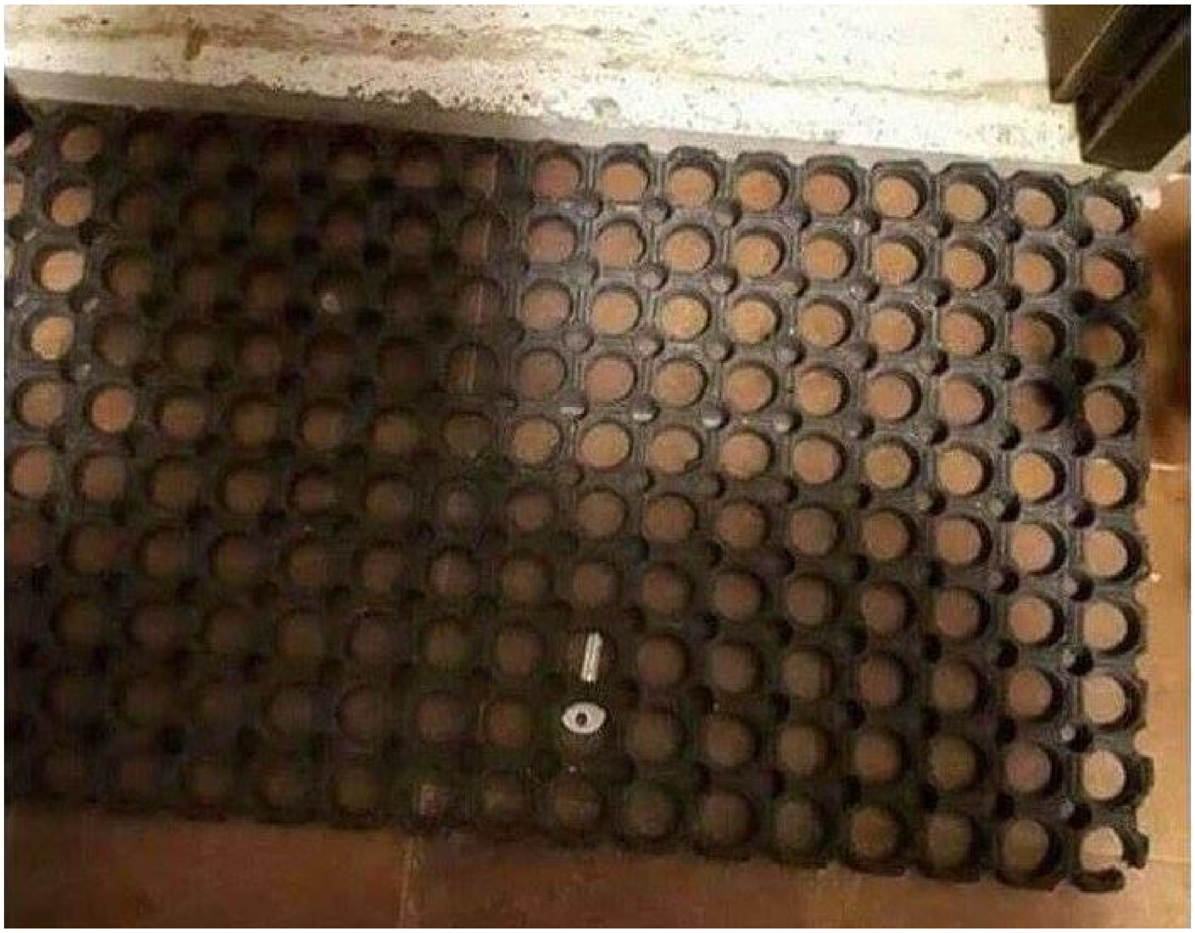
Security Now! #766 - 05-12-20
ThunderSpy

This week on Security Now!

This week we examine Firefox's recent move to 76 and slightly beyond, a wonderful new feature coming to Edge, the security responsibility that attends the use of WordPress, vBulletin and other complex and sophisticated web applications. We look at the plans for this summer's much-anticipated Black Hat and DEF CON conferences, a newly revealed CRITICAL bug affecting all of the past 6 years of Samsung Smartphones, and Zoom's latest security-boosting acquisition. I'll then provide an update on my SpinRite work which includes a bit of a rearrangement in sequence to provide another shorter-term deliverable. And then we look at the new ThunderSpy vulnerability that has the tech press huffing and puffing.

Hiding the key under the mat??



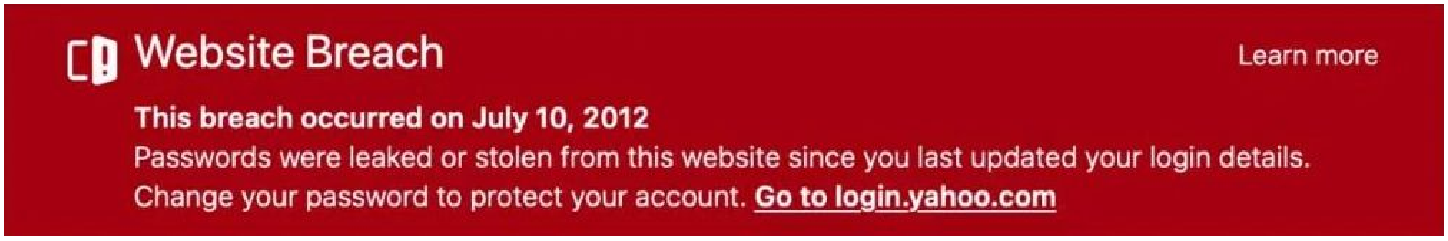
Browser News

Firefox 76 released with new features

Last Tuesday Mozilla released Firefox 76 for our desktop OSes, Windows, macOS, and Linux. This release fixed a few bugs and added a handful of nice new features.

