Security Now! #737 - 10-22-19

Biometric Mess

This week on Security Now!

This week we check in on the frenzy to turn CheckM8 into a consumer-friendly iOS jailbreak, on another instance of stealth steganography, on a number of changes to Firefox's URL display, and on the state of Microsoft's ElectionGuard open source voting system. We also look at a very serious flaw that was just found in Linux's Realtek WiFi driver and some welcome news from Yubico. We touch on a couple of miscellaneous media tidbits, then take a look at the ramifications of two recent biometric authentication failures and consider the challenges and inherent tradeoffs of biometric authentication.

Picture of the week:
Here come the CheckM8 Jailbreaks!
(See next page)


Whoa, got the drop o
'em, there he go,
checkmate
Whoa, we go
'em
Jay Rock

Security News

Checkra1n.com... *NOT* "Checkrain.com"
Following up on last week's main "CheckM8" podcast topic, a renowned jailbreak developer by the name of Luca Todesco has teased that there would soon be a consumer-friendly easy-to-use CheckM8-based jailbreak for all of the "forever-vulnerable" iOS devices.

And once it's available it will be at http://checkra1n.com.

Over the weekend the @checkra1n Twitter account handle has uploaded a photo showing two iPod Touch with the Checkra1n app installed. Another picture posted by a developer working on checkra1n shows an iPhone X running ELI Stage 3.

The checkra1n.com site currently has the wrong certificate configured for https... It's a LetsEncrypt cert for the domain: https://qwertyoruiop.com/ and that domain has a place holder page:

> qwertyoruiop - security researcher
>
> qwupz on freenode
> qwertyoruiop on irc.cracksby.kim
> @qwertyoruiopz on twitter

So if you https:// to checkra1n.com your browser will complain a lot. But you can just http:// to bring up the (currently) place holding page. The cert will presumably be fixed before the site is fleshed out.

And... DEFINITELY be careful that you don't go to "checkrain.com" since THAT site quickly sprung up to take advantage of unwitting users who are excited to break their phones out of iJail. It's a semi-malicious site offering "pay to load" junk apps... so stay away from that one!

Overall, though, and as expected, there is a frenzy of work going on behind the scenes to bring an easy-to-use "consumer friendly" jailbreak to iOS. Stay tuned for more news on that front!

Steganography finds a new host file format
As we know, steganography is the practice of hiding in plain sight. Well... kinda. A better way to put it would be adding unrelated data into an existing file in such a fashion that the original APPEARS to be unchanged, while still carrying the unrelated data.

We've talked in the past about JPG and PNG image format files having steganographic content added to them while still showing the original image. JPG is a bit trickier, since it's a deliberately highly lossy compressed format. It achieves very high compression of images by representing the original image in the frequency domain using a technique known as a discrete cosine transform. The point is, what's put in is not at all what comes back out. By comparison, PNG images, when they are not reduced to a color pallet, are inherently lossless. This makes hiding data in the least significant bits of image pixels much more straightforward.

Now, security researchers have detected malicious content being hidden by malware campaigns in the least significant bits of WAV format audio files.

Hiding in non-executable file formats makes sense, since these files, which are inherently benign -- at least from the standpoint of code execution -- are typically allowed to pass through firewalls and intrusion detection systems without interception.

Last June, Symantec researchers spotted the Russian-backed Turla threat group (aka Waterbug or Venomous Bear) which was delivering the publicly available Metasploit Meterpreter backdoor embedded within a WAV file audio track. And then more recently Cylance found that the same steganography method was employed to infect targeted devices with XMRig Monero cryptominers or Metasploit code designed to establish a reverse shell.

What's interesting about the delivery of malicious content via WAV file is that each sample of WAV data can be anything. So unless a human actually needs to listen to the file, there's no reason not to simply format the binary data with WAVE file headers and just have the rest of the file be binary. And, in at least some cases, that appears to be what was done. The security companies reported that when the files were played, some of the WAV files produced music that had no discernible quality issues or glitches. Others simply generated static (white noise)." -- and although binary data is not purely "white", it would definitely sound like static.

The Metasploit and XMRig payloads were discovered on the same machines hinting at a campaign designed to allow its operators to use their victims' devices for cryptojacking purposes, while also establishing a command and control reverse connection.

