Security Now! #785 - 09-22-20

Formal Verification

This week on Security Now!

This week we look at an important security update to Android for Firefox. We bid a fond farewell to Firefox Send and Notes, we look at the promise and growing popularity of the disastrously-named DuckDuckGo Internet search service, we dig into what's behind last Friday's Emergency Directive 20-04 from the DHS/CISA. We'll also take a look at the recent privacy and security improvements incorporated into Android 11 and iOS 14. We have a bit of errata, closing the loop feedback, and SpinRite news. Then we're going to take a look at the need for Formal Verification of our complex security protocols going forward in the context of another critical failure of a massively widespread system.

Browser News

Update to Firefox 79 for Android

An important LAN attack bug was recently fixed in Firefox 79 for Android.

Firefox locates other devices on the same LAN to share or receive content. An example might be sharing video streams with a Roku player. To accomplish this, Firefox uses the Simple Service Discovery Protocol, SSDP. And if that sounds familiar to you it's because we've often spoken of it and its mixed-blessing capabilities, limitations, and security shortfalls. Although it was originally intended to be a freestanding protocol, its adoption by the infamous Universal Plug n' Play (UPnP) system moved it into the UPnP specification. It's a simple text-based protocol based on HTTP over UDP. The system uses LAN-wide multicast addressing. On an IPv4 LAN, packets set to port 1900 at the IP address 239.255.255.250 will be heard and received by all devices on the LAN.

So, Firefox uses SSDP — Simple Service Discovery Protocol — to query all devices on a local network for anything that Firefox might be interested in sharing. When a device is discovered, Firefox's SSDP component retrieves the location of an XML file where that discovered device's configuration is stored. Chris Moberly, an Australian security researcher working for GitLab discovered that Android "Intent" commands could be placed into this returned XML file and that upon Firefox's retrieval of the file on Firefox for Android, those embedded commands would be executed.

The Android developer page explain “Intents” with the title of “Sending the User to Another App”, as follows:

> One of Android's most important features is an app's ability to send the user to another app based on an "action" it would like to perform. For example, if your app has the address of a business that you'd like to show on a map, you don't have to build an activity in your app that shows a map. Instead, you can create a request to view the address using an Intent. The Android system then starts an app that's able to show the address on a map.
>
> As previously explained, you must use intents to navigate between activities in your own app. You generally do so with an explicit intent, which defines the exact class name of the component you want to start. However, when you want to have a separate app perform an action, such as "view a map," you would use an implicit intent.
>
> Implicit intents do not declare the class name of the component to start, but instead declare an action to perform. The action specifies the thing you want to do, such as view, edit, send, or get something. Intents often also include data associated with the action, such as the address you want to view, or the email message you want to send. Depending on the intent you want to create, the data might be a Uri, one of several other data types, or the intent might not need data at all.

Okay. So it's probably not difficult for any of our listeners to figure out that Android Intents are powerful and not something that we'd like a remote attacker to be able to cause our Firefox browser to execute at will. But until Firefox for Android 79, that's exactly what was possible...

A hacker might walk into any environment with a large WiFi network, say an airport or a large mall. The attacker connects to the local WiFi network like everyone else has, and begins spamming the network with SSPD packets advertising its own maliciously-crafted XML file containing Android Intents. Any Android user who was using Firefox to browse the web during this attack could have their browser hijacked and taken to a malicious site, or forced to install a malicious Firefox extension.

https://gitlab.com/gitlab-com/gl-security/security-operations/gl-redteam/red-team-tech-notes/-/tree/master/firefox-android-2020

Last week, now that Firefox for Android has been updated, the vulnerability's discoverer posted further details along with PoC code and videos. Chris Moberly explains:

> The SSDP engine in Firefox for Android (68.11.0 and below) can be tricked into triggering Android intent URIs with zero user interaction. This attack can be leveraged by attackers on the same WiFi network and manifests as applications on the target device suddenly launching, without the users' permission, and conducting activities directed by the intent.
>
> The target simply has to have the Firefox application running on their phone. They do not need to access any malicious websites or click any malicious links. No attacker-in-the-middle or malicious app installation is required. They can simply be sipping coffee while on a cafe's WiFi, and their device will start launching application URIs under the attacker's control.
>
> I discovered this bug while the newest version of Firefox Mobile v79 was being rolled out globally. Google Play Store was still serving a vulnerable version at this time, but only for a short period. I reported the issue directly to Mozilla, just to be safe. They responded right away and were quite pleasant to work with, providing some good info on where exactly this bug came from. They were able to confirm that the vulnerable functionality was not included in the newest version and opened some issues to ensure that the offending code was not re-introduced at a later time.
>
> If you find a Firefox bug, I definitely recommend sending it straight to them. The process is very easy, the team members are smart and friendly, and it's a good way to support a project that has helped shape the way we use the web.
>
> As long as you have app updates enabled and have recently connected to WiFi, you should have received the new version and are safe from exploitation. You can verify this yourself by opening Firefox on your device, clicking the three dots next to the address bar, and navigating to "Settings -> About Firefox". If your version is 79 or above, you are safe.

Goodbye Forever Firefox “Send” and “Notes”... *(oh, how we loved ye)*
It was interesting and encouraging to hear Chris’ feedback about his vulnerability-reporting experience with the Firefox/Mozilla people. And this brings us to my final sad update on the disposition of Firefox “Send”.

As our listeners know, the encrypted file sending service was a favorite of Leo's and mine. So, ever since the service was suddenly suspended due to rampant abuse, I've been checking back from time to time, keeping tabs on it. And I've mentioned it several times here, wondering out loud whether something more must be going on since how difficult could it be to add some stronger anti-abuse features, like requiring a verified eMail address?

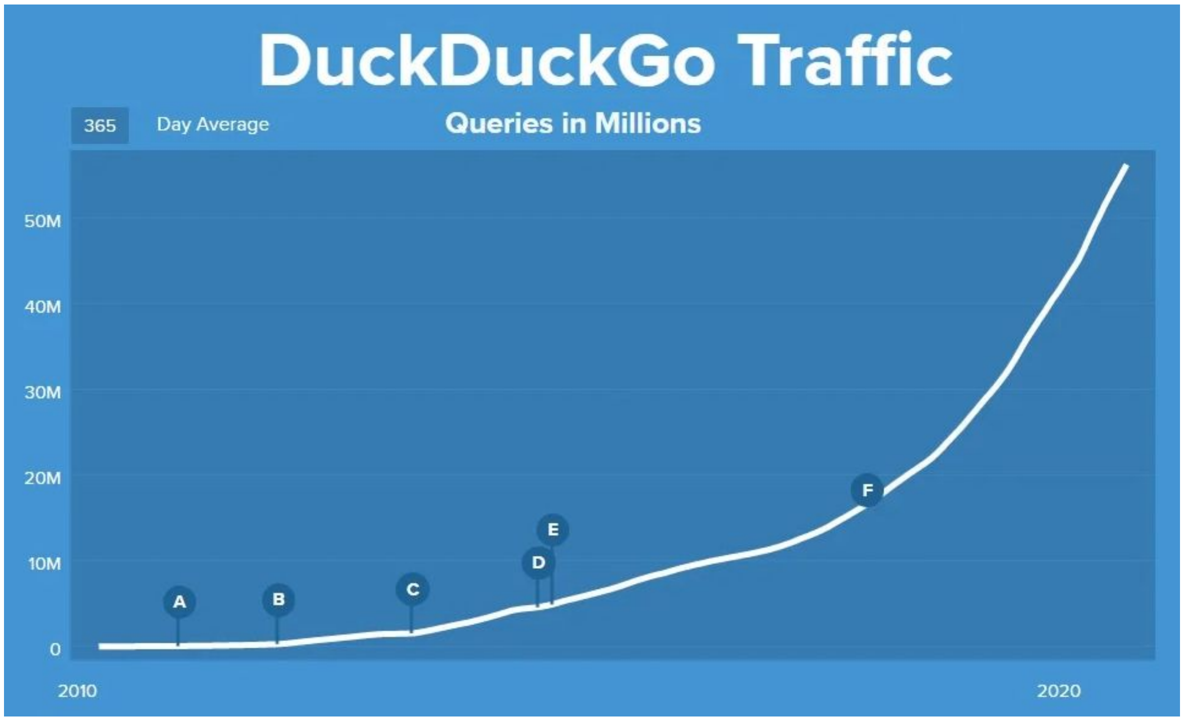
Well, now the other shoe has dropped... And we know that not only is “Send” gone for good, but so is whatever “Notes” was. (It was an inter-Firefox note-sync service.) “Send” was launched in March of 2019 and it became quite popular because it was one of the few dead-simple easy-to-use end-to-end encrypted and free large file sending solutions. Unfortunately, it wasn't only legitimate users who found it handly.