It was around this time last year that Mozilla first put a version of their "Lockwise" password manager out for trial. Lockwise was a homegrown Firefox extension and today, a year later, it has acquired a following, continued support and feature growth. So, Firefox 76 adds a few new features and twists to Lockwise.

Lockwise will now proactively notify its user when a website for which the user has stored credentials has suffered and reports a data breach that might compromise those saved login credentials.



Website Breach

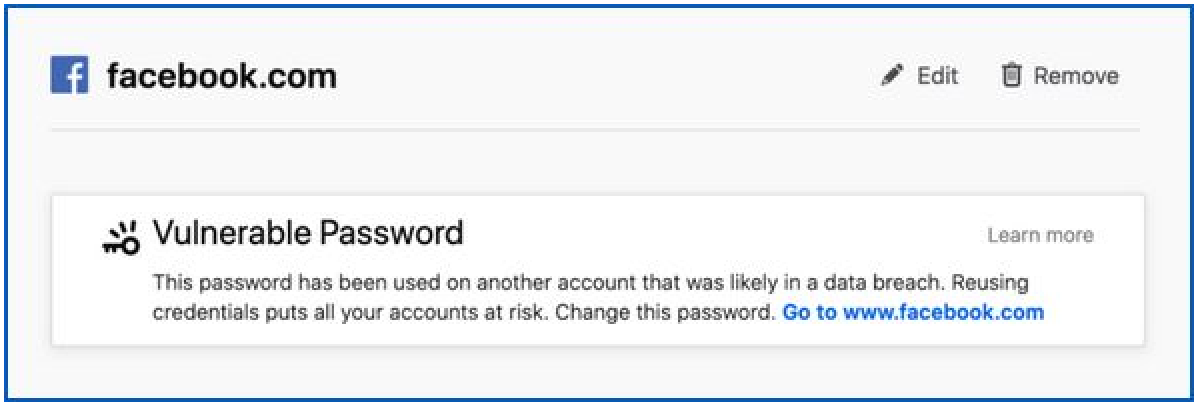
Learn more

This breach occurred on July 10, 2012

Passwords were leaked or stolen from this website since you last updated your login details. Change your password to protect your account. **Go to login.yahoo.com**

Lockwise will also warn users when their account credentials are actually found in a data breach dump and when that password is found to be reused on other websites.

A variation on this is the related "Vulnerable Password" warning:



facebook.com

Edit

Remove

Vulnerable Password

Learn more

This password has been used on another account that was likely in a data breach. Reusing credentials puts all your accounts at risk. Change this password. Go to www.facebook.com

As the notice indicates, it tells its user if any of the passwords Lockwise is aware of have been used on another account that may have been subject to a data breach.

And in a nice security-tightening measure, before displaying a user's saved login credentials, Lockwise will now fallback to requiring any users who have not established a master password for Lockwise to enter their local computer's login account password. This will prevent anyone who might obtain access to your logged-in computer session from snooping into your stored username and passwords. I have freaked out several users of Chrome by showing them how easy it is to have Chrome unmask ALL of the passwords they have asked their Chrome browser to save for them. Virtually no one appreciates that the entire list of past saved usernames and passwords is displayable after a few clicks. And it's always quite jarring. So this seems like a useful new feature.

We previously mentioned that Firefox would be adding a video "pop-out" feature to lift a page's playing video out into a separate floating window. That feature went live in Firefox 76. (Bleeping Computer noted that the feature didn't yet appear to be working reliably. So I imagine that it'll get more solid with future iterations.)

This release 76 also fixed 11 CVE-numbered security vulnerabilities, three of which was rated CRITICAL:

- CVE-2020-12387: Use-after-free during worker shutdown.
  As we know, any allowance of the use of freed memory is never good. So Mozilla notes that this results in a "potentially exploitable crash."

- CVE-2020-12388: Sandbox escape with improperly guarded Access Tokens.
  Since a sandbox escape means that a web page's rogue content might be able to escape from its confinement and interfere with other pages or with the host operating system… that's not good either.

- CVE-2020-12395: Memory safety bugs fixed in Firefox 76 and Firefox ESR 68.8.
  The third is really a sort of catchall for eight different subtle memory bugs that were found during Mozilla's routine security reviews and testing. So they're all fixed.

But Firefox 76 broke Amazon's Assistant!
Shortly after Firefox 76 first appeared, Mozilla encountered a pair of newly introduced bugs that it could not ignore. One caused the Amazon Assistant browser extension to no longer function correctly and the other fixed an obscure browser crash affecting Windows 7 32-bit systems with Nvidia graphics.

So Mozilla paused 76's continuing rollout until those could be fixed. Last Friday, three days after the release of 76, 76.0.1 became available to all Firefox users. Since I run with an instance of Firefox open permanently and maximized in a lower left hand screen, none of those updates were happening here. So I went to the About/Firefox dialog and was immediately asked for permission to restart Firefox. Which I did, of course, and now I have 76.0.1.