Cylance reported that the WAV file loaders used three different methods to decode and execute the malicious code:

- Loaders that employ Least Significant Bit (LSB) steganography to decode and execute a PE file.
- Loaders that employ a rand()-based decoding algorithm to decode and execute a PE file.
- Loaders that employ a rand()-based decoding algorithm to decode and execute shellcode.

Those rand() function encodings are just basically a one-time pad used to further obscure the hidden code from static observation.

The LSB steganography would retain playability of the file at the large expensive of only storing a one to a few binary bits of data per audio sample. But if playability was not needed, then all of the audio sample space could be used for code. Note that, of course, this file could never be compressed in an lossy way such as MP3 since it would never decompress identically.

As a convert communications channel these are interesting. But since the hidden data must be knowingly extracted at the receiving end, they all requires that there must already be something nefarious running on the receiving end. So it's not as if any of these stegosauruses can be launched at any unsuspecting machine with dire result.

As happened with Chrome, security display changes are coming to Firefox 70
We're currently at FF r69. Firefox 70 will be following the industry to change the way sites appear in the URL address bar.

Currently, under 69, there's a nice friendly GREEN padlock icon for HTTPS, and for sites presenting EV certificates the company's full name is shown. All of that changes in 70.

The padlock icon turns black to signify that this is the new norm. No more green.

Previously, only insecure pages with login forms would be shown with an angry red slashed black padlock. Now, in release 70, ALL pages of non-https (HTTP) sites will receive the angry red slash padlock regardless of whether or not they are requesting sensitive information from their visitors. HTTPS is the new norm.

And as with Chrome, the user will need to dig into the URL window-shade drop-down if they are curious to inspect the type of certificate being presented by the site. That's the only way in the future to see whether a site is using an EV certificate.

But when I did inspect the drop-down of the security site I was visiting at the time I was shocked by the number of unblocked trackers and 3rd-party tracking cookies that were being permitted. It turns out that I had not been keeping up with Firefox's changes. Firefox has been making some content blocking changes -- strengthening many defaults -- and in checking into what was happening I realized that I should have been changing my settings along the way. Since I imagine others may be in the same situation, I wanted to give everyone a heads-up...

Under Firefox's main menu the 2nd line item is "Content Blocking" I had earlier set it to "Custom" which was necessary at the time to enable the stronger protections. But now they are the default and "Custom" wasn't getting it done any longer. So I imagine that most of this podcast's listeners will wish to set their browsers to "Strict" and then make selective per-site exceptions where sites complain or things break under maximum security and privacy.

And speaking of Firefox...
I'm 100% sure that those who have been pushing Firefox in the direction of increased security and privacy felt well rewarded by the news that Germany's cyber-security agency is now recommending Firefox as the most secure browser.

The German Federal Office for Information Security (or BSI, which stands for the Bundesamt für Sicherheit in der Informationstechnik) tested Firefox, Chrome, IE, and Edge. Of those four, Firefox was only browser to pass all minimum requirements for mandatory security features, and thus Firefox received top marks during a recent audit performed by Germany's cyber-security agency.

The BSI tested Mozilla Firefox 68 (ESR), Google Chrome 76, Microsoft Internet Explorer 11, and Microsoft Edge 44. The tests did not include other browsers like Safari, Brave, Opera, or Vivaldi. (Which I do think is unfortunate, since several of them are even more security- and privacy-centric than Firefox.)

The audit was carried out using rules detailed in a guideline for "modern secure browsers" that the BSI published last month, in September 2019. The BSI uses this guide to advise government agencies and companies from the private sector on what browsers are safe to use. The first edition of the guide was published in 2017, then it was reviewed and updated over the summer. As a consequence the BSI updated its guide to incorporate an awareness of the improved security measures which have recently been added to modern browsers, including HSTS (Http Strict Transport Security), SRI (sub-resource integrity), CSP 2.0 (content security policy), telemetry handling, and improved certificate handling mechanisms.

