowed system admins to control the complexity level of the Spectre-class mitigations -- which also included an "off" mode.

But Sys Admins wanted some way to guarantee that Spectre mitigations would not somehow kick in, at all, ever, no matter what. So now the most recent kernel releases have three newer parameters.

The latest effort to have mitigations turned off --and stay off-- is the addition of the PR\_SPEC\_DISABLE\_NOEXEC control bit to the Linux kernel. This bit prevents child processes from starting in a state where the protections for Spectre v4 are still activated, despite being deactivated in the parent process.

Experts argue that some processes just don't need Spectre protections and the performance impact far outweighs the security impact, especially in closed systems where malicious code can't be introduced, such as graphics rendering farms, off-the-grid supercomputers, and other strictly confined systems where no third-party code is ever run. /ZDNet

And I think that's exactly right. As we've noted before, the only real -- even theoretically practical -- threat here is shared hosting systems where potentially malicious code might be running in an adjacent virtual machine. But things like Linux-based routers and many other server scenarios don't have any such exposure to possible-hostile code.

## Errata

Listener Bryan writes: Just finished SN699, where Steve tells everyone to pause the podcast, unplug your RV320, update it, then come back when that's done.

TLDR: According to Cisco, "Vulnerable Products: "This vulnerability affects Cisco Small Business RV320 and RV325 Dual Gigabit WAN VPN Routers running Firmware Releases 1.4.2.15 through 1.4.2.19." "Products Confirmed Not Vulnerable: "Only products listed in the Vulnerable Products section of this advisory are known to be affected by this vulnerability." That's from the security advisory for the Command Injection vulnerability; the Information Disclosure vulnerability only affects firmware releases 1.4.2.15 - 1.4.2.17.

I have family ties to a local business that's running an RV320, so as soon as I got home this evening I contacted them to begin planning for an ASAP update. Once I got logged into their router, saw their current firmware version, and read Cisco's Security Advisory notes, I relaxed a bit. They are running an older firmware version, not one that is affected.

Fwiw, Firmware version 1.4.2.15 appears to be the first 1.4.x release, which dropped 15-Sept-2017; and version 1.4.2.19 was released 29-Apr-2018. So this has been hanging out there for about 16 months. Not great, but much better than 5+ years.

That makes sense to me. Steve mentioned on the podcast that these were very popular "Enterprise and SMB" firewalls, but also mentioned that the number of exploitable devices found via scans was somewhere in the neighborhood of 9k. That number seemed low to me for such an ostensibly popular firewall. But if, apparently, only the last three firmware releases (before this newest, just-released 1.4.2.20) are subject to these attacks, that winnows down the field considerably. FYI, just in case anyone else starting running around with their hair on fire about this one. :-)

## SQLR Notes

I've been avoiding using the name of our illustrious XenForo coder, since I didn't know whether or not he wanted to remain anonymous. But I've been so impressed by his work that I wanted to give him an explicit shout out. He's "Rasmus Vind" with his own cool XenForo-driven site: (<https://www.hiveworkshop.com/>). He's a web developer who clearly knows his way around PHP, JavaScript, HTML, CSS and all the contemporary tools of the day... and I'm so glad that he happened upon the question I posted to the XenForo developer site back in May and asked whether he could be of any assistance.

**"Archimedes"** - Thursday at 6:48 PM: I've been lurking around the newsgroup for some time now, watching the development of SQLR with great anticipation. Seeing the news today that the forums were "SQLR ready" I figured I'd go ahead and take the plunge. What an incredibly seamless experience! Downloaded Steve's reference client and created an identity in less than 5 minutes (and that's only because I read everything slowly...). I purposefully didn't even install SQLR, and yet I was able to easily get going with the forums in Chrome. Logout and login is a snap! Incredible work, fantastic result. As a professional developer in the American corporate world, I really wish I had more time to help actively...but I'll definitely cheer on from the sidelines! -Archimedes

**"BrianOfLondon"** : All done! There's probably an orphan account called brianoflondon\_s that you can delete if you want. I've now associated this account with my SQLR ID which I'm using on my MacBook via Wine and on iPhone via Jeff's client. (That's Jeff Arthur in the UK.) Everything is working! Wordpress... as soon as that plugin is ready, boy do I want to try it out over on various wordpress sites.

### **Jason L.**

After I got locked out of my previous account (Jason), due to losing SQLR association and stupidly not creating a password or entering an email address, I decided it would probably be easier to just create a new account. This time I used a password and email address I then associated this account with my SQLR ID. It works. I have logged in several times using SQLR. I have even been able to use Jeff's Arthur's iOS client to have my computer login to the site. I think that is the coolest thing. I can't wait to demonstrate this to my friends and family's on their computers. I won't even touch their computers. I will just have them type the URL of these forums. They will think it is some sort of magic trick or I hacked them. LOL

I am assuming that once SQLR is stable here and there's no danger of losing associations again, there will be a way to remove password and/or email address; since the whole point of SQLR is that websites will not have secrets to keep safe. I realize that sites will probably have email addresses to store, but those aren't really secret and odds are they have already been stolen anyway. I know there is a feature in the client to only use SQLR to login, but figured that feature is not functional yet and I am not ready to do that until I know it is safe to do so.

## SpinRite

**Patrick McAuley in Guelph, Ontario, Canada**

Subject: Work on USB Drives?

Date: 29 Jan 2019 07:22:02

:

Simple question: will Spinrite work on an external USB-connected drive? Any PC Setup drive changes needed, as required for SATA drives?

## **Matt in Providence**

Subject: Stealth VPN to defeat VPN blocking WIFI

Date: 28 Jan 2019 12:30:51

:

Please add a show topic for this.

I have a need for VPN obfuscation/stealth VPN when using certain public wifi networks.

In the name of security, forget the terms of service.

I found a service named torguard for around \$5/mo.

I am also interested to know whether you can host your own OpenVPN server on aws that can act like a stealth VPN.

Or if running OpenWRT VPN on my home Linksys WRT router can act as a stealth VPN.

BTW I'm a proud licensee of SpinRite and it fixed a boot issue for me on a computer where I wasn't running it pre-emptively to prevent problems! Great product!

Thanks!

Notes: NordVPN is a sponsor on the TWiT network.

We were talking about the generic OpenVPN client the other day and I was ranting on, asking the question why would anyone use anything else.

For one thing, all of our experience tells us not to ever use some random, unknown, home-grown proprietary VPN system. And with the VPN problem already well and beautifully solved by OpenVPN, why would anyone.

The NordVPN site has a page which supports this notion:

<https://nordvpn.com/tutorials/windows-10/openvpn/>

OpenVPN – Windows 10 - You can set up a manual OpenVPN connection by using the OpenVPN GUI open-source application. This does not use the NordVPN application.

In other words, NordVPN is an example of exactly the way this should work. You have a good solid server-side provider (\$3/mo. with a 3-year commitment) And a bazillion servers located everywhere.

IKEv2/IPSec, OpenVPN UDP, OpenVPN TCP, Socks 5, HTTP Proxy, HTTP CyberSec Proxy, HTTP Proxy(SSL),HTTP CyberSec Proxy(SSL)

It appears to me that OpenWRT can function as an OpenVPN client, but not as a server. So if you wanted to roll your own OpenVPN server you would use pfSense running on any hardware of your choice.