Hallelujah!! Edge moves to silence those annoying notification requests.
When I'm in research mode, as I have been when doing fact-finding for ideas about the best ways to accomplish programming tasks that I haven't encountered before, I tend to roam around the Net a bit, straying from the beaten path. So, the other day I encountered a site that explained that in order to enable whatever it was that I wanted to do -- I no longer recall what, perhaps download something -- I needed to "enable downloads" by clicking the "Allow" buttons on the Notifications drop down. And there it was, sure enough, patiently hanging down below the URL waiting for me to either "Allow" or "Block" any and all of the website's future push notifications.

We've talked about this new and EXTREMELY abuse-prone facility before. It allows websites to which you have given permission to henceforth interact with your operating system, linking into its native notification system, to bother you anytime the site wishes to. Some ideas, like always needing to get my permission to use cookies, are very bad. This whole background website notification system is another bad idea.

So, blessedly, Edge is getting a feature that I sure hope the two browsers I use (Firefox and Chrome) will immediately adopt! Under "Site Permissions / Notifications" is:

"Quiet notification requests" with the description “This will prevent notification requests from interrupting you.”

With “Quiet notification requests” enabled, the annoying permission drop down will be replaced by a small and easy-to-ignore bell icon at the far right of the URL address bar. Then, if for some reason you should want to see the notification -- for example if you're being held ransom over accepting notifications before the site will do what you want -- you can click the bell to manage notifications or allow them for this site. And note that Edge is also providing a very clean “Manage” facility that will allow you to easily revoke any notification permissions you may come to regret.

The moral to this story is that sometimes you need to go looking for what you want. After writing this I became curious about Chrome and Firefox. And I’m glad I did, because BOTH of those browsers ***already*** offer the functional equivalent of this feature, and now it is enabled on both of the browsers I use daily. Open either browser’s “Settings” and use the page’s Search function to search for “notifications.”

Now... if we could just get a uniform API for all of those incessant “This site to use cookies, okay?” permission banners I would set mine to "Curmudgeon", which is short for "Yes yes yes cookies, I get it. Now don't ever ask me again!"

Security News

WordPress in the crosshairs | We now transition from browsers to browser apps.

WordPress is used by more than 60 million websites, including a bit more than 1/3rd the top 10 million websites as of this time last year. WordPress is a complex system written in PHP which supports 3rd-party add-ons. The terms “complex” and “3rd-party plug-ins” are both antithetical to security.

Starting near the end of last month and increasing since, Defiant, the publishers of the Wordfence security plugin, detected someone running a botnet or proxy system who began attacking WordPress sites from at least 24,000 separate IP addresses. For example, on just one day, the Sunday before last (May 3rd) Defiant logged over 20 million attacks against more than half a million websites. Defiant said that the attackers focused mostly on exploiting cross-site scripting (XSS) vulnerabilities in plugins that had already received fixes months or even years ago and had previously been targeted in other attacks.

Multiple vulnerabilities have been targeted, but there are five most targeted:

- A XSS vulnerability in the Easy2Map plugin, which was removed from the WordPress plugin repository in August of 2019, and which they estimate is likely installed on less than 3,000 sites. This accounted for more than half of all of the attacks.

- A XSS vulnerability in Blog Designer which was patched in 2019. They estimate that no more than 1,000 vulnerable installations remain, though this vulnerability was the target of previous campaigns.

- An options update vulnerability in WP GDPR Compliance patched in late 2018 which would allow attackers to change the site's home URL in addition to other options. Although this plugin has more than 100,000 installations, they estimate that no more than 5,000 vulnerable installations remain.

- An options update vulnerability in Total Donations which would allow attackers to change the site's home URL. This plugin was removed permanently from the Envato Marketplace in early 2019, and they estimate that less than 1,000 total installations remain.

- A XSS vulnerability in the Wordpress "Newspaper" theme which was patched in 2016. This vulnerability has also been targeted in the past.

The takeaway here is obvious to all of us: Admins of WordPress sites **must** be responsible about updating their plugins and remove those that are no longer present in the WordPress repository.

[And, if we haven't got the attention of all Wordpress admins yet…]

Critical WordPress plugin bugs present on over one million sites
Wordfence's Threat Intelligence team also reported last week that they had detected a new campaign attempting to abuse two other bugs in WordPress in ongoing attacks. These attacks are attempting to actively exploit security vulnerabilities in the Elementor Pro and Ultimate Addons for Elementor. The flaws potentially allow for the remote execution of arbitrary code and the full compromise of unpatched targets.

https://www.wordfence.com/blog/2020/05/combined-attack-on-elementor-pro-and-ultimate-addons-for-elementor-puts-1-million-sites-at-risk/

On May 6, 2020, our Threat Intelligence team received reports of active exploitation of vulnerabilities in two related plugins, Elementor Pro and Ultimate Addons for Elementor. We have reviewed the log files of compromised sites to confirm this activity.

As this is an active attack, we wanted to alert you so that you can take steps to protect your site. We are intentionally limiting the amount of information this post provides, because this is an ongoing attack, and the most critical vulnerability has not yet been patched. There are two plugins affected by this attack campaign. The first is Elementor Pro which is made by Elementor. This plugin has a zero day vulnerability which is exploitable if users have open registration.

UPDATE: May 7 2020, Elementor has released version 2.9.4 of Elementor Pro. Our threat intelligence team has verified that this patches this vulnerability. We recommend updating to this version immediately.