“Send” was initially taken offline after ZDNet reported that the service was being constantly abused by malware groups who liked both its end-to-end encryption and the fact that incoming files originated from “firefox.com”, a legitimate-looking domain. At that time, Mozilla said that Send's shutdown was temporary and promised to find a way to curb the service's abuse in malware operations. But after quite a bit more than a few weeks, things changed. As we recently noted here, Mozilla has recently laid off more than 250 employees as part of an effort to refocus its business on commercial products. So, today, most of the staff that was supposed to be working to re-engineer Send is gone, and the ones who are still around are now working on commercial products, such as Mozilla VPN, Firefox Monitor, and Firefox Private Network.

“DuckDuckGo” usage growth goes exponential

While we're talking about browsers and privacy, I thought that I ought to mention that Thursday before last the DuckDuckGo Twitter account tweeted:

A Record Breaking August at DuckDuckGo!
2B searches
4M app/extension installs
65M guesstimated active users (we don't actually know because... privacy)



Now, I'll confess to having an immediate problem with the name. DuckDuckGo? Really? Is this a serious operation? So I did my first bit of digging. If nothing else I was hoping for some clarity on the name.

What I learned was that DuckDuckGo is known (though not yet widely enough) as the search engine that doesn't -- and won't -- track its users. Of course. it aims to display relevant, timely results to their users. But their emphasis on protecting searcher privacy alters the process of how they collect, store and display information in their search results.

DuckDuckGo was launched 12 years ago in 2008 by Gabriel Weinberg with the goal of improving the search experience by reducing spam and including instant answers. After selling his previous company, Opobox, to classmates.com for $10 million in 2006, Gabriel found himself juggling a few different concepts: structured data, Quora-style Q&A and programmatically combating search spam.

He said: "I started all of these projects independently and none of them really took off. Then I thought, maybe if I put them all together, there might be an interesting experience there." The result was DuckDuckGo, a search engine offering direct answers to people's queries, rather than merely delivering a list of links. And below these Instant Answers, the site displays traditional, link-by-link search results syndicated from third parties like Bing, Yelp and Yahoo!.

However, unlike Google, DuckDuckGo has a no filter bubble policy, it doesn't track your browser, search history, device type, IP address, and usage behavior — and EVERONE receives the same results from the same search query. Even when users are not logged into Google, personalized results are still displayed. So the same search query for different users will display different results, sometimes drastically different results. And by compiling all this information across different products (email, mobile, maps, etc.), Google not only knows who you are, but according to Eric Schmidt, parent company Alphabet's executive Chairman, "we know where you've been and we can more or less know what you're thinking about."

To which Gabriel Weinburg would reply: "As long as you can tie searches together and you keep any shred of the information, any personal information that can tie things back to you, then I think it's not truly private."

And as for the name. That was unsatisfying. All I could learn was that Gabriel liked it. (God knows why.) But I'm sure we're stuck with it now. The name comes from a supposedly popular children's game, Duck, Duck, Goose (at least he shortened it).

So, we had never talked much about this search engine alternative, despite all the time we spend talking about the technological ways and means of tracking. These guys don't track, and their user base is growing fast. Despite their dumb name, they don't seem so dumb.

Security News

Microsoft's catastrophic "Zerologon" vulnerability.

Here, on Tuesday, September 22nd, we're now two weeks downstream from THIS month's patch Tuesday and six weeks from August's patch Tuesday. And after last week's blistering summary of the need to apply THIS month's patches I cannot imagine that anyone would not have done so by now, let alone LAST month's patches. But, difficult as it might be to believe, apparently not everyone listens to this podcast every week. I know, I know. Some things just cannot be explained. That's okay. For those who are listening, here's what's just happened...