According to the BSI's new guide, to be considered "secure," a modern browser must satisfy these minimum requirements: (Here comes a 21-point list, but I think it's important and interesting...) Afterward we'll look at where the other browsers came up short:

1. Must support TLS

2. Must have a list of trusted certificates- Must support extended validation (EV) certificates

3. Must verify loaded certificates against a Certification Revocation List (CRL) or an Online Certificate Status Protocol (OCSP)

4. The browser must use icons or color highlights to show when communications to a remote server is encrypted or in plaintext- Connections to remote websites running on expired certificates must be allowed only after specific user approval

5. Must support HTTP Strict Transport Security (HSTS) (RFC 6797)

6. Must support Same Origin Policy (SOP)- Must support Content Security Policy (CSP) 2.0

7. Must support Sub-resource integrity (SRI)

8. Must support automatic updates- Must support a separate update mechanism for crucial browser components and extensions

9. Browser updates must be signed and verifiable

10. Browser's password manager must store passwords in an encrypted form- Access to the browser's built-in password vault must be allowed only after the user has entered a master password

11. User must be able to delete passwords from the browser's password manager

12. Users must be able to block or delete cookie files- Users must be able to block or delete autocomplete history

13. Users must be able to block or delete browsing history

14. Organization admins must be able to configure or block browsers from sending telemetry/usage data- Browsers must support a mechanism to check for harmful

content/URLs

15. Browsers should let organizations run locally-stored URL blacklists

16. Must support a settings section where users can enable/disable plugins, extensions, or JavaScript- Browsers must be able to import centrally-created configuration settings, ideal for wide-scale enterprise deployments

17. Must allow admins to disable cloud-based profile synchronization features

18. Must run after its initialization with minimal rights in the operating system- Must support sandboxing. All browser components must be isolated from each other and the operating system. Communication between the isolated components may only take place via defined interfaces. Direct access to resources of isolated components must not be possible.

19. Web pages need to be isolated from each other, ideally in the form of stand-alone processes. Thread-level isolation is also allowed.

20. Browsers must be coded using programming languages that support stack and heap memory protections- Browser vendor must provide security updates no longer than 21 days after the public disclosure of a security flaw. If the primary browser vendor fails to provide a security update, organizations must move to a new browser.

21. Browsers must use OS memory protections like Address Space Layout Randomization (ASLR) or Data Execution Prevention (DEP).- Organization administrators must be able to regulate or block the installation of unsanctioned add-ons/extensions.

According to the BSI, Firefox is the only browser to support all 21 requirements. There were at least eight areas where the various other three browsers failed. To simplify things I'll first note that IE failed ALL of these eight requirements.

1. Neither Chrome, Edge or IE offer support for a master password.
2. IE has no built-in update mechanism.
3. Neither Chrome, Edge or IE offer an option to block telemetry collection
4. IE alone lacks support for SOP (Same Origin Policy), CSP (Content Security Policy) and SRI (Subresource Integrity) support
5. Neither IE nor Edge offer support for browser profiles and different configurations.
6. And Chrome, Edge and IE all lack a provision for "organizational transparency"

Again… since this is not a comprehensive list of all available browsers it's a bit skewed. But these Germans do seem to be a bit finicky with some of their less mainstream security and privacy features. Their requirements list has a decidedly managerial oversight feel, so it might well be that other well-regarded browsers such as Brave, Opera and Vivaldi might have also come up short, though not for reasons that would bother their users.

One more little bit of Firefox housekeeping
Mozilla has wisely realized that its own internal HTML/JavaScript browser pages (like the often-mentioned about:config) might somehow become victim of exploitation. So they've taken the wise precaution of preemptively blocking both inline and eval JavaScript on those pages to prevent any possibility of successful script injection attacks. This will mitigate a large class of potential cross-site scripting issues.

Firefox has 45 internal locally-hosted about: pages, including...

- about:config — panel to modify Firefox preferences and critical settings.
- about:downloads — your recent downloads done within Firefox.
- about:memory — shows the memory usage of Firefox.
- about:newtab — the default new tab page.
- about:plugins — lists all your plugins as well as other useful information.
- about:privatebrowsing — open a new private window.
- about:networking — displays networking information.

Since all of these pages are written in HTML/JavaScript and render within the powerful security context of the browser itself, they are prone to code injection attacks that, in case of a vulnerability, could allow remote attackers to inject and execute arbitrary code on behalf of the user, i.e., cross-site scripting (XSS) attacks.

To add a robust first line of defense against code injection attacks, even when there is a vulnerability, Mozilla has blocked the execution of all inline scripts, thus injected scripts as well, by implementing a strict Content Security Policies (CSP) to ensure the JavaScript code only executes when loaded from a packaged resource using the internal protocol.