The second affected plugin is Ultimate Addons for Elementor, which is made by Brainstorm Force. A vulnerability in this plugin allows the Elementor Pro vulnerability to be exploited, even if the site does not have user registration enabled.

We estimate that Elementor Pro is installed on over 1 million sites and that Ultimate Addons has an install base of roughly 110,000.

To be clear, this does not impact the free Elementor plugin with over 4 million installations available from the WordPress plugin repository. The Elementor Pro plugin is a separate download available from the Elementor.com website. We estimate that Elementor Pro has over 1 million active installations.

The vulnerability in Elementor Pro, which is rated Critical in severity, allows registered users to upload arbitrary files leading to Remote Code Execution. This is a zero day vulnerability.

An attacker able to remotely execute code on your site can install a backdoor or webshell to maintain access, gain full administrative access to WordPress, or even delete your site entirely. Due to the vulnerability being unpatched at this time, we are excluding any further information.

We have data via another vendor that indicates the Elementor team are working on a patch. We have contacted Elementor and did not immediately receive confirmation of this before publication.

The Ultimate Addons for Elementor plugin recently patched a vulnerability in version 1.24.2 that allows attackers to create subscriber-level users, even if registration is disabled on a WordPress site.

Attackers are able to directly target the zero day vulnerability in Elementor Pro on sites with open user registration.

In cases where a site does not have user registration enabled, attackers are using the Ultimate Addons for Elementor vulnerability on unpatched sites to register as a subscriber. Then they proceed to use the newly registered accounts to exploit the Elementor Pro zero day vulnerability and achieve remote code execution.
What you should do

There are a number of steps a site owner not using Wordfence Premium can take to protect their site from this active attack.

Upgrade Ultimate Addons for Elementor immediately. Make sure Ultimate Addons for Elementor is version 1.24.2 or greater.

Downgrade to Elementor free until a patch is released for Elementor Pro. You can do so by deactivating Elementor Pro and removing it from your site. This will remove the file upload vulnerability.

Once a patch is released, you can re-install the patched version of Elementor Pro on your site and regain any lost functionality. You may temporarily lose some design elements once downgraded, but in our tests these elements came back when reinstalling Elementor Pro. Nevertheless, a backup prior to downgrading is always prudent.

The takeaway from all this is that our software -- all of our software -- is becoming increasingly complex. And the development style that has evolved is powerfully modular, where 3rd-party code libraries are obtained and integrated at the code level, and 3rd-party plug-ins are obtained and integrated at the operational level. And in some cases that end point solution is further enhanced by an automation layer above which pushes its buttons. The result is a massive, complex, understandable-only-from-a-distance-in-overview, system about which no true assertion of knowable security can be made, because the system itself, at the level of detail required to make any such assertion, is unknowable… or at least unknown. Even if every modular component is itself secure (which history teaches is never effectively true), each module makes assumptions about the nature of its input and output, and those assumptions may not have been fully understood and appreciated by the developers who plugged all of the pieces together. As a result, the system will almost certainly be able to exhibit unexpected behavior when those assumptions are deliberately violated by attackers.

What we have created with this ad hoc assemblage of disparate components is inherently insecure. But it’s the choice we’ve made in the name of expediency. And it is seductively expedient. We obtain almost magical results when big systems are glued together. But what we don’t get is robust security. That’s not available with this approach.

So we’re left with doing the best we can. And that means monitoring and patching. And responsible patching requires knowing when there are patches. And knowing when there are patches requires maintaining a multitude of open lines of communication to the maintainers of every piece of this massive puzzle and then responding rationally with due speed when any significant new problem arises.

So, in summary, maintenance is the price we pay for expediency. And what we see is that all too often that price is not paid.

vBulletin

And against that backdrop, we have another example of a web-based security emergency:

vBulletin is widely used Internet forum software which currently powers more than 100,000 websites around the world. It's written in PHP with a SQL backend and is in use by many of the Fortune 500 and other major companies. Several of that system's early developers had a falling out with management and left to form XenForo where they started over from scratch to create a next generation system.

But vBulletin remains a very active going concern, and the maintainers of vBulletin recently announced a CRITICAL patch update without revealing any information or details about the underlying security vulnerability which is only identified by its CVE tracking designator of 2020-12720. But the word on the street is, anyone running vBulletin MUST UPDATE IMMEDIATELY to the latest release in their major release track.

Because vBulletin remains quite popular on the web, and because being written in PHP it is effectively running its own source code, it is one of the hackers' favorite targets. So the maintainers are clearly hoping that withholding the details of the flaw could buy some time for their users to apply the patch before hackers can reverse-engineer the fix and use that to exploit those sites which have not yet updated.

However, as has occurred previously, researchers and hackers both have already started reverse-engineering the patch to locate and understand the vulnerability. The National Vulnerability Database (NVD) has analyzed the flaw and revealed that it is indeed CRITICAL, and originating from an incorrect access control issue.

vBulletin's bulletin said: *"If you are using a version of vBulletin 5 Connect prior to 5.5.2, it is imperative that you upgrade as soon as possible."* Nothing has been said about whether vBulletin v3 and v4 might also be affected. If so that's really bad since there are still a large number of those old and now out of maintenance sites online -- almost as many v3 and v4 sites combined as v5.