After giving the world what should have been ample time to apply the August Windows updates, last Monday, Secura responsibly published a detailed behind-the-scenes description of just one of the nightmares that Microsoft fixed **last** month. Their blog posting begins:

https://www.secura.com/blog/zero-logon

Last month, Microsoft patched a very interesting vulnerability that would allow an attacker with a foothold on your internal network to essentially become Domain Admin with one click. All that is required is for a connection to the Domain Controller to be possible from the attacker's viewpoint.

Secura's security expert Tom Tervoort previously discovered a less severe Netlogon vulnerability last year that allowed workstations to be taken over, but the attacker required a Person-in-the-Middle (PitM) position for that to work. Now, he discovered this second, much more severe (CVSS score: 10.0) vulnerability in the protocol. By forging an authentication token for specific Netlogon functionality, he was able to call a function to set the computer password of the Domain Controller to a known value. After that, the attacker can use this new password to take control over the domain controller and steal credentials of a domain admin.

The vulnerability stems from a flaw in a cryptographic authentication scheme used by the Netlogon Remote Protocol, which among other things can be used to update computer passwords. This flaw allows attackers to impersonate any computer, including the domain controller itself, and execute remote procedure calls on their behalf.

Secura named this discovery and vulnerability "Zerologon" because it allows an attacker who initially has zero logon privileges on a network to completely take it over. If an attacker can establish not even a foothold —a toehold— on any machine within a Windows-based enterprise network, there is no longer any containment. Malware finds some way, any way, to run on any workstation of any large Windows domain based enterprise and it can obtain free reign over the entire network. It's able to trivially overwrite the credentials of the domain admin, resetting the password and completely taking over the enterprise.

So, what happened next? (This won't surprise anyone...)

Secura's reported details, coupled with how juicy exploitation of this would be, motivated the world's hackers, donning hats of all shades, to quickly create weaponized proof-of-concept exploits that went public within hours of Secura's report.

And the result of that is that the US Department of Homeland Security CISA, last Friday afternoon, issued Emergency Directive 20-04. https://cyber.dhs.gov/ed/20-04/

Directive 20-04 is titled: "Mitigate Netlogon Elevation of Privilege Vulnerability from August 2020 Patch Tuesday." — and it's not a request...

After citing a few relevant sections of the Homeland Security Act of 2002, which gives the DHS authority to compel action on the part of federal information system administrators, the Directive says...

On August 11, 2020, Microsoft released a software update to mitigate a critical vulnerability in Windows Server operating systems (CVE-2020-1472). The vulnerability in Microsoft Windows Netlogon Remote Protocol (MS-NRPC), a core authentication component of Active Directory, could allow an unauthenticated attacker with network access to a domain controller to completely compromise all Active Directory identity services.

Applying the update released on August 11 to domain controllers is currently the only mitigation to this vulnerability (aside from removing affected domain controllers from the network).

CISA has determined that this vulnerability poses an unacceptable risk to the Federal Civilian Executive Branch and requires an immediate and emergency action. This determination is based on the following:

- the availability of the exploit code in the wild increasing likelihood of any upatched domain controller being exploited;
- the widespread presence of the affected domain controllers across the federal enterprise;
- the high potential for a compromise of agency information systems;
- the grave impact of a successful compromise; and
- the continued presence of the vulnerability more than 30 days since the update was released.

CISA requires that agencies immediately apply the Windows Server August 2020 security update to all domain controllers.

Required Actions

This emergency directive requires the following actions:

- Update all Windows Servers with the domain controller role by 11:59 PM EDT, Monday, September 21, 2020.

In other words, last midnight, Eastern time.

In their whitepaper, Secura wrote: “This attack has a huge impact. It basically allows any attacker on the local network (such as a malicious insider or someone who simply plugged in a device to an on-premise network port) to completely compromise the Windows domain. The attack is completely unauthenticated: the attacker does not need any user credentials.”

Due to the severity of the problem, and trying to do the right thing, even with publishing the White Paper, Secura said they **had** developed their own exploit that works reliably, but given the risk, they weren't releasing it until they were confident Microsoft’s patch had been widely installed on vulnerable servers.