To achieve this, Mozilla had to rewrite all inline event handlers and move all inline JavaScript code out-of-line into separate packaged files for all 45 about: pages.

In a blog posting last week Mozilla wrote: "Not allowing any inline script in any of the about: pages limits the attack surface of arbitrary code execution and hence provides a strong first line of defense against code injection attacks."

When attackers can't inject script directly, they use the JavaScript function eval() and similar methods to trick the target applications into converting text into an executable JavaScript to achieve code injection.

So, in addition to inline scripts, Mozilla has also removed and blocked eval-like functions, which Mozilla feels is another "dangerous tool," as it parses and executes an arbitrary string in the same security context as itself.

They wrote: "If you run eval() with a string that could be affected by a malicious party, you may end up running malicious code on the user's machine with the permissions of your webpage/extension."

Google shares this feeling, writing: "eval is dangerous inside an extension because the code it executes has access to everything in the extension's high-permission environment."

So Mozilla rewrote all use of eval-like functions from system privileged contexts and the parent process in the codebase of its Firefox web browser. And on top of this, Mozilla also added eval() assertions that will disallow the use of eval() function and its relatives in system-privileged script contexts, and inform the Mozilla Security Team of yet unknown instances of eval().

What all of this means for we Firefox users is that Mozilla is being as proactive about the security of their users as it is sadly necessary for them to be in this day and age.

More on Microsoft's open source "ElectionGuard" election security system
We touched on this previously back in May. Now we know a lot more...

The good news is we're beginning to see some movement away from the closed, failed, and proprietary "election systems for profit" model. In a major move to help this effort, last May 6th during their Build developer conference Microsoft announced their free, open-source software development kit (SDK) called ElectionGuard that aims to enable end-to-end verification of voting.

The goods are now up on Github: https://github.com/microsoft/ElectionGuard-SDK

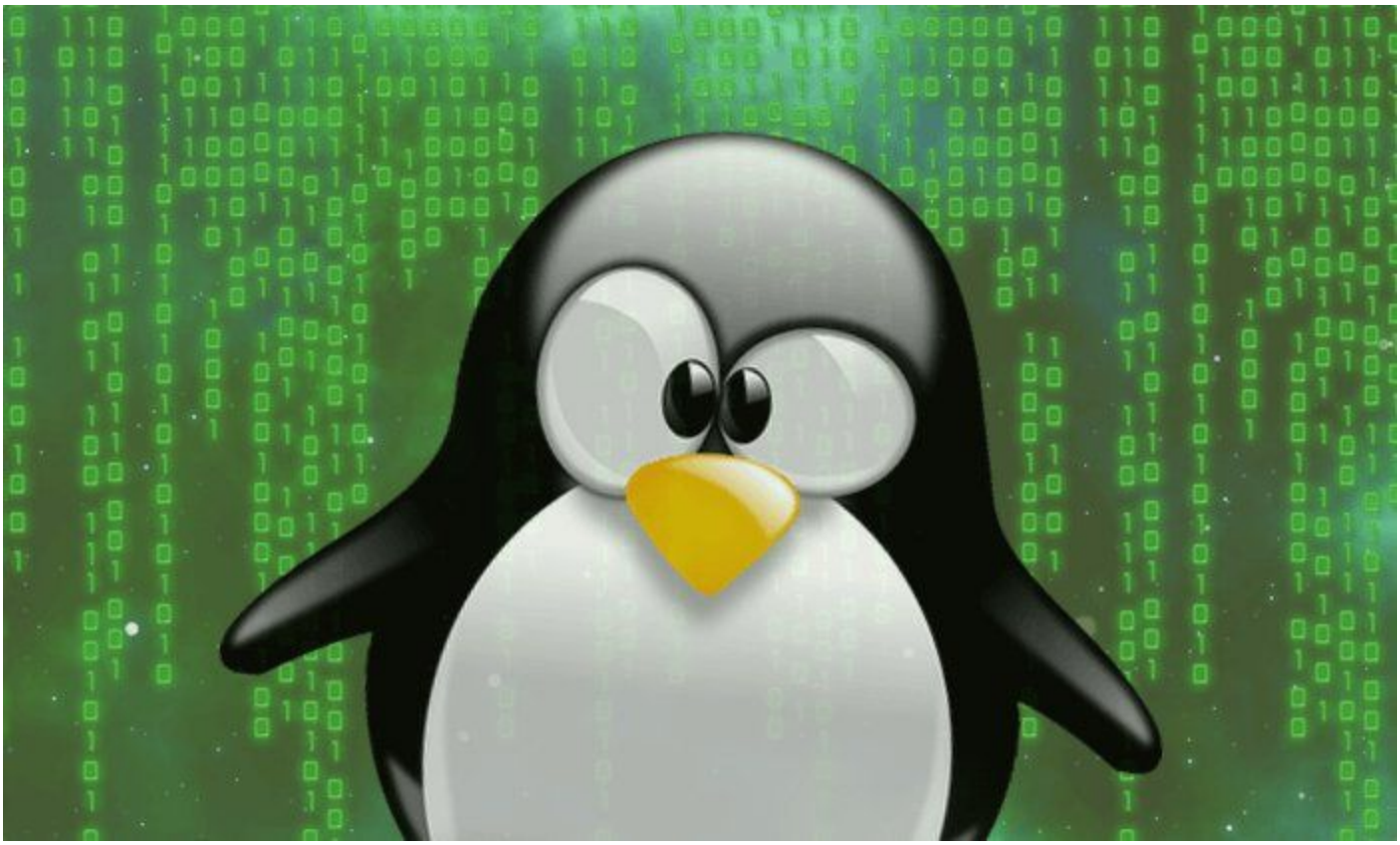
Microsoft's ElectionGuard SDK can be integrated into voting systems and has been designed to "enable end-to-end verification of elections, open results to third-party organizations for secure validation, and allow individual voters to confirm their votes were correctly counted."