This is another unfortunate thing we see with this sort of software: Just as with Microsoft finally refusing to maintain Windows 7 despite its still massive installed base, the vendors of complex web forum software occasionally produce a major upgrade that requires effort to make the move. I've already done that once with the XenForo forums I maintain. And it's necessary, since with time the maintenance of the older versions will fall off and then you're left with something that's no longer maintained and vulnerable forever. It's truly a mess.

Last September something very similar to this occurred with vBulletin when a remote code execution vulnerability was discovered and patched, and sites were urged to upgrade immediately. Many didn't and many were successfully attacked because the attackers waste zero time in pouncing. This is why users of these systems MUST be able to dedicate the time of someone who is just as committed to patching as the attackers.

In this case, although there is currently no proof-of-concept code available and no indication that this new vulnerability is being exploited in the wild, if history is any guide we may be taking about it next week.

But the clock is ticking since Charles Fol, a security engineer at Ambionics who was the confirmed discoverer of the vulnerability who also responsibly reported it to the vBulletin team, has stated that he plans to release more information during the French SSTIC security conference scheduled for the 3rd through the 5th of June, early next month... so, three weeks from tomorrow.

Forum administrators are advised to download and install respective patches for the following versions of their forum software as soon as possible.

5.6.1 Patch Level 1
5.6.0 Patch Level 1
5.5.6 Patch Level 1

Leo… for the first time ever I might consider actually attending this summer's Black Hat and DefCon security conferences! And, in fact, I would encourage all of our listeners to do so!

They are, as we know, the two biggest cyber-security conferences of the year. But, of course, for the first time ever they will not be held in Las Vegas. Thanks to the Coronavirus, they'll be going virtual!

The two conferences were initially scheduled to take place in Las Vegas, sequentially back-to-back during the first two weeks of August, with Black Hat from August 1 to August 6, and DEF CON from August 7 to the 9th.

But now they're following in the footsteps of many other cyber-security conferences that have necessarily switched into the cyber world to discuss cyber security. We don't have many details at the moment, but we do know that both conferences are planning to live-stream talks to paying attendees. Since no changes in dates have been announced, both conferences are expected to take place during their previously announced dates.

The Black Hat team has only posted a short statement, but Jeff Moss, DEF CON's manager went into some detail about what led up to the decision to go virtual:

Jeff wrote: "While I made the decision to cancel the in-person conference a month ago on April 11th, the delay in announcing has been due to learning how to actually cancel. It has taken weeks of working with staff, lawyers, accountants, and Caesars Palace. I didn't want to endanger the future of the conference by tweeting that we were canceling before we understood and were confident we could navigate the process."

Going forward, Jeff has said that DEF CON will continue to be an in-person event. Next year's edition, DEF CON 29, is currently scheduled to be an in-person event, planned for August 5-8, 2021.

Samsung has patched a CRITICAL bug affecting the past 6 years of Smartphones
Since 2014, the Android OS flavor running on Samsung devices have included a handler for the custom "Qmage" image format (.qmg). A security researcher with Google's Project Zero discovered a way to exploit how Skia, which is the Android graphics library, handles these Qmage images sent to a device. And this can be exploited as a 0-click attack through the reception of MMS messages without any user interaction.

This occurs because Android redirects all images sent to a device through the Skia library for processing -- such as generating thumbnail previews.

Google's researcher developed a proof-of-concept demo exploiting the bug against the Samsung Messages app, which is included in all Samsung devices and is the handler of all SMS and MMS messages. The bug was exploited by sending repeated MMS messages to a Samsung device. Each message attempted to guess the position of the Skia library in the Android phone's memory which is required to brute-force Android's ASLR (Address Space Layout Randomization) protection. Once the Skia library has been located in memory, a final MMS message delivers the actual Qmage payload, which executes the attacker's code on a device.

The attack typically requires between 50 and 300 MMS messages to probe and bypass ASLR, which usually takes around 100 minutes, on average. And while the attack might look noisy, it can be improved to execute without alerting the user.

The researcher wrote: “I have found ways to get MMS messages fully processed without triggering a notification sound on Android, so fully stealth attacks might be possible.”

The vulnerability was discovered in February and reported to Samsung who then patched the bug in this month's May 2020 security updates.

Thanks to ASLR, the attack is functionally limited to targeted attacks. So if you or someone you know are a Samsung smartphone user who might be a target of a targeted attack, be certain to obtain this month's Samsung patches.

No other smartphones appear to be impacted because only Samsung appears to have modified the Android OS to support the custom Qmage image format, which was developed by a South Korean company “Quramsoft.”

Zoom purchases Keybase
Last Thursday, May 7th, Zoom's CEO Eric Yuan blogged the news of their latest move for furthering the securiTy of their wildly successful teleconferencing platform:

https://blog.zoom.us/wordpress/2020/05/07/zoom-acquires-keybase-and-announces-goal-of-developing-the-most-broadly-used-enterprise-end-to-end-encryption-offering/

(And note from the URL that, speaking of WordPress, Zoom's blog is hosted on WordPress.)

Title: ***“Zoom Acquires Keybase and Announces Goal of Developing the Most Broadly Used Enterprise End-to-End Encryption Offering”***

We are proud to announce the acquisition of Keybase, another milestone in Zoom’s 90-day plan to further strengthen the security of our video communications platform. Since its launch in 2014, Keybase’s team of exceptional engineers has built a secure messaging and file-sharing service leveraging their deep encryption and security expertise. We are excited to integrate Keybase’s team into the Zoom family to help us build end-to-end encryption that can reach current Zoom scalability.