They warned, however, that it's not difficult to reverse engineer Microsoft's patch to develop an exploit. And sure enough, GitHub is now hosting at least three exploit scripts:

https://github.com/risksense/zerologon/
https://github.com/bb00/zer0dump
https://github.com/blackarrowsec/redteam-research/blob/master/CVE-2020-1472/CVE-2020-1472.py

The technical side of this is also interesting:

As long as an attacker has the ability to establish a TCP connection to any unpatched domain controller the attack can be launched. The vulnerability originates from the Windows implementation of AES-CFB8 — the use of the robust AES cipher in a Cipher Feedback mode. We've talked a lot about cipher modes. It's not enough to have a cipher that can, under the influence of a key, map any combination of 128 bits into another unique combination of 128 bits. That's necessary, but not sufficient. Under any given key, the same 128-bit input will produce the same 128-bit output. This means that the cipher will be leaking a great deal of information when it's used to encrypt a lengthy message. And if a message always begins the same way, the encrypted result will always be the same. Using a cipher algorithm in this way is known as ECB or Electronic Code Book mode.

Cryptographers have solved these various practical problems by using the cipher as the core of a larger encryption solution. The cipher algorithm is wrapped in additional functions to create various more complex encryption modes. Window's use of AES-CFB8 is one of these modes.

One of the ways to prevent the problem of the same plaintext message always producing the same encrypted message is by introducing a random "Initialization Vector." That's just a fancy term for deliberately introducing some varying material so that something is changing every time an encryption occurs, even if the message being encrypted does not change.

In this case, for the AES-CFB8 mode to provide security, these initialization vectors must be unique for each message. It turns out that Windows failed to enforce this requirement upon its AES-CFB8 communications and the Zerologon vulnerability exploits this failure by sending Netlogon messages that include zeros in various carefully chosen fields. The Secura writeup provides a deep dive into the cause of the vulnerability and a 5-step approach to exploiting it.

Fixing the problem...

It turns out that fixing the problem is a bit of a mess and is going to break some things, because there are places where non-Windows devices are deliberately not using secure RPC. That cannot be allowed to stand in the long term. So Microsoft is updating Windows in 2-stages:

The first stage is what they released last month. In addition to fixing the "Initialization Vector" bypass, this first update also prevents Windows Active Directory Domain controllers from accepting any unsecured RPC communications from other Windows endpoints. But it continues to permit — but logs — any non-Windows devices that are authenticating themselves over unsecured RPC connections. By monitoring this log, network admins will be alerted to devices that need to be updated before next February — when this grace period comes to an end.

On Patch Tuesday, February 9th, 2021, Microsoft will be releasing a second update that will fully enforce the use of only secure RPC across the domain, unless exceptions are specifically allowed by an administrator.

Android 11 adds security features

Our Smartphone mobile OSes are acquiring new privacy and security features — both behind the scenes and also visibly — and are, as a result, gradually becoming more secure. Looking at the useful user-facing features that Android and iOS have just added, it's easy to wonder why these features were not previously present. And I think the answer is that they do further complicate the user's experience by requiring more attention and engagement. We'll talk about iOS's new features next, but, for example, iOS 14 features a tiny orange dot at the top of the screen to indicate that the application has previously been given, and currently has, permission to use the smartphone's microphone and cameras. Given the original deliberately imposed simplicity of the Apple iPhone UI, there was certainly a time when notifying the user of this would have been considered noisy and unnecessary. The point is... that's clearly not true any longer. The goal and desire to "keep it simple" has certainly not diminished. But what has changed, against the best interests of smartphone users, is the nature of today's threat landscape. It is far more active than ever, so smartphone users need all the help they can get.

To this end, we now have Android 11 which brings billions of its users, those who can upgrade and all those who purchase Android 11 devices in the future, more control over their security and privacy. It's unfortunate that such control is needed... but it's better to have it than not. So what's new in 11?

One-time Permissions

Android 11 users get a very useful feature that iOS users have enjoyed for some time: One-time permissions, allowing users to grant apps single-use access to their device's more sensitive permissions, such as location, microphone, and camera. We're seeing variations of this on different platforms. For example, when a permission might be semi-permanent, users will be reminded after a while that such-and-such an app still has such-and-such a permission, and asked whether it should retain it.

And to that end, 11 will now autonomously revoke permissions for long-unused apps. This might be a bit of an annoyance, though those permissions can always be re-granted. This is part of working to keep the casual user more safe while attempting to minimize the impact of enforcing that safety.