ElectionGuard is back in the news now because Microsoft has followed through by wrapping their offering in a bug bounty program. They are inviting security researchers from across the world to help with the discovery of high impact vulnerabilities in the ElectionGuard SDK.

https://msrc-blog.microsoft.com/2019/10/18/introducing-the-electionguard-bounty-program/

In their blog posting announcing the bounty, they wrote... "The ElectionGuard Bounty program invites security researchers to partner with Microsoft to secure ElectionGuard users, and is a part of Microsoft's broader commitment to preserving and protecting electoral processes under the Defending Democracy Program. Researchers from across the globe, whether full-time cybersecurity professionals, part-time hobbyists, or students, are invited to discover high impact vulnerabilities in targeted areas of the ElectionGuard SDK and share them with Microsoft under Coordinated Vulnerability Disclosure (CVD)."

ElectionGuard Bounty offers cybersecurity researchers a reward of up to $15,000 for eligible submissions with a clear and concise proof of concept (POC) to demonstrate how the discovered vulnerability could be exploited to achieve an in-scope security impact. The ElectionGuard components that are currently in scope for bug bounty awards include ElectionGuard API SDK, ElectionGuard specification and documentation, and verifier reference implementation. Microsoft indicated that it will update the ElectionGuard bounty scope with additional components to award further research in the future.



A potentially serious flaw found in Realtek WiFi drivers.
This definitely affects Linux machines with Realtek WiFi enabled. And early indications are that any Realtek-based Android devices are likely also affected.

- As we know, kernel driver flaws are always worrisome because they can be extremely potent. And flaws are even more problematical when they affect Wireless interfaces.
- And they are more problematical when they affect systems in their default configuration.
- And they are still MORE problematical when they require no user-interaction to be exploited.
- And they are even MORE problematical when the flaw is a buffer overrun of remote attacker-provided data.

This bug, being tracked as CVE-2019-17666, has ALL of those properties. It was discovered by Nico Waisman, the principal security engineer at Github while he was examining the handling of "Notice of Absence" protocol packets. A patch to correct this is currently under revision, but has not yet been incorporated into the Linux kernel. If successfully weaponized -- and it's important to note that hasn't been done yet, so far as is publicly known -- it would allow attackers to to fully compromise vulnerable machines

The flaw is classified as CRITICAL in severity, exists in the "rtlwifi" driver. The driver has been found to be vulnerable to a buffer overflow attack. Systems lacking WiFi, or with their WiFi disabled, or using some other non-Realtek WiFi chip will be safe. Otherwise... Not so much.