This acquisition marks a key step for Zoom as we attempt to accomplish the creation of a truly private video communications platform that can scale to hundreds of millions of participants, while also having the flexibility to support Zoom’s wide variety of uses. Our goal is to provide the most privacy possible for every use case, while also balancing the needs of our users and our commitment to preventing harmful behavior on our platform. Keybase’s experienced team will be a critical part of this mission.

Today, audio and video content flowing between Zoom clients (e.g., Zoom Rooms, laptop computers, and smartphones running the Zoom app) is encrypted at each sending client device. It is not decrypted until it reaches the recipients’ devices. With the recent Zoom 5.0 release, Zoom clients now support encrypting content using industry-standard AES-GCM with

256-bit keys. However, the encryption keys for each meeting are generated by Zoom's servers. Additionally, some features that are widely used by Zoom clients, such as support for attendees to call into a phone bridge or use in-room meeting systems offered by other companies, will always require Zoom to keep some encryption keys in the cloud. However, for hosts who seek to prioritize privacy over compatibility, we will create a new solution.

Zoom will offer an end-to-end encrypted meeting mode to all paid accounts. Logged-in users will generate public cryptographic identities that are stored in a repository on Zoom's network and can be used to establish trust relationships between meeting attendees. An ephemeral per-meeting symmetric key will be generated by the meeting host. This key will be distributed between clients, enveloped with the asymmetric keypairs and rotated when there are significant changes to the list of attendees. The cryptographic secrets will be under the control of the host, and the host's client software will decide what devices are allowed to receive meeting keys, and thereby join the meeting. We are also investigating mechanisms that would allow enterprise users to provide additional levels of authentication.

These end-to-end encrypted meetings will not support phone bridges, cloud recording, or non-Zoom conference room systems. Zoom Rooms and Zoom Phone participants will be able to attend if explicitly allowed by the host. Encryption keys will be tightly controlled by the host, who will admit attendees. We believe this will provide equivalent or better security than existing consumer end-to-end encrypted messaging platforms, but with the video quality and scale that has made Zoom the choice of over 300 million daily meeting participants, including those at some of the world's largest enterprises.

As we do this work to further protect our users' privacy, we are also cognizant of our desire to prevent the use of Zoom's products to cause harm. To that end, we will be taking the following steps:

- We will continue to work with users to enhance the reporting mechanisms available to meeting hosts to report unwanted and disruptive attendees.
- Zoom does not and will not proactively monitor meeting contents, but our trust and safety team will continue to use automated tools to look for evidence of abusive users based upon other available data.
- Zoom has not and will not build a mechanism to decrypt live meetings for lawful intercept purposes.
- We also do not have a means to insert our employees or others into meetings without being reflected in the participant list. We will not build any cryptographic backdoors to allow for the secret monitoring of meetings.

Next Steps

We are committed to remaining transparent and open as we build our end-to-end encryption offering. We plan to publish a detailed draft cryptographic design on Friday, May 22. We will then host discussion sections with civil society, cryptographic experts, and customers to share more details and solicit feedback. Once we have assessed this feedback for integration into a final design, we will announce our engineering milestones and goals for deploying to Zoom users.

We look forward to welcoming the Keybase team and are excited for the possibilities of what we can build together.

SpinRite

Since last week, my work on SpinRite has been focused upon implementing an updated boot prep system. I wrote the current SpinRite v6 Windows boot media preparation system 16 years ago, back in 2004, and it contains code for compatibility with operation on Windows 98 which, after all, was only 6 years old at the time. Back then, the most reliable boot medium was the 1.44 megabyte “double density” floppy disk and CD’s were second. Not all systems could boot USB, but as all SpinRite owners know, I supported them all.

The largest USB thumb drive capacity numbered in the hundreds of megabytes -- not gigabytes -- so a cylinder, head & sector (CHS) boot sector, which was able to handle drives up to 8 gigs seemed entirely adequate at the time. It certainly is no longer. What’s more, if SpinRite is to have a future -- and I’m committed to that -- it will need to be able to boot on either older BIOS-based hardware or UEFI systems without either a classic BIOS or DOS compatibility.

So I’m in the process of building a new boot media prep system that’ll give me control over all aspects of the drives it prepares. For now that just means that any USB thumb drive it’s asked to prepare, regardless of size, will be made bootable on any BIOS & DOS system. And in the future we’ll be ready to add UEFI and operation without any DOS OS at all.

And speaking of UEFI... One of the things that has been brought to my attention during this return to work heading toward a new series of SpinRite releases is the growing importance of UEFI. Intel no longer supports the traditional BIOS at all, and Apple has dropped the so-called CSM -- compatibility support module -- from their more recent offerings. So this all suggests that the BIOS’s days are numbered and that the number may not be very big. As everyone knows, my plan has been to produce a series of v6.x releases to bring SpinRite up to the latest hardware standards. The idea was for that to buy me the time to start over from scratch on the reconceptualization of SpinRite as a drive, filesystem and file-aware fully multitasking data maintenance and recovery tool -- which I am really looking forward to doing. But, with the BIOS disappearing I’m worried that SpinRite’s near-future users (and even recent Mac purchasers) may again be left without a way to run SpinRite.