More incremental updates

In a significant and welcome change, Google has increased their Play Store app's integration on the device by allowing Android 11 devices to directly download and install critical OS security patches as modules instantly—just like an app—from Google's servers. This should allow Android 11 users to receive security and bug patches as soon as they're available instead of relying on the typically long — if ever — lead time of the loop through the original device manufacturers to release OS updates. Boy, is this a welcome improvement!

Tightening up location awareness

Android 11 separates an app's rights to obtain location information when it's the foreground add versus when it has been moved into the background. When an app requests permission to access your location, Android 11 initially only grants foreground location permission. If the app additionally desires access to location information while it's in the background, such apps are now required to produce a separate and independent permission request. And, granting that permission is not as simple as quickly clicking "yeah yeah, ok, whatever." No. Now, to enable background location access, users must go to a settings page and set the “Allow all the time” option for the app's location permissions.

And to even obtain such a permission setting, Google requires their app developers to explain why their app needs background location access, at all. Just wanting it because, why not? it would be neat to have will no longer fly.

It's clear that all of these changes will improve user security — especially “From-The-Store” OS updates. Yay for that! — But there is also now more for the user to do. And there's just no getting around that. We need to tighten up the security of our Smartphones.

iOS 14

The list of iOS updates and improvements is overwhelming. And as an iOS user, myself, I'm beginning to feel more and more as though I barely know how to use the device in my pocket. It can clearly do so much more than I ask of it. Hopefully most of those new features will be discoverable over time. But, on the privacy and security front, a few of the many changes stood out.

Restricted access to the LAN:

Apple noticed that while we're at home, very few of our apps have any need to access any devices on our home network. All they need is to see our border router and the wider Internet. Of course, in this era of IoT local network access is no longer entirely unneeded, but it is typically limited to one or two or a small number of explicit IoT interfacing apps. Certainly, the vast majority of the phone's apps are entirely ignorant of what's going on, on our home or corporate LAN, and the wisdom of always and only granting “minimal rights” teaches that by default, no apps should have local LAN access, and that only those apps which actually require it should have that granted to them.

iOS 14 adds this default restriction, which I think is brilliant. It's a bit like having automatic guess WiFi access offered by the WiFi access point, but having that enforced by the device's OS rather than the access point. Of course, it's not the same, but — belt and suspenders. It clearly offers a huge improvement in security for the OS not to allow random apps to see the LAN when they have no legitimate need to do so. And local privilege can be granted when it's needed. Bravo, Apple.

Thwarting MAC-tracking:

With iOS 14, Apple has improved WiFi MAC address randomizing. We talked about this 6 years ago, back in 2014, when iOS 8 added "unassociated MAC address randomization." The idea there was that so long as a wandering iOS device was not associated with an access point, it would be broadcasting a random changing MAC address. This went a long way toward limiting casual tracking by MAC.

But it didn't go all the way, because as soon as the iOS device began to actually associate with a WiFi access point it would revert to its actual Ethernet MAC address. Presumably, this was done so that WiFi access points which granted permission and access privileges based upon MAC address would continue to work. On the other hand, a device's MAC address cannot be a secret since its entire purpose is to be the endpoint's Ethernet address on the outside of the Ethernet packet and thus cannot be obscured.

In any event, iOS 14 now, by default, offers a new “Private Address” feature that gives every iDevice a unique per-access point MAC address to fully thwart the cross-network tracking that was still possible under the previous semi-random MAC addressing scheme. This can be disabled for a specific WiFi network is it causes problems, for example in a corporate setting. But overall, another Bravo for Apple.

Fuzzy location access:

With GPS in our phones, the phone itself knows precisely where it is, and even which direction it's facing, at all times. So it's possible for apps to be granted that awareness as well. But, as with LAN access, not all apps have the actual need for that level of location precision. Exact location can compromise privacy. So, again, applying the principle of least information, iOS 14 is able to provide "fuzzy location" information to apps that have no legitimate need to know exactly where you're standing this instant. Whereas a driving navigation app does need to know precisely where you are, a weather app only needs to know your approximate location within several miles to do its job.