The driver flaw can be triggered when an affected device is within radio range of a malicious device. As long as the Wi-Fi is turned on, it requires no interaction on the part of the end user. The malicious device exploits the vulnerability by using a power-saving feature known as a

Notice of Absence that's built into Wi-Fi Direct. Wi-Fi Direct is the peer-to-peer standard that allows two devices to connect over Wi-Fi without the need of any common access point. The attack would work by adding vendor-specific information elements to Wi-Fi beacons that, when received by a vulnerable device, trigger the buffer overflow in the Linux kernel.

Once again... the vulnerability only affects Linux devices that use a Realtek chip when Wi-Fi is turned on. The flaw cannot be triggered if Wi-Fi is turned off or if the device uses a Wi-Fi chip from a different manufacturer. And the reporting this indicate that Android devices with Realtek Wi-Fi chips may also be affected.

So far, all we've seen disclosed publicly is a denial-of-service which crashes the targeted device. So we don't yet know that anything more is possible. But hackers are hard at work right now trying to weaponize this vulnerability. This flaw was introduced into the Linux codebase six years ago, back in 2013. So imagine if an attack can be developed for mobile Android devices whose kernels will never be updated. Not good. With a potential payoff like that it's not difficult to understand why efforts are underway to leverage this into a remote code execution attack.

The Linux Gurus are on the case. Laura Abbott posted to the Linux Kernel Mailing List:

Nicolas Waisman noticed that even though noa_len is checked for a compatible length it's still possible to overrun the buffers of p2pinfo since there's no check on the upper bound of noa_num. Bounds check noa_num against P2P_MAX_NOA_NUM.

So security-conscious Linux users with Realtek WiFi will want to be on the lookout for an Realtek driver update which should be coming very soon, since this is not a mysterious or difficult-to-patch fix. And REALLY security conscious Linux users who are using devices with enabled Realtek WiFi chips might wish to consider turning off WiFi until their drivers are patched. I'm not preaching tinfoil here. I recognize that the likelihood of any one individual being attacked is vanishingly small, but it did recently jump up from zero… is all I'm saying.

Yubikey for local Windows login has been officially released

Yubico's solution for local hardware dongle protected login to Windows machines (Win7 through 10) has been available in preview form since last March, and I recently took a look at it for securing my laptop before traveling out of the country. It's very clean and simple.

The feature has been available for Mac and Linux machines for some time, but Yubico wanted to get it exactly right before declaring it ready for Windows. It is now.

Note that to be useful, the drive would also need to be locked down with Bitlocker to that its drive cannot simply be move to another machine and read. But with Bitlocker in place and strong local machine login protection provided by strong multifactor authentication, a roaming laptop is going to be about as secure as anyone could wish.

https://www.yubico.com/products/services-software/download/computer-login-tools/

Random Media Heads-Up:

Being a huge, long-standing fan of the Jason Bourne movies, I was very curious and hopeful to see what the USA Network had in store for us with last Tuesday's release of the first episode of their new "Treadstone" fiction series which is set in time after the Jason Bourne adventures. This first season will have 10 episodes, each releasing on a Tuesday. The first one was last Tuesday and I think I'm going to watch that first one again, since it jumped around a lot and treated me as though I was smarter than I apparently am. In other words, it left me feeling a bit confused. So the jury's still out on it, but I wanted to bring it to the attention of any other Jason Bourne fans, since it looks like it may develop into something worthwhile.

Sci-Fi

And speaking of worthwhile... the long-awaited and very much anticipated second book in Peter Hamilton's newest "Salvation" trilogy releases one week from today, on October 29th. The timing is perfect for me, since I just wrapped up a very enjoyable 5-book series "The Terran Fleet Command Saga" by Tori Harris. It's not Hamilton grade, but then, in my opinion, nothing else is. But I would recommend "The Terran Fleet Command Saga" to anyone who enjoys a nice, well-written, slow burn, well-assembled space opera combat mystery involving some enigmatic aliens. It had many really great moments.

And, of course, all of this is set against the background of Ryk Brown's ambitious and ongoing 75-book "Frontiers Saga" series. After finishing "The Terran Fleet Command Saga" I quickly read book 12 of the second set of 15, which had recently dropped, to see how Nathan, Cameron, Telles, Jessica and all of the rest were coming along.