So I’m thinking that SpinRite 7 should happen immediately after the SpinRite v6.x releases to address the disappearing BIOS and DOS problem by adding native UEFI booting and to run natively on the machine without an underlying OS. I’m obviously going to need that technology for what follows anyway, so it’s not a diversion, it just moves things around. And it gets us a SpinRite that ought to be able to span the time until SpinRite v8 is ready. (And, yes… “v8” is a super-cool name for SpinRite’s big rewrite.)

ThunderSpy

Okay... so first, here's the mixed-blessing summary by Björn Ruytenberg from the Eindhoven University of Technology, describing what his security research has uncovered:

Thunderspy targets devices with a Thunderbolt port. If your computer has such a port, an attacker who gets brief physical access to it can read and copy all your data, even if your drive is encrypted and your computer is locked or set to sleep.

[That sounds really bad, until he gets to the part about the screwdriver.]

Thunderspy is stealth, meaning that you cannot find any traces of the attack. It does not require your involvement, i.e., there is no phishing link or malicious piece of hardware that the attacker tricks you into using. Thunderspy works even if you follow best security practices by locking or suspending your computer when leaving briefly, and if your system administrator has set up the device with Secure Boot, strong BIOS and operating system account passwords, and enabled full disk encryption. All the attacker needs is 5 minutes alone with the computer, **a screwdriver**, and some easily portable hardware.

We have found 7 vulnerabilities in Intel's design and developed 9 realistic scenarios how these could be exploited by a malicious entity to get access to your system, past the defenses that Intel had set up for your protection.

We have developed a free and open-source tool, Spycheck, to determine if your system is vulnerable. If it is found to be vulnerable, Spycheck will guide you to recommendations on how to help protect your system.

So this clearly presents a worthwhile security advancement, but it's not really an "Evil Maid" attack... unless your Maid was trained at MIT.

To get a quick sense for what this means, the next step is to look at what he wrote in the Abstract of his formal security research paper. It provides some necessary and interesting background on Thunderbolt and what that implies:

https://thunderspy.io/assets/reports/breaking-thunderbolt-security-bjorn-ruytenberg-20200417.pdf

Thunderbolt is a proprietary I/O protocol promoted by Intel and included in a number of laptops, desktops, and other systems. As an external interconnect,it allows exposing the system's internal PCI Express (PCIe) domain to external devices. This enables high-bandwidth, low-latency use cases, such as external graphics cards. Being PCIe-based, Thunderbolt devices possess Direct Memory Access-enabled I/O, allowing complete access to the state of a PC and the ability to read and write all of system memory.

In recent years, the former characteristic (the ability to read and write all of system memory) has prompted research into attacks collectively known as evil maid, which require an attacker-controlled device and only seconds of physical access to the computer. Industry response has been two-fold. First, hardware and OS vendors incorporated support for DMA remapping using Input-Output Memory Management Units (IOMMUs), which imposes memory protections on DMA. However, following various implementation issues, OS vendors classified DMA remapping as an optional countermeasure requiring driver support. Second, revised Thunderbolt controllers introduced a software-based access control measure enabling users to authorize trusted devices only. As a result, unidentified devices should be barred from system access without prior user interaction.

In the context of Thunderbolt, studies have primarily focused on employing DMA and IOMMU attacks on the PCIe level. We therefore investigate the feasibility of breaking Thunderbolt protocol security, by analyzing the protocol and its software and hardware stack, as well as associated PCIe-based technology.

In our study, we have found and experimentally confirmed multiple vulnerabilities that break all primary Thunderbolt 3 security claims. Based on our ongoing research, in this report, we disclose the following vulnerabilities:

1. Inadequate firmware verification schemes,
2. Weak device authentication scheme,
3. Use of unauthenticated device metadata,
4. Backwards compatibility with legacy protocol versions,
5. Use of unauthenticated controller configurations,
6. SPI flash interface deficiencies, and
7. No Thunderbolt security on BootCamp.

Finally, we present nine practical exploitation scenarios.

Given an “evil maid” threat model and varying Security Levels, we demonstrate the ability to create arbitrary Thunderbolt device identities, clone user-authorized Thunderbolt devices, and finally obtain PCIe connectivity to perform DMA attacks. In addition, we show unauthenticated overriding of Security Level configurations, including the ability to disable Thunderbolt security entirely, and restoring Thunderbolt connectivity if the system is restricted to exclusively passing through USB and/or DisplayPort. We conclude this report by demonstrating the ability to permanently disable Thunderbolt security and block all future firmware updates.

We've used the very useful term “Security Perimeter” many times before. It's a conceptually clean way of establishing the idea of what's under protection, where the barrier to penetration is, what sort of protection is available and needed, and where the resulting vulnerabilities lie.
So, if nothing else, it seems very clear that this was necessary research. Even if its theoretical exploitability seems low, just look at how the purely theoretical Spectre and Meltdown vulnerabilities have hugely deepened our understanding of the exploitability of the many once-believed-to-be-safe CPU performance optimizations. So this is super useful stuff.

And there appears to be a real desire to export a system's internal bus through an easy-to-use serial connector. This dates back to the original 1394 Firewire where exploits of its exported hardware bus similarly afflicted that interface.

So here again it appears that just as with 1394 Firewire, security was layered on later, almost as an afterthought. On the other hand, some things cannot even theoretically be protected. As we observed a decade ago on this podcast, if the bad guys have access to the hardware -- like a DVD player which must locally decrypt its discs in order to play them -- then all bets are off.