Access to the infamous clipboard:

The invention of a shared system-wide copy and paste clipboard was one of the many brilliant innovations to arise from the user-interface work of Xerox PARC's Alto system. But, with great power comes great responsibility. And the irresponsible (or malicious) capture of our instantaneous clipboard content has been a longstanding security problem.

Believe it or not, in the great quest to track us, there are ad networks used in apps that continually spy on our clipboard as yet another means of locking onto and following us wherever we go.

iOS 14 adds a notification every time an app accesses the clipboard. And because some apps do have a legitimate need, at the developer level there are new APIs to allow apps to ask iOS what sort of contents the clipboard contains, without needing to obtains it and thus trigger a notification. This can dramatically reduce the visual noise created by irresponsible and possibly malicious apps.

Safari Password Monitoring:

As is in vogue these days, Safari has gained the ability to check whether any of the passwords it has saved for its users have appeared in any known password breach dumps. It does this safely without revealing such password collisions to anyone but its user, who might then wish to change their publicly disclosed password.

Safari also now includes a website privacy report that allows its user to tap a Privacy Report button to view all of the cross-site trackers that are being blocked by Safari's Intelligent Tracking Prevention system.

So, overall, from both Android 11 and iOS 14, a bunch of welcome improvements both behind the scenes and requiring a bit of additional attention from their users.

Errata

A bit of errata relating to TRIM commands and hard disc drives. Thanks to some feedback from some knowledgeable folks hanging out in GRC's Security Now newsgroup, I've learned that although it's clearly wrong for Windows to be issuing TRIM commands to hard disk drives that do not accept them and thus are logging errors, there are drives which employ a storage density-increasing technology known as Shingled Magnetic Recording (SMR). Employing Shingled Magnetic Recording can increase the storage density of a shingled drive by around 25%. But the technology it tricky because, just as with an SSD where a partial update is time consuming and expensive, these SMR hard drives can also benefit from being affirmatively notified when regions of their storage no longer contain active file system data and therefore do not need to be rewritten as part of a nearby update.

So... it's not as entirely nuts for Windows 10 to be issuing TRIM commands to spinning drives (or “spinners” as we have taken to calling them over in the SpinRite.dev newsgroup), but Windows should not be doing so indiscriminately, as it reportedly is.

Closing The Loop

Someone whose Twitter handle is “mehtechie” @mehtechie tweeted:
Just wanted to say thank you so much for InitDisk, just rescued a previously thought completely dead 2TB Crucial SSD.

SpinRite

GRC's new forum system is up and running and SQRL is working perfectly there. All of those final edge-cases I referred to last week appear to have been resolved and eliminated. So I'm pulling the final “ReadSpeed” benchmark together. We will shortly be doing a pre-release round of testing, then I'll have something for the wider world to play with, as well as a readily accessible place for “ReadSpeed”s users to hangout and interact.

Formal Verification

Formal verification of security protocols. That's clearly where we're headed. At this point the technology is still emerging and maturing. And we still see too many complex protocols being developed in an ad hoc manner. Or, what might be characterized as: “Yeah, that looks right. What do you think Joe? Does that look good to you, too?” What we see over and over and over is that seat-of-the-pants design of complex protocols is leaving too many yawning gaps through which bad guys can squeeze themselves.

So let's first step back and take a look at a new vulnerability, and then at how it was found using the still-emerging techniques of formal protocol verification.

Once again we have alarming headlines:

“Billions of devices vulnerable to new 'BLESA' Bluetooth security flaw”
“Bluetooth Spoofing Bug Affects Billions of IoT Devices”

Yes, here we are again, week after week.

https://friends.cs.purdue.edu/pubs/WOOT20.pdf

The research into this was conducted by a bunch of guys at Purdue University and is titled: “BLESA: Spoofing Attacks against Reconnections in Bluetooth Low Energy” — “B.L.E.S.A.” is Bluetooth Low Energy Spoofing Attacks.

The Abstract of their research explains what they have found:

The Bluetooth Low Energy (BLE) protocol ubiquitously enables energy-efficient wireless communication among resource-constrained devices. To ease its adoption, BLE requires limited or no user interaction to establish a connection between two devices. Unfortunately, this simplicity is the root cause of several security issues.