Each of these authors has very different flavors and styles. And if I was asked I would be unable to choose among them. I have never encountered any other author with Peter Hamilton's storytelling and reality creation talent. But for the Sci-Fi content-starved, Ryk Brown's Frontiers Saga is just so much fun!

Biometric Mess

Two failures of biometric authentication were in the news this week. So I thought that this would be an opportune time for us to check-in with biometrics to see how that's all going.

Sophos' story about the newly released Google Pixel 4 face unlock begins with the rhetorical question: *"Does it matter that Google's Pixel 4 'Face Unlock' works even if the owner has their eyes closed?"* … To which most of us, after thinking about it for a second, would answer **Yes!**

The Pixel 4 follows in the footsteps of Apple's FaceID technology, and similarly drops fingerprint recognition in its favor. But Chris Fox, a reporter for the BBC, discovered a potential issue – Face Unlock works when the user has his or her eyes closed, for example, when they're asleep.

And it's not necessary for Google to confirm this since it's already on the Pixel 4's help pages:

> *Your phone can also be unlocked by someone else if it's held up to your face, even if your eyes are closed. Keep your phone in a safe place, like your front pocket or handbag.*

So... the risk here is that someone might get hold of a device, for example someone's children, their spouse or partner, and unlock it simply by holding the screen to the face of the phone's sleeping (or otherwise unconscious or eyes-closed) owner.

And this appears to be unintended behavior since, according to the BBC and The Verge, images of the Pixel 4 which leaked before it launched, included a "require eyes to be open" setting in the setup menu. (See the photo on the previous page.) But that option has since been removed from all Pixel 4 devices sent out for review.

Since this is presumably an important security feature for any facial recognition system (Apple's FaceID offers the option to turn it off, but warns not to), my own guess would be that it wasn't working reliably enough and Google chose to simply remove the option. That's only speculation. What we know is that this important security feature will be missing from the devices when they begin shipping in two days, on October 24th.

Google has since stated that it plans to fix the issue <quote> "within months", but they've not been more specific. In the meantime, for anyone who is concerned about this, Google recommends using a PIN code or an unlock pattern.

So that's the first of the two recent biometric screw-ups...

Meanwhile... It turns out that the sexy ultrasonic fingerprint reader used by Samsung's flagship S10 and Note10 smartphones can be spoofed with a $3 screen protector… at least one that's not made by Samsung.

This recently came to light and made the news when a British woman claimed, and others including Samsung have since confirmed, that after fitting her new phone with a screen protector she was able to unlock her S10 using any one of her fingerprints, including ones not enrolled in the phone's authentication system.

And since that made her curious, she reportedly then asked her husband to try the same thing, and his thumbprints worked too, as did the same trick on her sister's Samsung. So obviously, something was wrong.

Samsung's initial response was:

> We're investigating this internally. We recommend all customers to use Samsung-authorised accessories, specifically designed for Samsung products.

Then, last week in comments to Reuters, Samsung admitted the problem was real and said it would release a software patch:

> We are investigating this issue and will be deploying a software patch soon. We encourage any customers with questions or who need support downloading the latest software to contact us directly.

The issue of the S10 and screen protectors was first noticed when the smartphone was launched in February 2019… but the issue failed to acquire critical mass.

Unlike earlier designs which used a dedicated sensor, the Qualcomm ultrasonic technology used by Samsung is embedded behind the screen and uses high-frequency sound waves which are modulated by the pressure of a user's finger against the screen glass.

It was noticed, however, that covering the screen with a protector could, in some instances, create a gap that could interfere with the integrity of the system's ultrasonics. Samsung's advice, which still seems a bit flaky, is to use its branded screen protectors that use special adhesives that remove the possibility of any gap. This whole technology seems sketchy at best.

The security industry, now finally attentive to this problem is recommending that any S10 or Note10 users disable fingerprint authentication and fall back to a PIN until the promised patch becomes available and is applied. At this time it's unclear whether that will arrive as an out-of-band patch or might be part of November's Android security update.

And Sophos noted that this is not the first time the S10's fingerprint reader has been in the spotlight. In April they reported on an anonymous researcher who appeared to show themselves unlocking a Samsung S10 using a 3D printed-fingerprint. And last April, Naked Security reported that the Nokia 9 PureView's fingerprint reader was fooled by a chewing gum packet.

So… biometrics: Convenient? Absolutely. As secure as we might hope and wish? That's much less clear.