Here, if you have an MIT trained "Evil Maid" with unfettered access to a system whose hardware is exporting its internal bus to the outside world, then yeah, once again all bets are off.
The research paper is 23 pages of detailed “here's how I did it” information, but a summary of the 7 vulnerabilities is not overly long and will give us a better sense for Thunderbolt’s measures and countermeasures:

1. **Inadequate firmware verification schemes:**
Thunderbolt host and device controllers operate using updatable firmware stored in its SPI flash. Using this feature, Thunderbolt hardware vendors occasionally provide firmware updates online to address product issues after release. To ensure firmware authenticity, *upon writing the image to the flash*, Thunderbolt controllers verify the firmware's embedded signature against Intel's public key stored in silicon. However, we have found authenticity is *not* verified at boot time, upon connecting the device, or at any later point. During our experiments, using a SPI programmer, we have written arbitrary, unsigned firmware directly onto the SPI flash. Subsequently, we have been able to verify Thunderbolt controller operation using our modified firmware.

   [So, in other words, the sanctioned way of writing the Thunderbolt controller firmware verifies the firmware's signature, but only at the time of that writing. So anything that bypasses the sanctioned means of updating the firmware can make any changes it wishes and the then-broken firmware will henceforth be run without ever being re-verified.]

2. **Weak device authentication scheme:**
As noted above, device identification comprises several strings and numerical identifiers. However, we have found none of the identifiers are linked to the Thunderbolt controller or one another, cryptographically or otherwise.

   [This means that impersonation of any previously authenticated device would be trivial. Briefly connect an identity capture device to anything that was previously permitted to connect. Read and capture the device's stated identity and simply impersonate that device to obtain full and unfettered access.]

3. **Use of unauthenticated device metadata:**
Thunderbolt controllers store device metadata in a firmware section referred to as Device ROM (DROM). We have found that the DROM is not cryptographically verified. Following from the first issue, this vulnerability enables constructing forged Thunderbolt device identities. In addition, when combined with the second issue, forged identities may partially or fully comprise arbitrary data.

   [So in other words, even if a previously authorized device is not available for identity cloning, it's possible to simply plant a malicious device's identity metadata into the system's Device ROM to give it access permission.]

4. **Backwards compatibility:**
Thunderbolt 3 host controllers support Thunderbolt 2 device connectivity, irrespective of Security Levels. This backwards compatibility subjects Thunderbolt 3 equipped systems to vulnerabilities introduced by Thunderbolt 2 hardware.

   [I didn't do any digging to determine what those vulnerabilities were, but presumably there were some and as a consequence of the need for backward compatibility, they are subject to returning.]

5. **Use of unauthenticated controller configurations:**
In UEFI, users may choose to employ a Security Level different from the default level. In storing Security Level state, we have determined that Thunderbolt employs two state machines, with one instance being present in UEFI, and another residing in host controller firmware. However, we have found firmware configuration authenticity is not verified at boot time, upon resuming from sleep, or at any later point. In addition, we have found

these state machines may be subjected to desynchronization, with controller firmware overriding UEFI state without being reflected in the latter. As such, this vulnerability subjects the Thunderbolt host controller to unauthenticated, covert overriding of Security Level configuration.

[So, again, it’s clear that the security of this system COULD have been made MUCH tighter.]

6. **SPI flash interface deficiencies:**
Thunderbolt systems rely on SPI flash to store controller firmware (vulnerability 1) and maintain their Security Level state (vulnerability 5). In our study, we have found Thunderbolt controllers lack handling hardware error conditions when interacting with flash devices. Specifically, we have determined that enabling flash write protection (1) prevents changing the Security Level configuration in UEFI, again without being reflected in the latter, and (2) prevents controller firmware from being updated, without such failures being reflected in Thunderbolt firmware update applications. As such, when combined with the fifth issue, this vulnerability allows for the cover and permanent, disabling of Thunderbolt security and will also silently block all future firmware updates.

7. **No Thunderbolt security on Boot Camp:**
Apple supports running Windows on Mac systems using the Boot Camp utility. Aside from Windows, this utility may also be used to install Linux. When running either operating system, Mac UEFI disables all Thunderbolt security by employing the Security Level “None” This vulnerability subjects the Mac system to trivial Thunderbolt-based DMA attacks.

So where does this leave us?

The complete lack of hardware-level enforcement of per-boot firmware verification, means that our current hardware systems CANNOT be made invulnerable to the “Evil MIT Grad” attack. Since the SPI chip holding the Thunderbolt firmware is on the system’s motherboard, and since SPI programmers are a dime a dozen, if someone with sufficient knowledge were to gain access to a machine they could override any protections.

But the more worrisome attack to my mind, since it’s something any old “Evil Maid” could pull off, when an authorized external Thunderbolt device is available, would be to simply unplug that device, plug it into a Thunderbolt device ID cloning tool, grab its identity, then turn around and impersonate the device with full access to the system’s internal PCIe bus.
While it would be nice to think that this could be fixed in the future, the fact that device identity metadata is not currently authenticated means that impersonation is trivial. And even if some future Thunderbolt 4 were to fix this, as it seems likely to... any backward compatibility to today’s or yesterday’s Thunderbolt would reopen the device to attack.

So our ultimate takeaway is that, by just about any definition of security, exporting a system’s internal bus is almost always a really bad idea.