In this paper, we analyze the security of the BLE link-layer, focusing on the scenario in which two previously-connected devices reconnect. Based on a formal analysis of the reconnection procedure defined by the BLE specification, we highlight two critical security weaknesses in the specification. As a result, even a device implementing the BLE protocol correctly may be vulnerable to spoofing attacks.

To demonstrate these design weaknesses, and further study their security implications, we develop BLE Spoofing Attacks (BLESA). These attacks enable an attacker to impersonate a BLE device and to provide spoofed data to another previously-paired device. BLESA can be easily carried out against some implementations of the BLE protocol, such as the one used in Linux. Additionally, for the BLE stack implementations used by Android and iOS, we found a logic bug enabling BLESA. We reported this security issue to the affected parties (Google and Apple), and they acknowledged our findings.

They then share some important and useful background...

> Bluetooth Low Energy (BLE) is the most widely utilized low-energy communication protocol, and, by 2023, the number of BLE-capable devices is expected to reach 5 billion. The BLE protocol enables wireless, short-range communication which allows two devices to connect and exchange data. A typical usage scenario consists of a smartphone and a “gadget” device (such as a fitness tracker) that communicate over BLE. Every BLE connection involves a device acting as a client (in this example, the smartphone) and another device acting as a server (in this example, the fitness tracker). The first time two devices connect (allowing them to exchange data), they perform a specific pairing procedure, which varies depending on the type of the connected devices and their user-interfaces’ capabilities.
>
> The major factors that have helped the rapid growth of the adoption of BLE-enabled devices are its low cost and the minimal setup effort required for end users. Unfortunately, previous research has shown that these user-friendly features have a negative impact on the security of this protocol. This concern is particularly worrisome since BLE communication is often used in security-sensitive devices, such as physical safety devices (e.g., locks) or health monitoring devices (e.g., medical implants).
>
> Researchers have pointed out and studied many implementation weaknesses in the BLE protocol by manually analyzing its specification. Additionally, some previous works have also performed a formal automated analysis of the BLE specification. However, these formal approaches only focused on limited aspects of the entire protocol, such as the BLE pairing mechanism. This limitation is due to the fact that it is challenging to fully formalize and automatically analyze the BLE specification. The challenges stem from the complexity of the BLE protocol, difficulties in modeling its multi-layer design, and its multiple variants and optional features (which allow this protocol to support a wide range of devices with significantly different capabilities).
>
> In this paper, we study a previously-unexplored mechanism of the BLE protocol. Specifically, we focus on the mechanism of dealing with reconnecting two previously connected devices. This mechanism comes into play when, for instance, a server device (e.g., a fitness tracker) moves out of the BLE wireless communication range of a connected client device (e.g., a smartphone), and then at a later time, the two devices get close enough to reestablish a connection.

So what we have, now, is a new awareness of a critical exploitable flaw in the re-connection of previously connected BLE devices.

As we know, reconnections occur when Bluetooth devices move out of range of each other and then move back into range later. Normally, when reconnecting, the two BLE devices should check each other's cryptographic keys which were negotiated during the pairing process, and reconnect and continue exchanging data via BLE.

But the Purdue team found that the official BLE specification did not contain sufficiently precise and strong enough language to describe the reconnection process. As a result, two systemic issues have made their way into BLE software implementations and down the software supply-chain:

- The authentication during the device reconnection is optional instead of mandatory.
- The authentication can potentially be circumvented if the user's device fails to enforce the IoT device to authenticate the communicated data.

And these two issues leave the door open for a BLESA attack — during which a nearby attacker is able to bypass reconnection verifications to send spoofed data to a BLE device.

Despite the somewhat vague language in the specification, the problem is not present in all BLE real-world implementations. The Purdue researchers analyzed multiple software used to support BLE on various operating systems. They found that BlueZ used in Linux-based IoT devices, Fluoride (for Android), and the iOS BLE stack were all vulnerable to BLESA attacks, while the BLE stack in Windows devices was immune.

In their paper published last month they wrote: "As of June 2020, while Apple has assigned the CVE-2020-9770 to the vulnerability and fixed it, the Android BLE implementation in our tested device, a Google Pixel XL running Android 10, remains vulnerable." And as for Linux-based IoT devices, the BlueZ development team said it would deprecate the part of its code that opens devices to BLESA attacks, and, instead, use code that implements proper BLE reconnection procedures, immune to BLESA.